User Guide

AWS CloudFormation



API Version 2010-05-15

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CloudFormation: User Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is CloudFormation? 1
Simplify infrastructure management 1
Quickly replicate your infrastructure 1
Easily control and track changes to your infrastructure 2
Getting started with CloudFormation 2
Related information 2
Getting started 3
How CloudFormation works 3
Key concepts 3
How CloudFormation works 7
Ways to get started with CloudFormation 11
Signing up for an AWS account 11
Sign up for an AWS account 11
Creating your first stack 11
Prerequisites 12
Create a CloudFormation stack with the console 12
Monitor stack creation 16
Test the web server 17
Troubleshooting 18
Clean up 19
Next steps 20
Best practices 21
Shorten the feedback loop to improve development velocity 22
Organize your stacks by lifecycle and ownership 23
Use cross-stack references to share values between stacks 24
Use CloudFormation StackSets for multi-account and multi-region deployments 24
Verify quotas for all resource types 25
Reuse templates to replicate stacks in multiple environments 25
Use modules to reuse resource configurations 25
Adopt infrastructure as code practices 26
Don't embed credentials in your templates 26
Use AWS-specific parameter types 27
Use parameter constraints 27
Use pseudo parameters to promote portability 27

Use `AWS::CloudFormation::Init` to deploy software applications on Amazon EC2 instances ... 28
Use the latest helper scripts ... 28
Validate templates before using them ... 29
Validate templates for organization policy compliance ... 29
Using YAML or JSON for template authoring ... 30
Implement a comprehensive tagging strategy ... 30
Leverage template macros for advanced transformations ... 31
Manage all stack resources through CloudFormation ... 31
Create change sets before updating your stacks ... 31
Use stack policies to protect resources ... 32
Use AWS CloudTrail to log CloudFormation calls ... 32
Use code reviews and revision controls to manage your templates ... 32
Update your Amazon EC2 instances regularly ... 33
Use drift detection regularly ... 33
Configure rollback triggers for automatic recovery ... 33
Implement effective stack refactoring strategies ... 34
Use CloudFormation Hooks for lifecycle management ... 34
Use IaC Generator to create templates from existing resources ... 34
Use template studio to edit templates visually ... 35
Consider using AWS Cloud Development Kit (AWS CDK) for complex infrastructure ... 35
Use IAM to control access ... 36
Apply the principle of least privilege ... 36
Secure sensitive parameters ... 36
Implement policy as code with Guard ... 37
Working with templates ... 38
Where templates get stored ... 39
Validating templates ... 39
Use with AI coding agents ... 40
Understand validation scope ... 40
Getting started with templates ... 2
Plan to use the CloudFormation template reference ... 41
Sample templates ... 41
Template format ... 41
Template structure ... 42
Comments ... 44

Specifications ... 45
Learn more ... 45
Using regular expressions ... 45
Template sections ... 47
Resources ... 48
Parameters ... 56
Outputs ... 70
Mappings ... 76
Metadata ... 83
Rules ... 89
Conditions ... 109
Transform ... 121
Format version ... 122
Description ... 123
CloudFormation Language Server ... 123
IDEs integrating with the CloudFormation Language Server ... 124
Getting started ... 124
Initializing a CloudFormation project in the IDE (VS Code only) ... 131
Need help? ... 132
CloudFormation Linter ... 132
Install `cfn-lint` ... 133
Run `cfn-lint` ... 133
Understand the results ... 134
Configure checks ... 135
Add custom rules ... 136
Add `cfn-lint` to your workflow ... 137
CloudFormation Validate ... 138
How validation works ... 139
Choose a validation method ... 139
Install the command-line tool ... 139
Run validation ... 140
Understand the results ... 141
Filter the results ... 141
Add custom rules ... 142
Validate new resource types and properties ... 143
Embed the validation library ... 143

Use with AWS CDK ... 145
CloudFormation Guard ... 145
Install Guard ... 146
Understand the Guard language ... 146
Write a rule ... 147
Validate a template ... 148
Test your rules ... 149
Generate rules from a template ... 150
Add Guard to your workflow ... 150
Template studio ... 151
Considerations ... 151
Permissions required for template studio ... 152
Open template studio ... 153
Work with the template and the diagram ... 154
Convert between YAML and JSON ... 154
Syntax problems as you type ... 155
Validate a template ... 155
Save a template to Amazon S3 ... 156
Copy or download a template or diagram ... 157
IaC generator ... 157
Considerations ... 158
Permissions ... 159
Commonly used commands ... 159
Migrate a template to the AWS CDK ... 160
Start a resource scan ... 160
View the scan summary ... 165
Create a template from scanned resources ... 166
Create a stack from scanned resources ... 171
Resolve write-only properties ... 172
Get values stored in other services ... 177
General considerations ... 178
Get Systems Manager plaintext value ... 179
Get Systems Manager secure string ... 182
Get Secrets Manager value ... 186
Get AWS values ... 190
Syntax ... 190

Available pseudo parameters ... 191
Examples ... 193
Get stack outputs ... 197
Exporting stack output values versus using nested stacks ... 197
Considerations ... 198
Listing exported output values ... 198
Listing stacks that import an exported output value ... 199
Referencing outputs without exports ... 200
Specify existing resources at runtime ... 201
Overview ... 202
Example ... 203
Considerations ... 206
Supported AWS-specific parameter types ... 207
Supported Systems Manager parameter types ... 209
Unsupported Systems Manager parameter types ... 209
Walkthroughs ... 210
Refer to resource outputs in another CloudFormation stack ... 211
Deploy applications on Amazon EC2 ... 215
Update a stack ... 225
Create a scaled and load-balanced application ... 242
Peer with a VPC in another account ... 255
Perform ECS blue/green deployments ... 266
Template snippets ... 290
General ... 291
Auto scaling ... 301
AWS Billing Console ... 333
CloudFormation ... 337
CloudFront ... 344
CloudWatch ... 356
CloudWatch Logs ... 362
DynamoDB ... 396
Amazon EC2 ... 400
Amazon ECS ... 437
Amazon EFS ... 460
Elastic Beanstalk ... 485
Elastic Load Balancing ... 490

IAM ... 495
AWS Lambda ... 513
Amazon Redshift ... 519
Amazon RDS ... 529
Route 53 ... 538
Amazon S3 ... 546
Amazon SNS ... 555
Amazon SQS ... 555
Amazon Timestream ... 556
Windows-based stacks ... 585
Bootstrapping Windows stacks ... 586
Use CloudFormation-supplied resource types ... 593
Custom resources ... 595
Template macros ... 640
Nested stacks ... 657
Wait conditions ... 664
Create reusable resource configurations with modules ... 670
Considerations when using modules ... 671
Understand module versioning ... 672
Use modules in a template ... 673
Use parameters to specify module values ... 674
Reference module resources ... 677
Creating and managing stacks ... 679
Interfaces for managing your stacks ... 681
Agent setup guide ... 681
Work with your coding agent ... 682
Prerequisites ... 682
What gets installed ... 682
Claude Code ... 683
Codex ... 683
Cursor ... 684
Kiro ... 684
GitHub Copilot ... 685
Windsurf ... 685
OpenCode ... 686
For all other agents compatible with agent skills and MCP Server configuration ... 687

What you can do with your agent ... 687
IAM permissions ... 688
Related resources ... 688
Create a stack ... 689
Creating a stack ... 689
Configure stack options ... 692
Preview the configuration of your stack ... 695
View stack information ... 696
Update your stack template ... 698
Understand update behaviors of stack resources ... 700
Update stacks using change sets ... 702
Create a change set ... 704
View a change set ... 710
Drift-aware change sets ... 718
Execute a change set ... 724
Delete a change set ... 726
Example change sets ... 728
Change sets for nested stacks ... 740
Validate stack deployments ... 747
How pre-deployment validation works ... 747
Considerations ... 748
Prerequisites ... 748
Disable pre-deployment validation ... 749
Validate a stack deployment (console) ... 750
Validate a stack deployment (AWS CLI) ... 751
Validation types ... 753
Resource limitations ... 755
Update stacks directly ... 769
Cancel a stack update ... 773
To cancel a stack update (console) ... 773
To cancel a stack update (AWS CLI) ... 773
Delete a stack ... 774
Related resources ... 776
View deleted stacks ... 776
Monitor stack progress ... 777
View stack events ... 777

View stack events by operation ... 786
View stack deployment timeline graph ... 790
Understand stack creation events ... 793
Monitor stack updates ... 795
Continue rolling back an update ... 797
Determine the cause of a stack failure ... 803
Stack failure options ... 804
Roll back your stack on alarm breach ... 814
Add rollback triggers during stack creation or updating ... 815
Add rollback triggers to a change set ... 816
View rollback triggers for a stack ... 816
View rollback triggers for a change set ... 817
Detect unmanaged configuration changes to stacks and resources ... 817
What is drift? ... 817
Drift detection status codes ... 819
Considerations when detecting drift ... 821
Detect drift on an entire CloudFormation stack ... 823
Detect drift on individual stack resources ... 829
Resolve drift with an import operation ... 833
Import AWS resources ... 843
Manually import AWS resources ... 844
Automatically import AWS resources ... 881
Reverting an import operation ... 909
Stack refactoring ... 915
How stack refactoring works ... 915
Stack refactoring considerations ... 916
Prerequisites ... 917
Refactor stacks (console) ... 917
Refactor stacks (AWS CLI) ... 918
Resource limitations ... 922
Resource type support ... 927
Use quick-create links to create CloudFormation stacks ... 1009
URL format ... 1009
Example ... 1010
Creating a stack using a quick-create link ... 1011
Express mode ... 1012

When to use express mode ... 1013
How express mode works ... 1013
How CloudFormation handles dependencies in express mode ... 1013
Getting started with express mode ... 1014
Rollback behavior with express mode ... 1016
Express mode and nested stacks ... 1016
Resource readiness considerations ... 1016
Unsupported features and limitations ... 1017
Comparison with AWS CDK hotswap deployments ... 1017
Related resources ... 1018
Understand stack deployment time ... 1018
How CloudFormation provisions a stack ... 1019
What affects deployment time ... 1019
Ways to shorten deployment time ... 1020
Related resources ... 1020
Example commands for the AWS CLI and PowerShell ... 1020
CancelUpdateStack ... 1021
ContinueUpdateRollback ... 1022
CreateStack ... 1023
Deploy ... 1029
DeleteStack ... 1029
DescribeStackEvents ... 1031
DescribeStackResource ... 1033
DescribeStackResources ... 1034
DescribeStacks ... 1038
GetTemplate ... 1041
ListStackResources ... 1043
ListStacks ... 1045
UpdateStack ... 1047
ValidateTemplate ... 1052
Upload local artifacts to an S3 bucket ... 1055
Managing stacks with StackSets ... 1058
StackSets concepts ... 1059
Administrator and target accounts ... 1060
CloudFormation StackSets ... 1060
Permission models for StackSets ... 1060

Stack instances ... 1061
StackSet operations ... 1062
StackSet operation options ... 1063
Tags ... 1065
StackSets status codes ... 1065
Stack instance status codes ... 1066
Prerequisites ... 1068
Enable AWS Regions that are disabled by default ... 1068
Grant self-managed permissions ... 1069
Activate trusted access ... 1086
Get started using a sample template ... 1091
Prerequisites ... 1092
Create a StackSet with a sample template from the console ... 1092
Monitor StackSet creation ... 1095
View StackSet results ... 1095
Update your StackSet ... 1095
Add stacks to your StackSet ... 1096
Clean up ... 1097
Next steps ... 1099
Create StackSets (self-managed permissions) ... 1099
Create a StackSet with self-managed permissions (console) ... 1100
Create a StackSet with self-managed permissions (AWS CLI) ... 1101
Create StackSets (service-managed permissions) ... 1103
Considerations ... 1103
Create a StackSet with service-managed permissions (console) ... 1104
Create a StackSet with service-managed permissions (AWS CLI) ... 1106
Enable-disable automatic deployments ... 1110
How automatic deployments work ... 1111
Considerations ... 1111
Enable or disable automatic deployments (console) ... 1112
Enable or disable automatic deployments (AWS CLI) ... 1112
Update StackSets ... 1113
Update your StackSet (console) ... 1113
Update your StackSet (AWS CLI) ... 1115
Add stacks to StackSets ... 1117
Add stacks to a StackSet (console) ... 1117

Add stacks to a StackSet (AWS CLI) ... 1119
Override parameters on stacks ... 1120
Override parameters on stacks (console) ... 1120
Override parameters on stacks (AWS CLI) ... 1122
Delete stacks from StackSets ... 1123
Delete stacks from your StackSet (console) ... 1124
Delete stacks from your StackSet (AWS CLI) ... 1126
Delete StackSets ... 1127
Delete a StackSet (console) ... 1127
Delete a StackSet (AWS CLI) ... 1128
Delete service roles (optional) ... 1128
Target account gates ... 1129
Requirements ... 1130
CloudFormation templates for creating Lambda functions ... 1131
Choose the Concurrency Mode ... 1131
How each Concurrency Mode works ... 1132
Choosing between Strict failure tolerance and Soft failure tolerance based on deployment speed ... 1134
Choosing your Concurrency Mode (console) ... 1136
Choosing your Concurrency Mode (AWS CLI) ... 1137
Detecting drift on StackSets ... 1138
How CloudFormation performs drift detection on a StackSet ... 1138
Detect drift on a StackSet (console) ... 1139
Detect drift on a StackSet (AWS CLI) ... 1140
Stopping drift detection on a StackSet ... 1147
Import stacks into StackSets ... 1148
Self-managed stack import ... 1148
Service-managed stack import ... 1152
Revert stack imports ... 1157
Best practices ... 1157
Defining the template ... 1158
Creating or adding stacks to the StackSet ... 1158
Updating stacks in a StackSet ... 1159
Sample templates ... 1159
Troubleshooting ... 1161
Common reasons for stack operation failure ... 1161

Retrying failed stack creation or update operations 1162
Stack instance deletion fails 1163
Stack import operation fails 1163
Stack instance failure count for StackSets operations 1163
Syncing stacks with Git source code 1170
How Git sync works 1171
How Git sync works 1171
Comments on pull requests 1172
Stack deployment file 1173
CloudFormation template file 1174
Template definition repository 1174
Prerequisites 1174
Git repository 1175
CloudFormation template 1175
Git sync service role 1175
IAM permissions for console users 1178
Create a stack from repository source code 1181
Create a stack from repository source code 1182
Update your stack from your Git repository 1184
Enable comments on pull requests 1185
Status dashboard 1186
Git sync status 1186
Latest sync events 1187
Supported stack states 1187
Managing extensions with the CloudFormation registry 1191
Related documentation 1191
Concepts 1192
Extension types 1192
Public extensions 1193
Activated extensions 1193
View the available and activated extensions 1194
Public extensions 1195
IAM role 1196
Automatically use new versions of extensions 1198
Use aliases to refer to extensions 1199
Commonly used AWS CLI commands for working with public extensions 1199

Activate a public extension ... 1200
Update a public extension ... 1204
Deactivate public extensions ... 1205
Private extensions ... 1206
IAM permissions ... 1207
Commonly used AWS CLI commands for working with private extensions ... 1207
Register a private extension ... 1207
Deregister private extensions ... 1210
Edit configuration data for extensions ... 1211
Permissions required to use dynamic references ... 1212
Edit configuration data for an extension (console) ... 1212
Edit configuration data for an extension (AWS CLI) ... 1213
Record resource types in AWS Config ... 1214
Preventing sensitive properties being recorded in a configuration item ... 1215
Continuous delivery ... 1216
Walkthrough: Building a pipeline for test and production stacks ... 1216
Prerequisites ... 1217
Walkthrough overview ... 1217
Step 1: Edit the artifact and upload it to an S3 Bucket ... 1218
Step 2: Create the pipeline stack ... 1220
Step 3: View the WordPress stack ... 1225
Step 4: Clean up resources ... 1226
See also ... 1227
Configuration properties reference ... 1227
Configuration properties (console) ... 1227
Configuration properties (JSON object) ... 1230
See also ... 1235
CloudFormation artifacts ... 1235
Stack template file ... 1236
Template configuration file ... 1236
See also ... 1237
Using parameter override functions with CodePipeline pipelines ... 1238
`Fn::GetArtifactAtt` ... 1238
`Fn::GetParam` ... 1240
See also ... 1242
Template Reference Guide ... 1243

Security 1244
Protect stacks from being deleted 1245
Controlling who can change termination protection on stacks 1246
Prevent updates to stack resources 1247
Example stack policy 1248
Defining a stack policy 1249
Setting a stack policy 1253
Updating protected resources 1254
Modifying a stack policy 1257
More example stack policies 1258
Data protection 1262
Encryption at rest 1263
Encryption in transit 1263
Internetwork traffic privacy 1263
Control access with IAM 1263
Defining IAM identity-based policies for CloudFormation 1264
Acknowledging IAM resources in CloudFormation templates 1273
Managing credentials for applications running on Amazon EC2 instances 1274
Granting temporary access (federated access) 1274
Identity-based policy examples 1276
AWS managed policies 1283
CloudFormation service role 1286
Cross-service confused deputy prevention 1287
FAS requests and permission evaluation 1289
Logging API calls 1291
CloudFormation information in CloudTrail 1292
Understanding CloudFormation log file entries 1293
Infrastructure security 1297
Resilience 1297
Compliance validation 1298
Configuration and vulnerability analysis 1298
Security best practices 1298
Use IAM to control access 1299
Do not embed credentials in your templates 1299
Use AWS CloudTrail to log CloudFormation calls 1300
AWS PrivateLink 1300

Considerations for CloudFormation VPC endpoints ... 1300
Creating an interface VPC endpoint for CloudFormation ... 1301
Creating a VPC endpoint policy for CloudFormation ... 1302
Monitoring with EventBridge ... 1304
CloudFormation and Git sync events overview ... 1305
Permissions ... 1306
Creating a custom event pattern ... 1307
Events detail reference ... 1308
Resource Status Change ... 1309
Stack Status Change ... 1311
Drift Detection Status Change ... 1313
StackSet Status Change ... 1316
StackSet Stack Instance Status Change ... 1317
StackSet Operation Status Change ... 1320
Repository Sync Status Change ... 1322
Resource Sync Status Change ... 1325
Quotas ... 1328
Feature availability ... 1336
StackSets and macros ... 1336
Troubleshooting ... 1337
Troubleshooting guide ... 1337
Troubleshooting errors ... 1338
Delete stack fails ... 1339
Dependency error ... 1340
AWS Config and AWS Systems Manager conflicts ... 1340
Error parsing parameter when passing a list ... 1341
Insufficient IAM permissions ... 1341
Invalid value or unsupported resource property ... 1341
Quota exceeded ... 1341
Nested stacks rollback failure ... 1342
No updates to perform ... 1342
Resource failed to stabilize during a create, update, or delete stack operation ... 1343
Security group does not exist in VPC ... 1343
Update rollback failed ... 1343
Wait condition didn't receive the required number of signals from an Amazon EC2 instance ... 1345

Resource removed from stack but not deleted 1346
Contacting support 1347
Pre-deployment validation errors 1347
Express mode completion behavior 1348
Troubleshooting with AWS assistant 1350
Features 1350
How it works 1350
Pricing 1351
Regional availability 1351
Limitations 1351
Document history 1352
Archived updates 1377

What is CloudFormation?

CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles that. The following scenarios demonstrate how CloudFormation can help.

Simplify infrastructure management

For a scalable web application that also includes a backend database, you might use an Auto Scaling group, an Elastic Load Balancing load balancer, and an Amazon Relational Database Service database instance. You might use each individual service to provision these resources and after you create the resources, you would have to configure them to work together. All these tasks can add complexity and time before you even get your application up and running.

Instead, you can create a CloudFormation template or modify an existing one. A *template* describes all your resources and their properties. When you use that template to create a CloudFormation stack, CloudFormation provisions the Auto Scaling group, load balancer, and database for you. After the stack has been successfully created, your AWS resources are up and running. You can delete the stack just as easily, which deletes all the resources in the stack. By using CloudFormation, you easily manage a collection of resources as a single unit.

Quickly replicate your infrastructure

If your application requires additional availability, you might replicate it in multiple regions so that if one region becomes unavailable, your users can still use your application in other regions. The challenge in replicating your application is that it also requires you to replicate your resources. Not only do you need to record all the resources that your application requires, but you must also provision and configure those resources in each region.

Reuse your CloudFormation template to create your resources in a consistent and repeatable manner. To reuse your template, describe your resources once and then provision the same resources over and over in multiple regions.

Easily control and track changes to your infrastructure

In some cases, you might have underlying resources that you want to upgrade incrementally. For example, you might change to a higher performing instance type in your Auto Scaling launch configuration so that you can reduce the maximum number of instances in your Auto Scaling group. If problems occur after you complete the update, you might need to roll back your infrastructure to the original settings. To do this manually, you not only have to remember which resources were changed, you also have to know what the original settings were.

When you provision your infrastructure with CloudFormation, the CloudFormation template describes exactly what resources are provisioned and their settings. Because these templates are text files, you simply track differences in your templates to track changes to your infrastructure, similar to the way developers control revisions to source code. For example, you can use a version control system with your templates so that you know exactly what changes were made, who made them, and when. If at any point you need to reverse changes to your infrastructure, you can use a previous version of your template.

Getting started with CloudFormation

CloudFormation is available through the CloudFormation console, API, AWS CLI, AWS SDKs , and through several integrations.

For an introduction to CloudFormation, see How CloudFormation works.

To start using CloudFormation, see Creating your first stack.

Related information

You can learn more about CloudFormation in this user guide, as well as the following resources:

- For product details and FAQs, see the AWS CloudFormation product page.
- For pricing information, see AWS CloudFormation pricing.

Getting started with CloudFormation

You can begin using CloudFormation through the AWS Management Console by creating a stack from an example template, which will help you learn the basics of stack creation. A *template* is a text file that defines all the resources within a stack. A *stack* is the deployment of a CloudFormation template. From a single template, you can create multiple stacks. Each stack contains a collection of AWS resources that can be managed as a single unit.

CloudFormation is a free service; however, you are charged for the AWS resources you include in your stacks at the current rates for each. For more information about AWS pricing, go to the detail page for each product on http://aws.amazon.com.

Topics

- How CloudFormation works
- Signing up for an AWS account
- Creating your first stack

How CloudFormation works

This topic describes how CloudFormation works and introduces you to the key concepts you'll need to know about as you use it.

Topics

- Key concepts
- How CloudFormation works
- Ways to get started with CloudFormation

Key concepts

When you use CloudFormation, you work with *templates* and *stacks*. You create templates to describe your AWS resources and their properties. Whenever you create a stack, CloudFormation provisions the resources that are described in your template.

Topics

- Templates
- Stacks
- Change sets

Templates

A CloudFormation template is a YAML or JSON formatted text file. You can save these files with any extension, such as `.yaml`, `.json`, `.template`, or `.txt`. CloudFormation uses these templates as blueprints for building your AWS resources. For example, in a template, you can describe an Amazon EC2 instance, such as the instance type, the AMI ID, block device mappings, and its Amazon EC2 key pair name. Whenever you create a stack, you also specify a template that CloudFormation uses to create whatever you described in the template.

For example, if you created a stack with the following template, CloudFormation provisions an instance with an `ami-0ff8a91507f77f867` AMI ID, `t2.micro` instance type, `testkey` key pair name, and an Amazon EBS volume.

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
```

```
    "Description": "A sample template",
    "Resources": {
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0ff8a91507f77f867",
                "InstanceType": "t2.micro",
                "KeyName": "testkey",
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sdm",
                        "Ebs": {
                            "VolumeType": "io1",
                            "Iops": 200,
                            "DeleteOnTermination": false,
                            "VolumeSize": 20
                        }
                    }
                ]
            }
        }
    }
}
```

You can also specify multiple resources in a single template and configure these resources to work together. For example, you can modify the previous template to include an Elastic IP address (EIP) and associate it with the Amazon EC2 instance, as shown in the following example:

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
```

```
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
  MyEIP:
    Type: AWS::EC2::EIP
    Properties:
      InstanceId: !Ref MyEC2Instance
```

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "A sample template",
    "Resources": {
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0ff8a91507f77f867",
                "InstanceType": "t2.micro",
                "KeyName": "testkey",
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sdm",
                        "Ebs": {
                            "VolumeType": "io1",
                            "Iops": 200,
                            "DeleteOnTermination": false,
                            "VolumeSize": 20
                        }
                    }
                ]
            }
        },
        "MyEIP": {
            "Type": "AWS::EC2::EIP",
            "Properties": {
                "InstanceId": {
                    "Ref": "MyEC2Instance"
                }
            }
        }
    }
}
```

The previous templates are centered around a single Amazon EC2 instance; however, CloudFormation templates have additional capabilities that you can use to build complex sets of resources and reuse those templates in multiple contexts. For example, you can add input parameters whose values are specified when you create a CloudFormation stack. In other words, you can specify a value like the instance type when you create a stack instead of when you create the template, making the template easier to reuse in different situations.

Stacks

When you use CloudFormation, you manage related resources as a single unit called a stack. You create, update, and delete a collection of resources by creating, updating, and deleting stacks. All the resources in a stack are defined by the stack's CloudFormation template. Suppose you created a template that includes an Auto Scaling group, Elastic Load Balancing load balancer, and an Amazon Relational Database Service (Amazon RDS) database instance. To create those resources, you create a stack by submitting the template that you created, and CloudFormation provisions all those resources for you.

Change sets

If you need to make changes to the running resources in a stack, you update the stack. Before making changes to your resources, you can generate a change set, which is a summary of your proposed changes. Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.

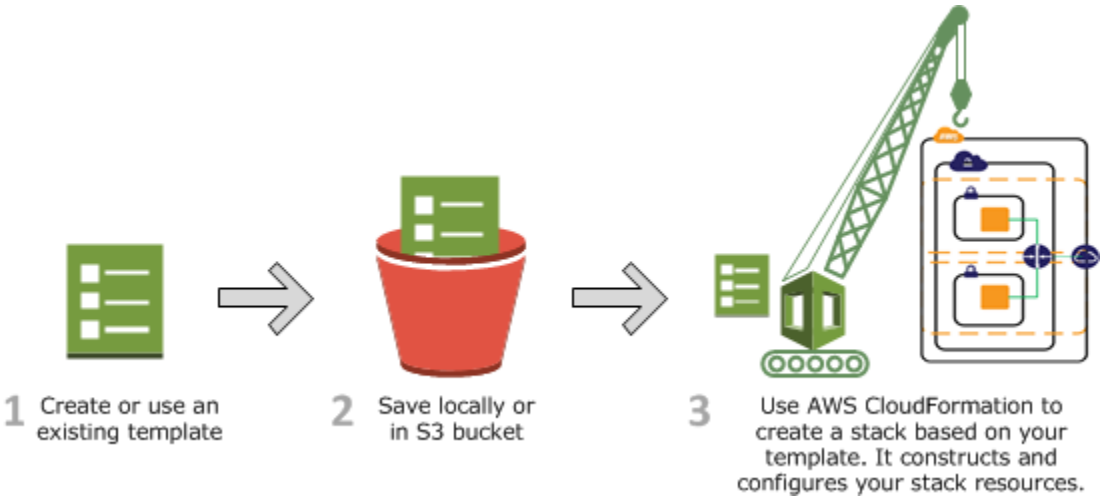
For example, if you change the name of an Amazon RDS database instance, CloudFormation will create a new database and delete the old one. You will lose the data in the old database unless you've already backed it up. If you generate a change set, you will see that your change will cause your database to be replaced, and you will be able to plan accordingly before you update your stack.

How CloudFormation works

When you use CloudFormation to create your stack, CloudFormation makes underlying service calls to AWS to provision and configure the resources described in your template. You need permission to create these resources. For example, to create EC2 instances by using CloudFormation, you need permissions to create instances. You manage these permissions with AWS Identity and Access Management (IAM).

The calls that CloudFormation makes are all declared by your template. For example, suppose you have a template that describes an EC2 instance with a `t2.micro` instance type. When you use that

template to create a stack, CloudFormation calls the Amazon EC2 create instance API and specifies the instance type as `t2.micro`. The following diagram summarizes the CloudFormation workflow for creating stacks.



To create a stack

1. Use a text editor to create a CloudFormation template in YAML or JSON format. The CloudFormation template describes the resources you want and their settings. Use Template studio to visualize and validate your template. This helps you make sure that your template is properly structured and free of syntax errors. For more information, see Working with CloudFormation templates.
2. Save the template locally or in an Amazon S3 bucket.
3. Create a CloudFormation stack by specifying the location of your template file, such as a path on your local computer or an Amazon S3 URL. If the template contains parameters, you can specify input values when you create the stack. Parameters allow you to pass in values to your template so that you can customize your resources each time you create a stack.

> **Note**
>
> If you specify a template file stored locally, CloudFormation uploads it to an S3 bucket in your AWS account. CloudFormation creates a bucket for each region in which you upload a template file. The buckets are accessible to anyone with Amazon Simple Storage Service (Amazon S3) permissions in your AWS account. If a bucket created by CloudFormation is already present, the template is added to that bucket.

You can use your own bucket and manage its permissions by manually uploading templates to Amazon S3. Then whenever you create or update a stack, specify the Amazon S3 URL of a template file.

Before provisioning resources, CloudFormation runs pre-deployment validation on your template. This checks for common errors such as invalid property syntax and resource name conflicts. If validation fails, the operation stops before any resources are created. Validation runs automatically on all Create Stack, Update Stack, and Create Change Set operations. To skip validation, use the `DisableValidation` parameter. For more information, see Validate stack deployments.

After all the resources have been created, CloudFormation reports that your stack has been created. You can then start using the resources in your stack. If stack creation fails, CloudFormation rolls back your changes by deleting the resources that it created.

Deployment time

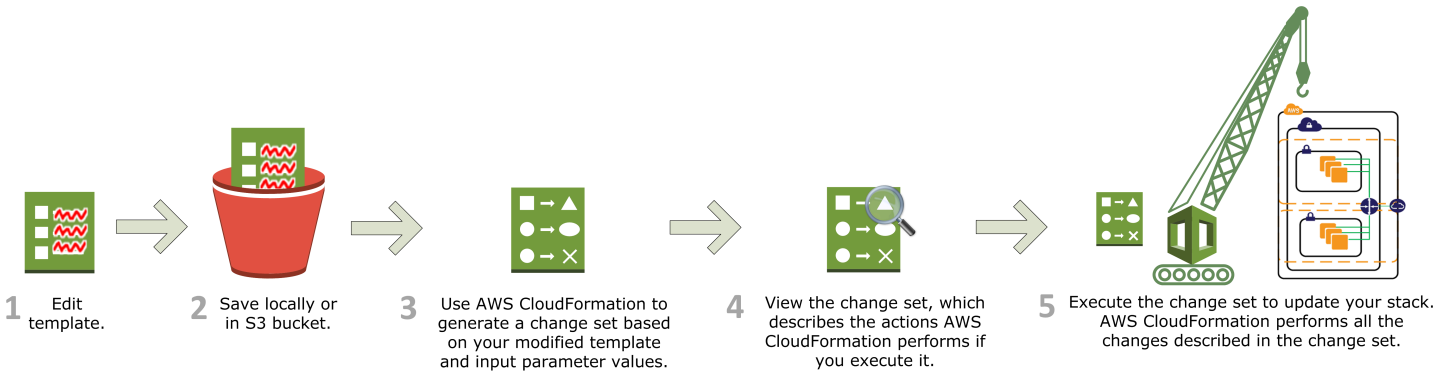
How long a stack operation takes depends on the resources in the stack. CloudFormation provisions independent resources in parallel and moves forward as soon as each resource is ready. In workloads that include resources such as database instances, load balancers, or NAT gateways, those resources take longer to become ready to serve traffic and account for most of the deployment time. For more information, see Understand stack deployment time.

Express mode

For faster stack operations during development, use express mode. Express mode completes stack operations as soon as CloudFormation applies the resource configuration, without waiting for resources to fully stabilize. For more information, see Express mode.

Updating a stack with a change set

When you need to update your stack's resources, you can modify the stack's template. You don't need to create a new stack and delete the old one. To update a stack, create a change set by submitting a modified version of the original stack template, different input parameter values, or both. CloudFormation compares the modified template with the original template and generates a change set. The change set lists the proposed changes. After reviewing the changes, you can start the change set to update your stack or you can create a new change set. The following diagram summarizes the workflow for updating a stack.



To update a stack with a change set

1. You can modify a CloudFormation stack template by using Template studio or a text editor. For more information, see Update your stack template.

   As you update your template, keep in mind that updates can cause interruptions. Depending on the resource and properties that you are updating, an update might interrupt or even replace an existing resource. For more information, see Understand update behaviors of stack resources.
2. Save the CloudFormation template locally or in an S3 bucket.
3. Create a change set by specifying the stack that you want to update and the location of the modified template, such as a path on your local computer or an Amazon S3 URL. For more information about creating change sets, see Update CloudFormation stacks using change sets.

   > **Note**
   >
   > If you specify a template that's stored on your local computer, CloudFormation automatically uploads your template to an S3 bucket in your AWS account.
4. View the change set to check that CloudFormation will perform the changes that you expect. For example, check whether CloudFormation will replace any critical stack resources. You can create as many change sets as you need until you have included the changes that you want.

   > **Important**
   >
   > Change sets don't indicate whether your stack update will be successful. For example, a change set doesn't check if you will surpass an account quota, if you're updating

a resource that doesn't support updates, or if you have insufficient permissions to modify a resource, which can cause a stack update to fail.

5. Initiate the change set that you want to apply to your stack. CloudFormation updates your stack by updating only the resources that you modified and signals that your stack has been successfully updated. If the stack updates fails, CloudFormation rolls back changes to restore the stack to the last known working state.

Ways to get started with CloudFormation

To create a hello world CloudFormation stack with the console, see Creating your first stack.

For guided learning, try the Getting Started with CloudFormation workshop, which offers hands-on experience with template development.

Signing up for an AWS account

Sign up for an AWS account

To get started with AWS, you need an AWS account. There are two ways to sign up for AWS. If you want complete control over account configuration and where your content is stored, or if you have regulated workloads such as HIPAA or FedRAMP, use Sign up for AWS (advanced). If you are a new user and want to use a login you already own to sign in to AWS and create an AWS environment with preconfigured defaults in select AWS Regions, use Sign up for AWS.

Note

For more information on how to manage who has access to what, see Control CloudFormation access with AWS Identity and Access Management.

Creating your first stack

This topic walks you through creating your first CloudFormation stack using the AWS Management Console. By following this tutorial, you'll learn how to provision basic AWS resources, monitor stack events, and generate outputs.

For this example, the CloudFormation template is written in YAML. YAML is a human-readable format that's widely used for defining infrastructure as code. As you learn more about CloudFormation, you might also encounter other templates in JSON format, but for this tutorial, YAML is chosen for its readability.

> **Note**
>
> CloudFormation is free, but you'll be charged for the Amazon EC2 and Amazon S3 resources you create. However, if you're new to AWS, you can take advantage of the Free Tier to minimize or eliminate costs during this learning process.

Topics

- Prerequisites
- Create a CloudFormation stack with the console
- Monitor stack creation
- Test the web server
- Troubleshooting
- Clean up
- Next steps

Prerequisites

- You must have access to an AWS account with an IAM user or role that has permissions to use Amazon EC2, Amazon S3, and CloudFormation, or administrative user access.
- You must have a Virtual Private Cloud (VPC) that has access to the internet. This walkthrough template requires a default VPC, which comes automatically with newer AWS accounts. If you don't have a default VPC, or if it was deleted, see the troubleshooting section in this topic for alternative solutions.

Create a CloudFormation stack with the console

To create a Hello world CloudFormation stack with the console

1. Open the CloudFormation console.

2. Choose **Create Stack**.
3. On the **Create stack** page, choose **Build with template studio**. The template opens in template studio.
4. To upload and validate the example template, do the following:

   a. Copy and paste the following CloudFormation template into the template editor:

```
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for WebServer with Security Group and EC2
 Instance

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.

  MyIP:
    Description: Your IP address in CIDR format (e.g. 203.0.113.1/32).
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access via my IP address
      SecurityGroupIngress:
        - IpProtocol: tcp
```

```
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP

  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData: !Base64 |
        #!/bin/bash
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html

Outputs:
  WebsiteURL:
    Value: !Join
      - ''
      - - http://
        - !GetAtt WebServer.PublicDnsName
    Description: Website URL
```

Before you move to the next step, let's take a moment to take a look at the template and understand some key CloudFormation concepts.

- The **`Parameters`** section declares values that can be passed to the template when you create the stack. Resources specified later in the template reference these values and use the data. Parameters are an effective way to specify information that you don't want to store in the template itself. They're also a way to specify information that might be unique to the specific application or configuration you are deploying.
- The template defines the following parameters:
  - **`LatestAmiId`** – Retrieves the latest Amazon Linux 2 AMI ID from the AWS Systems Manager Parameter Store.
  - **`InstanceType`** – Allows selection of the EC2 instance type (default: `t2.micro`, allowed: `t3.micro`, `t2.micro`).

- **`MyIP`** – Specifies the IP address range for HTTP access (default: 0.0.0.0/0, allowing access from any IP).

- The **`Resources`** section contains the definitions of the AWS resources you want to create with the template. Resource declarations are an efficient way to specify all these configuration settings at once. When you include resource declarations in a template, you can create and configure all the declared resources by using that template to create a stack. You can also create new stacks from the same template to launch identical configurations of resources.
- This template creates the following resources:
  - **`WebServerSecurityGroup`** – An EC2 security group that allows inbound HTTP traffic on port 80 from the specified IP range.
  - **`WebServer`** – An EC2 instance with the following configuration:
    - Uses the latest Amazon Linux 2 AMI
    - Applies the selected instance type
    - Adds the `WebServerSecurityGroup` to the `SecurityGroupIds` property
    - Includes a user data script to install Apache HTTP Server
- A logical name is specified at the beginning of each resource and parameter declaration. For example, `WebServerSecurityGroup` is the logical name assigned to the EC2 security group resource. The `Ref` function is then used to reference resources and parameters by their logical names in other parts of the template. When one resource references another resource, this creates a dependency between them.
- The **`Outputs`** section defines custom values that are returned after the stack creation. You can use the output values to return information from the resources in the stack, such as resource identifiers or URLs.
- The template defines one output:
  - **`WebsiteURL`** – The URL of the deployed web server, constructed using the EC2 instance's public DNS name. The `Join` function helps combine the fixed `http://` with the variable `PublicDnsName` into a single string, making it easy to output the full URL of the web server.

b. Choose **Validate** to make sure the YAML code is valid before uploading the template.

c. Choose **Save and continue** to save the template to an S3 bucket.

   d. From the dialog box that opens, make a note of the name of the S3 bucket so you can delete it later. Then, choose **Confirm and continue**. This returns you to the CloudFormation console with the S3 path to your template specified.

5. On the **Create stack** page, choose **Next**.
6. On the **Specify stack details** page, type a name in the **Stack name** field. The stack name can't contain spaces. For this example, use **`MyTestStack`**.
7. Under **Parameters**, specify parameter values as follows:

   - **LatestAmiId**: This is set by default to the latest Amazon Linux 2 AMI.
   - **InstanceType**: Choose either **t2.micro** or **t3.micro** for the EC2 instance type.

     > **Note**
     >
     > If you're new to AWS, you can use the free tier to launch and use a `t2.micro` instance for free for 12 months (in Regions where `t2.micro` is unavailable, you can use a `t3.micro` instance under the free tier).

   - **MyIP**: Specify your actual public IP address with a `/32` suffix. The `/32` suffix is used in CIDR notation to specify that a single IP address is allowed. It essentially means allow traffic to and from this specific IP address, and no others.
8. Choose **Next** twice to go to the **Review and create** page. For this tutorial, you can leave the defaults on the **Configure stack options** page as they are.
9. Review the information for the stack. When you're satisfied with the settings, choose **Submit**.

Monitor stack creation

After you choose **Submit**, CloudFormation begins creating the resources that are specified in the template. Your new stack, **`MyTestStack`**, appears in the list at the top portion of the **CloudFormation** console. Its status should be `CREATE_IN_PROGRESS`. You can see detailed status for a stack by viewing its events.

To view the events for the stack

1. On the CloudFormation console, choose the stack **`MyTestStack`** in the list.
2. In the stack details pane, choose the **Events** tab.

The console automatically refreshes the event list with the most recent events every 60 seconds.

The **Events** tab displays each major step in the creation of the stack sorted by the time of each event, with latest events on top.

The first event (at the bottom of the event list) is the start of the stack creation process:

```
2024-12-23 18:54 UTC-7 MyTestStack CREATE_IN_PROGRESS User initiated
```

Next are events that mark the beginning and completion of the creation of each resource. For example, creation of the EC2 instance results in the following entries:

```
2024-12-23 18:59 UTC-7 WebServer CREATE_COMPLETE
```

```
2024-12-23 18:54 UTC-7 WebServer CREATE_IN_PROGRESS Resource creation
initiated
```

The `CREATE_IN_PROGRESS` event is logged when CloudFormation reports that it has begun to create the resource. The `CREATE_COMPLETE` event is logged when the resource is successfully created.

When CloudFormation has successfully created the stack, you will see the following event at the top of the **Events** tab:

```
2024-12-23 19:17 UTC-7 MyTestStack CREATE_COMPLETE
```

If CloudFormation can't create a resource, it reports a `CREATE_FAILED` event and, by default, rolls back the stack and deletes any resources that have been created. The **Status Reason** column displays the issue that caused the failure.

After the stack is created, you can go to the **Resources** tab to view the EC2 instance and security group you created.

Test the web server

After the stack is successfully created, navigate to the **Outputs** tab in the CloudFormation console. Look for the **WebsiteURL** field. This will contain the public URL of your EC2 instance.

Open a browser and go to the URL listed under **WebsiteURL**. You should see a simple "Hello World!" message displayed in the browser.

This confirms that your EC2 instance is running Apache HTTP Server and serving a basic web page.

Troubleshooting

If you experience a rollback during stack creation, it might be due to a missing VPC. Here's how to resolve this issue.

No default VPC available

The template in this walkthrough requires a default VPC. If your stack creation fails because of VPC or subnet availability errors, you might not have a default VPC in your account. You have the following options:

- **Create a new default VPC** – You can create a new default VPC through the Amazon VPC console. For instructions, see Create a default VPC in the *Amazon VPC User Guide*.
- **Modify the template to specify a subnet** – If you have a non-default VPC, you can modify the template to explicitly specify the VPC and subnet IDs. Add the following parameter to the template:

  ```
  SubnetId:
    Description: The subnet ID to launch the instance into
    Type: AWS::EC2::Subnet::Id
  ```

  Then, update the `WebServer` resource to include the subnet ID:

  ```
  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      SubnetId: !Ref SubnetId
      UserData: !Base64 |
        #!/bin/bash
        yum update -y
        yum install -y httpd
        systemctl start httpd
  ```

```
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html
```

When creating the stack, you'll need to specify a subnet that has internet access for the web server to be reachable.

Clean up

To make sure you aren't charged for any unwanted services, you can clean up by deleting the stack and its resources. You can also delete the S3 bucket that stores the stack's template.

To delete the stack and its resources

1. Open the CloudFormation console.
2. On the **Stacks** page, select the option next to the name of the stack you created (**`MyTestStack`**) and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.
4. Monitor the progress of the stack deletion process on the **Event** tab. The status for **`MyTestStack`** changes to `DELETE_IN_PROGRESS`. When CloudFormation completes the deletion of the stack, it removes the stack from the list.

If you are done working with the example template and no longer need your Amazon S3 bucket, delete it. Before you can delete a bucket, you must first empty it. Emptying a bucket deletes all objects in it.

To empty and delete the Amazon S3 bucket

1. Open the Amazon S3 console.
2. In the navigation pane on the left side of the console, choose **Buckets**.
3. In the **Buckets** list, select the option next to the name of the bucket that you created for this tutorial, and then choose **Empty**.
4. On the **Empty bucket** page, confirm that you want to empty the bucket by typing **`permanently delete`** into the text field, and then choose **Empty**.
5. Monitor the progress of the bucket emptying process on the **Empty bucket: Status** page.
6. To return to your bucket list, choose **Exit**.

7. Select the option next to the name of the bucket, and then choose **Delete**.
8. When prompted for confirmation, type the name of the bucket and then choose **Delete bucket**.
9. Monitor the progress of the bucket deletion process from the **Buckets** list. When Amazon S3 completes the deletion of the bucket, it removes the bucket from the list.

Next steps

Congratulations! You successfully created a stack, monitored its creations, and used its output.

To continue learning:

- Learn more about templates so that you can create your own. For more information, see Working with CloudFormation templates.
- Try the Getting Started with CloudFormation workshop for more hands-on practice with template creation.
- For a shortened version of Getting Started with CloudFormation, see Deploy applications on Amazon EC2. This topic describes the same scenario of using a CloudFormation helper script, `cfn-init`, to bootstrap an Amazon EC2 instance.

CloudFormation best practices

Best practices are recommendations that can help you use CloudFormation more effectively and adopt safe practices throughout its entire workflow. Learn how to plan and organize your stacks, create templates that describe your resources and the software applications that run on them, and manage your stacks and their resources. The following best practices are based on real-world experience from current CloudFormation customers.

Planning and organizing

- Shorten the feedback loop to improve development velocity
- Organize your stacks by lifecycle and ownership
- Use cross-stack references to share values between stacks
- Use CloudFormation StackSets for multi-account and multi-region deployments
- Reuse templates to replicate stacks in multiple environments
- Verify quotas for all resource types
- Use modules to reuse resource configurations
- Adopt infrastructure as code practices

Creating templates

- Don't embed credentials in your templates
- Use AWS-specific parameter types
- Use parameter constraints
- Use pseudo parameters to promote portability
- Use `AWS::CloudFormation::Init` to deploy software applications on Amazon EC2 instances
- Use the latest helper scripts
- Validate templates before using them
- Using YAML or JSON for template authoring
- Implement a comprehensive tagging strategy
- Leverage template macros for advanced transformations

Managing stacks

- Manage all stack resources through CloudFormation

- Create change sets before updating your stacks
- Use stack policies to protect resources
- Use AWS CloudTrail to log CloudFormation calls
- Use code reviews and revision controls to manage your templates
- Update your Amazon EC2 instances regularly
- Use drift detection regularly
- Configure rollback triggers for automatic recovery
- Implement effective stack refactoring strategies
- Use CloudFormation Hooks for lifecycle management

Authoring tools

- Use IaC Generator to create templates from existing resources
- Use template studio to edit templates visually
- Consider using AWS Cloud Development Kit (AWS CDK) for complex infrastructure

Security and compliance

- Use IAM to control access
- Apply the principle of least privilege
- Secure sensitive parameters
- Implement policy as code with Guard

Shorten the feedback loop to improve development velocity

Adopt practices and tools that help you shorten the feedback loop for your infrastructure you describe with CloudFormation templates. This includes performing early linting and testing of your templates in your workstation; when you do, you have the opportunity to discover potential syntax and configuration issues even before you submit your contributions to a source code repository. Early discovery of such issues helps with preventing them from reaching formal lifecycle environments, such as development, quality assurance, and production. This early-testing, fail-fast approach gives you the benefits of reducing rework wait time, reducing potential areas of impact, and increasing your level of confidence in having successful provisioning operations.

Tooling choices that help you achieve fail-fast practices include the CloudFormation Linter (`cfn-lint`) and TaskCat command line tools. The `cfn-lint` tool gives you the ability to validate your CloudFormation templates against the CloudFormation Resource Specification. This includes

checking valid values for resource properties, as well as best practices. Plugins for `cfn-lint` are available for a number of code editors; this gives you the ability to visualize issues within your editor and to get direct linter feedback. You can also choose to integrate `cfn-lint` in your source code repository's configuration, so that you can perform template validation when you commit your contributions. For more information, see Git pre-commit validation of CloudFormation templates with `cfn-lint`. Once you have performed your initial linting—and fixed any issues `cfn-lint` might have raised—you can use TaskCat to test your templates by programmatically creating stacks in the AWS Regions you choose. TaskCat also generates a report with a pass/fail grades for each Region you chose.

For a step-by-step, hands-on walkthrough on how to use both tools to shorten the feedback loop, follow the Linting and Testing lab of the CloudFormation Workshop.

CloudFormation pre-deployment validation automatically catches common errors before any resources are provisioned. These errors include invalid property syntax and resource name conflicts. Pre-deployment validation runs by default on all Create Stack, Update Stack, and Create Change Set operations. No configuration is required. For CDK projects, `cdk validate` runs CloudFormation pre-deployment validation against your synthesized templates.

Organize your stacks by lifecycle and ownership

Use the lifecycle and ownership of your AWS resources to help you decide what resources should go in each stack. Initially, you might put all your resources in one stack, but as your stack grows in scale and broadens in scope, managing a single stack can be cumbersome and time consuming. By grouping resources with common lifecycles and ownership, owners can make changes to their set of resources by using their own process and schedule without affecting other resources.

For example, imagine a team of developers and engineers who own a website that's hosted on Amazon EC2 Auto Scaling instances behind a load balancer. Because the website has its own lifecycle and is maintained by the website team, you can create a stack for the website and its resources. Now imagine that the website also uses back-end databases, where the databases are in a separate stack that are owned and maintained by database administrators. Whenever the website team or database team needs to update their resources, they can do so without affecting each other's stack. If all resources were in a single stack, coordinating, and communicating updates can be difficult.

For additional guidance about organizing your stacks, you can use two common frameworks: a multi-layered architecture and service-oriented architecture (SOA).

A layered architecture organizes stacks into multiple horizontal layers that build on top of one another, where each layer has a dependency on the layer directly below it. You can have one or more stacks in each layer, but within each layer, your stacks should have AWS resources with similar lifecycles and ownership.

With a service-oriented architecture, you can organize big business problems into manageable parts. Each of these parts is a service that has a clearly defined purpose and represents a self-contained unit of functionality. You can map these services to a stack, where each stack has its own lifecycle and owners. These services (stacks) can be wired together so that they can interact with one another.

Use cross-stack references to share values between stacks

When you organize your AWS resources based on lifecycle and ownership, you might want to build a stack that uses resources that are in another stack. You can hardcode values or use input parameters to pass resource names and IDs. However, these methods can make templates difficult to reuse or can increase the overhead to get a stack running. Instead, use cross-stack references to share values between stacks. CloudFormation provides two approaches:

- `Fn::ImportValue` – Import values that another stack has explicitly exported. This creates a strong reference within the same account and Region. CloudFormation prevents you from deleting the exporting stack while other stacks depend on its exports.
- `Fn::GetStackOutput` – Reference any stack output directly, including outputs from stacks in other AWS accounts or Regions, without requiring explicit exports. This creates a weak reference that is resolved at create or update time.

For more information, see Get exported outputs from a deployed CloudFormation stack and Fn::GetStackOutput.

Use CloudFormation StackSets for multi-account and multi-region deployments

CloudFormation StackSets extend the capability of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation. Use StackSets for deploying common infrastructure components, compliance controls, or shared services across your organization.

When using StackSets, implement service-managed permissions with AWS Organizations for simplified permission management. This approach allows you to deploy StackSets to accounts within your organization without the need to manually configure IAM roles in each account.

For more information on StackSets see StackSets concepts.

Verify quotas for all resource types

Before launching a stack, ensure that you can create all the resources that you want without hitting your AWS account limits. If you hit a limit, CloudFormation won't create your stack successfully until you increase your quota or delete extra resources. Each service can have various limits that you should be aware of before launching a stack. For example, by default, you can only launch 2000 CloudFormation stacks per Region in your AWS account. For more information about limits and how to increase the default limits, see AWS service quotas in the *AWS General Reference*.

Reuse templates to replicate stacks in multiple environments

After you have your stacks and resources set up, you can reuse your templates to replicate your infrastructure in multiple environments. For example, you can create environments for development, testing, and production so that you can test changes before implementing them into production. To make templates reusable, use the parameters, mappings, and conditions sections so that you can customize your stacks when you create them. For example, for your development environments, you can specify a lower-cost instance type compared to your production environment, but all other configurations and settings remain the same. For more information about parameters, mappings, and conditions, see CloudFormation template sections.

Use modules to reuse resource configurations

As your infrastructure grows, common patterns can emerge in which you declare the same components in each of your templates. *Modules* are a way for you to package resource configurations for inclusion across stack templates, in a transparent, manageable, and repeatable way. Modules can encapsulate common service configurations and best practices as modular, customizable building blocks for you to include in your stack templates.

These building blocks can be for a single resource, like best practices for defining an Amazon Elastic Compute Cloud (Amazon EC2) instance, or they can be for multiple resources, to define common patterns of application architecture. These building blocks can be nested into other

modules, so you can stack your best practices into higher-level building blocks. CloudFormation modules are available in the CloudFormation registry, so you can use them just like a native resource. When you use a CloudFormation module, the module template is expanded into the consuming template, which makes it possible for you to access the resources inside the module using a Ref or Fn::GetAtt. For more information, see Create reusable resource configurations that can be included across templates with CloudFormation modules.

Adopt infrastructure as code practices

Treat your CloudFormation templates as code by implementing infrastructure as code (IaC) practices. Store your templates in version control systems, implement code reviews, and use automated testing to validate changes. This approach ensures consistency, improves collaboration, and provides an audit trail for infrastructure changes.

Consider implementing CI/CD pipelines for your infrastructure code to automate the testing and deployment of your CloudFormation templates. Tools like AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy can be used to create automated workflows for your infrastructure deployments.

For more information on implementing IaC best practices, see Using AWS CloudFormation as an IaC tool.

For more information on using continuous delivery with CloudFormation, see Continuous delivery with CodePipeline.

Don't embed credentials in your templates

Rather than embedding sensitive information in your CloudFormation templates, we recommend you use *dynamic references* in your stack template.

Dynamic references provide a compact, powerful way for you to reference external values that are stored and managed in other services, such as the AWS Systems Manager Parameter Store or AWS Secrets Manager. When you use a dynamic reference, CloudFormation retrieves the value of the specified reference when necessary during stack and change set operations, and passes the value to the appropriate resource. However, CloudFormation never stores the actual reference value. For more information, see Using dynamic references to specify template values.

AWS Secrets Manager helps you to securely encrypt, store, and retrieve credentials for your databases and other services. The AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management.

For more information on defining template parameters, see CloudFormation template Parameters syntax.

Use AWS-specific parameter types

If your template requires inputs for existing AWS-specific values, such as existing Amazon Virtual Private Cloud IDs or an Amazon EC2 key pair name, use AWS-specific parameter types. For example, you can specify a parameter as type `AWS::EC2::KeyPair::KeyName`, which takes an existing key pair name that's in your AWS account and in the Region where you are creating the stack. CloudFormation can quickly validate values for AWS-specific parameter types before creating your stack. Also, if you use the CloudFormation console, CloudFormation shows a drop down list of valid values, so you don't have to look up or memorize the correct VPC IDs or key pair names. For more information, see Specify existing resources at runtime with CloudFormation-supplied parameter types.

Use parameter constraints

With constraints, you can describe allowed input values so that CloudFormation catches any not valid values before creating a stack. You can set constraints such as a minimum length, maximum length, and allowed patterns. For example, you can set constraints on a database user name value so that it must be a minimum length of eight character and contain only alphanumeric characters. For more information, see CloudFormation template Parameters syntax.

Use pseudo parameters to promote portability

You can use pseudo parameters in your templates as arguments for intrinsic functions, such as `Ref` and `Fn::Sub`. Pseudo parameters are parameters that are predefined by CloudFormation. You don't declare them in your template. Using pseudo parameters in intrinsic functions increases the portability of your stack templates across Regions and accounts.

For example, imagine you wanted to create a template where, for a given resource property, you need to specify the Amazon Resource Name (ARN) of another existing resource. In this case, the existing resource is an AWS Systems Manager Parameter Store resource with the following ARN: `arn:aws:ssm:us-east-1:123456789012:parameter/MySampleParameter`. You will need to adapt the ARN format to your target AWS partition, Region, and account ID. Instead of hard-coding these values, you can use `AWS::Partition`, `AWS::Region`, and

AWS::AccountId pseudo parameters to make your template more portable. In this case, the following example shows you how to concatenate elements in an ARN with CloudFormation: !Sub 'arn:${AWS::Partition}:ssm:${AWS::Region}:${AWS::AccountId}:parameter/MySampleParameter.

For another example, assume you want to share resources or configurations across multiple stacks. In this example, assume you have created a subnet for your VPC, and then exported its ID for use with other stacks in the same AWS account and Region. In another stack, you reference the exported value of the subnet ID when describing an Amazon EC2 instance. For a detailed example of using the Export output field and Fn::ImportValue intrinsic function, see Refer to resource outputs in another CloudFormation stack.

Stack exports must be unique per account and Region. So, in this case, you can use the AWS::StackName pseudo parameter to create a prefix for your export. Since stack names must also be unique per account and Region, the usage of this pseudo parameter as a prefix increases the possibility of having a unique export name while also promoting a reusable approach across stacks from where you export values. Alternatively, you can use a prefix of your own choice.

Use AWS::CloudFormation::Init to deploy software applications on Amazon EC2 instances

When you launch stacks, you can install and configure software applications on Amazon EC2 instances by using the cfn-init helper script and the AWS::CloudFormation::Init resource. By using AWS::CloudFormation::Init, you can describe the configurations that you want rather than scripting procedural steps. You can also update configurations without recreating instances. And if anything goes wrong with your configuration, CloudFormation generates logs that you can use to investigate issues.

In your template, specify installation and configuration states in the AWS::CloudFormation::Init resource. For a walkthrough that shows how to use cfn-init and AWS::CloudFormation::Init, see Deploy applications on Amazon EC2.

Use the latest helper scripts

The CloudFormation helper scripts are updated periodically. Be sure you include the following command in the UserData property of your template before you call the helper scripts to ensure that your launched instances get the latest helper scripts:

```
yum install -y aws-cfn-bootstrap
```

For more information about getting the latest helper scripts, see the CloudFormation helper scripts reference in the *CloudFormation Template Reference Guide*.

Validate templates before using them

Before you use a template to create or update a stack, you can use CloudFormation to validate it. Validating a template can help you catch syntax and some semantic errors, such as circular dependencies, before CloudFormation creates any resources. If you use the CloudFormation console, the console automatically validates the template after you specify input parameters. For the AWS CLI or CloudFormation API, use the validate-template CLI command or ValidateTemplate API operation.

During validation, CloudFormation first checks if the template is valid JSON. If it isn't, CloudFormation checks if the template is valid YAML. If both checks fail, CloudFormation returns a template validation error.

To find more problems before deployment, you can also check templates on your computer. For more information about installing and using `cfn-lint`, `cloudformation-validate`, and the CloudFormation Language Server, see Validating templates. To understand the limitations of local validation, see Understand validation scope.

Validate templates for organization policy compliance

You can also validate your template for compliance to organization policy guidelines. AWS CloudFormation Guard (`cfn-guard`) is a command line interface (CLI) tool that provides a policy-as-code language to define rules that can check for both required and prohibited resource configurations. It then enables you to validate your templates against those rules. For example, administrators can create rules to ensure that users always create encrypted Amazon S3 buckets.

You can use `cfn-guard` either locally, while editing templates, or automatically as part of a CI/CD pipeline to stop deployment of non-compliant resources.

Additionally, `cfn-guard` includes a feature, `rulegen`, that enables you to extract rules from existing compliant CloudFormation templates.

For installation and usage examples, see Validate templates with Guard.

Using YAML or JSON for template authoring

CloudFormation supports both YAML and JSON formats for templates. Each has its advantages, and the choice depends on your specific needs:

Use YAML when

- You prioritize human readability and maintainability
- You want to include comments to document your template
- You're working on complex templates with nested structures
- You want to use YAML-specific features like anchors and aliases to reduce repetition

Use JSON when:

- You need to integrate with tools or systems that prefer JSON
- You're working with programmatic template generation or manipulation
- You require strict data validation

YAML is generally recommended for manual template authoring due to its readability and comment support. It's particularly useful for complex templates where the indentation-based structure helps visualize resource hierarchies. JSON can be advantageous in automated workflows or when working with APIs that expect JSON input. It's also beneficial when you need to ensure strict adherence to a specific structure. Regardless of the format you choose, focus on creating well-structured, documented, and maintainable templates. If using YAML, leverage its features like anchors and aliases to reduce repetition and improve maintainability.

Implement a comprehensive tagging strategy

Implement a consistent tagging strategy for all resources created by your CloudFormation templates. Tags help with resource organization, cost allocation, access control, and automation. Consider including tags for environment, owner, cost center, application, and purpose.

Use the `AWS::CloudFormation::Stack` resource's `Tags` property to apply tags to all supported resources in a stack. You can also use the `TagSpecifications` property available on many resource types to apply tags during resource creation.

For more information on tagging, see Resource tag.

Leverage template macros for advanced transformations

CloudFormation macros enable you to perform custom processing on templates, from simple actions like find-and-replace operations to complex transformations that generate additional resources. Use macros to extend the capabilities of CloudFormation templates and implement reusable patterns across your organization.

The AWS Serverless Application Model is an example of a macro that simplifies the development of serverless applications. Consider creating custom macros for organization-specific patterns and requirements.

For more information on using macros in your templates, see Overview of CloudFormation macros.

Manage all stack resources through CloudFormation

After you launch a stack, use the CloudFormation console, API, or AWS CLI to update resources in your stack. Don't make changes to stack resources outside of CloudFormation. Doing so can create a mismatch between your stack's template and the current state of your stack resources, which can cause errors if you update or delete the stack. This is known as drift. If a change is made to a resource outside of the CloudFormation template and you update the stack, the changes made directly to the resource will be discarded, and the resource configuration will revert to the configuration in the template.

For more information on drift, see What is drift?.

For more information on updating stacks, see Update a CloudFormation stack.

Create change sets before updating your stacks

Change sets allow you to see how proposed changes to a stack might impact your running resources before you implement them. CloudFormation doesn't make any changes to your stack until you run the change set, allowing you to decide whether to proceed with your proposed changes or create another change set.

Use change sets to check how your changes might impact your running resources, especially for critical resources. For example, if you change the name of an Amazon RDS database instance, CloudFormation will create a new database and delete the old one; you will lose the data in the old database unless you've already backed it up. If you generate a change set, you will see that your

change will replace your database. This can help you plan before you update your stack. For more information, see Update CloudFormation stacks using change sets.

> **Note**
>
> If you use direct stack updates instead of change sets, pre-deployment validation still validates your templates automatically. However, change sets provide additional benefits. Use change sets to preview which resources will be added, modified, or replaced before you execute the update.

Use stack policies to protect resources

Stack policies help protect critical stack resources from unintentional updates that could cause resources to be interrupted or even replaced. A stack policy is a JSON document that describes what update actions can be performed on designated resources. Specify a stack policy whenever you create a stack that has critical resources.

During a stack update, you must explicitly specify the protected resources that you want to update; otherwise, no changes are made to protected resources. For more information, see Prevent updates to stack resources.

Use AWS CloudTrail to log CloudFormation calls

AWS CloudTrail tracks anyone making CloudFormation API calls in your AWS account. API calls are logged whenever anyone uses the CloudFormation API, the CloudFormation console, a back-end console, or CloudFormation AWS CLI commands. Enable logging and specify an Amazon S3 bucket to store the logs. That way, if you ever need to, you can audit who made what CloudFormation call in your account.

For more information, see Logging CloudFormation API calls with AWS CloudTrail.

Use code reviews and revision controls to manage your templates

Your stack templates describe the configuration of your AWS resources, such as their property values. To review changes and to keep an exact history of your resources, use code reviews and

revision controls. These methods can help you track changes between different versions of your templates, which can help you track changes to your stack resources. Also, by maintaining a history, you can always revert your stack to a certain version of your template.

Update your Amazon EC2 instances regularly

On all your Amazon EC2 Windows instances and Amazon EC2 Linux instances created with CloudFormation, regularly run the `yum update` command to update the RPM package. This ensures that you get the latest fixes and security updates.

Use drift detection regularly

Regularly use the CloudFormation drift detection feature to identify resources that have been modified outside of CloudFormation management. Detecting and resolving drift helps maintain the integrity of your infrastructure as code approach and ensures that your templates accurately reflect the state of your deployed resources.

Consider implementing automated drift detection as part of your operational procedures. You can use AWS Lambda functions triggered by Amazon EventBridge rules to periodically check for drift and notify your team when discrepancies are detected.

For more information on drift, see Detect unmanaged configuration changes to stacks and resources with drift detection.

Configure rollback triggers for automatic recovery

Use rollback triggers to specify Amazon CloudWatch alarms that CloudFormation should monitor during stack creation and update operations. If any of the specified alarms go into the `ALARM` state, CloudFormation automatically rolls back the entire stack operation, helping to ensure that your infrastructure remains in a stable state.

Configure rollback triggers for critical metrics such as application error rates, system resource utilization, or custom business metrics that indicate the health of your application and infrastructure.

For more information on rollback triggers, see Roll back your stack on alarm breach.

Implement effective stack refactoring strategies

As your infrastructure evolves, you may need to refactor your CloudFormation stacks to improve maintainability, reduce complexity, or adapt to changing requirements. Stack refactoring involves restructuring your templates and resources while preserving their external behavior and functionality. Stack refactoring is beneficial to use with CloudFormation in the following ways:

- *Splitting monolithic stacks*: Breaking down large, complex stacks into smaller, more manageable stacks organized by lifecycle or ownership
- *Consolidating related resources*: Combining related resources from multiple stacks into a single, cohesive stack to simplify management
- *Extracting reusable components*: Moving common patterns into modules or nested stacks to promote reuse and consistency
- *Improving resource organization*: Restructuring resources within a stack to better reflect their relationships and dependencies

For more information on refactoring your CloudFormation stacks, see Stack refactoring.

Use CloudFormation Hooks for lifecycle management

CloudFormation Hooks provide code that proactively inspects the configuration of your AWS resources before provisioning, and perform complex validation checks. Hooks check if your resources, stacks, and changes sets are compliant with your organization's security, operational, and cost optimization needs. They providing warnings before a resource provisioning, or failing the operation and stopping it altogether, depending on how it has been configured. Violations and warnings are logged in Amazon CloudWatch to provide visibility into non-compliant deployments.

For more information on these best practices for Hooks see, AWS CloudFormation Hooks concepts.

For more information on what Hooks can do for your CloudFormation resources see, What are AWS CloudFormation Hooks?

Use IaC Generator to create templates from existing resources

The CloudFormation IaC (infrastructure as code) Generator helps you create CloudFormation templates from your existing AWS resources. This capability is particularly useful when you need

to replicate existing infrastructure, document manually created resources, or bring previously unmanaged resources under CloudFormation management. IaC generator is useful for creating your CloudFormation templates in the following ways:

- *Accelerated template creation*: Generate templates from existing resources instead of writing them from scratch
- *Consistent infrastructure*: Ensure new environments match existing ones by using generated templates as a starting point
- *Migration to infrastructure as code*: Gradually bring manually created resources under CloudFormation management
- *Documentation*: Create a record of your existing infrastructure in template form

For more information on IaC Generator, see Generate templates from existing resources with IaC generator.

Use template studio to edit templates visually

Template studio lets you edit a CloudFormation template while viewing a diagram of its resources and their relationships. Consider using template studio for the following tasks:

- *Template authoring*: Edit YAML or JSON and see the diagram update to match the template.
- *Template modernization*: Visualize existing templates to understand their structure and identify opportunities for improvement.
- *Training and onboarding*: Help team members understand CloudFormation concepts and AWS service relationships.
- *Reviews*: Use the resources diagram to support architecture, security, and compliance reviews.

For more information, see Template studio.

Consider using AWS Cloud Development Kit (AWS CDK) for complex infrastructure

For complex infrastructure requirements, consider using the CDK to define your cloud resources using familiar programming languages like TypeScript, Python, Java, and .NET. AWS CDK generates CloudFormation templates from your code, allowing you to leverage the full capabilities of

CloudFormation while using the abstractions and programming constructs of your preferred language.

The AWS CDK provides high-level constructs that encapsulate best practices and simplify the definition of common infrastructure patterns. This can significantly reduce the amount of code needed to define your infrastructure while ensuring adherence to best practices.

For more information on the CDK, see AWS Cloud Development Kit (AWS CDK).

Use IAM to control access

IAM is an AWS service that you can use to manage users and their permissions in AWS. You can use IAM with CloudFormation to specify what CloudFormation actions users can perform, such as viewing stack templates, creating stacks, or deleting stacks. Furthermore, anyone managing CloudFormation stacks will require permissions to resources within those stacks. For example, if users want to use CloudFormation to launch, update, or terminate Amazon EC2 instances, they must have permission to call the relevant Amazon EC2 actions.

In most cases, users require full access to manage all of the resources in a template. CloudFormation makes calls to create, modify, and delete those resources on their behalf. To separate permissions between a user and the CloudFormation service, use a service role. CloudFormation uses the service role's policy to make calls instead of the user's policy. For more information, see CloudFormation service role.

Apply the principle of least privilege

When configuring IAM roles for CloudFormation service roles or for resources created by your templates, always apply the principle of least privilege. Grant only the permissions necessary for the intended functionality, and avoid using wildcard permissions whenever possible.

Use IAM Access Analyzer to review the permissions granted to your CloudFormation service roles and identify unused permissions that can be removed. Regularly review and update IAM policies to ensure they remain aligned with your security requirements.

Secure sensitive parameters

For sensitive information such as passwords, API keys, and other secrets, use AWS Systems Manager Parameter Store or AWS Secrets Manager instead of embedding them directly in your

templates. Use dynamic references in your templates to securely retrieve these values during stack operations.

When using parameters in your templates, set the `NoEcho` property to `true` for sensitive parameters to prevent their values from being displayed in the console, API responses, or CLI output. Be aware that `NoEcho` doesn't prevent the value from being logged if it's passed to other services or resources that might log the value.

For more information on using AWS Systems Manager Parameter Store with CloudFormation see Get a plaintext value from AWS Systems Manager Parameter Store.

For more information on using the `NoEcho` property, see CloudFormation template Parameters syntax.

For more information on using AWS Secrets Manager with CloudFormation see Create AWS Secrets Manager secrets in AWS CloudFormation.

Implement policy as code with Guard

AWS CloudFormation Guard (`cfn-guard`) is a policy-as-code tool that allows you to define and enforce rules for your CloudFormation templates. Use `cfn-guard` to ensure that your templates comply with organizational policies, security best practices, and governance requirements.

Integrate `cfn-guard` into your CI/CD pipelines to automatically validate templates against your policy rules before deployment. This helps prevent non-compliant resources from being deployed to your environment and provides early feedback to developers about policy violations.

For setup and usage examples, see Validate templates with Guard.

Working with CloudFormation templates

An AWS CloudFormation template defines the AWS resources you want to create, update, or delete as part of a stack. It consists of several sections, but the only required section is the Resources section, which must declare at least one resource.

You can create templates using the following methods:

- **Template studio** – Edit templates and view their resources in a visual diagram.
- **Text Editor** – Write templates directly in JSON or YAML syntax.
- **IaC generator** – Generate templates from resources provisioned in your account that are not currently managed by CloudFormation. The IaC generator works with a wide range of resource types that are supported by the Cloud Control API in your Region.

This section provides a comprehensive guide on how to use the different sections of a CloudFormation template and how to start creating stack templates. It covers the following topics:

Topics

- Where templates get stored
- Validating templates
- Getting started with templates
- Sample templates
- CloudFormation template format
- CloudFormation template sections
- CloudFormation Language Server
- Validate templates with cfn-lint
- Validate templates with cloudformation-validate
- Validate templates with Guard
- Edit templates visually with template studio
- Generate templates from existing resources with IaC generator
- Get values stored in other services using dynamic references
- Get AWS values using pseudo parameters
- Get exported outputs from a deployed CloudFormation stack

- Specify existing resources at runtime with CloudFormation-supplied parameter types
- CloudFormation walkthroughs
- CloudFormation template snippets
- Deploy Windows-based stacks using CloudFormation
- Extend your template's capabilities with CloudFormation-supplied resource types
- Create reusable resource configurations that can be included across templates with CloudFormation modules

Where templates get stored

Amazon S3 bucket

You can store CloudFormation templates in an Amazon S3 bucket. When creating or updating a stack, you can specify the S3 URL of the template instead of uploading it directly.

If you upload templates directly through the AWS Management Console or AWS CLI, an S3 bucket is automatically created for you. For more information, see Create a stack from the CloudFormation console.

Git repository

With Git sync, you can store templates in a Git repository. When creating or updating a stack, you can specify the Git repository location and branch containing the template instead of uploading it directly or referencing an S3 URL. CloudFormation automatically monitors the specified repository and branch for template changes. For more information, see Create a stack from repository source code with Git sync.

Validating templates

Validate your templates before deployment so that you can find and fix problems earlier. The CloudFormation service and the tools in this section check different parts of a template. You can use them together.

You can check whether a template uses valid JSON or YAML with the validate-template CLI command or by specifying your template in the AWS Management Console. The console performs this check automatically. These service-side checks don't run the additional property, security, or best-practice checks provided by the following tools:

- CloudFormation Language Server – Get suggestions, documentation, and validation feedback while you write templates in an editor.
- CloudFormation Linter – Check resource properties, allowed values, and common problems from the command line, an editor, or an automated build.
- CloudFormation Validate – Run local, offline checks from the command line or a library, and add custom rules in Common Expression Language (CEL), Rego, or Guard. AWS CDK uses this validator automatically after it synthesizes your templates.
- CloudFormation Guard – Write policy rules and check templates against your organization's security, compliance, and governance requirements.

Use with AI coding agents

AI coding agents can help you author, validate, and troubleshoot CloudFormation templates using the `aws-cloudformation` skill from the Agent Toolkit for AWS on GitHub. For installation instructions, see Agent setup guide.

Understand validation scope

Local tools check the template and the rules that you give them. They can't guarantee that deployment will succeed with the current resources, permissions, and quotas in an AWS account and AWS Region. Before deployment, review a change set and fix any problems reported by the CloudFormation service.

Getting started with templates

To get started with creating a CloudFormation template, follow these steps:

1. **Choose resources** – Identify the AWS resources you want to include in your stack, such as EC2 instances, VPCs, security groups, and more.
2. **Write the template** – Write the template in JSON or YAML format, defining the resources and their properties.
3. **Save the template** – Save the template locally with a file extension like: `.json`, `.yaml`, or `.txt`.
4. **Validate the template** – Validate the template using the methods described in the Validating templates section.
5. **Create a stack** – Create a stack using the validated template.

Plan to use the CloudFormation template reference

As you write your templates, you can find documentation for the detailed syntax for different resource types in the AWS resource and property types reference.

Often, your stack templates will require intrinsic functions to assign property values that are not available until runtime and special attributes to control the behavior of resources. As you write your template, refer to the following resources for guidance:

- Intrinsic function reference – Some commonly used intrinsic functions include:
  - `Ref` – Retrieves the value of a parameter or the physical ID of a resource.
  - `Sub` – Substitutes placeholders in strings with actual values.
  - `GetAtt` – Returns the value of an attribute from a resource in the template.
  - `Join` – Joins a set of values into a single string.
- Resource attribute reference – Some commonly used special attributes include:
  - `DependsOn` – Use this attribute to specify that one resource must be created after another.
  - `DeletionPolicy` – Use this attribute to specify how CloudFormation should handle the deletion of a resource.

Sample templates

CloudFormation provides open-source stack templates that you can use to get started. For more information, see CloudFormation Sample Templates on GitHub.

Keep in mind that these templates are not meant to be production-ready. You should take the time to learn how they work, adapt them to your needs, and make sure that they meet your company's compliance standards.

Each template in this repository passes CloudFormation Linter (cfn-lint) checks, and also a basic set of AWS CloudFormation Guard rules based on the Center for Internet Security (CIS) Top 20, with exceptions for some rules where it made sense to keep the sample focused on a single use case.

CloudFormation template format

You can author CloudFormation templates in JSON or YAML formats. Both formats serve the same purpose but offer distinct advantages in terms of readability and complexity.

- **JSON** – JSON is a lightweight data interchange format that's easy for machines to parse and generate. However, it can become cumbersome for humans to read and write, especially for complex configurations. In JSON, the template is structured using nested braces `{}` and brackets `[]` to define resources, parameters, and other components. Its syntax requires explicit declaration of every element, which can make the template verbose but ensures strict adherence to a structured format.
- **YAML** – YAML is designed to be more human-readable and less verbose than JSON. It uses indentation rather than braces and brackets to denote nesting, which can make it easier to visualize the hierarchy of resources and parameters. YAML is often preferred for its clarity and ease of use, especially when dealing with more complex templates. However, YAML's reliance on indentation can lead to errors if the spacing is not consistent, which requires careful attention to maintain accuracy.

Template structure

CloudFormation templates are divided into different sections, and each section is designed to hold a specific type of information. Some sections must be declared in a specific order, and for others, the order doesn't matter. However, as you build your template, it can be helpful to use the logical order shown in the following examples because values in one section might refer to values from a previous section.

When authoring templates, don't use duplicate major sections, such as the `Resources` section. Although CloudFormation might accept the template, it will have an undefined behavior when processing the template, and might incorrectly provision resources, or return inexplicable errors.

JSON

The following example shows the structure of a JSON-formatted template with all available sections.

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },
```

```
  "Parameters" : {
    set of parameters
  },

  "Rules" : {
    set of rules
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

YAML

The following example shows the structure of a YAML-formatted template with all available sections.

```
---
AWSTemplateFormatVersion: version date

Description:
  String

Metadata:
  template metadata
```

```
Parameters:
  set of parameters

Rules:
  set of rules

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

Comments

In JSON-formatted templates, comments are not supported. JSON, by design, doesn't include a syntax for comments, which means you can't add comments directly within the JSON structure. However, if you need to include explanatory notes or documentation, you can consider adding metadata. For more information, see Metadata attribute.

In YAML-formatted templates, you can include inline comments by using the # symbol.

The following example shows a YAML template with inline comments.

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample CloudFormation template with YAML comments.
# Resources section
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      # Linux AMI
      ImageId: ami-1234567890abcdef0
      InstanceType: t2.micro
```

```
      KeyName: MyKey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

Specifications

CloudFormation supports the following JSON and YAML specifications:

JSON

CloudFormation follows the ECMA-404 JSON standard. For more information about the JSON format, see http://www.json.org.

YAML

CloudFormation supports the YAML Version 1.1 specification with a few exceptions. CloudFormation doesn't support the following features:

- The `binary`, `omap`, `pairs`, `set`, and `timestamp` tags
- Aliases
- Hash merges

For more information about YAML, see https://yaml.org/.

Learn more

For each resource you specify in your template, you define its properties and values using the specific syntax rules of either JSON or YAML. For more information about the template syntax for each format, see CloudFormation template sections.

Using regular expressions in CloudFormation templates

You can use regular expressions (commonly known as regexes) in a number of places within your CloudFormation templates, such as for the `AllowedPattern` property when creating a template parameter.

All regular expressions in CloudFormation conform to the Java regex syntax. For a comprehensive description of the Java regex syntax and its constructs, see java.util.regex.Pattern.

If you write your CloudFormation template in JSON syntax, you must escape any backslash characters (\) in your regular expression by adding an additional backslash. This is because JSON interprets backslashes as escape characters, and you need to escape them to ensure they're treated as literal backslashes in the regular expression.

For example, if you include a \d in your regular expression to match a digit character, you will need to write it as \\d in your JSON template.

In the following example, the `AllowedPattern` property specifies a regular expression that matches four consecutive digit characters (`\d{4}`). However, since the regular expression is defined in a JSON template, the backslash character needs to be escaped with an additional backslash (\\d).

```
{
  "Parameters": {
    "MyParameter": {
      "Type": "String",
      "AllowedPattern": "\\d{4}"
    }
  }
}
```

If you write your CloudFormation template in YAML syntax, you must surround the regular expression with single quotation marks ('). No additional escaping is required.

```
Parameters:
  MyParameter:
    Type: String
    AllowedPattern: '\d{4}'
```

Note

Regular expressions in CloudFormation are only supported for validation purposes in specific contexts like `AllowedPattern`. They are not supported as pattern matching operations in CloudFormation intrinsic functions, such as `Fn::Equals`, which perform exact string comparison only, not pattern matching.

CloudFormation template sections

Every CloudFormation template consists of one or more sections, each serving a specific purpose.

The **Resources** section is required in every CloudFormation template and forms the core of the template. This section specifies the stack resources and their properties, such as an Amazon EC2 instance or an Amazon S3 bucket. Each resource is defined with a unique logical ID, type, and specific configuration details.

The **Parameters** section, while optional, plays an important role in making templates more flexible. It allows users to pass values at runtime when creating or updating a stack. These parameters can be referenced in the `Resources` and `Outputs` sections, enabling customization without altering the template itself. For instance, you might use parameters to specify instance types or environment settings that vary between deployments.

The **Outputs** section, also optional, defines the values that are returned when viewing a stack's properties. Outputs provide useful information such as resource identifiers or URLs, which can be leveraged for operational purposes or for integration with other stacks. This section helps users retrieve and use important details about the resources created by the template.

Other optional sections include **Mappings**, which function like lookup tables to manage conditional values. With mappings, you define key-value pairs and use them with the `Fn::FindInMap` intrinsic function in the `Resources` and `Outputs` sections. This is useful for scenarios where you need to adjust configurations based on conditions such as AWS Region or environment.

Metadata and **Rules** sections, though less commonly used, provide additional functionality. `Metadata` can include additional information about the template, while `Rules` validates a parameter or a combination of parameters during stack creation or updates, ensuring they meet specific criteria. The **Conditions** section further enhances flexibility by controlling whether certain resources are created or properties are assigned a value based on conditions like environment type.

Lastly, the **Transform** section is used to apply macros during the processing of the template. For serverless applications (also referred to as Lambda applications), it specifies the version of the AWS Serverless Application Model (AWS SAM) to use. When you specify a transform, you can use AWS SAM syntax to declare resources in your template. The model defines the syntax that you can use and how it's processed. You can also use the `AWS::Include` transform to include template snippets that are stored separately from the main CloudFormation template.

The following topics provide more information and examples for using each section.

Topics

- CloudFormation template Resources syntax
- CloudFormation template Parameters syntax
- CloudFormation template Outputs syntax
- CloudFormation template Mappings syntax
- CloudFormation template Metadata syntax
- CloudFormation template Rules syntax
- CloudFormation template Conditions syntax
- CloudFormation template Transform section
- CloudFormation template format version syntax
- CloudFormation template Description syntax

CloudFormation template Resources syntax

The `Resources` section is a required top-level section in a CloudFormation template. It declares the AWS resources that you want CloudFormation to provision and configure as part of your stack.

Syntax

The `Resources` section uses the following syntax:

JSON

```
"Resources" : {
    "LogicalResourceName1" : {
        "Type" : "AWS::ServiceName::ResourceType",
        "Properties" : {
            "PropertyName1" : "PropertyValue1",
            ...
        }
    },

    "LogicalResourceName2" : {
        "Type" : "AWS::ServiceName::ResourceType",
        "Properties" : {
            "PropertyName1" : "PropertyValue1",
            ...
```

```
        }
    }
}
```

YAML

```
Resources:
  LogicalResourceName1:
    Type: AWS::ServiceName::ResourceType
    Properties:
      PropertyName1: PropertyValue1
      ...

  LogicalResourceName2:
    Type: AWS::ServiceName::ResourceType
    Properties:
      PropertyName1: PropertyValue1
      ...
```

Logical ID (also called *logical name*)

Within a CloudFormation template, resources are identified by their logical resource names. These names must be alphanumeric (A-Za-z0-9) and unique within the template. Logical names are used to reference resources from other sections of the template.

Resource type

Each resource must have a `Type` attribute, which defines the kind of AWS resource it is. The `Type` attribute has the format `AWS::`*`ServiceName`*`::`*`ResourceType`*. For example, the `Type` attribute for an Amazon S3 bucket is `AWS::S3::Bucket`.

For the full list of supported resource types, see the AWS resource and property types reference.

Resource properties

Resource properties are additional options that you can specify to define configuration details for the specific resource type. Some properties are required, while others are optional. Some properties have default values, so specifying those properties is optional.

For details on the properties supported for each resource type, see the topics in AWS resource and property types reference.

Property values can be literal strings, lists of strings, Booleans, dynamic references, parameter references, pseudo references, or the value returned by a function. The following examples show you how to declare different property value types:

JSON

```
"Properties" : {
    "String" : "A string value",
    "Number" : 123,
    "LiteralList" : [ "first-value", "second-value" ],
    "Boolean" : true
}
```

YAML

```
Properties:
  String: A string value
  Number: 123
  LiteralList:
    - first-value
    - second-value
  Boolean: true
```

Physical ID

In addition to the logical ID, certain resources also have a physical ID, which is the actual assigned name for that resource, such as an EC2 instance ID or an S3 bucket name. Use the physical IDs to identify resources outside of CloudFormation templates, but only after the resources have been created. For example, suppose you give an EC2 instance resource a logical ID of `MyEC2Instance`. When CloudFormation creates the instance, CloudFormation automatically generates and assigns a physical ID (such as `i-1234567890abcdef0`) to the instance. You can use this physical ID to identify the instance and view its properties (such as the DNS name) by using the Amazon EC2 console.

For Amazon S3 buckets and many other resources, CloudFormation automatically generates a unique physical name for the resource if you don't explicitly specify one. This physical name is based on a combination of the name of the CloudFormation stack, the resource's logical name specified in the CloudFormation template, and a unique ID. For example, if you have an Amazon S3 bucket with the logical name `MyBucket` in a stack named `MyStack`, CloudFormation might name the bucket with the following physical name `MyStack-MyBucket-abcdefghijk1`.

For resources that support custom names, you can assign your own physical names to help you quickly identify resources. For example, you can name an S3 bucket that stores logs as `MyPerformanceLogs`. For more information, see Name type.

Referencing resources

Often, you need to set properties on one resource based on the name or property of another resource. For example, you might create an EC2 instance that uses EC2 security groups, or a CloudFront distribution backed by an S3 bucket. All of these resources can be created in the same CloudFormation template.

CloudFormation provides intrinsic functions that you can use to refer to other resources and their properties. These functions allow you to create dependencies between resources and pass values from one resource to another.

The `Ref` function

The `Ref` function is commonly used to retrieve an identifying property of resources defined within the same CloudFormation template. What it returns depends on the type of resource. For most resources, it returns the physical name of the resource. However, for some resource types, it might return a different value, such as an IP address for an `AWS::EC2::EIP` resource or an Amazon Resource Name (ARN) for an Amazon SNS topic.

The following examples demonstrate how to use the `Ref` function in properties. In each of these examples, the `Ref` function will return the actual name of the `LogicalResourceName` resource declared elsewhere in the template. The `!Ref` syntax example in the YAML example is just a shorter way of writing the `Ref` function.

JSON

```
"Properties" : {
    "PropertyName" : { "Ref" : "LogicalResourceName" }
}
```

YAML

```
Properties:
  PropertyName1:
    Ref: LogicalResourceName
  PropertyName2: !Ref LogicalResourceName
```

For more detailed information about the `Ref` function, see Ref.

The `Fn::GetAtt` function

The `Ref` function is helpful if the parameter or the value returned for a resource is exactly what you want. However, you may need other attributes of a resource. For example, if you want to create a CloudFront distribution with an S3 origin, you need to specify the bucket location by using a DNS-style address. A number of resources have additional attributes whose values you can use in your template. To get these attributes, you use the `Fn::GetAtt` function.

The following examples demonstrate how to use the `GetAtt` function in properties. The `Fn::GetAtt` function takes two parameters, the logical name of the resource and the name of the attribute to be retrieved. The `!GetAtt` syntax example in the YAML example is just a shorter way of writing the `GetAtt` function.

JSON

```
"Properties" : {
    "PropertyName" : {
        "Fn::GetAtt" : [ "LogicalResourceName", "AttributeName" ]
    }
}
```

YAML

```
Properties:
  PropertyName1:
    Fn::GetAtt:
      - LogicalResourceName
      - AttributeName
  PropertyName2: !GetAtt LogicalResourceName.AttributeName
```

For more detailed information about the `GetAtt` function, see Fn::GetAtt.

Examples

The following examples illustrate how to declare resources and how CloudFormation templates can reference other resources defined within the same template and existing AWS resources.

Topics

- Declaring a single resource with a custom name
- Referencing other resources with the Ref function
- Referencing resource attributes using the Fn::GetAtt function

Declaring a single resource with a custom name

The following examples declare a single resource of type `AWS::S3::Bucket` with the logical name `MyBucket`. The `BucketName` property is set to *amzn-s3-demo-bucket*, which should be replaced with the desired name for your S3 bucket.

If you use this resource declaration to create a stack, CloudFormation will create an Amazon S3 bucket with default settings. For other resources, such as an Amazon EC2 instance or Auto Scaling group, CloudFormation requires more information.

JSON

```
{
    "Resources": {
        "MyBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": "amzn-s3-demo-bucket"
            }
        }
    }
}
```

YAML

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
```

Referencing other resources with the `Ref` function

The following examples show a resource declaration that defines an EC2 instance and a security group. The `Ec2Instance` resource references the `InstanceSecurityGroup` resource as part of its `SecurityGroupIds` property using the `Ref` function. It also includes an existing security group

(sg-12a4c434) that's not declared in the template. You use literal strings to refer to existing AWS resources.

JSON

```
{
    "Resources": {
        "Ec2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "SecurityGroupIds": [
                    {
                        "Ref": "InstanceSecurityGroup"
                    },
                    "sg-12a4c434"
                ],
                "KeyName": "MyKey",
                "ImageId": "ami-1234567890abcdef0"
            }
        },
        "InstanceSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Enable SSH access via port 22",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 22,
                        "ToPort": 22,
                        "CidrIp": "0.0.0.0/0"
                    }
                ]
            }
        }
    }
}
```

YAML

```
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
```

```
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
        - sg-12a4c434
      KeyName: MyKey
      ImageId: ami-1234567890abcdef0
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
```

Referencing resource attributes using the `Fn::GetAtt` function

The following examples show a resource declaration that defines a CloudFront distribution resource and an S3 bucket. The `MyDistribution` resource specifies the DNS name of the `MyBucket` resource using `Fn::GetAtt` function to get the bucket's `DomainName` attribute. You'll notice that the `Fn::GetAtt` function lists its two parameters in an array. For functions that take multiple parameters, you use an array to specify their parameters.

JSON

```
{
  "Resources": {
    "MyBucket": {
      "Type": "AWS::S3::Bucket"
    },
    "MyDistribution": {
      "Type": "AWS::CloudFront::Distribution",
      "Properties": {
        "DistributionConfig": {
          "Origins": [
            {
              "DomainName": {
                "Fn::GetAtt": [
                  "MyBucket",
                  "DomainName"
                ]
              },
              "Id": "MyS3Origin",
```

```
              "S3OriginConfig": {}
            }
          ],
          "Enabled": "true",
          "DefaultCacheBehavior": {
            "TargetOriginId": "MyS3Origin",
            "ForwardedValues": {
              "QueryString": "false"
            },
            "ViewerProtocolPolicy": "allow-all"
          }
        }
      }
    }
  }
}
```

YAML

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
  MyDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
          - DomainName: !GetAtt
              - MyBucket
              - DomainName
            Id: MyS3Origin
            S3OriginConfig: {}
        Enabled: 'true'
        DefaultCacheBehavior:
          TargetOriginId: MyS3Origin
          ForwardedValues:
            QueryString: 'false'
          ViewerProtocolPolicy: allow-all
```

CloudFormation template Parameters syntax

Use the optional `Parameters` section to customize your templates. With parameters, you can input custom values to your template each time you create or update a stack. By using parameters

in your templates, you can build reusable and flexible templates that can be tailored to specific scenarios.

By defining parameters of the appropriate type, you can choose from a list of identifiers of existing resources when you use the console to create your stack. For more information, see Specify existing resources at runtime with CloudFormation-supplied parameter types.

Parameters are a popular way to specify property values of stack resources. However, there may be settings that are region dependent or are somewhat complex for users to figure out because of other conditions or dependencies. In these cases, you might want to put some logic in the template itself so that users can specify simpler values (or none at all) to get the results that they want, such as by using a mapping. For more information, see CloudFormation template Mappings syntax.

Syntax

You declare parameters in a template's `Parameters` section, which uses the following general syntax:

JSON

```
"Parameters" : {
  "ParameterLogicalID" : {
    "Description": "Information about the parameter",
    "Type" : "DataType",
    "Default" : "value",
    "AllowedValues" : ["value1", "value2"]
  }
}
```

YAML

```
Parameters:
  ParameterLogicalID:
    Description: Information about the parameter
    Type: DataType
    Default: value
    AllowedValues:
      - value1
      - value2
```

A parameter contains a list of attributes that define its value and constraints against its value. The only required attribute is `Type`, which can be `String`, `Number`, or a CloudFormation-supplied parameter type. You can also add a `Description` attribute that describes what kind of value you should specify. The parameter's name and description appear in the **Specify Parameters** page when you use the template in the **Create Stack** wizard.

> **Note**
>
> By default, the CloudFormation console lists input parameters alphabetically by their logical ID. To override this default ordering and group related parameters together, you can use the `AWS::CloudFormation::Interface` metadata key in your template. For more information, see Organizing CloudFormation parameters with `AWS::CloudFormation::Interface` metadata.

For parameters with default values, CloudFormation uses the default values unless users specify another value. If you omit the default attribute, users are required to specify a value for that parameter. However, requiring the user to input a value does not ensure that the value is valid. To validate the value of a parameter, you can declare constraints or specify an AWS-specific parameter type.

For parameters without default values, users must specify a key name value at stack creation. If they don't, CloudFormation fails to create the stack and throws an exception:

```
Parameters: [KeyName] must have values
```

Properties

`AllowedPattern`

A regular expression that represents the patterns to allow for `String` or `CommaDelimitedList` types. When applied on a parameter of type `String`, the pattern must match the entire parameter value provided. When applied to a parameter of type `CommaDelimitedList`, the pattern must match each value in the list.

Required: No

`AllowedValues`

An array containing the list of values allowed for the parameter. When applied to a parameter of type `String`, the parameter value must be one of the allowed values. When applied to a parameter of type `CommaDelimitedList`, each value in the list must be one of the specified allowed values.

Required: No

> **Note**
>
> If you're using YAML and you want to use `Yes` and `No` strings for `AllowedValues`, use single-quotes to prevent the YAML parser from considering these boolean values.

`ConstraintDescription`

A string that explains a constraint when the constraint is violated. For example, without a constraint description, a parameter that has an allowed pattern of `[A-Za-z0-9]+` displays the following error message when the user specifies an invalid value:

```
Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+
```

By adding a constraint description, such as *must only contain letters (uppercase and lowercase) and numbers*, you can display the following customized error message:

```
Malformed input-Parameter MyParameter must only contain uppercase and
lowercase letters and numbers
```

Required: No

`Default`

A value of the appropriate type for the template to use if no value is specified when a stack is created. If you define constraints for the parameter, you must specify a value that adheres to those constraints.

Required: No

`Description`

A string of up to 4000 characters that describes the parameter.

Required: No

`MaxLength`

An integer value that determines the largest number of characters you want to allow for `String` types.

Required: No

`MaxValue`

A numeric value that determines the largest numeric value you want to allow for `Number` types.

Required: No

`MinLength`

An integer value that determines the smallest number of characters you want to allow for `String` types.

Required: No

`MinValue`

A numeric value that determines the smallest numeric value you want to allow for `Number` types.

Required: No

`NoEcho`

Whether to mask the parameter value to prevent it from being displayed in the console, command line tools, or API. If you set the `NoEcho` attribute to `true`, CloudFormation returns the parameter value masked as asterisks (*****) for any calls that describe the stack or stack events, except for information stored in the locations specified below.

Required: No

Important

Using the `NoEcho` attribute does not mask any information stored in the following:

- The `Metadata` template section. CloudFormation does not transform, modify, or redact any information you include in the `Metadata` section. For more information, see Metadata.

- The `Outputs` template section. For more information, see Outputs.
- The `Metadata` attribute of a resource definition. For more information, see `Metadata` attribute.

We strongly recommend you do not use these mechanisms to include sensitive information, such as passwords or secrets.

Important

Rather than embedding sensitive information directly in your CloudFormation templates, we recommend you use dynamic parameters in the stack template to reference sensitive information that is stored and managed outside of CloudFormation, such as in the AWS Systems Manager Parameter Store or AWS Secrets Manager.
For more information, see the Do not embed credentials in your templates best practice.

Important

We strongly recommend against including `NoEcho` parameters, or any sensitive data, in resource properties that are part of a resource's primary identifier.
When a `NoEcho` parameter is included in a property that forms a primary resource identifier, CloudFormation may use the *actual plaintext value* in the primary resource identifier. This resource ID may appear in any derived outputs or destinations.
To determine which resource properties comprise a resource type's primary identifier, refer to the resource reference documentation for that resource in the AWS resource and property types reference. In the **Return values** section, the `Ref` function return value represents the resource properties that comprise the resource type's primary identifier.

Type

The data type for the parameter (`DataType`).

Required: Yes

CloudFormation supports the following parameter types:

`String`

A literal string. You can use the following attributes to declare constraints: `MinLength`, `MaxLength`, `Default`, `AllowedValues`, and `AllowedPattern`.

For example, users could specify `"MyUserName"`.

`Number`

An integer or float. CloudFormation validates the parameter value as a number; however, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a string.

You can use the following attributes to declare constraints: `MinValue`, `MaxValue`, `Default`, and `AllowedValues`.

For example, users could specify `"8888"`.

`List<Number>`

An array of integers or floats that are separated by commas. CloudFormation validates the parameter value as numbers; however, when you use the parameter elsewhere in your template (for example, by using the `Ref` intrinsic function), the parameter value becomes a list of strings.

For example, users could specify `"80,20"`, and a `Ref` would result in `["80","20"]`.

`CommaDelimitedList`

An array of literal strings that are separated by commas. The total number of strings should be one more than the total number of commas. Also, each member string is space trimmed.

For example, users could specify `"test,dev,prod"`, and a `Ref` would result in `["test","dev","prod"]`.

AWS-specific parameter types

AWS values such as Amazon EC2 key pair names and VPC IDs. For more information, see Specify existing resources at runtime.

Systems Manager parameter types

Parameters that correspond to existing parameters in Systems Manager Parameter Store. You specify a Systems Manager parameter key as the value of the Systems Manager

parameter type, and CloudFormation retrieves the latest value from Parameter Store to use for the stack. For more information, see Specify existing resources at runtime.

General requirements for parameters

The following requirements apply when using parameters:

- You can have a maximum of 200 parameters in a CloudFormation template.
- Each parameter must be given a logical name (also called logical ID) that must be alphanumeric and unique among all logical names within the template.
- Each parameter must be assigned a parameter type that's supported by CloudFormation. For more information, see Type.
- Each parameter must be assigned a value at runtime for CloudFormation to successfully provision the stack. You can optionally specify a default value for CloudFormation to use unless another value is provided.
- Parameters must be declared and referenced from within the same template. You can reference parameters from the `Resources` and `Outputs` sections of the template.

Examples

Topics

- Simple string parameter
- Password parameter
- Referencing parameters
- Comma-delimited list parameter
- Return a value from a comma-delimited list parameter

Simple string parameter

The following example declares a parameter named `InstanceTypeParameter` of type `String`. This parameter lets you specify the Amazon EC2 instance type for the stack. If no value is provided during stack creation or update, CloudFormation uses the default value of `t2.micro`.

JSON

```
"Parameters" : {
```

```
  "InstanceTypeParameter" : {
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro.",
    "Type" : "String",
    "Default" : "t2.micro",
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"]
  }
}
```

YAML

```
Parameters:
  InstanceTypeParameter:
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
```

Password parameter

The following example declares a parameter named `DBPwd` of type `String` with no default value. The `NoEcho` property is set to `true` to prevent the parameter value from being displayed in stack descriptions. The minimum length that can be specified is `1`, and the maximum length that can be specified is `41`. The pattern allows lowercase and uppercase alphabetical characters and numerals. This example also illustrates the use of a regular expression for the `AllowedPattern` property.

JSON

```
"Parameters" : {
  "DBPwd" : {
    "NoEcho" : "true",
    "Description" : "The database admin account password",
    "Type" : "String",
    "MinLength" : "1",
    "MaxLength" : "41",
    "AllowedPattern" : "^[a-zA-Z0-9]*$"
  }
}
```

YAML

```
Parameters:
  DBPwd:
    NoEcho: true
    Description: The database admin account password
    Type: String
    MinLength: 1
    MaxLength: 41
    AllowedPattern: ^[a-zA-Z0-9]*$
```

Referencing parameters

You use the `Ref` intrinsic function to reference a parameter, and CloudFormation uses the parameter's value to provision the stack. You can reference parameters from the `Resources` and `Outputs` sections of the same template.

In the following example, the `InstanceType` property of the EC2 instance resource references the `InstanceTypeParameter` parameter value:

JSON

```
"Ec2Instance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "InstanceType" : { "Ref" : "InstanceTypeParameter" },
    "ImageId" : "ami-0ff8a91507f77f867"
  }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType:
      Ref: InstanceTypeParameter
    ImageId: ami-0ff8a91507f77f867
```

Comma-delimited list parameter

The `CommaDelimitedList` parameter type can be useful when you need to provide multiple values for a single property. The following example declares a parameter named `DbSubnetIpBlocks` with a default value of three CIDR blocks separated by commas.

JSON

```
"Parameters" : {
  "DbSubnetIpBlocks": {
    "Description": "Comma-delimited list of three CIDR blocks",
    "Type": "CommaDelimitedList",
    "Default": "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
  }
}
```

YAML

```
Parameters:
  DbSubnetIpBlocks:
    Description: "Comma-delimited list of three CIDR blocks"
    Type: CommaDelimitedList
    Default: "10.0.48.0/24, 10.0.112.0/24, 10.0.176.0/24"
```

Return a value from a comma-delimited list parameter

To refer to a specific value in a parameter's comma-delimited list, use the `Fn::Select` intrinsic function in the `Resources` section of your template. Pass the index value of the object that you want and a list of objects, as shown in the following example.

JSON

```
{
    "Parameters": {
        "VPC": {
            "Type": "String",
            "Default": "vpc-123456"
        },
        "VpcAzs": {
            "Type": "CommaDelimitedList",
            "Default": "us-west-2a, us-west-2b, us-west-2c"
        },
```

```
        "DbSubnetIpBlocks": {
            "Type": "CommaDelimitedList",
            "Default": "172.16.0.0/26, 172.16.0.64/26, 172.16.0.128/26"
        }
    },
    "Resources": {
        "DbSubnet1": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "AvailabilityZone": {
                    "Fn::Select": [
                      0,
                      {
                        "Ref": "VpcAzs"
                      }
                   ]
                },
                "VpcId": {
                    "Ref": "VPC"
                },
                "CidrBlock": {
                    "Fn::Select": [
                        0,
                        { "Ref": "DbSubnetIpBlocks" }
                    ]
                }
            }
        },
        "DbSubnet2": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "AvailabilityZone": {
                    "Fn::Sub": [
                        "${AWS::Region}${AZ}",
                        {
                            "AZ": {
                                "Fn::Select": [
                                    1,
                                    { "Ref": "VpcAzs" }
                                ]
                            }
                        }
                    ]
                },
```

```
                "VpcId": {
                    "Ref": "VPC"
                },
                "CidrBlock": {
                    "Fn::Select": [
                        1,
                        { "Ref": "DbSubnetIpBlocks" }
                    ]
                }
            }
        },
        "DbSubnet3": {
            "Type": "AWS::EC2::Subnet",
            "Properties": {
                "AvailabilityZone": {
                    "Fn::Sub": [
                        "${AWS::Region}${AZ}",
                        {
                            "AZ": {
                                "Fn::Select": [
                                    2,
                                    { "Ref": "VpcAzs" }
                                ]
                            }
                        }
                    ]
                },
                "VpcId": {
                    "Ref": "VPC"
                },
                "CidrBlock": {
                    "Fn::Select": [
                        2,
                        { "Ref": "DbSubnetIpBlocks" }
                    ]
                }
            }
        }
    }
}
```

YAML

```
Parameters:
  VPC:
    Type: String
    Default: vpc-123456
  VpcAzs:
    Type: CommaDelimitedList
    Default: us-west-2a, us-west-2b, us-west-2c
  DbSubnetIpBlocks:
    Type: CommaDelimitedList
    Default: 172.16.0.0/26, 172.16.0.64/26, 172.16.0.128/26
Resources:
  DbSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Select
        - 0
        - !Ref VpcAzs
      VpcId: !Ref VPC
      CidrBlock: !Select
        - 0
        - !Ref DbSubnetIpBlocks
  DbSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Sub
        - ${AWS::Region}${AZ}
        - AZ: !Select
            - 1
            - !Ref VpcAzs
      VpcId: !Ref VPC
      CidrBlock: !Select
        - 1
        - !Ref DbSubnetIpBlocks
  DbSubnet3:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Sub
        - ${AWS::Region}${AZ}
        - AZ: !Select
            - 2
            - !Ref VpcAzs
      VpcId: !Ref VPC
```

```
      CidrBlock: !Select
        - 2
        - !Ref DbSubnetIpBlocks
```

Related resources

CloudFormation also supports the use of dynamic references to specify property values dynamically. For example, you might need to reference secure strings stored in Systems Manager Parameter Store. For more information, see Get values stored in other services using dynamic references.

You can also use pseudo parameters within a `Ref` or a Sub function to dynamically populate values. For more information, see Get AWS values using pseudo parameters.

CloudFormation template Outputs syntax

The optional `Outputs` section declares output values for the stack. These output values can be used in various ways:

- **Capture important details about your resources** – An output is a convenient way to capture important information about your resources. For example, you can output the S3 bucket name for a stack to make the bucket easier to find. You can view output values in the **Outputs** tab of the CloudFormation console or by using the describe-stacks CLI command.
- **Cross-stack references** – You can import output values into other stacks to create references between stacks. This is helpful when you need to share resources or configurations across multiple stacks.
- **Cross-account and cross-Region references** – You can reference output values from stacks in other AWS accounts or Regions using the `Fn::GetStackOutput` function, without requiring explicit export declarations. For more information, see Fn::GetStackOutput in the *CloudFormation Template Reference Guide*.

> **Important**
>
> CloudFormation doesn't redact or obfuscate any information you include in the `Outputs` section. We strongly recommend you don't use this section to output sensitive information, such as passwords or secrets.
> Output values are available after the stack operation is complete. Stack output values aren't available when a stack status is in any of the `IN_PROGRESS` statuses. We don't

recommend establishing dependencies between a service runtime and the stack output value because output values might not be available at all times.

Syntax

The `Outputs` section consists of the key name `Outputs`. You can declare a maximum of 200 outputs in a template.

The following example demonstrates the structure of the `Outputs` section.

JSON

Use braces to enclose all output declarations. Delimit multiple outputs with commas.

```
"Outputs" : {
  "OutputLogicalID" : {
    "Description" : "Information about the value",
    "Value" : "Value to return",
    "Export" : {
      "Name" : "Name of resource to export"
    }
  }
}
```

YAML

```
Outputs:
  OutputLogicalID:
    Description: Information about the value
    Value: Value to return
    Export:
      Name: Name of resource to export
```

Output fields

The `Outputs` section can include the following fields.

Logical ID (also called *logical name*)

An identifier for the current output. The logical ID must be alphanumeric (`a-z`, `A-Z`, `0-9`) and unique within the template.

`Description` (optional)

A `String` type that describes the output value. The value for the description declaration must be a literal string that's between 0 and 1024 bytes in length. You can't use a parameter or function to specify the description.

`Value` (required)

The value of the property returned by the describe-stacks command. The value of an output can include literals, parameter references, pseudo parameters, a mapping value, or intrinsic functions.

`Export` (optional)

The name of the resource output to be exported for a cross-stack reference.

You can use intrinsic functions to customize the `Name` value of an export.

For more information, see Get exported outputs from a deployed CloudFormation stack.

To associate a condition with an output, define the condition in the Conditions section of the template.

Examples

The following examples illustrate how stack output works.

Topics

- Stack output
- Customize export name using Fn::Sub
- Customize export name using Fn::Join
- Return a URL constructed using Fn::Join

Stack output

In the following example, the output named `BackupLoadBalancerDNSName` returns the DNS name for the resource with the logical ID `BackupLoadBalancer` only when the `CreateProdResources` condition is true. The output named `InstanceID` returns the ID of the EC2 instance with the logical ID `EC2Instance`.

JSON

```
"Outputs" : {
  "BackupLoadBalancerDNSName" : {
    "Description": "The DNSName of the backup load balancer",
    "Value" : { "Fn::GetAtt" : [ "BackupLoadBalancer", "DNSName" ]},
    "Condition" : "CreateProdResources"
  },
  "InstanceID" : {
    "Description": "The Instance ID",
    "Value" : { "Ref" : "EC2Instance" }
  }
}
```

YAML

```
Outputs:
  BackupLoadBalancerDNSName:
    Description: The DNSName of the backup load balancer
    Value: !GetAtt BackupLoadBalancer.DNSName
    Condition: CreateProdResources
  InstanceID:
    Description: The Instance ID
    Value: !Ref EC2Instance
```

Customize export name using Fn::Sub

In the following examples, the output named `StackVPC` returns the ID of a VPC, and then exports the value for cross-stack referencing with the name `VPCID` appended to the stack's name.

JSON

```
"Outputs" : {
  "StackVPC" : {
    "Description" : "The ID of the VPC",
    "Value" : { "Ref" : "MyVPC" },
    "Export" : {
      "Name" : {"Fn::Sub": "${AWS::StackName}-VPCID" }
    }
  }
}
```

YAML

```
Outputs:
  StackVPC:
    Description: The ID of the VPC
    Value: !Ref MyVPC
    Export:
      Name: !Sub "${AWS::StackName}-VPCID"
```

For more information about the `Fn::Sub` function, see Fn::Sub.

Customize export name using `Fn::Join`

You can also use the `Fn::Join` function to construct values based on parameters, resource attributes, and other strings.

The following examples use the `Fn::Join` function to customize the export name instead of the `Fn::Sub` function. The example `Fn::Join` function concatenates the stack name with the name `VPCID` using a colon as a separator.

JSON

```
"Outputs" : {
  "StackVPC" : {
    "Description" : "The ID of the VPC",
    "Value" : { "Ref" : "MyVPC" },
    "Export" : {
      "Name" : { "Fn::Join" : [ ":", [ { "Ref" : "AWS::StackName" }, "VPCID" ] ] }
    }
  }
}
```

YAML

```
Outputs:
  StackVPC:
    Description: The ID of the VPC
    Value: !Ref MyVPC
    Export:
      Name: !Join [ ":", [ !Ref "AWS::StackName", VPCID ] ]
```

For more information about the `Fn::Join` function, see Fn::Join.

Return a URL constructed using `Fn::Join`

In the following example for a template that creates a WordPress site, `InstallURL` is the string returned by a `Fn::Join` function call that concatenates `http://`, the DNS name of the resource `ElasticLoadBalancer`, and `/wp-admin/install.php`. The output value would be similar to the following:

```
http://mywptests-elasticl-1gb51l6sl8y5v-206169572.aws-region.elb.amazonaws.com/wp-admin/install.php
```

JSON

```
{
    "Outputs": {
        "InstallURL": {
            "Value": {
                "Fn::Join": [
                    "",
                    [
                        "http://",
                        {
                            "Fn::GetAtt": [
                                "ElasticLoadBalancer",
                                "DNSName"
                            ]
                        },
                        "/wp-admin/install.php"
                    ]
                ]
            },
            "Description": "Installation URL of the WordPress website"
        }
    }
}
```

YAML

```
Outputs:
  InstallURL:
```

```
    Value: !Join
      - ''
      - - 'http://'
        - !GetAtt
          - ElasticLoadBalancer
          - DNSName
        - /wp-admin/install.php
    Description: Installation URL of the WordPress website
```

For more information about the `Fn::Join` function, see Fn::Join.

CloudFormation template Mappings syntax

The optional `Mappings` section helps you create key-value pairs that can be used to specify values based on certain conditions or dependencies.

One common use case for the `Mappings` section is to set values based on the AWS Region where the stack is deployed. This can be achieved by using the `AWS::Region` pseudo parameter. The `AWS::Region` pseudo parameter is a value that CloudFormation resolves to the region where the stack is created. Pseudo parameters are resolved by CloudFormation when you create the stack.

To retrieve values in a map, you can use the `Fn::FindInMap` intrinsic function within the `Resources` section of your template.

Syntax

The `Mappings` section uses the following syntax:

JSON

```
"Mappings" : {
  "MappingLogicalName" : {
    "Key1" : {
      "Name" : "Value1"
    },
    "Key2" : {
      "Name" : "Value2"
    },
    "Key3" : {
      "Name" : "Value3"
    }
```

```
  }
}
```

YAML

```
Mappings:
  MappingLogicalName:
    Key1:
      Name: Value1
    Key2:
      Name: Value2
    Key3:
      Name: Value3
```

- `MappingLogicalName` is the logical name for the mapping.
- Within the mapping, each map is a key followed by another mapping.
- The key must be a map of name-value pairs and unique within the mapping.
- The name-value pair is a label, and the value to map. By naming the values, you can map more than one set of values to a key.
- The keys in mappings must be literal strings.
- The values can be of type `String` or `List`.

> **Note**
>
> You can't include parameters, pseudo parameters, or intrinsic functions in the `Mappings` section.
> If the values in a mapping aren't currently used by your stack, you cannot update the mapping alone. You must include changes that add, modify, or delete resources.

Examples

Topics

- Basic mapping
- Mapping with multiple values
- Return a value from a mapping

- Input parameter and Fn::FindInMap

Basic mapping

The following example shows a `Mappings` section with a map `RegionToInstanceType`, which contains five keys that map to name-value pairs containing single string values. The keys are region names. Each name-value pair is an instance type from the T family that's available in the region represented by the key. The name-value pairs have a name (`InstanceType` in the example) and a value.

JSON

```
"Mappings" : {
  "RegionToInstanceType" : {
    "us-east-1"      : { "InstanceType" : "t2.micro" },
    "us-west-1"      : { "InstanceType" : "t2.micro" },
    "eu-west-1"      : { "InstanceType" : "t2.micro" },
    "eu-north-1"     : { "InstanceType" : "t3.micro" },
    "me-south-1"     : { "InstanceType" : "t3.micro" }
  }
}
```

YAML

```
Mappings:
  RegionToInstanceType:
    us-east-1:
      InstanceType: t2.micro
    us-west-1:
      InstanceType: t2.micro
    eu-west-1:
      InstanceType: t2.micro
    eu-north-1:
      InstanceType: t3.micro
    me-south-1:
      InstanceType: t3.micro
```

Mapping with multiple values

The following example has region keys that are mapped to two sets of values: one named `MyAMI1` and the other `MyAMI2`.

> **Note**
>
> The AMI IDs shown in these examples are placeholders for demonstration purposes. Whenever possible, consider using dynamic references to AWS Systems Manager parameters as an alternative to the `Mappings` section. To avoid updating all your templates with a new ID each time the AMI that you want to use changes, use a AWS Systems Manager parameter to retrieve the latest AMI ID when the stack is created or updated. The latest versions of commonly used AMIs are also available as public parameters in Systems Manager. For more information, see Get values stored in other services using dynamic references.

JSON

```
"Mappings" : {
  "RegionToAMI" : {
    "us-east-1"        : { "MyAMI1" : "ami-12345678901234567", "MyAMI2" :
 "ami-23456789012345678" },
    "us-west-1"        : { "MyAMI1" : "ami-34567890123456789", "MyAMI2" :
 "ami-45678901234567890" },
    "eu-west-1"        : { "MyAMI1" : "ami-56789012345678901", "MyAMI2" :
 "ami-67890123456789012" },
    "ap-southeast-1"   : { "MyAMI1" : "ami-78901234567890123", "MyAMI2" :
 "ami-89012345678901234" },
    "ap-northeast-1"   : { "MyAMI1" : "ami-90123456789012345", "MyAMI2" :
 "ami-01234567890123456" }
  }
}
```

YAML

```
Mappings:
  RegionToAMI:
    us-east-1:
      MyAMI1: ami-12345678901234567
      MyAMI2: ami-23456789012345678
    us-west-1:
      MyAMI1: ami-34567890123456789
      MyAMI2: ami-45678901234567890
    eu-west-1:
      MyAMI1: ami-56789012345678901
```

```
      MyAMI2: ami-67890123456789012
    ap-southeast-1:
      MyAMI1: ami-78901234567890123
      MyAMI2: ami-89012345678901234
    ap-northeast-1:
      MyAMI1: ami-90123456789012345
      MyAMI2: ami-01234567890123456
```

Return a value from a mapping

You can use the `Fn::FindInMap` function to return a named value based on a specified key. The following example template contains an Amazon EC2 resource whose `InstanceType` property is assigned by the `FindInMap` function. The `FindInMap` function specifies key as the AWS Region where the stack is created (using the `AWS::Region` pseudo parameter) and `InstanceType` as the name of the value to map to. The `ImageId` uses a Systems Manager parameter to dynamically retrieve the latest Amazon Linux 2 AMI. For more information about pseudo parameters, see Get AWS values using pseudo parameters.

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Mappings" : {
    "RegionToInstanceType" : {
      "us-east-1"      : { "InstanceType" : "t2.micro" },
      "us-west-1"      : { "InstanceType" : "t2.micro" },
      "eu-west-1"      : { "InstanceType" : "t2.micro" },
      "eu-north-1"     : { "InstanceType" : "t3.micro" },
      "me-south-1"     : { "InstanceType" : "t3.micro" }
    }
  },
  "Resources" : {
    "myEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
        "InstanceType" : { "Fn::FindInMap" : [ "RegionToInstanceType", { "Ref" :
 "AWS::Region" }, "InstanceType" ]}
      }
    }
  }
```

```
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Mappings:
  RegionToInstanceType:
    us-east-1:
      InstanceType: t2.micro
    us-west-1:
      InstanceType: t2.micro
    eu-west-1:
      InstanceType: t2.micro
    eu-north-1:
      InstanceType: t3.micro
    me-south-1:
      InstanceType: t3.micro
Resources:
  myEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      InstanceType: !FindInMap [RegionToInstanceType, !Ref 'AWS::Region', InstanceType]
```

Input parameter and `Fn::FindInMap`

The following example template shows how to create an EC2 instance using multiple mappings. The template uses nested mappings to automatically select the appropriate instance type and security group based on the target AWS Region and environment type (`Dev` or `Prod`). It also uses a Systems Manager parameter to dynamically retrieve the latest Amazon Linux 2 AMI.

JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Parameters" : {
    "EnvironmentType" : {
      "Description" : "The environment type (Dev or Prod)",
      "Type" : "String",
      "Default" : "Dev",
      "AllowedValues" : [ "Dev", "Prod" ]
```

```
    }
  },
  "Mappings" : {
    "RegionAndEnvironmentToInstanceType" : {
      "us-east-1"        : { "Dev" : "t3.micro", "Prod" : "c5.large" },
      "us-west-1"        : { "Dev" : "t2.micro", "Prod" : "m5.large" }
    },
    "RegionAndEnvironmentToSecurityGroup" : {
      "us-east-1"        : { "Dev" : "sg-12345678", "Prod" : "sg-abcdef01" },
      "us-west-1"        : { "Dev" : "sg-ghijkl23", "Prod" : "sg-45678abc" }
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
        "InstanceType" : { "Fn::FindInMap": [ "RegionAndEnvironmentToInstanceType",
 { "Ref": "AWS::Region" }, { "Ref": "EnvironmentType" } ]},
        "SecurityGroupIds" : [{ "Fn::FindInMap" :
 [ "RegionAndEnvironmentToSecurityGroup", { "Ref" : "AWS::Region" }, { "Ref" :
 "EnvironmentType" } ]}]
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvironmentType:
    Description: The environment type (Dev or Prod)
    Type: String
    Default: Dev
    AllowedValues:
      - Dev
      - Prod
Mappings:
  RegionAndEnvironmentToInstanceType:
    us-east-1:
      Dev: t3.micro
```

```
      Prod: c5.large
    us-west-1:
      Dev: t2.micro
      Prod: m5.large
  RegionAndEnvironmentToSecurityGroup:
    us-east-1:
      Dev: sg-12345678
      Prod: sg-abcdef01
    us-west-1:
      Dev: sg-ghijkl23
      Prod: sg-45678abc
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      InstanceType: !FindInMap [RegionAndEnvironmentToInstanceType, !Ref
 'AWS::Region', !Ref EnvironmentType]
      SecurityGroupIds:
        - !FindInMap [RegionAndEnvironmentToSecurityGroup, !Ref 'AWS::Region', !Ref
 EnvironmentType]
```

Related resources

These related topics can be helpful as you develop templates that use the `Fn::FindInMap` function.

- Fn::FindInMap
- Fn::FindInMap enhancements
- Fn::Sub

CloudFormation template Metadata syntax

`Metadata` stores additional information using JSON or YAML objects. The types of template-level metadata you can use in your template include:

Custom metadata

Stores user-defined key-value pairs. For example, you can provide additional information that doesn't impact resource creation but offers additional context about the infrastructure, team, or deployment specifics.

`AWS::CloudFormation::Interface`

Defines the grouping and ordering of input parameters when they're displayed in the CloudFormation console. By default, the CloudFormation console alphabetically sorts parameters by their logical ID.

`AWS::CloudFormation::Designer`

CloudFormation Designer (Designer) reached end of life on February 5, 2025.

Important

During a stack update, you cannot update the `Metadata` section by itself. You can update it only when you include changes that add, modify, or delete resources.
CloudFormation does not transform, modify, or redact any information you include in the `Metadata` section. Because of this, we strongly recommend you do not use this section to store sensitive information, such as passwords or secrets.

Syntax

To declare custom metadata in your CloudFormation template, use the following syntax:

JSON

```
"Metadata" : {
  "Instances" : {"Description" : "Information about the instances"},
  "Databases" : {"Description" : "Information about the databases"}
}
```

YAML

```
Metadata:
  Instances:
    Description: "Information about the instances"
```

```
  Databases:
    Description: "Information about the databases"
```

For the syntax for the `AWS::CloudFormation::Interface`, see Organizing CloudFormation parameters with `AWS::CloudFormation::Interface` metadata.

Organizing CloudFormation parameters with `AWS::CloudFormation::Interface` metadata

`AWS::CloudFormation::Interface` is a metadata key that defines how parameters are grouped and sorted in the CloudFormation console. By default, the console lists input parameters in alphabetical order by their logical IDs when you create or update stacks in the console. By using this key, you can define your own parameter grouping and ordering so that users can efficiently specify parameter values. For example, you could group all EC2-related parameters in one group and all VPC-related parameters in another group.

In the metadata key, you can specify the groups to create, the parameters to include in each group, and the order in which the console shows each parameter within its group.

You can also define labels for parameters. A label is a friendly name or description that the console displays instead of a parameter's logical ID. Labels are useful for helping users understand the values to specify for each parameter. For example, you could label a `KeyPair` parameter `Select an EC2 key pair`.

All parameters that you reference in the metadata key must be declared in the `Parameters` section of the template.

> **Note**
>
> Only the CloudFormation console uses the `AWS::CloudFormation::Interface` metadata key. AWS CLI and API calls don't use this key.

Syntax

To declare this entity in your CloudFormation template, use the following syntax:

JSON

```
"Metadata" : {
```

```
  "AWS::CloudFormation::Interface" : {
    "ParameterGroups": [
      {
        "Label": {
          "default": "Group Label"
        },
        "Parameters": [
          "Parameter1",
          "Parameter2"
        ]
      }
    ],
    "ParameterLabels": {
      "Parameter1": {
        "default": "Friendly Name for Parameter1"
      }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: Group Label
        Parameters:
          - Parameter1
          - Parameter2
    ParameterLabels:
      Parameter1:
        default: Friendly Name for Parameter1
```

Properties

`ParameterGroups`

A list of parameter group types, where you specify group names, the parameters in each group, and the order in which the parameters are shown.

Required: No

`Label`

A name for the parameter group.

Required: No

`default`

The default label that the CloudFormation console uses to name a parameter group.

Required: No

Type: String

`Parameters`

A list of case-sensitive parameter logical IDs to include in the group. Parameters must already be defined in the `Parameters` section of the template. A parameter can be included in only one parameter group.

The console lists the parameters that you don't associate with a parameter group in alphabetical order in the `Other parameters` group.

Required: No

Type: List of String values

`ParameterLabels`

A mapping of parameters and their friendly names that the CloudFormation console shows when a stack is created or updated.

Required: No

Parameter label

A label for a parameter. The label defines a friendly name or description that the CloudFormation console shows on the **Specify Parameters** page when a stack is created or updated. The parameter label must be the case-sensitive logical ID of a valid parameter that has been declared in the `Parameters` section of the template.

Required: No

`default`

The default label that the CloudFormation console uses to name a parameter.

Required: No

Type: String

Example

The following example defines two parameter groups: `Network Configuration` and `Amazon EC2 Configuration`. The `Network Configuration` group includes the `VPCID`, `SubnetId`, and `SecurityGroupID` parameters, which are defined in the `Parameters` section of the template (not shown). The order in which the console shows these parameters is defined by the order in which the parameters are listed, starting with the `VPCID` parameter. The example similarly groups and orders the `Amazon EC2 Configuration` parameters.

The example also defines a label for the `VPCID` parameter. The console will show **Which VPC should this be deployed to?** instead of the parameter's logical ID (`VPCID`).

JSON

```
"Metadata" : {
  "AWS::CloudFormation::Interface" : {
    "ParameterGroups" : [
      {
        "Label" : { "default" : "Network Configuration" },
        "Parameters" : [ "VPCID", "SubnetId", "SecurityGroupID" ]
      },
      {
        "Label" : { "default":"Amazon EC2 Configuration" },
        "Parameters" : [ "InstanceType", "KeyName" ]
      }
    ],
    "ParameterLabels" : {
      "VPCID" : { "default" : "Which VPC should this be deployed to?" }
    }
  }
}
```

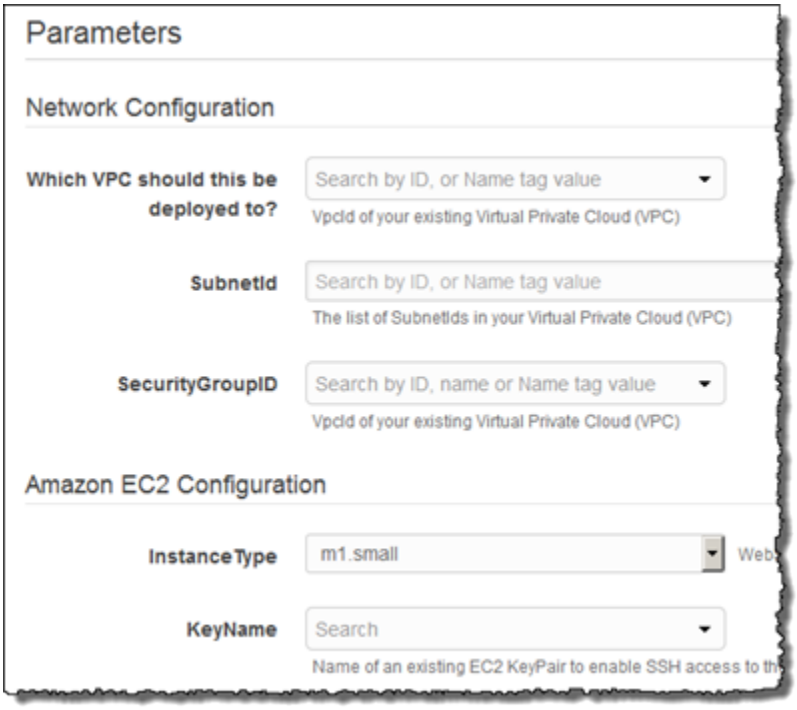
YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
```

```
          default: "Network Configuration"
        Parameters:
          - VPCID
          - SubnetId
          - SecurityGroupID
      - Label:
          default: "Amazon EC2 Configuration"
        Parameters:
          - InstanceType
          - KeyName
    ParameterLabels:
      VPCID:
        default: "Which VPC should this be deployed to?"
```

Parameter groups in the console

Using the metadata key from this example, the following figure shows how the console displays parameter groups when a stack is created or updated: **Parameter groups in the console**



CloudFormation template Rules syntax

The `Rules` section is an optional part of a CloudFormation template that enables custom validation logic. When included, this section contains rule functions that validate parameter values before CloudFormation creates or updates any resources.

Rules are useful when standard parameter constraints are insufficient. For example, when SSL is enabled, both a certificate and domain name must be provided. A rule can ensure that these dependencies are met.

Syntax

The `Rules` section uses the following syntax:

JSON

The `Rules` section of a template consists of the key name `Rules`, followed by a single colon. You must use braces to enclose all rule declarations. If you declare multiple rules, they're delimited by commas. For each rule, you declare a logical name in quotation marks followed by a colon and braces that enclose the rule condition and assertions.

```
{
    "Rules": {
        "LogicalRuleName1": {
            "RuleCondition": {
                "rule-specific intrinsic function": "Value"
            },
            "Assertions": [
                {
                    "Assert": {
                        "rule-specific intrinsic function": "Value"
                    },
                    "AssertDescription": "Information about this assert"
                },
                {
                    "Assert": {
                        "rule-specific intrinsic function": "Value"
                    },
                    "AssertDescription": "Information about this assert"
                }
            ]
        },
        "LogicalRuleName2": {
            "Assertions": [
                {
                    "Assert": {
                        "rule-specific intrinsic function": "Value"
                    },
                    "AssertDescription": "Information about this assert"
```

```
                }
            ]
        }
    }
}
```

YAML

```
Rules:
  LogicalRuleName1:
    RuleCondition:
      rule-specific intrinsic function: Value
    Assertions:
      - Assert:
          rule-specific intrinsic function: Value
        AssertDescription: Information about this assert
      - Assert:
          rule-specific intrinsic function: Value
        AssertDescription: Information about this assert
  LogicalRuleName2:
    Assertions:
      - Assert:
          rule-specific intrinsic function: Value
        AssertDescription: Information about this assert
```

Rules fields

The `Rules` section can include the following fields.

Logical ID (also called *logical name*)

A unique identifier for each rule.

`RuleCondition` (optional)

A property that determines when a rule takes effect. If you don't define a rule condition, the rule's assertions always take effect. For each rule, you can define only one rule condition.

`Assertions` (required)

One or more statements that specify the acceptable values for a particular parameter.

`Assert`

A condition that must evaluate to `true`.

`AssertDescription`

A message displayed when an assertion fails.

Rule-specific intrinsic functions

To define your rules, you must use *rule-specific functions*, which are functions that can only be used in the `Rules` section of a template. While these functions can be nested, the final result of a rule condition or assertion must be either `true` or `false`.

The following rule functions are available:

- Fn::And
- Fn::Contains
- Fn::EachMemberEquals
- Fn::EachMemberIn
- Fn::Equals
- Fn::If
- Fn::Not
- Fn::Or
- Fn::RefAll
- Fn::ValueOf
- Fn::ValueOfAll

These functions are used in the condition or assertions of a rule. The condition property determines if CloudFormation applies the assertions. If the condition evaluates to `true`, CloudFormation evaluates the assertions to verify whether a parameter value is valid when a provisioned product is created or updated. If a parameter value is invalid, CloudFormation does not create or update the stack. If the condition evaluates to `false`, CloudFormation doesn't check the parameter value and proceeds with the stack operation.

Examples

Topics

- Conditionally verify a parameter value
- Cross-parameter validation

Conditionally verify a parameter value

In the following example, the two rules check the value of the `InstanceType` parameter. Depending on the value of the environment parameter (`test` or `prod`), the user must specify `t3.medium` or `t3.large` for the `InstanceType` parameter. The `InstanceType` and `Environment` parameters must be declared in the `Parameters` section of the same template.

JSON

```
{
  "Rules": {
    "testInstanceType": {
      "RuleCondition": {
        "Fn::Equals": [
          {"Ref": "Environment"},
          "test"
        ]
      },
      "Assertions": [
        {
          "Assert": {
            "Fn::Contains": [
              ["t3.medium"],
              {"Ref": "InstanceType"}
            ]
          },
          "AssertDescription": "For a test environment, the instance type must be
 t3.medium"
        }
      ]
    },
    "prodInstanceType": {
      "RuleCondition": {
        "Fn::Equals": [
          {"Ref": "Environment"},
          "prod"
        ]
      },
      "Assertions": [
        {
          "Assert": {
            "Fn::Contains": [
              ["t3.large"],
```

```
              {"Ref": "InstanceType"}
            ]
          },
          "AssertDescription": "For a production environment, the instance type must be
 t3.large"
        }
      ]
    }
  }
}
```

YAML

```
Rules:
  testInstanceType:
    RuleCondition: !Equals
      - !Ref Environment
      - test
    Assertions:
      - Assert:
          'Fn::Contains':
            - - t3.medium
            - !Ref InstanceType
        AssertDescription: 'For a test environment, the instance type must be
 t3.medium'
  prodInstanceType:
    RuleCondition: !Equals
      - !Ref Environment
      - prod
    Assertions:
      - Assert:
          'Fn::Contains':
            - - t3.large
            - !Ref InstanceType
        AssertDescription: 'For a production environment, the instance type must be
 t3.large'
```

Cross-parameter validation

The following sample templates demonstrate the use of rules for cross-parameter validations. They create a sample website running on an Auto Scaling group behind a load balancer. The website is available on port 80 or 443 depending on input parameters. The instances in the Auto Scaling group can be configured to listen on any port (with 8888 as the default).

The rules in this template validate input parameters before stack creation. They verify that all subnets belong to the specified VPC and ensure that when the `UseSSL` parameter is set to `Yes`, both an SSL certificate ARN and hosted zone name are provided.

> **Note**
>
> You will be billed for the AWS resources used if you create a stack from this template.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "VpcId": {
      "Type": "AWS::EC2::VPC::Id",
      "Description": "VpcId of your existing Virtual Private Cloud (VPC)",
      "ConstraintDescription": "must be the VPC Id of an existing Virtual Private
 Cloud."
    },
    "Subnets": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "The list of SubnetIds in your Virtual Private Cloud (VPC)",
      "ConstraintDescription": "must be a list of at least two existing subnets
 associated with at least two different availability zones."
    },
    "InstanceType": {
      "Description": "WebServer EC2 instance type",
      "Type": "String",
      "Default": "t2.micro",
      "AllowedValues": ["t2.micro", "t3.micro"],
      "ConstraintDescription": "must be a valid EC2 instance type."
    },
    "KeyName": {
      "Description": "Name of an existing EC2 KeyPair to enable SSH access to the
 instances",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
    },
    "SSHLocation": {
      "Description": "The IP address range that can be used to SSH to the EC2
 instances",
```

```
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "UseSSL": {
      "AllowedValues": ["Yes", "No"],
      "Default": "No",
      "Description": "Select \"Yes\" to implement SSL, \"No\" to skip (default).",
      "Type": "String"
    },
    "ALBSSLCertificateARN": {
      "Default": "",
      "Description": "[Optional] The ARN of the SSL certificate to be used for the
 Application Load Balancer",
      "Type": "String"
    },
    "HostedZoneName": {
      "AllowedPattern": "^$|(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\\-]*[a-zA-Z0-9])\
\.)*([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]*[A-Za-z0-9])$",
      "Default": "",
      "Description": "[Optional] The domain name of a valid Hosted Zone on AWS.",
      "Type": "String"
    }
  },
  "Conditions": {
    "UseALBSSL": {"Fn::Equals": [{"Ref": "UseSSL"}, "Yes"]}
  },
  "Rules": {
    "SubnetsInVPC": {
      "Assertions": [
        {
          "Assert": {"Fn::EachMemberEquals": [{"Fn::ValueOf": ["Subnets", "VpcId"]},
 {"Ref": "VpcId"}]},
          "AssertDescription": "All subnets must be in the VPC"
        }
      ]
    },
    "ValidateHostedZone": {
      "RuleCondition": {"Fn::Equals": [{"Ref": "UseSSL"}, "Yes"]},
      "Assertions": [
        {
```

```
          "Assert": {"Fn::Not": [{"Fn::Equals": [{"Ref": "ALBSSLCertificateARN"},
""]}]},
          "AssertDescription": "ACM Certificate value cannot be empty if SSL is
required"
        },
        {
          "Assert": {"Fn::Not": [{"Fn::Equals": [{"Ref": "HostedZoneName"}, ""]}]},
          "AssertDescription": "Route53 Hosted Zone Name is mandatory when SSL is
required"
        }
      ]
    }
  },
  "Resources": {
    "WebServerGroup": {
      "Type": "AWS::AutoScaling::AutoScalingGroup",
      "Properties": {
        "VPCZoneIdentifier": {"Ref": "Subnets"},
        "LaunchTemplate": {
          "LaunchTemplateId": {"Ref": "LaunchTemplate"},
          "Version": {"Fn::GetAtt": ["LaunchTemplate","LatestVersionNumber"]}
        },
        "MinSize": "2",
        "MaxSize": "2",
        "TargetGroupARNs": [{"Ref": "ALBTargetGroup"}]
      },
      "CreationPolicy": {
        "ResourceSignal": {"Timeout": "PT15M"}
      },
      "UpdatePolicy": {
        "AutoScalingRollingUpdate": {
          "MinInstancesInService": "1",
          "MaxBatchSize": "1",
          "PauseTime": "PT15M",
          "WaitOnResourceSignals": true
        }
      }
    },
    "LaunchTemplate": {
      "Type": "AWS::EC2::LaunchTemplate",
      "Metadata": {
        "Comment": "Install a simple application",
        "AWS::CloudFormation::Init": {
          "config": {
```

```
            "packages": {"yum": {"httpd": []}},
            "files": {
              "/var/www/html/index.html": {
                "content": {"Fn::Join": ["\n", ["<h1>Congratulations, you have
successfully launched the AWS CloudFormation sample.</h1>"]]},
                "mode": "000644",
                "owner": "root",
                "group": "root"
              },
              "/etc/cfn/cfn-hup.conf": {
                "content": {"Fn::Join": ["", [
                  "[main]\n",
                  "stack=", {"Ref": "AWS::StackId"}, "\n",
                  "region=", {"Ref": "AWS::Region"}, "\n"
                ]]},
                "mode": "000400",
                "owner": "root",
                "group": "root"
              },
              "/etc/cfn/hooks.d/cfn-auto-reloader.conf": {
                "content": {"Fn::Join": ["", [
                  "[cfn-auto-reloader-hook]\n",
                  "triggers=post.update\n",
                  "path=Resources.LaunchTemplate.Metadata.AWS::CloudFormation::Init\n",
                  "action=/opt/aws/bin/cfn-init -v ",
                  "         --stack ", {"Ref": "AWS::StackName"},
                  "         --resource LaunchTemplate ",
                  "         --region ", {"Ref": "AWS::Region"}, "\n",
                  "runas=root\n"
                ]]},
                "mode": "000400",
                "owner": "root",
                "group": "root"
              }
            },
            "services": {
              "sysvinit": {
                "httpd": {
                  "enabled": "true",
                  "ensureRunning": "true"
                },
                "cfn-hup": {
                  "enabled": "true",
                  "ensureRunning": "true",
```

```
                  "files": [
                    "/etc/cfn/cfn-hup.conf",
                    "/etc/cfn/hooks.d/cfn-auto-reloader.conf"
                  ]
                }
              }
            }
          }
        },
        "Properties": {
          "LaunchTemplateName": {"Fn::Sub": "${AWS::StackName}-launch-template"},
          "LaunchTemplateData": {
            "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
            "SecurityGroupIds": [{"Ref": "InstanceSecurityGroup"}],
            "InstanceType": {"Ref": "InstanceType"},
            "KeyName": {"Ref": "KeyName"},
            "UserData": {
              "Fn::Base64": {"Fn::Join": ["", [
                "#!/bin/bash\n",
                "yum install -y aws-cfn-bootstrap\n",
                "/opt/aws/bin/cfn-init -v ",
                "         --stack ", {"Ref": "AWS::StackName"},
                "         --resource LaunchTemplate ",
                "         --region ", {"Ref": "AWS::Region"}, "\n",
                "/opt/aws/bin/cfn-signal -e $? ",
                "         --stack ", {"Ref": "AWS::StackName"},
                "         --resource WebServerGroup ",
                "         --region ", {"Ref": "AWS::Region"}, "\n"
              ]]}
            }
          }
        }
      },
      "ELBSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
          "GroupDescription": "Allow access to the ELB",
          "VpcId": {"Ref": "VpcId"},
          "SecurityGroupIngress": [{
            "Fn::If": [
              "UseALBSSL",
              {
```

```
            "IpProtocol": "tcp",
            "FromPort": 443,
            "ToPort": 443,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIp": "0.0.0.0/0"
          }
        ]
      }]
    }
  },
  "ApplicationLoadBalancer": {
    "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
    "Properties": {
      "Subnets": {"Ref": "Subnets"},
      "SecurityGroups": [{"Ref": "ELBSecurityGroup"}]
    }
  },
  "ALBListener": {
    "Type": "AWS::ElasticLoadBalancingV2::Listener",
    "Properties": {
      "DefaultActions": [{
        "Type": "forward",
        "TargetGroupArn": {"Ref": "ALBTargetGroup"}
      }],
      "LoadBalancerArn": {"Ref": "ApplicationLoadBalancer"},
      "Port": {"Fn::If": ["UseALBSSL", 443, 80]},
      "Protocol": {"Fn::If": ["UseALBSSL", "HTTPS", "HTTP"]},
      "Certificates": [{
        "Fn::If": [
          "UseALBSSL",
          {"CertificateArn": {"Ref": "ALBSSLCertificateARN"}},
          {"Ref": "AWS::NoValue"}
        ]
      }]
    }
  },
  "ALBTargetGroup": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
```

```
        "HealthCheckIntervalSeconds": 30,
        "HealthCheckTimeoutSeconds": 5,
        "HealthyThresholdCount": 3,
        "Port": 80,
        "Protocol": "HTTP",
        "UnhealthyThresholdCount": 5,
        "VpcId": {"Ref": "VpcId"}
      }
    },
    "InstanceSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Enable SSH access and HTTP access on the inbound port",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "SourceSecurityGroupId": {"Fn::Select": [0, {"Fn::GetAtt":
["ApplicationLoadBalancer", "SecurityGroups"]}]}
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIp": {"Ref": "SSHLocation"}
          }
        ],
        "VpcId": {"Ref": "VpcId"}
      }
    },
    "RecordSet": {
      "Type": "AWS::Route53::RecordSetGroup",
      "Condition": "UseALBSSL",
      "Properties": {
        "HostedZoneName": {"Fn::Join": ["", [{"Ref": "HostedZoneName"}, "."]]},
        "RecordSets": [{
          "Name": {"Fn::Join": ["", [
            {"Fn::Select": ["0", {"Fn::Split": [".", {"Fn::GetAtt":
["ApplicationLoadBalancer", "DNSName"]}]}]},
            ".",
            {"Ref": "HostedZoneName"},
            "."
          ]]},
```

```
          "Type": "A",
          "AliasTarget": {
            "DNSName": {"Fn::GetAtt": ["ApplicationLoadBalancer", "DNSName"]},
            "EvaluateTargetHealth": true,
            "HostedZoneId": {"Fn::GetAtt": ["ApplicationLoadBalancer",
 "CanonicalHostedZoneID"]}
          }
        }]
      }
    }
  },
  "Outputs": {
    "URL": {
      "Description": "URL of the website",
      "Value": {"Fn::Join": ["", [
        {"Fn::If": [
          "UseALBSSL",
          {"Fn::Join": ["", [
            "https://",
            {"Fn::Join": ["", [
              {"Fn::Select": ["0", {"Fn::Split": [".", {"Fn::GetAtt":
 ["ApplicationLoadBalancer", "DNSName"]}]}]},
              ".",
              {"Ref": "HostedZoneName"},
              "."
            ]]}
          ]]},
          {"Fn::Join": ["", [
            "http://",
            {"Fn::GetAtt": ["ApplicationLoadBalancer", "DNSName"]}
          ]]}
        ]}
      ]]}
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  VpcId:
    Type: AWS::EC2::VPC::Id
```

```
    Description: VpcId of your existing Virtual Private Cloud (VPC)
    ConstraintDescription: must be the VPC Id of an existing Virtual Private Cloud.
  Subnets:
    Type: List<AWS::EC2::Subnet::Id>
    Description: The list of SubnetIds in your Virtual Private Cloud (VPC)
    ConstraintDescription: >-
      must be a list of at least two existing subnets associated with at least
      two different availability zones. They should be residing in the selected
      Virtual Private Cloud.
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t3.micro
    ConstraintDescription: must be a valid EC2 instance type.
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  UseSSL:
    AllowedValues:
      - 'Yes'
      - 'No'
    ConstraintDescription: Select Yes to create a HTTPS Listener
    Default: 'No'
    Description: 'Select "Yes" to implement SSL, "No" to skip (default).'
    Type: String
  ALBSSLCertificateARN:
    Default: ''
    Description: >-
      [Optional] The ARN of the SSL certificate to be used for the Application
      Load Balancer
    Type: String
  HostedZoneName:
```

```
    AllowedPattern: >-
      ^$|(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-]*[a-zA-Z0-9])\.)*([A-Za-z0-9]|[A-Za-
z0-9][A-Za-z0-9\-]*[A-Za-z0-9])$
    Default: ''
    Description: '[Optional] The domain name of a valid Hosted Zone on AWS.'
    Type: String
Conditions:
  UseALBSSL: !Equals
    - !Ref UseSSL
    - 'Yes'
Rules:
  SubnetsInVPC:
    Assertions:
      - Assert:
          'Fn::EachMemberEquals':
            - 'Fn::ValueOf':
                - Subnets
                - VpcId
            - Ref: VpcId
        AssertDescription: All subnets must be in the VPC
  ValidateHostedZone:
    RuleCondition: !Equals
      - !Ref UseSSL
      - 'Yes'
    Assertions:
      - Assert: !Not
          - !Equals
            - !Ref ALBSSLCertificateARN
            - ''
        AssertDescription: ACM Certificate value cannot be empty if SSL is required
      - Assert: !Not
          - !Equals
            - !Ref HostedZoneName
            - ''
        AssertDescription: Route53 Hosted Zone Name is mandatory when SSL is required
Resources:
  WebServerGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier: !Ref Subnets
      LaunchTemplate:
        LaunchTemplateId: !Ref LaunchTemplate
        Version: !GetAtt LaunchTemplate.LatestVersionNumber
      MinSize: '2'
```

```
      MaxSize: '2'
      TargetGroupARNs:
        - !Ref ALBTargetGroup
    CreationPolicy:
      ResourceSignal:
        Timeout: PT15M
    UpdatePolicy:
      AutoScalingRollingUpdate:
        MinInstancesInService: '1'
        MaxBatchSize: '1'
        PauseTime: PT15M
        WaitOnResourceSignals: 'true'
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Metadata:
      Comment: Install a simple application
      AWS::CloudFormation::Init:
        config:
          packages:
            yum:
              httpd: []
          files:
            /var/www/html/index.html:
              content: !Join
                - |+
                - - >-
                    <h1>Congratulations, you have successfully launched the AWS
                    CloudFormation sample.</h1>
              mode: '000644'
              owner: root
              group: root
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |-
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.LaunchTemplate.Metadata.AWS::CloudFormation::Init
```

```
                action=/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 LaunchTemplate --region ${AWS::Region}
                runas=root
              mode: '000400'
              owner: root
              group: root
          services:
            sysvinit:
              httpd:
                enabled: 'true'
                ensureRunning: 'true'
              cfn-hup:
                enabled: 'true'
                ensureRunning: 'true'
                files:
                  - /etc/cfn/cfn-hup.conf
                  - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
        SecurityGroupIds:
        - !Ref InstanceSecurityGroup
        InstanceType: !Ref InstanceType
        KeyName: !Ref KeyName
        UserData: !Base64
          Fn::Sub: |
            #!/bin/bash
            yum install -y aws-cfn-bootstrap
            /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 LaunchTemplate --region ${AWS::Region}
            /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerGroup --region ${AWS::Region}
  ELBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow access to the ELB
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - !If
          - UseALBSSL
          - IpProtocol: tcp
            FromPort: 443
```

```
          ToPort: 443
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Subnets: !Ref Subnets
      SecurityGroups:
        - !Ref ELBSecurityGroup
  ALBListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref ALBTargetGroup
      LoadBalancerArn: !Ref ApplicationLoadBalancer
      Port: !If
        - UseALBSSL
        - 443
        - 80
      Protocol: !If
        - UseALBSSL
        - HTTPS
        - HTTP
      Certificates:
        - !If
          - UseALBSSL
          - CertificateArn: !Ref ALBSSLCertificateARN
          - !Ref 'AWS::NoValue'
  ALBTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 30
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 3
      Port: 80
      Protocol: HTTP
      UnhealthyThresholdCount: 5
      VpcId: !Ref VpcId
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
```

```
    Properties:
      GroupDescription: Enable SSH access and HTTP access on the inbound port
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          SourceSecurityGroupId: !Select
            - 0
            - !GetAtt
              - ApplicationLoadBalancer
              - SecurityGroups
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref SSHLocation
      VpcId: !Ref VpcId
  RecordSet:
    Type: AWS::Route53::RecordSetGroup
    Condition: UseALBSSL
    Properties:
      HostedZoneName: !Join
        - ''
        - - !Ref HostedZoneName
          - .
      RecordSets:
        - Name: !Join
            - ''
            - - !Select
                - '0'
                - !Split
                  - .
                  - !GetAtt
                    - ApplicationLoadBalancer
                    - DNSName
              - .
              - !Ref HostedZoneName
              - .
          Type: A
          AliasTarget:
            DNSName: !GetAtt
              - ApplicationLoadBalancer
              - DNSName
            EvaluateTargetHealth: true
            HostedZoneId: !GetAtt
```

```
              - ApplicationLoadBalancer
              - CanonicalHostedZoneID
Outputs:
  URL:
    Description: URL of the website
    Value: !Join
      - ''
      - - !If
          - UseALBSSL
          - !Join
            - ''
            - - 'https://'
              - !Join
                - ''
                - - !Select
                    - '0'
                    - !Split
                      - .
                      - !GetAtt
                        - ApplicationLoadBalancer
                        - DNSName
                  - .
                  - !Ref HostedZoneName
                  - .
          - !Join
            - ''
            - - 'http://'
              - !GetAtt
                - ApplicationLoadBalancer
                - DNSName
```

CloudFormation template Conditions syntax

The optional `Conditions` section contains statements that define the circumstances under which entities are created or configured. For example, you can create a condition and associate it with a resource or output so that CloudFormation creates the resource or output only if the condition is true. Similarly, you can associate a condition with a property so that CloudFormation sets the property to a specific value only if the condition is true. If the condition is false, CloudFormation sets the property to an alternative value that you specify.

You can use conditions when you want to reuse a template to create resources in different contexts, such as test versus production environments. For example, in your template, you can

add an `EnvironmentType` input parameter that accepts either `prod` or `test` as inputs. For the `prod` environment, you might include EC2 instances with certain capabilities, while for the `test` environment, you might use reduced capabilities to save money. This condition definition allows you to define which resources are created and how they're configured for each environment type.

Syntax

The `Conditions` section consists of the key name `Conditions`. Each condition declaration includes a logical ID and one or more intrinsic functions.

JSON

```
"Conditions": {
  "LogicalConditionName1": {
    "Intrinsic function": ...[
  },

  "LogicalConditionName2": {
    "Intrinsic function": ...
  }
}
```

YAML

```
Conditions:
  LogicalConditionName1:
    Intrinsic function:
      ...

  LogicalConditionName2:
    Intrinsic function:
      ...
```

How conditions work

To use conditions, follow these steps:

1. **Add a parameter definition** – Define the inputs that your conditions will evaluate in the `Parameters` section of your template. The conditions evaluate to true or false based on these input parameter values. Note that pseudo parameters are automatically available and don't

require explicit definition in the `Parameters` section. For more information about pseudo parameters, see Get AWS values using pseudo parameters.

2. **Add a condition definition** – Define conditions in the `Conditions` section using intrinsic functions such as `Fn::If` or `Fn::Equals`. These conditions determine when CloudFormation creates the associated resources. The conditions can be based on:
   - Input or pseudo parameter values
   - Other conditions
   - Mapping values

   However, you can't reference resource logical IDs or their attributes in conditions.
3. **Associate conditions with resources or outputs** – Reference conditions in resources or outputs using the `Condition` key and a condition's logical ID. Optionally, use `Fn::If` in other parts of the template (such as property values) to set values based on a condition. For more information, see Using the `Condition` key.

CloudFormation evaluates conditions when creating or updating a stack. CloudFormation creates entities that are associated with a true condition and ignores entities that are associated with a false condition. CloudFormation also re-evaluates these conditions during each stack update before modifying any resources. Entities that remain associated with a true condition are updated, while those that become associated with a false condition are deleted.

> **Important**
>
> During a stack update, you can't update conditions by themselves. You can update conditions only when you include changes that add, modify, or delete resources.

Condition intrinsic functions

You can use the following intrinsic functions to define conditions:

- Fn::And
- Fn::Equals
- Fn::ForEach
- Fn::If
- Fn::Not

- Fn::Or

> **Note**
>
> `Fn::If` is only supported in the metadata attribute, update policy attribute, and property values in the `Resources` section and `Outputs` sections of a template.

Using the `Condition` key

Once a condition is defined, you can apply it in several places in the template, such as `Resources` and `Outputs`, using the `Condition` key. The `Condition` key references a condition's logical name and returns the evaluated result of the specified condition.

Topics

- Associate conditions with resources
- Associate conditions with outputs
- Reference conditions in other conditions
- Conditionally return property values using Fn::If

Associate conditions with resources

To conditionally create resources, add the `Condition` key and the logical ID of the condition as an attribute to the resource. CloudFormation creates the resource only when the condition evaluates to true.

JSON

```
"NewVolume" : {
  "Type" : "AWS::EC2::Volume",
  "Condition" : "IsProduction",
  "Properties" : {
     "Size" : "100",
     "AvailabilityZone" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone" ]}
  }
}
```

YAML

```
NewVolume:
  Type: AWS::EC2::Volume
  Condition: IsProduction
  Properties:
    Size: 100
    AvailabilityZone: !GetAtt EC2Instance.AvailabilityZone
```

Associate conditions with outputs

You can also associate conditions with outputs. CloudFormation creates the output only when the associated condition evaluates to true.

JSON

```
"Outputs" : {
  "VolumeId" : {
    "Condition" : "IsProduction",
    "Value" : { "Ref" : "NewVolume" }
  }
}
```

YAML

```
Outputs:
  VolumeId:
    Condition: IsProduction
    Value: !Ref NewVolume
```

Reference conditions in other conditions

When defining conditions in the `Conditions` section, you can reference other conditions using the `Condition` key. This allows you to create more complex conditional logic by combining multiple conditions.

In the following example, the `IsProdAndFeatureEnabled` condition evaluates to true only if the `IsProduction` and `IsFeatureEnabled` conditions evaluate to true.

JSON

```
"Conditions": {
```

```
  "IsProduction" : {"Fn::Equals" : [{"Ref" : "Environment"}, "prod"]},
  "IsFeatureEnabled" : { "Fn::Equals" : [{"Ref" : "FeatureFlag"}, "enabled"]},
  "IsProdAndFeatureEnabled" : {
    "Fn::And" : [
      {"Condition" : "IsProduction"},
      {"Condition" : "IsFeatureEnabled"}
    ]
  }
}
```

YAML

```
Conditions:
  IsProduction:
    !Equals [!Ref Environment, "prod"]
  IsFeatureEnabled:
    !Equals [!Ref FeatureFlag, "enabled"]
  IsProdAndFeatureEnabled: !And
    - !Condition IsProduction
    - !Condition IsFeatureEnabled
```

Conditionally return property values using `Fn::If`

For more granular control, you can use the `Fn::If` intrinsic function to conditionally return one of two property values within resources or outputs. This function evaluates a condition and returns one value if the condition is true and another value if the condition is false.

Conditional property values

The following example demonstrates setting an EC2 instance type based on an environment condition. If the `IsProduction` condition evaluates to true, the instance type is set to `c5.xlarge`. Otherwise, it's set to `t3.small`.

JSON

```
"Properties" : {
  "InstanceType" : {
    "Fn::If" : [
      "IsProduction",
      "c5.xlarge",
      "t3.small"
```

```
    ]
  }
}
```

YAML

```
Properties:
  InstanceType: !If
    - IsProduction
    - c5.xlarge
    - t3.small
```

Conditional property removal

You can also use the `AWS::NoValue` pseudo parameter as a return value to remove the corresponding property when a condition is false.

JSON

```
"DBSnapshotIdentifier" : {
  "Fn::If" : [
    "UseDBSnapshot",
    {"Ref" : "DBSnapshotName"},
    {"Ref" : "AWS::NoValue"}
  ]
}
```

YAML

```
DBSnapshotIdentifier: !If
  - UseDBSnapshot
  - !Ref DBSnapshotName
  - !Ref "AWS::NoValue"
```

Examples

Topics

- Environment-based resource creation
- Multi-condition resource provisioning

Environment-based resource creation

This following examples provision an EC2 instance, and conditionally create and attach a new EBS volume only if the environment type is `prod`. If the environment is `test`, they just create the EC2 instance without the additional volume.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "EnvType": {
            "Description": "Environment type",
            "Default": "test",
            "Type": "String",
            "AllowedValues": [
                "prod",
                "test"
            ],
            "ConstraintDescription": "must specify prod or test"
        }
    },
    "Conditions": {
        "IsProduction": {
            "Fn::Equals": [
                {
                    "Ref": "EnvType"
                },
                "prod"
            ]
        }
    },
    "Resources": {
        "EC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-1234567890abcdef0",
                "InstanceType": "c5.xlarge"
            }
        },
        "MountPoint": {
            "Type": "AWS::EC2::VolumeAttachment",
            "Condition": "IsProduction",
```

```
            "Properties": {
                "InstanceId": {
                    "Ref": "EC2Instance"
                },
                "VolumeId": {
                    "Ref": "NewVolume"
                },
                "Device": "/dev/sdh"
            }
        },
        "NewVolume": {
            "Type": "AWS::EC2::Volume",
            "Condition": "IsProduction",
            "Properties": {
                "Size": 100,
                "AvailabilityZone": {
                    "Fn::GetAtt": [
                        "EC2Instance",
                        "AvailabilityZone"
                    ]
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvType:
    Description: Environment type
    Default: test
    Type: String
    AllowedValues:
      - prod
      - test
    ConstraintDescription: must specify prod or test
Conditions:
  IsProduction: !Equals
    - !Ref EnvType
    - prod
Resources:
```

```
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
      InstanceType: c5.xlarge
  MountPoint:
    Type: AWS::EC2::VolumeAttachment
    Condition: IsProduction
    Properties:
      InstanceId: !Ref EC2Instance
      VolumeId: !Ref NewVolume
      Device: /dev/sdh
  NewVolume:
    Type: AWS::EC2::Volume
    Condition: IsProduction
    Properties:
      Size: 100
      AvailabilityZone: !GetAtt
        - EC2Instance
        - AvailabilityZone
```

Multi-condition resource provisioning

The following examples conditionally create an S3 bucket if a bucket name is provided, and attach a bucket policy only when the environment is set to `prod`. If no bucket name is given or the environment is `test`, no resources are created.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "EnvType": {
            "Type": "String",
            "AllowedValues": [
                "prod",
                "test"
            ]
        },
        "BucketName": {
            "Default": "",
            "Type": "String"
        }
```

```
    },
    "Conditions": {
        "IsProduction": {
            "Fn::Equals": [
                {
                    "Ref": "EnvType"
                },
                "prod"
            ]
        },
        "CreateBucket": {
            "Fn::Not": [
                {
                    "Fn::Equals": [
                        {
                            "Ref": "BucketName"
                        },
                        ""
                    ]
                }
            ]
        },
        "CreateBucketPolicy": {
            "Fn::And": [
                {
                    "Condition": "IsProduction"
                },
                {
                    "Condition": "CreateBucket"
                }
            ]
        }
    },
    "Resources": {
        "Bucket": {
            "Type": "AWS::S3::Bucket",
            "Condition": "CreateBucket",
            "Properties": {
                "BucketName": {
                    "Ref": "BucketName"
                }
            }
        },
        "Policy": {
```

```
            "Type": "AWS::S3::BucketPolicy",
            "Condition": "CreateBucketPolicy",
            "Properties": {
                "Bucket": {
                    "Ref": "Bucket"
                },
                "PolicyDocument": { ... }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EnvType:
    Type: String
    AllowedValues:
      - prod
      - test
  BucketName:
    Default: ''
    Type: String
Conditions:
  IsProduction: !Equals
    - !Ref EnvType
    - prod
  CreateBucket: !Not
    - !Equals
      - !Ref BucketName
      - ''
  CreateBucketPolicy: !And
    - !Condition IsProduction
    - !Condition CreateBucket
Resources:
  Bucket:
    Type: AWS::S3::Bucket
    Condition: CreateBucket
    Properties:
      BucketName: !Ref BucketName
  Policy:
    Type: AWS::S3::BucketPolicy
```

```
    Condition: CreateBucketPolicy
    Properties:
      Bucket: !Ref Bucket
      PolicyDocument: ...
```

In this example, the `CreateBucketPolicy` condition demonstrates how to reference other conditions using the `Condition` key. The policy is created only when both the `IsProduction` and `CreateBucket` conditions evaluate to true.

> **Note**
>
> For more complex examples of using conditions, see the Condition attribute topic in the *CloudFormation Template Reference Guide*.

CloudFormation template Transform section

The optional `Transform` section specifies one or more macros that CloudFormation uses to process your template in some way.

Macros can perform simple tasks like finding and replacing text, or they can make more extensive transformations to the entire template. CloudFormation executes macros in the order that they're specified. When you create a change set, CloudFormation generates a change set that includes the processed template content. You can then review the changes and execute the change set. For more information about how macros work, see Perform custom processing on CloudFormation templates with template macros.

CloudFormation also supports *transforms*, which are macros hosted by CloudFormation. CloudFormation treats these transforms the same as any macros you create in terms of execution order and scope. For more information, see Transform reference.

To declare multiple macros, use a list format and specify one or more macros.

For example, in the template sample below, CloudFormation evaluates `MyMacro` and then `AWS::Serverless`, both of which can process the contents of the entire template because of their inclusion in the `Transform` section.

```
# Start of processable content for MyMacro and AWS::Serverless
Transform:
```

```
  - MyMacro
  - 'AWS::Serverless'
Resources:
  WaitCondition:
    Type: AWS::CloudFormation::WaitCondition
  MyBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: amzn-s3-demo-bucket
      Tags: [{"key":"value"}]
      CorsConfiguration: []
  MyEc2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
# End of processable content for MyMacro and AWS::Serverless
```

CloudFormation template format version syntax

The `AWSTemplateFormatVersion` section (optional) identifies the template format version that the template conforms to. The latest template format version is `2010-09-09` and is currently the only valid value.

The template format version isn't the same as the API version. The template format version can change independently of the API versions.

The value for the template format version declaration must be a literal string. You can't use a parameter or function to specify the template format version. If you don't specify a value, CloudFormation assumes the latest template format version. The following snippet is an example of a valid template format version declaration:

JSON

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
```

CloudFormation template Description syntax

The `Description` section (optional) enables you to include a text string that describes the template. This section must always follow the template format version section.

The value for the description declaration must be a literal string that is between 0 and 1024 bytes in length. You cannot use a parameter or function to specify the description. The following snippet is an example of a description declaration:

> **Important**
>
> During a stack update, you cannot update the `Description` section by itself. You can update it only when you include changes that add, modify, or delete resources.

JSON

```
"Description" : "Here are some details about the template."
```

YAML

```
Description: > Here are some details about the template.
```

CloudFormation Language Server

The AWS CloudFormation Language Server helps you write CloudFormation templates in an editor. As you type, it suggests resource types and properties, shows documentation, finds errors, lets you jump to definitions, and offers fixes for some problems. It uses the Language Server Protocol (LSP), which lets the server work with different editors. Starting and using the Language Server for these local authoring features doesn't require an AWS account or AWS credentials.

The AWS Toolkit integrations also use the Language Server for online workflows that call AWS services. These workflows can browse resources and stacks. They can also run service-side template validation, deploy templates, show change set differences, and monitor stack events. These online workflows are the only features that require an active AWS account and valid AWS credentials.

For release notes and setup examples for different editors, see CloudFormation Language Server on GitHub.

Topics

- IDEs integrating with the CloudFormation Language Server
- Getting started
- Initializing a CloudFormation project in the IDE (VS Code only)
- Need help?

IDEs integrating with the CloudFormation Language Server

AWS provides off-the-shelf integration with the CloudFormation Language Server through the AWS Toolkit for the following IDEs:

- Visual Studio Code
- JetBrains IDEs (version 2025.3 or later), including IntelliJ IDEA, WebStorm, and PyCharm

The following IDEs also support the CloudFormation Language Server:

- Kiro
- Cursor
- Most VS Code forks and distributions

For Visual Studio Code and JetBrains IDEs, install the AWS Toolkit. The Toolkit includes and starts the Language Server, so you don't need to download the server separately.

For another editor that supports LSP, download the file for your computer from the latest release on GitHub, extract it, and configure your editor to run the following command:

```
node /path/to/install-location/cfn-lsp-server-standalone.js --stdio
```

For supported Node.js versions, available downloads, file verification, and setup examples for different editors, see the standalone installation guide.

Getting started

The following steps show how to use the Language Server through the AWS Toolkit for Visual Studio Code and JetBrains IDEs. For another editor, follow its setup example in the standalone installation guide.

Topics

- Prerequisites
- Step 1: Install or upgrade the AWS Toolkit
- Step 2: Access CloudFormation in the AWS Toolkit
- Step 3: Validate, test, and refine your template
- Step 4: Navigate through the template
- Step 5: Validate and deploy

Prerequisites

Before you begin, make sure that:

- You are using a supported IDE on macOS, Windows, or Linux.
- For Visual Studio Code or a JetBrains IDE, install or update the AWS Toolkit.
- For another editor, install a supported Node.js version and download the Language Server file for your computer. The standalone installation guide lists the supported versions and files.

You don't need an AWS account or AWS credentials to start the Language Server or use its local authoring and template-checking features. Configure valid AWS credentials in the Toolkit only when you use online workflows that browse resources or stacks, call the CloudFormation service, or deploy a template.

Step 1: Install or upgrade the AWS Toolkit

Install or update to the latest version of the AWS Toolkit from your IDE's extension or plugin manager, then restart your IDE.

After installation, the AWS Toolkit automatically enables CloudFormation IDE support. When you first install or upgrade the AWS Toolkit with the CloudFormation Language Server, you are prompted to grant permission for AWS to collect anonymous usage data. This data helps AWS improve the CloudFormation Language Server and enhances the authoring experience. No sensitive information is collected and AWS does not record or store template content, resource configurations, or any identifiable customer data. You can change your telemetry preferences at any time from the IDE settings. Restart the IDE for the changes to take effect. The usage data collected focuses only on feature interactions and performance metrics. These insights help AWS

identify and prioritize improvements such as faster validation, enhanced autocomplete, and better error diagnostics.

Step 2: Access CloudFormation in the AWS Toolkit

After installing the AWS Toolkit, open the CloudFormation panel in your IDE. In VS Code, open the AWS Toolkit panel from the activity bar and choose **CLOUDFORMATION**. In JetBrains IDEs, open the **AWS Toolkit** tool window from the side bar and select the **CloudFormation** tab.

The CloudFormation panel contains the following sections:

- **Region**: Displays the current AWS Region. In VS Code, you can change it by selecting the Region name or by using the **AWS CloudFormation: Select Region** command from the command palette. In JetBrains IDEs, the Region is configured through the AWS Toolkit connection settings.
- **Stacks**: Displays a paginated list of CloudFormation stacks in your account. Expand a stack to view its **Change Sets** node, which lists the change sets associated with that stack. Use the View Stack Detail action to open the stack detail view, which displays the stack overview, events, outputs, and resources.
- **Resources**: After you add a resource type, the panel displays the AWS resources of that type in your account. You can view, refresh, copy, or import them into your template.

In JetBrains IDEs, the toolbar above the tree provides quick access to common actions including **Validate and Deploy**, **Rerun Validate and Deploy**, **Add Resource Type**, and **Refresh**. Actions are also available through right-click context menus on tree nodes.

Step 3: Validate, test, and refine your template

As you write your CloudFormation template, the CloudFormation Language Server runs in the background and provides local authoring assistance without AWS credentials:

- Code completion: Suggests resource types, parameters, and properties based on CloudFormation schemas.
- Extract to parameter: When your cursor is on a literal value in a template (for example, a string like `t2.micro`), the IDE offers a refactoring action to extract the value into the `Parameters` section and replace the literal with a `!Ref` to the new parameter. If the same literal value appears in multiple places, you can choose to extract all occurrences at once.

The Toolkit also provides an online workflow for adding existing AWS resources to your template. This workflow requires valid AWS credentials and uses the AWS Cloud Control API (CCAPI) to retrieve a resource's live configuration and properties, helping you clone or reuse existing infrastructure within your template.

To add resources to your template

- **Add a resource type**: In the AWS Toolkit CloudFormation panel, under **Resources**, add a resource type to browse. In VS Code, click the **Add +** icon or use the **AWS CloudFormation: Add Resource Types** command from the command palette. In JetBrains, click the **Add Resource Type** button in the toolbar or right-click the **Resources** node.
- **Search for a resource type**: In the search dialog, type the AWS resource type you want to add. Example:
  - `AWS::S3::Bucket`
  - `AWS::Lambda::Function`
- **Browse resources**: Under the **Resources** section, a paginated list of detected AWS resources in your account is displayed. If you have many resources, only the first page is shown. Use the navigation controls to move through additional pages and view all available resources.
- Choose the resource you want to include in your template.
- You can insert a resource into your template in two ways, depending on your goal:
  - **Clone an existing resource**: Create a new resource in your template using the live configuration and properties of an existing AWS resource.
  - **Import an existing resource**: Insert the actual resource into your stack by adding it to your template using its live state.

Tips

- You can refresh the **Resources** section at any time to view the latest list of resources available in your account or Region.
- If you are importing resources, do not add a resource that already belongs to an existing CloudFormation stack in the same account.
- To confirm if a resource is already managed by CloudFormation, use the information action next to the resource. In VS Code, click the **i** icon. In JetBrains IDEs, right-click the resource and choose **Get Stack Management Info**.

Add related resources

In VS Code, run the **AWS CloudFormation: Add Related Resources by Type** command from the command palette. Select a resource type from those already defined in your template. The IDE lists resource types that the selected type can reference through its properties. For example, if you select an `AWS::EC2::Instance`, the IDE may suggest `AWS::EC2::SecurityGroup` or `AWS::EC2::Subnet`. Select one or more related resource types, and then choose whether to create new resource definitions or import existing resources. This feature is currently not supported in JetBrains IDEs.

Static validation

The CloudFormation Language Server provides built-in static validation powered by CloudFormation Linter and CloudFormation Guard. These checks run locally behind the scenes and don't require AWS credentials. They help you identify syntax errors, compliance gaps, and best-practice issues before deployment.

Static validation overview

You will see two types of real-time static validations in the IDE:

- CloudFormation Linter (`cfn-lint`): Validates your template against CloudFormation resource specifications and schema rules.
- Guard (`cfn-guard`): Validates your template against compliance rules and organizational policy packs.

CloudFormation Linter (cfn-lint)

The CloudFormation Linter is integrated into the IDE to automatically check your template syntax and structure as you type.

- **Schema validation**: Detects syntax and schema errors to ensure your templates conform to CloudFormation resource schema.
- **Error highlighting**: Displays inline markers under issues, representing deployment blockers or warnings.
- **Hover help**: When you hover over an error, the IDE shows the diagnostic message associated with that issue. If a quick fix is available, it is also offered.

Guard integration

Guard validates your templates against rule sets that define compliance and security policies. The IDE runs Guard validations in real time through the CloudFormation Language Server, giving you immediate feedback while you author templates.

- **Default rule packs**: The IDE includes a pre-registered set of Guard rules focused on foundational best practices for resource security and configuration hygiene. To learn more, see the AWS Guard Rules Registry on GitHub.
- **Adding rule packs**: To add or modify rule sets, open your IDE settings and navigate to the Guard configuration section to select or upload additional Guard rule packs.

Tips: Understanding diagnostic indicators

- Blue indicators: Best practice hints or optimization recommendations.
- Yellow indicators: Warnings for non-blocking issues (for example, missing tags or parameters).
- Red indicators: Deployment blockers such as invalid property names, missing required fields, or schema mismatches.

Step 4: Navigate through the template

The IDE provides a structured, hierarchical view of your CloudFormation template, organized into sections such as `Parameters`, `Resources`, `Outputs`, and `Mappings`, showing each resource type and logical ID. This makes it easy to quickly locate and navigate to specific resources or parameters within large templates. In VS Code, the **Outline** panel in the **Explorer** sidebar displays this structure. In JetBrains IDEs, open the **Structure** tool window to view the template structure for the currently open file.

You can use **Go to Definition** for intrinsic functions such as `GetAtt` and `Ref`, allowing you to jump directly to the referenced resource or parameter in your template. This helps you trace dependencies, understand resource relationships, and make edits more efficiently.

Step 5: Validate and deploy

Validate and Deploy is an online workflow that calls the CloudFormation service and requires valid AWS credentials. When you are ready to deploy your template, use this feature to create a change set. The IDE validates your template, and if no blocking errors are found, it proceeds to create a

drift-aware change set. The IDE then shows a diff view so you can review all proposed changes before executing the change set.

In VS Code, open the command palette and run **AWS CloudFormation: Validate and Deploy**. The command palette guides you through selecting a template, stack name, parameters, capabilities, and other deployment options step by step. In JetBrains IDEs, use the **Validate and Deploy** toolbar button, right-click a template file in the editor, or right-click a stack in the tree. JetBrains presents a wizard dialog where you configure all deployment options including template selection, stack name, parameters, capabilities, tags, and advanced options.

How validation works

The IDE automatically performs a validation check before deployment and validates your template against common failure causes, including:

- Invalid property syntax or schema mismatches: These issues are typically caught by `cfn-lint` during authoring, but if you proceed to deploy without addressing them, CloudFormation's deployment-time validation surfaces the same errors before the stack is created or updated.
- Resource name conflicts with existing resources in your account.
- Service-specific constraints, such as S3 bucket name conflicts or missing encryption.

If the validation detects errors, the IDE highlights the issues directly in your template and lists the errors in the diagnostics panel. Each issue includes the specific property or resource that caused the failure, along with a suggested fix. If there are no blocking errors, you can proceed to the deployment phase.

If warnings are found (non-blocking issues), a dialog appears allowing you to either proceed with deployment or cancel and make corrections.

The IDE opens a drift-aware change set that displays any differences between your current template and the deployed stack configuration. This allows you to review, confirm, or cancel the change set before execution. Cancelling the deployment deletes the change set.

Drift-aware change sets enhance the CloudFormation deployment process by allowing you to handle stack drift safely. Stack drift occurs when the actual state of your resources differs from what is defined in your CloudFormation template, often due to manual changes made through the AWS Management Console, CLI, or SDK. CloudFormation drift-aware change sets compare your processed stack configuration with the live resource state, and the IDE surfaces these differences so you can bring resources back into compliance before deployment.

View stack events

When the deployment starts, you can monitor progress in real time from the CloudFormation panel. Under **Stack Events**, you see a list of operations performed during the deployment. Each event includes details such as:

- **Timestamp**: The time the event occurred
- **Resource**: The specific AWS resource being created, updated, or deleted
- **Status**: The current state of the operation (for example, `CREATE_IN_PROGRESS`, `UPDATE_COMPLETE`, or `ROLLBACK_IN_PROGRESS`)
- **Reason**: Additional context or error messages, if applicable

You can also view the stack's **Resources** and **Outputs** from this panel. The **Stack Events** view helps you track deployment progress, identify potential issues, and confirm when your stack has completed successfully.

Initializing a CloudFormation project in the IDE (VS Code only)

Initializing a CloudFormation project in the IDE helps you set up a structured workspace with the correct folders and environment configuration for service-side validation and deployment. These online workflows require AWS credentials, but the Language Server's local authoring and static validation features don't. You can initialize a new CloudFormation project directly from the IDE to create this recommended setup. This feature is currently available in VS Code only and is not supported in JetBrains IDEs.

To initialize a CloudFormation project:

- **Open the command palette**
  - From VS Code, open the command palette (`Ctrl+Shift+P` or `Cmd+Shift+P` on macOS).
  - Choose **AWS CloudFormation: CFN Init: Initialize Project**.
- **Choose a project directory**
  - By default, the IDE uses your current working directory.
  - You can change this path to any folder where you want to store your CloudFormation templates.
- **Select your AWS credential profile**

  - You are prompted to choose an AWS credential profile. The selected profile is used for environment detection, service-side validation, and deployment.
- **Set up your environment**
  - You are prompted to create or select an environment.
  - Environments define where and how your templates are deployed or validated (for example, dev, beta, or production). You can use **AWS CloudFormation: CFN Init: Add Environment** to select or change your environment.
  - You can use **AWS CloudFormation: CFN Init: Remove Environment** to remove the environment you have selected.
- **(Optional) Import parameter files**
  - If you already have existing parameter files, the IDE allows you to import them during initialization.
  - The IDE automatically detects compatible files and links them to your project for use in template validation and deployment.
- **Name and finalize the project**
  - Provide a project name, such as beta-environment, and complete the setup.
  - The IDE creates the initial project structure and configuration file for you.

You can run validations, preview deployments, or switch between environments directly from the IDE.

Need help?

Try the CloudFormation community on AWS re:Post.

Validate templates with `cfn-lint`

AWS CloudFormation Linter (`cfn-lint`) checks JSON and YAML templates against the CloudFormation resource provider schemas and additional rules. It can find invalid resource properties, unsupported property values, broken references, and common best-practice problems. Use it to catch these problems before you send a template to CloudFormation. If a template uses the AWS Serverless Application Model (AWS SAM) transform, `cfn-lint` applies the transform before it checks the template.

For release notes, the complete rule list, and all command options, see `cfn-lint` on GitHub.

For the limitations of local validation and steps to take before deployment, see Understand validation scope.

Topics

- Install cfn-lint
- Run cfn-lint
- Understand the results
- Configure checks
- Add custom rules
- Add cfn-lint to your workflow

Install `cfn-lint`

Install `cfn-lint` from the Python Package Index (PyPI):

```
python3 -m pip install cfn-lint
```

On macOS, you can instead install it with Homebrew:

```
brew install cfn-lint
```

To confirm the installation, print the installed version:

```
cfn-lint --version
```

For a Docker image and other installation methods, see the installation instructions on GitHub.

Run `cfn-lint`

Pass one or more template paths to the command:

```
cfn-lint template.yaml
cfn-lint template-one.yaml template-two.json
```

You can also use wildcards to check every template in a directory. Use ** to include subdirectories:

```
cfn-lint templates/*.yaml
```

```
cfn-lint templates/**/*.yaml
```

> **Note**
>
> In `bash`, run `shopt -s globstar` before you use `**` so that it matches subdirectories. In `zsh`, `**` matches subdirectories by default.

To read a template from standard input, pass `-` as the template path:

```
cat template.yaml | cfn-lint -
```

By default, `cfn-lint` checks templates against the schemas for `us-east-1`. To check templates for other AWS Regions, use the `--regions` option. Because this option accepts multiple values, add `--` before the template path or use `-t` to identify the template:

```
cfn-lint --regions us-east-1 us-west-2 -- template.yaml
cfn-lint --regions us-east-1 us-west-2 -t template.yaml
```

Some checks depend on parameter values. To check a template with specific values, pass them with `--parameters` or provide a parameter file with `--parameter-files`:

```
cfn-lint --parameters InstanceType=t3.micro Environment=prod -- template.yaml
cfn-lint --parameter-files params.json -- template.yaml
```

To list every rule that `cfn-lint` can apply, run the following command:

```
cfn-lint --list-rules
```

Understand the results

For each finding, `cfn-lint` prints the rule ID, a message, and the file, line, and column of the problem. For the complete list of rules and what each one checks, see cfn-lint rules on GitHub.

To change the output format, use `--format` with `quiet`, `parseable`, `json`, `junit`, `pretty`, or `sarif`:

```
cfn-lint --format json template.yaml
```

`cfn-lint` returns exit code `0` when it finds no issues and a nonzero exit code otherwise. To control which findings cause a nonzero exit code, use `--non-zero-exit-code`. You can use the exit code to stop an automated build when a template doesn't pass your selected checks.

Configure checks

You can configure `cfn-lint` with command-line options, a configuration file, or metadata in the template. When the same setting appears in more than one place, command-line options override template metadata, and template metadata overrides the configuration file.

Select and configure rules

To skip rules, use `--ignore-checks` with one or more rule IDs or rule ID prefixes. To run rules that are off by default, use `--include-checks`. To run a rule even when it matches an ignore setting, use `--mandatory-checks`.

```
cfn-lint --ignore-checks RULE_ID -- template.yaml
```

Some rules accept settings. Use `--configure-rule` with the format *`RULE_ID`*`:`*`key`*`=`*`value`*:

```
cfn-lint --configure-rule RULE_ID:key=value -- template.yaml
```

Use a configuration file

To keep settings with your templates, create a `.cfnlintrc`, `.cfnlintrc.yaml`, or `.cfnlintrc.yml` file in the directory where you run `cfn-lint`. You can also create a `~/.cfnlintrc` file in your home directory for settings that apply everywhere. The following example selects the templates to check, skips one template, sets the Regions, and skips a rule:

```
templates:
  - templates/**/*.yaml
ignore_templates:
  - templates/legacy.yaml
regions:
  - us-east-1
  - us-west-2
ignore_checks:
  - RULE_ID
```

When the configuration file selects templates, you can run `cfn-lint` without arguments to check all of them. For all settings, see Config file on GitHub.

Configure checks in the template

To configure `cfn-lint` for one template, add a `cfn-lint` key to the top-level `Metadata` section:

```
Metadata:
  cfn-lint:
    config:
      regions:
        - us-east-1
        - us-east-2
      ignore_checks:
        - RULE_ID
```

To skip a rule for a single resource, add the same key to that resource's `Metadata` attribute. The following example skips a rule for one instance only:

```
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Metadata:
      cfn-lint:
        config:
          ignore_checks:
            - RULE_ID
    Properties:
      InstanceType: t3.micro
      ImageId: ami-abc1234
```

Add custom rules

You can add your own checks without writing Python. A custom rules file contains one rule per line in the following format:

```
ResourceType Property Operator Value [ErrorLevel] [Message]
```

For example, the following rule in `custom_rules.txt` reports a warning when a template uses an expensive instance type:

```
AWS::EC2::Instance InstanceType NOT_EQUALS "m4.16xlarge" WARN "This is an expensive
 instance type, don't use it"
```

Pass the file with `--custom-rules` (or `-z`), or add a `custom_rules` setting to your configuration file:

```
cfn-lint --custom-rules custom_rules.txt -- template.yaml
```

For the supported operators, see Custom rules on GitHub. For more complex checks, you can write rules in Python and load them with `--append-rules`. To change what `cfn-lint` considers valid, such as requiring specific tags, you can provide a specification override file with `--override-spec`. For more information, see Getting started with rules and Customize specifications on GitHub.

Add `cfn-lint` to your workflow

You can run `cfn-lint` in the following places:

- **In your editor** – The CloudFormation Language Server uses `cfn-lint` to show problems while you type. For setup, see CloudFormation Language Server. Plugins are also available for other editors, including Vim, Emacs, and Sublime Text.
- **Before a Git commit** – Add `cfn-lint` to your `.pre-commit-config.yaml` file so that it runs on changed templates before each commit. Replace the `rev` value with the version that you want to use.

  ```
  repos:
    - repo: https://github.com/aws-cloudformation/cfn-lint
      rev: v1.56.0
      hooks:
        - id: cfn-lint
          files: templates/.*\.(json|yml|yaml)$
  ```

- **In an automated build** – Run `cfn-lint` as a build step and use the exit code to fail the build. A GitHub Action for `cfn-lint` is also available.
- **In your own Python code** – Import `cfnlint.api` and call `lint` or `lint_all` to check a template string and process the results in your application.

  ```
  from cfnlint.api import lint, ManualArgs
  ```

```
with open("template.yaml") as f:
    template = f.read()

matches = lint(template, config=ManualArgs(regions=["us-east-1"]))
for match in matches:
    print(match)
```

For editor plugins, the GitHub Action, and more integration examples, see `cfn-lint` on GitHub.

Validate templates with `cloudformation-validate`

AWS CloudFormation Validate (`cloudformation-validate`) helps you find template problems before deployment. It checks JSON and YAML templates locally for invalid template structure, broken references, security issues, and best-practice problems. Each finding includes a stable rule ID, a severity, the line and column in the template, the affected resource, and, when available, a suggested fix.

The tool includes the rules and CloudFormation resource schemas that it needs. As a result, it runs without network access or AWS credentials after installation. Validation of a typical template completes in less than a second. You can run it from the command line or embed it in a Rust, Node.js, Python, Go, or JVM application. You can also let AWS CDK run it after synthesis. All of these methods produce the same results.

For release notes and the complete list of built-in rules, see `cloudformation-validate` on GitHub.

For the limitations of local validation and steps to take before deployment, see Understand validation scope.

Topics

- How validation works
- Choose a validation method
- Install the command-line tool
- Run validation
- Understand the results
- Filter the results

- Add custom rules
- Validate new resource types and properties
- Embed the validation library
- Use with AWS CDK

How validation works

When you validate a template, `cloudformation-validate` resolves intrinsic functions such as `Ref`, `Fn::GetAtt`, `Fn::Sub`, and `Fn::If`, and then runs the following checks:

- **Schema checks** – Compares each resource with the CloudFormation resource provider schemas and reports type mismatches, missing required properties, invalid enumeration values, and pattern or constraint failures.
- **Rules** – Runs the built-in rules and any custom rules that you provide. These rules find semantic errors, broken cross-resource references, security risks, and best-practice problems.
- **State machine checks** – Validates the definitions of `AWS::StepFunctions::StateMachine` resources, including state types, `StartAt` and `Next` references, and required fields.

Choose a validation method

- **Command line** – Run `cfn-validate` to check one template or every template in a directory from a terminal or an automated build. See Install the command-line tool.
- **Library** – Add the validator to a Rust, Node.js, Python, Go, or JVM application and process the results in your code. See Embed the validation library.
- **AWS CDK** – Validate templates automatically after AWS CDK synthesizes them. See Use with AWS CDK.

With any method, you can add custom rules written in Common Expression Language (CEL), Rego, or the Guard rule language. See Add custom rules.

Install the command-line tool

The `cfn-validate` command is a single file with no dependencies. Open the latest release on GitHub and download the file for your operating system and processor. Files are available for Linux (x86-64 and ARM64), macOS (Apple silicon and Intel), and Windows (x86-64 and ARM64).

On Linux or macOS, make the file executable, rename it to `cfn-validate`, and move it to a directory on your PATH. The following example uses the Linux x86-64 file. Replace *VERSION* with the version that you downloaded.

```
chmod +x cfn-validate-VERSION-linux-x64
sudo mv cfn-validate-VERSION-linux-x64 /usr/local/bin/cfn-validate
```

On Windows, rename the file to `cfn-validate.exe` and move it to a directory on your PATH.

Each release also includes a signature file for every download and the public key used to sign it. To verify a download before you use it, see Verify a downloaded release asset on GitHub.

Run validation

The command uses the following syntax:

```
cfn-validate TEMPLATE_OR_DIRECTORY [OPTIONS]
```

Check one template:

```
cfn-validate template.yaml
```

Pass a directory to recursively check every `.yaml`, `.yml`, and `.json` file:

```
cfn-validate ./templates/
```

Some checks depend on the AWS Region or on parameter values. Use `--region` to set the `AWS::Region` pseudo parameter, `--parameter` to set a template parameter, and `--pseudo-parameter` to set other pseudo parameters such as `AWS::AccountId` or `AWS::Partition`. You can repeat `--parameter` and `--pseudo-parameter`:

```
cfn-validate template.yaml --region us-west-2 --parameter Environment=prod --parameter
 InstanceType=t3.micro
```

To treat warnings as errors, for example in an automated build, add `--strict`. To list every built-in rule, run `cfn-validate --list-rules`.

Understand the results

The command writes a JSON report to standard output. Each finding in the report includes the rule ID, severity, message, source location (line and column), the resource or other template entity, and a suggested fix when one is available. The default `--format detailed` report also includes the rule description and related context. Use `--format standard` for a compact report that omits those fields.

To limit the report to more serious findings, use `--level` with the minimum severity to include:

```
cfn-validate template.yaml --level error
```

The command returns exit code `0` when it finds no errors, 1 when it finds errors, and 2 for a usage or initialization error, such as an invalid option or a file that doesn't exist. With `--strict`, warnings also cause exit code 1.

Filter the results

You can include or exclude findings by rule ID, rule ID range, category, resource type, service prefix, or the logical ID of a template entity. Each filter has an `--include-` and an `--exclude-` form. The following examples show common filters:

```
# Suppress specific rules everywhere
cfn-validate template.yaml --exclude-ids RULE_ID,RULE_ID

# Report only security findings
cfn-validate template.yaml --include-categories Security

# Suppress one rule for one resource only
cfn-validate template.yaml --exclude-logical-id MyBucket=RULE_ID

# Suppress every rule for a resource type
cfn-validate template.yaml --exclude-resource-type AWS::EC2::Instance

# Suppress every rule for a service
cfn-validate template.yaml --exclude-service AWS::AutoScaling
```

Filters that target a resource type, service, or logical ID accept an optional =*RULE_ID* suffix to limit the filter to one rule. Without the suffix, the filter applies to every rule for that target. For all filter options, see the `cfn-validate` CLI reference on GitHub.

Add custom rules

You can add your own checks in any of the following formats. Each format assumes basic familiarity with its rule language. By default, custom rules run in addition to the built-in rules. To run only your custom rules, add `--disable-builtin-rules`.

- **CEL** (`.json`) – Property and data-driven checks written as CEL expressions in a JSON file. Load with `--rule-source`.
- **Rego** (`.rego`) – Complex cross-resource logic written in the Rego policy language. Load with `--rule-source`.
- **Guard** (`.guard`) – Declarative compliance rules in the Guard rule language. Load a file or directory with `--guard-rule-source`. The validator supports a subset of the Guard language. If a rule uses an unsupported construct, the validator rejects the rule when it loads instead of ignoring the construct. For the rule syntax, see Validate templates with Guard.

You can repeat either option to load multiple rule sources:

```
cfn-validate template.yaml --rule-source ./rules/checks.json --rule-source ./rules/
network.rego
cfn-validate template.yaml --guard-rule-source ./guard-rules/
```

The following CEL example in `checks.json` reports a finding for each Amazon S3 bucket without encryption. The `resource_type` field makes the rule run once for each resource of that type, and `{name}` in the message is replaced with the resource's logical ID.

```
{
  "rules": [
    {
      "rule_id": "CUSTOM001",
      "severity": "ERROR",
      "resource_type": "AWS::S3::Bucket",
      "expression": "!has(properties.BucketEncryption)",
      "message": "S3 bucket {name} must have encryption configured",
      "prop_path": "Properties.BucketEncryption",
      "suggested_fix": "Add BucketEncryption with SSEAlgorithm: aws:kms"
    }
  ]
}
```

For more information about the data available to rules, the supported CEL functions, and writing Rego rules, see the Custom Rules Reference on GitHub.

Validate new resource types and properties

The bundled schemas match the CloudFormation registry at the time of the release. If a template uses a resource type, property, or allowed value that CloudFormation published after that release, or a private resource type from your own registry, pass the resource provider schema with `--additional-schema`. The validator uses it to extend the bundled schema for an existing type or to validate a new type. You can pass a `.json` file or a directory of schema files, and repeat the option:

```
cfn-validate template.yaml --additional-schema ./schemas/
```

An additional schema never removes a constraint from the bundled schema. If a schema can't be read or applied, the command exits with code 2 instead of ignoring it.

Embed the validation library

Install the package for your programming language to run the same offline checks from your application. Create a validation engine once, reuse it for multiple templates, and process the structured diagnostics returned for each template. You can also configure custom CEL, Rego, or Guard rules and additional schemas through the library API.

Language	Package	Install	Documentation
Rust	`cloudformation-validate` on crates.io	`cargo add cloudformation-validate`	API and examples
Node.js	`@aws/cloudformation-validate` on npm	`npm install @aws/cloudformation-validate`	API and examples
Python	`cloudformation-validate` on PyPI	`python3 -m pip install cloudformation-validate`	API and examples
Go	`cloudformation-validate/src/`	`go get github.com/aws-cloudformation/cloudf`	API and examples

Language	Package	Install	Documentation
	bindings-go/go on pkg.go.dev	ormation-validate/src/bindings-go/go@latest	
JVM (Java or Kotlin)	software.amazon.cloudformation:cloudformation-validate on Maven Central	implementation("software.amazon.cloudformation:cloudformation-validate: *VERSION*") (Gradle) or the equivalent Maven dependency	API and examples

Every library exposes one method that validates a template and returns a report with the same diagnostics as the command-line tool. The following examples use `CompositeEngine`, the default engine, which runs the built-in rules and any custom CEL, Rego, or Guard rules that you configure. Each example validates one template and prints each finding.

Python:

```
from cloudformation_validate import CompositeEngine

engine = CompositeEngine()
report = engine.validate_template("template.yaml")
for d in report.diagnostics:
    print(f"[{d.severity.name}] {d.rule_id}: {d.message}")
```

Kotlin:

```
import software.amazon.cloudformation.validate.CompositeEngine
import java.io.File

fun main() {
    val engine = CompositeEngine()
    val report = engine.validateTemplate(File("template.yaml"))
    for (d in report.diagnostics) {
        println("[${d.severity}] ${d.ruleId}: ${d.message}")
    }
}
```

For supported language runtimes, version pinning, prerelease channels, and Maven and Gradle syntax, see the language binding installation guide on GitHub. For the API reference and complete examples in each language, see Embedding as a library on GitHub.

Use with AWS CDK

If you use AWS CDK, you don't need to install the validator separately. The AWS CDK construct library includes a default `CloudFormationValidatePlugin` that runs the same checks automatically after synthesizing your CloudFormation templates.

```
cdk synth
```

AWS CDK reports possible deployment failures and best-practice findings with its validation results. You can explicitly configure `CloudFormationValidatePlugin` when you want to add custom Rego or Guard rules.

For plugin configuration, acknowledgments, and validation reporting, see Template and Policy Validation in the *AWS CDK API Reference*.

Validate templates with Guard

AWS CloudFormation Guard (`cfn-guard`) is a policy-as-code tool. You write rules in the Guard language that describe required or prohibited configurations, and then Guard checks JSON or YAML data against those rules. For example, you can require every Amazon S3 bucket in a template to use encryption, or block security group rules that open port 22 to the internet.

Guard works with any JSON or YAML data. In addition to CloudFormation templates, you can check CloudFormation change sets and AWS Config configuration items. This topic focuses on checking templates.

For release notes and the complete command reference, see Guard on GitHub. For the complete language reference, see the Guard User Guide.

For the limitations of local validation and steps to take before deployment, see Understand validation scope.

Topics

- Install Guard
- Understand the Guard language

- Write a rule
- Validate a template
- Test your rules
- Generate rules from a template
- Add Guard to your workflow

Install Guard

On macOS, install Guard with Homebrew:

```
brew install cloudformation-guard
```

On Linux or macOS, you can instead run the installation script. The script installs the latest release to `~/.guard/bin`. Add that directory to your `PATH` after the script finishes.

```
curl --proto '=https' --tlsv1.2 -sSf https://raw.githubusercontent.com/aws-cloudformation/cloudformation-guard/main/install-guard.sh | sh
```

If you have a Rust toolchain, you can also install Guard with Cargo:

```
cargo install cfn-guard
```

To confirm the installation, print the installed version:

```
cfn-guard --version
```

For Windows installation, a Docker image, and release verification, see Setting up Guard.

Understand the Guard language

Now that Guard is installed, you can start writing rules. Guard rules are built from the following parts:

- **Queries** – Select values from the data with a dotted path, such as `Resources.MyBucket.Properties.BucketName`. Use `*` to select every value in a map, and `[*]` to select every item in a list. Add a filter in square brackets to narrow the selection. For example, `Resources.*[ Type == 'AWS::S3::Bucket' ]` selects every Amazon S3 bucket.

- **Clauses** – Make an assertion that evaluates to true or false. A clause combines a query with an operator, such as `==`, `!=`, `>=`, `in`, `exists`, `empty`, or `is_string`. Clauses on separate lines must all pass. To accept either of two clauses, join them with `or`.
- **Variables** – Store the result of a query with `let` and reference it later with a `%` prefix. Variables let you select a set of resources once and reuse it in several rules.
- **Named rules** – Group clauses in a `rule` block with a name. Add a `when` condition to skip the rule when it doesn't apply, such as when a template has no Amazon S3 buckets. Rules can reference other rules, so you can build complex policies from small pieces.
- **Custom messages** – Add text between << and >> after a clause to explain a failure in the output.

For the complete syntax, see Writing Guard rules.

Write a rule

Save rules in a file with the `.guard` extension. The following example in `rules.guard` requires every Amazon S3 bucket in a template to specify bucket encryption and to block public access:

```
let s3_buckets = Resources.*[ Type == 'AWS::S3::Bucket' ]

rule S3_BUCKET_ENCRYPTED when %s3_buckets !empty {
  %s3_buckets.Properties.BucketEncryption exists
  <<
    Violation: S3 buckets must specify BucketEncryption.
    Fix: Add a BucketEncryption property with ServerSideEncryptionConfiguration.
  >>
}

rule S3_BUCKET_PUBLIC_ACCESS_BLOCKED when %s3_buckets !empty {
  %s3_buckets.Properties.PublicAccessBlockConfiguration {
    BlockPublicAcls == true
    BlockPublicPolicy == true
    IgnorePublicAcls == true
    RestrictPublicBuckets == true
  }
}
```

The first line stores every Amazon S3 bucket in the `s3_buckets` variable. Each rule runs only when the template contains at least one bucket. The first rule checks that the `BucketEncryption` property exists and includes a message that Guard prints when the check fails. The second rule

checks four properties inside `PublicAccessBlockConfiguration`. Because the clauses are on separate lines, all four must pass.

You can keep many rules in one file, or split rules across files by topic, such as one file for storage rules and another for network rules. Guard evaluates every `.guard` file in a directory when you pass the directory to a command.

Validate a template

Pass the rule file to `--rules` and the template to `--data`. Add `--type CFNTemplate` so that Guard reports findings by resource logical ID.

```
cfn-guard validate --rules rules.guard --data template.yaml --type CFNTemplate
```

When the template passes, Guard prints a PASS status and returns exit code `0`. When a rule fails, Guard lists the failed rules and, for each noncompliant resource, the check that failed, the property path, and the reason. The output for a template whose bucket is missing `BucketEncryption` starts like the following:

```
template.yaml Status = FAIL
FAILED rules
rules.guard/S3_BUCKET_ENCRYPTED    FAIL
---
Evaluating data template.yaml against rules rules.guard
Number of non-compliant resources 1
Resource = MyBucket {
  Type      = AWS::S3::Bucket
  Rule = S3_BUCKET_ENCRYPTED {
    ...
```

Guard returns exit code 19 when one or more rules fail and 5 when it can't parse a rule or data file. You can use these exit codes to stop an automated build.

The following options are useful when you validate templates:

- Pass `--rules` or `--data` more than once, or pass a directory, to check multiple rule files or templates in one run. A directory that you pass to `--data` must contain only data files, and a directory that you pass to `--rules` must contain only rule files.
- Use `--show-summary` with `pass`, `fail`, `skip`, `all`, or none to control which rules appear in the summary table. The default is `fail`.

- Use `--output-format` with `json`, `yaml`, `junit`, or `sarif` to produce output for other tools. The `junit` and `sarif` formats work with most build systems and code scanning tools.
- Use `--input-parameters` to pass a JSON or YAML file with values that your rules reference, such as a list of approved security group IDs. This keeps environment-specific values out of your rules.

For all options, see validate in the *Guard User Guide*.

Test your rules

Before you use a rule in an automated workflow, test it with the built-in unit testing support. A test file lists sample inputs and the result that you expect for each rule: `PASS`, `FAIL`, or `SKIP`. The following example in `rules_tests.yaml` tests the `S3_BUCKET_ENCRYPTED` rule from the previous example:

```
---
- input:
    Resources: {}
  expectations:
    rules:
      S3_BUCKET_ENCRYPTED: SKIP
- input:
    Resources:
      MyBucket:
        Type: AWS::S3::Bucket
        Properties: {}
  expectations:
    rules:
      S3_BUCKET_ENCRYPTED: FAIL
- input:
    Resources:
      MyBucket:
        Type: AWS::S3::Bucket
        Properties:
          BucketEncryption:
            ServerSideEncryptionConfiguration:
              - ServerSideEncryptionByDefault:
                  SSEAlgorithm: aws:kms
  expectations:
    rules:
```

```
    S3_BUCKET_ENCRYPTED: PASS
```

Run the tests with the `test` command:

```
cfn-guard test --rules-file rules.guard --test-data rules_tests.yaml
```

Guard prints PASS or FAIL for each expectation and returns exit code 7 when any expectation fails. For more information, see Testing Guard rules.

Generate rules from a template

To get started quickly, you can generate rules from a template that already meets your requirements. The `rulegen` command reads the template and writes a rule for each resource type that requires the property values found in the template:

```
cfn-guard rulegen --template template.yaml --output generated.guard
```

Review the generated file and keep only the checks that represent your policy. Generated rules are a starting point and often require the exact values in the source template, which might be too strict for other templates.

Add Guard to your workflow

You can run Guard in the following places:

- **In your editor** – The CloudFormation Language Server runs Guard rule packs while you edit a template and shows failures inline. For setup, see CloudFormation Language Server.
- **Before a Git commit** – Guard provides a `pre-commit` hook that can run `validate` on changed templates or `test` on changed rules. Add the following to your `.pre-commit-config.yaml` file and replace the paths with your own:

```
repos:
  - repo: https://github.com/aws-cloudformation/cloudformation-guard
    rev: pre-commit-v0.0.2
    hooks:
      - id: cfn-guard
        args:
          - --operation=validate
          - --rules=rules/
```

```
        files: ^templates/.*
```

- **In an automated build** – Run `cfn-guard validate` as a build step and use the exit code to fail the build. Use `--output-format junit` or `--output-format sarif` to publish the results. A GitHub Action and a container image are also available. Pull the image from `public.ecr.aws/aws-cloudformation/cloudformation-guard`.
- **With `cloudformation-validate`** – The `cfn-validate` command and the AWS CDK validation plugin can run a supported subset of Guard rules together with their built-in checks. For more information, see Validate templates with `cloudformation-validate`.
- **During stack operations** – To enforce rules when CloudFormation or Cloud Control API creates, updates, or deletes resources, use Guard Hooks. For more information, see Guard Hooks.

For ready-to-use rules that cover common security and compliance controls, see the AWS Guard Rules Registry on GitHub.

Edit templates visually with template studio

Template studio is a template editor in the CloudFormation console. It shows your template as text beside a diagram of the resources that template declares. As you edit the template, the diagram redraws to match. You can validate the template, convert it between YAML and JSON, and save it to Amazon S3 without leaving the console.

The template text is authoritative. The diagram is a visual representation of that text, not a separate editable surface. You make all changes in the template.

Considerations

Keep the following in mind when you work in template studio:

- You author in the template, not on the diagram. The diagram shows the resources your template declares and how they refer to each other; you can't add or configure a resource by editing the diagram.
- Your edits are not saved automatically. Template studio asks for confirmation before you navigate away with unsaved changes. If you leave or refresh the page, unsaved changes are discarded. Save to Amazon S3, or copy or download the template, before you leave.
- Converting a template from YAML to JSON discards its comments, because JSON has no syntax for them. Converting back to YAML doesn't restore them.

- CloudFormation validation in template studio isn't currently available for drafts larger than 51,200 bytes (50 KiB). The current Validate action sends the draft through the `TemplateBody` parameter, which has this limit. This isn't the overall template-size limit. You can still save the template and use it in a stack operation, subject to CloudFormation quotas. For more information, see Validate a template.
- Saving writes the template to a bucket in your account, so templates you save are billed at standard Amazon S3 rates. For more information, see Amazon S3 pricing.
- You can download the resources diagram only when the canvas is displayed in the **Canvas** or **Both** view.

Permissions required for template studio

The permissions you need depend on the template source and the actions you perform:

`cloudformation:GetTemplate`

Required to open an existing stack's template. Grant access to the stack whose template you want to edit.

`s3:GetObject`

Required to read a template from Amazon S3 using your credentials. Grant access to the source template object. A presigned URL uses the authorization included in the URL.

`cloudformation:ValidateTemplate`

Required to choose **Validate**.

`s3:PutObject`

Required to save a template. Scope this permission to objects in the destination bucket, for example, `arn:aws:s3:::destination-bucket/*`, rather than all resources.

`s3:CreateBucket`

Required only when the console creates a missing destination bucket. This can be the console-managed bucket or a bucket name that you specify. Reusing an existing bucket does not require this permission.

`s3:ListBucket`

Required when saving to an existing bucket that you specify. Template studio uses `HeadBucket` to verify access, the expected bucket owner, and the bucket AWS Region before uploading.

For the console-managed bucket, this permission is optional. Without it, the save dialog still displays the bucket name. Saving requires `s3:PutObject` and, if the bucket must be created, `s3:CreateBucket`.

Open template studio

Where you open template studio from determines what it opens with, and what happens after you save.

To start a new template

1. Open the CloudFormation console.
2. On the navigation bar at the top of the screen, choose the AWS Region you intend to deploy in.
3. In the navigation pane, choose **Template studio**.

To build a template while creating a stack

1. On the **Create stack** page, choose **Build with template studio**.
2. Build the template, then choose **Save and continue**.
3. The stack wizard continues with the template you saved, so you don't have to upload it separately.

To edit the template of an existing stack

1. On the **Stacks** page, choose the name of the stack.
2. Either update the stack directly or create a change set.
3. On the **Prepare template** page, choose **Open in template studio**. The stack's current template opens.
4. Choose **Save and continue** to return to the update or change set with your edited template.

Elsewhere in the console, wherever a template is displayed, you can choose **Open in template studio** to work on it.

Work with the template and the diagram

Template studio opens with the template and the diagram side by side. Use the view control to choose what is on screen:

Template

The template only, at full width. Use this to read or write a template without the diagram.

Canvas

The diagram only, at full width. Use this for a large template where the diagram requires the full width.

Both

The template and the diagram together. Drag the divider between them to give either side more room.

The diagram groups related resources and draws a connection wherever one resource refers to another through `Ref`, `Fn::GetAtt`, `Fn::Sub`, or `DependsOn`. A resource that nothing references appears on its own, which makes an accidentally disconnected resource easy to spot.

> **Note**
>
> For example, three functions can assume the same IAM role. The diagram shows the role inside each function, but the template declares only one role. Check the template to see how many resources exist.

Convert between YAML and JSON

The format you save in determines the file name and content type of the object that template studio writes to Amazon S3.

To convert the template you are editing

1. Choose the format control, which names the format you are currently in.
2. Choose the other format.
3. Converting from YAML to JSON asks you to confirm, because comments can't be carried over. Choose **Convert** to continue.

Important

JSON has no syntax for comments. When you convert a YAML template that contains comments to JSON, those comments are removed, and converting back to YAML doesn't restore them. Copy anything you want to keep before you convert.

The format control is unavailable while the template can't be read, because there is nothing complete enough to convert. Fix the template first.

Syntax problems as you type

The editor checks the template's syntax while you write it and underlines what it can't read: an unclosed bracket, a misplaced indent, a duplicate key. Hover over an underline to see the problem. Where one line has several problems, they are numbered so you can tell them apart.

This is a check of the template's *syntax*, not of its contents. The editor doesn't know which resource types exist or which properties they take, so a template that is perfectly well-formed YAML can still be wrong in ways only CloudFormation can tell you about. Choose **Validate** for that.

Validate a template

Validation checks the template's syntax with the editor and checks the template with CloudFormation before you use it in a stack operation. Both checks use the current draft, including changes you haven't saved. A successful CloudFormation check doesn't override syntax errors reported by the editor.

Note

Template studio sends the draft to the `ValidateTemplate` API using the `TemplateBody` parameter. This parameter has a 51,200-byte (50 KiB) limit.
For a draft above this limit, template studio doesn't send the CloudFormation request. The CloudFormation check is unavailable, but this doesn't mean the template is invalid. The editor can still report syntax errors.
The same API supports templates up to 1 MB through `TemplateURL`. Template studio doesn't upload edited drafts for URL-based validation. You can save the template to Amazon S3 and validate it through a stack operation, subject to CloudFormation quotas.

To validate the template you are editing

1. Choose **Validate**.
2. A template that passes both checks is reported as **Valid**.
3. If either check finds errors, review them in the validation message. Editor errors include line and column numbers. Correct the errors in the editor and validate again.

Save a template to Amazon S3

CloudFormation reads templates for stack operations from Amazon S3, so saving is what makes your template available to the rest of the console.

Use dynamic references to secrets stored in Secrets Manager or secure strings in Systems Manager Parameter Store instead of including plaintext secrets in saved templates. For more information, see Get values stored in other services using dynamic references.

To save the template you are editing

1. Choose **Save and continue**.
2. Check the destination. By default, the template goes to the Amazon S3 bucket that CloudFormation manages for uploaded templates in the current AWS Region. If that bucket doesn't exist yet, CloudFormation creates it when you save and reuses it for templates you save later.
   - To choose your own destination instead, choose **Use a different bucket** and enter a **Bucket name**. The bucket is created if it doesn't already exist.
3. Choose **Confirm and continue**.

> **Note**
>
> Templates stored in Amazon S3 are billed at standard Amazon S3 rates, whichever bucket holds them. For more information, see Amazon S3 pricing.

Saving does not require a successful validation result. When validation finds errors, template studio asks you to confirm that you want to save anyway. Saving does not fix those errors; review and correct them before using the template in a stack operation.

Copy or download a template or diagram

Choose **Template actions** to take the template or the diagram out of the console:

Copy template

Copies the template as it appears in the editor to your clipboard.

Download template

Saves the template as a file, in the format you are currently editing in.

Download resources diagram

Saves the diagram as a PNG image. Because the image is a capture of the diagram, this is available only when the diagram is on screen.

Generate templates from existing resources with IaC generator

With the CloudFormation infrastructure as code generator (IaC generator), you can generate a template using AWS resources provisioned in your account that are not already managed by CloudFormation.

The following are benefits of the IaC generator:

- Bring entire applications under CloudFormation management or migrate them into an AWS CDK app.
- Generate templates without having to describe a resource property by property and then translate that into JSON or YAML syntax.
- Use the template to replicate resources in a new account or Region.

The IaC generation process consists of the following steps:

1. **Scan resources** – The first step is to start a scan of your resources. This scan is region-wide and expires after 30 days. During this time, you can create multiple templates from the same scan.
2. **Create your template** – To create the template, you have two options:
   - Create a new template from scratch and add the scanned resources and related resources to it.
   - Use an existing CloudFormation stack as a starting point and add the scanned resources and related resources to its template.

3. **Import resources** – Use your template to import the resources as a CloudFormation stack or migrate them into an AWS CDK app.

The IaC generator feature is available in all commercial Regions and supports many common AWS resource types. For a full list of supported resources, see Resource type support.

Topics

- Considerations
- IAM permissions required for scanning resources
- Commonly used commands for template generation, management, and deletion
- Migrate a template to the AWS CDK
- Start a resource scan with CloudFormation IaC generator
- View the scan summary in the CloudFormation console
- Create a CloudFormation template from resources scanned with IaC generator
- Create a CloudFormation stack from scanned resources
- Resolve write-only properties

Considerations

You can generate JSON or YAML templates for AWS resources that you have read access to. The templates for the IaC generator capability models cloud resources reliably and quickly without having to describe a resource property by property.

The following table lists the quotas available for the IaC generation feature.

Name	Full scan	Partial scan
Maximum number of resources that can be processed in a scan	100,000	100,000
Number of scans per day (for scans with less than 10,000 resources)	10	10
Number of scans per day (for scans with more than 10,000 resources)	1	1

Name	Full scan	Partial scan
Concurrent number of templates generating per account	5	5
Concurrent number of resources modeled for one template generation	5	5
Total number of resources that can be modeled in one template	500	500
Maximum number of generated templates per account	1,000	1,000

Important

IaC generator only supports AWS resources that are supported by Cloud Control API in your Region. For more information, see Resource type support.

IAM permissions required for scanning resources

To scan resources with IaC generator, your IAM principal (user, role, or group) must have:

- CloudFormation scanning permissions
- Read permissions for target AWS services

The scan scope is limited to resources you have read access to. Missing permissions won't cause scan failure but will exclude those resources.

For an example IAM policy that grants scanning and template management permissions, see Allow all IaC generator operations.

Commonly used commands for template generation, management, and deletion

The commonly used commands for working with IaC generator include:

- start-resource-scan to start a scan of the resources in the account in an AWS Region.
- describe-resource-scan to monitor the progress of a resource scan.

- list-resource-scans to list the resource scans in an AWS Region.
- list-resource-scan-resources to list the resources found during the resource scan.
- list-resource-scan-related-resources to list the resources related to your scanned resources.
- create-generated-template to generate a CloudFormation template from a set of scanned resources.
- update-generated-template to update the generated template.
- describe-generated-template to return information about a generated template.
- list-generated-templates to list all generated templates in your account and current Region.
- delete-generated-template to delete a generated template.

Migrate a template to the AWS CDK

The AWS Cloud Development Kit (AWS CDK) is an open-source software development framework that you can use to develop, manage, and deploy CloudFormation resources using popular programming languages.

The AWS CDK CLI provides an integration with IaC generator. Use the AWS CDK CLI `cdk migrate` command to convert the CloudFormation template and create a new CDK app that contains your resources. Then, you can use the AWS CDK to manage your resources and deploy to CloudFormation.

For more information, see Migrate to AWS CDK in the *AWS Cloud Development Kit (AWS CDK) Developer Guide*.

Start a resource scan with CloudFormation IaC generator

Before you create a template from existing resources, you first must initiate a resource scan to discover your current resources and their relationships.

You can start a resource scan using one of the following options. For first-time users of IaC generator, we recommend the first option.

- **Scan all resources (full scan)** – Scans all existing resources in the current account and Region. This scanning process can take up to 10 minutes for 1,000 resources.
- **Scan specific resources (partial scan)** – Manually select which resource types to scan in the current account and Region. This option provides a faster and more focused scanning process, making it ideal for iterative template development.

After the scan completes, you can choose which resources and their related resources to include when generating your template. When using partial scanning, related resources will only be available during template generation if either:

- You specifically selected them before starting the scan, or
- They were required to discover your selected resource types.

For example, if you select `AWS::EKS::Nodegroup` without selecting `AWS::EKS::Cluster`, IaC generator automatically includes `AWS::EKS::Cluster` resources in the scan because discovering the node group requires discovering the cluster first. In all other cases, the scan will only include the resources you specifically select.

> **Note**
>
> Before you continue, confirm that you have the permissions required to work with IaC generator. For more information, see IAM permissions required for scanning resources.

Topics

- Start a resource scan (console)
- Start a resource scan (AWS CLI)

Start a resource scan (console)

To start a resource scan of all resource types (full scan)

1. Open the IaC generator page of the CloudFormation console.
2. On the navigation bar at the top of the screen, choose the AWS Region that contains the resources to scan.
3. From the **Scans** panel, choose **Start a new scan** and then choose **Scan all resources**.

To start a resource scan of specific resource types (partial scan)

1. Open the IaC generator page of the CloudFormation console.
2. On the navigation bar at the top of the screen, choose the AWS Region that contains the resources to scan.

3. From the **Scans** panel, choose **Start a new scan** and then choose **Scan specific resources**.
4. In the **Start partial scan** dialog box, select up to 100 resource types, and then choose **Start scan**.

Start a resource scan (AWS CLI)

To start a resource scan of all resource types (full scan)

Use the following start-resource-scan command. Replace *us-east-1* with the AWS Region that contains the resources to scan.

```
aws cloudformation start-resource-scan --region us-east-1
```

If successful, this command returns the ARN of the scan. Note the ARN in the `ResourceScanId` property. You need it to create your template.

```
{
    "ResourceScanId":
      "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-
a3ef-3e6ecfa5da60"
}
```

To start a resource scan of specific resource types (partial scan)

1. Use the following cat command to store the resource types you want to scan in a JSON file named `config.json` in your home directory. The following is an example scanning configuration that scans for Amazon EC2 instances, security groups, and all Amazon S3 resources.

   ```
   $ cat > config.json
   [
     {
       "Types":[
         "AWS::EC2::Instance",
         "AWS::EC2::SecurityGroup",
         "AWS::S3::*"
       ]
     }
   ]
   ```

2. Use the start-resource-scan command with the `--scan-filters` option, along with the `config.json` file you created, to start the partial scan. Replace *us-east-1* with the AWS Region that contains the resources to scan.

```
aws cloudformation start-resource-scan --scan-filters file://config.json --
region us-east-1
```

If successful, this command returns the ARN of the scan. Note the ARN in the `ResourceScanId` property. You need it to create your template.

```
{
    "ResourceScanId":
      "arn:aws:cloudformation:region:account-id:resourceScan/0a699f15-489c-43ca-
a3ef-3e6ecfa5da60"
}
```

To monitor the progress of a resource scan

Use the describe-resource-scan command. For the `--resource-scan-id` option, replace the sample ARN with the actual ARN.

```
aws cloudformation describe-resource-scan --region us-east-1 \
  --resource-scan-id arn:aws:cloudformation:us-
east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60
```

If successful, this command returns output similar to the following:

```
{
    "ResourceScanId": "arn:aws:cloudformation:region:account-
id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60",
    "Status": "COMPLETE",
    "StartTime": "2023-08-21T03:10:38.485000+00:00",
    "EndTime": "2023-08-21T03:20:28.485000+00:00",
    "PercentageCompleted": 100.0,
    "ResourceTypes": [
        "AWS::CloudFront::CachePolicy",
        "AWS::CloudFront::OriginRequestPolicy",
        "AWS::EC2::DHCPOptions",
        "AWS::EC2::InternetGateway",
        "AWS::EC2::KeyPair",
```

```
        "AWS::EC2::NetworkAcl",
        "AWS::EC2::NetworkInsightsPath",
        "AWS::EC2::NetworkInterface",
        "AWS::EC2::PlacementGroup",
        "AWS::EC2::Route",
        "AWS::EC2::RouteTable",
        "AWS::EC2::SecurityGroup",
        "AWS::EC2::Subnet",
        "AWS::EC2::SubnetCidrBlock",
        "AWS::EC2::SubnetNetworkAclAssociation",
        "AWS::EC2::SubnetRouteTableAssociation",
        ...
    ],
    "ResourcesRead": 676
}
```

For a partial scan, the output will look similar to the following:

```
{
    "ResourceScanId": "arn:aws:cloudformation:region:account-
id:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60",
    "Status": "COMPLETE",
    "StartTime": "2025-03-06T18:24:19.542000+00:00",
    "EndTime": "2025-03-06T18:25:23.142000+00:00",
    "PercentageCompleted": 100.0,
    "ResourceTypes": [
        "AWS::EC2::Instance",
        "AWS::EC2::SecurityGroup",
        "AWS::S3::Bucket",
        "AWS::S3::BucketPolicy"
    ],
    "ResourcesRead": 65,
    "ScanFilters": [
        {
            "Types": [
                "AWS::EC2::Instance",
                "AWS::EC2::SecurityGroup",
                "AWS::S3::*"
            ]
        }
    ]
}
```

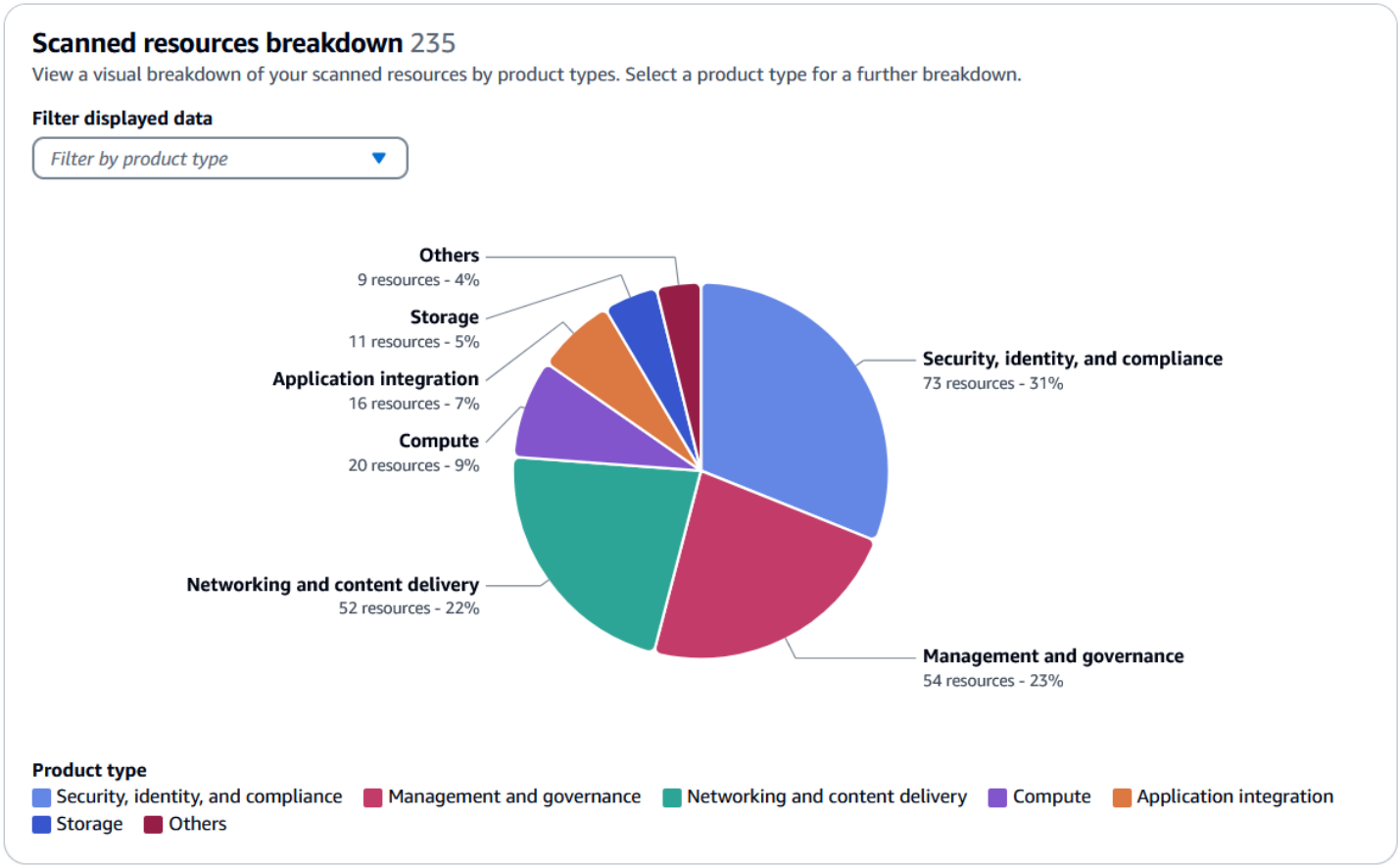
For a description of the fields in the output, see DescribeResourceScan in the *AWS CloudFormation API Reference*.

View the scan summary in the CloudFormation console

After the scan completes, you can view a visualization of resources found during the scan to help you identify the concentration of resources across different product types.

To view information about resources found during the scan

1. Open the IaC generator page of the CloudFormation console.
2. On the navigation bar at the top of the screen, choose the AWS Region that contains the resource scan to view.
3. From the navigation pane, choose **IaC generator**.
4. Under **Scanned resources breakdown**, you'll find a visual breakdown of the scanned resources by product type, for example, **Compute** and **Storage**.
5. To customize the number of product types displayed, choose **Filter displayed data**. This helps you tailor the visualization to focus on the product types that you're most interested in.
6. On the right side of the page is the **Scan summary details** panel. To open the panel, choose the **open panel** icon.



Create a CloudFormation template from resources scanned with IaC generator

This topic explains how to create a template from resources that were scanned using the IaC generator feature.

Create a template from scanned resources (console)

To create a stack template from scanned resources

1. Open the IaC generator page of the CloudFormation console.
2. On the navigation bar at the top of the screen, choose the AWS Region that contains the scanned resources.
3. From the **Templates** section, choose **Create template**.
4. Choose **Start from a new template**.

    a. For **Template name**, provide a name for your template.

b. (Optional) Configure your **Deletion policy** and **Update replace policy**.

c. Choose **Next** to add scanned resources to the template.

5. For **Add scanned resources**, browse the list of scanned resources and select the resources you want to add to your template. You can filter the resources by resource identifier, resource type, or tags. The filters are mutually inclusive.
6. When you've added all needed resources to your template, choose **Next** to exit the **Add scanned resources** page and proceed to the **Add related resources** page.
7. Review a recommended list of related resources. Related resources, such as Amazon EC2 instances and security groups, are interdependent and typically belong to the same workload. Select the related resources that you want to include in the generated template.

> **Note**
>
> We suggest that you add all related resources to this template.

8. Review the template details, scanned resources, and related resources.
9. Choose **Create template** to exit the **Review and create** page and create the template.

Create a template from scanned resources (AWS CLI)

To create a stack template from scanned resources

1. Use the list-resource-scan-resources command to list the resources found during the scan, optionally specifying the `--resource-identifier` option to limit the output. For the `--resource-scan-id` option, replace the sample ARN with the actual ARN.

```
aws cloudformation list-resource-scan-resources \
  --resource-scan-id arn:aws:cloudformation:us-
east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60 \
  --resource-identifier MyApp
```

The following is an example response, where `ManagedByStack` indicates whether CloudFormation manages the resource already. Copy the output. You need it for the next step.

```
{
    "Resources": [
        {
```

```
            "ResourceType": "AWS::EKS::Cluster",
            "ResourceIdentifier": {
                "ClusterName": "MyAppClusterName"
            },
            "ManagedByStack": false
        },
        {
            "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
            "ResourceIdentifier": {
                "AutoScalingGroupName": "MyAppASGName"
            },
            "ManagedByStack": false
        }
    ]
}
```

For a description of the fields in the output, see ScannedResource in the *AWS CloudFormation API Reference*.

2. Use the `cat` command to store the resource types and identifiers in a JSON file named `resources.json` in your home directory. The following is example JSON based on the example output in the previous step.

```
$ cat > resources.json
[
    {
        "ResourceType": "AWS::EKS::Cluster",
        "ResourceIdentifier": {
            "ClusterName": "MyAppClusterName"
        }
    },
    {
        "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
        "ResourceIdentifier": {
            "AutoScalingGroupName": "MyAppASGName"
        }
    }
]
```

3. Use the list-resource-scan-related-resources command, along with the `resources.json` file you created, to list the resources related to your scanned resources.

```
aws cloudformation list-resource-scan-related-resources \
```

```
  --resource-scan-id arn:aws:cloudformation:us-
east-1:123456789012:resourceScan/0a699f15-489c-43ca-a3ef-3e6ecfa5da60 \
  --resources file://resources.json
```

The following is an example response, where `ManagedByStack` indicates whether CloudFormation manages the resource already. Add these resources to the JSON file you created in the previous step. You'll need it to create your template.

```
{
    "RelatedResources": [
        {
            "ResourceType": "AWS::EKS::Nodegroup",
            "ResourceIdentifier": {
                "NodegroupName": "MyAppNodegroupName"
            },
            "ManagedByStack": false
        },
        {
            "ResourceType": "AWS::IAM::Role",
            "ResourceIdentifier": {
                "RoleId": "arn:aws::iam::account-id:role/MyAppIAMRole"
            },
            "ManagedByStack": false
        }
    ]
}
```

For a description of the fields in the output, see ScannedResource in the *AWS CloudFormation API Reference*.

> **Note**
>
> The input list of resources can't exceed a length of 100. To list related resources for more than 100 resources, run the **list-resource-scan-related-resources** command in batches of 100 and consolidate the results.
> Be aware that the output may contain duplicated resources in the list.

4. Use the create-generated-template command to create a new stack template, as follows, with these modifications:

- Replace *us-east-1* with the AWS Region that contains the scanned resources.
- Replace *MyTemplate* with the name of the template to create.

```
aws cloudformation create-generated-template --region us-east-1 \
 --generated-template-name MyTemplate \
  --resources file://resources.json
```

The following is an example `resources.json` file.

```
[
    {
        "ResourceType": "AWS::EKS::Cluster",
        "LogicalResourceId":"MyCluster",
        "ResourceIdentifier": {
            "ClusterName": "MyAppClusterName"
        }
    },
    {
        "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
        "LogicalResourceId":"MyASG",
        "ResourceIdentifier": {
            "AutoScalingGroupName": "MyAppASGName"
        }
    },
    {
        "ResourceType": "AWS::EKS::Nodegroup",
        "LogicalResourceId":"MyNodegroup",
        "ResourceIdentifier": {
            "NodegroupName": "MyAppNodegroupName"
        }
    },
    {
        "ResourceType": "AWS::IAM::Role",
        "LogicalResourceId":"MyRole",
        "ResourceIdentifier": {
            "RoleId": "arn:aws::iam::account-id:role/MyAppIAMRole"
        }
    }
]
```

If successful, this command returns the following.

```
{
  "Arn":
    "arn:aws:cloudformation:region:account-id:generatedtemplate/7fc8512c-d8cb-4e02-
b266-d39c48344e48",
  "Name": "MyTemplate"
}
```

Create a CloudFormation stack from scanned resources

After you create your template, use the **Canvas** view to preview its resources and their relationships before creating the stack and importing the scanned resources.

To create the stack and import the scanned resources

1. Open the IaC generator page of the CloudFormation console.
2. On the navigation bar at the top of the screen, choose the AWS Region for your template.
3. Choose the **Templates** tab, and then choose the name of your template to view more information.
4. On the **Template definition** tab, at the top of the **Template** section, you can switch the template from YAML to JSON syntax based on your preference.
5. Review the details of your template to make sure everything is set up correctly. To make it easier to review and understand the template, you can switch from the default code view to a graphical view of the infrastructure described in the template. To do so, under **Template**, choose **Canvas** instead of **Template**.

   Canvas actions

   - To focus on the details of a specific resource within your template, double-click a card to bring up the **Resource properties** panel.
   - To visually arrange and organize cards on the canvas, choose **Arrange** from the upper left of the canvas.
   - To zoom in and out of your canvas, use the zoom controls in the lower right of the canvas.
6. To view a specific resource in the console, choose the **Template resources** tab and then choose the physical ID of the resource you want to look at. This takes you to the console for that

specific resource. You can also add, remove, and resync resources in the template definition from the **Template resources** tab.

7. On the **Template definition** tab, IaC generator might issue warnings about resources that contain write-only properties. After reviewing the warnings, you can download the generated template and make any necessary changes. For more information, see Resolve write-only properties.
8. When you are satisfied with your template definition, on the **Template definition** tab, choose **Import to stack** and then choose **Next**.
9. On the **Specify stack** panel of the **Specify stack details** page, enter the name of your stack, and then choose **Next**.
10. Review and enter the parameters for the stack. Choose **Next**.
11. Review your options on the **Review changes** page and choose **Next**.
12. Review your details on the **Review and import** page and choose **Import resources**.

Resolve write-only properties

With the CloudFormation IaC generator, you can generate a template using resources provisioned in your account that are not already managed by CloudFormation. However, certain resource properties are designated as *write-only*, meaning they can be written but can't be read by CloudFormation, for example, a database password.

When generating CloudFormation templates from existing resources, write-only properties pose a challenge. In most cases, CloudFormation converts these properties into parameters in the generated template. This allows you to enter the properties as parameter values during import operations. However, there are scenarios where this conversion is not possible, and CloudFormation handles these cases differently.

Mutually exclusive properties

Some resources have multiple sets of mutually exclusive properties, at least some of which are write-only. In these cases, the IaC generator can't determine which set of exclusive properties was applied to the resource during creation. For example, you can provide the code for a AWS::Lambda::Function using one of these sets of properties.

- `Code/S3Bucket`, `Code/S3Key`, and optionally `Code/S3ObjectVersion`
- `Code/ImageUri`

- `Code/ZipFile`

All of these properties are write-only. The IaC generator selects one of the exclusive sets of properties and adds them to the generated template. Parameters are added for each of the write-only properties. The parameter names include `OneOf` and the parameter descriptions indicate that the corresponding property can be replaced with other exclusive properties. The IaC generator sets a warning type of `MUTUALLY_EXCLUSIVE_PROPERTIES` for the included properties.

Mutually exclusive types

In some cases, a write-only property can be of multiple data types. For example, the `Body` property of AWS::ApiGateway::RestApi can be either an `object` or a `string`. When this is the case, the IaC generator includes the property in the generated template using the type of `string` and sets a warning type of `MUTUALLY_EXCLUSIVE_TYPES`.

Array properties

If a write-only property has a type of `array`, the IaC generator can't include it in the generated template because parameters can only be scalar values. In this case, the property is omitted from the template, and a warning type of `UNSUPPORTED_PROPERTIES` is set.

Optional properties

For optional write-only properties, the IaC generator can’t detect if the property was used when setting up the resource. In this case, the property is omitted from the generated template, and a warning type of `UNSUPPORTED_PROPERTIES` is set.

Warnings and next steps

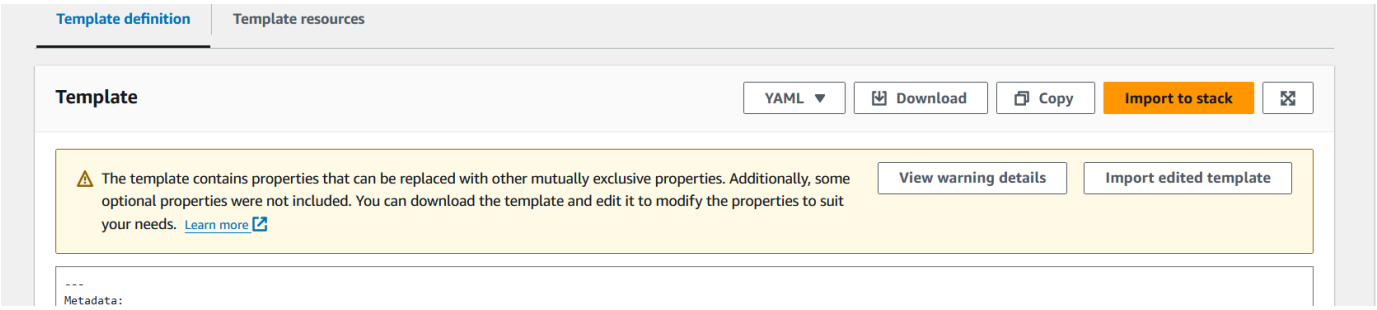
To determine which properties are write-only, you must look at the warnings returned by the IaC generator console. The AWS resource and property types reference doesn't indicate if a property is write-only, or if it supports multiple types.

Alternatively, you can see which properties are write-only from the resource provider schemas. To download the resource provider schemas, see the CloudFormation resource provider schemas.

To resolve issues with write-only properties

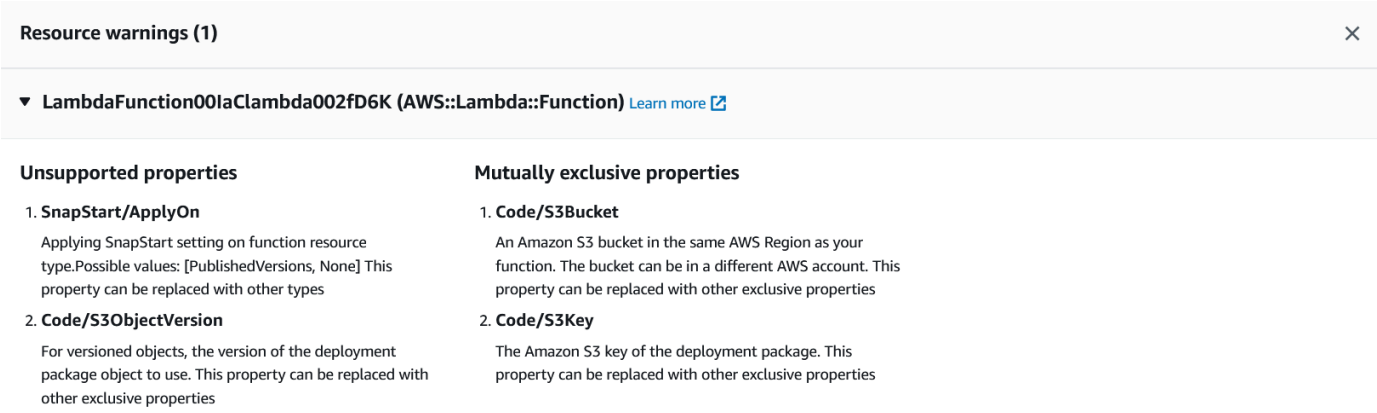
1. Open the IaC generator page of the CloudFormation console.

2. On the navigation bar at the top of the screen, choose the AWS Region for your template.
3. Choose the **Templates** tab, and then choose the name of the template you created.
4. On the **Template definition** tab, when the generated template includes resources with write-only properties, the IaC generator console displays a warning with a summary of the type of issues. For example:



5. Choose **View warning details** for more details. The resources with write-only properties are identified by the logical ID used in the generated template and resource type.

   Use the list of warnings to identify resources with write-only properties and look at each resource to determine what changes (if any) need to be made to the generated template.



6. If your template must be updated to resolve issues with write-only properties, do the following:

   a. Choose **Download** to download a copy of the template.

   b. Edit your template.

   c. When the changes are complete, you can choose the **Import edited template** button to continue the import process.

How to resolve issues with write-only properties in AWS::ApiGateway::RestAPI resources

This topic explains how to resolve issues with write-only properties in AWS::ApiGateway::RestApi resources when using the IaC generator.

Issue

When a generated template contains `AWS::ApiGateway::RestApi` resources, then warnings are generated stating that `Body`, `BodyS3Location`, and `CloneFrom` properties are identified as `UNSUPPORTED_PROPERTIES`. This is because these are optional write-only properties. The IaC generator doesn't know whether these properties were ever applied to the resource. Therefore, it omits these properties in the generated template.

Resolution

To set the `Body` property for your REST API, update your generated template.

1. Use the Amazon API Gateway GetExport API action to download the API. For example, by using the aws apigateway get-export AWS CLI command. For more information, see Export a REST API from API Gateway in the *API Gateway Developer Guide*.
2. Retrieve the `Body` property from the response of the `GetExport` API action. Upload it to an Amazon S3 bucket.
3. Download the generated template.
4. Add the `BodyS3Location/Bucket` and `BodyS3Location/Key` properties to the template, specifying the bucket name and key where the `Body` is stored.
5. Open the generated template in the IaC generator console and choose **Import edited template**.

How to resolve issues with write-only properties in AWS::Lambda::Function resources

This topic explains how to resolve issues with write-only properties in AWS::Lambda::Function resources when using the IaC generator.

Issue

The `AWS::Lambda::Function` resource has three mutually exclusive sets of properties for specifying the Lambda code:

- `Code/S3Bucket` and `Code/S3Key` properties, and optionally the `Code/S3ObjectVersion` property
- `Code/ImageUri` property
- `Code/ZipFile` property

Only one of these sets can be used for a given `AWS::Lambda::Function` resource.

The IaC generator can't determine which set of exclusive write-only properties was used to create or update the resource. As a result, it includes only the first set of properties in the generated template. The `Code/ImageUri` and `Code/ZipFile` properties are omitted.

Additionally, the IaC generator issues the following warnings:

- **`MUTUALLY_EXCLUSIVE_PROPERTIES`** – Warns that `Code/S3Bucket` and `Code/S3Key` are identified as mutually exclusive properties.
- **`UNSUPPORTED_PROPERTIES`** – Warns that the `Code/S3ObjectVersion` property is unsupported.

To include `AWS::Lambda::Function` resources in a generated template, you must download and update the template with the correct code properties.

Resolution

If you store your Lambda code in an Amazon S3 bucket and do not use the `S3ObjectVersion` property, you can import the generated template without any modifications. The IaC generator will ask you for the Amazon S3 bucket and key as template parameters during the import operation.

If you store your Lambda code as an Amazon ECR repository, you can update your template using the following instructions:

1. Download the generated template.
2. Remove the properties and corresponding parameters for the `Code/S3Bucket` and `Code/S3Key` properties from the generated template.
3. Replace the removed properties in the generated template with the `Code/ImageUri` property, specifying the URL for the Amazon ECR repository.

4. Open the generated template in the IaC generator console and choose the **Import edited template** button.

If you store your Lambda code as in a zip file, you can update your template using the following instructions:

1. Download the generated template.
2. Remove the properties and corresponding parameters for the `Code/S3Bucket` and `Code/S3Key` properties from the generated template.
3. Replace the removed properties in the generated template with the `Code/ZipFile` property.
4. Open the generated template in the IaC generator console and choose the **Import edited template** button.

If you don't have a copy of your Lambda code, you can update your template using the following instructions:

1. Use the AWS Lambda `GetFunction` API action (for example, by using the `aws lambda get-function` AWS CLI command.
2. In the response, the `RepositoryType` parameter is S3 if the code is in a Amazon S3 bucket, or ECR if the code is in an Amazon ECR repository.
3. In the response, the `Location` parameter contains a pre-signed URL that you can use to download the deployment package for 10 minutes. Download the code.
4. Upload the code to a Amazon S3 bucket.
5. Run an import operation with t he generated template and provide the bucket name and key as parameter values.

Get values stored in other services using dynamic references

Dynamic references provide a convenient way for you to specify external values stored and managed in other services and decouple sensitive information from your infrastructure-as-code templates. CloudFormation retrieves the value of the specified reference when necessary during stack and change set operations.

With dynamic references, you can:

- **Use secure strings** – For sensitive data, always use secure string parameters in AWS Systems Manager Parameter Store or secrets in AWS Secrets Manager to ensure your data is encrypted at rest.
- **Limit access** – Restrict access to the Parameter Store parameters or Secrets Manager secrets to only authorized principals and roles.
- **Rotate credentials** – Regularly rotate your sensitive data stored in Parameter Store or Secrets Manager to maintain a high level of security.
- **Automate rotation** – Leverage the automatic rotation features of Secrets Manager to periodically update and distribute your sensitive data across your applications and environments.

General considerations

The following are the general considerations for you to consider before you specify dynamic references in your CloudFormation templates:

- Avoid including dynamic references, or any sensitive data, in resource properties that are part of a resource's primary identifier. CloudFormation may use the actual plaintext value in the primary resource identifier, which could be a security risk. This resource ID may appear in any derived outputs or destinations.

  To determine which resource properties comprise a resource type's primary identifier, refer to the resource reference documentation for that resource in AWS resource and property types reference. In the **Return values** section, the `Ref` function return value represents the resource properties that comprise the resource type's primary identifier.
- You can include up to 60 dynamic references in a stack template.
- If you're using transforms (like `AWS::Include` or `AWS::Serverless`), CloudFormation doesn't resolve dynamic references before applying the transform. Instead, it passes the literal string of the dynamic reference to the transform, and resolves the references when you execute the change set using the template.
- Dynamic references can't be used for secure values (like those stored in Parameter Store or Secrets Manager) in custom resources.
- Dynamic references are also not supported in `AWS::CloudFormation::Init` metadata and Amazon EC2 `UserData` properties.
- Don't create a dynamic reference that ends with a backslash (\). CloudFormation can't resolve these references, which will cause stack operations to fail.

The following topics provide information and other considerations for using dynamic references.

Topics

- Get a plaintext value from Systems Manager Parameter Store
- Get a secure string value from Systems Manager Parameter Store
- Get a secret or secret value from Secrets Manager

Get a plaintext value from Systems Manager Parameter Store

When you're creating a CloudFormation template, you might want to use plaintext values stored in Parameter Store. Parameter Store is a capability of AWS Systems Manager. For an introduction to Parameter Store, see AWS Systems Manager Parameter Store in the *AWS Systems Manager User Guide*.

To use a plaintext value from Parameter Store within your template, you use a `ssm` dynamic reference. This reference allows you to access values from parameters of type `String` or `StringList` in Parameter Store.

To verify which version of an `ssm` dynamic reference will be used in a stack operation, create a change set for the stack operation. Then, review the processed template on the **Template** tab. For more information, see Create a change set for a CloudFormation stack.

When using `ssm` dynamic references, there are a few important things to keep in mind:

- CloudFormation doesn't support drift detection on dynamic references. For `ssm` dynamic references where you haven't specified the parameter version, we recommend that, if you update the parameter version in Systems Manager, you also perform a stack update operation on any stacks that include the `ssm` dynamic reference, in order to fetch the latest parameter version.
- To use a `ssm` dynamic reference in the `Parameters` section of your CloudFormation template, you must include a version number. CloudFormation doesn't allow you to reference a Parameter Store value without a version number in this section. Alternatively, you can define your parameter as a Systems Manager parameter type in your template. When you do this, you can specify a Systems Manager parameter key as the default value for your parameter. CloudFormation will then retrieve the latest version of the parameter value from Parameter Store, without you having to specify a version number. This can make your templates simpler and easier to maintain. For more information, see Specify existing resources at runtime with CloudFormation-supplied parameter types.

- For custom resources, CloudFormation resolves the `ssm` dynamic references before sending the request to the custom resource.
- CloudFormation doesn't support using dynamic references to reference a parameter shared from another AWS account.
- CloudFormation doesn't support using Systems Manager parameter labels in dynamic references.

Permissions

To specify a parameter stored in the Systems Manager Parameter Store, you must have permission to call GetParameters for the specified parameter. To learn how to create IAM policies that provide access to specific Systems Manager parameters, see Restricting access to Systems Manager parameters using IAM policies in the *AWS Systems Manager User Guide*.

Reference pattern

To reference a plaintext value stored in Systems Manager Parameter Store in your CloudFormation template, use the following `ssm` reference pattern.

```
{{resolve:ssm:parameter-name:version}}
```

Your reference must adhere to the following regular expression pattern for parameter-name and version:

```
{{resolve:ssm:[a-zA-Z0-9_.\-/]+(:\d+)?}}
```

`parameter-name`

The name of the parameter in the Parameter Store. The parameter name is case-sensitive.

Required.

`version`

An integer that specifies the version of the parameter to use. If you don't specify the exact version, CloudFormation uses the latest version of the parameter whenever you create or update the stack. For more information, see Working with parameter versions in the *AWS Systems Manager User Guide*.

Optional.

Examples

Topics

- Public AMI ID parameter
- Custom AMI ID parameter

Public AMI ID parameter

The following example creates an EC2 instance that references a public AMI parameter. The dynamic reference retrieves the latest Amazon Linux 2023 AMI ID from the public parameter. For more information about public parameters, see Discovering public parameters in Parameter Store in the *AWS Systems Manager User Guide*.

JSON

```
{
    "Resources": {
        "MyInstance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64}}",
                "InstanceType": "t2.micro"
            }
        }
    }
}
```

YAML

```
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64}}'
      InstanceType: t2.micro
```

Custom AMI ID parameter

The following example creates an EC2 launch template that references a custom AMI ID stored in the Parameter Store. The dynamic reference retrieves the AMI ID from version *2* of the *golden-ami* parameter any time an instance is launched from the launch template.

JSON

```
{
    "Resources": {
        "MyLaunchTemplate": {
            "Type": "AWS::EC2::LaunchTemplate",
            "Properties": {
                "LaunchTemplateName": {
                    "Fn::Sub": "${AWS::StackName}-launch-template"
                },
                "LaunchTemplateData": {
                    "ImageId": "{{resolve:ssm:golden-ami:2}}",
                    "InstanceType": "t2.micro"
                }
            }
        }
    }
}
```

YAML

```
Resources:
  MyLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: '{{resolve:ssm:golden-ami:2}}'
        InstanceType: t2.micro
```

Get a secure string value from Systems Manager Parameter Store

In CloudFormation, you can use sensitive data like passwords or license keys without exposing them directly in your templates by storing the sensitive data as a "secure string" in AWS Systems Manager Parameter Store. For an introduction to Parameter Store, see AWS Systems Manager Parameter Store in the *AWS Systems Manager User Guide*.

To use a Parameter Store secure string within your template, you use a `ssm-secure` dynamic reference. CloudFormation never stores the actual secure string value. Instead, it only stores the literal dynamic reference, which contains the plaintext parameter name of the secure string.

During stack creation or updates, CloudFormation accesses the secure string value as needed, without exposing the actual value. Secure strings can only be used for resource properties that support the `ssm-secure` dynamic reference pattern. For more information, see Resources that support dynamic parameter patterns for secure strings.

CloudFormation doesn't return the actual parameter value for secure strings in any API calls. It only returns the literal dynamic reference. When comparing changes using change sets, CloudFormation only compares the literal dynamic reference string. It doesn't resolve and compare the actual secure string values.

When using `ssm-secure` dynamic references, there are a few important things to keep in mind:

- CloudFormation can't access Parameter Store values from other AWS accounts.
- CloudFormation doesn't support using Systems Manager parameter labels or public parameters in dynamic references.
- In the `cn-north-1` and `cn-northwest-1` regions, secure strings aren't supported by Systems Manager.
- Dynamic references for secure values, such as `ssm-secure`, aren't currently supported in custom resources.
- If CloudFormation needs to roll back a stack update, and the previously specified version of a secure string parameter is no longer available, the rollback operation will fail. In such cases, you have two options:
  - Use `CONTINUE_UPDATE_ROLLBACK` to skip the resource.
  - Recreate the secure string parameter in the Systems Manager Parameter Store, and update it until the parameter version reaches the version used in the template. Then, use `CONTINUE_UPDATE_ROLLBACK` without skipping the resource.

Resources that support dynamic parameter patterns for secure strings

Resources that support the `ssm-secure` dynamic reference pattern include:

Resource	Property type	Properties
AWS::DirectoryService::MicrosoftAD		Password
AWS::DirectoryService::SimpleAD		Password
AWS::ElastiCache::ReplicationGroup		AuthToken
AWS::IAM::User	LoginProfile	Password
AWS::KinesisFirehose::DeliveryStream	RedshiftDestinationConfiguration	Password
AWS::OpsWorks::App	Source	Password
AWS::OpsWorks::Stack	CustomCookbooksSource	Password
AWS::OpsWorks::Stack	RdsDbInstances	DbPassword
AWS::RDS::DBCluster		MasterUserPassword
AWS::RDS::DBInstance		MasterUserPassword
AWS::Redshift::Cluster		MasterUserPassword

Reference pattern

To reference a secure string value from Systems Manager Parameter Store in your CloudFormation template, use the following `ssm-secure` reference pattern.

```
{{resolve:ssm-secure:parameter-name:version}}
```

Your reference must adhere to the following regular expression pattern for parameter-name and version:

```
{{resolve:ssm-secure:[a-zA-Z0-9_.\-/]+(:\d+)?}}
```

`parameter-name`

The name of the parameter in the Parameter Store. The parameter name is case-sensitive.

Required.

`version`

An integer that specifies the version of the parameter to use. If you don't specify the exact version, CloudFormation uses the latest version of the parameter whenever you create or update the stack. For more information, see Working with parameter versions in the *AWS Systems Manager User Guide*.

Optional.

Example

The following example uses an `ssm-secure` dynamic reference to set the password for an IAM user to a secure string stored in Parameter Store. As specified, CloudFormation will use version *`10`* of the *`IAMUserPassword`* parameter for stack and change set operations.

JSON

```
"MyIAMUser": {
  "Type": "AWS::IAM::User",
  "Properties": {
    "UserName": "MyUserName",
    "LoginProfile": {
      "Password": "{{resolve:ssm-secure:IAMUserPassword:10}}"
    }
  }
}
```

YAML

```
MyIAMUser:
  Type: AWS::IAM::User
  Properties:
    UserName: 'MyUserName'
    LoginProfile:
      Password: '{{resolve:ssm-secure:IAMUserPassword:10}}'
```

Get a secret or secret value from Secrets Manager

Secrets Manager is a service that allows you to securely store and manage secrets like database credentials, passwords, and third-party API keys. Using Secrets Manager, you can store and control access to these secrets centrally, so that you can replace hardcoded credentials in your code (including passwords), with an API call to Secrets Manager to retrieve the secret programmatically. For more information, see What is AWS Secrets Manager? in the *AWS Secrets Manager User Guide*.

To use entire secrets or secret values that are stored in Secrets Manager within your CloudFormation templates, you use `secretsmanager` dynamic references.

Best practices

Follow these best practices when using Secrets Manager dynamic references in your CloudFormation templates:

- **Use versionless references for your CloudFormation templates** – Store credentials in Secrets Manager and use dynamic references without specifying `version-stage` or `version-id` parameters to support proper secret rotation workflows.
- **Leverage automatic rotation** – Use Secrets Manager's automatic rotation feature with versionless dynamic references for credential management. This ensures your credentials are regularly updated without requiring template changes. For more information, see Rotate AWS Secrets Manager secrets.
- **Use versioned references sparingly** – Only specify explicit `version-stage` or `version-id` parameters for specific scenarios like testing or rollback situations.

Considerations

When using `secretsmanager` dynamic references, there are important considerations to keep in mind:

- CloudFormation doesn't track which version of a secret was used in previous deployments. Plan your secret management strategy carefully before implementing dynamic references. Use versionless references when possible to leverage automatic secret rotation. Monitor and validate resource updates when making changes to dynamic reference configurations, such as when transitioning from unversioned to versioned dynamic references, and vice versa.

- Updating only the secret value in Secrets Manager doesn't automatically cause CloudFormation to retrieve the new value. CloudFormation retrieves the secret value only during resource creation or updates that modify the resource containing the dynamic reference.

  For example, suppose your template includes an AWS::RDS::DBInstance resource where the `MasterPassword` property is set to a Secrets Manager dynamic reference. After creating a stack from this template, you update the secret's value in Secrets Manager. However, the `MasterPassword` property retains the old password value.

  To apply the new secret value, you'll need to modify the `AWS::RDS::DBInstance` resource in your CloudFormation template and perform a stack update.

  To avoid this manual process in the future, consider using Secrets Manager to automatically rotate the secret.
- Dynamic references for secure values, such as `secretsmanager`, aren't currently supported in custom resources.
- The `secretsmanager` dynamic reference can be used in all resource properties. Using the `secretsmanager` dynamic reference indicates that neither Secrets Manager nor CloudFormation logs should persist any resolved secret value. However, the secret value may show up in the service whose resource it's being used in. Review your usage to avoid leaking secret data.

Permissions

To specify a secret stored in Secrets Manager, you must have permission to call GetSecretValue for the secret.

Reference pattern

To reference Secrets Manager secrets in your CloudFormation template, use the following `secretsmanager` reference pattern.

```
{{resolve:secretsmanager:secret-id:secret-string:json-key:version-stage:version-id}}
```

`secret-id`

The name or ARN of the secret.

To access a secret in your AWS account, you need only specify the secret name. To access a secret in a different AWS account, specify the complete ARN of the secret.

Required.

`secret-string`

The only supported value is `SecretString`. The default is `SecretString`.

`json-key`

The key name of the key-value pair whose value you want to retrieve. If you don't specify a `json-key`, CloudFormation retrieves the entire secret text.

This segment may not include the colon character (`:`).

`version-stage`

The staging label of the version of the secret to use. Secrets Manager uses staging labels to keep track of different versions during the rotation process. If you use `version-stage` then don't specify `version-id`. If you don't specify either `version-stage` or `version-id`, then the default is the `AWSCURRENT` version.

This segment may not include the colon character (`:`).

`version-id`

The unique identifier of the version of the secret to use. If you specify `version-id`, then don't specify `version-stage`. If you don't specify either `version-stage` or `version-id`, then the default is the `AWSCURRENT` version.

This segment may not include the colon character (`:`).

Examples

Topics

- Retrieving user name and password values from a secret
- Retrieving the entire SecretString
- Retrieving a value from a specific version of a secret
- Retrieving secrets from another AWS account

Retrieving user name and password values from a secret

The following AWS::RDS::DBInstance example retrieves the user name and password values stored in the *MySecret* secret. This example shows the recommended pattern for versionless dynamic references, which automatically uses the AWSCURRENT version and supports Secrets Manager rotation workflows without requiring template changes.

JSON

```
{
    "MyRDSInstance": {
        "Type": "AWS::RDS::DBInstance",
        "Properties": {
            "DBName": "MyRDSInstance",
            "AllocatedStorage": "20",
            "DBInstanceClass": "db.t2.micro",
            "Engine": "mysql",
            "MasterUsername":
 "{{resolve:secretsmanager:MySecret:SecretString:username}}",
            "MasterUserPassword":
 "{{resolve:secretsmanager:MySecret:SecretString:password}}"
        }
    }
}
```

YAML

```
  MyRDSInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      DBName: MyRDSInstance
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MySecret:SecretString:username}}'
      MasterUserPassword: '{{resolve:secretsmanager:MySecret:SecretString:password}}'
```

Retrieving the entire SecretString

The following dynamic reference retrieves the SecretString for *MySecret*.

```
{{resolve:secretsmanager:MySecret}}
```

Alternatively:

```
{{resolve:secretsmanager:MySecret::::}}
```

Retrieving a value from a specific version of a secret

The following dynamic reference retrieves the *password* value for the *AWSPREVIOUS* version of *MySecret*.

```
{{resolve:secretsmanager:MySecret:SecretString:password:AWSPREVIOUS}}
```

Retrieving secrets from another AWS account

The following dynamic reference retrieves the SecretString for *MySecret* that's in another AWS account. You must specify the complete secret ARN to access secrets in another AWS account.

```
{{resolve:secretsmanager:arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret}}
```

The following dynamic reference retrieves the *password* value for *MySecret* that's in another AWS account. You must specify the complete secret ARN to access secrets in another AWS account.

```
{{resolve:secretsmanager:arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret:SecretString:password}}
```

Get AWS values using pseudo parameters

Pseudo parameters are built-in variables that provide access to important AWS environment information such as account IDs, Region names, and stack details that can change between deployments or environments.

You can use pseudo parameters instead of hard-coded values to make your templates more portable and easier to reuse across different AWS accounts and Regions.

Syntax

You can reference pseudo parameters using either the Ref intrinsic function or the Fn::Sub intrinsic function.

Ref

The `Ref` intrinsic function uses the following general syntax. For more information, see Ref.

JSON

```
{ "Ref" : "AWS::PseudoParameter" }
```

YAML

```
!Ref AWS::PseudoParameter
```

Fn::Sub

The `Fn::Sub` intrinsic function uses a different format that includes the `${}` syntax around the pseudo parameter. For more information, see Fn::Sub.

JSON

```
{ "Fn::Sub" : "${AWS::PseudoParameter}" }
```

YAML

```
!Sub '${AWS::PseudoParameter}'
```

Available pseudo parameters

`AWS::AccountId`

Returns the AWS account ID of the account in which the stack is being created, such as `123456789012`.

This pseudo parameter is commonly used when defining IAM roles, policies, and other resource policies that involve account-specific ARNs.

`AWS::NotificationARNs`

Returns the list of Amazon Resource Names (ARNs) for the Amazon SNS topics that receive stack event notifications. You can specify these ARNs through the `--notification-arns` option in the AWS CLI or through the console as you are creating or updating your stack.

Unlike other pseudo parameters that return a single value, `AWS::NotificationARNs` returns a list of ARNs. To access a specific ARN in the list, use the `Fn::Select` intrinsic function. For more information, see Fn::Select.

AWS::NoValue

Removes the corresponding resource property when specified as a return value in the `Fn::If` intrinsic function. For more information, see Fn::If.

This pseudo parameter is particularly useful for creating conditional resource properties that should only be included under certain conditions.

AWS::Partition

Returns the partition that the resource is in. For standard AWS Regions, the partition is `aws`. For resources in other partitions, the partition is `aws-`*`partitionname`*. For example, the partition for resources in the China (Beijing and Ningxia) Regions is `aws-cn` and the partition for resources in the AWS GovCloud (US-West) Region is `aws-us-gov`.

The partition forms part of the ARN for resources. Using `AWS::Partition` ensures your templates work correctly across different AWS partitions.

AWS::Region

Returns a string representing the Region in which the encompassing resource is being created, such as `us-west-2`.

This is one of the most commonly used pseudo parameters, as it allows templates to adapt to different AWS Regions without modification.

AWS::StackId

Returns the ID (ARN) of the stack, such as `arn:aws:cloudformation:us-west-2:123456789012:stack/teststack/51af3dc0-da77-11e4-872e-1234567db123`.

AWS::StackName

Returns the name of the stack, such as `teststack`.

The stack name is commonly used to create unique resource names that are easily identifiable as belonging to a specific stack.

AWS::URLSuffix

Returns the suffix for the AWS domain in the AWS Region where the stack is deployed. The suffix is typically `amazonaws.com`, but for the China (Beijing) Region, the suffix is `amazonaws.com.cn`.

This parameter is particularly useful when constructing URLs for AWS service endpoints.

Examples

Topics

- Basic usage
- Using AWS::NotificationARNs
- Conditional properties with AWS::NoValue

Basic usage

The following examples create two resources: an Amazon SNS topic and a CloudWatch alarm that sends notifications to that topic. They use `AWS::StackName`, `AWS::Region`, and `AWS::AccountId` to dynamically insert the stack name, current AWS Region, and account ID into resource names, descriptions, and ARNs.

JSON

```
{
    "Resources": {
        "MyNotificationTopic": {
            "Type": "AWS::SNS::Topic",
            "Properties": {
                "DisplayName": { "Fn::Sub": "Notifications for ${AWS::StackName}" }
            }
        },
        "CPUAlarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": { "Fn::Sub": "Alarm for high CPU in
 ${AWS::Region}" },
                "AlarmName": { "Fn::Sub": "${AWS::StackName}-HighCPUAlarm" },
                "MetricName": "CPUUtilization",
                "Namespace": "AWS/EC2",
                "Statistic": "Average",
                "Period": 300,
```

```
                "EvaluationPeriods": 1,
                "Threshold": 80,
                "ComparisonOperator": "GreaterThanThreshold",
                "AlarmActions": [{ "Fn::Sub": "arn:aws:sns:${AWS::Region}:
${AWS::AccountId}:${MyNotificationTopic}" }]
            }
        }
    }
}
```

YAML

```
Resources:
  MyNotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: !Sub Notifications for ${AWS::StackName}
  CPUAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: !Sub Alarm for high CPU in ${AWS::Region}
      AlarmName: !Sub ${AWS::StackName}-HighCPUAlarm
      MetricName: CPUUtilization
      Namespace: AWS/EC2
      Statistic: Average
      Period: 300
      EvaluationPeriods: 1
      Threshold: 80
      ComparisonOperator: GreaterThanThreshold
      AlarmActions:
        - !Sub arn:aws:sns:${AWS::Region}:${AWS::AccountId}:${MyNotificationTopic}
```

Using AWS::NotificationARNs

The following examples configure an Auto Scaling group to send notifications for instance launch events and launch errors. The configuration uses the `AWS::NotificationARNs` pseudo parameter, which provides a list of Amazon SNS topic ARNs that were specified during stack creation. The `Fn::Select` function chooses the first ARN from that list.

JSON

```
"myASG": {
```

```
   "Type": "AWS::AutoScaling::AutoScalingGroup",
   "Properties": {
      "LaunchTemplate": {
         "LaunchTemplateId": { "Ref": "myLaunchTemplate" },
         "Version": { "Fn::GetAtt": [ "myLaunchTemplate", "LatestVersionNumber" ] }
       },
       "MaxSize": "1",
       "MinSize": "1",
       "VPCZoneIdentifier": [
          "subnetIdAz1",
          "subnetIdAz2",
          "subnetIdAz3"
      ],
      "NotificationConfigurations" : [{
         "TopicARN" : { "Fn::Select" : [ "0", { "Ref" : "AWS::NotificationARNs" } ] },
         "NotificationTypes" : [ "autoscaling:EC2_INSTANCE_LAUNCH",
 "autoscaling:EC2_INSTANCE_LAUNCH_ERROR" ]
      }]
   }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MinSize: '1'
    MaxSize: '1'
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    NotificationConfigurations:
      - TopicARN:
          Fn::Select:
          - '0'
          - Ref: AWS::NotificationARNs
        NotificationTypes:
        - autoscaling:EC2_INSTANCE_LAUNCH
        - autoscaling:EC2_INSTANCE_LAUNCH_ERROR
```

Conditional properties with AWS::NoValue

The following examples create an Amazon RDS DB instance that uses a snapshot only if a snapshot ID is provided. If the `UseDBSnapshot` condition evaluates to true, CloudFormation uses the `DBSnapshotName` parameter value for the `DBSnapshotIdentifier` property. If the condition evaluates to false, CloudFormation removes the `DBSnapshotIdentifier` property.

JSON

```
"MyDB" : {
  "Type" : "AWS::RDS::DBInstance",
  "Properties" : {
    "AllocatedStorage" : "5",
    "DBInstanceClass" : "db.t2.small",
    "Engine" : "MySQL",
    "EngineVersion" : "5.5",
    "MasterUsername" : { "Ref" : "DBUser" },
    "MasterUserPassword" : { "Ref" : "DBPassword" },
    "DBParameterGroupName" : { "Ref" : "MyRDSParamGroup" },
    "DBSnapshotIdentifier" : {
      "Fn::If" : [
        "UseDBSnapshot",
        {"Ref" : "DBSnapshotName"},
        {"Ref" : "AWS::NoValue"}
      ]
    }
  }
}
```

YAML

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: MySQL
    EngineVersion: '5.5'
    MasterUsername:
      Ref: DBUser
    MasterUserPassword:
      Ref: DBPassword
    DBParameterGroupName:
```

```
      Ref: MyRDSParamGroup
    DBSnapshotIdentifier:
      Fn::If:
        - UseDBSnapshot
        - Ref: DBSnapshotName
        - Ref: AWS::NoValue
```

Get exported outputs from a deployed CloudFormation stack

When you have multiple stacks in the same AWS account and Region, you might want to share information between them. This is useful when one stack needs to use resources created by another stack.

For example, you might have one stack that creates network resources, such as subnets and security groups, for your web servers. Other stacks that create the actual web servers can then use the network resources created by the first stack. You don't need to hard code resource IDs in the stack's template or pass IDs as input parameters.

To share information between stacks, you *export* output values from one stack and *import* them into another stack. Here's how it works:

1. In the first stack's template (for example, the networking stack), you define certain values for export by using the `Export` field in the `Outputs` section. For more information, see CloudFormation template Outputs syntax.
2. When you create or update that stack, CloudFormation exports the output values, making them available to other stacks in the same AWS account and Region.
3. In the other stack's template, you use the Fn::ImportValue function to import the exported values from the first stack.
4. When you create or update the second stack (for example, the web server stack), CloudFormation automatically retrieves the exported values from the first stack and uses them.

For a walkthrough and sample templates, see Refer to resource outputs in another CloudFormation stack.

Exporting stack output values versus using nested stacks

A nested stack is a stack that you create within another stack by using the `AWS::CloudFormation::Stack` resource. With nested stacks, you deploy and manage all

resources from a single stack. You can use outputs from one stack in the nested stack group as inputs to another stack in the group. This differs from exporting values.

If you want to isolate information sharing to within a nested stack group, we suggest that you use nested stacks. To share information with other stacks (not just within the group of nested stacks), export values. For example, you can create a single stack with a subnet and then export its ID. Other stacks can use that subnet by importing its ID. Each stack doesn't need to create its own subnet. As long as stacks are importing the subnet ID, you can't change or delete it.

For more information about nested stacks, see Split a template into reusable pieces using nested stacks.

Considerations

The following restrictions apply to cross-stack references:

- For each AWS account, `Export` names must be unique within a Region.
- When using `Export` and `Fn::ImportValue`, cross-stack references are limited to the same account and Region. To reference stack outputs across accounts or Regions, use `Fn::GetStackOutput`.
- For outputs, the value of the `Name` property of an `Export` can't use `Ref` or `GetAtt` functions that depend on a resource.

  Similarly, the `ImportValue` function can't include `Ref` or `GetAtt` functions that depend on a resource.
- After another stack imports an output value, you can't delete the stack that is exporting the output value or modify the exported output value. All the imports must be removed before you can delete the exporting stack or modify the output value.

Listing exported output values

If you need to view the exported output values from your stacks, use one of the following methods:

To list exported output values (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.

3. From the left navigation pane, choose **Exports**.

To list exported output values (AWS CLI)

Use the following list-exports command. Replace *us-east-1* with your AWS Region.

```
aws cloudformation list-exports --region us-east-1
```

The following is example output.

```
{
    "Exports": [
        {
            "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
            "Name": "private-vpc-subnet-a",
            "Value": "subnet-07b410xmplddcfa03"
        },
        {
            "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
            "Name": "private-vpc-subnet-b",
            "Value": "subnet-075ed3xmplebd2fb1"
        },
        {
            "ExportingStackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
private-vpc/99764070-b56c-xmpl-bee8-062a88d1d800",
            "Name": "private-vpc-vpcid",
            "Value": "vpc-011d7xmpl100e9841"
        }
    ]
}
```

CloudFormation shows the names and values of the exported outputs for the current region and the stack they were exported from. To use an exported output value in another stack's template, you can reference it using the export name and the `Fn::ImportValue` function.

Listing stacks that import an exported output value

To delete or change exported output values, you must first find out which stacks are importing them.

To view the stacks that import an exported output value, use one of the following methods:

To list stacks that import an exported output value (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the left navigation pane, choose **Exports**.
3. To see which stacks import a given export value, choose the **Export Name** for that export value. CloudFormation displays the export details page, which lists all the stacks that are importing the value.

To list stacks that import an exported output value (AWS CLI)

Use the list-imports command. Replace *us-east-1* with your AWS Region and *private-vpc-vpcid* with the name of the exported output value.

```
aws cloudformation list-imports --region us-east-1 \
    --export-name private-vpc-vpcid
```

CloudFormation returns a list of stacks that are importing the value.

```
{
    "Imports": [
        "my-app-stack"
    ]
}
```

Once you know which stacks are importing a particular exported value, you need to modify those stacks to remove the `Fn::ImportValue` functions that reference the output values. You must remove all the imports that reference exported output values before you can delete or modify the exported output values.

Referencing outputs without exports

In addition to using `Export` and `Fn::ImportValue` for cross-stack references, you can use `Fn::GetStackOutput` to reference stack outputs directly. This approach does not require the referenced stack to declare an export and supports cross-account and cross-Region references.

The following table compares the two approaches:

Comparison of Export with Fn::ImportValue and Fn::GetStackOutput

Capability	`Export` + `Fn::ImportValue`	`Fn::GetStackOutput`
Same account, same Region	Supported	Supported
Cross-account	Not supported	Supported
Cross-Region	Not supported	Supported
Requires explicit Export	Yes	No
Reference type	Strong (blocks deletion of exporting stack)	Weak (resolved at deploy time)
Referential integrity	Enforced	Not enforced

When to use each approach:

- Use `Export` and `Fn::ImportValue` when you need strong referential integrity. With this approach, CloudFormation prevents you from deleting a stack that exports values consumed by other stacks, protecting you from accidentally breaking dependencies.
- Use `Fn::GetStackOutput` when you need to reference outputs across accounts or Regions, or when you prefer not to manage explicit exports. Be aware that `Fn::GetStackOutput` creates a weak reference. If the referenced stack or output is deleted, the consuming stack is not notified, and subsequent operations that re-resolve the reference will fail.

For more information, see Fn::GetStackOutput in the *CloudFormation Template Reference Guide*.

Specify existing resources at runtime with CloudFormation-supplied parameter types

When creating your template, you can create parameters that require users to input identifiers of existing AWS resources or Systems Manager parameters by using specialized parameter types provided by CloudFormation.

Topics

- Overview
- Example
- Considerations
- Supported AWS-specific parameter types
- Supported Systems Manager parameter types
- Unsupported Systems Manager parameter types

Overview

In CloudFormation, you can use parameters to customize your stacks by providing input values during stack creation or update. This feature makes your templates reusable and flexible across different scenarios.

Parameters are defined in the `Parameters` section of a CloudFormation template. Each parameter has a name and a type, and can have additional settings such as a default value and allowed values. For more information, see CloudFormation template Parameters syntax.

The parameter type determines the kind of input value the parameter can accept. For example, `Number` only accepts numeric values, while `String` accepts text input.

CloudFormation provides several additional parameter types that you can use in your templates to reference existing AWS resources and Systems Manager parameters.

These parameter types fall into two categories:

- **AWS-specific parameter types** – CloudFormation provides a set of parameter types that help catch invalid values when creating or updating a stack. When you use these parameter types, anyone who uses your template must specify valid values from the AWS account and Region they're creating the stack in.

  If they use the AWS Management Console, CloudFormation provides a prepopulated list of existing values from their account and Region. This way, the user doesn't have to remember and correctly type a specific name or ID. Instead, they just select values from a drop-down list. In some cases, they can even search for values by ID, name, or `Name` tag value.
- **Systems Manager parameter types** – CloudFormation also provides parameter types that correspond to existing parameters in Systems Manager Parameter Store. When you use these parameter types, anyone who uses your template must specify a Parameter Store key as the value of the Systems Manager parameter type, and CloudFormation then retrieves the latest

value from Parameter Store to use in their stack. This can be useful when you need to frequently update applications with new property values, such as new Amazon Machine Image (AMI) IDs. For information about the Parameter Store, see Systems Manager Parameter Store.

Once your parameters are defined in the `Parameters` section, you can reference parameter values throughout your CloudFormation template using the `Ref` function.

Example

The following example shows a template that uses the following parameter types.

- `AWS::EC2::VPC::Id`
- `AWS::EC2::Subnet::Id`
- `AWS::EC2::KeyPair::KeyName`
- `AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>`

To create a stack from this template, you must specify an existing VPC ID, subnet ID, and key pair name from your account. You can also specify an existing Parameter Store key that references the desired AMI ID or keep the default value of */aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2*. This public parameter is an alias for the regional AMI ID for the latest Amazon Linux 2 AMI. For more information about public parameters, see Discovering public parameters in Parameter Store in the *AWS Systems Manager User Guide*.

JSON

```
{
    "Parameters": {
        "VpcId": {
            "Description": "ID of an existing Virtual Private Cloud (VPC).",
            "Type": "AWS::EC2::VPC::Id"
        },
        "PublicSubnetId": {
            "Description": "ID of an existing public subnet within the specified VPC.",
            "Type": "AWS::EC2::Subnet::Id"
        },
        "KeyName": {
            "Description": "Name of an existing EC2 key pair to enable SSH access to
 the instance.",
            "Type": "AWS::EC2::KeyPair::KeyName"
```

```
        },
        "AMIId": {
            "Description": "Name of a Parameter Store parameter that stores the ID of
 the Amazon Machine Image (AMI).",
            "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
            "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
        }
    },
    "Resources": {
        "InstanceSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Enable SSH access via port 22",
                "VpcId": { "Ref": "VpcId" },
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 22,
                        "ToPort": 22,
                        "CidrIp": "0.0.0.0/0"
                    }
                ]
            }
        },
        "Ec2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "KeyName": { "Ref": "KeyName" },
                "ImageId": { "Ref": "AMIId" },
                "NetworkInterfaces": [
                    {
                        "AssociatePublicIpAddress": "true",
                        "DeviceIndex": "0",
                        "SubnetId": { "Ref": "PublicSubnetId" },
                        "GroupSet": [{ "Ref": "InstanceSecurityGroup" }]
                    }
                ]
            }
        }
    },
    "Outputs": {
        "InstanceId": {
            "Value": { "Ref": "Ec2Instance" }
        }
```

```
    }
}
```

YAML

```
Parameters:
  VpcId:
    Description: ID of an existing Virtual Private Cloud (VPC).
    Type: AWS::EC2::VPC::Id
  PublicSubnetId:
    Description: ID of an existing public subnet within the specified VPC.
    Type: AWS::EC2::Subnet::Id
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance.
    Type: AWS::EC2::KeyPair::KeyName
  AMIId:
    Description: Name of a Parameter Store parameter that stores the ID of the Amazon
 Machine Image (AMI).
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
Resources:
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH access via port 22
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      ImageId: !Ref AMIId
      NetworkInterfaces:
        - AssociatePublicIpAddress: "true"
          DeviceIndex: "0"
          SubnetId: !Ref PublicSubnetId
          GroupSet:
            - !Ref InstanceSecurityGroup
Outputs:
```

```
  InstanceId:
    Value: !Ref Ec2Instance
```

AWS CLI command to create the stack

The following create-stack command creates a stack based on the example template.

```
aws cloudformation create-stack --stack-name MyStack \
  --template-body file://sampletemplate.json \
  --parameters \
ParameterKey="VpcId",ParameterValue="vpc-a123baa3" \
ParameterKey="PublicSubnetId",ParameterValue="subnet-123a351e" \
ParameterKey="KeyName",ParameterValue="MyKeyName" \
ParameterKey="AMIId",ParameterValue="MyParameterKey"
```

To use a parameter type that accepts a list of strings , such as `List<AWS::EC2::Subnet::Id>`, you must escape the commas inside the `ParameterValue` with a double backslash, as shown in the following example.

```
--parameters ParameterKey="SubnetIDs",ParameterValue="subnet-5ea0c127\\,subnet-6194ea3b
\\,subnet-c87f2be0"
```

Considerations

It's strongly recommended that you use dynamic references to restrict access to sensitive configuration definitions, such as third-party credentials. For more information, see Get values stored in other services using dynamic references.

If you want to allow template users to specify values from different AWS accounts, don't use AWS-specific parameter types. Instead, define parameters of type `String` or `CommaDelimitedList`.

There are a few things to keep in mind with Systems Manager parameter types:

- You can see the resolved parameter values on the stack's **Parameters** tab in the console, or by running describe-stacks or describe-change-set. Remember, these values are set when the stack is created or updated, so they might be different from the latest values in Parameter Store.
- For stack updates, when you use the **Use existing value** option (or set `UsePreviousValue` to true), this means that you want to keep using the same Parameter Store key, not its value. CloudFormation always retrieves the latest value.

- If you specify any allowed values or other constraints, CloudFormation validates them against the parameter keys you specify, but not their values. You should validate the values in Parameter Store itself.
- When you create or update stacks and create change sets, CloudFormation uses whatever value exists in Parameter Store at the time. If a specified parameter doesn't exist in Parameter Store under the caller's AWS account, CloudFormation returns a validation error.
- When you execute a change set, CloudFormation uses the values that are specified in the change set. You should review these values before executing the change set because they might change in Parameter Store between the time that you create the change set and run it.
- For Parameter Store parameters stored in the same AWS account, you must provide the parameter name. For Parameter Store parameters shared by another AWS account, you must provide the full parameter ARN.

Supported AWS-specific parameter types

CloudFormation supports the following AWS-specific types:

`AWS::EC2::AvailabilityZone::Name`

An Availability Zone, such as `us-west-2a`.

`AWS::EC2::Image::Id`

An Amazon EC2 image ID, such as `ami-0ff8a91507f77f867`. Note that the CloudFormation console doesn't show a drop-down list of values for this parameter type.

`AWS::EC2::Instance::Id`

An Amazon EC2 instance ID, such as `i-1e731a32`.

`AWS::EC2::KeyPair::KeyName`

An Amazon EC2 key pair name.

`AWS::EC2::SecurityGroup::GroupName`

A default VPC security group name, such as `my-sg-abc`.

`AWS::EC2::SecurityGroup::Id`

A security group ID, such as `sg-a123fd85`.

`AWS::EC2::Subnet::Id`

A subnet ID, such as `subnet-123a351e`.

`AWS::EC2::Volume::Id`

An Amazon EBS volume ID, such as `vol-3cdd3f56`.

`AWS::EC2::VPC::Id`

A VPC ID, such as `vpc-a123baa3`.

`AWS::Route53::HostedZone::Id`

An Amazon Route 53 hosted zone ID, such as `Z23YXV4OVPL04A`.

`List<AWS::EC2::AvailabilityZone::Name>`

An array of Availability Zones for a region, such as `us-west-2a, us-west-2b`.

`List<AWS::EC2::Image::Id>`

An array of Amazon EC2 image IDs, such as `ami-0ff8a91507f77f867`, `ami-0a584ac55a7631c0c`. Note that the CloudFormation console doesn't show a drop-down list of values for this parameter type.

`List<AWS::EC2::Instance::Id>`

An array of Amazon EC2 instance IDs, such as `i-1e731a32, i-1e731a34`.

`List<AWS::EC2::SecurityGroup::GroupName>`

An array of default VPC security group names, such as `my-sg-abc, my-sg-def`.

`List<AWS::EC2::SecurityGroup::Id>`

An array of security group IDs, such as `sg-a123fd85, sg-b456fd85`.

`List<AWS::EC2::Subnet::Id>`

An array of subnet IDs, such as `subnet-123a351e, subnet-456b351e`.

`List<AWS::EC2::Volume::Id>`

An array of Amazon EBS volume IDs, such as `vol-3cdd3f56, vol-4cdd3f56`.

`List<AWS::EC2::VPC::Id>`

An array of VPC IDs, such as `vpc-a123baa3, vpc-b456baa3`.

`List<AWS::Route53::HostedZone::Id>`

An array of Amazon Route 53 hosted zone IDs, such as `Z23YXV4OVPL04A, Z23YXV4OVPL04B`.

Supported Systems Manager parameter types

CloudFormation supports the following Systems Manager parameter types:

`AWS::SSM::Parameter::Name`

The name of a Systems Manager parameter key. Use this parameter type only to check that a required parameter exists. CloudFormation won't retrieve the actual value associated with the parameter.

`AWS::SSM::Parameter::Value<String>`

A Systems Manager parameter whose value is a string. This corresponds to the `String` parameter type in Parameter Store.

`AWS::SSM::Parameter::Value<List<String>>` or
`AWS::SSM::Parameter::Value<CommaDelimitedList>`

A Systems Manager parameter whose value is a list of strings. This corresponds to the `StringList` parameter type in Parameter Store.

`AWS::SSM::Parameter::Value<`*`AWS-specific parameter type`*`>`

A Systems Manager parameter whose value is an AWS-specific parameter type.

For example, the following specifies the `AWS::EC2::KeyPair::KeyName` type:

- `AWS::SSM::Parameter::Value<AWS::EC2::KeyPair::KeyName>`

`AWS::SSM::Parameter::Value<List<`*`AWS-specific parameter type`*`>>`

A Systems Manager parameter whose value is a list of AWS-specific parameter types.

For example, the following specifies a list of `AWS::EC2::KeyPair::KeyName` types:

- `AWS::SSM::Parameter::Value<List<AWS::EC2::KeyPair::KeyName>>`

Unsupported Systems Manager parameter types

CloudFormation doesn't support the following Systems Manager parameter type:

- Lists of Systems Manager parameter types—for example: `List<AWS::SSM::Parameter::Value<String>>`

In addition, CloudFormation doesn't support defining template parameters as `SecureString` Systems Manager parameter types. However, you can specify secure strings as parameter *values* for certain resources. For more information, see Get values stored in other services using dynamic references.

CloudFormation walkthroughs

This documentation provides a collection of walkthroughs designed to give you hands-on practice with stack deployments.

- Refer to resource outputs in another CloudFormation stack – This walkthrough shows you how to reference outputs from one CloudFormation stack within another stack. Instead of including all resources in a single stack, you can create related AWS resources in separate stacks to create more modular and reusable templates.
- Deploy applications on Amazon EC2 – Learn how to use CloudFormation to automatically install, configure, and start up your application on Amazon EC2 instances. This way, you can easily duplicate deployments and update existing installations without connecting directly to the instances.
- Update a CloudFormation stack – Walk through a simple progression of updates to a running stack with CloudFormation.
- Create a scaled and load-balanced application – Discover how to use CloudFormation to create a scalable and load-balanced application. This walkthrough covers creating an Auto Scaling group, a load balancer, and other related resources to ensure your application can handle varying traffic loads and maintain high availability.
- Peer with a VPC in another AWS account – This walkthrough guides you through the process of creating a Virtual Private Cloud (VPC) peering connection between two VPCs in different AWS accounts. VPC peering helps you route traffic between the VPCs and access resources as if they were part of the same network.
- Perform ECS blue/green deployments through CodeDeploy using CloudFormation – Discover how to use CloudFormation to perform AWS CodeDeploy blue/green deployments on Amazon ECS. Blue/green deployments are a way to update your applications or services with minimal downtime.

Refer to resource outputs in another CloudFormation stack

This walkthrough shows you how to reference outputs from one CloudFormation stack within another stack to create more modular and reusable templates.

Instead of including all resources in a single stack, you create related AWS resources in separate stacks. Then, you can refer to required resource outputs from other stacks. By restricting cross-stack references to outputs, you control the parts of a stack that are referenced by other stacks.

For example, you might have a network stack with a VPC, a security group, and a subnet for public web applications, and a separate public web application stack. To ensure that the web applications use the security group and subnet from the network stack, you create a cross-stack reference that allows the web application stack to reference resource outputs from the network stack. With a cross-stack reference, owners of the web application stacks don't need to create or maintain networking rules or assets.

To create a cross-stack reference, use the `Export` output field to flag the value of a resource output for export. Then, use the `Fn::ImportValue` intrinsic function to import the value. For more information, see Get exported outputs from a deployed CloudFormation stack.

> **Note**
>
> CloudFormation is a free service. However, you are charged for the AWS resources that you include in your stacks at the current rate for each one. For more information about AWS pricing, see the detail page for each product.

Topics

- Use a sample template to create a network stack
- Use a sample template to create a web application stack
- Verify the stack works as designed
- Troubleshoot AMI mapping errors
- Clean up your resources

Use a sample template to create a network stack

Before you begin this walkthrough, check that you have IAM permissions to use all of the following services: Amazon VPC, Amazon EC2, and CloudFormation.

The network stack contains the VPC, security group, and subnet that you will use in the web application stack. In addition to these resources, the network stack creates an Internet gateway and routing tables to enable public access.

You must create this stack before you create the web application stack. If you create the web application stack first, it won't have a security group or subnet.

The stack template is available from the following URL: https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleNetworkCrossStack.template. To see the resources that the stack will create, choose the link, which opens the template. In the `Outputs` section, you can see the networking resources that the sample template exports. The names of the exported resources are prefixed with the stack's name in case you export networking resources from other stacks. When users import networking resources, they can specify from which stack the resources are imported.

To create the network stack

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, choose **Create stack** at top right, and then choose **With new resources (standard)**.
3. Choose **Choose an existing template**, and in the **Specify template** section, choose **Amazon S3 URL**.
4. For **Amazon S3 URL**, paste the following URL: **`https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleNetworkCrossStack.template`**.
5. Choose **Next**.
6. For **Stack name**, type **`SampleNetworkCrossStack`**, and then choose **Next**.

   > **Note**
   >
   > Record the name of this stack. You'll need the stack name when you launch the web application stack.

7. Choose **Next**. For this walkthrough, you don't need to add tags or specify advanced settings.
8. Ensure that the stack name and template URL are correct, and then choose **Create stack**.

It might take several minutes for CloudFormation to create your stack. Wait until all resources have been successfully created before proceeding to create the web application stack.

9. To monitor progress, view the stack events. For more information, see Monitor stack progress.

Use a sample template to create a web application stack

The web application stack creates an EC2 instance that uses the security group and subnet from the network stack.

You must create this stack in the same AWS Region as the network stack.

The stack template is available from the following URL: https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template. To see the resources that the stack will create, choose the link, which will open the template. In the `Resources` section, view the EC2 instance's properties. You can see how the networking resources are imported from another stack by using the `Fn::ImportValue` function.

To create the web application stack

1. From the **Stacks** page, choose **Create stack** at top right, and then choose **With new resources (standard)**.
2. Choose **Choose an existing template**, and in the **Specify template** section, choose **Amazon S3 URL**.
3. For **Amazon S3 URL**, paste the following URL: **`https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template`**.
4. Choose **Next**.
5. For **Stack name**, type **`SampleWebAppCrossStack`**. In the **Parameters** section, use the default value for the **NetworkStackName** parameter, and then choose **Next**.

   The sample template uses the parameter value to specify from which stack to import values.
6. Choose **Next**. For this walkthrough, you don't need to add tags or specify advanced settings.
7. Ensure that the stack name and template URL are correct, and then choose **Create stack**.

   It might take several minutes for CloudFormation to create your stack.

Verify the stack works as designed

After the stack has been created, view its resources and note the instance ID. For more information on viewing stack resources, see View stack information from the CloudFormation console.

To verify the instance's security group and subnet, view the instance's properties in the Amazon EC2 console. If the instance uses the security group and subnet from the `SampleNetworkCrossStack` stack, you have successfully created a cross-stack reference.

Use the console to view the stack outputs and the example website URL to verify that the web application is running. For more information, see View stack information from the CloudFormation console.

Troubleshoot AMI mapping errors

If you receive the error `Template error: Unable to get mapping for AWSRegionArch2AMI::[region]::HVM64`, the template doesn't include an AMI mapping for your AWS Region. Instead of updating the mapping, we recommend using Systems Manager public parameters to dynamically reference the latest AMIs:

1. Download the `SampleWebAppCrossStack` template to your local machine from: https://s3.amazonaws.com/cloudformation-examples/user-guide/cross-stack/SampleWebAppCrossStack.template.
2. Delete the entire `AWSRegionArch2AMI` mapping section.
3. Add the following Systems Manager parameter:

```
"LatestAmiId": {
  "Description": "The latest Amazon Linux 2 AMI from the Parameter Store",
    "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
    "Default": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
  }
```

4. Replace the existing `ImageId` reference:

```
"ImageId": { "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" } ,
 "HVM64" ] },
```

   With:

```
"ImageId": { "Ref": "LatestAmiId" },
```

This parameter automatically resolves to the latest Amazon Linux 2 AMI for the region where you deploy the stack.

For other Linux distributions, use the appropriate parameter path. For more information, see Discovering public parameters in Parameter Store in the *AWS Systems Manager User Guide*.

5. Upload the modified template to an S3 bucket in your account:

```
aws s3 cp SampleWebAppCrossStack.template s3://amzn-s3-demo-bucket/
```

6. When creating the stack, specify your S3 template URL instead of the example URL.

Clean up your resources

To ensure that you are not charged for unwanted services, delete the stacks.

To delete the stacks

1. In the CloudFormation console, choose the `SampleWebAppCrossStack` stack.
2. Choose **Actions**, and then choose **Delete stack**.
3. In the confirmation message, choose **Delete**.
4. After the stack has been deleted, repeat the same steps for the `SampleNetworkCrossStack` stack.

> **Note**
>
> Wait until CloudFormation completely deletes the `SampleWebAppCrossStack` stack. If the EC2 instance is still running in the VPC, CloudFormation won't delete the VPC in the `SampleNetworkCrossStack` stack.

Deploy applications on Amazon EC2

You can use CloudFormation to automatically install, configure, and start applications on Amazon EC2 instances. Doing so enables you to easily duplicate deployments and update existing

installations without connecting directly to the instance, which can save you a lot of time and effort.

CloudFormation includes a set of helper scripts (`cfn-init`, `cfn-signal`, `cfn-get-metadata`, and `cfn-hup`) that are based on `cloud-init`. You call these helper scripts from your CloudFormation templates to install, configure, and update applications on Amazon EC2 instances that are in the same template. For more information, see the CloudFormation helper scripts reference in the *CloudFormation Template Reference Guide*.

In the getting started tutorial, you created a simple web server using `UserData` with a basic bash script. While this worked for a simple "Hello World" page, real applications often need more sophisticated configuration, including:

- Multiple software packages installed in the correct order.
- Complex configuration files created with specific content.
- Services started and configured to run automatically.
- Error handling and validation of the setup process.

CloudFormation's helper scripts provide a more robust and maintainable way to configure EC2 instances compared to basic bash scripts in `UserData`. The `cfn-init` helper script reads configuration data from your template's metadata and applies it systematically to your instance.

In this tutorial, you'll learn how to use the `cfn-init` helper script and monitor the bootstrapping process.

> **Note**
>
> CloudFormation is free, but you'll be charged for the Amazon EC2 resources you create. However, if you're new to AWS, you can take advantage of the Free Tier to minimize or eliminate costs during this learning process.

Topics

- Prerequisites
- Understanding bootstrap concepts
- Start with a simple bootstrap example
- Adding files and commands

- Adding network security
- The complete bootstrap template
- Create the stack using the console
- Monitor the bootstrap process
- Test the bootstrapped web server
- Troubleshooting bootstrap issues
- Clean up resources
- Next steps

Prerequisites

- You must have completed the Creating your first stack tutorial or have equivalent experience with CloudFormation basics.
- You must have access to an AWS account with an IAM user or role that has permissions to use Amazon EC2 and CloudFormation, or administrative user access.
- You must have a Virtual Private Cloud (VPC) that has access to the internet. This tutorial template requires a default VPC, which comes automatically with newer AWS accounts. If you don't have a default VPC, or if it was deleted, see the troubleshooting section in the Creating your first stack tutorial for alternative solutions.

Understanding bootstrap concepts

Let's understand the key concepts that make bootstrapping work before creating the template.

The `cfn-init` helper script

CloudFormation provides Python helper scripts that you can use to install software and start services on an Amazon EC2 instance. The `cfn-init` script reads resource metadata from your template and applies the configuration to your instance.

The process works as follows:

1. You define the configuration in the `Metadata` section of your EC2 resource.
2. You call `cfn-init` from the `UserData` script.
3. `cfn-init` reads the metadata and applies the configuration.
4. Your instance is configured according to your specifications.

Metadata structure

The configuration is defined in a specific structure within your EC2 instance.

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Metadata:                       # Metadata section for the resource
      AWS::CloudFormation::Init:    # Required key that cfn-init looks for
        config:                     # Configuration name (you can have multiple)
          packages:                 # Install packages
          files:                    # Create files
          commands:                 # Run commands
          services:                 # Start/stop services
```

The `cfn-init` script processes these sections in a specific order: packages, groups, users, sources, files, commands, and then services.

Start with a simple bootstrap example

Let's start with a minimal bootstrap example that just installs and starts Apache.

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:                 # Install Apache web server
            yum:
              httpd: []
          services:                 # Start Apache and enable it to start on boot
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      UserData: !Base64             # Script that runs when instance starts
        Fn::Sub: |
          #!/bin/bash
          yum install -y aws-cfn-bootstrap
```

```
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource EC2Instance --
region ${AWS::Region}
```

This simple example demonstrates the core concepts:

- `packages` section installs the `httpd` package using yum. This works on Amazon Linux and other Linux distributions that use yum.
- `services` section ensures `httpd` starts and runs automatically.
- `UserData` installs the latest bootstrap tools and calls `cfn-init`.

Adding files and commands

Now, let's enhance our example by adding a custom web page and a log file in the `/var/log` directory on the EC2 instance.

Creating files

The `files` section allows you to create files on the instance with specific content. The vertical pipe (`|`) allows you to pass a literal block of text (HTML code) as the content of the file (`/var/www/html/index.html`).

```
files:
  /var/www/html/index.html:
    content: |
      <body>
        <h1>Congratulations, you have successfully launched the AWS CloudFormation
 sample.</h1>
      </body>
```

Running commands

The `commands` section lets you run shell commands during the bootstrap process. This command creates a log file at `/var/log/welcome.txt` on the EC2 instance. To view it, you need an Amazon EC2 key pair to use for SSH access and an IP address range that can be used to SSH to the instance (not covered here).

```
commands:
  createWelcomeLog:
    command: "echo 'cfn-init ran successfully!' > /var/log/welcome.txt"
```

Adding network security

Since we're setting up a web server, we need to allow web traffic (HTTP) to reach our EC2 instance. To do this, we'll create a security group that allows incoming traffic on port 80 from your IP address. EC2 instances also need to send traffic out to the internet, for example, to install package updates. By default, security groups allow all outgoing traffic. We'll then associate this security group with our EC2 instance using the `SecurityGroupIds` property.

```
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow HTTP access from my IP address
    SecurityGroupIngress:
      - IpProtocol: tcp
        Description: HTTP
        FromPort: 80
        ToPort: 80
        CidrIp: !Ref MyIP
```

The complete bootstrap template

Now, let's put all the pieces together. Here's the complete template that combines all the concepts we've discussed.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Bootstrap an EC2 instance with Apache web server using cfn-init

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.
```

```
  MyIP:
    Description: Your IP address in CIDR format (e.g. 203.0.113.1/32)
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access from my IP address
      SecurityGroupIngress:
        - IpProtocol: tcp
          Description: HTTP
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP

  WebServer:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:
            yum:
              httpd: []
          files:
            /var/www/html/index.html:
              content: |
                <body>
                  <h1>Congratulations, you have successfully launched the AWS
 CloudFormation sample.</h1>
                </body>
          commands:
            createWelcomeLog:
              command: "echo 'cfn-init ran successfully!' > /var/log/welcome.txt"
          services:
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
```

```
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData: !Base64
        Fn::Sub: |
          #!/bin/bash
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource WebServer --
region ${AWS::Region}
      Tags:
        - Key: Name
          Value: Bootstrap Tutorial Web Server

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServer.PublicDnsName}'
    Description: EC2 instance public DNS name
```

Create the stack using the console

The following procedure involves uploading the sample stack template from a file. Open a text editor on your local machine and add the template. Save the file with the name `samplelinux2stack.template`.

To launch the stack template

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Choose **Create stack**, **With new resources (standard)**.
3. Under **Specify template**, choose **Upload a template file**, **Choose file** to upload the `samplelinux2stack.template` file.
4. Choose **Next**.
5. On the **Specify stack details** page, type **`BootstrapTutorialStack`** as the stack name.
6. Under **Parameters**, do the following.
   - **LatestAmiId**: Leave the default value.
   - **InstanceType**: Choose either **t2.micro** or **t3.micro** for the EC2 instance type.

- **MyIP**: Enter your public IP address with a `/32` suffix.

7. Choose **Next** twice, then **Submit** to create the stack.

Monitor the bootstrap process

Bootstrap processes take longer than simple EC2 launches because additional software is being installed and configured.

To monitor bootstrap progress

1. In the CloudFormation console, select your stack and open the **Events** tab.
2. Watch for the `WebServer CREATE_IN_PROGRESS` event. The bootstrap process begins after the instance launches.
3. The bootstrap process typically takes a few minutes. You'll see `WebServer CREATE_COMPLETE` when it's finished.

If you want to see what's happening during the bootstrap process, you can check the instance logs.

To view bootstrap logs (optional)

1. Open the EC2 console and find your instance.
2. Select the instance, and then choose **Actions**, **Monitor and troubleshoot**, **Get system log** to see the bootstrap progress.
3. If you don't see the logs immediately, wait and refresh the page.

Test the bootstrapped web server

When your stack shows `CREATE_COMPLETE`, test your web server.

To test the web server

1. In the CloudFormation console, go to the **Outputs** tab for your stack.
2. Click on the **WebsiteURL** value to open your web server in a new tab.
3. You should see your custom web page with the message `Congratulations, you have successfully launched the AWS CloudFormation sample`.

> **Note**
>
> If the page doesn't load immediately, wait a minute and try again. The bootstrap process may still be completing even after the stack shows `CREATE_COMPLETE`.

Troubleshooting bootstrap issues

If your bootstrap process fails or your web server isn't working, here are common issues and solutions.

Common issues

- **Stack creation fails** – Check the **Events** tab for specific error messages.
- **Web server not accessible** – Verify your IP address is correct in the `MyIP` parameter. Remember to include `/32` at the end.
- **Bootstrap process fails** – The instance may launch but `cfn-init` fails. Check the system logs as described in the monitoring section.

Clean up resources

To avoid ongoing charges, you can clean up by deleting the stack and its resources.

To delete the stack and its resources

1. Open the CloudFormation console.
2. On the **Stacks** page, select the option next to the name of the stack you created (**`BootstrapTutorialStack`**) and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.
4. Monitor the progress of the stack deletion process on the **Event** tab. The status for **`BootstrapTutorialStack`** changes to `DELETE_IN_PROGRESS`. When CloudFormation completes the deletion of the stack, it removes the stack from the list.

Next steps

Congratulations! You've successfully learned how to bootstrap EC2 instances with CloudFormation. You now understand:

- How to use `cfn-init` helper scripts
- How to structure metadata for bootstrapping
- How to install packages, create files, run commands, and manage services
- How to monitor for bootstrap issues

To continue learning:

- Learn how to update a running stack and use the `cfn-hup` helper script. For more information, see Update a CloudFormation stack.
- Learn how to bootstrap a Windows stack. For more information, see Bootstrapping Windows-based CloudFormation stacks.
- Explore more complex bootstrap scenarios with multiple configuration sets. For more information, see cfn-init and AWS::CloudFormation::Init in the *CloudFormation Template Reference Guide*.
- Learn about `cfn-signal` for reporting bootstrap completion status. For more information, see cfn-signal in the *CloudFormation Template Reference Guide*.

Update a CloudFormation stack

> **Note**
>
> This tutorial builds on concepts from the Deploy applications on Amazon EC2 tutorial. If you haven't completed that tutorial, we recommend doing so first to understand EC2 bootstrapping with CloudFormation.

This topic demonstrates a simple progression of updates to a running stack. We will walk through the following steps:

1. **Create the initial stack** – Create a stack using a base Amazon Linux 2 AMI, installing the Apache Web Server and a simple PHP application using the CloudFormation helper scripts.
2. **Update the application** – Update one of the files in the application and deploy the software using CloudFormation.
3. **Add a key pair** – Add an Amazon EC2 key pair to the instance, and then update the security group to allow SSH access to the instance.

4. **Update the instance type** – Change the instance type of the underlying Amazon EC2 instance.
5. **Update the AMI** – Change the Amazon Machine Image (AMI) for the Amazon EC2 instance in your stack.

> **Note**
>
> CloudFormation is free, but you'll be charged for the Amazon EC2 resources you create. However, if you're new to AWS, you can take advantage of the Free Tier to minimize or eliminate costs during this learning process.

Topics

- Step 1: Create the initial stack
- Step 2: Update the application
- Step 3: Add SSH access with a key pair
- Step 4: Update the instance type
- Step 5: Update the AMI
- Availability and impact considerations
- Related resources

Step 1: Create the initial stack

We'll begin by creating a stack that we can use throughout the rest of this topic. We have provided a simple template that launches a single instance PHP web application hosted on the Apache Web Server and running on an Amazon Linux 2 AMI.

The Apache Web Server, PHP, and the simple PHP application are all installed by the CloudFormation helper scripts that are installed by default on the Amazon Linux 2 AMI. The following template snippet shows the metadata that describes the packages and files to install, in this case the Apache Web Server and the PHP infrastructure from the Yum repository for the Amazon Linux 2 AMI. The snippet also shows the `Services` section, which ensures that the Apache Web Server is running.

```
WebServerInstance:
  Type: AWS::EC2::Instance
```

```
  Metadata:
    AWS::CloudFormation::Init:
      config:
        packages:
          yum:
            httpd: []
            php: []
        files:
          /var/www/html/index.php:
            content: |
              <?php
              echo '<h1>Hello World!</h1>';
              ?>
            mode: '000644'
            owner: apache
            group: apache
        services:
          systemd:
            httpd:
              enabled: true
              ensureRunning: true
```

The application itself is a "Hello World" example that's entirely defined within the template. For a real-world application, the files may be stored on Amazon S3, GitHub, or another repository and referenced from the template. CloudFormation can download packages (such as RPMs or RubyGems), and reference individual files and expand `.zip` and `.tar` files to create the application artifacts on the Amazon EC2 instance.

The template enables and configures the `cfn-hup` daemon to listen for changes to the configuration defined in the metadata for the Amazon EC2 instance. By using the `cfn-hup` daemon, you can update application software, such as the version of Apache or PHP, or you can update the PHP application file itself from CloudFormation. The following snippet from the same Amazon EC2 resource in the template shows the pieces necessary to configure `cfn-hup` to call `cfn-init` every two minutes to notice and apply updates to the metadata. Otherwise, the `cfn-init` only runs once at start up.

```
files:
  /etc/cfn/cfn-hup.conf:
    content: !Sub |
      [main]
      stack=${AWS::StackId}
```

```
      region=${AWS::Region}
      # The interval used to check for changes to the resource metadata in minutes.
 Default is 15
      interval=2
    mode: '000400'
    owner: root
    group: root
  /etc/cfn/hooks.d/cfn-auto-reloader.conf:
    content: !Sub |
      [cfn-auto-reloader-hook]
      triggers=post.update
      path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
      action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --region
 ${AWS::Region}
      runas=root
services:
  systemd:
    cfn-hup:
      enabled: true
      ensureRunning: true
      files:
        - /etc/cfn/cfn-hup.conf
        - /etc/cfn/hooks.d/cfn-auto-reloader.conf
```

To complete the stack, in the `Properties` section of the Amazon EC2 instance definition, the `UserData` property contains the `cloud-init` script that calls `cfn-init` to install the packages and files. For more information, see the CloudFormation helper scripts reference in the *CloudFormation Template Reference Guide*. The template also creates an Amazon EC2 security group.

```
AWSTemplateFormatVersion: 2010-09-09

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues:
```

```
      - t3.nano
      - t3.micro
      - t3.small
      - t3.medium
      - t3a.nano
      - t3a.micro
      - t3a.small
      - t3a.medium
      - m5.large
      - m5.xlarge
      - m5.2xlarge
      - m5a.large
      - m5a.xlarge
      - m5a.2xlarge
      - c5.large
      - c5.xlarge
      - c5.2xlarge
      - r5.large
      - r5.xlarge
      - r5.2xlarge
      - r5a.large
      - r5a.xlarge
      - r5a.2xlarge
    ConstraintDescription: must be a valid EC2 instance type.

Resources:
  WebServerInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          # Get the latest CloudFormation package
          yum update -y aws-cfn-bootstrap
          # Run cfn-init
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 WebServerInstance --region ${AWS::Region} || error_exit 'Failed to run cfn-init'

          # Start up the cfn-hup daemon to listen for changes to the EC2 instance
 metadata
```

```
          /opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'
          # Signal success or failure
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerInstance --region ${AWS::Region}
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:
            yum:
              httpd: []
              php: []
          files:
            /var/www/html/index.php:
              content: |
                <?php
                echo "<h1>Hello World!</h1>";
                ?>
              mode: '000644'
              owner: apache
              group: apache
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
                # The interval used to check for changes to the resource metadata in
 minutes. Default is 15
                interval=2
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
                action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --
region ${AWS::Region}
                runas=root
          services:
            systemd:
              httpd:
                enabled: true
                ensureRunning: true
```

```
              cfn-hup:
                enabled: true
                ensureRunning: true
                files:
                  - /etc/cfn/cfn-hup.conf
                  - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    CreationPolicy:
      ResourceSignal:
        Timeout: PT5M

  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServerInstance.PublicDnsName}'
    Description: URL of the web application
```

To launch a stack from this template

1. Copy the template and save it locally on your system as a text file. Note the location because you'll need to use the file in a subsequent step.
2. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. Choose **Create stack, With new resources (standard)**.
4. Choose **Choose an existing template**.
5. Under **Specify template**, choose **Upload a template file** and browse to the file that you created in the first step, and then choose **Next**.
6. On the **Specify stack details** page, enter **`UpdateTutorial`** as the stack name.
7. Under **Parameters**, keep all parameters the same and choose **Next** twice.
8. On the **Review and create** screen, choose **Submit**.

After the status of your stack is CREATE_COMPLETE, the **Outputs** tab will display the URL of your website. If you choose the value of the WebsiteURL output, you will see your new PHP application working.

Step 2: Update the application

Now that we have deployed the stack, let's update the application. We'll make a simple change to the text that's printed out by the application. To do so, we'll add an echo command to the index.php file as shown in this template snippet:

```
files:
  /var/www/html/index.php:
    content: |
      <?php
      echo "<h1>Hello World!</h1>";
      echo "<p>This is an updated version of our application.</p>";
      ?>
    mode: '000644'
    owner: apache
    group: apache
```

Use a text editor to manually edit the template file that you saved locally.

Now, update the stack.

To update the stack with your updated template

1. In the CloudFormation console, select your **UpdateTutorial** stack.
2. Choose **Update, Make a direct update**.
3. Choose **Replace existing template**.
4. Under **Specify template**, choose **Upload a template file** and upload your modified template file, and then choose **Next**.
5. On the **Specify stack details** page, keep all parameters the same and choose **Next** twice.
6. On the **Review** page, review the changes. Under **Changes**, you should see that CloudFormation will update the WebServerInstance resource.
7. Choose **Submit**.

When your stack is in the UPDATE_COMPLETE state, you can choose the WebsiteURL output value again to verify that the changes to your application have taken effect. The cfn-hup daemon runs

every 2 minutes, so it may take up to 2 minutes for the application to change once the stack has been updated.

To see the set of resources that were updated, go to the CloudFormation console. On the **Events** tab, look at the stack events. In this particular case, the metadata for the Amazon EC2 instance `WebServerInstance` was updated, which caused CloudFormation to also reevaluate the other resources (`WebServerSecurityGroup`) to ensure that there were no other changes. None of the other stack resources were modified. CloudFormation will update only those resources in the stack that are affected by any changes to the stack. Such changes can be direct, such as property or metadata changes, or they can be due to dependencies or data flows through `Ref`, `GetAtt`, or other intrinsic template functions. For more information, see Intrinsic function reference.

This simple update illustrates the process. However, you can make much more complex changes to the files and packages that are deployed to your Amazon EC2 instances. For example, you might decide that you need to add MySQL to the instance, along with PHP support for MySQL. To do so, simply add the additional packages and files along with any additional services to the configuration and then update the stack to deploy the changes.

```
packages:
  yum:
    httpd: []
    php: []
    mysql: []
    php-mysql: []
    mysql-server: []
    mysql-libs: []

  ...

services:
  systemd:
    httpd:
      enabled: true
      ensureRunning: true
    cfn-hup:
      enabled: true
      ensureRunning: true
      files:
        - /etc/cfn/cfn-hup.conf
        - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    mysqld:
```

```
      enabled: true
      ensureRunning: true
```

You can update the CloudFormation metadata to update to new versions of the packages used by the application. In the previous examples, the version property for each package is empty, indicating that `cfn-init` should install the latest version of the package.

```
packages:
  yum:
    httpd: []
    php: []
```

You can optionally specify a version string for a package. If you change the version string in subsequent update stack calls, the new version of the package will be deployed. Here's an example of using version numbers for RubyGems packages. Any package that supports versioning can have specific versions.

```
packages:
  rubygems:
    mysql: []
    rubygems-update:
      - "1.6.2"
    rake:
      - "0.8.7"
    rails:
      - "2.3.11"
```

Step 3: Add SSH access with a key pair

You can also update a resource in the template to add properties that weren't originally specified in the template. To illustrate that, we'll add an Amazon EC2 key pair to an existing EC2 instance and then open up port 22 in the Amazon EC2 security group so that you can use Secure Shell (SSH) to access the instance.

To add SSH access to an existing Amazon EC2 instance

1. Add two additional parameters to the template to pass in the name of an existing Amazon EC2 key pair and SSH location.

   ```
   Parameters:
   ```

```
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the
 instance
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.

  SSHLocation:
    Description: The IP address that can be used to SSH to the EC2 instances in
 CIDR format (e.g. 203.0.113.1/32)
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
```

2. Add the KeyName property to the Amazon EC2 instance.

```
WebServerInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref LatestAmiId
    InstanceType: !Ref InstanceType
    KeyName: !Ref KeyName
    SecurityGroupIds:
      - !Ref WebServerSecurityGroup
```

3. Add port 22 and the SSH location to the ingress rules for the Amazon EC2 security group.

```
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref SSHLocation
```

4. Update the stack using the same steps as explained in Step 2: Update the application.

Step 4: Update the instance type

Now let's demonstrate how to update the underlying infrastructure by changing the instance type.

The stack we have built so far uses a t3.micro Amazon EC2 instance. Let's suppose that your newly created website is getting more traffic than a t3.micro instance can handle, and now you want to move to an m5.large Amazon EC2 instance type. If the architecture of the instance type changes, the instance must be created with a different AMI. However, both the t3.micro and m5.large use the same CPU architectures and run Amazon Linux 2 (x86_64) AMIs . For more information, see Compatibility for changing the instance type in the *Amazon EC2 User Guide*.

Let's use the template that we modified in the previous step to change the instance type. Because `InstanceType` was an input parameter to the template, we don't need to modify the template; we can change the value of the parameter on the **Specify stack details** page.

To update the stack with a new parameter value

1. In the CloudFormation console, select your **`UpdateTutorial`** stack.
2. Choose **Update, Make a direct update**.
3. Choose **Use existing template**, and then choose **Next**.
4. On the **Specify stack details** page, change the value of the **InstanceType** text box from `t3.micro` to `m5.large`. Then, choose **Next** twice.
5. On the **Review** page, review the changes. Under **Changes**, you should see that CloudFormation will update the `WebServerInstance` resource.
6. Choose **Submit**.

You can dynamically change the instance type of an EBS-backed Amazon EC2 instance by starting and stopping the instance. CloudFormation tries to optimize the change by updating the instance type and restarting the instance, so the instance ID doesn't change. When the instance is restarted, however, the public IP address of the instance does change. To ensure that the Elastic IP address is bound correctly after the change, CloudFormation will also update the Elastic IP address. You can see the changes in the CloudFormation console on the **Events** tab.

To check the instance type from the AWS Management Console, open the Amazon EC2 console, and locate your instance there.

Step 5: Update the AMI

Now let's update our stack to use Amazon Linux 2023, which is the next generation of Amazon Linux.

Updating the AMI is a major change that requires replacing the instance. We can't simply start and stop the instance to modify the AMI; CloudFormation considers this a change to an immutable property of the resource. In order to make a change to an immutable property, CloudFormation must launch a replacement resource, in this case a new Amazon EC2 instance running the new AMI.

Let's look at how we might update our stack template to use Amazon Linux 2023. The key changes include updating the AMI parameter and changing from `yum` to `dnf` package manager.

```
AWSTemplateFormatVersion: 2010-09-09

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2023 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues:
      - t3.nano
      - t3.micro
      - t3.small
      - t3.medium
      - t3a.nano
      - t3a.micro
      - t3a.small
      - t3a.medium
      - m5.large
      - m5.xlarge
      - m5.2xlarge
      - m5a.large
      - m5a.xlarge
      - m5a.2xlarge
      - c5.large
      - c5.xlarge
      - c5.2xlarge
```

```
      - r5.large
      - r5.xlarge
      - r5.2xlarge
      - r5a.large
      - r5a.xlarge
      - r5a.2xlarge
    ConstraintDescription: must be a valid EC2 instance type.

  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.

  SSHLocation:
    Description: The IP address that can be used to SSH to the EC2 instances in CIDR
 format (e.g. 203.0.113.1/32)
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}\/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      KeyName: !Ref KeyName
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          # Get the latest CloudFormation package
          dnf update -y aws-cfn-bootstrap
          # Run cfn-init
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 WebServerInstance --region ${AWS::Region} || error_exit 'Failed to run cfn-init'

          # Start up the cfn-hup daemon to listen for changes to the EC2 instance
 metadata
          /opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'
```

```
          # Signal success or failure
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerInstance --region ${AWS::Region}
    Metadata:
      AWS::CloudFormation::Init:
        config:
          packages:
            dnf:
              httpd: []
              php: []
          files:
            /var/www/html/index.php:
              content: |
                <?php
                echo "<h1>Hello World!</h1>";
                echo "<p>This is an updated version of our application.</p>";
                echo "<p>Running on Amazon Linux 2023!</p>";
                ?>
              mode: '000644'
              owner: apache
              group: apache
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
                # The interval used to check for changes to the resource metadata in
 minutes. Default is 15
                interval=2
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.WebServerInstance.Metadata.AWS::CloudFormation::Init
                action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerInstance --
region ${AWS::Region}
                runas=root
          services:
            systemd:
              httpd:
                enabled: true
```

```
                ensureRunning: true
              cfn-hup:
                enabled: true
                ensureRunning: true
                files:
                  - /etc/cfn/cfn-hup.conf
                  - /etc/cfn/hooks.d/cfn-auto-reloader.conf
    CreationPolicy:
      ResourceSignal:
        Timeout: PT5M

  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref SSHLocation

Outputs:
  WebsiteURL:
    Value: !Sub 'http://${WebServerInstance.PublicDnsName}'
    Description: URL of the web application
```

Update the stack using the same steps as explained in Step 2: Update the application.

After the new instance is running, CloudFormation updates the other resources in the stack to point to the new resource. When all new resources are created, the old resource is deleted, a process known as UPDATE_CLEANUP. This time, you will notice that the instance ID and application URL of the instance in the stack has changed as a result of the update. The events in the **Event** table contain a description "Requested update has a change to an immutable property and hence creating a new physical resource" to indicate that a resource was replaced.

Alternatively: If you have application code written into the AMI that you want to update, you can use the same stack update mechanism to update the AMI to load your new application.

To update the AMI with custom application code

1. Create your new AMI containing your application or operating system changes. For more information, see Create an Amazon EBS-backed AMI in the *Amazon EC2 User Guide*.
2. Update your template to incorporate the new AMI ID.
3. Update the stack using the same steps as explained in Step 2: Update the application.

When you update the stack, CloudFormation detects that the AMI ID has changed, and then it triggers a stack update in the same way as we initiated the one above.

Availability and impact considerations

Different properties have different impacts on the resources in the stack. You can use CloudFormation to update any property; however, before you make any changes, you should consider these questions:

1. How does the update affect the resource itself? For example, updating an alarm threshold will render the alarm inactive during the update. As we have seen, changing the instance type requires that the instance be stopped and restarted. CloudFormation uses the update or modify actions for the underlying resources to make changes to resources. To understand the impact of updates, you should check the documentation for the specific resources.
2. Is the change mutable or immutable? Some changes to resource properties, such as changing the AMI on an Amazon EC2 instance, aren't supported by the underlying services. In the case of mutable changes, CloudFormation will use the Update or Modify type APIs for the underlying resources. For immutable property changes, CloudFormation will create new resources with the updated properties and then link them to the stack before deleting the old resources. Although CloudFormation tries to reduce the down time of the stack resources, replacing a resource is a multistep process, and it will take time. During stack reconfiguration, your application will not be fully operational. For example, it may not be able to serve requests or access a database.

Related resources

For more information about using CloudFormation to start applications and on integrating with other configuration and deployment services such as Puppet and Opscode Chef, see the following whitepapers:

- Bootstrapping applications via CloudFormation

- Integrating CloudFormation with Opscode Chef
- Integrating CloudFormation with Puppet

Create a scaled and load-balanced application

For this walkthrough, you create a stack that helps you set up a scaled and load-balanced application. The walkthrough provides a sample template that you use to create the stack. The example template provisions an Auto Scaling group, an Application Load Balancer, security groups that control traffic to the load balancer and to the Auto Scaling group, and an Amazon SNS notification configuration to publish notifications about scaling activities.

This template creates one or more Amazon EC2 instances and an Application Load Balancer. You will be billed for the AWS resources used if you create a stack from this template.

Full stack template

Let's start with the template.

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  InstanceType:
    Description: The EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues:
      - t3.micro
      - t3.small
      - t3.medium
  KeyName:
    Description: Name of an existing EC2 key pair to allow SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'
  OperatorEmail:
    Description: The email address to notify when there are any scaling activities
    Type: String
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
```

```
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  Subnets:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: At least two public subnets in different Availability Zones in the
 selected VPC
  VPC:
    Type: AWS::EC2::VPC::Id
    Description: A virtual private cloud (VPC) that enables resources in public subnets
 to connect to the internet
Resources:
  ELBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: ELB Security Group
      VpcId: !Ref VPC
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
  EC2SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: EC2 Security Group
      VpcId: !Ref VPC
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        SourceSecurityGroupId:
          Fn::GetAtt:
          - ELBSecurityGroup
          - GroupId
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref SSHLocation
  EC2TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
```

```
      HealthCheckIntervalSeconds: 30
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 15
      HealthyThresholdCount: 5
      Matcher:
        HttpCode: '200'
      Name: EC2TargetGroup
      Port: 80
      Protocol: HTTP
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: '20'
      UnhealthyThresholdCount: 3
      VpcId: !Ref VPC
  ALBListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref EC2TargetGroup
      LoadBalancerArn: !Ref ApplicationLoadBalancer
      Port: 80
      Protocol: HTTP
  ApplicationLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      Subnets: !Ref Subnets
      SecurityGroups:
        - !GetAtt ELBSecurityGroup.GroupId
  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: !Ref LatestAmiId
        InstanceType: !Ref InstanceType
        KeyName: !Ref KeyName
        SecurityGroupIds:
          - !Ref EC2SecurityGroup
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            yum update -y
```

```
            yum install -y httpd
            systemctl start httpd
            systemctl enable httpd
            echo "<h1>Hello World!</h1>" > /var/www/html/index.html
  NotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !Ref OperatorEmail
          Protocol: email
  WebServerGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref LaunchTemplate
        Version: !GetAtt LaunchTemplate.LatestVersionNumber
      MaxSize: '3'
      MinSize: '1'
      NotificationConfigurations:
        - TopicARN: !Ref NotificationTopic
          NotificationTypes: ['autoscaling:EC2_INSTANCE_LAUNCH',
 'autoscaling:EC2_INSTANCE_LAUNCH_ERROR', 'autoscaling:EC2_INSTANCE_TERMINATE',
 'autoscaling:EC2_INSTANCE_TERMINATE_ERROR']
      TargetGroupARNs:
        - !Ref EC2TargetGroup
      VPCZoneIdentifier: !Ref Subnets
```

JSON

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Parameters":{
    "InstanceType":{
      "Description":"The EC2 instance type",
      "Type":"String",
      "Default":"t3.micro",
      "AllowedValues":[
        "t3.micro",
        "t3.small",
        "t3.medium"
      ]
    },
    "KeyName":{
```

```
      "Description":"Name of an existing EC2 key pair to allow SSH access to the
instances",
      "Type":"AWS::EC2::KeyPair::KeyName"
    },
    "LatestAmiId":{
      "Description":"The latest Amazon Linux 2 AMI from the Parameter Store",
      "Type":"AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
      "Default":"/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
    },
    "OperatorEmail":{
      "Description":"The email address to notify when there are any scaling
activities",
      "Type":"String"
    },
    "SSHLocation":{
      "Description":"The IP address range that can be used to SSH to the EC2
instances",
      "Type":"String",
      "MinLength":9,
      "MaxLength":18,
      "Default":"0.0.0.0/0",
      "ConstraintDescription":"Must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "Subnets":{
      "Type":"List<AWS::EC2::Subnet::Id>",
      "Description":"At least two public subnets in different Availability Zones in the
selected VPC"
    },
    "VPC":{
      "Type":"AWS::EC2::VPC::Id",
      "Description":"A virtual private cloud (VPC) that enables resources in public
subnets to connect to the internet"
    }
  },
  "Resources":{
    "ELBSecurityGroup":{
      "Type":"AWS::EC2::SecurityGroup",
      "Properties":{
        "GroupDescription":"ELB Security Group",
        "VpcId":{
          "Ref":"VPC"
        },
        "SecurityGroupIngress":[
          {
```

```
            "IpProtocol":"tcp",
            "FromPort":80,
            "ToPort":80,
            "CidrIp":"0.0.0.0/0"
          }
        ]
      }
    },
    "EC2SecurityGroup":{
      "Type":"AWS::EC2::SecurityGroup",
      "Properties":{
        "GroupDescription":"EC2 Security Group",
        "VpcId":{
          "Ref":"VPC"
        },
        "SecurityGroupIngress":[
          {
            "IpProtocol":"tcp",
            "FromPort":80,
            "ToPort":80,
            "SourceSecurityGroupId":{
              "Fn::GetAtt":[
                "ELBSecurityGroup",
                "GroupId"
              ]
            }
          },
          {
            "IpProtocol":"tcp",
            "FromPort":22,
            "ToPort":22,
            "CidrIp":{
              "Ref":"SSHLocation"
            }
          }
        ]
      }
    },
    "EC2TargetGroup":{
      "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties":{
        "HealthCheckIntervalSeconds":30,
        "HealthCheckProtocol":"HTTP",
        "HealthCheckTimeoutSeconds":15,
```

```
        "HealthyThresholdCount":5,
        "Matcher":{
          "HttpCode":"200"
        },
        "Name":"EC2TargetGroup",
        "Port":80,
        "Protocol":"HTTP",
        "TargetGroupAttributes":[
          {
            "Key":"deregistration_delay.timeout_seconds",
            "Value":"20"
          }
        ],
        "UnhealthyThresholdCount":3,
        "VpcId":{
          "Ref":"VPC"
        }
      }
    },
    "ALBListener":{
      "Type":"AWS::ElasticLoadBalancingV2::Listener",
      "Properties":{
        "DefaultActions":[
          {
            "Type":"forward",
            "TargetGroupArn":{
              "Ref":"EC2TargetGroup"
            }
          }
        ],
        "LoadBalancerArn":{
          "Ref":"ApplicationLoadBalancer"
        },
        "Port":80,
        "Protocol":"HTTP"
      }
    },
    "ApplicationLoadBalancer":{
      "Type":"AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties":{
        "Scheme":"internet-facing",
        "Subnets":{
          "Ref":"Subnets"
        },
```

```
        "SecurityGroups":[
          {
            "Fn::GetAtt":[
              "ELBSecurityGroup",
              "GroupId"
            ]
          }
        ]
      }
    },
    "LaunchTemplate":{
      "Type":"AWS::EC2::LaunchTemplate",
      "Properties":{
        "LaunchTemplateName":{
          "Fn::Sub":"${AWS::StackName}-launch-template"
        },
        "LaunchTemplateData":{
          "ImageId":{
            "Ref":"LatestAmiId"
          },
          "InstanceType":{
            "Ref":"InstanceType"
          },
          "KeyName":{
            "Ref":"KeyName"
          },
          "SecurityGroupIds":[
            {
              "Ref":"EC2SecurityGroup"
            }
          ],
          "UserData":{
            "Fn::Base64":{
              "Fn::Join":[
                "",
                [
                  "#!/bin/bash\n",
                  "yum update -y\n",
                  "yum install -y httpd\n",
                  "systemctl start httpd\n",
                  "systemctl enable httpd\n",
                  "echo \"<h1>Hello World!</h1>\" > /var/www/html/index.html"
                ]
              ]
```

```
          }
        }
      }
    }
  },
  "NotificationTopic":{
    "Type":"AWS::SNS::Topic",
    "Properties":{
      "Subscription":[
        {
          "Endpoint":{
            "Ref":"OperatorEmail"
          },
          "Protocol":"email"
        }
      ]
    }
  },
  "WebServerGroup":{
    "Type":"AWS::AutoScaling::AutoScalingGroup",
    "Properties":{
      "LaunchTemplate":{
        "LaunchTemplateId":{
          "Ref":"LaunchTemplate"
        },
        "Version":{
          "Fn::GetAtt":[
            "LaunchTemplate",
            "LatestVersionNumber"
          ]
        }
      },
      "MaxSize":"3",
      "MinSize":"1",
      "NotificationConfigurations":[
        {
          "TopicARN":{
            "Ref":"NotificationTopic"
          },
          "NotificationTypes":[
            "autoscaling:EC2_INSTANCE_LAUNCH",
            "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
            "autoscaling:EC2_INSTANCE_TERMINATE",
            "autoscaling:EC2_INSTANCE_TERMINATE_ERROR"
```

```
            ]
          }
        ],
        "TargetGroupARNs":[
          {
            "Ref":"EC2TargetGroup"
          }
        ],
        "VPCZoneIdentifier":{
          "Ref":"Subnets"
        }
      }
    }
  }
}
```

Template walkthrough

The first part of this template specifies the `Parameters`. Each parameter must be assigned a value at runtime for CloudFormation to successfully provision the stack. Resources specified later in the template reference these values and use the data.

- `InstanceType`: The type of EC2 instance that Amazon EC2 Auto Scaling provisions. If not specified, a default of `t3.micro` is used.
- `KeyName`: An existing EC2 key pair to allow SSH access to the instances.
- `LatestAmiId`: The Amazon Machine Image (AMI) for the instances. If not specified, your instances are launched with an Amazon Linux 2 AMI, using an AWS Systems Manager public parameter maintained by AWS. For more information, see Finding public parameters in the *AWS Systems Manager User Guide*.
- `OperatorEmail`: The email address where you want to send scaling activity notifications.
- `SSHLocation`: The IP address range that can be used to SSH to the instances.
- `Subnets`: At least two public subnets in different Availability Zones.
- `VPC`: A virtual private cloud (VPC) in your account that enables resources in public subnets to connect to the internet.

> **Note**
>
> You can use the default VPC and default subnets to allow instances to access the internet. If using your own VPC, make sure that it has a subnet mapped to each

Availability Zone of the Region you are working in. At minimum, you must have two public subnets available to create the load balancer.

The next part of this template specifies the `Resources`. This section specifies the stack resources and their properties.

AWS::EC2::SecurityGroup resource `ELBSecurityGroup`

- `SecurityGroupIngress` contains a TCP ingress rule that allows access from *all IP addresses* ("CidrIp" : "0.0.0.0/0") on port 80.

AWS::EC2::SecurityGroup resource `EC2SecurityGroup`

- `SecurityGroupIngress` contains two ingress rules: 1) a TCP ingress rule that allows SSH access (port 22) from the IP address range that you provide for the `SSHLocation` input parameter and 2) a TCP ingress rule that allows access from the load balancer by specifying the load balancer's security group. The GetAtt function is used to get the ID of the security group with the logical name `ELBSecurityGroup`.

AWS::ElasticLoadBalancingV2::TargetGroup resource `EC2TargetGroup`

- `Port`, `Protocol`, and `HealthCheckProtocol` specify the EC2 instance port (80) and protocol (HTTP) that the `ApplicationLoadBalancer` routes traffic to and that Elastic Load Balancing uses to check the health of the EC2 instances.
- `HealthCheckIntervalSeconds` specifies that the EC2 instances have an interval of 30 seconds between health checks. The `HealthCheckTimeoutSeconds` is defined as the length of time Elastic Load Balancing waits for a response from the health check target (15 seconds in this example). After the timeout period lapses, Elastic Load Balancing marks that EC2 instance's health check as unhealthy. When an EC2 instance fails three consecutive health checks (`UnhealthyThresholdCount`), Elastic Load Balancing stops routing traffic to that EC2 instance until that instance has five consecutive healthy health checks (`HealthyThresholdCount`). At that point, Elastic Load Balancing considers the instance healthy and begins routing traffic to the instance again.
- `TargetGroupAttributes` updates the deregistration delay value of the target group to 20 seconds. By default, Elastic Load Balancing waits 300 seconds before completing the deregistration process.

AWS::ElasticLoadBalancingV2::Listener resource `ALBListener`

- `DefaultActions` specifies the port that the load balancer listens to, the target group where the load balancer forwards requests, and the protocol used to route requests.

AWS::ElasticLoadBalancingV2::LoadBalancer resource `ApplicationLoadBalancer`

- `Subnets` takes the value of the `Subnets` input parameter as the list of public subnets where the load balancer nodes will be created.
- `SecurityGroup` gets the ID of the security group that acts as a virtual firewall for your load balancer nodes to control incoming traffic. The GetAtt function is used to get the ID of the security group with the logical name `ELBSecurityGroup`.

AWS::EC2::LaunchTemplate resource `LaunchTemplate`

- `ImageId` takes the value of the `LatestAmiId` input parameter as the AMI to use.
- `KeyName` takes the value of the `KeyName` input parameter as the EC2 key pair to use.
- `SecurityGroupIds` gets the ID of the security group with the logical name `EC2SecurityGroup` that acts as a virtual firewall for your EC2 instances to control incoming traffic.
- `UserData` is a configuration script that runs after the instance is up and running. In this example, the script installs Apache and creates an index.html file.

AWS::SNS::Topic resource `NotificationTopic`

- `Subscription` takes the value of the `OperatorEmail` input parameter as the email address for the recipient of the notifications when there are any scaling activities.

AWS::AutoScaling::AutoScalingGroup resource `WebServerGroup`

- `MinSize` and `MaxSize` set the minimum and maximum number of EC2 instances in the Auto Scaling group.
- `TargetGroupARNs` takes the ARN of the target group with the logical name `EC2TargetGroup`. As this Auto Scaling group scales, it automatically registers and deregisters instances with this target group.

- `VPCZoneIdentifier` takes the value of the `Subnets` input parameter as the list of public subnets where the EC2 instances can be created.

Step 1: Launch the stack

Before you launch the stack, check that you have AWS Identity and Access Management (IAM) permissions to use all of the following services: Amazon EC2, Amazon EC2 Auto Scaling, AWS Systems Manager, Elastic Load Balancing, Amazon SNS, and CloudFormation.

The following procedure involves uploading the sample stack template from a file. Open a text editor on your local machine and add one of the templates. Save the file with the name `sampleloadbalancedappstack.template`.

To launch the stack template

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Choose **Create stack**, **With new resources (standard)**.
3. Under **Specify template**, choose **Upload a template file**, **Choose file** to upload the `sampleloadbalancedappstack.template` file.
4. Choose **Next**.
5. On the **Specify stack details** page, type the stack name (for example, **`SampleLoadBalancedAppStack`**).
6. Under **Parameters**, review the parameters for the stack and provide values for all parameters that don't have default values, including **OperatorEmail**, **SSHLocation**, **KeyName**, **VPC**, and **Subnets**.
7. Choose **Next** twice.
8. On the **Review** page, review and confirm the settings.
9. Choose **Submit**.

   You can view the status of the stack in the CloudFormation console in the **Status** column. When CloudFormation has successfully created the stack, you receive a status of **CREATE_COMPLETE**.

> **Note**
>
> After you create the stack, you must confirm the subscription before the email address can start to receive notifications. For more information, see Get Amazon SNS notifications when your Auto Scaling group scales in the *Amazon EC2 Auto Scaling User Guide*.

Step 2: Clean up your sample resources

To make sure that you aren't charged for unused sample resources, delete the stack.

To delete the stack

1. In the CloudFormation console, select the **SampleLoadBalancedAppStack** stack.
2. Choose **Delete**.
3. In the confirmation message, choose **Delete stack**.

   The status for **SampleLoadBalancedAppStack** changes to **DELETE_IN_PROGRESS**. When CloudFormation completes the deletion of the stack, it removes the stack from the list.

Use the sample template from this walkthrough to build your own stack templates. For more information, see Tutorial: Set up a scaled and load-balanced application in the *Amazon EC2 Auto Scaling User Guide*.

Peer with a VPC in another AWS account

You can peer with a Virtual Private Cloud (VPC) in another AWS account by using AWS::EC2::VPCPeeringConnection. This creates a networking connection between two VPCs that enables you to route traffic between them so they can communicate as if they were within the same network. A VPC peering connection can help facilitate data access and data transfer.

To establish a VPC peering connection, you need to authorize two separate AWS accounts within a single CloudFormation stack.

For more information about VPC peering and its limitations, see the Amazon VPC Peering Guide.

Prerequisites

1. You need a peer VPC ID, a peer AWS account ID, and a cross-account access role for the peering connection.

   > **Note**
   >
   > This walkthrough refers to two accounts: First is an account that allows cross-account peering (the *accepter account*). Second is an account that requests the peering connection (the *requester account*).

2. To accept the VPC peering connection, the cross-account access role must be assumable by you. The resource behaves the same way as a VPC peering connection resource in the same account. For information about how an IAM administrator grants permissions to assume the cross-account role, see Grant a user permissions to switch roles in the *IAM User Guide*.

Step 1: Create a VPC and a cross-account role

In this step, you'll create the VPC and role in the *accepter account*.

To create a VPC and a cross-account access role

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the **Stacks** page, choose **Create stack** at top right, and then choose **With new resources (standard)**.
3. For **Prerequisite - Prepare template**, choose **Choose an existing template** and then **Upload a template file**, **Choose file**.
4. Open a text editor on your local machine and add one of the following templates. Save the file and return to the console to select it as the template file.

   Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a VPC and an assumable role for cross account VPC
 peering.",
  "Parameters": {
```

```
    "PeerRequesterAccountId": {
      "Type": "String"
    }
  },
  "Resources": {
    "vpc": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "10.1.0.0/16",
        "EnableDnsSupport": false,
        "EnableDnsHostnames": false,
        "InstanceTenancy": "default"
      }
    },
    "peerRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Principal": {
                "AWS": {
                  "Ref": "PeerRequesterAccountId"
                }
              },
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow"
            }
          ]
        },
        "Path": "/",
        "Policies": [
          {
            "PolicyName": "root",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": "ec2:AcceptVpcPeeringConnection",
                  "Resource": "*"
                }
```

```
              ]
            }
          }
        ]
      }
    }
  },
  "Outputs": {
    "VPCId": {
      "Value": {
        "Ref": "vpc"
      }
    },
    "RoleARN": {
      "Value": {
        "Fn::GetAtt": [
          "peerRole",
          "Arn"
        ]
      }
    }
  }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and an assumable role for cross account VPC peering.
Parameters:
  PeerRequesterAccountId:
    Type: String
Resources:
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.1.0.0/16
      EnableDnsSupport: false
      EnableDnsHostnames: false
      InstanceTenancy: default
  peerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
        Statement:
          - Principal:
              AWS: !Ref PeerRequesterAccountId
            Action:
              - 'sts:AssumeRole'
            Effect: Allow
      Path: /
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action: 'ec2:AcceptVpcPeeringConnection'
                Resource: '*'
Outputs:
  VPCId:
    Value: !Ref vpc
  RoleARN:
    Value: !GetAtt
      - peerRole
      - Arn
```

5. Choose **Next**.
6. Give the stack a name (for example, **`VPC-owner`**), and then enter the AWS account ID of the *requester account* in the **PeerRequesterAccountId** field.
7. Accept the defaults, and then choose **Next**.
8. Choose **I acknowledge that CloudFormation might create IAM resources**, and then choose **Create stack**.

Step 2: Create a template that includes `AWS::EC2::VPCPeeringConnection`

Now that you've created the VPC and cross-account role, you can peer with the VPC using another AWS account (the *requester account*).

To create a template that includes the AWS::EC2::VPCPeeringConnection resource

1. Go back to the CloudFormation console home page.
2. From the **Stacks** page, choose **Create stack** at top right, and then choose **With new resources (standard)**.

3. For **Prerequisite - Prepare template**, choose **Choose an existing template** and then **Upload a template file**, **Choose file**.
4. Open a text editor on your local machine and add one of the following templates. Save the file and return to the console to select it as the template file.

Example JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a VPC and a VPC Peering connection using the PeerRole to
 accept.",
  "Parameters": {
    "PeerVPCAccountId": {
      "Type": "String"
    },
    "PeerVPCId": {
      "Type": "String"
    },
    "PeerRoleArn": {
      "Type": "String"
    }
  },
  "Resources": {
    "vpc": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": "10.2.0.0/16",
        "EnableDnsSupport": false,
        "EnableDnsHostnames": false,
        "InstanceTenancy": "default"
      }
    },
    "vpcPeeringConnection": {
      "Type": "AWS::EC2::VPCPeeringConnection",
      "Properties": {
        "VpcId": {
          "Ref": "vpc"
        },
        "PeerVpcId": {
          "Ref": "PeerVPCId"
        },
        "PeerOwnerId": {
          "Ref": "PeerVPCAccountId"
```

```
        },
        "PeerRoleArn": {
          "Ref": "PeerRoleArn"
        }
      }
    }
  },
  "Outputs": {
    "VPCId": {
      "Value": {
        "Ref": "vpc"
      }
    },
    "VPCPeeringConnectionId": {
      "Value": {
        "Ref": "vpcPeeringConnection"
      }
    }
  }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and a VPC Peering connection using the PeerRole to
 accept.
Parameters:
  PeerVPCAccountId:
    Type: String
  PeerVPCId:
    Type: String
  PeerRoleArn:
    Type: String
Resources:
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.2.0.0/16
      EnableDnsSupport: false
      EnableDnsHostnames: false
      InstanceTenancy: default
  vpcPeeringConnection:
    Type: AWS::EC2::VPCPeeringConnection
```

```
    Properties:
      VpcId: !Ref vpc
      PeerVpcId: !Ref PeerVPCId
      PeerOwnerId: !Ref PeerVPCAccountId
      PeerRoleArn: !Ref PeerRoleArn
Outputs:
  VPCId:
    Value: !Ref vpc
  VPCPeeringConnectionId:
    Value: !Ref vpcPeeringConnection
```

5. Choose **Next**.
6. Give the stack a name (for example, **`VPC-peering-connection`**).
7. Accept the defaults, and then choose **Next**.
8. Choose **I acknowledge that CloudFormation might create IAM resources**, and then choose **Create stack**.

Create a template with a highly restrictive policy

You might want to create a highly restrictive policy for peering your VPC with another AWS account.

The following example template shows how to change the VPC peer owner template (the *accepter account* created in Step 1 above) so that it's more restrictive.

Example JSON

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Description":"Create a VPC and an assumable role for cross account VPC peering.",
  "Parameters":{
    "PeerRequesterAccountId":{
      "Type":"String"
    }
  },
  "Resources":{
    "peerRole":{
      "Type":"AWS::IAM::Role",
      "Properties":{
        "AssumeRolePolicyDocument":{
          "Statement":[
```

```
            {
              "Action":[
                "sts:AssumeRole"
              ],
              "Effect":"Allow",
              "Principal":{
                "AWS":{
                  "Ref":"PeerRequesterAccountId"
                }
              }
            }
          ]
        },
        "Path":"/",
        "Policies":[
          {
            "PolicyDocument":{
              "Statement":[
                {
                  "Action":"ec2:acceptVpcPeeringConnection",
                  "Effect":"Allow",
                  "Resource":{
                    "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}"
                  }
                },
                {
                  "Action":"ec2:acceptVpcPeeringConnection",
                  "Condition":{
                    "StringEquals":{
                      "ec2:AccepterVpc":{
                        "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/
${vpc}"
                      }
                    }
                  },
                  "Effect":"Allow",
                  "Resource":{
                    "Fn::Sub":"arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc-
peering-connection/*"
                  }
                }
              ],
              "Version":"2012-10-17"
            },
```

```
            "PolicyName":"root"
          }
        ]
      }
    },
    "vpc":{
      "Type":"AWS::EC2::VPC",
      "Properties":{
        "CidrBlock":"10.1.0.0/16",
        "EnableDnsHostnames":false,
        "EnableDnsSupport":false,
        "InstanceTenancy":"default"
      }
    }
  },
  "Outputs":{
    "RoleARN":{
      "Value":{
        "Fn::GetAtt":[
          "peerRole",
          "Arn"
        ]
      }
    },
    "VPCId":{
      "Value":{
        "Ref":"vpc"
      }
    }
  }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create a VPC and an assumable role for cross account VPC peering.
Parameters:
  PeerRequesterAccountId:
    Type: String
Resources:
  peerRole:
    Type: AWS::IAM::Role
    Properties:
```

```
      AssumeRolePolicyDocument:
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              AWS:
                Ref: PeerRequesterAccountId
      Path: /
      Policies:
        - PolicyDocument:
            Statement:
              - Action: 'ec2:acceptVpcPeeringConnection'
                Effect: Allow
                Resource:
                  'Fn::Sub': 'arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/${vpc}'
              - Action: 'ec2:acceptVpcPeeringConnection'
                Condition:
                  StringEquals:
                    'ec2:AccepterVpc':
                      'Fn::Sub': 'arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc/
${vpc}'
                Effect: Allow
                Resource:
                  'Fn::Sub': >-
                    arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:vpc-peering-
connection/*
            Version: 2012-10-17
          PolicyName: root
  vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.1.0.0/16
      EnableDnsHostnames: false
      EnableDnsSupport: false
      InstanceTenancy: default
Outputs:
  RoleARN:
    Value:
      'Fn::GetAtt':
        - peerRole
        - Arn
  VPCId:
    Value:
```

```
      Ref: vpc
```

To access the VPC, you can use the same requester template as in Step 2 above.

For more information, see Identity and access management for VPC peering in the *Amazon VPC Peering Guide*.

Perform ECS blue/green deployments through CodeDeploy using CloudFormation

To update an application running on Amazon Elastic Container Service (Amazon ECS), you can use a CodeDeploy blue/green deployment strategy. This strategy helps minimize interruptions caused by changing application versions.

In a blue/green deployment, you create a new application environment (referred to as *green*) alongside your current, live environment (referred to as *blue*). This allows you to monitor and test the green environment before routing live traffic from the blue environment to the green environment. After the green environment is serving live traffic, you can safely terminate the blue environment.

To perform CodeDeploy blue/green deployments on ECS using CloudFormation, you include the following information in your stack template:

- A `Hooks` section that describes a `AWS::CodeDeploy::BlueGreen` hook.
- A `Transform` section that specifies the `AWS::CodeDeployBlueGreen` transform.

The following topics guide you through setting up a CloudFormation template for a blue/green deployment on ECS.

Topics

- About blue/green deployments
- Considerations when managing ECS blue/green deployments using CloudFormation
- AWS::CodeDeploy::BlueGreen hook syntax
- Blue/green deployment template example

About blue/green deployments

This topic provides an overview of how performing blue/green deployments with CloudFormation works. It also explains how to prepare your CloudFormation template for blue/green deployments.

Topics

- How it works
- Resource updates that initiate green deployments
- Preparing your template to perform ECS blue/green deployments
- Modeling your blue/green deployment using CloudFormation resources
- Change sets
- Monitoring stack events
- IAM permissions for blue/green deployments

How it works

When using CloudFormation to perform ECS blue/green deployments through CodeDeploy, you start by creating a stack template that defines the resources for both your blue and green application environments, including specifying the traffic routing and stabilization settings to use. Next, you create a stack from that template. This generates your blue (current) application. CloudFormation only creates the blue resources during stack creation. Resources for a green deployment aren't created until they're required.

Then, if in a future stack update you update the task definition or task set resources in your blue application, CloudFormation does the following:

- Generates all the necessary green application environment resources
- Shifts the traffic based on the specified traffic routing parameters
- Deletes the blue resources

If an error occurs at any point before the green deployment is successful and finalized, CloudFormation rolls the stack back to its state before the entire green deployment was initiated.

Resource updates that initiate green deployments

When you perform a stack update that updates certain properties of specific ECS resources, CloudFormation initiates a green deployment process. The resources that initiate this process are:

- AWS::ECS::TaskDefinition
- AWS::ECS::TaskSet

However, if the updates to these resources don't involve property changes that require replacement, a green deployment won't be initiated. For more information, see Understand update behaviors of stack resources.

It's important to note that you can't combine updates to the above resources with updates to other resources in the same stack update operation. If you need to update both the listed resources and other resources within the same stack, you have two options:

- Perform two separate stack update operations: one that includes only the updates to the above resources, and a separate stack update that includes changes to any other resources.
- Remove the `Transform` and `Hooks` sections from your template and then perform the stack update. In this case, CloudFormation won't perform a green deployment.

Preparing your template to perform ECS blue/green deployments

To enable blue/green deployments on your stack, include the following sections in your stack template before performing a stack update.

- Add a reference to the `AWS::CodeDeployBlueGreen` transform to your template:

```
"Transform": [
  "AWS::CodeDeployBlueGreen"
],
```

- Add a `Hooks` section that invokes the `AWS::CodeDeploy::BlueGreen` hook and specifies the properties for your deployment. For more information, see `AWS::CodeDeploy::BlueGreen` hook syntax.
- In the `Resources` section, define the blue and green resources for your deployment.

You can add these sections when you first create the template (that's, before creating the stack itself), or you can add them to an existing template before performing a stack update. If you specify the blue/green deployment for a new stack, CloudFormation only creates the blue resources during stack creation — resources for the green deployment aren't created until they're required during a stack update.

Modeling your blue/green deployment using CloudFormation resources

To perform CodeDeploy blue/green deployment on ECS, your CloudFormation template needs to include the resources that model your deployment, such as an Amazon ECS service and load balancer. For more details on what these resources represent, see Before you begin an Amazon ECS deployment in the *AWS CodeDeploy User Guide*.

Requirement	Resource	Required/Optional	Initiates blue/green deployment if replaced?
Amazon ECS cluster	AWS::ECS::Cluster	Optional. The default cluster can be used.	No
Amazon ECS service	AWS::ECS::Service	Required.	No
Application or Network Load Balancer	AWS::ECS::Service LoadBalancer	Required.	No
Production listener	AWS::ElasticLoadBalancingV2::Listener	Required.	No
Test listener	AWS::ElasticLoadBalancingV2::Listener	Optional.	No
Two target groups	AWS::ElasticLoadBalancingV2::TargetGroup	Required.	No
Amazon ECS task definition	AWS::ECS::TaskDefinition	Required.	Yes
Container for your Amazon ECS application	AWS::ECS::TaskDefinition Container Definition Name	Required.	No

Requirement	Resource	Required/Optional	Initiates blue/green deployment if replaced?
Port for your replacement task set	AWS::ECS::TaskDefinition PortMapping ContainerPort	Required.	No

Change sets

We strongly recommend that you create a change set before performing a stack update that will initiate a green deployment. This allows you to see the actual changes that will be made to your stack before performing stack update. Be aware that resource changes may not be listed in the order in which they will be performed during the stack update. For more information, see Update CloudFormation stacks using change sets.

Whether you use change sets or direct stack operations for your blue/green deployments, CloudFormation automatically runs pre-deployment validation to catch common template errors before any resources are provisioned. This includes property syntax validation and resource name conflict detection on all stack operations, plus additional checks (service quota limits, Recorder conflicts, and ECR repository delete readiness) available as warnings during change set creation.

Monitoring stack events

You can view the stack events generated at each step of the ECS deployment on the **Events** tab of the **Stack** page, and using the AWS CLI. For more information, see Monitor stack progress.

IAM permissions for blue/green deployments

In order for CloudFormation to successfully perform the blue-green deployments, you must have the following CodeDeploy permissions:

- `codedeploy:Get*`
- `codedeploy:CreateCloudFormationDeployment`

For more information, see Actions, resources, and condition keys for CodeDeploy in the *Service Authorization Reference*.

Considerations when managing ECS blue/green deployments using CloudFormation

The process of using CloudFormation to perform your ECS blue/green deployments through CodeDeploy is different from a standard ECS deployment using just CodeDeploy. For a detailed understanding of these differences, see Differences between Amazon ECS blue/green deployments through CodeDeploy and CloudFormation in the *AWS CodeDeploy User Guide*.

When managing your blue/green deployment using CloudFormation, there are certain limitations and considerations to keep in mind:

- Only updates to certain resources will initiate a green deployment. For more information, see Resource updates that initiate green deployments.
- You can't include updates to resources that initiate green deployments and updates to other resources in the same stack update. For more information, see Resource updates that initiate green deployments.
- You can only specify a single ECS service as the deployment target.
- Parameters whose values are obfuscated by CloudFormation can't be updated by CodeDeploy during a green deployment, and will lead to an error and stack update failure. These include:
  - Parameters defined with the `NoEcho` attribute.
  - Parameters that use dynamic references to retrieve their values from external services. For more information about dynamic references, see Get values stored in other services using dynamic references.
- To cancel a green deployment that's still in progress, cancel the stack update in CloudFormation, not CodeDeploy or ECS. For more information, see Cancel a stack update. After an update has finished, you can't cancel it. However, you can update a stack again with any previous settings.
- The following CloudFormation features are not currently supported for templates that define ECS blue/green deployments:
  - Declaring outputs or using Fn::ImportValue to import values from other stacks.
  - Importing resources. For more information about importing resources, see Import AWS resources into a CloudFormation stack.
  - Using the `AWS::CodeDeploy::BlueGreen` hook in a template that includes nested stack resources. For more information about nested stacks, see Split a template into reusable pieces using nested stacks.
  - Using the `AWS::CodeDeploy::BlueGreen` hook in a nested stack.

AWS::CodeDeploy::BlueGreen hook syntax

The following syntax describes the structure of an AWS::CodeDeploy::BlueGreen hook for ECS blue/green deployments.

Syntax

```
"Hooks": {
  "Logical ID": {
    "Type": "AWS::CodeDeploy::BlueGreen",
    "Properties": {
      "TrafficRoutingConfig": {
        "Type": "Traffic routing type",
        "TimeBasedCanary": {
          "StepPercentage": Integer,
          "BakeTimeMins": Integer
        },
        "TimeBasedLinear": {
          "StepPercentage": Integer,
          "BakeTimeMins": Integer
        }
      },
      "AdditionalOptions": {"TerminationWaitTimeInMinutes": Integer},
      "LifecycleEventHooks": {
        "BeforeInstall": "FunctionName",
        "AfterInstall": "FunctionName",
        "AfterAllowTestTraffic": "FunctionName",
        "BeforeAllowTraffic": "FunctionName",
        "AfterAllowTraffic": "FunctionName"
      },
      "ServiceRole": "CodeDeployServiceRoleName",
      "Applications": [
        {
          "Target": {
            "Type": "AWS::ECS::Service",
            "LogicalID": "Logical ID of AWS::ECS::Service"
          },
          "ECSAttributes": {
            "TaskDefinitions": [
              "Logical ID of AWS::ECS::TaskDefinition (Blue)",
              "Logical ID of AWS::ECS::TaskDefinition (Green)"
            ],
            "TaskSets": [
```

```
              "Logical ID of AWS::ECS::TaskSet (Blue)",
              "Logical ID of AWS::ECS::TaskSet (Green)"
            ],
            "TrafficRouting": {
              "ProdTrafficRoute": {
                "Type": "AWS::ElasticLoadBalancingV2::Listener",
                "LogicalID": "Logical ID of AWS::ElasticLoadBalancingV2::Listener
(Production)"
              },
              "TestTrafficRoute": {
                "Type": "AWS::ElasticLoadBalancingV2::Listener",
                "LogicalID": "Logical ID of AWS::ElasticLoadBalancingV2::Listener
(Test)"
              },
              "TargetGroups": [
                "Logical ID of AWS::ElasticLoadBalancingV2::TargetGroup (Blue)",
                "Logical ID of AWS::ElasticLoadBalancingV2::TargetGroup (Green)"
              ]
            }
          }
        }
      ]
    }
  }
}
```

Properties

Logical ID (also called *logical name*)

The logical ID of a hook declared in the `Hooks` section of the template. The logical ID must be alphanumeric (A-Za-z0-9) and unique within the template.

Required: Yes

`Type`

The type of hook. `AWS::CodeDeploy::BlueGreen`

Required: Yes

`Properties`

Properties of the hook.

Required: Yes

`TrafficRoutingConfig`

Traffic routing configuration settings.

Required: No

The default configuration is time-based canary traffic shifting, with a 15% step percentage and a five minute bake time.

`Type`

The type of traffic shifting used by the deployment configuration.

Valid values: AllAtOnce | TimeBasedCanary | TimeBasedLinear

Required: Yes

`TimeBasedCanary`

Specifies a configuration that shifts traffic from one version of the deployment to another in two increments.

Required: Conditional: If you specify `TimeBasedCanary` as the traffic routing type, you must include the `TimeBasedCanary` parameter.

`StepPercentage`

The percentage of traffic to shift in the first increment of a `TimeBasedCanary` deployment. The step percentage must be 14% or greater.

Required: No

`BakeTimeMins`

The number of minutes between the first and second traffic shifts of a `TimeBasedCanary` deployment.

Required: No

`TimeBasedLinear`

Specifies a configuration that shifts traffic from one version of the deployment to another in equal increments, with an equal number of minutes between each increment.

Required: Conditional: If you specify `TimeBasedLinear` as the traffic routing type, you must include the `TimeBasedLinear` parameter.

`StepPercentage`

The percentage of traffic that's shifted at the start of each increment of a `TimeBasedLinear` deployment. The step percentage must be 14% or greater.

Required: No

`BakeTimeMins`

The number of minutes between each incremental traffic shift of a `TimeBasedLinear` deployment.

Required: No

`AdditionalOptions`

Additional options for the blue/green deployment.

Required: No

`TerminationWaitTimeInMinutes`

Specifies time to wait, in minutes, before terminating the blue resources.

Required: No

`LifecycleEventHooks`

Use lifecycle event hooks to specify a Lambda function that CodeDeploy can call to validate a deployment. You can use the same function or a different one for deployment lifecyle events. Following completion of the validation tests, the Lambda `AfterAllowTraffic` function calls back CodeDeploy and delivers a result of `Succeeded` or `Failed`. For more information, see AppSpec 'hooks' section in the *AWS CodeDeploy User Guide.*

Required: No

`BeforeInstall`

Function to use to run tasks before the replacement task set is created.

Required: No

`AfterInstall`

Function to use to run tasks after the replacement task set is created and one of the target groups is associated with it.

Required: No

`AfterAllowTestTraffic`

Function to use to run tasks after the test listener serves traffic to the replacement task set.

Required: No

`BeforeAllowTraffic`

Function to use to run tasks after the second target group is associated with the replacement task set, but before traffic is shifted to the replacement task set.

Required: No

`AfterAllowTraffic`

Function to use to run tasks after the second target group serves traffic to the replacement task set.

Required: No

`ServiceRole`

The execution role for CloudFormation to use to perform the blue-green deployments. For a list of the necessary permissions, see IAM permissions for blue/green deployments.

Required: No

`Applications`

Specifies properties of the Amazon ECS application.

Required: Yes

`Target`

Required: Yes

`Type`

The type of the resource.

Required: Yes

`LogicalID`

The logical id of the resource.

Required: Yes

`ECSAttributes`

The resources that represent the various requirements of your Amazon ECS application deployment.

Required: Yes

`TaskDefinitions`

The logical ID of the AWS::ECS::TaskDefinition resource to run the Docker container that contains your Amazon ECS application.

Required: Yes

`TaskSets`

The logical IDs of the AWS::ECS::TaskSet resources to use as task sets for the application.

Required: Yes

`TrafficRouting`

Specifies resources used for traffic routing.

Required: Yes

`ProdTrafficRoute`

The listener to be used by your load balancer to direct traffic to your target groups.

Required: Yes

`Type`

The type of the resource. `AWS::ElasticLoadBalancingV2::Listener`

Required: Yes

`LogicalID`

The logical ID of the resource.

Required: Yes

`TestTrafficRoute`

The listener to be used by your load balancer to direct traffic to your target groups.

Required: Yes

`Type`

The type of the resource. `AWS::ElasticLoadBalancingV2::Listener`

Required: Yes

`LogicalID`

The logical ID of the resource.

Required: No

`TargetGroups`

Logical ID of resources to use as target groups to route traffic to the registered target.

Required: Yes

Blue/green deployment template example

The following example template sets up a CodeDeploy blue/green deployment on ECS, with a traffic routing progress of 15 percent per step and a stabilization period of 5 minutes between each step.

Creating a stack with the template will provision the initial configuration of the deployment. If you then made any changes to properties in the `BlueTaskSet` resource that require that resource be replaced, CloudFormation will then initiate a green deployment as part of the stack update.

JSON

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Parameters":{
    "Vpc":{ "Type":"AWS::EC2::VPC::Id" },
    "Subnet1":{ "Type":"AWS::EC2::Subnet::Id" },
    "Subnet2":{ "Type":"AWS::EC2::Subnet::Id" }
  },
  "Transform":[ "AWS::CodeDeployBlueGreen" ],
  "Hooks":{
    "CodeDeployBlueGreenHook":{
      "Type":"AWS::CodeDeploy::BlueGreen",
      "Properties":{
        "TrafficRoutingConfig":{
          "Type":"TimeBasedCanary",
          "TimeBasedCanary":{
            "StepPercentage":15,
            "BakeTimeMins":5
          }
        },
        "Applications":[
          {
            "Target":{
              "Type":"AWS::ECS::Service",
              "LogicalID":"ECSDemoService"
            },
            "ECSAttributes":{
              "TaskDefinitions":[ "BlueTaskDefinition","GreenTaskDefinition" ],
              "TaskSets":[ "BlueTaskSet","GreenTaskSet" ],
              "TrafficRouting":{
                "ProdTrafficRoute":{
                  "Type":"AWS::ElasticLoadBalancingV2::Listener",
                  "LogicalID":"ALBListenerProdTraffic"
                },
                "TargetGroups":[ "ALBTargetGroupBlue","ALBTargetGroupGreen" ]
              }
            }
          }
        ]
      }
    }
  },
  "Resources":{
```

```
    "ExampleSecurityGroup":{
      "Type":"AWS::EC2::SecurityGroup",
      "Properties":{
        "GroupDescription":"Security group for ec2 access",
        "VpcId":{ "Ref":"Vpc" },
        "SecurityGroupIngress":[
          {
            "IpProtocol":"tcp",
            "FromPort":80,
            "ToPort":80,
            "CidrIp":"0.0.0.0/0"
          },
          {
            "IpProtocol":"tcp",
            "FromPort":8080,
            "ToPort":8080,
            "CidrIp":"0.0.0.0/0"
          },
          {
            "IpProtocol":"tcp",
            "FromPort":22,
            "ToPort":22,
            "CidrIp":"0.0.0.0/0"
          }
        ]
      }
    },
    "ALBTargetGroupBlue":{
      "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties":{
        "HealthCheckIntervalSeconds":5,
        "HealthCheckPath":"/",
        "HealthCheckPort":"80",
        "HealthCheckProtocol":"HTTP",
        "HealthCheckTimeoutSeconds":2,
        "HealthyThresholdCount":2,
        "Matcher":{ "HttpCode":"200" },
        "Port":80,
        "Protocol":"HTTP",
        "Tags":[{ "Key":"Group","Value":"Example" }],
        "TargetType":"ip",
        "UnhealthyThresholdCount":4,
        "VpcId":{ "Ref":"Vpc" }
      }
```

```
    },
    "ALBTargetGroupGreen":{
      "Type":"AWS::ElasticLoadBalancingV2::TargetGroup",
      "Properties":{
        "HealthCheckIntervalSeconds":5,
        "HealthCheckPath":"/",
        "HealthCheckPort":"80",
        "HealthCheckProtocol":"HTTP",
        "HealthCheckTimeoutSeconds":2,
        "HealthyThresholdCount":2,
        "Matcher":{ "HttpCode":"200" },
        "Port":80,
        "Protocol":"HTTP",
        "Tags":[{ "Key":"Group","Value":"Example" }],
        "TargetType":"ip",
        "UnhealthyThresholdCount":4,
        "VpcId":{ "Ref":"Vpc" }
      }
    },
    "ExampleALB":{
      "Type":"AWS::ElasticLoadBalancingV2::LoadBalancer",
      "Properties":{
        "Scheme":"internet-facing",
        "SecurityGroups":[{ "Ref":"ExampleSecurityGroup" }],
        "Subnets":[{ "Ref":"Subnet1" },{ "Ref":"Subnet2" }],
        "Tags":[{ "Key":"Group","Value":"Example" }],
        "Type":"application",
        "IpAddressType":"ipv4"
      }
    },
    "ALBListenerProdTraffic":{
      "Type":"AWS::ElasticLoadBalancingV2::Listener",
      "Properties":{
        "DefaultActions":[
          {
            "Type":"forward",
            "ForwardConfig":{
              "TargetGroups":[
                {
                  "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" },
                  "Weight":1
                }
              ]
            }
```

```
          }
        ],
        "LoadBalancerArn":{ "Ref":"ExampleALB" },
        "Port":80,
        "Protocol":"HTTP"
      }
    },
    "ALBListenerProdRule":{
      "Type":"AWS::ElasticLoadBalancingV2::ListenerRule",
      "Properties":{
        "Actions":[
          {
            "Type":"forward",
            "ForwardConfig":{
              "TargetGroups":[
                {
                  "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" },
                  "Weight":1
                }
              ]
            }
          }
        ],
        "Conditions":[
          {
            "Field":"http-header",
            "HttpHeaderConfig":{
              "HttpHeaderName":"User-Agent",
              "Values":[ "Mozilla" ]
            }
          }
        ],
        "ListenerArn":{ "Ref":"ALBListenerProdTraffic" },
        "Priority":1
      }
    },
    "ECSTaskExecutionRole":{
      "Type":"AWS::IAM::Role",
      "Properties":{
        "AssumeRolePolicyDocument":{
          "Version": "2012-10-17",
          "Statement":[
            {
              "Sid":"",
```

```
              "Effect":"Allow",
              "Principal":{
                "Service":"ecs-tasks.amazonaws.com"
              },
              "Action":"sts:AssumeRole"
            }
          ]
        },
        "ManagedPolicyArns":[ "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" ]
      }
    },
    "BlueTaskDefinition":{
      "Type":"AWS::ECS::TaskDefinition",
      "Properties":{
        "ExecutionRoleArn":{
          "Fn::GetAtt":[ "ECSTaskExecutionRole","Arn" ]
        },
        "ContainerDefinitions":[
          {
            "Name":"DemoApp",
            "Image":"nginxdemos/hello:latest",
            "Essential":true,
            "PortMappings":[
              {
                "HostPort":80,
                "Protocol":"tcp",
                "ContainerPort":80
              }
            ]
          }
        ],
        "RequiresCompatibilities":[ "FARGATE" ],
        "NetworkMode":"awsvpc",
        "Cpu":"256",
        "Memory":"512",
        "Family":"ecs-demo"
      }
    },
    "ECSDemoCluster":{
      "Type":"AWS::ECS::Cluster",
      "Properties":{}
    },
    "ECSDemoService":{
```

```
      "Type":"AWS::ECS::Service",
      "Properties":{
        "Cluster":{ "Ref":"ECSDemoCluster" },
        "DesiredCount":1,
        "DeploymentController":{ "Type":"EXTERNAL" }
      }
    },
    "BlueTaskSet":{
      "Type":"AWS::ECS::TaskSet",
      "Properties":{
        "Cluster":{ "Ref":"ECSDemoCluster" },
        "LaunchType":"FARGATE",
        "NetworkConfiguration":{
          "AwsVpcConfiguration":{
            "AssignPublicIp":"ENABLED",
            "SecurityGroups":[{ "Ref":"ExampleSecurityGroup" }],
            "Subnets":[{ "Ref":"Subnet1" },{ "Ref":"Subnet2" }]
          }
        },
        "PlatformVersion":"1.4.0",
        "Scale":{
          "Unit":"PERCENT",
          "Value":100
        },
        "Service":{ "Ref":"ECSDemoService"},
        "TaskDefinition":{ "Ref":"BlueTaskDefinition" },
        "LoadBalancers":[
          {
            "ContainerName":"DemoApp",
            "ContainerPort":80,
            "TargetGroupArn":{ "Ref":"ALBTargetGroupBlue" }
          }
        ]
      }
    },
    "PrimaryTaskSet":{
      "Type":"AWS::ECS::PrimaryTaskSet",
      "Properties":{
        "Cluster":{ "Ref":"ECSDemoCluster" },
        "Service":{ "Ref":"ECSDemoService" },
        "TaskSetId":{ "Fn::GetAtt":[ "BlueTaskSet","Id" ]
        }
      }
    }
```

```
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  Vpc:
    Type: AWS::EC2::VPC::Id
  Subnet1:
    Type: AWS::EC2::Subnet::Id
  Subnet2:
    Type: AWS::EC2::Subnet::Id
Transform:
  - 'AWS::CodeDeployBlueGreen'
Hooks:
  CodeDeployBlueGreenHook:
    Type: AWS::CodeDeploy::BlueGreen
    Properties:
      TrafficRoutingConfig:
        Type: TimeBasedCanary
        TimeBasedCanary:
          StepPercentage: 15
          BakeTimeMins: 5
      Applications:
        - Target:
            Type: AWS::ECS::Service
            LogicalID: ECSDemoService
          ECSAttributes:
            TaskDefinitions:
              - BlueTaskDefinition
              - GreenTaskDefinition
            TaskSets:
              - BlueTaskSet
              - GreenTaskSet
            TrafficRouting:
              ProdTrafficRoute:
                Type: AWS::ElasticLoadBalancingV2::Listener
                LogicalID: ALBListenerProdTraffic
              TargetGroups:
                - ALBTargetGroupBlue
                - ALBTargetGroupGreen
Resources:
```

```
  ExampleSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for ec2 access
      VpcId: !Ref Vpc
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 8080
          ToPort: 8080
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
  ALBTargetGroupBlue:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 5
      HealthCheckPath: /
      HealthCheckPort: '80'
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 2
      HealthyThresholdCount: 2
      Matcher:
        HttpCode: '200'
      Port: 80
      Protocol: HTTP
      Tags:
        - Key: Group
          Value: Example
      TargetType: ip
      UnhealthyThresholdCount: 4
      VpcId: !Ref Vpc
  ALBTargetGroupGreen:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 5
      HealthCheckPath: /
      HealthCheckPort: '80'
      HealthCheckProtocol: HTTP
```

```
      HealthCheckTimeoutSeconds: 2
      HealthyThresholdCount: 2
      Matcher:
        HttpCode: '200'
      Port: 80
      Protocol: HTTP
      Tags:
        - Key: Group
          Value: Example
      TargetType: ip
      UnhealthyThresholdCount: 4
      VpcId: !Ref Vpc
  ExampleALB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      SecurityGroups:
        - !Ref ExampleSecurityGroup
      Subnets:
        - !Ref Subnet1
        - !Ref Subnet2
      Tags:
        - Key: Group
          Value: Example
      Type: application
      IpAddressType: ipv4
  ALBListenerProdTraffic:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          ForwardConfig:
            TargetGroups:
              - TargetGroupArn: !Ref ALBTargetGroupBlue
                Weight: 1
      LoadBalancerArn: !Ref ExampleALB
      Port: 80
      Protocol: HTTP
  ALBListenerProdRule:
    Type: AWS::ElasticLoadBalancingV2::ListenerRule
    Properties:
      Actions:
        - Type: forward
          ForwardConfig:
```

```
            TargetGroups:
              - TargetGroupArn: !Ref ALBTargetGroupBlue
                Weight: 1
      Conditions:
        - Field: http-header
          HttpHeaderConfig:
            HttpHeaderName: User-Agent
            Values:
              - Mozilla
      ListenerArn: !Ref ALBListenerProdTraffic
      Priority: 1
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Sid: ''
            Effect: Allow
            Principal:
              Service: ecs-tasks.amazonaws.com
            Action: 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'
  BlueTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ExecutionRoleArn: !GetAtt
        - ECSTaskExecutionRole
        - Arn
      ContainerDefinitions:
        - Name: DemoApp
          Image: 'nginxdemos/hello:latest'
          Essential: true
          PortMappings:
            - HostPort: 80
              Protocol: tcp
              ContainerPort: 80
      RequiresCompatibilities:
        - FARGATE
      NetworkMode: awsvpc
      Cpu: '256'
      Memory: '512'
      Family: ecs-demo
```

```
  ECSDemoCluster:
    Type: AWS::ECS::Cluster
    Properties: {}
  ECSDemoService:
    Type: AWS::ECS::Service
    Properties:
      Cluster: !Ref ECSDemoCluster
      DesiredCount: 1
      DeploymentController:
        Type: EXTERNAL
  BlueTaskSet:
    Type: AWS::ECS::TaskSet
    Properties:
      Cluster: !Ref ECSDemoCluster
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsVpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - !Ref ExampleSecurityGroup
          Subnets:
            - !Ref Subnet1
            - !Ref Subnet2
      PlatformVersion: 1.4.0
      Scale:
        Unit: PERCENT
        Value: 100
      Service: !Ref ECSDemoService
      TaskDefinition: !Ref BlueTaskDefinition
      LoadBalancers:
        - ContainerName: DemoApp
          ContainerPort: 80
          TargetGroupArn: !Ref ALBTargetGroupBlue
  PrimaryTaskSet:
    Type: AWS::ECS::PrimaryTaskSet
    Properties:
      Cluster: !Ref ECSDemoCluster
      Service: !Ref ECSDemoService
      TaskSetId: !GetAtt
        - BlueTaskSet
        - Id
```

CloudFormation template snippets

This section provides a number of example scenarios that you can use to understand how to declare various CloudFormation template parts. You can also use the snippets as a starting point for sections of your custom templates.

> **Note**
>
> Because CloudFormation templates must be JSON compliant, there is no provision for a line continuation character. The wrapping of the snippets in this document may be random if the line is longer than 68 characters.

Topics

- General template snippets
- Auto scaling CloudFormation template snippets
- AWS Billing Console template snippets
- CloudFormation template snippets
- Amazon CloudFront template snippets
- Amazon CloudWatch template snippets
- Amazon CloudWatch Logs template snippets
- Amazon DynamoDB template snippets
- Amazon EC2 CloudFormation template snippets
- Amazon Elastic Container Service sample templates
- Amazon Elastic File System Sample Template
- Elastic Beanstalk template snippets
- Elastic Load Balancing template snippets
- AWS Identity and Access Management template snippets
- AWS Lambda template
- Amazon Redshift template snippets
- Amazon RDS template snippets
- Route 53 template snippets
- Amazon S3 template snippets

- Amazon SNS template snippets
- Amazon SQS template snippets
- Amazon Timestream template snippets

General template snippets

The following examples show different CloudFormation template features that aren't specific to an AWS service.

Topics

- Base64 encoded UserData property
- Base64 encoded UserData property with AccessKey and SecretKey
- Parameters section with one literal string parameter
- Parameters section with string parameter with regular expression constraint
- Parameters section with number parameter with MinValue and MaxValue constraints
- Parameters section with number parameter with AllowedValues constraint
- Parameters section with one literal CommaDelimitedList parameter
- Parameters section with parameter value based on pseudo parameter
- Mapping section with three mappings
- Description based on literal string
- Outputs section with one literal string output
- Outputs section with one resource reference and one pseudo reference output
- Outputs section with an output based on a function, a literal string, a reference, and a pseudo parameter
- Template format version
- AWS Tags property

Base64 encoded UserData property

This example shows the assembly of a `UserData` property using the `Fn::Base64` and `Fn::Join` functions. The references `MyValue` and `MyName` are parameters that must be defined in the `Parameters` section of the template. The literal string `Hello World` is just another value this example passes in as part of the `UserData`.

JSON

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ ",", [
            { "Ref" : "MyValue" },
            { "Ref" : "MyName" },
            "Hello World" ] ]
    }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     Ref: MyValue
     Ref: MyName
     Hello World
```

Base64 encoded UserData property with AccessKey and SecretKey

This example shows the assembly of a `UserData` property using the `Fn::Base64` and `Fn::Join` functions. It includes the `AccessKey` and `SecretKey` information. The references `AccessKey` and `SecretKey` are parameters that must be defined in the Parameters section of the template.

JSON

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ "", [
            "ACCESS_KEY=", { "Ref" : "AccessKey" },
            "SECRET_KEY=", { "Ref" : "SecretKey" } ]
        ]
    }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     ACCESS_KEY=${AccessKey}
```

```
     SECRET_KEY=${SecretKey}
```

Parameters section with one literal string parameter

The following example depicts a valid Parameters section declaration in which a single `String` type parameter is declared.

JSON

```
"Parameters" : {
    "UserName" : {
        "Type" : "String",
        "Default" : "nonadmin",
        "Description" : "Assume a vanilla user if no command-line spec provided"
    }
}
```

YAML

```
Parameters:
  UserName:
    Type: String
    Default: nonadmin
    Description: Assume a vanilla user if no command-line spec provided
```

Parameters section with string parameter with regular expression constraint

The following example depicts a valid Parameters section declaration in which a single `String` type parameter is declared. The `AdminUserAccount` parameter has a default of `admin`. The parameter value must have a minimum length of 1, a maximum length of 16, and contains alphabetical characters and numbers but must begin with an alphabetical character.

JSON

```
"Parameters" : {
    "AdminUserAccount": {
      "Default": "admin",
      "NoEcho": "true",
      "Description" : "The admin account user name",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "16",
```

```
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
    }
}
```

YAML

```
Parameters:
  AdminUserAccount:
    Default: admin
    NoEcho: true
    Description: The admin account user name
    Type: String
    MinLength: 1
    MaxLength: 16
    AllowedPattern: '[a-zA-Z][a-zA-Z0-9]*'
```

Parameters section with number parameter with MinValue and MaxValue constraints

The following example depicts a valid Parameters section declaration in which a single `Number` type parameter is declared. The `WebServerPort` parameter has a default of 80 and a minimum value 1 and maximum value 65535.

JSON

```
"Parameters" : {
    "WebServerPort": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "MinValue": "1",
      "MaxValue": "65535"
    }
}
```

YAML

```
Parameters:
  WebServerPort:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
```

```
    MinValue: 1
    MaxValue: 65535
```

Parameters section with number parameter with AllowedValues constraint

The following example depicts a valid Parameters section declaration in which a single `Number` type parameter is declared. The `WebServerPort` parameter has a default of 80 and allows only values of 80 and 8888.

JSON

```
"Parameters" : {
    "WebServerPortLimited": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "AllowedValues" : ["80", "8888"]
    }
}
```

YAML

```
Parameters:
  WebServerPortLimited:
    Default: 80
    Description: TCP/IP port for the web server
    Type: Number
    AllowedValues:
    - 80
    - 8888
```

Parameters section with one literal CommaDelimitedList parameter

The following example depicts a valid `Parameters` section declaration in which a single `CommaDelimitedList` type parameter is declared. The `NoEcho` property is set to `TRUE`, which will mask its value with asterisks (*****) in the **describe-stacks** output, except for information stored in the locations specified below.

> **Important**
>
> Using the `NoEcho` attribute does not mask any information stored in the following:

- The `Metadata` template section. CloudFormation does not transform, modify, or redact any information you include in the `Metadata` section. For more information, see Metadata.
- The `Outputs` template section. For more information, see Outputs.
- The `Metadata` attribute of a resource definition. For more information, see `Metadata` attribute.

We strongly recommend you do not use these mechanisms to include sensitive information, such as passwords or secrets.

Important

Rather than embedding sensitive information directly in your CloudFormation templates, we recommend you use dynamic parameters in the stack template to reference sensitive information that is stored and managed outside of CloudFormation, such as in the AWS Systems Manager Parameter Store or AWS Secrets Manager.
For more information, see the Do not embed credentials in your templates best practice.

JSON

```
"Parameters" : {
    "UserRoles" : {
        "Type" : "CommaDelimitedList",
        "Default" : "guest,newhire",
        "NoEcho" : "TRUE"
    }
}
```

YAML

```
Parameters:
  UserRoles:
    Type: CommaDelimitedList
    Default: "guest,newhire"
    NoEcho: true
```

Parameters section with parameter value based on pseudo parameter

The following example shows commands in the EC2 user data that use the pseudo parameters `AWS::StackName` and `AWS::Region`. For more information about pseudo parameters, see Get AWS values using pseudo parameters.

JSON

```
         "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
            "#!/bin/bash -xe\n",
            "yum install -y aws-cfn-bootstrap\n",

            "/opt/aws/bin/cfn-init -v ",
            "         --stack ", { "Ref" : "AWS::StackName" },
            "         --resource LaunchConfig ",
            "         --region ", { "Ref" : "AWS::Region" }, "\n",

            "/opt/aws/bin/cfn-signal -e $? ",
            "         --stack ", { "Ref" : "AWS::StackName" },
            "         --resource WebServerGroup ",
            "         --region ", { "Ref" : "AWS::Region" }, "\n"
        ]]}}
      }
```

YAML

```
UserData:
  Fn::Base64: !Sub |
     #!/bin/bash -xe
     yum update -y aws-cfn-bootstrap
     /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource LaunchConfig --
region ${AWS::Region}
     /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource WebServerGroup
 --region ${AWS::Region}
```

Mapping section with three mappings

The following example depicts a valid `Mapping` section declaration that contains three mappings. The map, when matched with a mapping key of `Stop`, `SlowDown`, or `Go`, provides the RGB values assigned to the corresponding `RGBColor` attribute.

JSON

```
"Mappings" : {
    "LightColor" : {
        "Stop" : {
            "Description" : "red",
            "RGBColor" : "RED 255 GREEN 0 BLUE 0"
        },
        "SlowDown" : {
            "Description" : "yellow",
            "RGBColor" : "RED 255 GREEN 255 BLUE 0"
        },
        "Go" : {
            "Description" : "green",
            "RGBColor" : "RED 0 GREEN 128 BLUE 0"
        }
    }
}
```

YAML

```
Mappings:
  LightColor:
    Stop:
      Description: red
      RGBColor: "RED 255 GREEN 0 BLUE 0"
    SlowDown:
      Description: yellow
      RGBColor: "RED 255 GREEN 255 BLUE 0"
    Go:
      Description: green
      RGBColor: "RED 0 GREEN 128 BLUE 0"
```

Description based on literal string

The following example depicts a valid `Description` section declaration where the value is based on a literal string. This snippet can be for templates, parameters, resources, properties, or outputs.

JSON

```
"Description" : "Replace this value"
```

YAML

```
Description: "Replace this value"
```

Outputs section with one literal string output

This example shows a output assignment based on a literal string.

JSON

```
"Outputs" : {
    "MyPhone" : {
        "Value" : "Please call 555-5555",
        "Description" : "A random message for aws cloudformation describe-stacks"
    }
}
```

YAML

```
Outputs:
  MyPhone:
    Value: Please call 555-5555
    Description: A random message for aws cloudformation describe-stacks
```

Outputs section with one resource reference and one pseudo reference output

This example shows an `Outputs` section with two output assignments. One is based on a resource, and the other is based on a pseudo reference.

JSON

```

"Outputs" : {
   "SNSTopic" : { "Value" : { "Ref" : "MyNotificationTopic" } },
   "StackName" : { "Value" : { "Ref" : "AWS::StackName" } }
}
```

YAML

```
Outputs:
  SNSTopic:
    Value: !Ref MyNotificationTopic
```

```
  StackName:
    Value: !Ref AWS::StackName
```

Outputs section with an output based on a function, a literal string, a reference, and a pseudo parameter

This example shows an Outputs section with one output assignment. The Join function is used to concatenate the value, using a percent sign as the delimiter.

JSON

```
"Outputs" : {
    "MyOutput" : {
        "Value" : { "Fn::Join" :
            [ "%", [ "A-string", {"Ref" : "AWS::StackName" } ] ]
        }
    }
}
```

YAML

```
Outputs:
  MyOutput:
    Value: !Join [ %, [ 'A-string', !Ref 'AWS::StackName' ]]
```

Template format version

The following snippet depicts a valid `AWSTemplateFormatVersion` section declaration.

JSON

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
```

AWS Tags property

This example shows an AWS `Tags` property. You would specify this property within the Properties section of a resource. When the resource is created, it will be tagged with the tags you declare.

JSON

```
"Tags" : [
      {
        "Key" : "keyname1",
        "Value" : "value1"
      },
      {
        "Key" : "keyname2",
        "Value" : "value2"
      }
    ]
```

YAML

```
Tags:
  -
    Key: "keyname1"
    Value: "value1"
  -
    Key: "keyname2"
    Value: "value2"
```

Auto scaling CloudFormation template snippets

With Amazon EC2 Auto Scaling, you can automatically scale Amazon EC2 instances, either with scaling policies or with scheduled scaling. Auto Scaling groups are collections of Amazon EC2 instances that enable automatic scaling and fleet management features, such as scaling policies, scheduled actions, health checks, lifecycle hooks, and load balancing.

Application Auto Scaling provides automatic scaling of different resources beyond Amazon EC2, either with scaling policies or with scheduled scaling.

You can create and configure Auto Scaling groups, scaling policies, scheduled actions, and other auto scaling resources as part of your infrastructure using CloudFormation templates. Templates make it easy to manage and automate the deployment of auto scaling resources in a repeatable and consistent manner.

The following example template snippets describe CloudFormation resources or components for Amazon EC2 Auto Scaling and Application Auto Scaling. These snippets are designed to be integrated into a template and are not intended to be run independently.

Snippet categories

- Configure Amazon EC2 Auto Scaling resources with CloudFormation
- Configure Application Auto Scaling resources with CloudFormation

Configure Amazon EC2 Auto Scaling resources with CloudFormation

The following examples show different snippets to include in templates for use with Amazon EC2 Auto Scaling.

Snippet categories

- Create a single instance Auto Scaling group
- Create an Auto Scaling group with an attached load balancer
- Create an Auto Scaling group with notifications
- Create an Auto Scaling group that uses a CreationPolicy and an UpdatePolicy
- Create a step scaling policy
- Mixed instances group examples
- Launch configuration examples

Create a single instance Auto Scaling group

This example shows an `AWS::AutoScaling::AutoScalingGroup` resource with a single instance to help you get started. The `VPCZoneIdentifier` property of the Auto Scaling group specifies a list of existing subnets in three different Availability Zones. You must specify the applicable subnet IDs from your account before you create your stack. The `LaunchTemplate` property references an `AWS::EC2::LaunchTemplate` resource with the logical name `myLaunchTemplate` that is defined elsewhere in your template.

> **Note**
>
> For examples of launch templates, see Create launch templates with CloudFormation in the Amazon EC2 snippets section and the Examples section in the `AWS::EC2::LaunchTemplate` resource.

JSON

```
"myASG" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
      "LaunchTemplate" : {
        "LaunchTemplateId" : {
          "Ref" : "myLaunchTemplate"
        },
        "Version" : {
          "Fn::GetAtt" : [
            "myLaunchTemplate",
            "LatestVersionNumber"
          ]
        }
      },
      "MaxSize" : "1",
      "MinSize" : "1"
   }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MaxSize: '1'
    MinSize: '1'
```

Create an Auto Scaling group with an attached load balancer

This example shows an AWS::AutoScaling::AutoScalingGroup resource for load balancing over multiple servers. It specifies the logical names of AWS resources declared elsewhere in the same template.

1. The VPCZoneIdentifier property specifies the logical names of two AWS::EC2::Subnet resources where the Auto Scaling group's EC2 instances will be created: myPublicSubnet1 and myPublicSubnet2.
2. The LaunchTemplate property specifies an AWS::EC2::LaunchTemplate resource with the logical name myLaunchTemplate.
3. The TargetGroupARNs property lists the target groups for an Application Load Balancer or Network Load Balancer used to route traffic to the Auto Scaling group. In this example, one target group is specified, an AWS::ElasticLoadBalancingV2::TargetGroup resource with the logical name myTargetGroup.

JSON

```
"myServerGroup" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "VPCZoneIdentifier" : [ { "Ref" : "myPublicSubnet1" }, { "Ref" :
 "myPublicSubnet2" } ],
      "LaunchTemplate" : {
        "LaunchTemplateId" : {
          "Ref" : "myLaunchTemplate"
        },
        "Version" : {
          "Fn::GetAtt" : [
            "myLaunchTemplate",
            "LatestVersionNumber"
          ]
        }
      },
      "MaxSize" : "5",
      "MinSize" : "1",
      "TargetGroupARNs" : [ { "Ref" : "myTargetGroup" } ]
   }
}
```

YAML

```
myServerGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - !Ref myPublicSubnet1
      - !Ref myPublicSubnet2
    LaunchTemplate:
      LaunchTemplateId: !Ref myLaunchTemplate
      Version: !GetAtt myLaunchTemplate.LatestVersionNumber
    MaxSize: '5'
    MinSize: '1'
    TargetGroupARNs:
      - !Ref myTargetGroup
```

See also

For a detailed example that creates an Auto Scaling group with a target tracking scaling policy based on the `ALBRequestCountPerTarget` predefined metric for your Application Load Balancer, see the Examples section in the `AWS::AutoScaling::ScalingPolicy` resource.

Create an Auto Scaling group with notifications

This example shows an `AWS::AutoScaling::AutoScalingGroup` resource that sends Amazon SNS notifications when the specified events take place. The `NotificationConfigurations` property specifies the SNS topic where CloudFormation sends a notification and the events that will cause CloudFormation to send notifications. When the events specified by `NotificationTypes` occur, CloudFormation will send a notification to the SNS topic specified by `TopicARN`. When you launch the stack, CloudFormation creates an `AWS::SNS::Subscription` resource (`snsTopicForAutoScalingGroup`) that's declared within the same template.

The `VPCZoneIdentifier` property of the Auto Scaling group specifies a list of existing subnets in three different Availability Zones. You must specify the applicable subnet IDs from your account before you create your stack. The `LaunchTemplate` property references the logical name of an `AWS::EC2::LaunchTemplate` resource declared elsewhere in the same template.

JSON

```
"myASG" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
```

```
  "DependsOn": [
    "snsTopicForAutoScalingGroup"
  ],
  "Properties" : {
    "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
    "LaunchTemplate" : {
      "LaunchTemplateId" : {
        "Ref" : "logicalName"
      },
      "Version" : {
        "Fn::GetAtt" : [
          "logicalName",
          "LatestVersionNumber"
        ]
      }
    },
    "MaxSize" : "5",
    "MinSize" : "1",
    "NotificationConfigurations" : [
      {
        "TopicARN" : { "Ref" : "snsTopicForAutoScalingGroup" },
        "NotificationTypes" : [
          "autoscaling:EC2_INSTANCE_LAUNCH",
          "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
          "autoscaling:EC2_INSTANCE_TERMINATE",
          "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
          "autoscaling:TEST_NOTIFICATION"
        ]
      }
    ]
  }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  DependsOn:
    - snsTopicForAutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
```

```
      - subnetIdAz3
    LaunchTemplate:
      LaunchTemplateId: !Ref logicalName
      Version: !GetAtt logicalName.LatestVersionNumber
    MaxSize: '5'
    MinSize: '1'
    NotificationConfigurations:
      - TopicARN: !Ref snsTopicForAutoScalingGroup
        NotificationTypes:
          - autoscaling:EC2_INSTANCE_LAUNCH
          - autoscaling:EC2_INSTANCE_LAUNCH_ERROR
          - autoscaling:EC2_INSTANCE_TERMINATE
          - autoscaling:EC2_INSTANCE_TERMINATE_ERROR
          - autoscaling:TEST_NOTIFICATION
```

Create an Auto Scaling group that uses a `CreationPolicy` and an `UpdatePolicy`

The following example shows how to add `CreationPolicy` and `UpdatePolicy` attributes to an `AWS::AutoScaling::AutoScalingGroup` resource.

The sample creation policy prevents the Auto Scaling group from reaching `CREATE_COMPLETE` status until CloudFormation receives `Count` number of success signals when the group is ready. To signal that the Auto Scaling group is ready, a `cfn-signal` helper script added to the launch template's user data (not shown) is run on the instances. If the instances don't send a signal within the specified `Timeout`, CloudFormation assumes that the instances were not created, the resource creation fails, and CloudFormation rolls the stack back.

The sample update policy instructs CloudFormation to perform a rolling update using the `AutoScalingRollingUpdate` property. The rolling update makes changes to the Auto Scaling group in small batches (for this example, instance by instance) based on the `MaxBatchSize` and a pause time between batches of updates based on the `PauseTime`. The `MinInstancesInService` attribute specifies the minimum number of instances that must be in service within the Auto Scaling group while CloudFormation updates old instances.

The `WaitOnResourceSignals` attribute is set to `true`. CloudFormation must receive a signal from each new instance within the specified `PauseTime` before continuing the update. While the stack update is in progress, the following EC2 Auto Scaling processes are suspended: `HealthCheck`, `ReplaceUnhealthy`, `AZRebalance`, `AlarmNotification`, and `ScheduledActions`. Note: Don't suspend the `Launch`, `Terminate`, or `AddToLoadBalancer` (if the Auto Scaling group is being used with Elastic Load Balancing) process types because doing so can prevent the rolling update from functioning properly.

The `VPCZoneIdentifier` property of the Auto Scaling group specifies a list of existing subnets in three different Availability Zones. You must specify the applicable subnet IDs from your account before you create your stack. The `LaunchTemplate` property references the logical name of an `AWS::EC2::LaunchTemplate` resource declared elsewhere in the same template.

For more information about the `CreationPolicy` and `UpdatePolicy` attributes, see Resource attribute reference.

JSON

```
{
  "Resources":{
    "myASG":{
      "CreationPolicy":{
        "ResourceSignal":{
          "Count":"3",
          "Timeout":"PT15M"
        }
      },
      "UpdatePolicy":{
        "AutoScalingRollingUpdate":{
          "MinInstancesInService":"3",
          "MaxBatchSize":"1",
          "PauseTime":"PT12M5S",
          "WaitOnResourceSignals":"true",
          "SuspendProcesses":[
            "HealthCheck",
            "ReplaceUnhealthy",
            "AZRebalance",
            "AlarmNotification",
            "ScheduledActions",
            "InstanceRefresh"
          ]
        }
      },
      "Type":"AWS::AutoScaling::AutoScalingGroup",
      "Properties":{
        "VPCZoneIdentifier":[ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
        "LaunchTemplate":{
          "LaunchTemplateId":{
            "Ref":"logicalName"
          },
          "Version":{
```

```
            "Fn::GetAtt":[
              "logicalName",
              "LatestVersionNumber"
            ]
          }
        },
        "MaxSize":"5",
        "MinSize":"3"
      }
    }
  }
}
```

YAML

```
---
Resources:
  myASG:
    CreationPolicy:
      ResourceSignal:
        Count: '3'
        Timeout: PT15M
    UpdatePolicy:
      AutoScalingRollingUpdate:
        MinInstancesInService: '3'
        MaxBatchSize: '1'
        PauseTime: PT12M5S
        WaitOnResourceSignals: true
        SuspendProcesses:
          - HealthCheck
          - ReplaceUnhealthy
          - AZRebalance
          - AlarmNotification
          - ScheduledActions
          - InstanceRefresh
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
        - subnetIdAz1
        - subnetIdAz2
        - subnetIdAz3
      LaunchTemplate:
        LaunchTemplateId: !Ref logicalName
```

```
        Version: !GetAtt logicalName.LatestVersionNumber
      MaxSize: '5'
      MinSize: '3'
```

Create a step scaling policy

This example shows an AWS::AutoScaling::ScalingPolicy resource that scales out the Auto Scaling group using a step scaling policy. The AdjustmentType property specifies ChangeInCapacity, which means that the ScalingAdjustment represents the number of instances to add (if ScalingAdjustment is positive) or delete (if it is negative). In this example, ScalingAdjustment is 1; therefore, the policy increments the number of EC2 instances in the group by 1 when the alarm threshold is breached.

The AWS::CloudWatch::Alarm resource CPUAlarmHigh specifies the scaling policy ASGScalingPolicyHigh as the action to run when the alarm is in an ALARM state (AlarmActions). The Dimensions property references the logical name of an AWS::AutoScaling::AutoScalingGroup resource declared elsewhere in the same template.

JSON

```
{
  "Resources":{
    "ASGScalingPolicyHigh":{
      "Type":"AWS::AutoScaling::ScalingPolicy",
      "Properties":{
        "AutoScalingGroupName":{ "Ref":"logicalName" },
        "PolicyType":"StepScaling",
        "AdjustmentType":"ChangeInCapacity",
        "StepAdjustments":[
          {
            "MetricIntervalLowerBound":0,
            "ScalingAdjustment":1
          }
        ]
      }
    },
    "CPUAlarmHigh":{
      "Type":"AWS::CloudWatch::Alarm",
      "Properties":{
        "EvaluationPeriods":"2",
        "Statistic":"Average",
```

```
        "Threshold":"90",
        "AlarmDescription":"Scale out if CPU > 90% for 2 minutes",
        "Period":"60",
        "AlarmActions":[ { "Ref":"ASGScalingPolicyHigh" } ],
        "Namespace":"AWS/EC2",
        "Dimensions":[
          {
            "Name":"AutoScalingGroupName",
            "Value":{ "Ref":"logicalName" }
          }
        ],
        "ComparisonOperator":"GreaterThanThreshold",
        "MetricName":"CPUUtilization"
      }
    }
  }
}
```

YAML

```
---
Resources:
  ASGScalingPolicyHigh:
    Type: AWS::AutoScaling::ScalingPolicy
    Properties:
      AutoScalingGroupName: !Ref logicalName
      PolicyType: StepScaling
      AdjustmentType: ChangeInCapacity
      StepAdjustments:
        - MetricIntervalLowerBound: 0
          ScalingAdjustment: 1
  CPUAlarmHigh:
    Type: AWS::CloudWatch::Alarm
    Properties:
      EvaluationPeriods: 2
      Statistic: Average
      Threshold: 90
      AlarmDescription: 'Scale out if CPU > 90% for 2 minutes'
      Period: 60
      AlarmActions:
        - !Ref ASGScalingPolicyHigh
      Namespace: AWS/EC2
      Dimensions:
```

```
        - Name: AutoScalingGroupName
          Value:
            !Ref logicalName
      ComparisonOperator: GreaterThanThreshold
      MetricName: CPUUtilization
```

See also

For more example templates for scaling policies, see the Examples section in the `AWS::AutoScaling::ScalingPolicy` resource.

Mixed instances group examples

Create an Auto Scaling group using attribute-based instance type selection

This example shows an `AWS::AutoScaling::AutoScalingGroup` resource that contains the information to launch a mixed instances group using attribute-based instance type selection. You specify the minimum and maximum values for the `VCpuCount` property and the minimum value for the `MemoryMiB` property. Any instance types used by the Auto Scaling group must match your required instance attributes.

The `VPCZoneIdentifier` property of the Auto Scaling group specifies a list of existing subnets in three different Availability Zones. You must specify the applicable subnet IDs from your account before you create your stack. The `LaunchTemplate` property references the logical name of an `AWS::EC2::LaunchTemplate` resource declared elsewhere in the same template.

JSON

```
{
  "Resources":{
    "myASG":{
      "Type":"AWS::AutoScaling::AutoScalingGroup",
      "Properties":{
        "VPCZoneIdentifier":[
          "subnetIdAz1",
          "subnetIdAz2",
          "subnetIdAz3"
        ],
        "MixedInstancesPolicy":{
          "LaunchTemplate":{
            "LaunchTemplateSpecification":{
              "LaunchTemplateId":{
```

```
                "Ref":"logicalName"
              },
              "Version":{
                "Fn::GetAtt":[
                  "logicalName",
                  "LatestVersionNumber"
                ]
              }
            },
            "Overrides":[
              {
                "InstanceRequirements":{
                  "VCpuCount":{
                    "Min":2,
                    "Max":4
                  },
                  "MemoryMiB":{
                    "Min":2048
                  }
                }
              }
            ]
          }
        },
        "MaxSize":"5",
        "MinSize":"1"
      }
    }
  }
}
```

YAML

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      VPCZoneIdentifier:
        - subnetIdAz1
        - subnetIdAz2
        - subnetIdAz3
      MixedInstancesPolicy:
```

```
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref logicalName
            Version: !GetAtt logicalName.LatestVersionNumber
          Overrides:
            - InstanceRequirements:
                VCpuCount:
                  Min: 2
                  Max: 4
                MemoryMiB:
                  Min: 2048
      MaxSize: '5'
      MinSize: '1'
```

Launch configuration examples

Create a launch configuration

This example shows an AWS::AutoScaling::LaunchConfiguration resource for an Auto Scaling group where you specify values for the ImageId, InstanceType, and SecurityGroups properties. The SecurityGroups property specifies both the logical name of an AWS::EC2::SecurityGroup resource that's specified elsewhere in the template, and an existing EC2 security group named myExistingEC2SecurityGroup.

JSON

```
"mySimpleConfig" : {
   "Type" : "AWS::AutoScaling::LaunchConfiguration",
   "Properties" : {
      "ImageId" : "ami-02354e95b3example",
      "InstanceType" : "t3.micro",
      "SecurityGroups" : [ { "Ref" : "logicalName" }, "myExistingEC2SecurityGroup" ]
   }
}
```

YAML

```
mySimpleConfig:
  Type: AWS::AutoScaling::LaunchConfiguration
  Properties:
```

```
    ImageId: ami-02354e95b3example
    InstanceType: t3.micro
    SecurityGroups:
      - !Ref logicalName
      - myExistingEC2SecurityGroup
```

Create an Auto Scaling group that uses a launch configuration

This example shows an AWS::AutoScaling::AutoScalingGroup resource with a single instance. The VPCZoneIdentifier property of the Auto Scaling group specifies a list of existing subnets in three different Availability Zones. You must specify the applicable subnet IDs from your account before you create your stack. The LaunchConfigurationName property references an AWS::AutoScaling::LaunchConfiguration resource with the logical name mySimpleConfig that is defined in your template.

JSON

```
"myASG" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "VPCZoneIdentifier" : [ "subnetIdAz1", "subnetIdAz2", "subnetIdAz3" ],
      "LaunchConfigurationName" : { "Ref" : "mySimpleConfig" },
      "MaxSize" : "1",
      "MinSize" : "1"
   }
}
```

YAML

```
myASG:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    VPCZoneIdentifier:
      - subnetIdAz1
      - subnetIdAz2
      - subnetIdAz3
    LaunchConfigurationName: !Ref mySimpleConfig
    MaxSize: '1'
    MinSize: '1'
```

Configure Application Auto Scaling resources with CloudFormation

This section provides CloudFormation template examples for Application Auto Scaling scaling policies and scheduled actions for different AWS resources.

> **Important**
>
> When an Application Auto Scaling snippet is included in the template, you might need to declare a dependency on the specific scalable resource that's created through the template using the `DependsOn` attribute. This overrides the default parallelism and directs CloudFormation to operate on resources in a specified order. Otherwise, the scaling configuration might be applied before the resource has been set up completely.
> For more information, see DependsOn attribute.

Snippet categories

- Create a scaling policy for an AppStream fleet
- Create a scaling policy for an Aurora DB cluster
- Create a scaling policy for a DynamoDB table
- Create a scaling policy for an Amazon ECS service (metrics: average CPU and memory)
- Create a scaling policy for an Amazon ECS service (metric: average request count per target)
- Create a scheduled action with a cron expression for a Lambda function
- Create a scheduled action with an at expression for a Spot Fleet

Create a scaling policy for an AppStream fleet

This snippet shows how to create a policy and apply it to an `AWS::AppStream::Fleet` resource using the `AWS::ApplicationAutoScaling::ScalingPolicy` resource. The `AWS::ApplicationAutoScaling::ScalableTarget` resource declares a scalable target to which this policy is applied. Application Auto Scaling can scale the number of fleet instances at a minimum of 1 instance and a maximum of 20. The policy keeps the average capacity utilization of the fleet at 75 percent, with scale-out and scale-in cooldown periods of 300 seconds (5 minutes).

It uses the `Fn::Join` and `Rev` intrinsic functions to construct the `ResourceId` property with the logical name of the `AWS::AppStream::Fleet` resource that's specified in the same template. For more information, see Intrinsic function reference.

JSON

```
{
  "Resources" : {
    "ScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 20,
        "MinCapacity" : 1,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet" },
        "ServiceNamespace" : "appstream",
        "ScalableDimension" : "appstream:fleet:DesiredCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "fleet",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        }
      }
    },
    "ScalingPolicyAppStreamFleet" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu75" },
        "PolicyType" : "TargetTrackingScaling",
        "ServiceNamespace" : "appstream",
        "ScalableDimension" : "appstream:fleet:DesiredCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "fleet",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
```

```
        },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 75,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "AppStreamAverageCapacityUtilization"
          },
          "ScaleInCooldown" : 300,
          "ScaleOutCooldown" : 300
        }
      }
    }
  }
}
```

YAML

```
---
Resources:
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 20
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet'
      ServiceNamespace: appstream
      ScalableDimension: appstream:fleet:DesiredCapacity
      ResourceId: !Join
        - /
        - - fleet
          - !Ref logicalName
  ScalingPolicyAppStreamFleet:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu75
      PolicyType: TargetTrackingScaling
      ServiceNamespace: appstream
      ScalableDimension: appstream:fleet:DesiredCapacity
      ResourceId: !Join
        - /
        - - fleet
```

```
          - !Ref logicalName
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 75
        PredefinedMetricSpecification:
          PredefinedMetricType: AppStreamAverageCapacityUtilization
        ScaleInCooldown: 300
        ScaleOutCooldown: 300
```

Create a scaling policy for an Aurora DB cluster

In this snippet, you register an `AWS::RDS::DBCluster` resource. The `AWS::ApplicationAutoScaling::ScalableTarget` resource indicates that the DB cluster should be dynamically scaled to have from one to eight Aurora Replicas. You also apply a target tracking scaling policy to the cluster using the `AWS::ApplicationAutoScaling::ScalingPolicy` resource.

In this configuration, the `RDSReaderAverageCPUUtilization` predefined metric is used to adjust an Aurora DB cluster based on an average CPU utilization of 40 percent across all Aurora Replicas in that Aurora DB cluster. The configuration provides a scale-in cooldown period of 10 minutes and a scale-out cooldown period of 5 minutes.

This example uses the `Fn::Sub` intrinsic function to construct the `ResourceId` property with the logical name of the `AWS::RDS::DBCluster` resource that is specified in the same template. For more information, see Intrinsic function reference.

JSON

```
{
  "Resources" : {
    "ScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 8,
        "MinCapacity" : 1,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/rds.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_RDSCluster" },
        "ServiceNamespace" : "rds",
        "ScalableDimension" : "rds:cluster:ReadReplicaCount",
        "ResourceId" : { "Fn::Sub" : "cluster:${logicalName}" }
      }
    },
```

```
    "ScalingPolicyDBCluster" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu40" },
        "PolicyType" : "TargetTrackingScaling",
        "ServiceNamespace" : "rds",
        "ScalableDimension" : "rds:cluster:ReadReplicaCount",
        "ResourceId" : { "Fn::Sub" : "cluster:${logicalName}" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 40,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "RDSReaderAverageCPUUtilization"
          },
          "ScaleInCooldown" : 600,
          "ScaleOutCooldown" : 300
        }
      }
    }
  }
}
```

YAML

```
---
Resources:
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 8
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/rds.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster'
      ServiceNamespace: rds
      ScalableDimension: rds:cluster:ReadReplicaCount
      ResourceId: !Sub cluster:${logicalName}
  ScalingPolicyDBCluster:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu40
      PolicyType: TargetTrackingScaling
      ServiceNamespace: rds
      ScalableDimension: rds:cluster:ReadReplicaCount
```

```
      ResourceId: !Sub cluster:${logicalName}
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 40
        PredefinedMetricSpecification:
          PredefinedMetricType: RDSReaderAverageCPUUtilization
        ScaleInCooldown: 600
        ScaleOutCooldown: 300
```

Create a scaling policy for a DynamoDB table

This snippet shows how to create a policy with the `TargetTrackingScaling` policy type and apply it to an `AWS::DynamoDB::Table` resource using the `AWS::ApplicationAutoScaling::ScalingPolicy` resource. The `AWS::ApplicationAutoScaling::ScalableTarget` resource declares a scalable target to which this policy is applied, with a minimum of five write capacity units and a maximum of 15. The scaling policy scales the table's write capacity throughput to maintain the target utilization at 50 percent based on the `DynamoDBWriteCapacityUtilization` predefined metric.

It uses the `Fn::Join` and `Ref` intrinsic functions to construct the `ResourceId` property with the logical name of the `AWS::DynamoDB::Table` resource that's specified in the same template. For more information, see Intrinsic function reference.

> **Note**
>
> For more information about how to create an CloudFormation template for DynamoDB resources, see the blog post How to use CloudFormation to configure auto scaling for Amazon DynamoDB tables and indexes on the AWS Database Blog.

JSON

```
{
  "Resources" : {
    "WriteCapacityScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : 15,
        "MinCapacity" : 5,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable" },
```

```
        "ServiceNamespace" : "dynamodb",
        "ScalableDimension" : "dynamodb:table:WriteCapacityUnits",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "table",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        }
      }
    },
    "WriteScalingPolicy" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : "WriteScalingPolicy",
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "WriteCapacityScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 50.0,
          "ScaleInCooldown" : 60,
          "ScaleOutCooldown" : 60,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "DynamoDBWriteCapacityUtilization"
          }
        }
      }
    }
  }
}
```

YAML

```
---
Resources:
  WriteCapacityScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 15
      MinCapacity: 5
```

```
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable'
      ServiceNamespace: dynamodb
      ScalableDimension: dynamodb:table:WriteCapacityUnits
      ResourceId: !Join
        - /
        - - table
          - !Ref logicalName
  WriteScalingPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: WriteScalingPolicy
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref WriteCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 50.0
        ScaleInCooldown: 60
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: DynamoDBWriteCapacityUtilization
```

Create a scaling policy for an Amazon ECS service (metrics: average CPU and memory)

This snippet shows how to create a policy and apply it to an `AWS::ECS::Service` resource using the `AWS::ApplicationAutoScaling::ScalingPolicy` resource. The `AWS::ApplicationAutoScaling::ScalableTarget` resource declares a scalable target to which this policy is applied. Application Auto Scaling can scale the number of tasks at a minimum of 1 task and a maximum of 6.

It creates two scaling policies with the `TargetTrackingScaling` policy type. The policies are used to scale the ECS service based on the service's average CPU and memory usage. It uses the `Fn::Join` and `Ref` intrinsic functions to construct the `ResourceId` property with the logical names of the `AWS::ECS::Cluster` (`myContainerCluster`) and `AWS::ECS::Service` (`myService`) resources that are specified in the same template. For more information, see Intrinsic function reference.

JSON

```
{
  "Resources" : {
```

```
    "ECSScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : "6",
        "MinCapacity" : "1",
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService" },
        "ServiceNamespace" : "ecs",
        "ScalableDimension" : "ecs:service:DesiredCount",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "service",
              {
                "Ref" : "myContainerCluster"
              },
              {
                "Fn::GetAtt" : [
                  "myService",
                  "Name"
                ]
              }
            ]
          ]
        }
      }
    },
    "ServiceScalingPolicyCPU" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-cpu70" },
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 70.0,
          "ScaleInCooldown" : 180,
          "ScaleOutCooldown" : 60,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "ECSServiceAverageCPUUtilization"
          }
        }
      }
```

```
    },
    "ServiceScalingPolicyMem" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : { "Fn::Sub" : "${AWS::StackName}-target-tracking-mem90" },
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : 90.0,
          "ScaleInCooldown" : 180,
          "ScaleOutCooldown" : 60,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "ECSServiceAverageMemoryUtilization"
          }
        }
      }
    }
  }
}
```

YAML

```
---
Resources:
  ECSScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 6
      MinCapacity: 1
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ecs.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService'
      ServiceNamespace: ecs
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId: !Join
        - /
        - - service
          - !Ref myContainerCluster
          - !GetAtt myService.Name
  ServiceScalingPolicyCPU:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-cpu70
```

```
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 70.0
        ScaleInCooldown: 180
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: ECSServiceAverageCPUUtilization
  ServiceScalingPolicyMem:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: !Sub ${AWS::StackName}-target-tracking-mem90
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 90.0
        ScaleInCooldown: 180
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: ECSServiceAverageMemoryUtilization
```

Create a scaling policy for an Amazon ECS service (metric: average request count per target)

The following example applies a target tracking scaling policy with the `ALBRequestCountPerTarget` predefined metric to an ECS service. The policy is used to add capacity to the ECS service when the request count per target (per minute) exceeds the target value. Because the value of `DisableScaleIn` is set to `true`, the target tracking policy won't remove capacity from the scalable target.

It uses the `Fn::Join` and `Fn::GetAtt` intrinsic functions to construct the `ResourceLabel` property with the logical names of the `AWS::ElasticLoadBalancingV2::LoadBalancer` (`myLoadBalancer`) and `AWS::ElasticLoadBalancingV2::TargetGroup` (`myTargetGroup`) resources that are specified in the same template. For more information, see Intrinsic function reference.

The `MaxCapacity` and `MinCapacity` properties of the scalable target and the `TargetValue` property of the scaling policy reference parameter values that you pass to the template when creating or updating a stack.

JSON

```
{
```

```
  "Resources" : {
    "ECSScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
        "MaxCapacity" : { "Ref" : "MaxCount" },
        "MinCapacity" : { "Ref" : "MinCount" },
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService" },
        "ServiceNamespace" : "ecs",
        "ScalableDimension" : "ecs:service:DesiredCount",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "service",
              {
                "Ref" : "myContainerCluster"
              },
              {
                "Fn::GetAtt" : [
                  "myService",
                  "Name"
                ]
              }
            ]
          ]
        }
      }
    },
    "ServiceScalingPolicyALB" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalingPolicy",
      "Properties" : {
        "PolicyName" : "alb-requests-per-target-per-minute",
        "PolicyType" : "TargetTrackingScaling",
        "ScalingTargetId" : { "Ref" : "ECSScalableTarget" },
        "TargetTrackingScalingPolicyConfiguration" : {
          "TargetValue" : { "Ref" : "ALBPolicyTargetValue" },
          "ScaleInCooldown" : 180,
          "ScaleOutCooldown" : 30,
          "DisableScaleIn" : true,
          "PredefinedMetricSpecification" : {
            "PredefinedMetricType" : "ALBRequestCountPerTarget",
            "ResourceLabel" : {
```

```
              "Fn::Join" : [
                "/",
                [
                  {
                    "Fn::GetAtt" : [
                      "myLoadBalancer",
                      "LoadBalancerFullName"
                    ]
                  },
                  {
                    "Fn::GetAtt" : [
                      "myTargetGroup",
                      "TargetGroupFullName"
                    ]
                  }
                ]
              ]
            }
          }
        }
      }
    }
  }
}
```

YAML

```
---
Resources:
  ECSScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: !Ref MaxCount
      MinCapacity: !Ref MinCount
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ecs.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService'
      ServiceNamespace: ecs
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId: !Join
        - /
        - - service
          - !Ref myContainerCluster
```

```
          - !GetAtt myService.Name
  ServiceScalingPolicyALB:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: alb-requests-per-target-per-minute
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref ECSScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: !Ref ALBPolicyTargetValue
        ScaleInCooldown: 180
        ScaleOutCooldown: 30
        DisableScaleIn: true
        PredefinedMetricSpecification:
          PredefinedMetricType: ALBRequestCountPerTarget
          ResourceLabel: !Join
            - '/'
            - - !GetAtt myLoadBalancer.LoadBalancerFullName
              - !GetAtt myTargetGroup.TargetGroupFullName
```

Create a scheduled action with a cron expression for a Lambda function

This snippet registers the provisioned concurrency for a function alias (`AWS::Lambda::Alias`) named BLUE using the `AWS::ApplicationAutoScaling::ScalableTarget` resource. It also creates a scheduled action with a recurring schedule using a cron expression. The time zone for the recurring schedule is UTC.

It uses the `Fn::Join` and `Ref` intrinsic functions in the `RoleARN` property to specify the ARN of the service-linked role. It uses the `Fn::Sub` intrinsic function to construct the `ResourceId` property with the logical name of the `AWS::Lambda::Function` or `AWS::Serverless::Function` resource that is specified in the same template. For more information, see Intrinsic function reference.

> **Note**
>
> You can't allocate provisioned concurrency on an alias that points to the unpublished version ($LATEST).
> For more information about how to create an CloudFormation template for Lambda resources, see the blog post Scheduling AWS Lambda Provisioned Concurrency for recurring peak usage on the AWS Compute Blog.

JSON

```
{
  "ScalableTarget" : {
    "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
    "Properties" : {
      "MaxCapacity" : 250,
      "MinCapacity" : 0,
      "RoleARN" : {
        "Fn::Join" : [
          ":",
          [
            "arn:aws:iam:",
            {
              "Ref" : "AWS::AccountId"
            },
            "role/aws-service-role/lambda.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency"
          ]
        ]
      },
      "ServiceNamespace" : "lambda",
      "ScalableDimension" : "lambda:function:ProvisionedConcurrency",
      "ResourceId" : { "Fn::Sub" : "function:${logicalName}:BLUE" },
      "ScheduledActions" : [
        {
          "ScalableTargetAction" : {
            "MinCapacity" : "250"
          },
          "ScheduledActionName" : "my-scale-out-scheduled-action",
          "Schedule" : "cron(0 18 * * ? *)",
          "EndTime" : "2022-12-31T12:00:00.000Z"
        }
      ]
    }
  }
}
```

YAML

```
ScalableTarget:
  Type: AWS::ApplicationAutoScaling::ScalableTarget
  Properties:
```

```
    MaxCapacity: 250
    MinCapacity: 0
    RoleARN: !Join
      - ':'
      - - 'arn:aws:iam:'
        - !Ref 'AWS::AccountId'
        - role/aws-service-role/lambda.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency
    ServiceNamespace: lambda
    ScalableDimension: lambda:function:ProvisionedConcurrency
    ResourceId: !Sub function:${logicalName}:BLUE
    ScheduledActions:
      - ScalableTargetAction:
          MinCapacity: 250
        ScheduledActionName: my-scale-out-scheduled-action
        Schedule: 'cron(0 18 * * ? *)'
        EndTime: '2022-12-31T12:00:00.000Z'
```

Create a scheduled action with an `at` expression for a Spot Fleet

This snippet shows how to create two scheduled actions that occur only once for an `AWS::EC2::SpotFleet` resource using the `AWS::ApplicationAutoScaling::ScalableTarget` resource. The time zone for each one-time scheduled action is UTC.

It uses the `Fn::Join` and `Ref` intrinsic functions to construct the `ResourceId` property with the logical name of the `AWS::EC2::SpotFleet` resource that is specified in the same template. For more information, see Intrinsic function reference.

> **Note**
>
> The Spot Fleet request must have a request type of `maintain`. Automatic scaling isn't supported for one-time requests or Spot blocks.

JSON

```
{
  "Resources" : {
    "SpotFleetScalableTarget" : {
      "Type" : "AWS::ApplicationAutoScaling::ScalableTarget",
      "Properties" : {
```

```
        "MaxCapacity" : 0,
        "MinCapacity" : 0,
        "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ec2.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest" },
        "ServiceNamespace" : "ec2",
        "ScalableDimension" : "ec2:spot-fleet-request:TargetCapacity",
        "ResourceId" : {
          "Fn::Join" : [
            "/",
            [
              "spot-fleet-request",
              {
                "Ref" : "logicalName"
              }
            ]
          ]
        },
        "ScheduledActions" : [
          {
            "ScalableTargetAction" : {
              "MaxCapacity" : "10",
              "MinCapacity" : "10"
            },
            "ScheduledActionName" : "my-scale-out-scheduled-action",
            "Schedule" : "at(2022-05-20T13:00:00)"
          },
          {
            "ScalableTargetAction" : {
              "MaxCapacity" : "0",
              "MinCapacity" : "0"
            },
            "ScheduledActionName" : "my-scale-in-scheduled-action",
            "Schedule" : "at(2022-05-20T21:00:00)"
          }
        ]
      }
    }
  }
}
```

YAML

```
---
Resources:
  SpotFleetScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 0
      MinCapacity: 0
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/ec2.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest'
      ServiceNamespace: ec2
      ScalableDimension: 'ec2:spot-fleet-request:TargetCapacity'
      ResourceId: !Join
        - /
        - - spot-fleet-request
          - !Ref logicalName
      ScheduledActions:
        - ScalableTargetAction:
            MaxCapacity: 10
            MinCapacity: 10
          ScheduledActionName: my-scale-out-scheduled-action
          Schedule: 'at(2022-05-20T13:00:00)'
        - ScalableTargetAction:
            MaxCapacity: 0
            MinCapacity: 0
          ScheduledActionName: my-scale-in-scheduled-action
          Schedule: 'at(2022-05-20T21:00:00)'
```

AWS Billing Console template snippets

This example creates one pricing plan with a 10% global markup pricing rule. This pricing plan is attached to the billing group. The billing group also has two custom line items which applies a $10 charge and a 10% charge on top of the billing group total cost.

JSON

```
{
   "Parameters": {
      "LinkedAccountIds": {
         "Type": "ListNumber"
```

```
      },
      "PrimaryAccountId": {
         "Type": "Number"
      }
   },
   "Resources": {
      "TestPricingRule": {
         "Type": "AWS::BillingConductor::PricingRule",
         "Properties": {
            "Name": "TestPricingRule",
            "Description": "Test pricing rule created through Cloudformation. Mark
everything by 10%.",
            "Type": "MARKUP",
            "Scope": "GLOBAL",
            "ModifierPercentage": 10
         }
      },
      "TestPricingPlan": {
         "Type": "AWS::BillingConductor::PricingPlan",
         "Properties": {
            "Name": "TestPricingPlan",
            "Description": "Test pricing plan created through Cloudformation.",
            "PricingRuleArns": [
               {"Fn::GetAtt": ["TestPricingRule", "Arn"]}
            ]
         }
      },
      "TestBillingGroup": {
         "Type": "AWS::BillingConductor::BillingGroup",
         "Properties": {
            "Name": "TestBillingGroup",
            "Description": "Test billing group created through Cloudformation with 1
linked account. The linked account is also the primary account.",
            "PrimaryAccountId": {
               "Ref": "PrimaryAccountId"
            },
            "AccountGrouping": {
               "LinkedAccountIds": null
            },
            "ComputationPreference": {
               "PricingPlanArn": {
                 "Fn::GetAtt": ["TestPricingPlan", "Arn"]
               }
            }
```

```
         }
      },
      "TestFlatCustomLineItem": {
         "Type": "AWS::BillingConductor::CustomLineItem",
         "Properties": {
            "Name": "TestFlatCustomLineItem",
            "Description": "Test flat custom line item created through Cloudformation
 for a $10 charge.",
            "BillingGroupArn": {
              "Fn::GetAtt": ["TestBillingGroup", "Arn"]
            },
            "CustomLineItemChargeDetails": {
               "Flat": {
                  "ChargeValue": 10
               },
               "Type": "FEE"
            }
         }
      },
      "TestPercentageCustomLineItem": {
         "Type": "AWS::BillingConductor::CustomLineItem",
         "Properties": {
            "Name": "TestPercentageCustomLineItem",
            "Description": "Test percentage custom line item created through
 Cloudformation for a %10 additional charge on the overall total bill of the billing
 group.",
            "BillingGroupArn": {
              "Fn::GetAtt": ["TestBillingGroup", "Arn"]
            },
            "CustomLineItemChargeDetails": {
               "Percentage": {
                  "PercentageValue": 10,
                  "ChildAssociatedResources": [
                     {"Fn::GetAtt": ["TestBillingGroup", "Arn"]}
                  ]
               },
               "Type": "FEE"
            }
         }
      }
   }
}
```

YAML

```
Parameters:
  LinkedAccountIds:
    Type: ListNumber
  PrimaryAccountId:
    Type: Number
Resources:
  TestPricingRule:
    Type: AWS::BillingConductor::PricingRule
    Properties:
      Name: 'TestPricingRule'
      Description: 'Test pricing rule created through Cloudformation. Mark everything
 by 10%.'
      Type: 'MARKUP'
      Scope: 'GLOBAL'
      ModifierPercentage: 10
  TestPricingPlan:
    Type: AWS::BillingConductor::PricingPlan
    Properties:
      Name: 'TestPricingPlan'
      Description: 'Test pricing plan created through Cloudformation.'
      PricingRuleArns:
        - !GetAtt TestPricingRule.Arn
  TestBillingGroup:
    Type: AWS::BillingConductor::BillingGroup
    Properties:
      Name: 'TestBillingGroup'
      Description: 'Test billing group created through Cloudformation with 1 linked
 account. The linked account is also the primary account.'
      PrimaryAccountId: !Ref PrimaryAccountId
      AccountGrouping:
        LinkedAccountIds: !Ref LinkedAccountIds
      ComputationPreference:
        PricingPlanArn: !GetAtt TestPricingPlan.Arn
  TestFlatCustomLineItem:
    Type: AWS::BillingConductor::CustomLineItem
    Properties:
      Name: 'TestFlatCustomLineItem'
      Description: 'Test flat custom line item created through Cloudformation for a $10
 charge.'
      BillingGroupArn: !GetAtt TestBillingGroup.Arn
      CustomLineItemChargeDetails:
```

```
        Flat:
          ChargeValue: 10
        Type: 'FEE'
  TestPercentageCustomLineItem:
    Type: AWS::BillingConductor::CustomLineItem
    Properties:
      Name: 'TestPercentageCustomLineItem'
      Description: 'Test percentage custom line item created through Cloudformation for
a %10 additional charge on the overall total bill of the billing group.'
      BillingGroupArn: !GetAtt TestBillingGroup.Arn
      CustomLineItemChargeDetails:
        Percentage:
          PercentageValue: 10
          ChildAssociatedResources:
            - !GetAtt TestBillingGroup.Arn
        Type: 'FEE'
```

CloudFormation template snippets

Topics

- Nested stacks
- Wait condition

Nested stacks

Nesting a stack in a template

This example template contains a nested stack resource called `myStack`. When CloudFormation creates a stack from the template, it creates the `myStack`, whose template is specified in the `TemplateURL` property. The output value `StackRef` returns the stack ID for `myStack` and the value `OutputFromNestedStack` returns the output value `BucketName` from within the `myStack` resource. The `Outputs.`*`nestedstackoutputname`* format is reserved for specifying output values from nested stacks and can be used anywhere within the containing template.

For more information, see AWS::CloudFormation::Stack.

JSON

```
{
```

```
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myStack" : {
        "Type" : "AWS::CloudFormation::Stack",
        "Properties" : {
         "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-1/
S3_Bucket.template",
              "TimeoutInMinutes" : "60"
        }
        }
    },
    "Outputs": {
       "StackRef": {"Value": { "Ref" : "myStack"}},
       "OutputFromNestedStack" : {
             "Value" : { "Fn::GetAtt" : [ "myStack", "Outputs.BucketName" ] }
       }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cloudformation-templates-us-east-1/
S3_Bucket.template
      TimeoutInMinutes: '60'
Outputs:
  StackRef:
    Value: !Ref myStack
  OutputFromNestedStack:
    Value: !GetAtt myStack.Outputs.BucketName
```

Nesting a stack with input parameters in a template

This example template contains a stack resource that specifies input parameters. When CloudFormation creates a stack from this template, it uses the value pairs declared within the `Parameters` property as the input parameters for the template used to create the `myStackWithParams` stack. In this example, the `InstanceType` and `KeyName` parameters are specified.

For more information, see AWS::CloudFormation::Stack.

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myStackWithParams" : {
          "Type" : "AWS::CloudFormation::Stack",
        "Properties" : {
            "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-
east-1/EC2ChooseAMI.template",
            "Parameters" : {
                "InstanceType" : "t2.micro",
                "KeyName" : "mykey"
            }
           }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myStackWithParams:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cloudformation-templates-us-east-1/
EC2ChooseAMI.template
      Parameters:
        InstanceType: t2.micro
        KeyName: mykey
```

Wait condition

Using a wait condition with an Amazon EC2 instance

> **Important**
>
> For Amazon EC2 and Auto Scaling resources, we recommend that you use a CreationPolicy attribute instead of wait conditions. Add a CreationPolicy attribute to those resources, and

use the cfn-signal helper script to signal when an instance creation process has completed successfully.

If you can't use a creation policy, you view the following example template, which declares an Amazon EC2 instance with a wait condition. The `myWaitCondition` wait condition uses `myWaitConditionHandle` for signaling, uses the `DependsOn` attribute to specify that the wait condition will trigger after the Amazon EC2 instance resource has been created, and uses the `Timeout` property to specify a duration of 4500 seconds for the wait condition. In addition, the presigned URL that signals the wait condition is passed to the Amazon EC2 instance with the `UserData` property of the `Ec2Instance` resource, thus allowing an application or script running on that Amazon EC2 instance to retrieve the presigned URL and employ it to signal a success or failure to the wait condition. You need to use `cfn-signal` or create the application or script that signals the wait condition. The output value `ApplicationData` contains the data passed back from the wait condition signal.

For more information, see Create wait conditions in a CloudFormation template.

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Mappings" : {
        "RegionMap" : {
            "us-east-1" : {
                "AMI" : "ami-0123456789abcdef0"
            },
            "us-west-1" : {
                "AMI" : "ami-0987654321fedcba0"
            },
            "eu-west-1" : {
                "AMI" : "ami-0abcdef123456789a"
            },
            "ap-northeast-1" : {
                "AMI" : "ami-0fedcba987654321b"
            },
            "ap-southeast-1" : {
                "AMI" : "ami-0c1d2e3f4a5b6c7d8"
            }
        }
    },
```

```
    "Resources" : {
        "Ec2Instance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "UserData" : { "Fn::Base64" : {"Ref" : "myWaitHandle"}},
                "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" :
 "AWS::Region" }, "AMI" ]}
            }
        },
        "myWaitHandle" : {
            "Type" : "AWS::CloudFormation::WaitConditionHandle",
            "Properties" : {
            }
        },
        "myWaitCondition" : {
            "Type" : "AWS::CloudFormation::WaitCondition",
            "DependsOn" : "Ec2Instance",
            "Properties" : {
                "Handle" : { "Ref" : "myWaitHandle" },
                "Timeout" : "4500"
            }
        }
    },
    "Outputs" : {
        "ApplicationData" : {
            "Value" : { "Fn::GetAtt" : [ "myWaitCondition", "Data" ]},
            "Description" : "The data passed back as part of signalling the
 WaitCondition."
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Mappings:
  RegionMap:
    us-east-1:
      AMI: ami-0123456789abcdef0
    us-west-1:
      AMI: ami-0987654321fedcba0
    eu-west-1:
      AMI: ami-0abcdef123456789a
```

```
    ap-northeast-1:
      AMI: ami-0fedcba987654321b
    ap-southeast-1:
      AMI: ami-0c1d2e3f4a5b6c7d8
Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      UserData:
        Fn::Base64: !Ref myWaitHandle
      ImageId:
        Fn::FindInMap:
        - RegionMap
        - Ref: AWS::Region
        - AMI
  myWaitHandle:
    Type: AWS::CloudFormation::WaitConditionHandle
    Properties: {}
  myWaitCondition:
    Type: AWS::CloudFormation::WaitCondition
    DependsOn: Ec2Instance
    Properties:
      Handle: !Ref myWaitHandle
      Timeout: '4500'
Outputs:
  ApplicationData:
    Value: !GetAtt myWaitCondition.Data
    Description: The data passed back as part of signalling the WaitCondition.
```

Using the cfn-signal helper script to signal a wait condition

This example shows a `cfn-signal` command line that signals success to a wait condition. You need to define the command line in the `UserData` property of the EC2 instance.

JSON

```
"UserData": {
  "Fn::Base64": {
    "Fn::Join": [
      "",
      [
         "#!/bin/bash -xe\n",
         "/opt/aws/bin/cfn-signal --exit-code 0 '",
```

```
          {
            "Ref": "myWaitHandle"
          },
          "'\n"
        ]
      ]
    }
}
```

YAML

```
UserData:
  Fn::Base64: !Sub |
    #!/bin/bash -xe
    /opt/aws/bin/cfn-signal --exit-code 0 '${myWaitHandle}'
```

Using Curl to signal a wait condition

This example shows a Curl command line that signals success to a wait condition.

```
curl -T /tmp/a "https://cloudformation-waitcondition-test.s3.amazonaws.com/
arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack
%2Fstack-gosar-20110427004224-test-stack-with-WaitCondition--VEYW
%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVoo
%3D"
```

Where the file /tmp/a contains the following JSON structure:

```
{
  "Status" : "SUCCESS",
  "Reason" : "Configuration Complete",
  "UniqueId" : "ID1234",
  "Data" : "Application has completed configuration."
}
```

This example shows a Curl command line that sends the same success signal except it sends the JSON as a parameter on the command line.

```
curl -X PUT -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS","Reason" :
 "Configuration Complete","UniqueId" : "ID1234","Data" : "Application
```

```
 has completed configuration."}' "https://cloudformation-waitcondition-
test.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack
%2Fstack-gosar-20110427004224-test-stack-with-WaitCondition--VEYW
%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVoo
%3D"
```

Amazon CloudFront template snippets

Use these sample template snippets with your Amazon CloudFront distribution resource in CloudFormation. For more information, see the Amazon CloudFront resource type reference.

Topics

- Amazon CloudFront distribution resource with an Amazon S3 origin
- Amazon CloudFront distribution resource with custom origin
- Amazon CloudFront distribution with multi-origin support
- Amazon CloudFront distribution with a Lambda function as origin
- See also

Amazon CloudFront distribution resource with an Amazon S3 origin

The following example template shows an Amazon CloudFront Distribution using an S3Origin and legacy origin access identity (OAI). For information about using origin access control (OAC) instead, see Restricting access to an Amazon Simple Storage Service origin in the *Amazon CloudFront Developer Guide*.

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myDistribution" : {
            "Type" : "AWS::CloudFront::Distribution",
            "Properties" : {
                "DistributionConfig" : {
                    "Origins" : [ {
                        "DomainName" : "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "Id" : "myS3Origin",
                        "S3OriginConfig" : {
```

```
                            "OriginAccessIdentity" : "origin-access-identity/
cloudfront/E127EXAMPLE51Z"
                        }
                    }],
                    "Enabled" : "true",
                    "Comment" : "Some comment",
                    "DefaultRootObject" : "index.html",
                    "Logging" : {
                        "IncludeCookies" : "false",
                        "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
                        "Prefix" : "myprefix"
                    },
                    "Aliases" : [ "mysite.example.com", "yoursite.example.com" ],
                    "DefaultCacheBehavior" : {
                        "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS",
 "PATCH", "POST", "PUT" ],
                        "TargetOriginId" : "myS3Origin",
                        "ForwardedValues" : {
                            "QueryString" : "false",
                            "Cookies" : { "Forward" : "none" }
                        },
                        "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
                        "ViewerProtocolPolicy" : "allow-all"
                    },
                   "PriceClass" : "PriceClass_200",
                   "Restrictions" : {
                       "GeoRestriction" : {
                           "RestrictionType" : "whitelist",
                           "Locations" : [ "AQ", "CV" ]
                       }
                   },
                   "ViewerCertificate" : { "CloudFrontDefaultCertificate" : "true" }
               }
           }
       }
   }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myDistribution:
```

```
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
        - DomainName: amzn-s3-demo-bucket.s3.amazonaws.com
          Id: myS3Origin
          S3OriginConfig:
            OriginAccessIdentity: origin-access-identity/cloudfront/E127EXAMPLE51Z
        Enabled: 'true'
        Comment: Some comment
        DefaultRootObject: index.html
        Logging:
          IncludeCookies: 'false'
          Bucket: amzn-s3-demo-logging-bucket.s3.amazonaws.com
          Prefix: myprefix
        Aliases:
        - mysite.example.com
        - yoursite.example.com
        DefaultCacheBehavior:
          AllowedMethods:
          - DELETE
          - GET
          - HEAD
          - OPTIONS
          - PATCH
          - POST
          - PUT
          TargetOriginId: myS3Origin
          ForwardedValues:
            QueryString: 'false'
            Cookies:
              Forward: none
          TrustedSigners:
          - 1234567890EX
          - 1234567891EX
          ViewerProtocolPolicy: allow-all
        PriceClass: PriceClass_200
        Restrictions:
          GeoRestriction:
            RestrictionType: whitelist
            Locations:
            - AQ
            - CV
        ViewerCertificate:
```

```
          CloudFrontDefaultCertificate: 'true'
```

Amazon CloudFront distribution resource with custom origin

The following example template shows an Amazon CloudFront Distribution using a CustomOrigin.

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myDistribution" : {
            "Type" : "AWS::CloudFront::Distribution",
            "Properties" : {
                "DistributionConfig" : {
                    "Origins" : [ {
                            "DomainName" : "www.example.com",
                            "Id" : "myCustomOrigin",
                            "CustomOriginConfig" : {
                                "HTTPPort" : "80",
                                "HTTPSPort" : "443",
                                "OriginProtocolPolicy" : "http-only"
                            }
                    } ],
                    "Enabled" : "true",
                    "Comment" : "Somecomment",
                    "DefaultRootObject" : "index.html",
                    "Logging" : {
                        "IncludeCookies" : "true",
                        "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
                        "Prefix": "myprefix"
                    },
                    "Aliases" : [
                        "mysite.example.com",
                        "*.yoursite.example.com"
                    ],
                    "DefaultCacheBehavior" : {
                        "TargetOriginId" : "myCustomOrigin",
                        "SmoothStreaming" : "false",
                        "ForwardedValues" : {
                            "QueryString" : "false",
                            "Cookies" : { "Forward" : "all" }
                        },
```

```
                        "TrustedSigners" : [
                            "1234567890EX",
                            "1234567891EX"
                        ],
                        "ViewerProtocolPolicy" : "allow-all"
                    },
                    "CustomErrorResponses" : [ {
                        "ErrorCode" : "404",
                        "ResponsePagePath" : "/error-pages/404.html",
                        "ResponseCode" : "200",
                        "ErrorCachingMinTTL" : "30"
                    } ],
                    "PriceClass" : "PriceClass_200",
                    "Restrictions" : {
                        "GeoRestriction" : {
                            "RestrictionType" : "whitelist",
                            "Locations" : [ "AQ", "CV" ]
                        }
                    },
                    "ViewerCertificate": { "CloudFrontDefaultCertificate" : "true" }
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
        - DomainName: www.example.com
          Id: myCustomOrigin
          CustomOriginConfig:
            HTTPPort: '80'
            HTTPSPort: '443'
            OriginProtocolPolicy: http-only
        Enabled: 'true'
        Comment: Somecomment
```

```
        DefaultRootObject: index.html
        Logging:
          IncludeCookies: 'true'
          Bucket: amzn-s3-demo-logging-bucket.s3.amazonaws.com
          Prefix: myprefix
        Aliases:
        - mysite.example.com
        - "*.yoursite.example.com"
        DefaultCacheBehavior:
          TargetOriginId: myCustomOrigin
          SmoothStreaming: 'false'
          ForwardedValues:
            QueryString: 'false'
            Cookies:
              Forward: all
          TrustedSigners:
          - 1234567890EX
          - 1234567891EX
          ViewerProtocolPolicy: allow-all
        CustomErrorResponses:
        - ErrorCode: '404'
          ResponsePagePath: "/error-pages/404.html"
          ResponseCode: '200'
          ErrorCachingMinTTL: '30'
        PriceClass: PriceClass_200
        Restrictions:
          GeoRestriction:
            RestrictionType: whitelist
            Locations:
            - AQ
            - CV
        ViewerCertificate:
          CloudFrontDefaultCertificate: 'true'
```

Amazon CloudFront distribution with multi-origin support

The following example template shows how to declare a CloudFront Distribution with multi-origin support. In the DistributionConfig, a list of origins is provided and a DefaultCacheBehavior is set.

JSON

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
```

```
    "Resources" : {
        "myDistribution" : {
            "Type" : "AWS::CloudFront::Distribution",
            "Properties" : {
                "DistributionConfig" : {
                    "Origins" : [ {
                        "Id" : "myS3Origin",
                        "DomainName" : "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "S3OriginConfig" : {
                            "OriginAccessIdentity" : "origin-access-identity/
cloudfront/E127EXAMPLE51Z"
                        }
                     },
                     {
                         "Id" : "myCustomOrigin",
                         "DomainName" : "www.example.com",
                         "CustomOriginConfig" : {
                             "HTTPPort" : "80",
                             "HTTPSPort" : "443",
                             "OriginProtocolPolicy" : "http-only"
                         }
                      }
                    ],
                    "Enabled" : "true",
                    "Comment" : "Some comment",
                    "DefaultRootObject" : "index.html",
                    "Logging" : {
                        "IncludeCookies" : "true",
                        "Bucket" : "amzn-s3-demo-logging-bucket.s3.amazonaws.com",
                        "Prefix" : "myprefix"
                    },
                    "Aliases" : [ "mysite.example.com", "yoursite.example.com" ],
                    "DefaultCacheBehavior" : {
                        "TargetOriginId" : "myS3Origin",
                        "ForwardedValues" : {
                            "QueryString" : "false",
                            "Cookies" : { "Forward" : "all" }
                         },
                        "TrustedSigners" : [ "1234567890EX", "1234567891EX"  ],
                        "ViewerProtocolPolicy" : "allow-all",
                        "MinTTL" : "100",
                        "SmoothStreaming" : "true"
                    },
                    "CacheBehaviors" : [ {
```

```
                            "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS",
 "PATCH", "POST", "PUT" ],
                            "TargetOriginId" : "myS3Origin",
                            "ForwardedValues" : {
                                "QueryString" : "true",
                                "Cookies" : { "Forward" : "none" }
                            },
                            "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
                            "ViewerProtocolPolicy" : "allow-all",
                            "MinTTL" : "50",
                            "PathPattern" : "images1/*.jpg"
                        },
                        {
                            "AllowedMethods" : [ "DELETE", "GET", "HEAD", "OPTIONS",
 "PATCH", "POST", "PUT" ],
                            "TargetOriginId" : "myCustomOrigin",
                            "ForwardedValues" : {
                                "QueryString" : "true",
                                "Cookies" : { "Forward" : "none" }
                            },
                            "TrustedSigners" : [ "1234567890EX", "1234567891EX"  ],
                            "ViewerProtocolPolicy" : "allow-all",
                            "MinTTL" : "50",
                            "PathPattern" : "images2/*.jpg"
                        }
                    ],
                    "CustomErrorResponses" : [ {
                        "ErrorCode" : "404",
                        "ResponsePagePath" : "/error-pages/404.html",
                        "ResponseCode" : "200",
                        "ErrorCachingMinTTL" : "30"
                    } ],
                    "PriceClass" : "PriceClass_All",
                    "ViewerCertificate" : { "CloudFrontDefaultCertificate" : "true" }
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Resources:
  myDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        Origins:
        - Id: myS3Origin
          DomainName: amzn-s3-demo-bucket.s3.amazonaws.com
          S3OriginConfig:
            OriginAccessIdentity: origin-access-identity/cloudfront/E127EXAMPLE51Z
        - Id: myCustomOrigin
          DomainName: www.example.com
          CustomOriginConfig:
            HTTPPort: '80'
            HTTPSPort: '443'
            OriginProtocolPolicy: http-only
        Enabled: 'true'
        Comment: Some comment
        DefaultRootObject: index.html
        Logging:
          IncludeCookies: 'true'
          Bucket: amzn-s3-demo-logging-bucket.s3.amazonaws.com
          Prefix: myprefix
        Aliases:
        - mysite.example.com
        - yoursite.example.com
        DefaultCacheBehavior:
          TargetOriginId: myS3Origin
          ForwardedValues:
            QueryString: 'false'
            Cookies:
              Forward: all
          TrustedSigners:
          - 1234567890EX
          - 1234567891EX
          ViewerProtocolPolicy: allow-all
          MinTTL: '100'
          SmoothStreaming: 'true'
        CacheBehaviors:
        - AllowedMethods:
          - DELETE
          - GET
          - HEAD
          - OPTIONS
```

```
        - PATCH
        - POST
        - PUT
        TargetOriginId: myS3Origin
        ForwardedValues:
          QueryString: 'true'
          Cookies:
            Forward: none
        TrustedSigners:
        - 1234567890EX
        - 1234567891EX
        ViewerProtocolPolicy: allow-all
        MinTTL: '50'
        PathPattern: images1/*.jpg
      - AllowedMethods:
        - DELETE
        - GET
        - HEAD
        - OPTIONS
        - PATCH
        - POST
        - PUT
        TargetOriginId: myCustomOrigin
        ForwardedValues:
          QueryString: 'true'
          Cookies:
            Forward: none
        TrustedSigners:
        - 1234567890EX
        - 1234567891EX
        ViewerProtocolPolicy: allow-all
        MinTTL: '50'
        PathPattern: images2/*.jpg
      CustomErrorResponses:
      - ErrorCode: '404'
        ResponsePagePath: "/error-pages/404.html"
        ResponseCode: '200'
        ErrorCachingMinTTL: '30'
      PriceClass: PriceClass_All
      ViewerCertificate:
        CloudFrontDefaultCertificate: 'true'
```

Amazon CloudFront distribution with a Lambda function as origin

The following example creates a CloudFront distribution that fronts a specified Lambda function URL (provided as a parameter), enabling HTTPS-only access, caching, compression, and global delivery. It configures the Lambda URL as a custom HTTPS origin and applies a standard AWS caching policy. The distribution is optimized for performance with HTTP/2 and IPv6 support and outputs the CloudFront domain name, allowing users to access the Lambda function through a secure, CDN-backed endpoint. For more information, see Using Amazon CloudFront with AWS Lambda as origin to accelerate your web applications on the AWS Blog.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "LambdaEndpoint": {
            "Type": "String",
            "Description": "The Lambda function URL endpoint without the 'https://'"
        }
    },
    "Resources": {
        "MyDistribution": {
            "Type": "AWS::CloudFront::Distribution",
            "Properties": {
                "DistributionConfig": {
                    "PriceClass": "PriceClass_All",
                    "HttpVersion": "http2",
                    "IPV6Enabled": true,
                    "Origins": [
                        {
                            "DomainName": {
                                "Ref": "LambdaEndpoint"
                            },
                            "Id": "LambdaOrigin",
                            "CustomOriginConfig": {
                                "HTTPSPort": 443,
                                "OriginProtocolPolicy": "https-only"
                            }
                        }
                    ],
                    "Enabled": "true",
                    "DefaultCacheBehavior": {
```

```
                        "TargetOriginId": "LambdaOrigin",
                        "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e58f6",
                        "ViewerProtocolPolicy": "redirect-to-https",
                        "SmoothStreaming": "false",
                        "Compress": "true"
                    }
                }
            }
        }
    },
    "Outputs": {
        "CloudFrontDomain": {
            "Description": "CloudFront default domain name configured",
            "Value": {
                "Fn::Sub": "https://${MyDistribution.DomainName}/"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  LambdaEndpoint:
    Type: String
    Description: The Lambda function URL endpoint without the 'https://'
Resources:
  MyDistribution:
    Type: AWS::CloudFront::Distribution
    Properties:
      DistributionConfig:
        PriceClass: PriceClass_All
        HttpVersion: http2
        IPV6Enabled: true
        Origins:
        - DomainName: !Ref LambdaEndpoint
          Id: LambdaOrigin
          CustomOriginConfig:
            HTTPSPort: 443
            OriginProtocolPolicy: https-only
        Enabled: 'true'
        DefaultCacheBehavior:
```

```
          TargetOriginId: LambdaOrigin
          CachePolicyId: '658327ea-f89d-4fab-a63d-7e88639e58f6'
          ViewerProtocolPolicy: redirect-to-https
          SmoothStreaming: 'false'
          Compress: 'true'
Outputs:
  CloudFrontDomain:
    Description: CloudFront default domain name configured
    Value: !Sub https://${MyDistribution.DomainName}/
```

See also

For an example of adding a custom alias to a Route 53 record to make a friendly name for a CloudFront distribution, see Alias resource record set for a CloudFront distribution.

Amazon CloudWatch template snippets

Use these sample template snippets to help describe your Amazon CloudWatch resources in CloudFormation. For more information, see the Amazon CloudWatch resource type reference.

Topics

- Billing alarm
- CPU utilization alarm
- Recover an Amazon Elastic Compute Cloud instance
- Create a basic dashboard
- Create a dashboard with side-by-side widgets

Billing alarm

In the following sample, Amazon CloudWatch sends an email notification when charges to your AWS account exceed the alarm threshold. To receive usage notifications, enable billing alerts. For more information, see Create a billing alarm to monitor your estimated AWS charges in the *Amazon CloudWatch User Guide*.>

JSON

```
"SpendingAlarm": {
  "Type": "AWS::CloudWatch::Alarm",
  "Properties": {
```

```
    "AlarmDescription": { "Fn::Join": ["", [
      "Alarm if AWS spending is over $",
      { "Ref": "AlarmThreshold" }
    ]]},
    "Namespace": "AWS/Billing",
    "MetricName": "EstimatedCharges",
    "Dimensions": [{
      "Name": "Currency",
      "Value" : "USD"
    }],
    "Statistic": "Maximum",
    "Period": "21600",
    "EvaluationPeriods": "1",
    "Threshold": { "Ref": "AlarmThreshold" },
    "ComparisonOperator": "GreaterThanThreshold",
    "AlarmActions": [{
      "Ref": "BillingAlarmNotification"
    }],
    "InsufficientDataActions": [{
      "Ref": "BillingAlarmNotification"
    }]
  }
}
```

YAML

```
SpendingAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription:
      'Fn::Join':
        - ''
        - - Alarm if AWS spending is over $
          - !Ref: AlarmThreshold
    Namespace: AWS/Billing
    MetricName: EstimatedCharges
    Dimensions:
    - Name: Currency
      Value: USD
    Statistic: Maximum
    Period: '21600'
    EvaluationPeriods: '1'
    Threshold:
```

```
      !Ref: "AlarmThreshold"
    ComparisonOperator: GreaterThanThreshold
    AlarmActions:
    - !Ref: "BillingAlarmNotification"
    InsufficientDataActions:
    - !Ref: "BillingAlarmNotification"
```

CPU utilization alarm

The following sample snippet creates an alarm that sends a notification when the average CPU utilization of an Amazon EC2 instance exceeds 90 percent for more than 60 seconds over three evaluation periods.

JSON

```
"CPUAlarm" : {
  "Type" : "AWS::CloudWatch::Alarm",
  "Properties" : {
    "AlarmDescription" : "CPU alarm for my instance",
    "AlarmActions" : [ { "Ref" : "logical name of an AWS::SNS::Topic resource" } ],
    "MetricName" : "CPUUtilization",
    "Namespace" : "AWS/EC2",
    "Statistic" : "Average",
    "Period" : "60",
    "EvaluationPeriods" : "3",
    "Threshold" : "90",
    "ComparisonOperator" : "GreaterThanThreshold",
    "Dimensions" : [ {
      "Name" : "InstanceId",
      "Value" : { "Ref" : "logical name of an AWS::EC2::Instance resource" }
    } ]
  }
}
```

YAML

```
CPUAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: CPU alarm for my instance
    AlarmActions:
    - !Ref: "logical name of an AWS::SNS::Topic resource"
    MetricName: CPUUtilization
```

```
    Namespace: AWS/EC2
    Statistic: Average
    Period: '60'
    EvaluationPeriods: '3'
    Threshold: '90'
    ComparisonOperator: GreaterThanThreshold
    Dimensions:
    - Name: InstanceId
      Value: !Ref: "logical name of an AWS::EC2::Instance resource"
```

Recover an Amazon Elastic Compute Cloud instance

The following CloudWatch alarm recovers an EC2 instance when it has status check failures for 15 consecutive minutes. For more information about alarm actions, see Create alarms to stop, terminate, reboot, or recover an EC2 instance in the *Amazon CloudWatch User Guide*.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters" : {
    "RecoveryInstance" : {
      "Description" : "The EC2 instance ID to associate this alarm with.",
      "Type" : "AWS::EC2::Instance::Id"
    }
  },
  "Resources": {
    "RecoveryTestAlarm": {
      "Type": "AWS::CloudWatch::Alarm",
      "Properties": {
        "AlarmDescription": "Trigger a recovery when instance status check fails for 15
 consecutive minutes.",
        "Namespace": "AWS/EC2" ,
        "MetricName": "StatusCheckFailed_System",
        "Statistic": "Minimum",
        "Period": "60",
        "EvaluationPeriods": "15",
        "ComparisonOperator": "GreaterThanThreshold",
        "Threshold": "0",
        "AlarmActions": [ {"Fn::Join" : ["", ["arn:aws:automate:", { "Ref" :
 "AWS::Region" }, ":ec2:recover" ]]} ],
        "Dimensions": [{"Name": "InstanceId","Value": {"Ref": "RecoveryInstance"}}]
      }
```

```
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  RecoveryInstance:
    Description: The EC2 instance ID to associate this alarm with.
    Type: AWS::EC2::Instance::Id
Resources:
  RecoveryTestAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: Trigger a recovery when instance status check fails for 15
        consecutive minutes.
      Namespace: AWS/EC2
      MetricName: StatusCheckFailed_System
      Statistic: Minimum
      Period: '60'
      EvaluationPeriods: '15'
      ComparisonOperator: GreaterThanThreshold
      Threshold: '0'
      AlarmActions: [ !Sub "arn:aws:automate:${AWS::Region}:ec2:recover" ]
      Dimensions:
      - Name: InstanceId
        Value: !Ref: RecoveryInstance
```

Create a basic dashboard

The following example creates a simple CloudWatch dashboard with one metric widget displaying CPU utilization, and one text widget displaying a message.

JSON

```
{
    "BasicDashboard": {
        "Type": "AWS::CloudWatch::Dashboard",
        "Properties": {
            "DashboardName": "Dashboard1",
            "DashboardBody": "{\"widgets\":[{\"type\":\"metric\",\"x\":0,\"y\":0,
\"width\":12,\"height\":6,\"properties\":{\"metrics\":[[\"AWS/EC2\",\"CPUUtilization
```

```
\",\"InstanceId\",\"i-012345\"]],\"period\":300,\"stat\":\"Average\",\"region\":\"us-
east-1\",\"title\":\"EC2 Instance CPU\"}},{\"type\":\"text\",\"x\":0,\"y\":7,\"width
\":3,\"height\":3,\"properties\":{\"markdown\":\"Hello world\"}}]}"
        }
    }
}
```

YAML

```
BasicDashboard:
  Type: AWS::CloudWatch::Dashboard
  Properties:
    DashboardName: Dashboard1
    DashboardBody: '{"widgets":
[{"type":"metric","x":0,"y":0,"width":12,"height":6,"properties":{"metrics":[["AWS/
EC2","CPUUtilization","InstanceId","i-012345"]],"period":300,"stat":"Average","region":"us-
east-1","title":"EC2 Instance CPU"}},
{"type":"text","x":0,"y":7,"width":3,"height":3,"properties":{"markdown":"Hello
 world"}}]}'
```

Create a dashboard with side-by-side widgets

The following example creates a dashboard with two metric widgets that appear side by side.

JSON

```
{
    "DashboardSideBySide": {
        "Type": "AWS::CloudWatch::Dashboard",
        "Properties": {
            "DashboardName": "Dashboard1",
            "DashboardBody": "{\"widgets\":[{\"type\":\"metric\",\"x\":0,\"y\":0,
\"width\":12,\"height\":6,\"properties\":{\"metrics\":[[\"AWS/EC2\",\"CPUUtilization
\",\"InstanceId\",\"i-012345\"]],\"period\":300,\"stat\":\"Average\",\"region\":
\"us-east-1\",\"title\":\"EC2 Instance CPU\"}},{\"type\":\"metric\",\"x\":12,\"y\":0,
\"width\":12,\"height\":6,\"properties\":{\"metrics\":[[\"AWS/S3\",\"BucketSizeBytes\",
\"BucketName\",\"amzn-s3-demo-bucket\"]],\"period\":86400,\"stat\":\"Maximum\",\"region
\":\"us-east-1\",\"title\":\"amzn-s3-demo-bucket bytes\"}}]}"
        }
    }
}
```

YAML

```
DashboardSideBySide:
  Type: AWS::CloudWatch::Dashboard
  Properties:
    DashboardName: Dashboard1
    DashboardBody: '{"widgets":
[{"type":"metric","x":0,"y":0,"width":12,"height":6,"properties":{"metrics":[["AWS/
EC2","CPUUtilization","InstanceId","i-012345"]],"period":300,"stat":"Average","region":"us-
east-1","title":"EC2 Instance CPU"}},
{"type":"metric","x":12,"y":0,"width":12,"height":6,"properties":
{"metrics":[["AWS/S3","BucketSizeBytes","BucketName","amzn-s3-demo-
bucket"]],"period":86400,"stat":"Maximum","region":"us-east-1","title":"amzn-s3-demo-
bucket bytes"}}]}'
```

Amazon CloudWatch Logs template snippets

Amazon CloudWatch Logs can monitor your system, application, and custom log files from Amazon EC2 instances or other sources. You can use CloudFormation to provision and manage log groups and metric filters. For more information about Amazon CloudWatch Logs, see the Amazon CloudWatch Logs User Guide.

Topics

- Send logs to CloudWatch Logs from a Linux instance
- Send logs to CloudWatch Logs from a Windows instance
- See also

Send logs to CloudWatch Logs from a Linux instance

The following template demonstrates how to set up a web server on Amazon Linux 2023 with CloudWatch Logs integration. The template performs the following tasks:

- Installs Apache and PHP.
- Configures the CloudWatch agent to forward Apache access logs to CloudWatch Logs.
- Sets up an IAM role to allow the CloudWatch agent to send log data to CloudWatch Logs.
- Creates custom alarms and notifications to monitor for 404 errors or high bandwidth usage.

Log events from the web server provide metric data for CloudWatch alarms. The two metric filters describe how the log information is transformed into CloudWatch metrics. The 404 metric counts the number of 404 occurrences. The size metric tracks the size of a request. The two CloudWatch alarms will send notifications if there are more than two 404s within 2 minutes or if the average request size is over 3500 KB over 10 minutes.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Sample template that sets up and configures CloudWatch Logs on
 Amazon Linux 2023 instance.",
    "Parameters": {
        "KeyName": {
            "Description": "Name of an existing EC2 KeyPair to enable SSH access to the
 instances",
            "Type": "AWS::EC2::KeyPair::KeyName",
            "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
        },
        "SSHLocation": {
            "Description": "The IP address range that can be used to SSH to the EC2
 instances",
            "Type": "String",
            "MinLength": "9",
            "MaxLength": "18",
            "Default": "0.0.0.0/0",
            "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\
\d{1,2})",
            "ConstraintDescription": "must be a valid IP CIDR range of the form
 x.x.x.x/x."
        },
        "OperatorEmail": {
            "Description": "Email address to notify when CloudWatch alarms are
 triggered (404 errors or high bandwidth usage)",
            "Type": "String"
        }
    },
    "Resources": {
        "LogRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
```

```
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ec2.amazonaws.com"
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ]
                    }
                ]
            },
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "LogRolePolicy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "logs:PutLogEvents",
                                    "logs:DescribeLogStreams",
                                    "logs:DescribeLogGroups",
                                    "logs:CreateLogGroup",
                                    "logs:CreateLogStream"
                                ],
                                "Resource": "*"
                            }
                        ]
                    }
                }
            ]
        }
    },
    "LogRoleInstanceProfile": {
        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [{"Ref": "LogRole"}]
        }
```

```
        },
        "WebServerSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Enable HTTP access via port 80 and SSH access via
 port 22",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "CidrIp": "0.0.0.0/0"
                    },
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 22,
                        "ToPort": 22,
                        "CidrIp": {"Ref": "SSHLocation"}
                    }
                ]
            }
        },
        "WebServerHost": {
            "Type": "AWS::EC2::Instance",
            "Metadata": {
                "Comment": "Install a simple PHP application on Amazon Linux 2023",
                "AWS::CloudFormation::Init": {
                    "config": {
                        "packages": {
                            "dnf": {
                                "httpd": [],
                                "php": [],
                                "php-fpm": []
                            }
                        },
                        "files": {
                            "/etc/amazon-cloudwatch-agent/amazon-cloudwatch-
agent.json": {
                                "content": {
                                    "logs": {
                                        "logs_collected": {
                                            "files": {
                                                "collect_list": [{
```

```
                                                    "file_path": "/var/log/httpd/
access_log",
                                                    "log_group_name": {"Ref":
 "WebServerLogGroup"},
                                                    "log_stream_name": "{instance_id}/
apache.log",
                                                    "timestamp_format": "%d/%b/%Y:%H:
%M:%S %z"
                                                }]
                                            }
                                        }
                                    }
                                },
                                "mode": "000644",
                                "owner": "root",
                                "group": "root"
                            },
                            "/var/www/html/index.php": {
                                "content": "<?php\necho '<h1>AWS CloudFormation sample
 PHP application on Amazon Linux 2023</h1>';\n?>\n",
                                "mode": "000644",
                                "owner": "apache",
                                "group": "apache"
                            },
                            "/etc/cfn/cfn-hup.conf": {
                                "content":  {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "[main]\n",
                                            "stack=",
                                            {"Ref": "AWS::StackId"},
                                            "\n",
                                            "region=",
                                            {"Ref": "AWS::Region"},
                                            "\n"
                                        ]
                                    ]
                                },
                                "mode": "000400",
                                "owner": "root",
                                "group": "root"
                            },
                            "/etc/cfn/hooks.d/cfn-auto-reloader.conf": {
```

```
                                "content": {
                                    "Fn::Join": [
                                        "",
                                        [
                                            "[cfn-auto-reloader-hook]\n",
                                            "triggers=post.update\n",

 "path=Resources.WebServerHost.Metadata.AWS::CloudFormation::Init\n",
                                            "action=/opt/aws/bin/cfn-init -s ",
                                            {"Ref": "AWS::StackId"},
                                            " -r WebServerHost ",
                                            " --region     ",
                                            {"Ref": "AWS::Region"},
                                            "\n",
                                            "runas=root\n"
                                        ]
                                    ]
                                }
                            }
                        },
                        "services": {
                            "systemd": {
                                "httpd": {
                                    "enabled": "true",
                                    "ensureRunning": "true"
                                },
                                "php-fpm": {
                                    "enabled": "true",
                                    "ensureRunning": "true"
                                }
                            }
                        }
                    }
                }
            },
            "CreationPolicy": {
                "ResourceSignal": {
                    "Timeout": "PT5M"
                }
            },
            "Properties": {
                "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-
ami-kernel-6.1-x86_64}}",
                "KeyName": {"Ref": "KeyName"},
```

```
                "InstanceType": "t3.micro",
                "SecurityGroupIds": [{"Ref": "WebServerSecurityGroup"}],
                "IamInstanceProfile": {"Ref": "LogRoleInstanceProfile"},
                "UserData": {"Fn::Base64": {"Fn::Join": [ "", [
                    "#!/bin/bash\n",
                    "dnf update -y aws-cfn-bootstrap\n",
                    "dnf install -y amazon-cloudwatch-agent\n",
                    "/opt/aws/bin/cfn-init -v --stack ", {"Ref": "AWS::StackName"}, "
 --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "\n",
                    "\n",
                    "# Verify Apache log directory exists and create if needed\n",
                    "mkdir -p /var/log/httpd\n",
                    "\n",
                    "# Start CloudWatch agent\n",
                    "/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -
a fetch-config -m ec2 -c file:/etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json
 -s\n",
                    "\n",
                    "# Signal success\n",
                    "/opt/aws/bin/cfn-signal -e $? --stack ", {"Ref":
 "AWS::StackName"}, " --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "\n"
                ]]}}
            }
        },
        "WebServerLogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "DeletionPolicy": "Retain",
            "UpdateReplacePolicy": "Retain",
            "Properties": {
                "RetentionInDays": 7
            }
        },
        "404MetricFilter": {
            "Type": "AWS::Logs::MetricFilter",
            "Properties": {
                "LogGroupName": {"Ref": "WebServerLogGroup"},
                "FilterPattern": "[ip, identity, user_id, timestamp, request,
 status_code = 404, size, ...]",
                "MetricTransformations": [
                    {
                        "MetricValue": "1",
                        "MetricNamespace": "test/404s",
                        "MetricName": "test404Count"
                    }
```

```
                ]
            }
        },
        "BytesTransferredMetricFilter": {
            "Type": "AWS::Logs::MetricFilter",
            "Properties": {
                "LogGroupName": {"Ref": "WebServerLogGroup"},
                "FilterPattern": "[ip, identity, user_id, timestamp, request,
status_code, size, ...]",
                "MetricTransformations": [
                    {
                        "MetricValue": "$size",
                        "MetricNamespace": "test/BytesTransferred",
                        "MetricName": "testBytesTransferred"
                    }
                ]
            }
        },
        "404Alarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": "The number of 404s is greater than 2 over 2
minutes",
                "MetricName": "test404Count",
                "Namespace": "test/404s",
                "Statistic": "Sum",
                "Period": "60",
                "EvaluationPeriods": "2",
                "Threshold": "2",
                "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
                "ComparisonOperator": "GreaterThanThreshold"
            }
        },
        "BandwidthAlarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": "The average volume of traffic is greater 3500 KB
over 10 minutes",
                "MetricName": "testBytesTransferred",
                "Namespace": "test/BytesTransferred",
                "Statistic": "Average",
                "Period": "300",
                "EvaluationPeriods": "2",
                "Threshold": "3500",
```

```
                "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
                "ComparisonOperator": "GreaterThanThreshold"
            }
        },
        "AlarmNotificationTopic": {
            "Type": "AWS::SNS::Topic",
            "Properties": {
                "Subscription": [{"Endpoint": {"Ref": "OperatorEmail"}, "Protocol":
 "email"}]
            }
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The instance ID of the web server",
            "Value": {"Ref": "WebServerHost"}
        },
        "WebsiteURL": {
            "Value": {"Fn::Sub": "http://${WebServerHost.PublicDnsName}"},
            "Description": "URL for the web server"
        },
        "PublicIP": {
            "Description": "Public IP address of the web server",
            "Value": {"Fn::GetAtt": ["WebServerHost","PublicIp"]
            }
        },
        "CloudWatchLogGroupName": {
            "Description": "The name of the CloudWatch log group",
            "Value": {"Ref": "WebServerLogGroup"}
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Sample template that sets up and configures CloudWatch Logs on Amazon
 Linux 2023 instance.
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
```

```
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  OperatorEmail:
    Description: Email address to notify when CloudWatch alarms are triggered (404
 errors or high bandwidth usage)
    Type: String
Resources:
  LogRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      Policies:
        - PolicyName: LogRolePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'logs:PutLogEvents'
                  - 'logs:DescribeLogStreams'
                  - 'logs:DescribeLogGroups'
                  - 'logs:CreateLogGroup'
                  - 'logs:CreateLogStream'
                Resource: '*'
  LogRoleInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles:
```

```
      - !Ref LogRole
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80 and SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref SSHLocation
  WebServerHost:
    Type: AWS::EC2::Instance
    Metadata:
      Comment: Install a simple PHP application on Amazon Linux 2023
      'AWS::CloudFormation::Init':
        config:
          packages:
            dnf:
              httpd: []
              php: []
              php-fpm: []
          files:
            /etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json:
              content: !Sub |
                {
                  "logs": {
                    "logs_collected": {
                      "files": {
                        "collect_list": [
                          {
                            "file_path": "/var/log/httpd/access_log",
                            "log_group_name": "${WebServerLogGroup}",
                            "log_stream_name": "{instance_id}/apache.log",
                            "timestamp_format": "%d/%b/%Y:%H:%M:%S %z"
                          }
                        ]
                      }
                    }
                  }
                }
```

```
              mode: '000644'
              owner: root
              group: root
            /var/www/html/index.php:
              content: |
                <?php echo '<h1>AWS CloudFormation sample PHP application on Amazon
 Linux 2023</h1>';
                ?>
              mode: '000644'
              owner: apache
              group: apache
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
              content: !Sub |
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.WebServerHost.Metadata.AWS::CloudFormation::Init
                action=/opt/aws/bin/cfn-init -s ${AWS::StackId} -r WebServerHost --
region ${AWS::Region}
                runas=root
          services:
            systemd:
              httpd:
                enabled: 'true'
                ensureRunning: 'true'
              php-fpm:
                enabled: 'true'
                ensureRunning: 'true'
    CreationPolicy:
      ResourceSignal:
        Timeout: PT5M
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-
kernel-6.1-x86_64}}'
      KeyName: !Ref KeyName
      InstanceType: t3.micro
      SecurityGroupIds:
```

```
        - !Ref WebServerSecurityGroup
      IamInstanceProfile: !Ref LogRoleInstanceProfile
      UserData: !Base64
        Fn::Sub: |
          #!/bin/bash
          dnf update -y aws-cfn-bootstrap
          dnf install -y amazon-cloudwatch-agent
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource WebServerHost
 --region ${AWS::Region}

          # Verify Apache log directory exists and create if needed
          mkdir -p /var/log/httpd

          # Start CloudWatch agent
          /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -c file:/etc/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json -s

          # Signal success
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 WebServerHost --region ${AWS::Region}
          echo "Done"
  WebServerLogGroup:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
    Properties:
      RetentionInDays: 7
  404MetricFilter:
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref WebServerLogGroup
      FilterPattern: >-
        [ip, identity, user_id, timestamp, request, status_code = 404, size, ...]
      MetricTransformations:
        - MetricValue: '1'
          MetricNamespace: test/404s
          MetricName: test404Count
  BytesTransferredMetricFilter:
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref WebServerLogGroup
      FilterPattern: '[ip, identity, user_id, timestamp, request, status_code,
 size, ...]'
      MetricTransformations:
```

```
        - MetricValue: $size
          MetricNamespace: test/BytesTransferred
          MetricName: testBytesTransferred
  404Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: The number of 404s is greater than 2 over 2 minutes
      MetricName: test404Count
      Namespace: test/404s
      Statistic: Sum
      Period: '60'
      EvaluationPeriods: '2'
      Threshold: '2'
      AlarmActions:
        - !Ref AlarmNotificationTopic
      ComparisonOperator: GreaterThanThreshold
  BandwidthAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: The average volume of traffic is greater 3500 KB over 10
 minutes
      MetricName: testBytesTransferred
      Namespace: test/BytesTransferred
      Statistic: Average
      Period: '300'
      EvaluationPeriods: '2'
      Threshold: '3500'
      AlarmActions:
        - !Ref AlarmNotificationTopic
      ComparisonOperator: GreaterThanThreshold
  AlarmNotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !Ref OperatorEmail
          Protocol: email
Outputs:
  InstanceId:
    Description: The instance ID of the web server
    Value: !Ref WebServerHost
  WebsiteURL:
    Value: !Sub 'http://${WebServerHost.PublicDnsName}'
    Description: URL for the web server
  PublicIP:
```

```
    Description: Public IP address of the web server
    Value: !GetAtt WebServerHost.PublicIp
  CloudWatchLogGroupName:
    Description: The name of the CloudWatch log group
    Value: !Ref WebServerLogGroup
```

Send logs to CloudWatch Logs from a Windows instance

The following template configures CloudWatch Logs for a Windows 2012R2 instance.

The CloudWatch Logs agent on Windows (SSM agent on Windows 2012R2 and Windows 2016 AMIs) only sends logs after it's started, so any logs that are generated before startup aren't sent. To work around this, the template helps to ensure that the agent starts before any logs are written by:

- Configuring the agent setup as the first `config` item in cfn-init `configSets`.
- Using `waitAfterCompletion` to insert a pause after the command that starts the agent.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Sample template that sets up and configures CloudWatch Logs on
 Windows 2012R2 instance.",
    "Parameters": {
        "KeyPair": {
            "Description": "Name of an existing EC2 KeyPair to enable RDP access to the
 instances",
            "Type": "AWS::EC2::KeyPair::KeyName",
            "ConstraintDescription": "must be the name of an existing EC2 KeyPair."
        },
        "RDPLocation": {
            "Description": "The IP address range that can be used to RDP to the EC2
 instances",
            "Type": "String",
            "MinLength": "9",
            "MaxLength": "18",
            "Default": "0.0.0.0/0",
            "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\
\d{1,2})",
            "ConstraintDescription": "must be a valid IP CIDR range of the form
 x.x.x.x/x."
        },
```

```
        "OperatorEmail": {
            "Description": "Email address to notify when CloudWatch alarms are
triggered (404 errors)",
            "Type": "String"
        }
    },
    "Resources": {
        "WebServerSecurityGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Enable HTTP access via port 80 and RDP access via
port 3389",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": "80",
                        "ToPort": "80",
                        "CidrIp": "0.0.0.0/0"
                    },
                    {
                        "IpProtocol": "tcp",
                        "FromPort": "3389",
                        "ToPort": "3389",
                        "CidrIp": {"Ref": "RDPLocation"}
                    }
                ]
            }
        },
        "LogRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ec2.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ]
```

```
                    }
                ]
            },
            "ManagedPolicyArns": [
                "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
            ],
            "Path": "/",
            "Policies": [
                {
                    "PolicyName": "LogRolePolicy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "logs:Create*",
                                    "logs:PutLogEvents",
                                    "s3:GetObject"
                                ],
                                "Resource": [
                                    "arn:aws:logs:*:*:*",
                                    "arn:aws:s3:::*"
                                ]
                            }
                        ]
                    }
                }
            ]
        }
    },
    "LogRoleInstanceProfile": {
        "Type": "AWS::IAM::InstanceProfile",
        "Properties": {
            "Path": "/",
            "Roles": [{"Ref": "LogRole"}]
        }
    },
    "WebServerHost": {
        "Type": "AWS::EC2::Instance",
        "CreationPolicy": {
            "ResourceSignal": {
                "Timeout": "PT15M"
            }
```

```
            },
            "Metadata": {
                "AWS::CloudFormation::Init": {
                    "configSets": {
                        "config": [
                            "00-ConfigureCWLogs",
                            "01-InstallWebServer",
                            "02-ConfigureApplication",
                            "03-Finalize"
                        ]
                    },
                    "00-ConfigureCWLogs": {
                        "files": {
                            "C:\\Program Files\\Amazon\\SSM\\Plugins\\awsCloudWatch\
\AWS.EC2.Windows.CloudWatch.json": {
                                "content": {
                                    "EngineConfiguration": {
                                        "Components": [
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "ApplicationEventLog",
                                                "Parameters": {
                                                    "Levels": "7",
                                                    "LogName": "Application"
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "SystemEventLog",
                                                "Parameters": {
                                                    "Levels": "7",
                                                    "LogName": "System"
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "SecurityEventLog",
                                                "Parameters": {
                                                    "Levels": "7",
                                                    "LogName": "Security"
                                                }
```

```
                                                },
                                                {
                                                    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                    "Id": "EC2ConfigLog",
                                                    "Parameters": {
                                                        "CultureName": "en-US",
                                                        "Encoding": "ASCII",
                                                        "Filter": "EC2ConfigLog.txt",
                                                        "LogDirectoryPath": "C:\\Program
 Files\\Amazon\\Ec2ConfigService\\Logs",
                                                        "TimeZoneKind": "UTC",
                                                        "TimestampFormat": "yyyy-MM-
ddTHH:mm:ss.fffZ:"
                                                    }
                                                },
                                                {
                                                    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                    "Id": "CfnInitLog",
                                                    "Parameters": {
                                                        "CultureName": "en-US",
                                                        "Encoding": "ASCII",
                                                        "Filter": "cfn-init.log",
                                                        "LogDirectoryPath": "C:\\cfn\\log",
                                                        "TimeZoneKind": "Local",
                                                        "TimestampFormat": "yyyy-MM-dd
 HH:mm:ss,fff"
                                                    }
                                                },
                                                {
                                                    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                                    "Id": "IISLogs",
                                                    "Parameters": {
                                                        "CultureName": "en-US",
                                                        "Encoding": "UTF-8",
                                                        "Filter": "",
                                                        "LineCount": "3",
                                                        "LogDirectoryPath": "C:\\inetpub\
\logs\\LogFiles\\W3SVC1",
                                                        "TimeZoneKind": "UTC",
                                                        "TimestampFormat": "yyyy-MM-dd
 HH:mm:ss"
```

```
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent,AWS.E
                                                "Id": "MemoryPerformanceCounter",
                                                "Parameters": {
                                                    "CategoryName": "Memory",
                                                    "CounterName": "Available MBytes",
                                                    "DimensionName": "",
                                                    "DimensionValue": "",
                                                    "InstanceName": "",
                                                    "MetricName": "Memory",
                                                    "Unit": "Megabytes"
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchApplicationEventLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
ApplicationEventLog",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchSystemEventLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
SystemEventLog",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
```

```
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchSecurityEventLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
SecurityEventLog",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchEC2ConfigLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
EC2ConfigLog",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchCfnInitLog",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
CfnInitLog",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatchIISLogs",
                                                "Parameters": {
                                                    "AccessKey": "",
```

```
                                                    "LogGroup": {"Ref": "LogGroup"},
                                                    "LogStream": "{instance_id}/
IISLogs",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            },
                                            {
                                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch",
                                                "Id": "CloudWatch",
                                                "Parameters": {
                                                    "AccessKey": "",
                                                    "NameSpace": "Windows/Default",
                                                    "Region": {"Ref": "AWS::Region"},
                                                    "SecretKey": ""
                                                }
                                            }
                                        ],
                                        "Flows": {
                                            "Flows": [

 "ApplicationEventLog,CloudWatchApplicationEventLog",

 "SystemEventLog,CloudWatchSystemEventLog",

 "SecurityEventLog,CloudWatchSecurityEventLog",
                                                "EC2ConfigLog,CloudWatchEC2ConfigLog",
                                                "CfnInitLog,CloudWatchCfnInitLog",
                                                "IISLogs,CloudWatchIISLogs",
                                                "MemoryPerformanceCounter,CloudWatch"
                                            ]
                                        },
                                        "PollInterval": "00:00:05"
                                    },
                                    "IsEnabled": true
                                }
                            }
                        },
                        "commands": {
                            "0-enableSSM": {
                                "command": "powershell.exe -Command \"Set-Service -Name
 AmazonSSMAgent -StartupType Automatic\" ",
                                "waitAfterCompletion": "0"
```

```
                            },
                            "1-restartSSM": {
                                "command": "powershell.exe -Command \"Restart-Service
 AmazonSSMAgent \"",
                                "waitAfterCompletion": "30"
                            }
                        }
                    },
                    "01-InstallWebServer": {
                        "commands": {
                            "01_install_webserver": {
                                "command": "powershell.exe -Command \"Install-
WindowsFeature Web-Server  -IncludeAllSubFeature\"",
                                "waitAfterCompletion": "0"
                            }
                        }
                    },
                    "02-ConfigureApplication": {
                        "files": {
                            "c:\\Inetpub\\wwwroot\\index.htm": {
                                "content": "<html> <head> <title>Test Application
 Page</title> </head> <body> <h1>Congratulations!! Your IIS server is configured.</h1>
 </body> </html>"
                            }
                        }
                    },
                    "03-Finalize": {
                        "commands": {
                            "00_signal_success": {
                                "command": {
                                    "Fn::Sub": "cfn-signal.exe -e 0 --resource
 WebServerHost --stack ${AWS::StackName} --region ${AWS::Region}"
                                },
                                "waitAfterCompletion": "0"
                            }
                        }
                    }
                }
            },
            "Properties": {
                "KeyName": {
                    "Ref": "KeyPair"
                },
```

```
                "ImageId": "{{resolve:ssm:/aws/service/ami-windows-latest/
Windows_Server-2012-R2_RTM-English-64Bit-Base}}",
                "InstanceType": "t2.xlarge",
                "SecurityGroupIds": [{"Ref": "WebServerSecurityGroup"}],
                "IamInstanceProfile": {"Ref": "LogRoleInstanceProfile"},
                "UserData": {
                    "Fn::Base64": {
                        "Fn::Join": [
                            "",
                            [
                                "<script>\n",
                                "wmic product where \"description='Amazon SSM Agent' \"
 uninstall\n",
                                "wmic product where \"description='aws-cfn-bootstrap'
 \" uninstall \n",
                                "start /wait c:\\Windows\\system32\\msiexec /passive /
qn /i https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-win64-
latest.msi\n",
                                "powershell.exe -Command \"iwr https://
s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/
AmazonSSMAgentSetup.exe  -UseBasicParsing -OutFile C:\\AmazonSSMAgentSetup.exe\"\n",
                                "start /wait C:\\AmazonSSMAgentSetup.exe /install /
quiet\n",
                                "cfn-init.exe -v -c config -s ", {"Ref":
 "AWS::StackName"}, " --resource WebServerHost --region ", {"Ref": "AWS::Region"}, "
 \n",
                                "</script>\n"
                            ]
                        ]
                    }
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "Properties": {
                "RetentionInDays": 7
            }
        },
        "404MetricFilter": {
            "Type": "AWS::Logs::MetricFilter",
            "Properties": {
                "LogGroupName": {"Ref": "LogGroup"},
```

```
                "FilterPattern": "[timestamps, serverip, method, uri, query, port,
dash, clientip, useragent, status_code = 404, ...]",
                "MetricTransformations": [
                    {
                        "MetricValue": "1",
                        "MetricNamespace": "test/404s",
                        "MetricName": "test404Count"
                    }
                ]
            }
        },
        "404Alarm": {
            "Type": "AWS::CloudWatch::Alarm",
            "Properties": {
                "AlarmDescription": "The number of 404s is greater than 2 over 2
minutes",
                "MetricName": "test404Count",
                "Namespace": "test/404s",
                "Statistic": "Sum",
                "Period": "60",
                "EvaluationPeriods": "2",
                "Threshold": "2",
                "AlarmActions": [{"Ref": "AlarmNotificationTopic"}],
                "ComparisonOperator": "GreaterThanThreshold"
            }
        },
        "AlarmNotificationTopic": {
            "Type": "AWS::SNS::Topic",
            "Properties": {
                "Subscription": [{"Endpoint": {"Ref": "OperatorEmail"}, "Protocol":
"email"}]
            }
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The instance ID of the web server",
            "Value": {"Ref": "WebServerHost"}
        },
        "WebsiteURL": {
            "Value": {"Fn::Sub": "http://${WebServerHost.PublicDnsName}"},
            "Description": "URL for the web server"
        },
        "PublicIP": {
```

```
            "Description": "Public IP address of the web server",
            "Value": {"Fn::GetAtt": ["WebServerHost","PublicIp"]}
        },
        "CloudWatchLogGroupName": {
            "Description": "The name of the CloudWatch log group",
            "Value": {"Ref": "LogGroup"}
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: >-
  Sample template that sets up and configures CloudWatch Logs on Windows 2012R2
 instance.
Parameters:
  KeyPair:
    Description: Name of an existing EC2 KeyPair to enable RDP access to the instances
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  RDPLocation:
    Description: The IP address range that can be used to RDP to the EC2 instances
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  OperatorEmail:
    Description: Email address to notify when CloudWatch alarms are triggered (404
 errors)
    Type: String
Resources:
  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP access via port 80 and RDP access via port 3389
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '80'
          ToPort: '80'
          CidrIp: 0.0.0.0/0
```

```
      - IpProtocol: tcp
        FromPort: '3389'
        ToPort: '3389'
        CidrIp: !Ref RDPLocation
  LogRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
      Path: /
      Policies:
        - PolicyName: LogRolePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'logs:Create*'
                  - 'logs:PutLogEvents'
                  - 's3:GetObject'
                Resource:
                  - 'arn:aws:logs:*:*:*'
                  - 'arn:aws:s3:::*'
  LogRoleInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles:
        - !Ref LogRole
  WebServerHost:
    Type: AWS::EC2::Instance
    CreationPolicy:
      ResourceSignal:
        Timeout: PT15M
    Metadata:
```

```
      'AWS::CloudFormation::Init':
        configSets:
          config:
            - 00-ConfigureCWLogs
            - 01-InstallWebServer
            - 02-ConfigureApplication
            - 03-Finalize
        00-ConfigureCWLogs:
          files:
            'C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch
\AWS.EC2.Windows.CloudWatch.json':
              content: !Sub |
                {
                  "EngineConfiguration": {
                      "Components": [
                          {
                              "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                              "Id": "ApplicationEventLog",
                              "Parameters": {
                                  "Levels": "7",
                                  "LogName": "Application"
                              }
                          },
                          {
                              "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                              "Id": "SystemEventLog",
                              "Parameters": {
                                  "Levels": "7",
                                  "LogName": "System"
                              }
                          },
                          {
                              "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
                              "Id": "SecurityEventLog",
                              "Parameters": {
                                  "Levels": "7",
                                  "LogName": "Security"
                              }
                          },
                          {
```

```
                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                "Id": "EC2ConfigLog",
                                "Parameters": {
                                    "CultureName": "en-US",
                                    "Encoding": "ASCII",
                                    "Filter": "EC2ConfigLog.txt",
                                    "LogDirectoryPath": "C:\\Program Files\\Amazon\
\Ec2ConfigService\\Logs",
                                    "TimeZoneKind": "UTC",
                                    "TimestampFormat": "yyyy-MM-ddTHH:mm:ss.fffZ:"
                                }
                            },
                            {
                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                "Id": "CfnInitLog",
                                "Parameters": {
                                    "CultureName": "en-US",
                                    "Encoding": "ASCII",
                                    "Filter": "cfn-init.log",
                                    "LogDirectoryPath": "C:\\cfn\\log",
                                    "TimeZoneKind": "Local",
                                    "TimestampFormat": "yyyy-MM-dd HH:mm:ss,fff"
                                }
                            },
                            {
                                "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
                                "Id": "IISLogs",
                                "Parameters": {
                                    "CultureName": "en-US",
                                    "Encoding": "UTF-8",
                                    "Filter": "",
                                    "LineCount": "3",
                                    "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\
\W3SVC1",
                                    "TimeZoneKind": "UTC",
                                    "TimestampFormat": "yyyy-MM-dd HH:mm:ss"
                                }
                            },
                            {
                                "FullName":
 "AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent,AWS.E
```

```
                            "Id": "MemoryPerformanceCounter",
                            "Parameters": {
                                "CategoryName": "Memory",
                                "CounterName": "Available MBytes",
                                "DimensionName": "",
                                "DimensionValue": "",
                                "InstanceName": "",
                                "MetricName": "Memory",
                                "Unit": "Megabytes"
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchApplicationEventLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/ApplicationEventLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchSystemEventLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/SystemEventLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchSecurityEventLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/SecurityEventLog",
                                "Region": "${AWS::Region}",
```

```
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchEC2ConfigLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/EC2ConfigLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchCfnInitLog",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/CfnInitLog",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatchIISLogs",
                            "Parameters": {
                                "AccessKey": "",
                                "LogGroup": "${LogGroup}",
                                "LogStream": "{instance_id}/IISLogs",
                                "Region": "${AWS::Region}",
                                "SecretKey": ""
                            }
                        },
                        {
                            "FullName":
"AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent,AWS.EC2.Windows.CloudWatch",
                            "Id": "CloudWatch",
                            "Parameters": {
```

```
                            "AccessKey": "",
                            "NameSpace": "Windows/Default",
                            "Region": "${AWS::Region}",
                            "SecretKey": ""
                        }
                    }
                ],
                "Flows": {
                    "Flows": [
                        "ApplicationEventLog,CloudWatchApplicationEventLog",
                        "SystemEventLog,CloudWatchSystemEventLog",
                        "SecurityEventLog,CloudWatchSecurityEventLog",
                        "EC2ConfigLog,CloudWatchEC2ConfigLog",
                        "CfnInitLog,CloudWatchCfnInitLog",
                        "IISLogs,CloudWatchIISLogs",
                        "MemoryPerformanceCounter,CloudWatch"
                    ]
                },
                "PollInterval": "00:00:05"
            },
            "IsEnabled": true
          }
    commands:
      0-enableSSM:
        command: >-
          powershell.exe -Command "Set-Service -Name AmazonSSMAgent
          -StartupType Automatic"
        waitAfterCompletion: '0'
      1-restartSSM:
        command: powershell.exe -Command "Restart-Service AmazonSSMAgent "
        waitAfterCompletion: '30'
  01-InstallWebServer:
    commands:
      01_install_webserver:
        command: >-
          powershell.exe -Command "Install-WindowsFeature Web-Server
          -IncludeAllSubFeature"
        waitAfterCompletion: '0'
  02-ConfigureApplication:
    files:
      'c:\Inetpub\wwwroot\index.htm':
        content: >-
          <html> <head> <title>Test Application Page</title> </head>
          <body> <h1>Congratulations !! Your IIS server is
```

```
                configured.</h1> </body> </html>
        03-Finalize:
          commands:
            00_signal_success:
              command: !Sub >-
                cfn-signal.exe -e 0 --resource WebServerHost --stack
                ${AWS::StackName} --region ${AWS::Region}
              waitAfterCompletion: '0'
    Properties:
      KeyName: !Ref KeyPair
      ImageId: "{{resolve:ssm:/aws/service/ami-windows-latest/Windows_Server-2012-
R2_RTM-English-64Bit-Base}}"
      InstanceType: t2.xlarge
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      IamInstanceProfile: !Ref LogRoleInstanceProfile
      UserData: !Base64
        'Fn::Sub': >
          <script>

          wmic product where "description='Amazon SSM Agent' " uninstall

          wmic product where "description='aws-cfn-bootstrap' " uninstall

          start /wait c:\\Windows\\system32\\msiexec /passive /qn /i
          https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-win64-
latest.msi

          powershell.exe -Command "iwr
          https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/
AmazonSSMAgentSetup.exe
          -UseBasicParsing -OutFile C:\\AmazonSSMAgentSetup.exe"

          start /wait C:\\AmazonSSMAgentSetup.exe /install /quiet

          cfn-init.exe -v -c config -s ${AWS::StackName} --resource
          WebServerHost --region ${AWS::Region}

          </script>
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      RetentionInDays: 7
  404MetricFilter:
```

```
    Type: AWS::Logs::MetricFilter
    Properties:
      LogGroupName: !Ref LogGroup
      FilterPattern: >-
        [timestamps, serverip, method, uri, query, port, dash, clientip,
        useragent, status_code = 404, ...]
      MetricTransformations:
        - MetricValue: '1'
          MetricNamespace: test/404s
          MetricName: test404Count
  404Alarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmDescription: The number of 404s is greater than 2 over 2 minutes
      MetricName: test404Count
      Namespace: test/404s
      Statistic: Sum
      Period: '60'
      EvaluationPeriods: '2'
      Threshold: '2'
      AlarmActions:
        - !Ref AlarmNotificationTopic
      ComparisonOperator: GreaterThanThreshold
  AlarmNotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Endpoint: !Ref OperatorEmail
          Protocol: email
Outputs:
  InstanceId:
    Description: The instance ID of the web server
    Value: !Ref WebServerHost
  WebsiteURL:
    Value: !Sub 'http://${WebServerHost.PublicDnsName}'
    Description: URL for the web server
  PublicIP:
    Description: Public IP address of the web server
    Value: !GetAtt
      - WebServerHost
      - PublicIp
  CloudWatchLogGroupName:
    Description: The name of the CloudWatch log group
```

```
    Value: !Ref LogGroup
```

See also

For more information about CloudWatch Logs resources, see AWS::Logs::LogGroup or AWS::Logs::MetricFilter.

Amazon DynamoDB template snippets

Topics

- Application Auto Scaling with an Amazon DynamoDB table
- See also

Application Auto Scaling with an Amazon DynamoDB table

This example sets up Application Auto Scaling for a `AWS::DynamoDB::Table` resource. The template defines a `TargetTrackingScaling` scaling policy that scales up the `WriteCapacityUnits` throughput for the table.

JSON

```
{
    "Resources": {
        "DDBTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "AttributeDefinitions": [
                    {
                        "AttributeName": "ArtistId",
                        "AttributeType": "S"
                    },
                    {
                        "AttributeName": "Concert",
                        "AttributeType": "S"
                    },
                    {
                        "AttributeName": "TicketSales",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
```

```
                {
                    "AttributeName": "ArtistId",
                    "KeyType": "HASH"
                },
                {
                    "AttributeName": "Concert",
                    "KeyType": "RANGE"
                }
            ],
            "GlobalSecondaryIndexes": [
                {
                    "IndexName": "GSI",
                    "KeySchema": [
                        {
                            "AttributeName": "TicketSales",
                            "KeyType": "HASH"
                        }
                    ],
                    "Projection": {
                        "ProjectionType": "KEYS_ONLY"
                    },
                    "ProvisionedThroughput": {
                        "ReadCapacityUnits": 5,
                        "WriteCapacityUnits": 5
                    }
                }
            ],
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 5,
                "WriteCapacityUnits": 5
            }
        }
    },
    "WriteCapacityScalableTarget": {
        "Type": "AWS::ApplicationAutoScaling::ScalableTarget",
        "Properties": {
            "MaxCapacity": 15,
            "MinCapacity": 5,
            "ResourceId": {
                "Fn::Join": [
                    "/",
                    [
                        "table",
                        {
```

```
                                "Ref": "DDBTable"
                            }
                        ]
                    ]
                },
                "RoleARN" : { "Fn::Sub" : "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable" },
                "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
                "ServiceNamespace": "dynamodb"
            }
        },
        "WriteScalingPolicy": {
            "Type": "AWS::ApplicationAutoScaling::ScalingPolicy",
            "Properties": {
                "PolicyName": "WriteAutoScalingPolicy",
                "PolicyType": "TargetTrackingScaling",
                "ScalingTargetId": {
                    "Ref": "WriteCapacityScalableTarget"
                },
                "TargetTrackingScalingPolicyConfiguration": {
                    "TargetValue": 50,
                    "ScaleInCooldown": 60,
                    "ScaleOutCooldown": 60,
                    "PredefinedMetricSpecification": {
                        "PredefinedMetricType": "DynamoDBWriteCapacityUtilization"
                    }
                }
            }
        }
    }
}
```

YAML

```
Resources:
  DDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: "ArtistId"
          AttributeType: "S"
        - AttributeName: "Concert"
```

```
          AttributeType: "S"
        - AttributeName: "TicketSales"
          AttributeType: "S"
      KeySchema:
        - AttributeName: "ArtistId"
          KeyType: "HASH"
        - AttributeName: "Concert"
          KeyType: "RANGE"
      GlobalSecondaryIndexes:
        - IndexName: "GSI"
          KeySchema:
            - AttributeName: "TicketSales"
              KeyType: "HASH"
          Projection:
            ProjectionType: "KEYS_ONLY"
          ProvisionedThroughput:
            ReadCapacityUnits: 5
            WriteCapacityUnits: 5
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
  WriteCapacityScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    Properties:
      MaxCapacity: 15
      MinCapacity: 5
      ResourceId: !Join
        - /
        - - table
          - !Ref DDBTable
      RoleARN:
        Fn::Sub: 'arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable'
      ScalableDimension: dynamodb:table:WriteCapacityUnits
      ServiceNamespace: dynamodb
  WriteScalingPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    Properties:
      PolicyName: WriteAutoScalingPolicy
      PolicyType: TargetTrackingScaling
      ScalingTargetId: !Ref WriteCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 50.0
```

```
        ScaleInCooldown: 60
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: DynamoDBWriteCapacityUtilization
```

See also

For more information, see the blog post How to use CloudFormation to configure auto scaling for DynamoDB tables and indexes on the AWS Database Blog.

For more information about DynamoDB resources, see AWS::DynamoDB::Table.

Amazon EC2 CloudFormation template snippets

Amazon EC2 provides scalable compute capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. These virtual servers, known as instances, can run a variety of operating systems and applications, and can be customized to meet your specific requirements. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in usage.

You can define and provision Amazon EC2 instances as part of your infrastructure using CloudFormation templates. Templates make it easy to manage and automate the deployment of Amazon EC2 resources in a repeatable and consistent manner.

The following example template snippets describe CloudFormation resources or components for Amazon EC2. These snippets are designed to be integrated into a template and are not intended to be run independently.

Snippet categories

- Configure Amazon EC2 instances with CloudFormation
- Create launch templates with CloudFormation
- Manage security groups with CloudFormation
- Allocate and associate Elastic IP addresses with CloudFormation
- Configure Amazon VPC resources with CloudFormation

Configure Amazon EC2 instances with CloudFormation

The following snippets demonstrate how to configure Amazon EC2 instances using CloudFormation.

Snippet categories

- General Amazon EC2 configurations
- Specify the block device mappings for an instance

General Amazon EC2 configurations

The following snippets demonstrate general configurations for Amazon EC2 instances using CloudFormation.

Example snippets

- Create an Amazon EC2 instance in a specified Availability Zone
- Configure a tagged Amazon EC2 instance with an EBS volume and user data
- Define DynamoDB table name in user data for Amazon EC2 instance launch
- Create an Amazon EBS volume with DeletionPolicy

Create an Amazon EC2 instance in a specified Availability Zone

The following snippet creates an Amazon EC2 instance in the specified Availability Zone using an AWS::EC2::Instance resource. The code for an Availability Zone is its Region code followed by a letter identifier. You can launch an instance into a single Availability Zone.

JSON

```
"Ec2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "AvailabilityZone": "aa-example-1a",
        "ImageId": "ami-1234567890abcdef0"
    }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
```

Configure a tagged Amazon EC2 instance with an EBS volume and user data

The following snippet creates an Amazon EC2 instance with a tag, an EBS volume, and user data. It uses an AWS::EC2::Instance resource. In the same template, you must define an AWS::EC2::SecurityGroup resource, an AWS::SNS::Topic resource, and an AWS::EC2::Volume resource. The `KeyName` must be defined in the `Parameters` section of the template.

Tags can help you to categorize AWS resources based on your preferences, such as by purpose, owner, or environment. User data allows for the provisioning of custom scripts or data to an instance during launch. This data facilitates task automation, software configuration, package installation, and other actions on an instance during initialization.

For more information about tagging your resources, see Tag your Amazon EC2 resources in the *Amazon EC2 User Guide*.

For information about user data, see Use instance metadata to manage your EC2 instance in the *Amazon EC2 User Guide*.

JSON

```
"Ec2Instance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "KeyName": { "Ref": "KeyName" },
    "SecurityGroups": [ { "Ref": "Ec2SecurityGroup" } ],
    "UserData": {
      "Fn::Base64": {
        "Fn::Join": [ ":", [
            "PORT=80",
            "TOPIC=",
            { "Ref": "MySNSTopic" }
          ]
        ]
      }
    },
    "InstanceType": "aa.size",
    "AvailabilityZone": "aa-example-1a",
    "ImageId": "ami-1234567890abcdef0",
    "Volumes": [
      {
        "VolumeId": { "Ref": "MyVolumeResource" },
        "Device": "/dev/sdk"
      }
```

```
    ],
    "Tags": [ { "Key": "Name", "Value": "MyTag" } ]
  }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    KeyName: !Ref KeyName
    SecurityGroups:
      - !Ref Ec2SecurityGroup
    UserData:
      Fn::Base64:
        Fn::Join:
          - ":"
          - - "PORT=80"
            - "TOPIC="
            - !Ref MySNSTopic
    InstanceType: aa.size
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
    Volumes:
      - VolumeId: !Ref MyVolumeResource
        Device: "/dev/sdk"
    Tags:
      - Key: Name
        Value: MyTag
```

Define DynamoDB table name in user data for Amazon EC2 instance launch

The following snippet creates an Amazon EC2 instance and defines a DynamoDB table name in the user data to pass to the instance at launch. It uses an AWS::EC2::Instance resource. You can define parameters or dynamic values in the user data to pass an EC2 instance at launch.

For more information about user data, see Use instance metadata to manage your EC2 instance in the *Amazon EC2 User Guide*.

JSON

```
"Ec2Instance": {
```

```
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "UserData": {
            "Fn::Base64": {
                "Fn::Join": [
                    "",
                    [
                        "TableName=",
                        {
                            "Ref": "DynamoDBTableName"
                        }
                    ]
                ]
            }
        },
        "AvailabilityZone": "aa-example-1a",
        "ImageId": "ami-1234567890abcdef0"
    }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    UserData:
      Fn::Base64:
        Fn::Join:
          - ''
          - - 'TableName='
            - Ref: DynamoDBTableName
    AvailabilityZone: aa-example-1a
    ImageId: ami-1234567890abcdef0
```

Create an Amazon EBS volume with `DeletionPolicy`

The following snippets create an Amazon EBS volume using an Amazon EC2 AWS::EC2::Volume resource. You can use the `Size` or `SnapshotID` properties to define the volume, but not both. A `DeletionPolicy` attribute is set to create a snapshot of the volume when the stack is deleted.

For more information about the `DeletionPolicy` attribute, see DeletionPolicy attribute.

For more information about creating Amazon EBS volumes, see Create an Amazon EBS volume.

JSON

This snippet creates an Amazon EBS volume with a specified **size**. The size is set to 10, but you can adjust it as needed. The AWS::EC2::Volume resource allows you to specify either the size or a snapshot ID but not both.

```
"MyEBSVolume": {
    "Type": "AWS::EC2::Volume",
    "Properties": {
        "Size": "10",
        "AvailabilityZone": {
            "Ref": "AvailabilityZone"
        }
    },
    "DeletionPolicy": "Snapshot"
}
```

This snippet creates an Amazon EBS volume using a provided **snapshot ID**. The AWS::EC2::Volume resource allows you to specify either the size or a snapshot ID but not both.

```
"MyEBSVolume": {
    "Type": "AWS::EC2::Volume",
    "Properties": {
        "SnapshotId" : "snap-1234567890abcdef0",
        "AvailabilityZone": {
            "Ref": "AvailabilityZone"
        }
    },
    "DeletionPolicy": "Snapshot"
}
```

YAML

This snippet creates an Amazon EBS volume with a specified **size**. The size is set to 10, but you can adjust it as needed. The AWS::EC2::Volume resource allows you to specify either the size or a snapshot ID but not both.

```
MyEBSVolume:
  Type: AWS::EC2::Volume
  Properties:
    Size: 10
```

```
    AvailabilityZone:
      Ref: AvailabilityZone
  DeletionPolicy: Snapshot
```

This snippet creates an Amazon EBS volume using a provided **snapshot ID**. The AWS::EC2::Volume resource allows you to specify either the size or a snapshot ID but not both.

```
MyEBSVolume:
  Type: AWS::EC2::Volume
  Properties:
    SnapshotId: snap-1234567890abcdef0
    AvailabilityZone:
      Ref: AvailabilityZone
  DeletionPolicy: Snapshot
```

Specify the block device mappings for an instance

A block device mapping defines the block devices, which includes instance store volumes and EBS volumes, to attach to an instance. You can specify a block device mapping when creating an AMI so that the mapping is used by all instances launched from the AMI. Alternatively, you can specify a block device mapping when you launch an instance, so that the mapping overrides the one specified in the AMI from which the instance was launched.

You can use the following template snippets to specify the block device mappings for your EBS or instance store volumes using the `BlockDeviceMappings` property of an AWS::EC2::Instance resource.

For more information about block device mappings, see Block device mappings for volumes on Amazon EC2 instances in the *Amazon EC2 User Guide*.

Scenarios

- Specify the block device mappings for two EBS volumes
- Specify the block device mapping for an instance store volume

Specify the block device mappings for two EBS volumes

JSON

```
"Ec2Instance": {
```

```
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
      "KeyName": { "Ref": "KeyName" },
      "InstanceType": { "Ref": "InstanceType" },
      "SecurityGroups": [{ "Ref": "Ec2SecurityGroup" }],
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": { "VolumeSize": "50" }
        },
        {
          "DeviceName": "/dev/sdm",
          "Ebs": { "VolumeSize": "100" }
        }
      ]
    }
  }
}
```

YAML

```
EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      KeyName: !Ref KeyName
      InstanceType: !Ref InstanceType
      SecurityGroups:
        - !Ref Ec2SecurityGroup
      BlockDeviceMappings:
        -
          DeviceName: /dev/sda1
          Ebs:
            VolumeSize: 50
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeSize: 100
```

Specify the block device mapping for an instance store volume

JSON

```
"Ec2Instance" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "ImageId" : "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
    "KeyName" : { "Ref" : "KeyName" },
    "InstanceType": { "Ref": "InstanceType" },
    "SecurityGroups" : [{ "Ref" : "Ec2SecurityGroup" }],
    "BlockDeviceMappings" : [
      {
        "DeviceName"  : "/dev/sdc",
        "VirtualName" : "ephemeral0"
      }
    ]
  }
}
```

YAML

```
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-
gp2}}'
    KeyName: !Ref KeyName
    InstanceType: !Ref InstanceType
    SecurityGroups:
      - !Ref Ec2SecurityGroup
    BlockDeviceMappings:
      - DeviceName: /dev/sdc
        VirtualName: ephemeral0
```

Create launch templates with CloudFormation

This section provides an example for creating an Amazon EC2 launch template using CloudFormation. Launch templates allow you to create templates for configuring and provisioning Amazon EC2 instances within AWS. With launch templates, you can store launch parameters so

that you do not have to specify them every time you launch an instance. For more examples, see the Examples section in the `AWS::EC2::LaunchTemplate` resource.

For more information about launch templates, see Store instance launch parameters in Amazon EC2 launch templates in the *Amazon EC2 User Guide*.

For information about creating launch templates for use with Auto Scaling groups, see Auto Scaling launch templates in the *Amazon EC2 Auto Scaling User Guide*.

Snippet categories

- Create a launch template that specifies security groups, tags, user data, and an IAM role

Create a launch template that specifies security groups, tags, user data, and an IAM role

This snippet shows an AWS::EC2::LaunchTemplate resource that contains the configuration information to launch an instance. You specify values for the `ImageId`, `InstanceType`, `SecurityGroups`, `UserData`, and `TagSpecifications` properties. The `SecurityGroups` property specifies an existing EC2 security group and a new security group. The `Ref` function gets the ID of the AWS::EC2::SecurityGroup resource `myNewEC2SecurityGroup` that's declared elsewhere in the stack template.

The launch template includes a section for custom user data. You can pass in configuration tasks and scripts that run when an instance launches in this section. In this example, the user data installs the AWS Systems Manager Agent and starts the agent.

The launch template also includes an IAM role that allows applications running on instances to perform actions on your behalf. This example shows an AWS::IAM::Role resource for the launch template, which uses the `IamInstanceProfile` property to specify the IAM role. The `Ref` function gets the name of the AWS::IAM::InstanceProfile resource `myInstanceProfile`. To configure the permissions of the IAM role, you specify a value for the `ManagedPolicyArns` property.

JSON

```
{
  "Resources":{
    "myLaunchTemplate":{
      "Type":"AWS::EC2::LaunchTemplate",
      "Properties":{
        "LaunchTemplateName":{ "Fn::Sub": "${AWS::StackName}-launch-template" },
```

```
        "LaunchTemplateData":{
          "ImageId":"ami-02354e95b3example",
          "InstanceType":"t3.micro",
          "IamInstanceProfile":{
            "Name":{
              "Ref":"myInstanceProfile"
            }
          },
          "SecurityGroupIds":[
            {
              "Ref":"myNewEC2SecurityGroup"
            },
            "sg-083cd3bfb8example"
          ],
          "UserData":{
            "Fn::Base64":{
              "Fn::Join": [
                "", [
                  "#!/bin/bash\n",
                  "cd /tmp\n",
                  "yum install -y https://s3.amazonaws.com/ec2-downloads-windows/
SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm\n",
                  "systemctl enable amazon-ssm-agent\n",
                  "systemctl start amazon-ssm-agent\n"
                ]
              ]
            }
          },
          "TagSpecifications":[
            {
              "ResourceType":"instance",
              "Tags":[
                {
                  "Key":"environment",
                  "Value":"development"
                }
              ]
            },
            {
              "ResourceType":"volume",
              "Tags":[
                {
                  "Key":"environment",
                  "Value":"development"
```

```
              }
            ]
          }
        ]
      }
    }
  },
  "myInstanceRole":{
    "Type":"AWS::IAM::Role",
    "Properties":{
      "RoleName":"InstanceRole",
      "AssumeRolePolicyDocument":{
        "Version": "2012-10-17",
        "Statement":[
          {
            "Effect":"Allow",
            "Principal":{
              "Service":[
                "ec2.amazonaws.com"
              ]
            },
            "Action":[
              "sts:AssumeRole"
            ]
          }
        ]
      },
      "ManagedPolicyArns":[
        "arn:aws:iam::aws:policy/myCustomerManagedPolicy"
      ]
    }
  },
  "myInstanceProfile":{
    "Type":"AWS::IAM::InstanceProfile",
    "Properties":{
      "Path":"/",
      "Roles":[
        {
          "Ref":"myInstanceRole"
        }
      ]
    }
  }
}
```

```
}
```

YAML

```
---
Resources:
  myLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: ami-02354e95b3example
        InstanceType: t3.micro
        IamInstanceProfile:
          Name: !Ref myInstanceProfile
        SecurityGroupIds:
        - !Ref myNewEC2SecurityGroup
        - sg-083cd3bfb8example
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            cd /tmp
            yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/
latest/linux_amd64/amazon-ssm-agent.rpm
            systemctl enable amazon-ssm-agent
            systemctl start amazon-ssm-agent
        TagSpecifications:
        - ResourceType: instance
          Tags:
          - Key: environment
            Value: development
        - ResourceType: volume
          Tags:
          - Key: environment
            Value: development
  myInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: InstanceRole
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: 'Allow'
```

```
          Principal:
            Service:
            - 'ec2.amazonaws.com'
          Action:
          - 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/myCustomerManagedPolicy'
  myInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: '/'
      Roles:
      - !Ref myInstanceRole
```

Manage security groups with CloudFormation

The following snippets demonstrate how to use CloudFormation to manage security groups and Amazon EC2 instances to control access to your AWS resources.

Snippet categories

- Associate an Amazon EC2 instance with a security group
- Create security groups with ingress rules
- Create an Elastic Load Balancer with a security group ingress rule

Associate an Amazon EC2 instance with a security group

The following example snippets demonstrate how to associate an Amazon EC2 instance with a default Amazon VPC security group using CloudFormation.

Example snippets

- Associate an Amazon EC2 instance with a default VPC security group
- Create an Amazon EC2 instance with an attached volume and security group

Associate an Amazon EC2 instance with a default VPC security group

The following snippet creates an Amazon VPC, a subnet within the VPC, and an Amazon EC2 instance. The VPC is created using an AWS::EC2::VPC resource. The IP address range for the VPC is defined in the larger template and is referenced by the `MyVPCCIDRRange` parameter.

A subnet is created within the VPC using an AWS::EC2:: Subnet resource. The subnet is associated with the VPC, which is referenced as `MyVPC`.

An EC2 instance is launched within the VPC and subnet using an AWS::EC2::Instance resource. This resource specifies the Amazon Machine Image (AMI) to use to launch the instance, the subnet where the instance will run, and the security group to associate with the instance. The `ImageId` uses a Systems Manager parameter to dynamically retrieve the latest Amazon Linux 2 AMI.

The security group ID is obtained using the `Fn::GetAtt` function, which retrieves the default security group from the `MyVPC` resource.

The instance is placed within the `MySubnet` resource defined in the snippet.

When you create a VPC using CloudFormation, AWS automatically creates default resources within the VPC, including a default security group. However, when you define a VPC within a CloudFormation template, you may not have access to the IDs of these default resources when you create the template. To access and use the default resources specified in the template, you can use intrinsic functions such as `Fn::GetAtt`. This function allows you to work with the default resources that are automatically created by CloudFormation.

JSON

```
"MyVPC": {
    "Type": "AWS::EC2::VPC",
    "Properties": {
        "CidrBlock": {
            "Ref": "MyVPCCIDRRange"
        },
        "EnableDnsSupport": false,
        "EnableDnsHostnames": false,
        "InstanceTenancy": "default"
    }
},
"MySubnet": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
        "CidrBlock": {
            "Ref": "MyVPCCIDRRange"
        },
        "VpcId": {
            "Ref": "MyVPC"
        }
```

```
    }
},
"MyInstance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}",
        "SecurityGroupIds": [
            {
                "Fn::GetAtt": [
                    "MyVPC",
                    "DefaultSecurityGroup"
                ]
            }
        ],
        "SubnetId": {
            "Ref": "MySubnet"
        }
    }
}
```

YAML

```
MyVPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock:
      Ref: MyVPCCIDRRange
    EnableDnsSupport: false
    EnableDnsHostnames: false
    InstanceTenancy: default
MySubnet:
  Type: AWS::EC2::Subnet
  Properties:
    CidrBlock:
      Ref: MyVPCCIDRRange
    VpcId:
      Ref: MyVPC
MyInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-
gp2}}'
```

```
    SecurityGroupIds:
      - Fn::GetAtt:
          - MyVPC
          - DefaultSecurityGroup
    SubnetId:
      Ref: MySubnet
```

Create an Amazon EC2 instance with an attached volume and security group

The following snippet creates an Amazon EC2 instance using an AWS::EC2::Instance resource, which is launched from a designated AMI . The instance is associated with a security group that allows incoming SSH traffic on port 22 from a specified IP address, using an AWS::EC2::SecurityGroup resource . It creates a 100 GB Amazon EBS volume using an AWS::EC2::Volume resource. The volume is created in the same availability zone as the instance, as specified by the `GetAtt` function, and is mounted to the instance at the `/dev/sdh` device.

For more information about creating Amazon EBS volumes, see Create an Amazon EBS volume.

JSON

```
"Ec2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "SecurityGroups": [
            {
                "Ref": "InstanceSecurityGroup"
            }
        ],
        "ImageId": "ami-1234567890abcdef0"
    }
},
"InstanceSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Enable SSH access via port 22",
        "SecurityGroupIngress": [
            {
                "IpProtocol": "tcp",
                "FromPort": "22",
                "ToPort": "22",
                "CidrIp": "192.0.2.0/24"
            }
        ]
```

```
    }
},
"NewVolume": {
    "Type": "AWS::EC2::Volume",
    "Properties": {
        "Size": "100",
        "AvailabilityZone": {
            "Fn::GetAtt": [
                "Ec2Instance",
                "AvailabilityZone"
            ]
        }
    }
},
"MountPoint": {
    "Type": "AWS::EC2::VolumeAttachment",
    "Properties": {
        "InstanceId": {
            "Ref": "Ec2Instance"
        },
        "VolumeId": {
            "Ref": "NewVolume"
        },
        "Device": "/dev/sdh"
    }
}
```

YAML

```
Ec2Instance:
  Type: AWS::EC2::Instance
  Properties:
    SecurityGroups:
      - !Ref InstanceSecurityGroup
    ImageId: ami-1234567890abcdef0
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable SSH access via port 22
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
```

```
        CidrIp: 192.0.2.0/24
NewVolume:
  Type: AWS::EC2::Volume
  Properties:
    Size: 100
    AvailabilityZone: !GetAtt [Ec2Instance, AvailabilityZone]
MountPoint:
  Type: AWS::EC2::VolumeAttachment
  Properties:
    InstanceId: !Ref Ec2Instance
    VolumeId: !Ref NewVolume
    Device: /dev/sdh
```

Create security groups with ingress rules

The following example snippets demonstrate how to configure security groups with specific ingress rules using CloudFormation.

Snippets

- Create security group with ingress rules for SSH and HTTP access
- Create a security group with ingress rules for HTTP and SSH access from specified CIDR ranges
- Create cross-referencing security groups with ingress rules

Create security group with ingress rules for SSH and HTTP access

The following snippet describes two security group ingress rules using an AWS::EC2::SecurityGroup resource. The first ingress rule allows SSH (port 22) access from an existing security group named `MyAdminSecurityGroup`, which is owned by the AWS account with the account number `1111-2222-3333`. The second ingress rule allows HTTP (port 80) access from a different security group named `MySecurityGroupCreatedInCFN`, which is created in the same template. The `Ref` function is used to reference the logical name of the security group created in the same template.

In the first ingress rule, you must add a value for both the `SourceSecurityGroupName` and `SourceSecurityGroupOwnerId` properties. In the second ingress rule, `MySecurityGroupCreatedInCFNTemplate` references a different security group, which is created in the same template. Verify that the logical name `MySecurityGroupCreatedInCFNTemplate` matches the actual logical name of the security group resource that you specify in the larger template.

For more information about security groups, see Amazon EC2 security groups for your Amazon EC2 instances in the *Amazon EC2 User Guide*.

JSON

```
"SecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Allow connections from specified source security group",
        "SecurityGroupIngress": [
            {
                "IpProtocol": "tcp",
                "FromPort": "22",
                "ToPort": "22",
                "SourceSecurityGroupName": "MyAdminSecurityGroup",
                "SourceSecurityGroupOwnerId": "1111-2222-3333"
            },
            {
                "IpProtocol": "tcp",
                "FromPort": "80",
                "ToPort": "80",
                "SourceSecurityGroupName": {
                    "Ref": "MySecurityGroupCreatedInCFNTemplate"
                }
            }
        ]
    }
}
```

YAML

```
SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow connections from specified source security group
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: '22'
        ToPort: '22'
        SourceSecurityGroupName: MyAdminSecurityGroup
        SourceSecurityGroupOwnerId: '1111-2222-3333'
      - IpProtocol: tcp
        FromPort: '80'
```

```
        ToPort: '80'
        SourceSecurityGroupName:
          Ref: MySecurityGroupCreatedInCFNTemplate
```

Create a security group with ingress rules for HTTP and SSH access from specified CIDR ranges

The following snippet creates a security group for an Amazon EC2 instance with two inbound rules. The inbound rules allow incoming TCP traffic on the specified ports from the designated CIDR ranges. An AWS::EC2::SecurityGroup resource is used to specify the rules. You must specify a protocol for each rule. For TCP, you must specify a port or port range. If you do not specify either a source security group or a CIDR range, the stack will launch successfully, but the rule will not be applied to the security group.

For more information about security groups, see Amazon EC2 security groups for your Amazon EC2 instances in the *Amazon EC2 User Guide*.

JSON

```
"ServerSecurityGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription": "Allow connections from specified CIDR ranges",
    "SecurityGroupIngress": [
      {
        "IpProtocol": "tcp",
        "FromPort": "80",
        "ToPort": "80",
        "CidrIp": "192.0.2.0/24"
      },
      {
        "IpProtocol": "tcp",
        "FromPort": "22",
        "ToPort": "22",
        "CidrIp": "192.0.2.0/24"
      }
    ]
  }
}
```

YAML

```
ServerSecurityGroup:
```

```
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Allow connections from specified CIDR ranges
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 192.0.2.0/24
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 192.0.2.0/24
```

Create cross-referencing security groups with ingress rules

The following snippet uses the AWS::EC2::SecurityGroup resource to create two Amazon EC2 security groups, SGroup1 and SGroup2. Ingress rules that allow communication between the two security groups are created by using the AWS::EC2::SecurityGroupIngress resource. SGroup1Ingress establishes an ingress rule for SGroup1 that allows incoming TCP traffic on port 80 from the source security group, SGroup2. SGroup2Ingress establishes an ingress rule for SGroup2 that allows incoming TCP traffic on port 80 from the source security group, SGroup1.

JSON

```
"SGroup1": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "EC2 instance access"
    }
},
"SGroup2": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "EC2 instance access"
    }
},
"SGroup1Ingress": {
    "Type": "AWS::EC2::SecurityGroupIngress",
    "Properties": {
        "GroupName": {
            "Ref": "SGroup1"
        },
        "IpProtocol": "tcp",
```

```
        "ToPort": "80",
        "FromPort": "80",
        "SourceSecurityGroupName": {
            "Ref": "SGroup2"
        }
    }
},
"SGroup2Ingress": {
    "Type": "AWS::EC2::SecurityGroupIngress",
    "Properties": {
        "GroupName": {
            "Ref": "SGroup2"
        },
        "IpProtocol": "tcp",
        "ToPort": "80",
        "FromPort": "80",
        "SourceSecurityGroupName": {
            "Ref": "SGroup1"
        }
    }
}
```

YAML

```
SGroup1:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: EC2 Instance access
SGroup2:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: EC2 Instance access
SGroup1Ingress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupName: !Ref SGroup1
    IpProtocol: tcp
    ToPort: 80
    FromPort: 80
    SourceSecurityGroupName: !Ref SGroup2
SGroup2Ingress:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
```

```
    GroupName: !Ref SGroup2
    IpProtocol: tcp
    ToPort: 80
    FromPort: 80
    SourceSecurityGroupName: !Ref SGroup1
```

Create an Elastic Load Balancer with a security group ingress rule

The following template creates an AWS::ElasticLoadBalancing::LoadBalancer resource in the specified availability zone. The AWS::ElasticLoadBalancing::LoadBalancer resource is configured to listen on port 80 for HTTP traffic and direct requests to instances also on port 80. The Elastic Load Balancer is responsible for load balancing incoming HTTP traffic among the instances.

Additionally, this template generates an AWS::EC2::SecurityGroup resource associated with the load balancer. This security group is created with a single ingress rule, described as `ELB ingress group`, which allows incoming TCP traffic on port 80. The source for this ingress rule is defined using the `Fn::GetAtt` fucntion to retrieve attributes from the load balancer resource. `SourceSecurityGroupOwnerId` uses `Fn::GetAtt` to obtain the `OwnerAlias` of the source security group of the load balancer. `SourceSecurityGroupName` uses `Fn::Getatt` to obtain the `GroupName` of the source security group of the ELB.

This setup ensures secure communication between the ELB and the instances.

For more information about load balancing, see the Elastic Load Balancing User Guide.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MyELB": {
            "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
            "Properties": {
                "AvailabilityZones": [
                    "aa-example-1a"
                ],
                "Listeners": [
                    {
                        "LoadBalancerPort": "80",
                        "InstancePort": "80",
                        "Protocol": "HTTP"
                    }
```

```
                ]
            }
        },
        "MyELBIngressGroup": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "ELB ingress group",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "SourceSecurityGroupOwnerId": {
                            "Fn::GetAtt": [
                                "MyELB",
                                "SourceSecurityGroup.OwnerAlias"
                            ]
                        },
                        "SourceSecurityGroupName": {
                            "Fn::GetAtt": [
                                "MyELB",
                                "SourceSecurityGroup.GroupName"
                            ]
                        }
                    }
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyELB:
    Type: AWS::ElasticLoadBalancing::LoadBalancer
    Properties:
      AvailabilityZones:
        - aa-example-1a
      Listeners:
        - LoadBalancerPort: '80'
          InstancePort: '80'
```

```
        Protocol: HTTP
  MyELBIngressGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: ELB ingress group
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '80'
          ToPort: '80'
          SourceSecurityGroupOwnerId:
            Fn::GetAtt:
              - MyELB
              - SourceSecurityGroup.OwnerAlias
          SourceSecurityGroupName:
            Fn::GetAtt:
              - MyELB
              - SourceSecurityGroup.GroupName
```

Allocate and associate Elastic IP addresses with CloudFormation

The following template snippets are examples related to Elastic IP addresses (EIPs) in Amazon EC2. These examples cover allocation, association, and management of EIPs for your instances.

Example snippets

- Allocate an Elastic IP address and associate it with an Amazon EC2 instance
- Associate an Elastic IP address to an Amazon EC2 instance by specifying the IP address
- Associate an Elastic IP address to an Amazon EC2 instance by specifying the allocation ID of the IP address

Allocate an Elastic IP address and associate it with an Amazon EC2 instance

The following snippet allocates an Amazon EC2 Elastic IP (EIP) address and associates it with an Amazon EC2 instance using an AWS::EC2::EIP resource. You can allocate an EIP address from an address pool owned by AWS or from an address pool created from a public IPv4 address range you bring to AWS for use with your AWS resources using bring your own IP addresses (BYOIP). In this example, the EIP is allocated from an address pool owned by AWS.

For more information about Elastic IP addresses, see Elastic IP address in the *Amazon EC2 User Guide*.

JSON

```
"ElasticIP": {
    "Type": "AWS::EC2::EIP",
    "Properties": {
        "InstanceId": {
            "Ref": "Ec2Instance"
        }
    }
}
```

YAML

```
ElasticIP:
  Type: AWS::EC2::EIP
  Properties:
    InstanceId: !Ref EC2Instance
```

Associate an Elastic IP address to an Amazon EC2 instance by specifying the IP address

The following snippet associates an existing Amazon EC2 Elastic IP address to an EC2 instance using an AWS::EC2::EIPAssociation resource. You must first allocate an Elastic IP address for use in your account. An Elastic IP address can be associated with a single instance.

JSON

```
"IPAssoc": {
  "Type": "AWS::EC2::EIPAssociation",
  "Properties": {
    "InstanceId": {
      "Ref": "Ec2Instance"
    },
    "EIP": "192.0.2.0"
  }
}
```

YAML

```
IPAssoc:
  Type: AWS::EC2::EIPAssociation
  Properties:
```

```
    InstanceId: !Ref EC2Instance
    EIP: 192.0.2.0
```

Associate an Elastic IP address to an Amazon EC2 instance by specifying the allocation ID of the IP address

The following snippet associates an existing Elastic IP address to an Amazon EC2 instance by specifying the allocation ID using an AWS::EC2::EIPAssociation resource. An allocation ID is assigned to an Elastic IP address upon Elastic IP address allocation.

JSON

```
"IPAssoc": {
    "Type": "AWS::EC2::EIPAssociation",
    "Properties": {
        "InstanceId": {
            "Ref": "Ec2Instance"
        },
        "AllocationId": "eipalloc-1234567890abcdef0"
    }
}
```

YAML

```
IPAssoc:
  Type: AWS::EC2::EIPAssociation
  Properties:
    InstanceId: !Ref EC2Instance
    AllocationId: eipalloc-1234567890abcdef0
```

Configure Amazon VPC resources with CloudFormation

This section provides examples for configuring Amazon VPC resources using CloudFormation. VPCs allow you to create a virtual network within AWS, and these snippets show how to configure aspects of VPCs to meet your networking requirements.

Example snippets

- Enable IPv6 egress-only internet access in a VPC
- Elastic network interface (ENI) template snippets

Enable IPv6 egress-only internet access in a VPC

An egress-only internet gateway allows instances within a VPC to access the internet and prevent resources on the internet from communicating with the instances. The following snippet enables IPv6 egress-only internet access from within a VPC. It creates a VPC with an IPv4 address range of `10.0.0/16` using an AWS::EC2::VPC resource. A route table is associated with this VPC resource using an AWS::EC2::RouteTable resource. The route table manages routes for instances within the VPC. An AWS::EC2::EgressOnlyInternetGateway is used to create an egress-only internet gateway to enable IPv6 communication for outbound traffic from instances within the VPC, while preventing inbound traffic. An AWS::EC2::Route resource is specified to create an IPv6 route in the route table that directs all outbound IPv6 traffic (`::/0`) to the egress-only internet gateway.

For more information about egress-only internet gateways, see Enable outbound IPv6 traffic using an egress-only internet gateway in the *Amazon VPC User Guide*.

JSON

```
"DefaultIpv6Route": {
    "Type": "AWS::EC2::Route",
    "Properties": {
        "DestinationIpv6CidrBlock": "::/0",
        "EgressOnlyInternetGatewayId": {
            "Ref": "EgressOnlyInternetGateway"
        },
        "RouteTableId": {
            "Ref": "RouteTable"
        }
    }
},
"EgressOnlyInternetGateway": {
    "Type": "AWS::EC2::EgressOnlyInternetGateway",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
        }
    }
},
"RouteTable": {
    "Type": "AWS::EC2::RouteTable",
    "Properties": {
        "VpcId": {
            "Ref": "VPC"
```

```
        }
    }
},
"VPC": {
    "Type": "AWS::EC2::VPC",
    "Properties": {
        "CidrBlock": "10.0.0.0/16"
    }
}
```

YAML

```
DefaultIpv6Route:
  Type: AWS::EC2::Route
  Properties:
    DestinationIpv6CidrBlock: "::/0"
    EgressOnlyInternetGatewayId:
      Ref: "EgressOnlyInternetGateway"
    RouteTableId:
      Ref: "RouteTable"
EgressOnlyInternetGateway:
  Type: AWS::EC2::EgressOnlyInternetGateway
  Properties:
    VpcId:
      Ref: "VPC"
RouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId:
      Ref: "VPC"
VPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: "10.0.0.0/16"
```

Elastic network interface (ENI) template snippets

Create an Amazon EC2 instance with attached elastic network interfaces (ENIs)

The following example snippet creates an Amazon EC2 instance using an AWS::EC2::Instance resource in the specified Amazon VPC and subnet. It attaches two network interfaces (ENIs) with the instance, associates Elastic IP addresses to the instances through the attached ENIs, and configures the security group for SSH and HTTP access. User data is supplied to the instance as

part of the launch configuration when the instance is created. The user data includes a script encoded in `base64` format to ensure it is passed to the instance. When the instance is launched, the script runs automatically as part of the bootstrapping process. It installs `ec2-net-utils`, configures the network interfaces, and starts the HTTP service.

To determine the appropriate Amazon Machine Image (AMI) based on the selected Region, the snippet uses a `Fn::FindInMap` function that looks up values in a `RegionMap` mapping. This mapping must be defined in the larger template. The two network interfaces are created using AWS::EC2::NetworkInterface resources. Elastic IP addresses are specified using AWS::EC2::EIP resources allocated to the `vpc` domain. These elastic IP addresses are associated with the network interfaces using AWS::EC2::EIPAssociation resources.

The `Outputs` section defines values or resources that you want to access after the stack is created. In this snippet, the defined output is `InstancePublicIp`, which represents the public IP address of the EC2 instance created by the stack. You can retrieve this output in the **Output** tab on the CloudFormation console, or using the describe-stacks command.

For more information about elastic network interfaces, see Elastic network interfaces.

JSON

```
"Resources": {
    "ControlPortAddress": {
        "Type": "AWS::EC2::EIP",
        "Properties": {
            "Domain": "vpc"
        }
    },
    "AssociateControlPort": {
        "Type": "AWS::EC2::EIPAssociation",
        "Properties": {
            "AllocationId": {
                "Fn::GetAtt": [
                    "ControlPortAddress",
                    "AllocationId"
                ]
            },
            "NetworkInterfaceId": {
                "Ref": "controlXface"
            }
        }
    },
```

```
    "WebPortAddress": {
        "Type": "AWS::EC2::EIP",
        "Properties": {
            "Domain": "vpc"
        }
    },
    "AssociateWebPort": {
        "Type": "AWS::EC2::EIPAssociation",
        "Properties": {
            "AllocationId": {
                "Fn::GetAtt": [
                    "WebPortAddress",
                    "AllocationId"
                ]
            },
            "NetworkInterfaceId": {
                "Ref": "webXface"
            }
        }
    },
    "SSHSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "VpcId": {
                "Ref": "VpcId"
            },
            "GroupDescription": "Enable SSH access via port 22",
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 22,
                    "IpProtocol": "tcp",
                    "ToPort": 22
                }
            ]
        }
    },
    "WebSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "VpcId": {
                "Ref": "VpcId"
            },
            "GroupDescription": "Enable HTTP access via user-defined port",
```

```
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "FromPort": 80,
                    "IpProtocol": "tcp",
                    "ToPort": 80
                }
            ]
        }
    },
    "controlXface": {
        "Type": "AWS::EC2::NetworkInterface",
        "Properties": {
            "SubnetId": {
                "Ref": "SubnetId"
            },
            "Description": "Interface for controlling traffic such as SSH",
            "GroupSet": [
                {
                    "Fn::GetAtt": [
                        "SSHSecurityGroup",
                        "GroupId"
                    ]
                }
            ],
            "SourceDestCheck": true,
            "Tags": [
                {
                    "Key": "Network",
                    "Value": "Control"
                }
            ]
        }
    },
    "webXface": {
        "Type": "AWS::EC2::NetworkInterface",
        "Properties": {
            "SubnetId": {
                "Ref": "SubnetId"
            },
            "Description": "Interface for web traffic",
            "GroupSet": [
                {
                    "Fn::GetAtt": [
```

```
                    "WebSecurityGroup",
                    "GroupId"
                ]
            }
        ],
        "SourceDestCheck": true,
        "Tags": [
            {
                "Key": "Network",
                "Value": "Web"
            }
        ]
    }
},
"Ec2Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "ImageId": {
            "Fn::FindInMap": [
                "RegionMap",
                {
                    "Ref": "AWS::Region"
                },
                "AMI"
            ]
        },
        "KeyName": {
            "Ref": "KeyName"
        },
        "NetworkInterfaces": [
            {
                "NetworkInterfaceId": {
                    "Ref": "controlXface"
                },
                "DeviceIndex": "0"
            },
            {
                "NetworkInterfaceId": {
                    "Ref": "webXface"
                },
                "DeviceIndex": "1"
            }
        ],
        "Tags": [
```

```
                {
                    "Key": "Role",
                    "Value": "Test Instance"
                }
            ],
            "UserData": {
                "Fn::Base64": {
                    "Fn::Sub": "#!/bin/bash -xe\nyum install ec2-net-utils -y\nec2ifup
 eth1\nservice httpd start\n"
                }
            }
        }
    }
},
"Outputs": {
    "InstancePublicIp": {
        "Description": "Public IP Address of the EC2 Instance",
        "Value": {
            "Fn::GetAtt": [
                "Ec2Instance",
                "PublicIp"
            ]
        }
    }
}
```

YAML

```
Resources:
  ControlPortAddress:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
  AssociateControlPort:
    Type: AWS::EC2::EIPAssociation
    Properties:
      AllocationId:
        Fn::GetAtt:
          - ControlPortAddress
          - AllocationId
      NetworkInterfaceId:
        Ref: controlXface
  WebPortAddress:
```

```
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
  AssociateWebPort:
    Type: AWS::EC2::EIPAssociation
    Properties:
      AllocationId:
        Fn::GetAtt:
          - WebPortAddress
          - AllocationId
      NetworkInterfaceId:
        Ref: webXface
  SSHSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
        Ref: VpcId
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          FromPort: 22
          IpProtocol: tcp
          ToPort: 22
  WebSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
        Ref: VpcId
      GroupDescription: Enable HTTP access via user-defined port
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          FromPort: 80
          IpProtocol: tcp
          ToPort: 80
  controlXface:
    Type: AWS::EC2::NetworkInterface
    Properties:
      SubnetId:
        Ref: SubnetId
      Description: Interface for controlling traffic such as SSH
      GroupSet:
        - Fn::GetAtt:
            - SSHSecurityGroup
            - GroupId
```

```
      SourceDestCheck: true
      Tags:
        - Key: Network
          Value: Control
  webXface:
    Type: AWS::EC2::NetworkInterface
    Properties:
      SubnetId:
        Ref: SubnetId
      Description: Interface for web traffic
      GroupSet:
        - Fn::GetAtt:
            - WebSecurityGroup
            - GroupId
      SourceDestCheck: true
      Tags:
        - Key: Network
          Value: Web
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId:
        Fn::FindInMap:
          - RegionMap
          - Ref: AWS::Region
          - AMI
      KeyName:
        Ref: KeyName
      NetworkInterfaces:
        - NetworkInterfaceId:
            Ref: controlXface
          DeviceIndex: "0"
        - NetworkInterfaceId:
            Ref: webXface
          DeviceIndex: "1"
      Tags:
        - Key: Role
          Value: Test Instance
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum install ec2-net-utils -y
          ec2ifup eth1
          service httpd start
```

```
Outputs:
  InstancePublicIp:
    Description: Public IP Address of the EC2 Instance
    Value:
      Fn::GetAtt:
        - Ec2Instance
        - PublicIp
```

Amazon Elastic Container Service sample templates

Amazon Elastic Container Service (Amazon ECS) is a container management service that makes it easy to run, stop, and manage Docker containers on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances.

Create a cluster with the AL2023 Amazon ECS-Optimized-AMI

Define a cluster that uses a capacity provider that launches AL2023 instances on Amazon EC2.

> **Important**
>
> For the latest AMI IDs, see Amazon ECS-optimized AMI in the *Amazon Elastic Container Service Developer Guide*.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "EC2 ECS cluster that starts out empty, with no EC2 instances yet. An
 ECS capacity provider automatically launches more EC2 instances as required on the fly
 when you request ECS to launch services or standalone tasks.",
  "Parameters": {
      "InstanceType": {
          "Type": "String",
          "Description": "EC2 instance type",
          "Default": "t2.medium",
          "AllowedValues": [
              "t1.micro",
              "t2.2xlarge",
              "t2.large",
              "t2.medium",
              "t2.micro",
```

```
                "t2.nano",
                "t2.small",
                "t2.xlarge",
                "t3.2xlarge",
                "t3.large",
                "t3.medium",
                "t3.micro",
                "t3.nano",
                "t3.small",
                "t3.xlarge"
            ]
        },
        "DesiredCapacity": {
            "Type": "Number",
            "Default": "0",
            "Description": "Number of EC2 instances to launch in your ECS cluster."
        },
        "MaxSize": {
            "Type": "Number",
            "Default": "100",
            "Description": "Maximum number of EC2 instances that can be launched in your
 ECS cluster."
        },
        "ECSAMI": {
            "Description": "The Amazon Machine Image ID used for the cluster",
            "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
            "Default": "/aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/
image_id"
        },
        "VpcId": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "VPC ID where the ECS cluster is launched",
            "Default": "vpc-1234567890abcdef0"
        },
        "SubnetIds": {
            "Type": "List<AWS::EC2::Subnet::Id>",
            "Description": "List of subnet IDs where the EC2 instances will be launched",
            "Default": "subnet-021345abcdef67890"
        }
    },
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
```

```
            "ClusterSettings": [
                {
                    "Name": "containerInsights",
                    "Value": "enabled"
                }
            ]
        }
    },
    "ECSAutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "DependsOn": [
            "ECSCluster",
            "EC2Role"
        ],
        "Properties": {
            "VPCZoneIdentifier": {
                "Ref": "SubnetIds"
            },
            "LaunchTemplate": {
                "LaunchTemplateId": {
                    "Ref": "ContainerInstances"
                },
                "Version": {
                    "Fn::GetAtt": [
                        "ContainerInstances",
                        "LatestVersionNumber"
                    ]
                }
            },
            "MinSize": 0,
            "MaxSize": {
                "Ref": "MaxSize"
            },
            "DesiredCapacity": {
                "Ref": "DesiredCapacity"
            },
            "NewInstancesProtectedFromScaleIn": true
        },
        "UpdatePolicy": {
            "AutoScalingReplacingUpdate": {
                "WillReplace": "true"
            }
        }
    },
```

```
      "ContainerInstances": {
          "Type": "AWS::EC2::LaunchTemplate",
          "Properties": {
              "LaunchTemplateName": "asg-launch-template",
              "LaunchTemplateData": {
                  "ImageId": {
                      "Ref": "ECSAMI"
                  },
                  "InstanceType": {
                      "Ref": "InstanceType"
                  },
                  "IamInstanceProfile": {
                      "Name": {
                          "Ref": "EC2InstanceProfile"
                      }
                  },
                  "SecurityGroupIds": [
                      {
                          "Ref": "ContainerHostSecurityGroup"
                      }
                  ],
                  "UserData": {
                      "Fn::Base64": {
                          "Fn::Sub": "#!/bin/bash -xe\n echo ECS_CLUSTER=${ECSCluster}
 >> /etc/ecs/ecs.config\n yum install -y aws-cfn-bootstrap\n /opt/aws/bin/cfn-init -
v --stack ${AWS::StackId} --resource ContainerInstances --configsets full_install --
region ${AWS::Region} &\n"
                      }
                  },
                  "MetadataOptions": {
                      "HttpEndpoint": "enabled",
                      "HttpTokens": "required"
                  }
              }
          }
      },
      "EC2InstanceProfile": {
          "Type": "AWS::IAM::InstanceProfile",
          "Properties": {
              "Path": "/",
              "Roles": [
                  {
                      "Ref": "EC2Role"
                  }
```

```
            ]
        }
    },
    "CapacityProvider": {
        "Type": "AWS::ECS::CapacityProvider",
        "Properties": {
            "AutoScalingGroupProvider": {
                "AutoScalingGroupArn": {
                    "Ref": "ECSAutoScalingGroup"
                },
                "ManagedScaling": {
                    "InstanceWarmupPeriod": 60,
                    "MinimumScalingStepSize": 1,
                    "MaximumScalingStepSize": 100,
                    "Status": "ENABLED",
                    "TargetCapacity": 100
                },
                "ManagedTerminationProtection": "ENABLED"
            }
        }
    },
    "CapacityProviderAssociation": {
        "Type": "AWS::ECS::ClusterCapacityProviderAssociations",
        "Properties": {
            "CapacityProviders": [
                {
                    "Ref": "CapacityProvider"
                }
            ],
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DefaultCapacityProviderStrategy": [
                {
                    "Base": 0,
                    "CapacityProvider": {
                        "Ref": "CapacityProvider"
                    },
                    "Weight": 1
                }
            ]
        }
    },
    "ContainerHostSecurityGroup": {
```

```
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Access to the EC2 hosts that run containers",
                "VpcId": {
                    "Ref": "VpcId"
                }
            }
        },
        "EC2Role": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ec2.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "Path": "/",
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role",
                    "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
                ]
            }
        },
        "ECSTaskExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ecs-tasks.amazonaws.com"
```

```
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ],
                        "Condition": {
                            "ArnLike": {
                                "aws:SourceArn": {
                                    "Fn::Sub": "arn:${AWS::Partition}:ecs:
${AWS::Region}:${AWS::AccountId}:*"
                                }
                            },
                            "StringEquals": {
                                "aws:SourceAccount": {
                                      "Fn::Sub": "${AWS::AccountId}"
                                  }
                            }
                        }
                    }
                ]
            },
            "Path": "/",
            "ManagedPolicyArns": [
                "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
            ]
        }
    }
  },
  "Outputs": {
      "ClusterName": {
          "Description": "The ECS cluster into which to launch resources",
          "Value": "ECSCluster"
      },
      "ECSTaskExecutionRole": {
          "Description": "The role used to start up a task",
          "Value": "ECSTaskExecutionRole"
      },
      "CapacityProvider": {
          "Description": "The cluster capacity provider that the service should use to
 request capacity when it wants to start up a task",
          "Value": "CapacityProvider"
      }
  }
```

```
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: EC2 ECS cluster that starts out empty, with no EC2 instances yet.
  An ECS capacity provider automatically launches more EC2 instances as required
  on the fly when you request ECS to launch services or standalone tasks.
Parameters:
  InstanceType:
    Type: String
    Description: EC2 instance type
    Default: "t2.medium"
    AllowedValues:
      - t1.micro
      - t2.2xlarge
      - t2.large
      - t2.medium
      - t2.micro
      - t2.nano
      - t2.small
      - t2.xlarge
      - t3.2xlarge
      - t3.large
      - t3.medium
      - t3.micro
      - t3.nano
      - t3.small
      - t3.xlarge
  DesiredCapacity:
    Type: Number
    Default: "0"
    Description: Number of EC2 instances to launch in your ECS cluster.
  MaxSize:
    Type: Number
    Default: "100"
    Description: Maximum number of EC2 instances that can be launched in your ECS
 cluster.
  ECSAMI:
    Description: The Amazon Machine Image ID used for the cluster
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id
```

```
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID where the ECS cluster is launched
    Default: vpc-1234567890abcdef0
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
    Description: List of subnet IDs where the EC2 instances will be launched
    Default: "subnet-021345abcdef67890"
Resources:
# This is authorizes ECS to manage resources on your
  # account on your behalf. This role is likely already created on your account
  # ECSRole:
  #  Type: AWS::IAM::ServiceLinkedRole
  #  Properties:
  #    AWSServiceName: 'ecs.amazonaws.com'

   # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterSettings:
        - Name: containerInsights
          Value: enabled

  # Autoscaling group. This launches the actual EC2 instances that will register
  # themselves as members of the cluster, and run the docker containers.
  ECSAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    DependsOn:
      # This is to ensure that the ASG gets deleted first before these
    # resources, when it comes to stack teardown.
      - ECSCluster
      - EC2Role
    Properties:
      VPCZoneIdentifier:
        Ref: SubnetIds
      LaunchTemplate:
        LaunchTemplateId: !Ref ContainerInstances
        Version: !GetAtt ContainerInstances.LatestVersionNumber
      MinSize: 0
      MaxSize:
        Ref: MaxSize
      DesiredCapacity:
        Ref: DesiredCapacity
```

```
      NewInstancesProtectedFromScaleIn: true
    UpdatePolicy:
      AutoScalingReplacingUpdate:
        WillReplace: "true"
  # The config for each instance that is added to the cluster
  ContainerInstances:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: "asg-launch-template"
      LaunchTemplateData:
        ImageId:
          Ref: ECSAMI
        InstanceType:
          Ref: InstanceType
        IamInstanceProfile:
          Name: !Ref EC2InstanceProfile
        SecurityGroupIds:
          - !Ref ContainerHostSecurityGroup
        # This injected configuration file is how the EC2 instance
      # knows which ECS cluster on your AWS account it should be joining
        UserData:
          Fn::Base64: !Sub |
           #!/bin/bash -xe
            echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
            yum install -y aws-cfn-bootstrap
            /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource
ContainerInstances --configsets full_install --region ${AWS::Region} &
         # Disable IMDSv1, and require IMDSv2
        MetadataOptions:
          HttpEndpoint: enabled
          HttpTokens: required
  EC2InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles:
      - !Ref EC2Role
  # Create an ECS capacity provider to attach the ASG to the ECS cluster
  # so that it autoscales as we launch more containers
  CapacityProvider:
    Type: AWS::ECS::CapacityProvider
    Properties:
      AutoScalingGroupProvider:
        AutoScalingGroupArn: !Ref ECSAutoScalingGroup
```

```
      ManagedScaling:
        InstanceWarmupPeriod: 60
        MinimumScalingStepSize: 1
        MaximumScalingStepSize: 100
        Status: ENABLED
        # Percentage of cluster reservation to try to maintain
        TargetCapacity: 100
      ManagedTerminationProtection: ENABLED
 # Create a cluster capacity provider assocation so that the cluster
# will use the capacity provider
CapacityProviderAssociation:
  Type: AWS::ECS::ClusterCapacityProviderAssociations
  Properties:
    CapacityProviders:
      - !Ref CapacityProvider
    Cluster: !Ref ECSCluster
    DefaultCapacityProviderStrategy:
      - Base: 0
        CapacityProvider: !Ref CapacityProvider
        Weight: 1
# A security group for the EC2 hosts that will run the containers.
# This can be used to limit incoming traffic to or outgoing traffic
# from the container's host EC2 instance.
ContainerHostSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Access to the EC2 hosts that run containers
    VpcId:
      Ref: VpcId
# Role for the EC2 hosts. This allows the ECS agent on the EC2 hosts
# to communciate with the ECS control plane, as well as download the docker
# images from ECR to run on your host.
EC2Role:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ec2.amazonaws.com
          Action:
            - sts:AssumeRole
    Path: /
```

```
      ManagedPolicyArns:
      # See reference: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/
security-iam-awsmanpol.html#security-iam-awsmanpol-AmazonEC2ContainerServiceforEC2Role
        - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
      # This managed policy allows us to connect to the instance using SSM
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
  # This is a role which is used within Fargate to allow the Fargate agent
  # to download images, and upload logs.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ecs-tasks.amazonaws.com
            Action:
              - sts:AssumeRole
            Condition:
              ArnLike:
                aws:SourceArn: !Sub arn:${AWS::Partition}:ecs:${AWS::Region}:
${AWS::AccountId}:*
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
      Path: /
      # This role enables all features of ECS. See reference:
    # https://docs.aws.amazon.com/AmazonECS/latest/developerguide/security-iam-
awsmanpol.html#security-iam-awsmanpol-AmazonECSTaskExecutionRolePolicy
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ClusterName:
    Description: The ECS cluster into which to launch resources
    Value: ECSCluster
  ECSTaskExecutionRole:
    Description: The role used to start up a task
    Value: ECSTaskExecutionRole
  CapacityProvider:
    Description: The cluster capacity provider that the service should use to
      request capacity when it wants to start up a task
    Value: CapacityProvider
```

Deploy a service

The following template defines a service that uses the capacity provider to request AL2023 capacity to run on. Containers will be launched onto the AL2023 instances as they come online:

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "An example service that deploys in AWS VPC networking mode on EC2
 capacity. Service uses a capacity provider to request EC2 instances to run on. Service
 runs with networking in private subnets, but still accessible to the internet via a
 load balancer hosted in public subnets.",
  "Parameters": {
      "VpcId": {
          "Type": "String",
          "Description": "The VPC that the service is running inside of"
      },
      "PublicSubnetIds": {
          "Type": "List<AWS::EC2::Subnet::Id>",
          "Description": "List of public subnet ID's to put the load balancer in"
      },
      "PrivateSubnetIds": {
          "Type": "List<AWS::EC2::Subnet::Id>",
          "Description": "List of private subnet ID's that the AWS VPC tasks are in"
      },
      "ClusterName": {
          "Type": "String",
          "Description": "The name of the ECS cluster into which to launch capacity."
      },
      "ECSTaskExecutionRole": {
          "Type": "String",
          "Description": "The role used to start up an ECS task"
      },
      "CapacityProvider": {
          "Type": "String",
          "Description": "The cluster capacity provider that the service should use to
 request capacity when it wants to start up a task"
      },
      "ServiceName": {
          "Type": "String",
          "Default": "web",
          "Description": "A name for the service"
```

```
      },
      "ImageUrl": {
          "Type": "String",
          "Default": "public.ecr.aws/docker/library/nginx:latest",
          "Description": "The url of a docker image that contains the application
 process that will handle the traffic for this service"
      },
      "ContainerCpu": {
          "Type": "Number",
          "Default": 256,
          "Description": "How much CPU to give the container. 1024 is 1 CPU"
      },
      "ContainerMemory": {
          "Type": "Number",
          "Default": 512,
          "Description": "How much memory in megabytes to give the container"
      },
      "ContainerPort": {
          "Type": "Number",
          "Default": 80,
          "Description": "What port that the application expects traffic on"
      },
      "DesiredCount": {
          "Type": "Number",
          "Default": 2,
          "Description": "How many copies of the service task to run"
      }
  },
  "Resources": {
      "TaskDefinition": {
          "Type": "AWS::ECS::TaskDefinition",
          "Properties": {
              "Family": {
                  "Ref": "ServiceName"
              },
              "Cpu": {
                  "Ref": "ContainerCpu"
              },
              "Memory": {
                  "Ref": "ContainerMemory"
              },
              "NetworkMode": "awsvpc",
              "RequiresCompatibilities": [
                  "EC2"
```

```
            ],
            "ExecutionRoleArn": {
                "Ref": "ECSTaskExecutionRole"
            },
            "ContainerDefinitions": [
                {
                    "Name": {
                        "Ref": "ServiceName"
                    },
                    "Cpu": {
                        "Ref": "ContainerCpu"
                    },
                    "Memory": {
                        "Ref": "ContainerMemory"
                    },
                    "Image": {
                        "Ref": "ImageUrl"
                    },
                    "PortMappings": [
                        {
                            "ContainerPort": {
                                "Ref": "ContainerPort"
                            },
                            "HostPort": {
                                "Ref": "ContainerPort"
                            }
                        }
                    ],
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "mode": "non-blocking",
                            "max-buffer-size": "25m",
                            "awslogs-group": {
                                "Ref": "LogGroup"
                            },
                            "awslogs-region": {
                                "Ref": "AWS::Region"
                            },
                            "awslogs-stream-prefix": {
                                "Ref": "ServiceName"
                            }
                        }
                    }
```

```
                }
            ]
        }
    },
    "Service": {
        "Type": "AWS::ECS::Service",
        "DependsOn": "PublicLoadBalancerListener",
        "Properties": {
            "ServiceName": {
                "Ref": "ServiceName"
            },
            "Cluster": {
                "Ref": "ClusterName"
            },
            "PlacementStrategies": [
                {
                    "Field": "attribute:ecs.availability-zone",
                    "Type": "spread"
                },
                {
                    "Field": "cpu",
                    "Type": "binpack"
                }
            ],
            "CapacityProviderStrategy": [
                {
                    "Base": 0,
                    "CapacityProvider": {
                        "Ref": "CapacityProvider"
                    },
                    "Weight": 1
                }
            ],
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "SecurityGroups": [
                        {
                            "Ref": "ServiceSecurityGroup"
                        }
                    ],
                    "Subnets": {
                        "Ref": "PrivateSubnetIds"
                    }
                }
```

```
            },
            "DeploymentConfiguration": {
                "MaximumPercent": 200,
                "MinimumHealthyPercent": 75
            },
            "DesiredCount": {
                "Ref": "DesiredCount"
            },
            "TaskDefinition": {
                "Ref": "TaskDefinition"
            },
            "LoadBalancers": [
                {
                    "ContainerName": {
                        "Ref": "ServiceName"
                    },
                    "ContainerPort": {
                        "Ref": "ContainerPort"
                    },
                    "TargetGroupArn": {
                        "Ref": "ServiceTargetGroup"
                    }
                }
            ]
        }
    },
    "ServiceSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Security group for service",
            "VpcId": {
                "Ref": "VpcId"
            }
        }
    },
    "ServiceTargetGroup": {
        "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
        "Properties": {
            "HealthCheckIntervalSeconds": 6,
            "HealthCheckPath": "/",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 2,
            "TargetType": "ip",
```

```
            "Port": {
                "Ref": "ContainerPort"
            },
            "Protocol": "HTTP",
            "UnhealthyThresholdCount": 10,
            "VpcId": {
                "Ref": "VpcId"
            },
            "TargetGroupAttributes": [
                {
                    "Key": "deregistration_delay.timeout_seconds",
                    "Value": 0
                }
            ]
        }
    },
    "PublicLoadBalancerSG": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Access to the public facing load balancer",
            "VpcId": {
                "Ref": "VpcId"
            },
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "IpProtocol": -1
                }
            ]
        }
    },
    "PublicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internet-facing",
            "LoadBalancerAttributes": [
                {
                    "Key": "idle_timeout.timeout_seconds",
                    "Value": "30"
                }
            ],
            "Subnets": {
                "Ref": "PublicSubnetIds"
            },
```

```
            "SecurityGroups": [
                {
                    "Ref": "PublicLoadBalancerSG"
                }
            ]
        }
    },
    "PublicLoadBalancerListener": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
        "Properties": {
            "DefaultActions": [
                {
                    "Type": "forward",
                    "ForwardConfig": {
                        "TargetGroups": [
                            {
                                "TargetGroupArn": {
                                    "Ref": "ServiceTargetGroup"
                                },
                                "Weight": 100
                            }
                        ]
                    }
                }
            ],
            "LoadBalancerArn": {
                "Ref": "PublicLoadBalancer"
            },
            "Port": 80,
            "Protocol": "HTTP"
        }
    },
    "ServiceIngressfromLoadBalancer": {
        "Type": "AWS::EC2::SecurityGroupIngress",
        "Properties": {
            "Description": "Ingress from the public ALB",
            "GroupId": {
                "Ref": "ServiceSecurityGroup"
            },
            "IpProtocol": -1,
            "SourceSecurityGroupId": {
                "Ref": "PublicLoadBalancerSG"
            }
        }
```

```
      },
      "LogGroup": {
          "Type": "AWS::Logs::LogGroup"
      }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >-
  An example service that deploys in AWS VPC networking mode on EC2 capacity.
  Service uses a capacity provider to request EC2 instances to run on. Service
  runs with networking in private subnets, but still accessible to the internet
  via a load balancer hosted in public subnets.
Parameters:
  VpcId:
    Type: String
    Description: The VPC that the service is running inside of
  PublicSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of public subnet ID's to put the load balancer in
  PrivateSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of private subnet ID's that the AWS VPC tasks are in
  ClusterName:
    Type: String
    Description: The name of the ECS cluster into which to launch capacity.
  ECSTaskExecutionRole:
    Type: String
    Description: The role used to start up an ECS task
  CapacityProvider:
    Type: String
    Description: >-
      The cluster capacity provider that the service should use to request
      capacity when it wants to start up a task
  ServiceName:
    Type: String
    Default: web
    Description: A name for the service
  ImageUrl:
    Type: String
    Default: 'public.ecr.aws/docker/library/nginx:latest'
```

```
    Description: >-
      The url of a docker image that contains the application process that will
      handle the traffic for this service
  ContainerCpu:
    Type: Number
    Default: 256
    Description: How much CPU to give the container. 1024 is 1 CPU
  ContainerMemory:
    Type: Number
    Default: 512
    Description: How much memory in megabytes to give the container
  ContainerPort:
    Type: Number
    Default: 80
    Description: What port that the application expects traffic on
  DesiredCount:
    Type: Number
    Default: 2
    Description: How many copies of the service task to run
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: !Ref ServiceName
      Cpu: !Ref ContainerCpu
      Memory: !Ref ContainerMemory
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - EC2
      ExecutionRoleArn: !Ref ECSTaskExecutionRole
      ContainerDefinitions:
        - Name: !Ref ServiceName
          Cpu: !Ref ContainerCpu
          Memory: !Ref ContainerMemory
          Image: !Ref ImageUrl
          PortMappings:
            - ContainerPort: !Ref ContainerPort
              HostPort: !Ref ContainerPort
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-group: !Ref LogGroup
```

```
            awslogs-region: !Ref AWS::Region
            awslogs-stream-prefix: !Ref ServiceName
  Service:
    Type: AWS::ECS::Service
    DependsOn: PublicLoadBalancerListener
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ClusterName
      PlacementStrategies:
        - Field: 'attribute:ecs.availability-zone'
          Type: spread
        - Field: cpu
          Type: binpack
      CapacityProviderStrategy:
        - Base: 0
          CapacityProvider: !Ref CapacityProvider
          Weight: 1
      NetworkConfiguration:
        AwsvpcConfiguration:
          SecurityGroups:
            - !Ref ServiceSecurityGroup
          Subnets: !Ref PrivateSubnetIds
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: !Ref DesiredCount
      TaskDefinition: !Ref TaskDefinition
      LoadBalancers:
        - ContainerName: !Ref ServiceName
          ContainerPort: !Ref ContainerPort
          TargetGroupArn: !Ref ServiceTargetGroup
  ServiceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for service
      VpcId: !Ref VpcId
  ServiceTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
```

```
      TargetType: ip
      Port: !Ref ContainerPort
      Protocol: HTTP
      UnhealthyThresholdCount: 10
      VpcId: !Ref VpcId
      TargetGroupAttributes:
        - Key: deregistration_delay.timeout_seconds
          Value: 0
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: -1
  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
        - Key: idle_timeout.timeout_seconds
          Value: '30'
      Subnets: !Ref PublicSubnetIds
      SecurityGroups:
        - !Ref PublicLoadBalancerSG
  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
        - Type: forward
          ForwardConfig:
            TargetGroups:
              - TargetGroupArn: !Ref ServiceTargetGroup
                Weight: 100
      LoadBalancerArn: !Ref PublicLoadBalancer
      Port: 80
      Protocol: HTTP
  ServiceIngressfromLoadBalancer:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId: !Ref ServiceSecurityGroup
      IpProtocol: -1
```

```
      SourceSecurityGroupId: !Ref PublicLoadBalancerSG
  LogGroup:
    Type: AWS::Logs::LogGroup
```

Amazon Elastic File System Sample Template

Amazon Elastic File System (Amazon EFS) is a file storage service for Amazon Elastic Compute Cloud (Amazon EC2) instances. With Amazon EFS, your applications have storage when they need it because storage capacity grows and shrinks automatically as you add and remove files.

The following sample template deploys EC2 instances (in an Auto Scaling group) that are associated with an Amazon EFS file system. To associate the instances with the file system, the instances run the cfn-init helper script, which downloads and installs the `nfs-utils` yum package, creates a new directory, and then uses the file system's DNS name to mount the file system at that directory. The file system's DNS name resolves to a mount target's IP address in the Amazon EC2 instance's Availability Zone. For more information about the DNS name structure, see Mounting File Systems in the *Amazon Elastic File System User Guide*.

To measure Network File System activity, the template includes custom Amazon CloudWatch metrics. The template also creates a VPC, subnet, and security groups. To allow the instances to communicate with the file system, the VPC must have DNS enabled, and the mount target and the EC2 instances must be in the same Availability Zone (AZ), which is specified by the subnet.

The security group of the mount target enables a network connection to TCP port 2049, which is required for an NFSv4 client to mount a file system. For more information on security groups for EC2 instances and mount targets, see Security in the *Amazon Elastic File System User Guide*.

> **Note**
>
> If you make an update to the mount target that causes it to be replaced, instances or applications that use the associated file system might be disrupted. This can cause uncommitted writes to be lost. To avoid disruption, stop your instances when you update the mount target by setting the desired capacity to zero. This allows the instances to unmount the file system before the mount target is deleted. After the mount update has completed, start your instances in a subsequent update by setting the desired capacity.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "This template creates an Amazon EFS file system and mount target and
 associates it with Amazon EC2 instances in an Auto Scaling group. **WARNING** This
 template creates Amazon EC2 instances and related resources. You will be billed for
 the AWS resources used if you create a stack from this template.",
  "Parameters": {
    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t2.small",
      "AllowedValues" : [
        "t1.micro",
        "t2.nano",
        "t2.micro",
        "t2.small",
        "t2.medium",
        "t2.large",
        "m1.small",
        "m1.medium",
        "m1.large",
        "m1.xlarge",
        "m2.xlarge",
        "m2.2xlarge",
        "m2.4xlarge",
        "m3.medium",
        "m3.large",
        "m3.xlarge",
        "m3.2xlarge",
        "m4.large",
        "m4.xlarge",
        "m4.2xlarge",
        "m4.4xlarge",
        "m4.10xlarge",
        "c1.medium",
        "c1.xlarge",
        "c3.large",
        "c3.xlarge",
        "c3.2xlarge",
        "c3.4xlarge",
        "c3.8xlarge",
```

```
        "c4.large",
        "c4.xlarge",
        "c4.2xlarge",
        "c4.4xlarge",
        "c4.8xlarge",
        "g2.2xlarge",
        "g2.8xlarge",
        "r3.large",
        "r3.xlarge",
        "r3.2xlarge",
        "r3.4xlarge",
        "r3.8xlarge",
        "i2.xlarge",
        "i2.2xlarge",
        "i2.4xlarge",
        "i2.8xlarge",
        "d2.xlarge",
        "d2.2xlarge",
        "d2.4xlarge",
        "d2.8xlarge",
        "hi1.4xlarge",
        "hs1.8xlarge",
        "cr1.8xlarge",
        "cc2.8xlarge",
        "cg1.4xlarge"
      ],
      "ConstraintDescription" : "must be a valid EC2 instance type."
    },
    "KeyName": {
      "Type": "AWS::EC2::KeyPair::KeyName",
      "Description": "Name of an existing EC2 key pair to enable SSH access to the EC2
instances"
    },
    "AsgMaxSize": {
      "Type": "Number",
      "Description": "Maximum size and initial desired capacity of Auto Scaling Group",
      "Default": "2"
    },
    "SSHLocation" : {
      "Description" : "The IP address range that can be used to connect to the EC2
instances by using SSH",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
```

```
      "Default": "0.0.0.0/0",
      "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
    },
    "VolumeName" : {
      "Description" : "The name to be used for the EFS volume",
      "Type": "String",
      "MinLength": "1",
      "Default": "myEFSvolume"
    },
    "MountPoint" : {
      "Description" : "The Linux mount point for the EFS volume",
      "Type": "String",
      "MinLength": "1",
      "Default": "myEFSvolume"
    }
  },
  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "HVM64"  },
      "t2.nano"     : { "Arch" : "HVM64"  },
      "t2.micro"    : { "Arch" : "HVM64"  },
      "t2.small"    : { "Arch" : "HVM64"  },
      "t2.medium"   : { "Arch" : "HVM64"  },
      "t2.large"    : { "Arch" : "HVM64"  },
      "m1.small"    : { "Arch" : "HVM64"  },
      "m1.medium"   : { "Arch" : "HVM64"  },
      "m1.large"    : { "Arch" : "HVM64"  },
      "m1.xlarge"   : { "Arch" : "HVM64"  },
      "m2.xlarge"   : { "Arch" : "HVM64"  },
      "m2.2xlarge"  : { "Arch" : "HVM64"  },
      "m2.4xlarge"  : { "Arch" : "HVM64"  },
      "m3.medium"   : { "Arch" : "HVM64"  },
      "m3.large"    : { "Arch" : "HVM64"  },
      "m3.xlarge"   : { "Arch" : "HVM64"  },
      "m3.2xlarge"  : { "Arch" : "HVM64"  },
      "m4.large"    : { "Arch" : "HVM64"  },
      "m4.xlarge"   : { "Arch" : "HVM64"  },
      "m4.2xlarge"  : { "Arch" : "HVM64"  },
      "m4.4xlarge"  : { "Arch" : "HVM64"  },
      "m4.10xlarge" : { "Arch" : "HVM64"  },
      "c1.medium"   : { "Arch" : "HVM64"  },
      "c1.xlarge"   : { "Arch" : "HVM64"  },
      "c3.large"    : { "Arch" : "HVM64"  },
```

```
      "c3.xlarge"   : { "Arch" : "HVM64"  },
      "c3.2xlarge"  : { "Arch" : "HVM64"  },
      "c3.4xlarge"  : { "Arch" : "HVM64"  },
      "c3.8xlarge"  : { "Arch" : "HVM64"  },
      "c4.large"    : { "Arch" : "HVM64"  },
      "c4.xlarge"   : { "Arch" : "HVM64"  },
      "c4.2xlarge"  : { "Arch" : "HVM64"  },
      "c4.4xlarge"  : { "Arch" : "HVM64"  },
      "c4.8xlarge"  : { "Arch" : "HVM64"  },
      "g2.2xlarge"  : { "Arch" : "HVMG2"  },
      "g2.8xlarge"  : { "Arch" : "HVMG2"  },
      "r3.large"    : { "Arch" : "HVM64"  },
      "r3.xlarge"   : { "Arch" : "HVM64"  },
      "r3.2xlarge"  : { "Arch" : "HVM64"  },
      "r3.4xlarge"  : { "Arch" : "HVM64"  },
      "r3.8xlarge"  : { "Arch" : "HVM64"  },
      "i2.xlarge"   : { "Arch" : "HVM64"  },
      "i2.2xlarge"  : { "Arch" : "HVM64"  },
      "i2.4xlarge"  : { "Arch" : "HVM64"  },
      "i2.8xlarge"  : { "Arch" : "HVM64"  },
      "d2.xlarge"   : { "Arch" : "HVM64"  },
      "d2.2xlarge"  : { "Arch" : "HVM64"  },
      "d2.4xlarge"  : { "Arch" : "HVM64"  },
      "d2.8xlarge"  : { "Arch" : "HVM64"  },
      "hi1.4xlarge" : { "Arch" : "HVM64"  },
      "hs1.8xlarge" : { "Arch" : "HVM64"  },
      "cr1.8xlarge" : { "Arch" : "HVM64"  },
      "cc2.8xlarge" : { "Arch" : "HVM64"  }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"        : {"HVM64" : "ami-0ff8a91507f77f867", "HVMG2" :
"ami-0a584ac55a7631c0c"},
      "us-west-2"        : {"HVM64" : "ami-a0cfeed8", "HVMG2" :
"ami-0e09505bc235aa82d"},
      "us-west-1"        : {"HVM64" : "ami-0bdb828fd58c52235", "HVMG2" :
"ami-066ee5fd4a9ef77f1"},
      "eu-west-1"        : {"HVM64" : "ami-047bb4163c506cd98", "HVMG2" :
"ami-0a7c483d527806435"},
      "eu-west-2"        : {"HVM64" : "ami-f976839e", "HVMG2" : "NOT_SUPPORTED"},
      "eu-west-3"        : {"HVM64" : "ami-0ebc281c20e89ba4b", "HVMG2" :
"NOT_SUPPORTED"},
      "eu-central-1"     : {"HVM64" : "ami-0233214e13e500f77", "HVMG2" :
"ami-06223d46a6d0661c7"},
```

```
      "ap-northeast-1"   : {"HVM64" : "ami-06cd52961ce9f0d85", "HVMG2" :
"ami-053cdd503598e4a9d"},
      "ap-northeast-2"   : {"HVM64" : "ami-0a10b2721688ce9d2", "HVMG2" :
"NOT_SUPPORTED"},
      "ap-northeast-3"   : {"HVM64" : "ami-0d98120a9fb693f07", "HVMG2" :
"NOT_SUPPORTED"},
      "ap-southeast-1"   : {"HVM64" : "ami-08569b978cc4dfa10", "HVMG2" :
"ami-0be9df32ae9f92309"},
      "ap-southeast-2"   : {"HVM64" : "ami-09b42976632b27e9b", "HVMG2" :
"ami-0a9ce9fecc3d1daf8"},
      "ap-south-1"       : {"HVM64" : "ami-0912f71e06545ad88", "HVMG2" :
"ami-097b15e89dbdcfcf4"},
      "us-east-2"        : {"HVM64" : "ami-0b59bfac6be064b78", "HVMG2" :
"NOT_SUPPORTED"},
      "ca-central-1"     : {"HVM64" : "ami-0b18956f", "HVMG2" : "NOT_SUPPORTED"},
      "sa-east-1"        : {"HVM64" : "ami-07b14488da8ea02a0", "HVMG2" :
"NOT_SUPPORTED"},
      "cn-north-1"       : {"HVM64" : "ami-0a4eaf6c4454eda75", "HVMG2" :
"NOT_SUPPORTED"},
      "cn-northwest-1"   : {"HVM64" : "ami-6b6a7d09", "HVMG2" : "NOT_SUPPORTED"}
    }
  },
  "Resources": {
    "CloudWatchPutMetricsRole" : {
      "Type"  : "AWS::IAM::Role",
      "Properties" : {
          "AssumeRolePolicyDocument" : {
              "Statement" : [ {
                  "Effect" : "Allow",
                  "Principal" : {
                      "Service" : [ "ec2.amazonaws.com" ]
                  },
                  "Action" : [ "sts:AssumeRole" ]
              } ]
          },
          "Path" : "/"
      }
    },
    "CloudWatchPutMetricsRolePolicy" : {
        "Type" : "AWS::IAM::Policy",
        "Properties" : {
            "PolicyName" : "CloudWatch_PutMetricData",
            "PolicyDocument" : {
              "Version": "2012-10-17",
```

```
            "Statement": [
              {
                "Sid": "CloudWatchPutMetricData",
                "Effect": "Allow",
                "Action": ["cloudwatch:PutMetricData"],
                "Resource": ["*"]
              }
            ]
          },
          "Roles" : [ { "Ref" : "CloudWatchPutMetricsRole" } ]
      }
    },
    "CloudWatchPutMetricsInstanceProfile" : {
      "Type" : "AWS::IAM::InstanceProfile",
      "Properties" : {
        "Path" : "/",
        "Roles" : [ { "Ref" : "CloudWatchPutMetricsRole" } ]
      }
    },
    "VPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "EnableDnsSupport" : "true",
        "EnableDnsHostnames" : "true",
        "CidrBlock": "10.0.0.0/16",
        "Tags": [ {"Key": "Application", "Value": { "Ref": "AWS::StackId"} } ]
      }
    },
    "InternetGateway" : {
      "Type" : "AWS::EC2::InternetGateway",
      "Properties" : {
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },
          { "Key" : "Network", "Value" : "Public" }
        ]
      }
    },
    "GatewayToInternet" : {
      "Type" : "AWS::EC2::VPCGatewayAttachment",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "InternetGatewayId" : { "Ref" : "InternetGateway" }
      }
    },
```

```
    "RouteTable":{
      "Type":"AWS::EC2::RouteTable",
      "Properties":{
        "VpcId": {"Ref":"VPC"}
      }
    },
    "SubnetRouteTableAssoc": {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "RouteTableId" : {"Ref":"RouteTable"},
        "SubnetId" : {"Ref":"Subnet"}
      }
    },
    "InternetGatewayRoute": {
        "Type":"AWS::EC2::Route",
        "Properties":{
            "DestinationCidrBlock":"0.0.0.0/0",
            "RouteTableId":{"Ref":"RouteTable"},
            "GatewayId":{"Ref":"InternetGateway"}
        }
    },
    "Subnet": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": { "Ref": "VPC" },
        "CidrBlock": "10.0.0.0/24",
        "Tags": [ { "Key": "Application", "Value": { "Ref": "AWS::StackId" } } ]
      }
    },
    "InstanceSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "VpcId": { "Ref": "VPC" },
        "GroupDescription": "Enable SSH access via port 22",
        "SecurityGroupIngress": [
          { "IpProtocol": "tcp", "FromPort": 22, "ToPort": 22, "CidrIp": { "Ref":
"SSHLocation" } },
          { "IpProtocol": "tcp", "FromPort": 80, "ToPort": 80, "CidrIp": "0.0.0.0/0" }
         ]
      }
    },
    "MountTargetSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
```

```
        "VpcId": { "Ref": "VPC" },
        "GroupDescription": "Security group for mount target",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 2049,
            "ToPort": 2049,
            "CidrIp": "0.0.0.0/0"
          }
        ]
      }
    },
    "FileSystem": {
      "Type": "AWS::EFS::FileSystem",
      "Properties": {
        "PerformanceMode": "generalPurpose",
        "FileSystemTags": [
          {
            "Key": "Name",
            "Value": { "Ref" : "VolumeName" }
          }
        ]
      }
    },
    "MountTarget": {
      "Type": "AWS::EFS::MountTarget",
      "Properties": {
        "FileSystemId": { "Ref": "FileSystem" },
        "SubnetId": { "Ref": "Subnet" },
        "SecurityGroups": [ { "Ref": "MountTargetSecurityGroup" } ]
      }
    },
    "LaunchConfiguration": {
      "Type": "AWS::AutoScaling::LaunchConfiguration",
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
          "configSets" : {
            "MountConfig" : [ "setup", "mount" ]
          },
          "setup" : {
            "packages" : {
              "yum" : {
                "nfs-utils" : []
              }
```

```
            },
            "files" : {
              "/home/ec2-user/post_nfsstat" : {
                "content" : { "Fn::Join" : [ "", [
                      "#!/bin/bash\n",
                      "\n",
                      "INPUT=\"$(cat)\"\n",
                      "CW_JSON_OPEN='{ \"Namespace\": \"EFS\", \"MetricData\": [ '\n",
                      "CW_JSON_CLOSE=' ] }'\n",
                      "CW_JSON_METRIC=''\n",
                      "METRIC_COUNTER=0\n",
                      "\n",
                      "for COL in 1 2 3 4 5 6; do\n",
                      "\n",
                      " COUNTER=0\n",
                      " METRIC_FIELD=$COL\n",
                      " DATA_FIELD=$(($COL+($COL-1)))\n",
                      "\n",
                      " while read line; do\n",
                      "   if [[ COUNTER -gt 0 ]]; then\n",
                      "\n",
                      "     LINE=`echo $line | tr -s ' ' `\n",
                      "     AWS_COMMAND=\"aws cloudwatch put-metric-data --region ",
 { "Ref": "AWS::Region" }, "\"\n",
                      "     MOD=$(( $COUNTER % 2))\n",
                      "\n",
                      "     if [ $MOD -eq 1 ]; then\n",
                      "       METRIC_NAME=`echo $LINE | cut -d ' ' -f $METRIC_FIELD`
\n",
                      "     else\n",
                      "       METRIC_VALUE=`echo $LINE | cut -d ' ' -f $DATA_FIELD`\n",
                      "     fi\n",
                      "\n",
                      "     if [[ -n \"$METRIC_NAME\" && -n \"$METRIC_VALUE\" ]]; then
\n",
                      "       INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-
data/instance-id)\n",
                      "       CW_JSON_METRIC=\"$CW_JSON_METRIC { \\\"MetricName\\\": \\
\"$METRIC_NAME\\\", \\\"Dimensions\\\": [{\\\"Name\\\": \\\"InstanceId\\\", \\\"Value\\
\": \\\"$INSTANCE_ID\\\"} ], \\\"Value\\\": $METRIC_VALUE },\"\n",
                      "       unset METRIC_NAME\n",
                      "       unset METRIC_VALUE\n",
                      "\n",
                      "       METRIC_COUNTER=$((METRIC_COUNTER+1))\n",
```

```
                    "       if [ $METRIC_COUNTER -eq 20 ]; then\n",
                    "         # 20 is max metric collection size, so we have to
submit here\n",
                    "         aws cloudwatch put-metric-data --region ", { "Ref":
"AWS::Region" }, " --cli-input-json \"`echo $CW_JSON_OPEN ${CW_JSON_METRIC%?}
$CW_JSON_CLOSE`\"\n",
                    "\n",
                    "         # reset\n",
                    "         METRIC_COUNTER=0\n",
                    "         CW_JSON_METRIC=''\n",
                    "       fi\n",
                    "     fi  \n",
                    "\n",
                    "\n",
                    "\n",
                    "     COUNTER=$((COUNTER+1))\n",
                    "   fi\n",
                    "\n",
                    "   if [[ \"$line\" == \"Client nfs v4:\" ]]; then\n",
                    "     # the next line is the good stuff \n",
                    "     COUNTER=$((COUNTER+1))\n",
                    "   fi\n",
                    " done <<< \"$INPUT\"\n",
                    "done\n",
                    "\n",
                    "# submit whatever is left\n",
                    "aws cloudwatch put-metric-data --region ", { "Ref":
"AWS::Region" }, " --cli-input-json \"`echo $CW_JSON_OPEN ${CW_JSON_METRIC%?}
$CW_JSON_CLOSE`\""
                ] ] },
              "mode": "000755",
              "owner": "ec2-user",
              "group": "ec2-user"
            },
            "/home/ec2-user/crontab" : {
              "content" : { "Fn::Join" : [ "", [
                "* * * * * /usr/sbin/nfsstat | /home/ec2-user/post_nfsstat\n"
              ] ] },
              "owner": "ec2-user",
              "group": "ec2-user"
            }
          },
          "commands" : {
            "01_createdir" : {
```

```
                "command" : {"Fn::Join" : [ "", [ "mkdir /", { "Ref" :
 "MountPoint" }]]}
              }
            }
          },
          "mount" : {
            "commands" : {
              "01_mount" : {
                "command" : { "Fn::Sub": "sudo mount -t nfs4 -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 ${FileSystem}.efs.
${AWS::Region}.amazonaws.com:/ /${MountPoint}"}
              },
              "02_permissions" : {
                "command" : {"Fn::Join" : [ "", [ "chown ec2-user:ec2-user /",
 { "Ref" : "MountPoint" }]]}
              }
            }
          }
        }
      },
      "Properties": {
        "AssociatePublicIpAddress" : true,
        "ImageId": {
          "Fn::FindInMap": [ "AWSRegionArch2AMI", { "Ref": "AWS::Region" }, {
            "Fn::FindInMap": [ "AWSInstanceType2Arch", { "Ref": "InstanceType" },
 "Arch" ]
          } ]
        },
        "InstanceType": { "Ref": "InstanceType" },
        "KeyName": { "Ref": "KeyName" },
        "SecurityGroups": [ { "Ref": "InstanceSecurityGroup" } ],
        "IamInstanceProfile" : { "Ref" : "CloudWatchPutMetricsInstanceProfile" },
        "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
             "#!/bin/bash -xe\n",
             "yum install -y aws-cfn-bootstrap\n",

             "/opt/aws/bin/cfn-init -v ",
             "         --stack ", { "Ref" : "AWS::StackName" },
             "         --resource LaunchConfiguration ",
             "         --configsets MountConfig ",
             "         --region ", { "Ref" : "AWS::Region" }, "\n",

             "crontab /home/ec2-user/crontab\n",
```

```
              "/opt/aws/bin/cfn-signal -e $? ",
              "         --stack ", { "Ref" : "AWS::StackName" },
              "         --resource AutoScalingGroup ",
              "         --region ", { "Ref" : "AWS::Region" }, "\n"
        ]]}}
      }
    },
    "AutoScalingGroup": {
      "Type": "AWS::AutoScaling::AutoScalingGroup",
      "DependsOn": ["MountTarget", "GatewayToInternet"],
      "CreationPolicy" : {
        "ResourceSignal" : {
          "Timeout" : "PT15M",
          "Count"   : { "Ref": "AsgMaxSize" }
        }
      },
      "Properties": {
        "VPCZoneIdentifier": [ { "Ref": "Subnet" } ],
        "LaunchConfigurationName": { "Ref": "LaunchConfiguration" },
        "MinSize": "1",
        "MaxSize": { "Ref": "AsgMaxSize" },
        "DesiredCapacity": { "Ref": "AsgMaxSize" },
        "Tags": [ {
          "Key": "Name",
          "Value": "EFS FileSystem Mounted Instance",
          "PropagateAtLaunch": "true"
        } ]
      }
    }
  },
  "Outputs" : {
    "MountTargetID" : {
      "Description" : "Mount target ID",
      "Value" :  { "Ref" : "MountTarget" }
    },
    "FileSystemID" : {
      "Description" : "File system ID",
      "Value" :  { "Ref" : "FileSystem" }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: This template creates an Amazon EFS file system and mount target and
  associates it with Amazon EC2 instances in an Auto Scaling group. **WARNING** This
  template creates Amazon EC2 instances and related resources. You will be billed
  for the AWS resources used if you create a stack from this template.
Parameters:
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.small
    AllowedValues:
      - t1.micro
      - t2.nano
      - t2.micro
      - t2.small
      - t2.medium
      - t2.large
      - m1.small
      - m1.medium
      - m1.large
      - m1.xlarge
      - m2.xlarge
      - m2.2xlarge
      - m2.4xlarge
      - m3.medium
      - m3.large
      - m3.xlarge
      - m3.2xlarge
      - m4.large
      - m4.xlarge
      - m4.2xlarge
      - m4.4xlarge
      - m4.10xlarge
      - c1.medium
      - c1.xlarge
      - c3.large
      - c3.xlarge
      - c3.2xlarge
      - c3.4xlarge
      - c3.8xlarge
      - c4.large
```

```
      - c4.xlarge
      - c4.2xlarge
      - c4.4xlarge
      - c4.8xlarge
      - g2.2xlarge
      - g2.8xlarge
      - r3.large
      - r3.xlarge
      - r3.2xlarge
      - r3.4xlarge
      - r3.8xlarge
      - i2.xlarge
      - i2.2xlarge
      - i2.4xlarge
      - i2.8xlarge
      - d2.xlarge
      - d2.2xlarge
      - d2.4xlarge
      - d2.8xlarge
      - hi1.4xlarge
      - hs1.8xlarge
      - cr1.8xlarge
      - cc2.8xlarge
      - cg1.4xlarge
    ConstraintDescription: must be a valid EC2 instance type.
  KeyName:
    Type: AWS::EC2::KeyPair::KeyName
    Description: Name of an existing EC2 key pair to enable SSH access to the ECS
      instances
  AsgMaxSize:
    Type: Number
    Description: Maximum size and initial desired capacity of Auto Scaling Group
    Default: '2'
  SSHLocation:
    Description: The IP address range that can be used to connect to the EC2 instances
      by using SSH
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})"
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  VolumeName:
    Description: The name to be used for the EFS volume
```

```
    Type: String
    MinLength: '1'
    Default: myEFSvolume
  MountPoint:
    Description: The Linux mount point for the EFS volume
    Type: String
    MinLength: '1'
    Default: myEFSvolume
Mappings:
  AWSInstanceType2Arch:
    t1.micro:
      Arch: HVM64
    t2.nano:
      Arch: HVM64
    t2.micro:
      Arch: HVM64
    t2.small:
      Arch: HVM64
    t2.medium:
      Arch: HVM64
    t2.large:
      Arch: HVM64
    m1.small:
      Arch: HVM64
    m1.medium:
      Arch: HVM64
    m1.large:
      Arch: HVM64
    m1.xlarge:
      Arch: HVM64
    m2.xlarge:
      Arch: HVM64
    m2.2xlarge:
      Arch: HVM64
    m2.4xlarge:
      Arch: HVM64
    m3.medium:
      Arch: HVM64
    m3.large:
      Arch: HVM64
    m3.xlarge:
      Arch: HVM64
    m3.2xlarge:
      Arch: HVM64
```

```
    m4.large:
      Arch: HVM64
    m4.xlarge:
      Arch: HVM64
    m4.2xlarge:
      Arch: HVM64
    m4.4xlarge:
      Arch: HVM64
    m4.10xlarge:
      Arch: HVM64
    c1.medium:
      Arch: HVM64
    c1.xlarge:
      Arch: HVM64
    c3.large:
      Arch: HVM64
    c3.xlarge:
      Arch: HVM64
    c3.2xlarge:
      Arch: HVM64
    c3.4xlarge:
      Arch: HVM64
    c3.8xlarge:
      Arch: HVM64
    c4.large:
      Arch: HVM64
    c4.xlarge:
      Arch: HVM64
    c4.2xlarge:
      Arch: HVM64
    c4.4xlarge:
      Arch: HVM64
    c4.8xlarge:
      Arch: HVM64
    g2.2xlarge:
      Arch: HVMG2
    g2.8xlarge:
      Arch: HVMG2
    r3.large:
      Arch: HVM64
    r3.xlarge:
      Arch: HVM64
    r3.2xlarge:
      Arch: HVM64
```

```
    r3.4xlarge:
      Arch: HVM64
    r3.8xlarge:
      Arch: HVM64
    i2.xlarge:
      Arch: HVM64
    i2.2xlarge:
      Arch: HVM64
    i2.4xlarge:
      Arch: HVM64
    i2.8xlarge:
      Arch: HVM64
    d2.xlarge:
      Arch: HVM64
    d2.2xlarge:
      Arch: HVM64
    d2.4xlarge:
      Arch: HVM64
    d2.8xlarge:
      Arch: HVM64
    hi1.4xlarge:
      Arch: HVM64
    hs1.8xlarge:
      Arch: HVM64
    cr1.8xlarge:
      Arch: HVM64
    cc2.8xlarge:
      Arch: HVM64
  AWSRegionArch2AMI:
    us-east-1:
      HVM64: ami-0ff8a91507f77f867
      HVMG2: ami-0a584ac55a7631c0c
    us-west-2:
      HVM64: ami-a0cfeed8
      HVMG2: ami-0e09505bc235aa82d
    us-west-1:
      HVM64: ami-0bdb828fd58c52235
      HVMG2: ami-066ee5fd4a9ef77f1
    eu-west-1:
      HVM64: ami-047bb4163c506cd98
      HVMG2: ami-0a7c483d527806435
    eu-west-2:
      HVM64: ami-f976839e
      HVMG2: NOT_SUPPORTED
```

```
    eu-west-3:
      HVM64: ami-0ebc281c20e89ba4b
      HVMG2: NOT_SUPPORTED
    eu-central-1:
      HVM64: ami-0233214e13e500f77
      HVMG2: ami-06223d46a6d0661c7
    ap-northeast-1:
      HVM64: ami-06cd52961ce9f0d85
      HVMG2: ami-053cdd503598e4a9d
    ap-northeast-2:
      HVM64: ami-0a10b2721688ce9d2
      HVMG2: NOT_SUPPORTED
    ap-northeast-3:
      HVM64: ami-0d98120a9fb693f07
      HVMG2: NOT_SUPPORTED
    ap-southeast-1:
      HVM64: ami-08569b978cc4dfa10
      HVMG2: ami-0be9df32ae9f92309
    ap-southeast-2:
      HVM64: ami-09b42976632b27e9b
      HVMG2: ami-0a9ce9fecc3d1daf8
    ap-south-1:
      HVM64: ami-0912f71e06545ad88
      HVMG2: ami-097b15e89dbdcfcf4
    us-east-2:
      HVM64: ami-0b59bfac6be064b78
      HVMG2: NOT_SUPPORTED
    ca-central-1:
      HVM64: ami-0b18956f
      HVMG2: NOT_SUPPORTED
    sa-east-1:
      HVM64: ami-07b14488da8ea02a0
      HVMG2: NOT_SUPPORTED
    cn-north-1:
      HVM64: ami-0a4eaf6c4454eda75
      HVMG2: NOT_SUPPORTED
    cn-northwest-1:
      HVM64: ami-6b6a7d09
      HVMG2: NOT_SUPPORTED
Resources:
  CloudWatchPutMetricsRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
      Statement:
      - Effect: Allow
        Principal:
          Service:
          - ec2.amazonaws.com
        Action:
        - sts:AssumeRole
    Path: "/"
CloudWatchPutMetricsRolePolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: CloudWatch_PutMetricData
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Sid: CloudWatchPutMetricData
        Effect: Allow
        Action:
        - cloudwatch:PutMetricData
        Resource:
        - "*"
    Roles:
    - Ref: CloudWatchPutMetricsRole
CloudWatchPutMetricsInstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Path: "/"
    Roles:
    - Ref: CloudWatchPutMetricsRole
VPC:
  Type: AWS::EC2::VPC
  Properties:
    EnableDnsSupport: 'true'
    EnableDnsHostnames: 'true'
    CidrBlock: 10.0.0.0/16
    Tags:
    - Key: Application
      Value:
        Ref: AWS::StackId
InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
    - Key: Application
```

```
      Value:
        Ref: AWS::StackName
    - Key: Network
      Value: Public
GatewayToInternet:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    VpcId:
      Ref: VPC
    InternetGatewayId:
      Ref: InternetGateway
RouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId:
      Ref: VPC
SubnetRouteTableAssoc:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId:
      Ref: RouteTable
    SubnetId:
      Ref: Subnet
InternetGatewayRoute:
  Type: AWS::EC2::Route
  Properties:
    DestinationCidrBlock: 0.0.0.0/0
    RouteTableId:
      Ref: RouteTable
    GatewayId:
      Ref: InternetGateway
Subnet:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId:
      Ref: VPC
    CidrBlock: 10.0.0.0/24
    Tags:
    - Key: Application
      Value:
        Ref: AWS::StackId
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
```

```
      VpcId:
        Ref: VPC
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp:
          Ref: SSHLocation
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
  MountTargetSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId:
        Ref: VPC
      GroupDescription: Security group for mount target
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 2049
        ToPort: 2049
        CidrIp: 0.0.0.0/0
  FileSystem:
    Type: AWS::EFS::FileSystem
    Properties:
      PerformanceMode: generalPurpose
      FileSystemTags:
      - Key: Name
        Value:
          Ref: VolumeName
  MountTarget:
    Type: AWS::EFS::MountTarget
    Properties:
      FileSystemId:
        Ref: FileSystem
      SubnetId:
        Ref: Subnet
      SecurityGroups:
      - Ref: MountTargetSecurityGroup
  LaunchConfiguration:
    Type: AWS::AutoScaling::LaunchConfiguration
    Metadata:
```

```
      AWS::CloudFormation::Init:
        configSets:
          MountConfig:
          - setup
          - mount
        setup:
          packages:
            yum:
              nfs-utils: []
          files:
            "/home/ec2-user/post_nfsstat":
              content: !Sub |
                #!/bin/bash

                INPUT="$(cat)"
                CW_JSON_OPEN='{ "Namespace": "EFS", "MetricData": [ '
                CW_JSON_CLOSE=' ] }'
                CW_JSON_METRIC=''
                METRIC_COUNTER=0

                for COL in 1 2 3 4 5 6; do

                 COUNTER=0
                 METRIC_FIELD=$COL
                 DATA_FIELD=$(($COL+($COL-1)))

                 while read line; do
                   if [[ COUNTER -gt 0 ]]; then

                     LINE=`echo $line | tr -s ' ' `
                     AWS_COMMAND="aws cloudwatch put-metric-data --region
 ${AWS::Region}"
                     MOD=$(( $COUNTER % 2))

                     if [ $MOD -eq 1 ]; then
                       METRIC_NAME=`echo $LINE | cut -d ' ' -f $METRIC_FIELD`
                     else
                       METRIC_VALUE=`echo $LINE | cut -d ' ' -f $DATA_FIELD`
                     fi

                     if [[ -n "$METRIC_NAME" && -n "$METRIC_VALUE" ]]; then
                       INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/
instance-id)
```

```
                        CW_JSON_METRIC="$CW_JSON_METRIC { \"MetricName\": \"$METRIC_NAME
\", \"Dimensions\": [{\"Name\": \"InstanceId\", \"Value\": \"$INSTANCE_ID\"} ], \"Value
\": $METRIC_VALUE },"
                        unset METRIC_NAME
                        unset METRIC_VALUE

                        METRIC_COUNTER=$((METRIC_COUNTER+1))
                        if [ $METRIC_COUNTER -eq 20 ]; then
                          # 20 is max metric collection size, so we have to submit here
                          aws cloudwatch put-metric-data --region ${AWS::Region} --cli-
input-json "`echo $CW_JSON_OPEN ${!CW_JSON_METRIC%?} $CW_JSON_CLOSE`"

                          # reset
                          METRIC_COUNTER=0
                          CW_JSON_METRIC=''
                        fi
                      fi



                      COUNTER=$((COUNTER+1))
                    fi

                    if [[ "$line" == "Client nfs v4:" ]]; then
                      # the next line is the good stuff
                      COUNTER=$((COUNTER+1))
                    fi
                  done <<< "$INPUT"
                done

                # submit whatever is left
                aws cloudwatch put-metric-data --region ${AWS::Region} --cli-input-json
 "`echo $CW_JSON_OPEN ${!CW_JSON_METRIC%?} $CW_JSON_CLOSE`"
              mode: '000755'
              owner: ec2-user
              group: ec2-user
            "/home/ec2-user/crontab":
              content: "* * * * * /usr/sbin/nfsstat | /home/ec2-user/post_nfsstat\n"
              owner: ec2-user
              group: ec2-user
          commands:
            01_createdir:
              command: !Sub "mkdir /${MountPoint}"
        mount:
```

```
          commands:
            01_mount:
              command: !Sub >
                mount -t nfs4 -o nfsvers=4.1 ${FileSystem}.efs.
${AWS::Region}.amazonaws.com:/ /${MountPoint}
            02_permissions:
              command: !Sub "chown ec2-user:ec2-user /${MountPoint}"
    Properties:
      AssociatePublicIpAddress: true
      ImageId:
        Fn::FindInMap:
        - AWSRegionArch2AMI
        - Ref: AWS::Region
        - Fn::FindInMap:
          - AWSInstanceType2Arch
          - Ref: InstanceType
          - Arch
      InstanceType:
        Ref: InstanceType
      KeyName:
        Ref: KeyName
      SecurityGroups:
      - Ref: InstanceSecurityGroup
      IamInstanceProfile:
        Ref: CloudWatchPutMetricsInstanceProfile
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource
 LaunchConfiguration --configsets MountConfig --region ${AWS::Region}
          crontab /home/ec2-user/crontab
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 AutoScalingGroup --region ${AWS::Region}
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    DependsOn:
    - MountTarget
    - GatewayToInternet
    CreationPolicy:
      ResourceSignal:
        Timeout: PT15M
        Count:
          Ref: AsgMaxSize
```

```
    Properties:
      VPCZoneIdentifier:
      - Ref: Subnet
      LaunchConfigurationName:
        Ref: LaunchConfiguration
      MinSize: '1'
      MaxSize:
        Ref: AsgMaxSize
      DesiredCapacity:
        Ref: AsgMaxSize
      Tags:
      - Key: Name
        Value: EFS FileSystem Mounted Instance
        PropagateAtLaunch: 'true'
Outputs:
  MountTargetID:
    Description: Mount target ID
    Value:
      Ref: MountTarget
  FileSystemID:
    Description: File system ID
    Value:
      Ref: FileSystem
```

Elastic Beanstalk template snippets

With Elastic Beanstalk, you can quickly deploy and manage applications in AWS without worrying about the infrastructure that runs those applications. The following sample template can help you describe Elastic Beanstalk resources in your CloudFormation template.

Elastic Beanstalk sample PHP

The following sample template deploys a sample PHP web application that's stored in an Amazon S3 bucket. The environment is also an auto-scaling, load-balancing environment, with a minimum of two Amazon EC2 instances and a maximum of six. It shows an Elastic Beanstalk environment that uses a legacy launch configuration. For information about using a launch template instead, see Launch Templates in the *AWS Elastic Beanstalk Developer Guide*.

Replace *`solution-stack`* with a solution stack name (platform version). For a list of available solution stacks, use the AWS CLI command **aws elasticbeanstalk list-available-solution-stacks**.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "sampleApplication": {
            "Type": "AWS::ElasticBeanstalk::Application",
            "Properties": {
                "Description": "AWS Elastic Beanstalk Sample Application"
            }
        },
        "sampleApplicationVersion": {
            "Type": "AWS::ElasticBeanstalk::ApplicationVersion",
            "Properties": {
                "ApplicationName": {
                    "Ref": "sampleApplication"
                },
                "Description": "AWS ElasticBeanstalk Sample Application Version",
                "SourceBundle": {
                    "S3Bucket": {
                        "Fn::Sub": "elasticbeanstalk-samples-${AWS::Region}"
                    },
                    "S3Key": "php-newsample-app.zip"
                }
            }
        },
        "sampleConfigurationTemplate": {
            "Type": "AWS::ElasticBeanstalk::ConfigurationTemplate",
            "Properties": {
                "ApplicationName": {
                    "Ref": "sampleApplication"
                },
                "Description": "AWS ElasticBeanstalk Sample Configuration Template",
                "OptionSettings": [
                    {
                        "Namespace": "aws:autoscaling:asg",
                        "OptionName": "MinSize",
                        "Value": "2"
                    },
                    {
                        "Namespace": "aws:autoscaling:asg",
                        "OptionName": "MaxSize",
                        "Value": "6"
                    },
```

```
                {
                    "Namespace": "aws:elasticbeanstalk:environment",
                    "OptionName": "EnvironmentType",
                    "Value": "LoadBalanced"
                },
                {
                    "Namespace": "aws:autoscaling:launchconfiguration",
                    "OptionName": "IamInstanceProfile",
                    "Value": {
                        "Ref": "MyInstanceProfile"
                    }
                }
            ],
            "SolutionStackName": "solution-stack"
        }
    },
    "sampleEnvironment": {
        "Type": "AWS::ElasticBeanstalk::Environment",
        "Properties": {
            "ApplicationName": {
                "Ref": "sampleApplication"
            },
            "Description": "AWS ElasticBeanstalk Sample Environment",
            "TemplateName": {
                "Ref": "sampleConfigurationTemplate"
            },
            "VersionLabel": {
                "Ref": "sampleApplicationVersion"
            }
        }
    },
    "MyInstanceRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ec2.amazonaws.com"
                            ]
                        },
```

```
                            "Action": [
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "Description": "Beanstalk EC2 role",
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/AWSElasticBeanstalkWebTier",
                    "arn:aws:iam::aws:policy/AWSElasticBeanstalkMulticontainerDocker",
                    "arn:aws:iam::aws:policy/AWSElasticBeanstalkWorkerTier"
                ]
            }
        },
        "MyInstanceProfile": {
            "Type": "AWS::IAM::InstanceProfile",
            "Properties": {
                "Roles": [
                    {
                        "Ref": "MyInstanceRole"
                    }
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  sampleApplication:
    Type: AWS::ElasticBeanstalk::Application
    Properties:
      Description: AWS Elastic Beanstalk Sample Application
  sampleApplicationVersion:
    Type: AWS::ElasticBeanstalk::ApplicationVersion
    Properties:
      ApplicationName:
        Ref: sampleApplication
      Description: AWS ElasticBeanstalk Sample Application Version
      SourceBundle:
        S3Bucket: !Sub "elasticbeanstalk-samples-${AWS::Region}"
```

```
        S3Key: php-newsample-app.zip
  sampleConfigurationTemplate:
    Type: AWS::ElasticBeanstalk::ConfigurationTemplate
    Properties:
      ApplicationName:
        Ref: sampleApplication
      Description: AWS ElasticBeanstalk Sample Configuration Template
      OptionSettings:
      - Namespace: aws:autoscaling:asg
        OptionName: MinSize
        Value: '2'
      - Namespace: aws:autoscaling:asg
        OptionName: MaxSize
        Value: '6'
      - Namespace: aws:elasticbeanstalk:environment
        OptionName: EnvironmentType
        Value: LoadBalanced
      - Namespace: aws:autoscaling:launchconfiguration
        OptionName: IamInstanceProfile
        Value: !Ref MyInstanceProfile
      SolutionStackName: solution-stack
  sampleEnvironment:
    Type: AWS::ElasticBeanstalk::Environment
    Properties:
      ApplicationName:
        Ref: sampleApplication
      Description: AWS ElasticBeanstalk Sample Environment
      TemplateName:
        Ref: sampleConfigurationTemplate
      VersionLabel:
        Ref: sampleApplicationVersion
  MyInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Description: Beanstalk EC2 role
```

```
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSElasticBeanstalkWebTier
        - arn:aws:iam::aws:policy/AWSElasticBeanstalkMulticontainerDocker
        - arn:aws:iam::aws:policy/AWSElasticBeanstalkWorkerTier
  MyInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref MyInstanceRole
```

Elastic Load Balancing template snippets

To create an Application Load Balancer, a Network Load Balancer, or a Gateway Load Balancer, use the V2 resource types, which start with `AWS::ElasticLoadBalancingV2`. To create a Classic Load Balancer, use the resource types that start with `AWS::ElasticLoadBalancing`.

Contents

- ELBv2 resources
- Classic Load Balancer resources

ELBv2 resources

This example defines an Application Load Balancer with an HTTP listener and a default action that forwards traffic to the target group. The load balancer uses the default health check settings. The target group has two registered EC2 instances.

YAML

```
Resources:
  myLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
```

```
  myHTTPlistener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: HTTP
      Port: 80
      DefaultActions:
        - Type: "forward"
          TargetGroupArn: !Ref myTargetGroup

  myTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Name: "my-target-group"
      Protocol: HTTP
      Port: 80
      TargetType: instance
      VpcId: !Ref myVPC
      Targets:
        - Id: !GetAtt Instance1.InstanceId
          Port: 80
        - Id: !GetAtt Instance2.InstanceId
          Port: 80
```

JSON

```
{
    "Resources": {
        "myLoadBalancer": {
            "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
            "Properties": {
                "Name": "my-alb",
                "Type": "application",
                "Scheme": "internal",
                "Subnets": [
                    {
                        "Ref": "subnet-AZ1"
                    },
                    {
                        "Ref": "subnet-AZ2"
                    }
                ],
```

```
                "SecurityGroups": [
                    {
                        "Ref": "mySecurityGroup"
                    }
                ]
            }
        },
        "myHTTPlistener": {
            "Type": "AWS::ElasticLoadBalancingV2::Listener",
            "Properties": {
                "LoadBalancerArn": {
                    "Ref": "myLoadBalancer"
                },
                "Protocol": "HTTP",
                "Port": 80,
                "DefaultActions": [
                    {
                        "Type": "forward",
                        "TargetGroupArn": {
                            "Ref": "myTargetGroup"
                        }
                    }
                ]
            }
        },
        "myTargetGroup": {
            "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
            "Properties": {
                "Name": "my-target-group",
                "Protocol": "HTTP",
                "Port": 80,
                "TargetType": "instance",
                "VpcId": {
                    "Ref": "myVPC"
                },
                "Targets": [
                    {
                        "Id": {
                            "Fn::GetAtt": [
                                "Instance1",
                                "InstanceId"
                            ]
                        },
                        "Port": 80
```

```
                    },
                    {
                        "Id": {
                            "Fn::GetAtt": [
                                "Instance2",
                                "InstanceId"
                            ]
                        },
                        "Port": 80
                    }
                ]
            }
        }
    }
}
```

Classic Load Balancer resources

This example defines a Classic Load Balancer with an HTTP listener and no registered EC2 instances. The load balancer uses the default health check settings.

YAML

```
myLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
    - "us-east-1a"
    Listeners:
    - LoadBalancerPort: '80'
      InstancePort: '80'
      Protocol: HTTP
```

JSON

```
"myLoadBalancer" : {
    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties" : {
        "AvailabilityZones" : [ "us-east-1a" ],
        "Listeners" : [ {
            "LoadBalancerPort" : "80",
            "InstancePort" : "80",
```

```
            "Protocol" : "HTTP"
        } ]
    }
}
```

This example defines a Classic Load Balancer with an HTTP listener, two registered EC2 instances, and custom health check settings.

YAML

```
myClassicLoadBalancer:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
    AvailabilityZones:
    - "us-east-1a"
    Instances:
    - Ref: Instance1
    - Ref: Instance2
    Listeners:
    - LoadBalancerPort: '80'
      InstancePort: '80'
      Protocol: HTTP
    HealthCheck:
      Target: HTTP:80/
      HealthyThreshold: '3'
      UnhealthyThreshold: '5'
      Interval: '30'
      Timeout: '5'
```

JSON

```
"myClassicLoadBalancer" : {
    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties" : {
        "AvailabilityZones" : [ "us-east-1a" ],
        "Instances" : [
            { "Ref" : "Instance1" },
            { "Ref" : "Instance2" }
        ],
        "Listeners" : [ {
            "LoadBalancerPort" : "80",
            "InstancePort" : "80",
```

```
            "Protocol" : "HTTP"
        } ],

        "HealthCheck" : {
            "Target" : "HTTP:80/",
            "HealthyThreshold" : "3",
            "UnhealthyThreshold" : "5",
            "Interval" : "30",
            "Timeout" : "5"
        }
    }
}
```

AWS Identity and Access Management template snippets

This section contains AWS Identity and Access Management template snippets.

Topics

- Declaring an IAM user resource
- Declaring an IAM access key resource
- Declaring an IAM group resource
- Adding users to a group
- Declaring an IAM policy
- Declaring an Amazon S3 bucket policy
- Declaring an Amazon SNS topic policy
- Declaring an Amazon SQS policy
- IAM role template examples

> **Important**
>
> When creating or updating a stack using a template containing IAM resources, you must acknowledge the use of IAM capabilities. For more information, see Acknowledging IAM resources in CloudFormation templates.

Declaring an IAM user resource

This snippet shows how to declare an AWS::IAM::User resource to create an IAM user. The user is declared with the path (`"/"`) and a login profile with the password (`myP@ssW0rd`).

The policy document named `giveaccesstoqueueonly` gives the user permission to perform all Amazon SQS actions on the Amazon SQS queue resource `myqueue`, and denies access to all other Amazon SQS queue resources. The `Fn::GetAtt` function gets the Arn attribute of the AWS::SQS::Queue resource `myqueue`.

The policy document named `giveaccesstotopiconly` is added to the user to give the user permission to perform all Amazon SNS actions on the Amazon SNS topic resource `mytopic` and to deny access to all other Amazon SNS resources. The `Ref` function gets the ARN of the AWS::SNS::Topic resource `mytopic`.

JSON

```
"myuser" : {
   "Type" : "AWS::IAM::User",
   "Properties" : {
      "Path" : "/",
      "LoginProfile" : {
         "Password" : "myP@ssW0rd"
      },
      "Policies" : [ {
         "PolicyName" : "giveaccesstoqueueonly",
         "PolicyDocument" : {
            "Version": "2012-10-17",
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sqs:*" ],
               "Resource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
               } ]
            }, {
               "Effect" : "Deny",
               "Action" : [ "sqs:*" ],
               "NotResource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
               } ]
            }
         ] }
```

```
      }, {
         "PolicyName" : "giveaccesstotopiconly",
         "PolicyDocument" : {
            "Version": "2012-10-17",
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sns:*" ],
               "Resource" : [ { "Ref" : "mytopic" } ]
            }, {
               "Effect" : "Deny",
               "Action" : [ "sns:*" ],
               "NotResource" : [ { "Ref" : "mytopic" } ]
            } ]
         }
      } ]
   }
}
```

YAML

```
myuser:
  Type: AWS::IAM::User
  Properties:
    Path: "/"
    LoginProfile:
      Password: myP@ssW0rd
    Policies:
    - PolicyName: giveaccesstoqueueonly
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - sqs:*
          Resource:
          - !GetAtt myqueue.Arn
        - Effect: Deny
          Action:
          - sqs:*
          NotResource:
          - !GetAtt myqueue.Arn
    - PolicyName: giveaccesstotopiconly
      PolicyDocument:
```

```
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - sns:*
          Resource:
          - !Ref mytopic
        - Effect: Deny
          Action:
          - sns:*
          NotResource:
          - !Ref mytopic
```

Declaring an IAM access key resource

This snippet shows an AWS::IAM::AccessKey resource. The `myaccesskey` resource creates an access key and assigns it to an IAM user that is declared as an AWS::IAM::User resource in the template.

JSON

```
"myaccesskey" : {
   "Type" : "AWS::IAM::AccessKey",
   "Properties" : {
      "UserName" : { "Ref" : "myuser" }
   }
}
```

YAML

```
myaccesskey:
  Type: AWS::IAM::AccessKey
  Properties:
    UserName:
      !Ref myuser
```

You can get the secret key for an `AWS::IAM::AccessKey` resource using the `Fn::GetAtt` function. One way to retrieve the secret key is to put it into an `Output` value. You can get the access key using the `Ref` function. The following `Output` value declarations get the access key and secret key for `myaccesskey`.

JSON

```
"AccessKeyformyaccesskey" : {
   "Value" : { "Ref" : "myaccesskey" }
},
"SecretKeyformyaccesskey" : {
   "Value" : {
      "Fn::GetAtt" : [ "myaccesskey", "SecretAccessKey" ]
   }
}
```

YAML

```
AccessKeyformyaccesskey:
  Value:
    !Ref myaccesskey
SecretKeyformyaccesskey:
  Value: !GetAtt myaccesskey.SecretAccessKey
```

You can also pass the AWS access key and secret key to an Amazon EC2 instance or Auto Scaling group defined in the template. The following AWS::EC2::Instance declaration uses the `UserData` property to pass the access key and secret key for the `myaccesskey` resource.

JSON

```
"myinstance" : {
   "Type" : "AWS::EC2::Instance",
   "Properties" : {
      "AvailabilityZone" : "us-east-1a",
      "ImageId" : "ami-0ff8a91507f77f867",
      "UserData" : {
         "Fn::Base64" : {
            "Fn::Join" : [
               "", [
                  "ACCESS_KEY=", {
                     "Ref" : "myaccesskey"
                  },
                  "&",
                  "SECRET_KEY=",
                  {
                     "Fn::GetAtt" : [
                        "myaccesskey",
```

```
                        "SecretAccessKey"
                    ]
                }
            ]
        ]
    }
  }
}
```

YAML

```
myinstance:
  Type: AWS::EC2::Instance
  Properties:
    AvailabilityZone: "us-east-1a"
    ImageId: ami-0ff8a91507f77f867
    UserData:
      Fn::Base64: !Sub "ACCESS_KEY=${myaccesskey}&SECRET_KEY=
${myaccesskey.SecretAccessKey}"
```

Declaring an IAM group resource

This snippet shows an AWS::IAM::Group resource. The group has a path (`"/myapplication/"`). The policy document named `myapppolicy` is added to the group to allow the group's users to perform all Amazon SQS actions on the Amazon SQS queue resource myqueue and deny access to all other Amazon SQS resources except `myqueue`.

To assign a policy to a resource, IAM requires the Amazon Resource Name (ARN) for the resource. In the snippet, the `Fn::GetAtt` function gets the ARN of the AWS::SQS::Queue resource queue.

JSON

```
"mygroup" : {
   "Type" : "AWS::IAM::Group",
   "Properties" : {
      "Path" : "/myapplication/",
      "Policies" : [ {
         "PolicyName" : "myapppolicy",
         "PolicyDocument" : {
            "Version": "2012-10-17",
            "Statement" : [ {
```

```
                "Effect" : "Allow",
                "Action" : [ "sqs:*" ],
                "Resource" : [ {
                   "Fn::GetAtt" : [ "myqueue", "Arn" ]
                } ]
             },
             {
                "Effect" : "Deny",
                "Action" : [ "sqs:*" ],
                "NotResource" : [ { "Fn::GetAtt" : [ "myqueue", "Arn" ] } ]
             }
          ] }
       } ]
   }
}
```

YAML

```
mygroup:
  Type: AWS::IAM::Group
  Properties:
    Path: "/myapplication/"
    Policies:
    - PolicyName: myapppolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - sqs:*
          Resource: !GetAtt myqueue.Arn
        - Effect: Deny
          Action:
          - sqs:*
          NotResource: !GetAtt myqueue.Arn
```

Adding users to a group

The AWS::IAM::UserToGroupAddition resource adds users to a group. In the following snippet, the addUserToGroup resource adds the following users to an existing group named myexistinggroup2: the existing user existinguser1 and the user myuser which is declared as an AWS::IAM::User resource in the template.

JSON

```
"addUserToGroup" : {
   "Type" : "AWS::IAM::UserToGroupAddition",
   "Properties" : {
      "GroupName" : "myexistinggroup2",
      "Users" : [ "existinguser1", { "Ref" : "myuser" } ]
   }
}
```

YAML

```
addUserToGroup:
  Type: AWS::IAM::UserToGroupAddition
  Properties:
    GroupName: myexistinggroup2
    Users:
    - existinguser1
    - !Ref myuser
```

Declaring an IAM policy

This snippet shows how to create a policy and apply it to multiple groups using an AWS::IAM::Policy resource named `mypolicy`. The `mypolicy` resource contains a `PolicyDocument` property that allows `GetObject`, `PutObject`, and `PutObjectAcl` actions on the objects in the S3 bucket represented by the ARN `arn:aws:s3:::myAWSBucket`. The `mypolicy` resource applies the policy to an existing group named `myexistinggroup1` and a group `mygroup` that's declared in the template as an AWS::IAM::Group resource. This example shows how to apply a policy to a group using the `Groups` property; however, you can alternatively use the `Users` property to add a policy document to a list of users.

JSON

```
"mypolicy" : {
   "Type" : "AWS::IAM::Policy",
   "Properties" : {
      "PolicyName" : "mygrouppolicy",
      "PolicyDocument" : {
         "Version": "2012-10-17",
         "Statement" : [ {
            "Effect" : "Allow",
```

```
            "Action" : [
               "s3:GetObject" , "s3:PutObject" , "s3:PutObjectAcl" ],
            "Resource" : "arn:aws:s3:::myAWSBucket/*"
         } ]
      },
      "Groups" : [ "myexistinggroup1", { "Ref" : "mygroup" } ]
   }
}
```

YAML

```
mypolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: mygrouppolicy
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Effect: Allow
        Action:
        - s3:GetObject
        - s3:PutObject
        - s3:PutObjectAcl
        Resource: arn:aws:s3:::myAWSBucket/*
    Groups:
    - myexistinggroup1
    - !Ref mygroup
```

Declaring an Amazon S3 bucket policy

This snippet shows how to create a policy and apply it to an Amazon S3 bucket using the AWS::S3::BucketPolicy resource. The `mybucketpolicy` resource declares a policy document that allows the `user1` IAM user to perform the `GetObject` action on all objects in the S3 bucket to which this policy is applied. In the snippet, the `Fn::GetAtt` function gets the ARN of the `user1` resource. The `mybucketpolicy` resource applies the policy to the `AWS::S3::BucketPolicy` resource mybucket. The `Ref` function gets the bucket name of the `mybucket` resource.

JSON

```
"mybucketpolicy" : {
   "Type" : "AWS::S3::BucketPolicy",
   "Properties" : {
```

```
      "PolicyDocument" : {
         "Id" : "MyPolicy",
         "Version": "2012-10-17",
         "Statement" : [ {
            "Sid" : "ReadAccess",
            "Action" : [ "s3:GetObject" ],
            "Effect" : "Allow",
            "Resource" : { "Fn::Join" : [
                  "", [ "arn:aws:s3:::", { "Ref" : "mybucket" } , "/*" ]
               ] },
            "Principal" : {
               "AWS" : { "Fn::GetAtt" : [ "user1", "Arn" ] }
            }
         } ]
      },
      "Bucket" : { "Ref" : "mybucket" }
   }
}
```

YAML

```
mybucketpolicy:
  Type: AWS::S3::BucketPolicy
  Properties:
    PolicyDocument:
      Id: MyPolicy
      Version: '2012-10-17'
      Statement:
      - Sid: ReadAccess
        Action:
        - s3:GetObject
        Effect: Allow
        Resource: !Sub "arn:aws:s3:::${mybucket}/*"
        Principal:
          AWS: !GetAtt user1.Arn
    Bucket: !Ref mybucket
```

Declaring an Amazon SNS topic policy

This snippet shows how to create a policy and apply it to an Amazon SNS topic using the AWS::SNS::TopicPolicy resource. The `mysnspolicy` resource contains a `PolicyDocument` property that allows the AWS::IAM::User resource `myuser` to perform the `Publish` action on an

AWS::SNS::Topic resource `mytopic`. In the snippet, the `Fn::GetAtt` function gets the ARN for the `myuser` resource and the `Ref` function gets the ARN for the `mytopic` resource.

JSON

```
"mysnspolicy" : {
   "Type" : "AWS::SNS::TopicPolicy",
   "Properties" : {
      "PolicyDocument" :  {
         "Id" : "MyTopicPolicy",
         "Version": "2012-10-17",
         "Statement" : [ {
            "Sid" : "My-statement-id",
            "Effect" : "Allow",
            "Principal" : {
               "AWS" : { "Fn::GetAtt" : [ "myuser", "Arn" ] }
            },
            "Action" : "sns:Publish",
            "Resource" : "*"
         } ]
      },
      "Topics" : [ { "Ref" : "mytopic" } ]
   }
}
```

YAML

```
mysnspolicy:
  Type: AWS::SNS::TopicPolicy
  Properties:
    PolicyDocument:
      Id: MyTopicPolicy
      Version: '2012-10-17'
      Statement:
      - Sid: My-statement-id
        Effect: Allow
        Principal:
          AWS: !GetAtt myuser.Arn
        Action: sns:Publish
        Resource: "*"
    Topics:
    - !Ref mytopic
```

Declaring an Amazon SQS policy

This snippet shows how to create a policy and apply it to an Amazon SQS queue using the AWS::SQS::QueuePolicy resource. The `PolicyDocument` property allows the existing user `myapp` (specified by its ARN) to perform the `SendMessage` action on an existing queue, which is specified by its URL, and an AWS::SQS::Queue resource myqueue. The Ref function gets the URL for the `myqueue` resource.

JSON

```
"mysqspolicy" : {
   "Type" : "AWS::SQS::QueuePolicy",
   "Properties" : {
      "PolicyDocument" : {
         "Id" : "MyQueuePolicy",
         "Version": "2012-10-17",
         "Statement" : [ {
            "Sid" : "Allow-User-SendMessage",
            "Effect" : "Allow",
            "Principal" : {
               "AWS" : "arn:aws:iam::123456789012:user/myapp"
            },
            "Action" : [ "sqs:SendMessage" ],
            "Resource" : "*"
         } ]
      },
      "Queues" : [
         "https://sqs.us-east-2aws-region.amazonaws.com/123456789012/myexistingqueue",
         { "Ref" : "myqueue" }
      ]
   }
}
```

YAML

```
mysqspolicy:
  Type: AWS::SQS::QueuePolicy
  Properties:
    PolicyDocument:
      Id: MyQueuePolicy
      Version: '2012-10-17'
      Statement:
```

```
      - Sid: Allow-User-SendMessage
        Effect: Allow
        Principal:
          AWS: arn:aws:iam::123456789012:user/myapp
        Action:
        - sqs:SendMessage
        Resource: "*"
    Queues:
    - https://sqs.aws-region.amazonaws.com/123456789012/myexistingqueue
    - !Ref myqueue
```

IAM role template examples

This section provides CloudFormation template examples for IAM roles for EC2 instances.

For more information, see IAM roles for Amazon EC2 in the *Amazon EC2 User Guide*.

IAM role with EC2

In this example, the instance profile is referenced by the `IamInstanceProfile` property of the EC2 instance. Both the instance policy and role policy reference AWS::IAM::Role.

JSON

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "myEC2Instance": {
         "Type": "AWS::EC2::Instance",
         "Version": "2009-05-15",
         "Properties": {
            "ImageId": "ami-0ff8a91507f77f867",
            "InstanceType": "m1.small",
            "Monitoring": "true",
            "DisableApiTermination": "false",
            "IamInstanceProfile": {
               "Ref": "RootInstanceProfile"
            }
         }
      },
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
```

```
            "AssumeRolePolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "root",
            "PolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
               } ]
            },
            "Roles": [ { "Ref": "RootRole" } ]
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "RootRole" } ]
         }
      }
   }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myEC2Instance:
```

```
    Type: AWS::EC2::Instance
    Version: '2009-05-15'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: m1.small
      Monitoring: 'true'
      DisableApiTermination: 'false'
      IamInstanceProfile:
        !Ref RootInstanceProfile
  RootRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: "/"
  RolePolicies:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: root
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action: "*"
          Resource: "*"
      Roles:
      - !Ref RootRole
  RootInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: "/"
      Roles:
      - !Ref RootRole
```

IAM role with Auto Scaling group

In this example, the instance profile is referenced by the `IamInstanceProfile` property of an Amazon EC2 Auto Scaling launch configuration.

JSON

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "myLCOne": {
         "Type": "AWS::AutoScaling::LaunchConfiguration",
         "Version": "2009-05-15",
         "Properties": {
            "ImageId": "ami-0ff8a91507f77f867",
            "InstanceType": "m1.small",
            "InstanceMonitoring": "true",
            "IamInstanceProfile": { "Ref": "RootInstanceProfile" }
         }
      },
      "myASGrpOne": {
         "Type": "AWS::AutoScaling::AutoScalingGroup",
         "Version": "2009-05-15",
         "Properties": {
            "AvailabilityZones": [ "us-east-1a" ],
            "LaunchConfigurationName": { "Ref": "myLCOne" },
            "MinSize": "0",
            "MaxSize": "0",
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": "120"
         }
      },
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
```

```
            },
            "Path": "/"
         }
      },
      "RolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "root",
            "PolicyDocument": {
               "Version": "2012-10-17",
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
               } ]
            },
            "Roles": [ { "Ref": "RootRole" } ]
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "RootRole" } ]
         }
      }
   }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  myLCOne:
    Type: AWS::AutoScaling::LaunchConfiguration
    Version: '2009-05-15'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: m1.small
      InstanceMonitoring: 'true'
      IamInstanceProfile:
        !Ref RootInstanceProfile
  myASGrpOne:
```

```
    Type: AWS::AutoScaling::AutoScalingGroup
    Version: '2009-05-15'
    Properties:
      AvailabilityZones:
      - "us-east-1a"
      LaunchConfigurationName:
        !Ref myLCOne
      MinSize: '0'
      MaxSize: '0'
      HealthCheckType: EC2
      HealthCheckGracePeriod: '120'
  RootRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - ec2.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: "/"
  RolePolicies:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: root
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action: "*"
          Resource: "*"
      Roles:
      - !Ref RootRole
  RootInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: "/"
      Roles:
      - !Ref RootRole
```

AWS Lambda template

The following template uses an AWS Lambda (Lambda) function and custom resource to append a new security group to a list of existing security groups. This function is useful when you want to build a list of security groups dynamically, so that your list includes both new and existing security groups. For example, you can pass a list of existing security groups as a parameter value, append the new value to the list, and then associate all your values with an EC2 instance. For more information about the Lambda function resource type, see AWS::Lambda::Function.

In the example, when CloudFormation creates the `AllSecurityGroups` custom resource, CloudFormation invokes the `AppendItemToListFunction` Lambda function. CloudFormation passes the list of existing security groups and a new security group (`NewSecurityGroup`) to the function, which appends the new security group to the list and then returns the modified list. CloudFormation uses the modified list to associate all security groups with the `MyEC2Instance` resource.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Parameters": {
        "ExistingSecurityGroups": {
            "Type": "List<AWS::EC2::SecurityGroup::Id>"
        },
        "ExistingVPC": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "The VPC ID that includes the security groups in the
 ExistingSecurityGroups parameter."
        },
        "InstanceType": {
            "Type": "String",
            "Default": "t2.micro",
            "AllowedValues": [
                "t2.micro",
                "t3.micro"
            ]
        }
    },

    "Resources": {
        "SecurityGroup": {
```

```
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Allow HTTP traffic to the host",
                "VpcId": {
                    "Ref": "ExistingVPC"
                },
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "CidrIp": "0.0.0.0/0"
                    }
                ],
                "SecurityGroupEgress": [
                    {
                        "IpProtocol": "tcp",
                        "FromPort": 80,
                        "ToPort": 80,
                        "CidrIp": "0.0.0.0/0"
                    }
                ]
            }
        },
        "AllSecurityGroups": {
            "Type": "Custom::Split",
            "Properties": {
                "ServiceToken": {
                    "Fn::GetAtt": [
                        "AppendItemToListFunction",
                        "Arn"
                    ]
                },
                "List": {
                    "Ref": "ExistingSecurityGroups"
                },
                "AppendedItem": {
                    "Ref": "SecurityGroup"
                }
            }
        },
        "AppendItemToListFunction": {
            "Type": "AWS::Lambda::Function",
            "Properties": {
```

```
                "Handler": "index.handler",
                "Role": {
                    "Fn::GetAtt": [
                        "LambdaExecutionRole",
                        "Arn"
                    ]
                },
                "Code": {
                    "ZipFile": {
                        "Fn::Join": [
                            "",
                            [
                                "var response = require('cfn-response');",
                                "exports.handler = function(event, context) {",
                                "   var responseData = {Value:
 event.ResourceProperties.List};",
                                "
 responseData.Value.push(event.ResourceProperties.AppendedItem);",
                                "   response.send(event, context, response.SUCCESS,
 responseData);",
                                "};"
                            ]
                        ]
                    }
                },
                "Runtime": "nodejs20.x"
            }
        },
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-
ami-hvm-x86_64-gp2}}",
                "SecurityGroupIds": {
                    "Fn::GetAtt": [
                        "AllSecurityGroups",
                        "Value"
                    ]
                },
                "InstanceType": {
                    "Ref": "InstanceType"
                }
            }
        },
```

```
        "LambdaExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "lambda.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "root",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "logs:*"
                                    ],
                                    "Resource": "arn:aws:logs:*:*:*"
                                }
                            ]
                        }
                    }
                ]
            }
        }
    },
    "Outputs": {
        "AllSecurityGroups": {
            "Description": "Security Groups that are associated with the EC2 instance",
            "Value": {
```

```
                "Fn::Join": [
                    ", ",
                    {
                        "Fn::GetAtt": [
                            "AllSecurityGroups",
                            "Value"
                        ]
                    }
                ]
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  ExistingSecurityGroups:
    Type: List<AWS::EC2::SecurityGroup::Id>
  ExistingVPC:
    Type: AWS::EC2::VPC::Id
    Description: The VPC ID that includes the security groups in the
 ExistingSecurityGroups parameter.
  InstanceType:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t3.micro
Resources:
  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP traffic to the host
      VpcId: !Ref ExistingVPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      SecurityGroupEgress:
        - IpProtocol: tcp
```

```
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
  AllSecurityGroups:
    Type: Custom::Split
    Properties:
      ServiceToken: !GetAtt AppendItemToListFunction.Arn
      List: !Ref ExistingSecurityGroups
      AppendedItem: !Ref SecurityGroup
  AppendItemToListFunction:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile: !Join
          - ''
          - - var response = require('cfn-response');
            - exports.handler = function(event, context) {
            - '   var responseData = {Value: event.ResourceProperties.List};'
            - '   responseData.Value.push(event.ResourceProperties.AppendedItem);'
            - '   response.send(event, context, response.SUCCESS, responseData);'
            - '};'
      Runtime: nodejs20.x
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: '{{resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2}}'
      SecurityGroupIds: !GetAtt AllSecurityGroups.Value
      InstanceType: !Ref InstanceType
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
```

```
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - logs:*
                Resource: arn:aws:logs:*:*:*
Outputs:
  AllSecurityGroups:
    Description: Security Groups that are associated with the EC2 instance
    Value: !Join
      - ', '
      - !GetAtt AllSecurityGroups.Value
```

Amazon Redshift template snippets

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. You can use CloudFormation to provision and manage Amazon Redshift clusters.

Amazon Redshift cluster

The following sample template creates an Amazon Redshift cluster according to the parameter values that are specified when the stack is created. The cluster parameter group that is associated with the Amazon Redshift cluster enables user activity logging. The template also launches the Amazon Redshift clusters in an Amazon VPC that is defined in the template. The VPC includes an internet gateway so that you can access the Amazon Redshift clusters from the Internet. However, the communication between the cluster and the Internet gateway must also be enabled, which is done by the route table entry.

> **Note**
>
> The template includes the `IsMultiNodeCluster` condition so that the `NumberOfNodes` parameter is declared only when the `ClusterType` parameter value is set to `multi-node`.

The example defines the `MysqlRootPassword` parameter with its `NoEcho` property set to `true`. If you set the `NoEcho` attribute to `true`, CloudFormation returns the parameter value masked as

asterisks (*****) for any calls that describe the stack or stack events, except for information stored in the locations specified below.

Important

Using the `NoEcho` attribute does not mask any information stored in the following:

- The `Metadata` template section. CloudFormation does not transform, modify, or redact any information you include in the `Metadata` section. For more information, see Metadata.
- The `Outputs` template section. For more information, see Outputs.
- The `Metadata` attribute of a resource definition. For more information, see `Metadata` attribute.

We strongly recommend you do not use these mechanisms to include sensitive information, such as passwords or secrets.

Important

Rather than embedding sensitive information directly in your CloudFormation templates, we recommend you use dynamic parameters in the stack template to reference sensitive information that is stored and managed outside of CloudFormation, such as in the AWS Systems Manager Parameter Store or AWS Secrets Manager.
For more information, see the Do not embed credentials in your templates best practice.

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters" : {
    "DatabaseName" : {
      "Description" : "The name of the first database to be created when the cluster is
 created",
      "Type" : "String",
      "Default" : "dev",
      "AllowedPattern" : "([a-z]|[0-9])+"
    },
```

```
    "ClusterType" : {
      "Description" : "The type of cluster",
      "Type" : "String",
      "Default" : "single-node",
      "AllowedValues" : [ "single-node", "multi-node" ]
    },
    "NumberOfNodes" : {
      "Description" : "The number of compute nodes in the cluster. For multi-node
 clusters, the NumberOfNodes parameter must be greater than 1",
      "Type" : "Number",
      "Default" : "1"
    },
    "NodeType" : {
      "Description" : "The type of node to be provisioned",
      "Type" : "String",
      "Default" : "ds2.xlarge",
      "AllowedValues" : [ "ds2.xlarge", "ds2.8xlarge", "dc1.large", "dc1.8xlarge" ]
    },
    "MasterUsername" : {
      "Description" : "The user name that is associated with the master user account
 for the cluster that is being created",
      "Type" : "String",
      "Default" : "defaultuser",
      "AllowedPattern" : "([a-z])([a-z]|[0-9])*"
    },
    "MasterUserPassword" :  {
      "Description" : "The password that is associated with the master user account for
 the cluster that is being created.",
      "Type" : "String",
      "NoEcho" : "true"
    },
    "InboundTraffic" : {
      "Description" : "Allow inbound traffic to the cluster from this CIDR range.",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" : "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\
\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },
    "PortNumber" : {
      "Description" : "The port number on which the cluster accepts incoming
 connections.",
```

```
      "Type" : "Number",
      "Default" : "5439"
    }
  },
  "Conditions" : {
    "IsMultiNodeCluster" : {
      "Fn::Equals" : [{ "Ref" : "ClusterType" }, "multi-node" ]
    }
  },
  "Resources" : {
    "RedshiftCluster" : {
      "Type" : "AWS::Redshift::Cluster",
      "DependsOn" : "AttachGateway",
      "Properties" : {
        "ClusterType" : { "Ref" : "ClusterType" },
        "NumberOfNodes" : { "Fn::If" : [ "IsMultiNodeCluster",  { "Ref" :
"NumberOfNodes" }, { "Ref" : "AWS::NoValue" }]},
        "NodeType" : { "Ref" : "NodeType" },
        "DBName" : { "Ref" : "DatabaseName" },
        "MasterUsername" : { "Ref" : "MasterUsername" },
        "MasterUserPassword" : { "Ref" : "MasterUserPassword" },
        "ClusterParameterGroupName" : { "Ref" : "RedshiftClusterParameterGroup" },
        "VpcSecurityGroupIds" : [ { "Ref" : "SecurityGroup" } ],
        "ClusterSubnetGroupName" : { "Ref" : "RedshiftClusterSubnetGroup" },
        "PubliclyAccessible" : "true",
        "Port" : { "Ref" : "PortNumber" }
      }
    },
    "RedshiftClusterParameterGroup" : {
      "Type" : "AWS::Redshift::ClusterParameterGroup",
      "Properties" : {
        "Description" : "Cluster parameter group",
        "ParameterGroupFamily" : "redshift-1.0",
        "Parameters" : [{
          "ParameterName" : "enable_user_activity_logging",
          "ParameterValue" : "true"
        }]
      }
    },
    "RedshiftClusterSubnetGroup" : {
      "Type" : "AWS::Redshift::ClusterSubnetGroup",
      "Properties" : {
        "Description" : "Cluster subnet group",
        "SubnetIds" : [ { "Ref" : "PublicSubnet" } ]
```

```
      }
    },
    "VPC" : {
      "Type" : "AWS::EC2::VPC",
      "Properties" : {
        "CidrBlock" : "10.0.0.0/16"
      }
    },
    "PublicSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "CidrBlock" : "10.0.0.0/24",
        "VpcId" : { "Ref" : "VPC" }
      }
    },
    "SecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Security group",
        "SecurityGroupIngress" : [ {
          "CidrIp" : { "Ref": "InboundTraffic" },
          "FromPort" : { "Ref" : "PortNumber" },
          "ToPort" : { "Ref" : "PortNumber" },
          "IpProtocol" : "tcp"
        } ],
        "VpcId" : { "Ref" : "VPC" }
      }
    },
    "myInternetGateway" : {
      "Type" : "AWS::EC2::InternetGateway"
    },
    "AttachGateway" : {
      "Type" : "AWS::EC2::VPCGatewayAttachment",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "InternetGatewayId" : { "Ref" : "myInternetGateway" }
      }
    },
    "PublicRouteTable" : {
      "Type" : "AWS::EC2::RouteTable",
      "Properties" : {
        "VpcId" : {
          "Ref" : "VPC"
        }
```

```
      }
    },
    "PublicRoute" : {
      "Type" : "AWS::EC2::Route",
      "DependsOn" : "AttachGateway",
      "Properties"  : {
        "RouteTableId" : {
          "Ref" : "PublicRouteTable"
        },
        "DestinationCidrBlock" : "0.0.0.0/0",
        "GatewayId" : {
          "Ref" : "myInternetGateway"
        }
      }
    },
    "PublicSubnetRouteTableAssociation" : {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "SubnetId" : {
          "Ref" : "PublicSubnet"
        },
        "RouteTableId" : {
          "Ref" : "PublicRouteTable"
        }
      }
    }
  },
  "Outputs" : {
    "ClusterEndpoint" : {
      "Description" : "Cluster endpoint",
      "Value" : { "Fn::Join" : [ ":", [ { "Fn::GetAtt" : [ "RedshiftCluster",
"Endpoint.Address" ] }, { "Fn::GetAtt" : [ "RedshiftCluster",
"Endpoint.Port" ] } ] ] }
    },
    "ClusterName" : {
      "Description" : "Name of cluster",
      "Value" : { "Ref" : "RedshiftCluster" }
    },
    "ParameterGroupName" : {
      "Description" : "Name of parameter group",
      "Value" : { "Ref" : "RedshiftClusterParameterGroup" }
    },
    "RedshiftClusterSubnetGroupName" : {
      "Description" : "Name of cluster subnet group",
```

```
      "Value" : { "Ref" : "RedshiftClusterSubnetGroup" }
    },
    "RedshiftClusterSecurityGroupName" : {
      "Description" : "Name of cluster security group",
      "Value" : { "Ref" : "SecurityGroup" }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  DatabaseName:
    Description: The name of the first database to be created when the cluster is
      created
    Type: String
    Default: dev
    AllowedPattern: "([a-z]|[0-9])+"
  ClusterType:
    Description: The type of cluster
    Type: String
    Default: single-node
    AllowedValues:
    - single-node
    - multi-node
  NumberOfNodes:
    Description: The number of compute nodes in the cluster. For multi-node clusters,
      the NumberOfNodes parameter must be greater than 1
    Type: Number
    Default: '1'
  NodeType:
    Description: The type of node to be provisioned
    Type: String
    Default: ds2.xlarge
    AllowedValues:
    - ds2.xlarge
    - ds2.8xlarge
    - dc1.large
    - dc1.8xlarge
  MasterUsername:
    Description: The user name that is associated with the master user account for
      the cluster that is being created
```

```
    Type: String
    Default: defaultuser
    AllowedPattern: "([a-z])([a-z]|[0-9])*"
  MasterUserPassword:
    Description: The password that is associated with the master user account for
      the cluster that is being created.
    Type: String
    NoEcho: 'true'
  InboundTraffic:
    Description: Allow inbound traffic to the cluster from this CIDR range.
    Type: String
    MinLength: '9'
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})"
    ConstraintDescription: must be a valid CIDR range of the form x.x.x.x/x.
  PortNumber:
    Description: The port number on which the cluster accepts incoming connections.
    Type: Number
    Default: '5439'
Conditions:
  IsMultiNodeCluster:
    Fn::Equals:
    - Ref: ClusterType
    - multi-node
Resources:
  RedshiftCluster:
    Type: AWS::Redshift::Cluster
    DependsOn: AttachGateway
    Properties:
      ClusterType:
        Ref: ClusterType
      NumberOfNodes:
        Fn::If:
        - IsMultiNodeCluster
        - Ref: NumberOfNodes
        - Ref: AWS::NoValue
      NodeType:
        Ref: NodeType
      DBName:
        Ref: DatabaseName
      MasterUsername:
        Ref: MasterUsername
      MasterUserPassword:
```

```
      Ref: MasterUserPassword
    ClusterParameterGroupName:
      Ref: RedshiftClusterParameterGroup
    VpcSecurityGroupIds:
    - Ref: SecurityGroup
    ClusterSubnetGroupName:
      Ref: RedshiftClusterSubnetGroup
    PubliclyAccessible: 'true'
    Port:
      Ref: PortNumber
RedshiftClusterParameterGroup:
  Type: AWS::Redshift::ClusterParameterGroup
  Properties:
    Description: Cluster parameter group
    ParameterGroupFamily: redshift-1.0
    Parameters:
    - ParameterName: enable_user_activity_logging
      ParameterValue: 'true'
RedshiftClusterSubnetGroup:
  Type: AWS::Redshift::ClusterSubnetGroup
  Properties:
    Description: Cluster subnet group
    SubnetIds:
    - Ref: PublicSubnet
VPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: 10.0.0.0/16
PublicSubnet:
  Type: AWS::EC2::Subnet
  Properties:
    CidrBlock: 10.0.0.0/24
    VpcId:
      Ref: VPC
SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Security group
    SecurityGroupIngress:
    - CidrIp:
        Ref: InboundTraffic
      FromPort:
        Ref: PortNumber
      ToPort:
```

```
          Ref: PortNumber
        IpProtocol: tcp
      VpcId:
        Ref: VPC
  myInternetGateway:
    Type: AWS::EC2::InternetGateway
  AttachGateway:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId:
        Ref: VPC
      InternetGatewayId:
        Ref: myInternetGateway
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId:
        Ref: VPC
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: AttachGateway
    Properties:
      RouteTableId:
        Ref: PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId:
        Ref: myInternetGateway
  PublicSubnetRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId:
        Ref: PublicSubnet
      RouteTableId:
        Ref: PublicRouteTable
Outputs:
  ClusterEndpoint:
    Description: Cluster endpoint
    Value: !Sub "${RedshiftCluster.Endpoint.Address}:${RedshiftCluster.Endpoint.Port}"
  ClusterName:
    Description: Name of cluster
    Value:
      Ref: RedshiftCluster
  ParameterGroupName:
    Description: Name of parameter group
```

```
    Value:
      Ref: RedshiftClusterParameterGroup
  RedshiftClusterSubnetGroupName:
    Description: Name of cluster subnet group
    Value:
      Ref: RedshiftClusterSubnetGroup
  RedshiftClusterSecurityGroupName:
    Description: Name of cluster security group
    Value:
      Ref: SecurityGroup
```

See also

AWS::Redshift::Cluster

Amazon RDS template snippets

Topics

- Amazon RDS DB instance resource
- Amazon RDS oracle database DB instance resource
- Amazon RDS DBSecurityGroup resource for CIDR range
- Amazon RDS DBSecurityGroup with an Amazon EC2 security group
- Multiple VPC security groups
- Amazon RDS database instance in a VPC security group

Amazon RDS DB instance resource

This example shows an Amazon RDS DB Instance resource with managed master user password. For more information, see Password management with AWS Secrets Manager in the *Amazon RDS User Guide* and Password management with AWS Secrets Manager in the *Aurora User Guide*. Because the optional `EngineVersion` property isn't specified, the default engine version is used for this DB Instance. For details about the default engine version and other default settings, see CreateDBInstance. The `DBSecurityGroups` property authorizes network ingress to the `AWS::RDS::DBSecurityGroup` resources named `MyDbSecurityByEC2SecurityGroup` and MyDbSecurityByCIDRIPGroup. For details, see AWS::RDS::DBInstance. The DB Instance resource also has a `DeletionPolicy` attribute set to `Snapshot`. With the `Snapshot` `DeletionPolicy` set, CloudFormation will take a snapshot of this DB Instance before deleting it during stack deletion.

JSON

```
"MyDB" : {
 "Type" : "AWS::RDS::DBInstance",
 "Properties" : {
     "DBSecurityGroups" : [
        {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" :
 "MyDbSecurityByCIDRIPGroup"} ],
     "AllocatedStorage" : "5",
     "DBInstanceClass" : "db.t2.small",
     "Engine" : "MySQL",
     "MasterUsername" : "MyName",
     "ManageMasterUserPassword" : true,
     "MasterUserSecret" : {
        "KmsKeyId" : {"Ref" : "KMSKey"}
     }
 },
 "DeletionPolicy" : "Snapshot"
}
```

YAML

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    DBSecurityGroups:
    - Ref: MyDbSecurityByEC2SecurityGroup
    - Ref: MyDbSecurityByCIDRIPGroup
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: MySQL
    MasterUsername: MyName
    ManageMasterUserPassword: true
    MasterUserSecret:
      KmsKeyId: !Ref KMSKey
  DeletionPolicy: Snapshot
```

Amazon RDS oracle database DB instance resource

This example creates an Oracle Database DB Instance resource with managed master user password. For more information, see Password management with AWS Secrets Manager in the *Amazon RDS User Guide*. The example specifies the `Engine` as `oracle-ee` with a license model

of bring-your-own-license. For details about the settings for Oracle Database DB instances, see CreateDBInstance. The DBSecurityGroups property authorizes network ingress to the AWS::RDS::DBSecurityGroup resources named MyDbSecurityByEC2SecurityGroup and MyDbSecurityByCIDRIPGroup. For details, see AWS::RDS::DBInstance. The DB Instance resource also has a DeletionPolicy attribute set to Snapshot. With the Snapshot DeletionPolicy set, CloudFormation will take a snapshot of this DB Instance before deleting it during stack deletion.

JSON

```
"MyDB" : {
 "Type" : "AWS::RDS::DBInstance",
 "Properties" : {
     "DBSecurityGroups" : [
        {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" :
 "MyDbSecurityByCIDRIPGroup"} ],
     "AllocatedStorage" : "5",
     "DBInstanceClass" : "db.t2.small",
     "Engine" : "oracle-ee",
     "LicenseModel" : "bring-your-own-license",
     "MasterUsername" : "master",
     "ManageMasterUserPassword" : true,
     "MasterUserSecret" : {
        "KmsKeyId" : {"Ref" : "KMSKey"}
     }
 },
 "DeletionPolicy" : "Snapshot"
}
```

YAML

```
MyDB:
  Type: AWS::RDS::DBInstance
  Properties:
    DBSecurityGroups:
    - Ref: MyDbSecurityByEC2SecurityGroup
    - Ref: MyDbSecurityByCIDRIPGroup
    AllocatedStorage: '5'
    DBInstanceClass: db.t2.small
    Engine: oracle-ee
    LicenseModel: bring-your-own-license
    MasterUsername: master
    ManageMasterUserPassword: true
```

```
    MasterUserSecret:
      KmsKeyId: !Ref KMSKey
  DeletionPolicy: Snapshot
```

Amazon RDS DBSecurityGroup resource for CIDR range

This example shows an Amazon RDS `DBSecurityGroup` resource with ingress authorization for the specified CIDR range in the format `ddd.ddd.ddd.ddd/dd`. For details, see AWS::RDS::DBSecurityGroup and Ingress.

JSON

```
"MyDbSecurityByCIDRIPGroup" : {
 "Type" : "AWS::RDS::DBSecurityGroup",
 "Properties" : {
     "GroupDescription" : "Ingress for CIDRIP",
     "DBSecurityGroupIngress" : {
         "CIDRIP" : "192.168.0.0/32"
     }
 }
}
```

YAML

```
MyDbSecurityByCIDRIPGroup:
  Type: AWS::RDS::DBSecurityGroup
  Properties:
    GroupDescription: Ingress for CIDRIP
    DBSecurityGroupIngress:
      CIDRIP: "192.168.0.0/32"
```

Amazon RDS DBSecurityGroup with an Amazon EC2 security group

This example shows an AWS::RDS::DBSecurityGroup resource with ingress authorization from an Amazon EC2 security group referenced by `MyEc2SecurityGroup`.

To do this, you define an EC2 security group and then use the intrinsic `Ref` function to refer to the EC2 security group within your `DBSecurityGroup`.

JSON

```
"DBInstance" : {
   "Type": "AWS::RDS::DBInstance",
   "Properties": {
      "DBName"            : { "Ref" : "DBName" },
      "Engine"            : "MySQL",
      "MasterUsername"    : { "Ref" : "DBUsername" },
      "DBInstanceClass"   : { "Ref" : "DBClass" },
      "DBSecurityGroups"  : [ { "Ref" : "DBSecurityGroup" } ],
      "AllocatedStorage"  : { "Ref" : "DBAllocatedStorage" },
      "MasterUserPassword": { "Ref" : "DBPassword" }
   }
},

"DBSecurityGroup": {
   "Type": "AWS::RDS::DBSecurityGroup",
   "Properties": {
      "DBSecurityGroupIngress": {
         "EC2SecurityGroupName": {
            "Fn::GetAtt": ["WebServerSecurityGroup", "GroupName"]
         }
      },
      "GroupDescription" : "Frontend Access"
   }
},

"WebServerSecurityGroup" : {
   "Type" : "AWS::EC2::SecurityGroup",
   "Properties" : {
      "GroupDescription" : "Enable HTTP access via port 80 and SSH access",
      "SecurityGroupIngress" : [
         {"IpProtocol" : "tcp", "FromPort" : 80, "ToPort" : 80, "CidrIp" :
 "0.0.0.0/0"},
         {"IpProtocol" : "tcp", "FromPort" : 22, "ToPort" : 22, "CidrIp" : "0.0.0.0/0"}
      ]
   }
}
```

YAML

This example is extracted from the following full example: Drupal_Single_Instance_With_RDS.template

```
DBInstance:
```

```
  Type: AWS::RDS::DBInstance
  Properties:
    DBName:
      Ref: DBName
    Engine: MySQL
    MasterUsername:
      Ref: DBUsername
    DBInstanceClass:
      Ref: DBClass
    DBSecurityGroups:
    - Ref: DBSecurityGroup
    AllocatedStorage:
      Ref: DBAllocatedStorage
    MasterUserPassword:
      Ref: DBPassword
DBSecurityGroup:
  Type: AWS::RDS::DBSecurityGroup
  Properties:
    DBSecurityGroupIngress:
      EC2SecurityGroupName:
        Ref: WebServerSecurityGroup
    GroupDescription: Frontend Access
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable HTTP access via port 80 and SSH access
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 80
      ToPort: 80
      CidrIp: 0.0.0.0/0
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: 0.0.0.0/0
```

Multiple VPC security groups

This example shows an AWS::RDS::DBSecurityGroup resource with ingress authorization for multiple Amazon EC2 VPC security groups in AWS::RDS::DBSecurityGroupIngress.

JSON

```
{
   "Resources" : {
      "DBinstance" : {
         "Type" : "AWS::RDS::DBInstance",
         "Properties" : {
            "AllocatedStorage" : "5",
            "DBInstanceClass" : "db.t2.small",
           "DBName" : {"Ref": "MyDBName" },
            "DBSecurityGroups" : [ { "Ref" : "DbSecurityByEC2SecurityGroup" } ],
            "DBSubnetGroupName" : { "Ref" : "MyDBSubnetGroup" },
            "Engine" : "MySQL",
           "MasterUserPassword": { "Ref" : "MyDBPassword" },
           "MasterUsername"    : { "Ref" : "MyDBUsername" }
         },
          "DeletionPolicy" : "Snapshot"
      },
      "DbSecurityByEC2SecurityGroup" : {
         "Type" : "AWS::RDS::DBSecurityGroup",
         "Properties" : {
            "GroupDescription" : "Ingress for Amazon EC2 security group",
           "EC2VpcId" : { "Ref" : "MyVPC" },
            "DBSecurityGroupIngress" : [ {
               "EC2SecurityGroupId" : "sg-b0ff1111",
               "EC2SecurityGroupOwnerId" : "111122223333"
            }, {
               "EC2SecurityGroupId" : "sg-ffd722222",
               "EC2SecurityGroupOwnerId" : "111122223333"
            } ]
         }
      }
   }
}
```

YAML

```
Resources:
  DBinstance:
    Type: AWS::RDS::DBInstance
    Properties:
      AllocatedStorage: '5'
      DBInstanceClass: db.t2.small
```

```
      DBName:
        Ref: MyDBName
      DBSecurityGroups:
      - Ref: DbSecurityByEC2SecurityGroup
      DBSubnetGroupName:
        Ref: MyDBSubnetGroup
      Engine: MySQL
      MasterUserPassword:
        Ref: MyDBPassword
      MasterUsername:
        Ref: MyDBUsername
    DeletionPolicy: Snapshot
  DbSecurityByEC2SecurityGroup:
    Type: AWS::RDS::DBSecurityGroup
    Properties:
      GroupDescription: Ingress for Amazon EC2 security group
      EC2VpcId:
        Ref: MyVPC
      DBSecurityGroupIngress:
      - EC2SecurityGroupId: sg-b0ff1111
        EC2SecurityGroupOwnerId: '111122223333'
      - EC2SecurityGroupId: sg-ffd722222
        EC2SecurityGroupOwnerId: '111122223333'
```

Amazon RDS database instance in a VPC security group

This example shows an Amazon RDS database instance associated with an Amazon EC2 VPC security group.

JSON

```
{
  "DBEC2SecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties" : {
      "GroupDescription": "Open database for access",
      "SecurityGroupIngress" : [{
        "IpProtocol" : "tcp",
        "FromPort" : 3306,
        "ToPort" : 3306,
        "SourceSecurityGroupName" : { "Ref" : "WebServerSecurityGroup" }
      }]
    }
```

```
  },
  "DBInstance" : {
    "Type": "AWS::RDS::DBInstance",
    "Properties": {
      "DBName"            : { "Ref" : "DBName" },
      "Engine"            : "MySQL",
      "MultiAZ"           : { "Ref": "MultiAZDatabase" },
      "MasterUsername"    : { "Ref" : "DBUser" },
      "DBInstanceClass"   : { "Ref" : "DBClass" },
      "AllocatedStorage"  : { "Ref" : "DBAllocatedStorage" },
      "MasterUserPassword": { "Ref" : "DBPassword" },
      "VPCSecurityGroups" : [ { "Fn::GetAtt": [ "DBEC2SecurityGroup", "GroupId" ] } ]
    }
  }
}
```

YAML

```
DBEC2SecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Open database for access
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 3306
      ToPort: 3306
      SourceSecurityGroupName:
        Ref: WebServerSecurityGroup
DBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName:
      Ref: DBName
    Engine: MySQL
    MultiAZ:
      Ref: MultiAZDatabase
    MasterUsername:
      Ref: DBUser
    DBInstanceClass:
      Ref: DBClass
    AllocatedStorage:
      Ref: DBAllocatedStorage
    MasterUserPassword:
```

```
      Ref: DBPassword
    VPCSecurityGroups:
    - !GetAtt DBEC2SecurityGroup.GroupId
```

Route 53 template snippets

Topics

- Amazon Route 53 resource record set using hosted zone name or ID
- Using RecordSetGroup to set up weighted resource record sets
- Using RecordSetGroup to set up an alias resource record set
- Alias resource record set for a CloudFront distribution

Amazon Route 53 resource record set using hosted zone name or ID

When you create an Amazon Route 53 resource record set, you must specify the hosted zone where you want to add it. CloudFormation provides two ways to specify a hosted zone:

- You can explicitly specify the hosted zone using the `HostedZoneId` property.
- You can have CloudFormation find the hosted zone using the `HostedZoneName` property. If you use the `HostedZoneName` property and there are multiple hosted zones with the same name, CloudFormation doesn't create the stack.

Adding RecordSet using HostedZoneId

This example adds an Amazon Route 53 resource record set containing an SPF record for the domain name `mysite.example.com` that uses the `HostedZoneId` property to specify the hosted zone.

JSON

```
"myDNSRecord" : {
  "Type" : "AWS::Route53::RecordSet",
  "Properties" :
  {
    "HostedZoneId" : "Z3DG6IL3SJCGPX",
    "Name" : "mysite.example.com.",
    "Type" : "SPF",
    "TTL" : "900",
```

```
    "ResourceRecords" : [ "\"v=spf1 ip4:192.168.0.1/16 -all\"" ]
  }
}
```

YAML

```
myDNSRecord:
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: Z3DG6IL3SJCGPX
    Name: mysite.example.com.
    Type: SPF
    TTL: '900'
    ResourceRecords:
    - '"v=spf1 ip4:192.168.0.1/16 -all"'
```

Adding RecordSet using HostedZoneName

This example adds an Amazon Route 53 resource record set for the domain name "mysite.example.com" using the `HostedZoneName` property to specify the hosted zone.

JSON

```
"myDNSRecord2" : {
            "Type" : "AWS::Route53::RecordSet",
            "Properties" : {
                "HostedZoneName" : "example.com.",
                "Name" : "mysite.example.com.",
                "Type" : "A",
                "TTL" : "900",
                "ResourceRecords" : [
                    "192.168.0.1",
                    "192.168.0.2"
                ]
            }
        }
```

YAML

```
myDNSRecord2:
  Type: AWS::Route53::RecordSet
  Properties:
```

```
    HostedZoneName: example.com.
    Name: mysite.example.com.
    Type: A
    TTL: '900'
    ResourceRecords:
    - 192.168.0.1
    - 192.168.0.2
```

Using RecordSetGroup to set up weighted resource record sets

This example uses an AWS::Route53::RecordSetGroup to set up two CNAME records for the "example.com." hosted zone. The `RecordSets` property contains the CNAME record sets for the "mysite.example.com" DNS name. Each record set contains an identifier (`SetIdentifier`) and weight (`Weight`). The proportion of internet traffic that's routed to the resources is based on the following calculations:

- `Frontend One`: `140/(140+60)` = `140/200` = 70%
- `Frontend Two`: `60/(140+60)` = `60/200` = 30%

For more information about weighted resource record sets, see Weighted routing in the *Amazon Route 53 Developer Guide*.

JSON

```
       "myDNSOne" : {
           "Type" : "AWS::Route53::RecordSetGroup",
           "Properties" : {
               "HostedZoneName" : "example.com.",
               "Comment" : "Weighted RR for my frontends.",
               "RecordSets" : [
                 {
                   "Name" : "mysite.example.com.",
                   "Type" : "CNAME",
                   "TTL" : "900",
                   "SetIdentifier" : "Frontend One",
                   "Weight" : "140",
                   "ResourceRecords" : ["example-ec2.amazonaws.com"]
                 },
                 {
                   "Name" : "mysite.example.com.",
                   "Type" : "CNAME",
```

```
                    "TTL" : "900",
                    "SetIdentifier" : "Frontend Two",
                    "Weight" : "60",
                    "ResourceRecords" : ["example-ec2-larger.amazonaws.com"]
                  }
                  ]
            }
        }
```

YAML

```
myDNSOne:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Weighted RR for my frontends.
    RecordSets:
    - Name: mysite.example.com.
      Type: CNAME
      TTL: '900'
      SetIdentifier: Frontend One
      Weight: '140'
      ResourceRecords:
      - example-ec2.amazonaws.com
    - Name: mysite.example.com.
      Type: CNAME
      TTL: '900'
      SetIdentifier: Frontend Two
      Weight: '60'
      ResourceRecords:
      - example-ec2-larger.amazonaws.com
```

Using RecordSetGroup to set up an alias resource record set

The following examples use an AWS::Route53::RecordSetGroup to set up an alias resource record set named `example.com` that routes traffic to an ELB Version 1 (Classic) load balancer and a Version 2 (Application or Network) load balancer. The AliasTarget property specifies the hosted zone ID and DNS name for the `myELB` `LoadBalancer` by using the `GetAtt` intrinsic function. `GetAtt` retrieves different properties of `myELB` resource, depending on whether you're routing traffic to a Version 1 or Version 2 load balancer:

- Version 1 load balancer: `CanonicalHostedZoneNameID` and `DNSName`

- Version 2 load balancer: `CanonicalHostedZoneID` and `DNSName`

For more information about alias resource record sets, see Choosing between alias and non-alias records in the *Route 53 Developer Guide*.

JSON for version 1 load balancer

```
      "myELB" : {
        "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties" : {
            "AvailabilityZones" : [ "us-east-1a" ],
            "Listeners" : [ {
                "LoadBalancerPort" : "80",
                "InstancePort" : "80",
                "Protocol" : "HTTP"
            } ]
        }
      },
      "myDNS" : {
        "Type" : "AWS::Route53::RecordSetGroup",
        "Properties" : {
          "HostedZoneName" : "example.com.",
          "Comment" : "Zone apex alias targeted to myELB LoadBalancer.",
          "RecordSets" : [
            {
              "Name" : "example.com.",
              "Type" : "A",
              "AliasTarget" : {
                  "HostedZoneId" : { "Fn::GetAtt" : ["myELB",
 "CanonicalHostedZoneNameID"] },
                  "DNSName" : { "Fn::GetAtt" : ["myELB","DNSName"] }
              }
            }
          ]
        }
    }
```

YAML for version 1 load balancer

```
myELB:
  Type: AWS::ElasticLoadBalancing::LoadBalancer
  Properties:
```

```
    AvailabilityZones:
    - "us-east-1a"
    Listeners:
    - LoadBalancerPort: '80'
      InstancePort: '80'
      Protocol: HTTP
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Zone apex alias targeted to myELB LoadBalancer.
    RecordSets:
    - Name: example.com.
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt 'myELB.CanonicalHostedZoneNameID'
        DNSName: !GetAtt 'myELB.DNSName'
```

JSON for version 2 load balancer

```
      "myELB" : {
        "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties" : {
            "Subnets" : [
                {"Ref": "SubnetAZ1"},
                {"Ref" : "SubnetAZ2"}
            ]
        }
      },
      "myDNS" : {
        "Type" : "AWS::Route53::RecordSetGroup",
        "Properties" : {
          "HostedZoneName" : "example.com.",
          "Comment" : "Zone apex alias targeted to myELB LoadBalancer.",
          "RecordSets" : [
            {
              "Name" : "example.com.",
              "Type" : "A",
              "AliasTarget" : {
                  "HostedZoneId" : { "Fn::GetAtt" : ["myELB",
 "CanonicalHostedZoneID"] },
                  "DNSName" : { "Fn::GetAtt" : ["myELB","DNSName"] }
              }
```

```
            }
          ]
        }
    }
```

YAML for version 2 load balancer

```
myELB:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Subnets:
    - Ref: SubnetAZ1
    - Ref: SubnetAZ2
myDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneName: example.com.
    Comment: Zone apex alias targeted to myELB LoadBalancer.
    RecordSets:
    - Name: example.com.
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt 'myELB.CanonicalHostedZoneID'
        DNSName: !GetAtt 'myELB.DNSName'
```

Alias resource record set for a CloudFront distribution

The following example creates an alias A record that points a custom domain name to an existing CloudFront distribution. `myHostedZoneID` is assumed to be either a reference to an actual `AWS::Route53::HostedZone` resource in the same template or a parameter. `myCloudFrontDistribution` refers to an `AWS::CloudFront::Distribution` resource within the same template. The alias record uses the standard CloudFront hosted zone ID (`Z2FDTNDATAQYW2`) and automatically resolves the distribution's domain name using `Fn::GetAtt`. This setup allows web traffic to be routed from the custom domain to the CloudFront distribution without requiring an IP address.

Note

When you create alias resource record sets, you must specify `Z2FDTNDATAQYW2` for the `HostedZoneId` property. Alias resource record sets for CloudFront can't be created in a private zone.

JSON

```
{
    "myDNS": {
        "Type": "AWS::Route53::RecordSetGroup",
        "Properties": {
            "HostedZoneId": {
                "Ref": "myHostedZoneID"
            },
            "RecordSets": [
                {
                    "Name": {
                        "Ref": "myRecordSetDomainName"
                    },
                    "Type": "A",
                    "AliasTarget": {
                        "HostedZoneId": "Z2FDTNDATAQYW2",
                        "DNSName": {
                            "Fn::GetAtt": [
                                "myCloudFrontDistribution",
                                "DomainName"
                            ]
                        },
                        "EvaluateTargetHealth": false
                    }
                }
            ]
        }
    }
}
```

YAML

```
myDNS:
  Type: AWS::Route53::RecordSetGroup
```

```
  Properties:
    HostedZoneId: !Ref myHostedZoneID
    RecordSets:
      - Name: !Ref myRecordSetDomainName
        Type: A
        AliasTarget:
          HostedZoneId: Z2FDTNDATAQYW2
          DNSName: !GetAtt
            - myCloudFrontDistribution
            - DomainName
          EvaluateTargetHealth: false
```

Amazon S3 template snippets

Use these Amazon S3 sample templates to help describe your Amazon S3 buckets with CloudFormation. For more examples, see the Examples section in the `AWS::S3::Bucket` resource.

Topics

- Creating an Amazon S3 bucket with defaults
- Creating an Amazon S3 bucket for website hosting and with a DeletionPolicy
- Creating a static website using a custom domain

Creating an Amazon S3 bucket with defaults

This example uses a AWS::S3::Bucket to create a bucket with default settings.

JSON

```
"myS3Bucket" : {
      "Type" : "AWS::S3::Bucket"
      }
```

YAML

```
MyS3Bucket:
    Type: AWS::S3::Bucket
```

Creating an Amazon S3 bucket for website hosting and with a `DeletionPolicy`

This example creates a bucket as a website and disables Block Public Access (public read permissions are required for buckets set up for website hosting). A public bucket policy is then added to the bucket. Because this bucket resource has a `DeletionPolicy` attribute set to `Retain`, CloudFormation will not delete this bucket when it deletes the stack. The `Output` section uses `Fn::GetAtt` to retrieve the `WebsiteURL` attribute and `DomainName` attribute of the `S3Bucket` resource.

> **Note**
>
> The following examples assume the `BlockPublicPolicy` and `RestrictPublicBuckets` Block Public Access settings have been disabled at the account level.

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "PublicAccessBlockConfiguration": {
                    "BlockPublicAcls": false,
                    "BlockPublicPolicy": false,
                    "IgnorePublicAcls": false,
                    "RestrictPublicBuckets": false
                },
                "WebsiteConfiguration": {
                    "IndexDocument": "index.html",
                    "ErrorDocument": "error.html"
                }
            },
            "DeletionPolicy": "Retain",
            "UpdateReplacePolicy": "Retain"
        },
        "BucketPolicy": {
            "Type": "AWS::S3::BucketPolicy",
            "Properties": {
                "PolicyDocument": {
                    "Id": "MyPolicy",
```

```
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "PublicReadForGetBucketObjects",
                        "Effect": "Allow",
                        "Principal": "*",
                        "Action": "s3:GetObject",
                        "Resource": {
                            "Fn::Join": [
                                "",
                                [
                                    "arn:aws:s3:::",
                                    {
                                        "Ref": "S3Bucket"
                                    },
                                    "/*"
                                ]
                            ]
                        }
                    }
                ]
            },
            "Bucket": {
                "Ref": "S3Bucket"
            }
        }
    }
},
"Outputs": {
    "WebsiteURL": {
        "Value": {
            "Fn::GetAtt": [
                "S3Bucket",
                "WebsiteURL"
            ]
        },
        "Description": "URL for website hosted on S3"
    },
    "S3BucketSecureURL": {
        "Value": {
            "Fn::Join": [
                "",
                [
                    "https://",
```

```
                        {
                            "Fn::GetAtt": [
                                "S3Bucket",
                                "DomainName"
                            ]
                        }
                    ]
                ]
            },
            "Description": "Name of S3 bucket to hold website content"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      PublicAccessBlockConfiguration:
        BlockPublicAcls: false
        BlockPublicPolicy: false
        IgnorePublicAcls: false
        RestrictPublicBuckets: false
      WebsiteConfiguration:
        IndexDocument: index.html
        ErrorDocument: error.html
    DeletionPolicy: Retain
    UpdateReplacePolicy: Retain
  BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      PolicyDocument:
        Id: MyPolicy
        Version: 2012-10-17
        Statement:
          - Sid: PublicReadForGetBucketObjects
            Effect: Allow
            Principal: '*'
            Action: 's3:GetObject'
            Resource: !Join
```

```
              - ''
              - - 'arn:aws:s3:::'
                - !Ref S3Bucket
                - /*
      Bucket: !Ref S3Bucket
Outputs:
  WebsiteURL:
    Value: !GetAtt
      - S3Bucket
      - WebsiteURL
    Description: URL for website hosted on S3
  S3BucketSecureURL:
    Value: !Join
      - ''
      - - 'https://'
        - !GetAtt
          - S3Bucket
          - DomainName
    Description: Name of S3 bucket to hold website content
```

Creating a static website using a custom domain

You can use Route 53 with a registered domain. The following sample assumes that you have already created a hosted zone in Route 53 for your domain. The example creates two buckets for website hosting. The root bucket hosts the content, and the other bucket redirects `www.`*`domainname`*`.com` requests to the root bucket. The record sets map your domain name to Amazon S3 endpoints.

You will also need to add a bucket policy, as shown in the examples above.

For more information about using a custom domain, see Tutorial: Configuring a static website using a custom domain registered with Route 53 in the *Amazon Simple Storage Service User Guide*.

> **Note**
>
> The following examples assume the `BlockPublicPolicy` and `RestrictPublicBuckets` Block Public Access settings have been disabled at the account level.

JSON

```
{
```

```
    "AWSTemplateFormatVersion": "2010-09-09",
    "Mappings" : {
        "RegionMap" : {
            "us-east-1" : { "S3hostedzoneID" : "Z3AQBSTGFYJSTF", "websiteendpoint" :
"s3-website-us-east-1.amazonaws.com" },
            "us-west-1" : { "S3hostedzoneID" : "Z2F56UZL2M1ACD", "websiteendpoint" :
"s3-website-us-west-1.amazonaws.com" },
            "us-west-2" : { "S3hostedzoneID" : "Z3BJ6K6RIION7M", "websiteendpoint" :
"s3-website-us-west-2.amazonaws.com" },
            "eu-west-1" : { "S3hostedzoneID" : "Z1BKCTXD74EZPE", "websiteendpoint" :
"s3-website-eu-west-1.amazonaws.com" },
            "ap-southeast-1" : { "S3hostedzoneID" : "Z3O0J2DXBE1FTB",
"websiteendpoint" : "s3-website-ap-southeast-1.amazonaws.com" },
            "ap-southeast-2" : { "S3hostedzoneID" : "Z1WCIGYICN2BYD",
"websiteendpoint" : "s3-website-ap-southeast-2.amazonaws.com" },
            "ap-northeast-1" : { "S3hostedzoneID" : "Z2M4EHUR26P7ZW",
"websiteendpoint" : "s3-website-ap-northeast-1.amazonaws.com" },
            "sa-east-1" : { "S3hostedzoneID" : "Z31GFT0UA1I2HV", "websiteendpoint" :
"s3-website-sa-east-1.amazonaws.com" }
        }
    },
    "Parameters": {
        "RootDomainName": {
            "Description": "Domain name for your website (example.com)",
            "Type": "String"
        }
    },
    "Resources": {
        "RootBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName" : {"Ref":"RootDomainName"},
                "PublicAccessBlockConfiguration": {
                    "BlockPublicAcls": false,
                    "BlockPublicPolicy": false,
                    "IgnorePublicAcls": false,
                    "RestrictPublicBuckets": false
                },
                "WebsiteConfiguration": {
                    "IndexDocument":"index.html",
                    "ErrorDocument":"404.html"
                }
            }
        },
```

```
        "WWWBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "BucketName": {
                    "Fn::Join": ["", ["www.", {"Ref":"RootDomainName"}]]
                },
                "AccessControl": "BucketOwnerFullControl",
                "WebsiteConfiguration": {
                    "RedirectAllRequestsTo": {
                        "HostName": {"Ref": "RootBucket"}
                    }
                }
            }
        },
        "myDNS": {
            "Type": "AWS::Route53::RecordSetGroup",
            "Properties": {
                "HostedZoneName": {
                    "Fn::Join": ["", [{"Ref": "RootDomainName"}, "."]]
                },
                "Comment": "Zone apex alias.",
                "RecordSets": [
                    {
                        "Name": {"Ref": "RootDomainName"},
                        "Type": "A",
                        "AliasTarget": {
                            "HostedZoneId": {"Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "S3hostedzoneID"]},
                            "DNSName": {"Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "websiteendpoint"]}
                        }
                    },
                    {
                        "Name": {
                            "Fn::Join": ["", ["www.", {"Ref":"RootDomainName"}]]
                        },
                        "Type": "CNAME",
                        "TTL" : "900",
                        "ResourceRecords" : [
                            {"Fn::GetAtt":["WWWBucket", "DomainName"]}
                        ]
                    }
                ]
            }
```

```
        }
    },
    "Outputs": {
        "WebsiteURL": {
            "Value": {"Fn::GetAtt": ["RootBucket", "WebsiteURL"]},
            "Description": "URL for website hosted on S3"
        }
    }
}
```

YAML

```
Parameters:
  RootDomainName:
    Description: Domain name for your website (example.com)
    Type: String
Mappings:
  RegionMap:
    us-east-1:
      S3hostedzoneID: Z3AQBSTGFYJSTF
      websiteendpoint: s3-website-us-east-1.amazonaws.com
    us-west-1:
      S3hostedzoneID: Z2F56UZL2M1ACD
      websiteendpoint: s3-website-us-west-1.amazonaws.com
    us-west-2:
      S3hostedzoneID: Z3BJ6K6RIION7M
      websiteendpoint: s3-website-us-west-2.amazonaws.com
    eu-west-1:
      S3hostedzoneID: Z1BKCTXD74EZPE
      websiteendpoint: s3-website-eu-west-1.amazonaws.com
    ap-southeast-1:
      S3hostedzoneID: Z3O0J2DXBE1FTB
      websiteendpoint: s3-website-ap-southeast-1.amazonaws.com
    ap-southeast-2:
      S3hostedzoneID: Z1WCIGYICN2BYD
      websiteendpoint: s3-website-ap-southeast-2.amazonaws.com
    ap-northeast-1:
      S3hostedzoneID: Z2M4EHUR26P7ZW
      websiteendpoint: s3-website-ap-northeast-1.amazonaws.com
    sa-east-1:
      S3hostedzoneID: Z31GFT0UA1I2HV
      websiteendpoint: s3-website-sa-east-1.amazonaws.com
Resources:
```

```
  RootBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref RootDomainName
      PublicAccessBlockConfiguration:
        BlockPublicAcls: false
        BlockPublicPolicy: false
        IgnorePublicAcls: false
        RestrictPublicBuckets: false
      WebsiteConfiguration:
        IndexDocument: index.html
        ErrorDocument: 404.html
  WWWBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub
        - www.${Domain}
        - Domain: !Ref RootDomainName
      AccessControl: BucketOwnerFullControl
      WebsiteConfiguration:
        RedirectAllRequestsTo:
          HostName: !Ref RootBucket
  myDNS:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      HostedZoneName: !Sub
        - ${Domain}.
        - Domain: !Ref RootDomainName
      Comment: Zone apex alias.
      RecordSets:
        - Name: !Ref RootDomainName
          Type: A
          AliasTarget:
            HostedZoneId: !FindInMap [ RegionMap, !Ref 'AWS::Region', S3hostedzoneID]
            DNSName: !FindInMap [ RegionMap, !Ref 'AWS::Region', websiteendpoint]
        - Name: !Sub
            - www.${Domain}
            - Domain: !Ref RootDomainName
          Type: CNAME
          TTL: 900
          ResourceRecords:
            - !GetAtt WWWBucket.DomainName
Outputs:
  WebsiteURL:
```

```
    Value: !GetAtt RootBucket.WebsiteURL
    Description: URL for website hosted on S3
```

Amazon SNS template snippets

This example shows an Amazon SNS topic resource. It requires a valid email address.

JSON

```
"MySNSTopic" : {
    "Type" : "AWS::SNS::Topic",
    "Properties" : {
        "Subscription" : [ {
            "Endpoint" : "add valid email address",
            "Protocol" : "email"
        } ]
    }
}
```

YAML

```
MySNSTopic:
  Type: AWS::SNS::Topic
  Properties:
    Subscription:
    - Endpoint: "add valid email address"
      Protocol: email
```

Amazon SQS template snippets

This example shows an Amazon SQS queue.

JSON

```
"MyQueue" : {
    "Type" : "AWS::SQS::Queue",
    "Properties" : {
        "VisibilityTimeout" : "value"
    }
}
```

YAML

```
MyQueue:
  Type: AWS::SQS::Queue
  Properties:
    VisibilityTimeout: value
```

Amazon Timestream template snippets

Amazon Timestream for InfluxDB makes it easy for application developers and DevOps teams to run fully managed InfluxDB databases on AWS for real-time time-series applications using open-source APIs. You can quickly create an InfluxDB database that handles demanding time- series workloads. With a few simple API calls, you can set up, migrate, operate, and scale an InfluxDB database on AWS with automated software patching, backups, and recovery. You can also find these samples at awslabs/amazon-timestream-tools/tree/mainline/integrations/cloudformation/timestream-influxdb on GitHub.

Topics

- Minimal sample using default values
- More complete example with parameters

These CloudFormation templates create the following resources that are needed to successfully create, connect to, and monitor a Amazon Timestream for InfluxDB instance:

Amazon VPC

- VPC
- One or more Subnet
- InternetGateway
- RouteTable
- SecurityGroup

Amazon S3

- Bucket

Amazon Timestream

- `InfluxDBInstance`

Minimal sample using default values

This example deploys a multi- AZ and publicly accessible instance using default values when possible.

JSON

```
{
  "Metadata": {
    "AWS::CloudFormation::Interface": {
      "ParameterGroups": [
        {
          "Label": {"default": "Amazon Timestream for InfluxDB Configuration"},
          "Parameters": [
            "DbInstanceName",
            "InfluxDBPassword"
          ]
        }
      ],
      "ParameterLabels": {
        "VPCCIDR": {"default": "VPC CIDR"}
      }
    }
  },
  "Parameters": {
    "DbInstanceName": {
      "Description": "The name that uniquely identifies the DB instance when
 interacting with the Amazon Timestream for InfluxDB API and CLI commands. This name
 will also be a prefix included in the endpoint. DB instance names must be unique per
 customer and per Region.",
      "Type": "String",
      "Default": "mydbinstance",
      "MinLength": 3,
      "MaxLength": 40,
      "AllowedPattern": "^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$"
    },
    "InfluxDBPassword": {
      "Description": "The password of the initial admin user created in InfluxDB. This
 password will allow you to access the InfluxDB UI to perform various administrative
```

```
tasks and also use the InfluxDB CLI to create an operator token. These attributes will
be stored in a Secret created in AWS Secrets Manager in your account.",
     "Type": "String",
     "NoEcho": true,
     "MinLength": 8,
     "MaxLength": 64,
     "AllowedPattern": "^[a-zA-Z0-9]+$"
   }
 },
 "Resources": {
   "VPC": {
     "Type": "AWS::EC2::VPC",
     "Properties": {"CidrBlock": "10.0.0.0/16"}
   },
   "InternetGateway": {"Type": "AWS::EC2::InternetGateway"},
   "InternetGatewayAttachment": {
     "Type": "AWS::EC2::VPCGatewayAttachment",
     "Properties": {
       "InternetGatewayId": {"Ref": "InternetGateway"},
       "VpcId": {"Ref": "VPC"}
     }
   },
   "Subnet1": {
     "Type": "AWS::EC2::Subnet",
     "Properties": {
       "VpcId": {"Ref": "VPC"},
       "AvailabilityZone": {
         "Fn::Select": [
           0,
           {"Fn::GetAZs": ""}
         ]
       },
       "CidrBlock": {
         "Fn::Select": [
           0,
           {
             "Fn::Cidr": [
               {
                 "Fn::GetAtt": [
                   "VPC",
                   "CidrBlock"
                 ]
               },
               2,
```

```
              12
            ]
          }
        ]
      },
      "MapPublicIpOnLaunch": true
    }
  },
  "Subnet2": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
      "VpcId": {"Ref": "VPC"},
      "AvailabilityZone": {
        "Fn::Select": [
          1,
          {"Fn::GetAZs": ""}
        ]
      },
      "CidrBlock": {
        "Fn::Select": [
          1,
          {
            "Fn::Cidr": [
              {
                "Fn::GetAtt": [
                  "VPC",
                  "CidrBlock"
                ]
              },
              2,
              12
            ]
          }
        ]
      },
      "MapPublicIpOnLaunch": true
    }
  },
  "RouteTable": {
    "Type": "AWS::EC2::RouteTable",
    "Properties": {
      "VpcId": {"Ref": "VPC"}
    }
  },
```

```
    "DefaultRoute": {
      "Type": "AWS::EC2::Route",
      "DependsOn": "InternetGatewayAttachment",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": {"Ref": "InternetGateway"}
      }
    },
    "Subnet1RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet1"}
      }
    },
    "Subnet2RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet2"}
      }
    },
    "InfluxDBSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupName": "influxdb-sg",
        "GroupDescription": "Security group allowing port 8086 ingress for InfluxDB",
        "VpcId": {"Ref": "VPC"}
      }
    },
    "InfluxDBSecurityGroupIngress": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "GroupId": {"Ref": "InfluxDBSecurityGroup"},
        "IpProtocol": "tcp",
        "CidrIp": "0.0.0.0/0",
        "FromPort": 8086,
        "ToPort": 8086
      }
    },
    "InfluxDBLogsS3Bucket": {
      "Type": "AWS::S3::Bucket",
      "DeletionPolicy": "Retain"
```

```
    },
    "InfluxDBLogsS3BucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {"Ref": "InfluxDBLogsS3Bucket"},
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "s3:PutObject",
              "Effect": "Allow",
              "Resource": {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/
*"},
              "Principal": {"Service": "timestream-influxdb.amazonaws.com"}
            },
            {
              "Action": "s3:*",
              "Effect": "Deny",
              "Resource": [
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/*"},
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}"}
              ],
              "Principal": "*",
              "Condition": {
                "Bool": {"aws:SecureTransport": false}
              }
            }
          ]
        }
      }
    },
    "DbInstance": {
      "Type": "AWS::Timestream::InfluxDBInstance",
      "DependsOn": "InfluxDBLogsS3BucketPolicy",
      "Properties": {
        "AllocatedStorage": 20,
        "DbInstanceType": "db.influx.medium",
        "Name": {"Ref": "DbInstanceName"},
        "Password": {"Ref": "InfluxDBPassword"},
        "PubliclyAccessible": true,
        "DeploymentType": "WITH_MULTIAZ_STANDBY",
        "VpcSecurityGroupIds": [
          {"Ref": "InfluxDBSecurityGroup"}
        ],
```

```
        "VpcSubnetIds": [
          {"Ref": "Subnet1"},
          {"Ref": "Subnet2"}
        ],
        "LogDeliveryConfiguration": {
          "S3Configuration": {
            "BucketName": {"Ref": "InfluxDBLogsS3Bucket"},
            "Enabled": true
          }
        }
      }
    }
  },
  "Outputs": {
    "VPC": {
      "Description": "A reference to the VPC used to create network resources",
      "Value": {"Ref": "VPC"}
    },
    "Subnets": {
      "Description": "A list of the subnets created",
      "Value": {
        "Fn::Join": [
          ",",
          [
            {"Ref": "Subnet1"},
            {"Ref": "Subnet2"}
          ]
        ]
      }
    },
    "Subnet1": {
      "Description": "A reference to the subnet in the 1st Availability Zone",
      "Value": {"Ref": "Subnet1"}
    },
    "Subnet2": {
      "Description": "A reference to the subnet in the 2nd Availability Zone",
      "Value": {"Ref": "Subnet2"}
    },
    "InfluxDBSecurityGroup": {
      "Description": "Security group with port 8086 ingress rule",
      "Value": {"Ref": "InfluxDBSecurityGroup"}
    },
    "InfluxDBLogsS3Bucket": {
      "Description": "S3 Bucket containing InfluxDB logs from the DB instance",
```

```
      "Value": {"Ref": "InfluxDBLogsS3Bucket"}
    },
    "DbInstance": {
      "Description": "A reference to the Timestream for InfluxDB DB instance",
      "Value": {"Ref": "DbInstance"}
    },
    "InfluxAuthParametersSecretArn": {
      "Description": "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret
 containing the initial InfluxDB authorization parameters. The secret value is a JSON
 formatted key-value pair holding InfluxDB authorization values: organization, bucket,
 username, and password.",
      "Value": {
        "Fn::GetAtt": [
          "DbInstance",
          "InfluxAuthParametersSecretArn"
        ]
      }
    },
    "Endpoint": {
      "Description": "The endpoint URL to connect to InfluxDB",
      "Value": {
        "Fn::Join": [
          "",
          [
            "https://",
            {
              "Fn::GetAtt": [
                "DbInstance",
                "Endpoint"
              ]
            },
            ":8086"
          ]
        ]
      }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
```

```
    ParameterGroups:
      -
        Label:
          default: "Amazon Timestream for InfluxDB Configuration"
        Parameters:
          - DbInstanceName
          - InfluxDBPassword
    ParameterLabels:
      VPCCIDR:
        default: VPC CIDR

Parameters:
  DbInstanceName:
    Description: The name that uniquely identifies the DB instance when interacting
 with the Amazon Timestream for InfluxDB API and CLI commands. This name will also be a
 prefix included in the endpoint. DB instance names must be unique per customer and per
 Region.
    Type: String
    Default: mydbinstance
    MinLength: 3
    MaxLength: 40
    AllowedPattern: ^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$
  InfluxDBPassword:
    Description: The password of the initial admin user created in InfluxDB. This
 password will allow you to access the InfluxDB UI to perform various administrative
 tasks and also use the InfluxDB CLI to create an operator token. These attributes will
 be stored in a Secret created in AWS Secrets Manager in your account.
    Type: String
    NoEcho: true
    MinLength: 8
    MaxLength: 64
    AllowedPattern: ^[a-zA-Z0-9]+$

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
```

```
      VpcId: !Ref VPC
  Subnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Select [0, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
      MapPublicIpOnLaunch: true
  Subnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs  '']
      CidrBlock: !Select [1, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
      MapPublicIpOnLaunch: true
  RouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
  DefaultRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref RouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  Subnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet1
  Subnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet2
  InfluxDBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "influxdb-sg"
      GroupDescription: "Security group allowing port 8086 ingress for InfluxDB"
      VpcId: !Ref VPC
  InfluxDBSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
```

```
    Properties:
      GroupId: !Ref InfluxDBSecurityGroup
      IpProtocol: tcp
      CidrIp: 0.0.0.0/0
      FromPort: 8086
      ToPort: 8086
  InfluxDBLogsS3Bucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
  InfluxDBLogsS3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref InfluxDBLogsS3Bucket
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Action: "s3:PutObject"
            Effect: Allow
            Resource: !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/*
            Principal:
              Service: timestream-influxdb.amazonaws.com
          - Action: "s3:*"
            Effect: Deny
            Resource:
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/*
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}
            Principal: "*"
            Condition:
              Bool:
                aws:SecureTransport: false
  DbInstance:
    Type: AWS::Timestream::InfluxDBInstance
    DependsOn: InfluxDBLogsS3BucketPolicy
    Properties:
      AllocatedStorage: 20
      DbInstanceType: db.influx.medium
      Name: !Ref DbInstanceName
      Password: !Ref InfluxDBPassword
      PubliclyAccessible: true
      DeploymentType: WITH_MULTIAZ_STANDBY
      VpcSecurityGroupIds:
        - !Ref InfluxDBSecurityGroup
      VpcSubnetIds:
        - !Ref Subnet1
```

```
        - !Ref Subnet2
      LogDeliveryConfiguration:
        S3Configuration:
          BucketName: !Ref InfluxDBLogsS3Bucket
          Enabled: true

Outputs:
  # Network Resources
  VPC:
    Description: A reference to the VPC used to create network resources
    Value: !Ref VPC
  Subnets:
    Description: A list of the subnets created
    Value: !Join [",", [!Ref Subnet1, !Ref Subnet2]]
  Subnet1:
    Description: A reference to the subnet in the 1st Availability Zone
    Value: !Ref Subnet1
  Subnet2:
    Description: A reference to the subnet in the 2nd Availability Zone
    Value: !Ref Subnet2
  InfluxDBSecurityGroup:
    Description: Security group with port 8086 ingress rule
    Value: !Ref InfluxDBSecurityGroup

  # Timestream for InfluxDB Resources
  InfluxDBLogsS3Bucket:
    Description: S3 Bucket containing InfluxDB logs from the DB instance
    Value: !Ref InfluxDBLogsS3Bucket
  DbInstance:
    Description: A reference to the Timestream for InfluxDB DB instance
    Value: !Ref DbInstance
  InfluxAuthParametersSecretArn:
    Description: "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret
 containing the initial InfluxDB authorization parameters. The secret value is a JSON
 formatted key-value pair holding InfluxDB authorization values: organization, bucket,
 username, and password."
    Value: !GetAtt DbInstance.InfluxAuthParametersSecretArn
  Endpoint:
    Description: The endpoint URL to connect to InfluxDB
    Value: !Join ["", ["https://", !GetAtt DbInstance.Endpoint, ":8086"]]
```

More complete example with parameters

This example template dynamically alters the network resources based on the parameters provided. Parameters include `PubliclyAccessible` and `DeploymentType`.

JSON

```
{
  "Metadata": {
    "AWS::CloudFormation::Interface": {
      "ParameterGroups": [
        {
          "Label": {"default": "Network Configuration"},
          "Parameters": ["VPCCIDR"]
        },
        {
          "Label": {"default": "Amazon Timestream for InfluxDB Configuration"},
          "Parameters": [
            "DbInstanceName",
            "InfluxDBUsername",
            "InfluxDBPassword",
            "InfluxDBOrganization",
            "InfluxDBBucket",
            "DbInstanceType",
            "DbStorageType",
            "AllocatedStorage",
            "PubliclyAccessible",
            "DeploymentType"
          ]
        }
      ],
      "ParameterLabels": {
        "VPCCIDR": {"default": "VPC CIDR"}
      }
    }
  },
  "Parameters": {
    "VPCCIDR": {
      "Description": "Please enter the IP range (CIDR notation) for the new VPC",
      "Type": "String",
      "Default": "10.0.0.0/16"
    },
    "DbInstanceName": {
```

```
      "Description": "The name that uniquely identifies the DB instance when
interacting with the Amazon Timestream for InfluxDB API and CLI commands. This name
will also be a prefix included in the endpoint. DB instance names must be unique per
customer and per Region.",
      "Type": "String",
      "Default": "mydbinstance",
      "MinLength": 3,
      "MaxLength": 40,
      "AllowedPattern": "^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$"
    },
    "InfluxDBUsername": {
      "Description": "The username of the initial admin user created in InfluxDB. Must
start with a letter and can't end with a hyphen or contain two consecutive hyphens.
For example, my-user1. This username will allow you to access the InfluxDB UI to
perform various administrative tasks and also use the InfluxDB CLI to create an
operator token. These attributes will be stored in a Secret created in AWS Secrets
Manager in your account.",
      "Type": "String",
      "Default": "admin",
      "MinLength": 1,
      "MaxLength": 64
    },
    "InfluxDBPassword": {
      "Description": "The password of the initial admin user created in InfluxDB. This
password will allow you to access the InfluxDB UI to perform various administrative
tasks and also use the InfluxDB CLI to create an operator token. These attributes will
be stored in a Secret created in AWS Secrets Manager in your account.",
      "Type": "String",
      "NoEcho": true,
      "MinLength": 8,
      "MaxLength": 64,
      "AllowedPattern": "^[a-zA-Z0-9]+$"
    },
    "InfluxDBOrganization": {
      "Description": "The name of the initial organization for the initial admin user
in InfluxDB. An InfluxDB organization is a workspace for a group of users.",
      "Type": "String",
      "Default": "org",
      "MinLength": 1,
      "MaxLength": 64
    },
    "InfluxDBBucket": {
      "Description": "The name of the initial InfluxDB bucket. All InfluxDB data
is stored in a bucket. A bucket combines the concept of a database and a retention
```

```
period (the duration of time that each data point persists). A bucket belongs to an
organization.",
     "Type": "String",
     "Default": "bucket",
     "MinLength": 2,
     "MaxLength": 64,
     "AllowedPattern": "^[^_\\\"][^\\\"]*$"
   },
   "DeploymentType": {
     "Description": "Specifies whether the Timestream for InfluxDB is deployed as
Single-AZ or with a MultiAZ Standby for High availability",
     "Type": "String",
     "Default": "WITH_MULTIAZ_STANDBY",
     "AllowedValues": [
       "SINGLE_AZ",
       "WITH_MULTIAZ_STANDBY"
     ]
   },
   "AllocatedStorage": {
     "Description": "The amount of storage to allocate for your DB storage type in GiB
(gibibytes).",
     "Type": "Number",
     "Default": 400,
     "MinValue": 20,
     "MaxValue": 16384
   },
   "DbInstanceType": {
     "Description": "The Timestream for InfluxDB DB instance type to run InfluxDB
on.",
     "Type": "String",
     "Default": "db.influx.medium",
     "AllowedValues": [
       "db.influx.medium",
       "db.influx.large",
       "db.influx.xlarge",
       "db.influx.2xlarge",
       "db.influx.4xlarge",
       "db.influx.8xlarge",
       "db.influx.12xlarge",
       "db.influx.16xlarge"
     ]
   },
   "DbStorageType": {
```

```
      "Description": "The Timestream for InfluxDB DB storage type to read and write
 InfluxDB data.",
      "Type": "String",
      "Default": "InfluxIOIncludedT1",
      "AllowedValues": [
        "InfluxIOIncludedT1",
        "InfluxIOIncludedT2",
        "InfluxIOIncludedT3"
      ]
    },
    "PubliclyAccessible": {
      "Description": "Configures the DB instance with a public IP to facilitate
 access.",
      "Type": "String",
      "Default": true,
      "AllowedValues": [
        true,
        false
      ]
    }
  },
  "Conditions": {
    "IsMultiAZ": {
      "Fn::Equals": [
        {"Ref": "DeploymentType"},
        "WITH_MULTIAZ_STANDBY"
      ]
    },
    "IsPublic": {
      "Fn::Equals": [
        {"Ref": "PubliclyAccessible"},
        true
      ]
    }
  },
  "Resources": {
    "VPC": {
      "Type": "AWS::EC2::VPC",
      "Properties": {
        "CidrBlock": {"Ref": "VPCCIDR"}
      }
    },
    "InternetGateway": {
      "Type": "AWS::EC2::InternetGateway",
```

```
      "Condition": "IsPublic"
    },
    "InternetGatewayAttachment": {
      "Type": "AWS::EC2::VPCGatewayAttachment",
      "Condition": "IsPublic",
      "Properties": {
        "InternetGatewayId": {"Ref": "InternetGateway"},
        "VpcId": {"Ref": "VPC"}
      }
    },
    "Subnet1": {
      "Type": "AWS::EC2::Subnet",
      "Properties": {
        "VpcId": {"Ref": "VPC"},
        "AvailabilityZone": {
          "Fn::Select": [
            0,
            {"Fn::GetAZs": ""}
          ]
        },
        "CidrBlock": {
          "Fn::Select": [
            0,
            {
              "Fn::Cidr": [
                {
                  "Fn::GetAtt": [
                    "VPC",
                    "CidrBlock"
                  ]
                },
                2,
                12
              ]
            }
          ]
        },
        "MapPublicIpOnLaunch": {
          "Fn::If": [
            "IsPublic",
            true,
            false
          ]
        }
```

```
      }
    },
    "Subnet2": {
      "Type": "AWS::EC2::Subnet",
      "Condition": "IsMultiAZ",
      "Properties": {
        "VpcId": {"Ref": "VPC"},
        "AvailabilityZone": {
          "Fn::Select": [
            1,
            {"Fn::GetAZs": ""}
          ]
        },
        "CidrBlock": {
          "Fn::Select": [
            1,
            {
              "Fn::Cidr": [
                {
                  "Fn::GetAtt": [
                    "VPC",
                    "CidrBlock"
                  ]
                },
                2,
                12
              ]
            }
          ]
        },
        "MapPublicIpOnLaunch": {
          "Fn::If": [
            "IsPublic",
            true,
            false
          ]
        }
      }
    },
    "RouteTable": {
      "Type": "AWS::EC2::RouteTable",
      "Properties": {
        "VpcId": {"Ref": "VPC"}
      }
```

```
    },
    "DefaultRoute": {
      "Type": "AWS::EC2::Route",
      "Condition": "IsPublic",
      "DependsOn": "InternetGatewayAttachment",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": {"Ref": "InternetGateway"}
      }
    },
    "Subnet1RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet1"}
      }
    },
    "Subnet2RouteTableAssociation": {
      "Type": "AWS::EC2::SubnetRouteTableAssociation",
      "Condition": "IsMultiAZ",
      "Properties": {
        "RouteTableId": {"Ref": "RouteTable"},
        "SubnetId": {"Ref": "Subnet2"}
      }
    },
    "InfluxDBSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupName": "influxdb-sg",
        "GroupDescription": "Security group allowing port 8086 ingress for InfluxDB",
        "VpcId": {"Ref": "VPC"}
      }
    },
    "InfluxDBSecurityGroupIngress": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "GroupId": {"Ref": "InfluxDBSecurityGroup"},
        "IpProtocol": "tcp",
        "CidrIp": "0.0.0.0/0",
        "FromPort": 8086,
        "ToPort": 8086
      }
    },
```

```
    "InfluxDBLogsS3Bucket": {
      "Type": "AWS::S3::Bucket",
      "DeletionPolicy": "Retain"
    },
    "InfluxDBLogsS3BucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {"Ref": "InfluxDBLogsS3Bucket"},
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": "s3:PutObject",
              "Effect": "Allow",
              "Resource": {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/
*"},
              "Principal": {"Service": "timestream-influxdb.amazonaws.com"}
            },
            {
              "Action": "s3:*",
              "Effect": "Deny",
              "Resource": [
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}/*"},
                {"Fn::Sub": "arn:aws:s3:::${InfluxDBLogsS3Bucket}"}
              ],
              "Principal": "*",
              "Condition": {
                "Bool": {"aws:SecureTransport": false}
              }
            }
          ]
        }
      }
    },
    "DbInstance": {
      "Type": "AWS::Timestream::InfluxDBInstance",
      "DependsOn": "InfluxDBLogsS3BucketPolicy",
      "Properties": {
        "DbStorageType": {"Ref": "DbStorageType"},
        "AllocatedStorage": {"Ref": "AllocatedStorage"},
        "DbInstanceType": {"Ref": "DbInstanceType"},
        "Name": {"Ref": "DbInstanceName"},
        "Username": {"Ref": "InfluxDBUsername"},
        "Password": {"Ref": "InfluxDBPassword"},
```

```
        "Organization": {"Ref": "InfluxDBOrganization"},
        "Bucket": {"Ref": "InfluxDBBucket"},
        "PubliclyAccessible": {
          "Fn::If": [
            "IsPublic",
            true,
            false
          ]
        },
        "DeploymentType": {"Ref": "DeploymentType"},
        "VpcSecurityGroupIds": [
          {"Ref": "InfluxDBSecurityGroup"}
        ],
        "VpcSubnetIds": {
          "Fn::If": [
            "IsMultiAZ",
            [
              {"Ref": "Subnet1"},
              {"Ref": "Subnet2"}
            ],
            [
              {"Ref": "Subnet1"}
            ]
          ]
        },
        "LogDeliveryConfiguration": {
          "S3Configuration": {
            "BucketName": {"Ref": "InfluxDBLogsS3Bucket"},
            "Enabled": true
          }
        }
      }
    }
  },
  "Outputs": {
    "VPC": {
      "Description": "A reference to the VPC used to create network resources",
      "Value": {"Ref": "VPC"}
    },
    "Subnets": {
      "Description": "A list of the subnets created",
      "Value": {
        "Fn::If": [
          "IsMultiAZ",
```

```
          {
            "Fn::Join": [
              ",",
              [
                {"Ref": "Subnet1"},
                {"Ref": "Subnet2"}
              ]
            ]
          },
          {"Ref": "Subnet1"}
        ]
      }
    },
    "Subnet1": {
      "Description": "A reference to the subnet in the 1st Availability Zone",
      "Value": {"Ref": "Subnet1"}
    },
    "Subnet2": {
      "Condition": "IsMultiAZ",
      "Description": "A reference to the subnet in the 2nd Availability Zone",
      "Value": {"Ref": "Subnet2"}
    },
    "InfluxDBSecurityGroup": {
      "Description": "Security group with port 8086 ingress rule",
      "Value": {"Ref": "InfluxDBSecurityGroup"}
    },
    "InfluxDBLogsS3Bucket": {
      "Description": "S3 Bucket containing InfluxDB logs from the DB instance",
      "Value": {"Ref": "InfluxDBLogsS3Bucket"}
    },
    "DbInstance": {
      "Description": "A reference to the Timestream for InfluxDB DB instance",
      "Value": {"Ref": "DbInstance"}
    },
    "InfluxAuthParametersSecretArn": {
      "Description": "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret
containing the initial InfluxDB authorization parameters. The secret value is a JSON
formatted key-value pair holding InfluxDB authorization values: organization, bucket,
username, and password.",
      "Value": {
        "Fn::GetAtt": [
          "DbInstance",
          "InfluxAuthParametersSecretArn"
        ]
```

```
      }
    },
    "Endpoint": {
      "Description": "The endpoint URL to connect to InfluxDB",
      "Value": {
        "Fn::Join": [
          "",
          [
            "https://",
            {
              "Fn::GetAtt": [
                "DbInstance",
                "Endpoint"
              ]
            },
            ":8086"
          ]
        ]
      }
    }
  }
}
```

YAML

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Network Configuration"
        Parameters:
          - VPCCIDR
      -
        Label:
          default: "Amazon Timestream for InfluxDB Configuration"
        Parameters:
          - DbInstanceName
          - InfluxDBUsername
          - InfluxDBPassword
          - InfluxDBOrganization
          - InfluxDBBucket
          - DbInstanceType
```

```
          - DbStorageType
          - AllocatedStorage
          - PubliclyAccessible
          - DeploymentType
    ParameterLabels:
      VPCCIDR:
        default: VPC CIDR

Parameters:
  # Network Configuration
  VPCCIDR:
    Description: Please enter the IP range (CIDR notation) for the new VPC
    Type: String
    Default: 10.0.0.0/16
  # Timestream for InfluxDB Configuration
  DbInstanceName:
    Description: The name that uniquely identifies the DB instance when interacting
 with the Amazon Timestream for InfluxDB API and CLI commands. This name will also be a
 prefix included in the endpoint. DB instance names must be unique per customer and per
 Region.
    Type: String
    Default: mydbinstance
    MinLength: 3
    MaxLength: 40
    AllowedPattern: ^[a-zA-z][a-zA-Z0-9]*(-[a-zA-Z0-9]+)*$
  # InfluxDB initial user configurations
  InfluxDBUsername:
    Description: The username of the initial admin user created in InfluxDB. Must start
 with a letter and can't end with a hyphen or contain two consecutive hyphens. For
 example, my-user1. This username will allow you to access the InfluxDB UI to perform
 various administrative tasks and also use the InfluxDB CLI to create an operator
 token. These attributes will be stored in a Secret created in AWS Secrets Manager in
 your account.
    Type: String
    Default: admin
    MinLength: 1
    MaxLength: 64
  InfluxDBPassword:
    Description: The password of the initial admin user created in InfluxDB. This
 password will allow you to access the InfluxDB UI to perform various administrative
 tasks and also use the InfluxDB CLI to create an operator token. These attributes will
 be stored in a Secret created in AWS in your account.
    Type: String
    NoEcho: true
```

```
    MinLength: 8
    MaxLength: 64
    AllowedPattern: ^[a-zA-Z0-9]+$
  InfluxDBOrganization:
    Description: The name of the initial organization for the initial admin user in
InfluxDB. An InfluxDB organization is a workspace for a group of users.
    Type: String
    Default: org
    MinLength: 1
    MaxLength: 64
  InfluxDBBucket:
    Description: The name of the initial InfluxDB bucket. All InfluxDB data is stored
in a bucket. A bucket combines the concept of a database and a retention period (the
duration of time that each data point persists). A bucket belongs to an organization.
    Type: String
    Default: bucket
    MinLength: 2
    MaxLength: 64
    AllowedPattern: ^[^_\"][^\"]*$
  DeploymentType:
    Description: Specifies whether the Timestream for InfluxDB is deployed as Single-AZ
or with a MultiAZ Standby for High availability
    Type: String
    Default: WITH_MULTIAZ_STANDBY
    AllowedValues:
      - SINGLE_AZ
      - WITH_MULTIAZ_STANDBY
  AllocatedStorage:
    Description: The amount of storage to allocate for your DB storage type in GiB
(gibibytes).
    Type: Number
    Default: 400
    MinValue: 20
    MaxValue: 16384
  DbInstanceType:
    Description: The Timestream for InfluxDB DB instance type to run InfluxDB on.
    Type: String
    Default: db.influx.medium
    AllowedValues:
      - db.influx.medium
      - db.influx.large
      - db.influx.xlarge
      - db.influx.2xlarge
      - db.influx.4xlarge
```

```
      - db.influx.8xlarge
      - db.influx.12xlarge
      - db.influx.16xlarge
  DbStorageType:
    Description: The Timestream for InfluxDB DB storage type to read and write InfluxDB
 data.
    Type: String
    Default: InfluxIOIncludedT1
    AllowedValues:
      - InfluxIOIncludedT1
      - InfluxIOIncludedT2
      - InfluxIOIncludedT3
  PubliclyAccessible:
    Description: Configures the DB instance with a public IP to facilitate access.
    Type: String
    Default: true
    AllowedValues:
      - true
      - false

Conditions:
  IsMultiAZ: !Equals [!Ref DeploymentType, WITH_MULTIAZ_STANDBY]
  IsPublic: !Equals [!Ref PubliclyAccessible, true]

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VPCCIDR
  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Condition: IsPublic
  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Condition: IsPublic
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  Subnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [0, !GetAZs '']
      CidrBlock: !Select [0, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
```

```
      MapPublicIpOnLaunch: !If [IsPublic, true, false]
  Subnet2:
    Type: AWS::EC2::Subnet
    Condition: IsMultiAZ
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [1, !GetAZs  '']
      CidrBlock: !Select [1, !Cidr [!GetAtt VPC.CidrBlock, 2, 12 ]]
      MapPublicIpOnLaunch: !If [IsPublic, true, false]
  RouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
  DefaultRoute:
    Type: AWS::EC2::Route
    Condition: IsPublic
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref RouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  Subnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet1
  Subnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Condition: IsMultiAZ
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref Subnet2
  InfluxDBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "influxdb-sg"
      GroupDescription: "Security group allowing port 8086 ingress for InfluxDB"
      VpcId: !Ref VPC
  InfluxDBSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref InfluxDBSecurityGroup
      IpProtocol: tcp
      CidrIp: 0.0.0.0/0
```

```
      FromPort: 8086
      ToPort: 8086
  InfluxDBLogsS3Bucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
  InfluxDBLogsS3BucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref InfluxDBLogsS3Bucket
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Action: "s3:PutObject"
            Effect: Allow
            Resource: !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/InfluxLogs/*
            Principal:
              Service: timestream-influxdb.amazonaws.com
          - Action: "s3:*"
            Effect: Deny
            Resource:
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}/*
              - !Sub arn:aws:s3:::${InfluxDBLogsS3Bucket}
            Principal: "*"
            Condition:
              Bool:
                aws:SecureTransport: false
  DbInstance:
    Type: AWS::Timestream::InfluxDBInstance
    DependsOn: InfluxDBLogsS3BucketPolicy
    Properties:
      DbStorageType: !Ref DbStorageType
      AllocatedStorage: !Ref AllocatedStorage
      DbInstanceType: !Ref DbInstanceType
      Name: !Ref DbInstanceName
      Username: !Ref InfluxDBUsername
      Password: !Ref InfluxDBPassword
      Organization: !Ref InfluxDBOrganization
      Bucket: !Ref InfluxDBBucket
      PubliclyAccessible: !If [IsPublic, true, false]
      DeploymentType: !Ref DeploymentType
      VpcSecurityGroupIds:
        - !Ref InfluxDBSecurityGroup
      VpcSubnetIds: !If
        - IsMultiAZ
```

```
        -
          - !Ref Subnet1
          - !Ref Subnet2
        -
          - !Ref Subnet1
      LogDeliveryConfiguration:
        S3Configuration:
          BucketName: !Ref InfluxDBLogsS3Bucket
          Enabled: true

Outputs:
  # Network Resources
  VPC:
    Description: A reference to the VPC used to create network resources
    Value: !Ref VPC
  Subnets:
    Description: A list of the subnets created
    Value: !If
      - IsMultiAZ
      - !Join [",", [!Ref Subnet1, !Ref Subnet2]]
      - !Ref Subnet1
  Subnet1:
    Description: A reference to the subnet in the 1st Availability Zone
    Value: !Ref Subnet1
  Subnet2:
    Condition: IsMultiAZ
    Description: A reference to the subnet in the 2nd Availability Zone
    Value: !Ref Subnet2
  InfluxDBSecurityGroup:
    Description: Security group with port 8086 ingress rule
    Value: !Ref InfluxDBSecurityGroup

  # Timestream for InfluxDB Resources
  InfluxDBLogsS3Bucket:
    Description: S3 Bucket containing InfluxDB logs from the DB instance
    Value: !Ref InfluxDBLogsS3Bucket
  DbInstance:
    Description: A reference to the Timestream for InfluxDB DB instance
    Value: !Ref DbInstance
  InfluxAuthParametersSecretArn:
    Description: "The Amazon Resource Name (ARN) of the AWS Secrets Manager secret
 containing the initial InfluxDB authorization parameters. The secret value is a JSON
 formatted key-value pair holding InfluxDB authorization values: organization, bucket,
 username, and password."
```

```
    Value: !GetAtt DbInstance.InfluxAuthParametersSecretArn
  Endpoint:
    Description: The endpoint URL to connect to InfluxDB
    Value: !Join ["", ["https://", !GetAtt DbInstance.Endpoint, ":8086"]]
```

Deploy Windows-based stacks using CloudFormation

This page provides links to technical reference documentation for CloudFormation resources commonly used in Windows-based deployments.

CloudFormation provides support for deploying and managing Microsoft Windows stacks through Infrastructure as Code (IaC). You can use CloudFormation for automated provisioning of Windows-based EC2 instances, SQL Server on Amazon RDS, and Microsoft Active Directory through Directory Service.

AWS provides pre-configured Amazon Machine Images (AMIs) specifically designed for Windows platforms to help you quickly deploy applications on Amazon EC2. These AMIs include default Microsoft settings and AWS-specific customizations. With CloudFormation, you can choose an appropriate AMI, launch an instance, and access it using Remote Desktop Connection, just as you would with any other Windows Server. The AMIs contain essential software components, including EC2Launch (versions vary by Windows Server edition), AWS Systems Manager, CloudFormation, AWS Tools for PowerShell, and various network, storage, and graphics drivers to ensure optimal performance and compatibility with AWS services. For more information, see the AWS Windows AMI Reference.

CloudFormation also supports software configuration tools, such as `UserData` scripts, which can run PowerShell or batch commands when an EC2 instance first boots up. It also offers helper scripts (`cfn-init`, `cfn-signal`, `cfn-get-metadata`, and `cfn-hup`) and supports the `AWS::CloudFormation::Init` metadata for managing packages, files, and services on Windows instances.

For enterprise environments, CloudFormation enables domain joining, Windows license management through EC2 licensing models, and secure credential handling with AWS Secrets Manager. Combined with version-controlled templates and repeatable deployments, CloudFormation helps organizations maintain consistent, secure, and scalable Windows environments across multiple AWS Regions and accounts.

For details on CloudFormation resources commonly used in Windows-based deployments, see the following technical reference topics.

Resource type	Description
AWS::EC2::Instance	For launching Windows EC2 instances.
AWS::EC2::SecurityGroup	To define firewall rules for Windows workloads.
AWS::AutoScaling::AutoScalingGroup AWS::EC2::LaunchTemplate	For scaling Windows EC2 instances.
AWS::DirectoryService::MicrosoftAD	For deploying Microsoft Active Directory.
AWS::FSx::FileSystem	For deploying FSx for Windows File Server.
AWS::RDS::DBInstance	For provisioning SQL Server on Amazon RDS.
AWS::CloudFormation::Init	Used within EC2 metadata for configuring instances. For more information, see Bootstrapping Windows-based CloudFormation stacks.
AWS::SecretsManager::Secret	For securely managing credentials and Windows passwords.
AWS::SSM::Parameter	For storing configuration values securely.
AWS::IAM::InstanceProfile AWS::IAM::Role	For granting permissions to applications running on EC2 instances.

Bootstrapping Windows-based CloudFormation stacks

This topic describes how to bootstrap a Windows stack and troubleshoot stack creation issues.

Topics

- User data in EC2 instances
- CloudFormation helper scripts
- Example of bootstrapping a Windows stack

- Escape backslashes in Windows file paths
- Manage Windows services
- Troubleshoot stack creation issues

User data in EC2 instances

User data is an Amazon EC2 feature that allows you to pass scripts or configuration information to an EC2 instance when it launches.

For Windows EC2 instances:

- You can use either batch scripts (using `<script>` tags) or PowerShell scripts (using `<powershell>` tags).
- Script execution is handled by EC2Launch.

> **Important**
>
> If you are creating your own Windows AMI for use with CloudFormation, make sure that EC2Launch v2 is properly configured. EC2Launch v2 is required for CloudFormation bootstrapping tools to properly initialize and configure Windows instances during stack creation. For more information, see Use the EC2Launch v2 agent to perform tasks during EC2 Windows instance launch in the *Amazon EC2 User Guide*.
> For information about AWS Windows AMIs, see the AWS Windows AMI Reference.

CloudFormation helper scripts

Helper scripts are utilities for configuring instances during the bootstrapping process. Used with Amazon EC2 user data, they provide powerful configuration options.

CloudFormation provides the following Python helper scripts that you can use to install software and start services on an Amazon EC2 instance that you create as part of your stack:

- `cfn-init` – Use to retrieve and interpret resource metadata, install packages, create files, and start services.
- `cfn-signal` – Use to signal with a `CreationPolicy`, so you can synchronize other resources in the stack when the prerequisite resource or application is ready.

- `cfn-get-metadata` – Use to retrieve metadata for a resource or path to a specific key.
- `cfn-hup` – Use to check for updates to metadata and execute custom hooks when changes are detected.

You call the scripts directly from your template. The scripts work in conjunction with resource metadata that's defined in the same template. The scripts run on the Amazon EC2 instance during the stack creation process.

For more information, see the CloudFormation helper scripts reference in the *CloudFormation Template Reference Guide*.

Example of bootstrapping a Windows stack

Let's examine example snippets from a Windows Server template that performs the following actions:

- Launches an EC2 instance named `TestInstance` from a Windows Server 2022 AMI.
- Creates a simple test file to verify `cfn-init` is working.
- Configures `cfn-hup` for ongoing configuration management.
- Uses a `CreationPolicy` to ensure the instance signals successful completion.

The `cfn-init` helper script is used to perform each of these actions based on information in the `AWS::CloudFormation::Init` resource in the template.

The `AWS::CloudFormation::Init` section is named `TestInstance` and begins with the following declaration.

```
TestInstance:
  Type: AWS::EC2::Instance
  Metadata:
    AWS::CloudFormation::Init:
      configSets:
        default:
          - create_files
          - start_services
```

After this, the `files` section of `AWS::CloudFormation::Init` is declared.

```
      create_files:
        files:
          c:\cfn\test.txt:
            content: !Sub |
              Hello from ${AWS::StackName}
          c:\cfn\cfn-hup.conf:
            content: !Sub |
              [main]
              stack=${AWS::StackName}
              region=${AWS::Region}
              interval=2
          c:\cfn\hooks.d\cfn-auto-reloader.conf:
            content: !Sub |
              [cfn-auto-reloader-hook]
              triggers=post.update
              path=Resources.TestInstance.Metadata.AWS::CloudFormation::Init
              action=cfn-init.exe -v -s ${AWS::StackName} -r TestInstance -c default --
region ${AWS::Region}
```

Three files are created here and placed in the `C:\cfn` directory on the server instance:

- `test.txt`, a simple test file that verifies `cfn-init` is working correctly and can create files with dynamic content.
- `cfn-hup.conf`, the configuration file for `cfn-hup` with a 2-minute check interval.
- `cfn-auto-reloader.conf`, the configuration file for the hook used by `cfn-hup` to initiate an update (calling `cfn-init`) when the metadata in `AWS::CloudFormation::Init` changes.

Next is the `start_services` section, which configures Windows services.

```
      start_services:
        services:
          windows:
            cfn-hup:
              enabled: true
              ensureRunning: true
              files:
                - c:\cfn\cfn-hup.conf
                - c:\cfn\hooks.d\cfn-auto-reloader.conf
```

This section ensures that the `cfn-hup` service is started and will automatically restart if the configuration files are modified. The service monitors for changes to the CloudFormation metadata and re-runs `cfn-init` when updates are detected.

Next is the `Properties` section.

```
TestInstance:
  Type: AWS::EC2::Instance
  CreationPolicy:
    ResourceSignal:
      Timeout: PT20M
  Metadata:
    AWS::CloudFormation::Init:
      # ... metadata configuration ...
  Properties:
    InstanceType: t2.large
    ImageId: '{{resolve:ssm:/aws/service/ami-windows-latest/Windows_Server-2022-
English-Full-Base}}'
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
    KeyName: !Ref KeyPairName
    UserData:
      Fn::Base64: !Sub |
        <powershell>
        cfn-init.exe -v -s ${AWS::StackName} -r TestInstance -c default --region
 ${AWS::Region}
        cfn-signal.exe -e $lastexitcode --stack ${AWS::StackName} --resource
 TestInstance --region ${AWS::Region}
        </powershell>
```

In this section, the `UserData` property contains a PowerShell script that will be executed by EC2Launch, surrounded by `<powershell>` tags. The script runs `cfn-init` with the `default` configSet, then uses `cfn-signal` to report the exit code back to CloudFormation. The `CreationPolicy` is used to ensure the instance is properly configured before the stack creation is considered complete.

The `ImageId` property uses a Systems Manager Parameter Store public parameter to automatically retrieve the latest Windows Server 2022 AMI ID. This approach eliminates the need for region-specific AMI mappings and ensures you always get the most recent AMI. Using Systems Manager parameters for AMI IDs is a best practice for maintaining current AMI references. If you plan to

connect to your instance, make sure that the `SecurityGroupIds` property references a security group that allows RDP access.

The `CreationPolicy` is declared as part of the resource properties and specifies a timeout period. The `cfn-signal` command in the user data signals when the instance configuration is complete:

```
TestInstance:
  Type: AWS::EC2::Instance
  CreationPolicy:
    ResourceSignal:
      Timeout: PT20M
  Properties:
    # ... other properties ...
```

Because the bootstrapping process is minimal and only creates files and starts services, the `CreationPolicy` waits 20 minutes (PT20M) before timing out. The timeout is specified using ISO 8601 duration format. Note that Windows instances generally take longer to launch than Linux instances, so test thoroughly to determine the best timeout values for your needs.

If all goes well, the `CreationPolicy` completes successfully and you can access the Windows Server instance using its public IP address. Once stack creation is complete, the instance ID and public IP address will be displayed in the **Outputs** tab of the CloudFormation console.

```
Outputs:
  InstanceId:
    Value: !Ref TestInstance
    Description: Instance ID of the Windows Server
  PublicIP:
    Value: !GetAtt TestInstance.PublicIp
    Description: Public IP address of the Windows Server
```

You can also manually verify that the bootstrapping worked correctly by connecting to the instance via RDP and checking that the file `C:\cfn\test.txt` exists and contains the expected content. For more information about connecting to Windows instances, see Connect to your Windows instance using RDP in the *Amazon EC2 User Guide*.

Escape backslashes in Windows file paths

When referencing Windows paths in CloudFormation templates, always remember to properly escape backslashes (\) according to the template format you're using.

- For JSON templates, you must use double backslashes in Windows file paths because JSON treats backslash as an escape character. The first backslash escapes the second, resulting in the interpretation of a single, literal backslash.

```
"commands" : {
  "1-extract" : {
    "command" : "C:\\SharePoint\\SharePointFoundation2010.exe /extract:C:\\SharePoint
\\SPF2010 /quiet /log:C:\\SharePoint\\SharePointFoundation2010-extract.log"
  }
}
```

- For YAML templates, single backslashes are typically sufficient.

```
commands:
  1-extract:
    command: C:\SharePoint\SharePointFoundation2010.exe /extract:C:\SharePoint
\SPF2010 /quiet /log:C:\SharePoint\SharePointFoundation2010-extract.log
```

Manage Windows services

You manage Windows services in the same way as Linux services, except that you use a `windows` key instead of `sysvinit`. The following example starts the `cfn-hup` service, sets it to Automatic, and restarts the service if `cfn-init` modifies the `c:\cfn\cfn-hup.conf` or `c:\cfn\hooks.d\cfn-auto-reloader.conf` configuration files.

```
      services:
        windows:
          cfn-hup:
            enabled: true
            ensureRunning: true
            files:
              - c:\cfn\cfn-hup.conf
              - c:\cfn\hooks.d\cfn-auto-reloader.conf
```

You can manage other Windows services in the same way by using the name, not the display name, to reference the service.

Troubleshoot stack creation issues

If your stack fails during creation, the default behavior is to rollback on failure. While this is normally a good default because it avoids unnecessary charges, it makes it difficult to debug why your stack creation is failing.

To turn this behavior off when creating or updating your stack with the CloudFormation console, choose the **Preserve successfully provisioned resources** option under **Stack failure options**. For more information, see Choose how to handle failures when provisioning resources. This allows you to log into your instance and view the log files to pinpoint issues encountered when running your startup scripts.

Important logs to look at are:

- The EC2 configuration log at `%ProgramData%\Amazon\EC2Launch\log\agent.log`
- The **cfn-init** log at `C:\cfn\log\cfn-init.log` (check exit codes and error messages for specific failure points)

For more logs, see the following topics in the *Amazon EC2 User Guide*:

- EC2Launch directory structure
- EC2Launch v2 directory structure

For more information about troubleshooting bootstrapping issues, see How do I troubleshoot helper scripts that won't bootstrap in a CloudFormation stack with Windows instances?.

Extend your template's capabilities with CloudFormation-supplied resource types

CloudFormation offers several resource types that you can use in your stack template to extend its capabilities beyond those of a simple stack template.

These resource types include:

Resource type	Description	Documentation
Custom resources	The `AWS::CloudFormation::CustomResource` resource type allows you to create	Custom resources

Resource type	Description	Documentation
	custom resources that can perform specific provisioning tasks or include resources that aren't available as CloudFormation resource types.	
Macros	The `AWS::CloudFormation::Macro` resource type defines a reusable piece of code that can perform custom processing on CloudFormation templates. Macros can modify your templates, generate additional resources , or perform other custom operations during stack creation or updates.	Template macros
Nested stacks	The `AWS::CloudFormation::Stack` resource type allows you to create nested stacks within your CloudFormation templates for a more modular and reusable stack architectures.	Nested stacks
StackSet	The `AWS::CloudFormation::StackSet` resource type creates or updates a CloudFormation StackSet, which is a container for stacks that can be deployed across multiple AWS accounts and Regions.	Managing stacks with StackSets
Wait condition	The `AWS::CloudFormation::WaitCondition` resource type pauses stack creation or update until a specific condition is met, such as the successful completion of a long-running process or the availability of external resources.	Wait conditions

Resource type	Description	Documentation
Wait condition handle	The `AWS::CloudFormation::WaitConditionHandle` resource type works together with the `AWS::CloudFormation::WaitCondition` resource type. It provides a presigned URL that's used to send signals indicating that a specific condition has been met. These signals allow the stack creation or update process to proceed.	Wait conditions

Create custom provisioning logic with custom resources

Custom resources provide a way for you to write custom provisioning logic into your CloudFormation templates and have CloudFormation run it anytime you create, update (if you changed the custom resource), or delete a stack. This can be useful when your provisioning requirements involve complex logic or workflows that can't be expressed with CloudFormation's built-in resource types.

For example, you might want to include resources that aren't available as CloudFormation resource types. You can include those resources by using custom resources. That way, you can still manage all your related resources in a single stack.

To define a custom resource in your CloudFormation template, you use the `AWS::CloudFormation::CustomResource` or `Custom::`*`MyCustomResourceTypeName`* resource type. Custom resources require one property, the service token, which specifies where CloudFormation sends requests to, such as an Amazon SNS topic or a Lambda function.

The following topics provide information on how to use custom resources.

Topics

- How custom resources work
- Response timeout
- CloudFormation custom resource request and response reference
- Amazon SNS-backed custom resources
- Lambda-backed custom resources

Note

The CloudFormation registry and custom resources each offer their own benefits. Custom resources offer the following benefits:

- You don't need to register the resource.
- You can include an entire resource as part of a template without registering.
- Supports the `Create`, `Update`, and `Delete` operations

Advantages that registry based resources offer include the following:

- Supports the modeling, provisioning, and managing of third-party application resources
- Supports the `Create`, `Read`, `Update`, `Delete`, and `List` (CRUDL) operations
- Supports drift detection on private and third-party resource types

Unlike custom resources, registry based resources won't need to associate an Amazon SNS topic or a Lambda function to perform CRUDL operations. For more information, see Managing extensions with the CloudFormation registry.

How custom resources work

The general process for setting up a new custom resource includes the following steps. These steps involve two roles: the *custom resource provider* who owns the custom resource and the *template developer* who creates a template that includes a custom resource type. This can be the same person, but if not, the custom resource provider should work with the template developer.

1. The custom resource provider writes logic that determines how to handle requests from CloudFormation and perform actions on the custom resource.
2. The custom resource provider creates the Amazon SNS topic or Lambda function where CloudFormation can send requests to. The Amazon SNS topic or Lambda function must be in the same Region where the stack will be created.
3. The custom resource provider gives the Amazon SNS topic ARN or Lambda function ARN to the template developer.
4. The template developer defines the custom resource in their CloudFormation template. This includes a service token and any input data parameters. The service token and the structure

of the input data are defined by the custom resource provider. The service token specifies the Amazon SNS topic ARN or Lambda function ARN and is always required, but the input data is optional depending on the custom resource.

Now, whenever anyone uses the template to create, update, or delete the custom resource, CloudFormation sends a request to the specified service token, and then waits for a response before proceeding with the stack operation.

The following summarizes the flow for creating a stack from the template:

1. CloudFormation sends a request to the specified service token. The request includes information such as the request type and a pre-signed Amazon S3 bucket URL, where the custom resource sends responses to. For more information about what's included in the request, see CloudFormation custom resource request and response reference.

   The following sample data shows what CloudFormation includes in a `Create` request. In this example, `ResourceProperties` allows CloudFormation to create a custom payload to send to the Lambda function.

   ```
   {
      "RequestType" : "Create",
      "RequestId" : "unique id for this create request",
      "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
   cd98-11ea-90d5-0a9cd3354c10",
      "ResponseURL" : "http://pre-signed-S3-url-for-response",
      "ResourceType" : "Custom::TestResource",
      "LogicalResourceId" : "MyTestResource",
      "ResourceProperties" : {
         "Name" : "Value",
         "List" : [ "1", "2", "3" ]
      }
   }
   ```

2. The custom resource provider processes the CloudFormation request and returns a response of `SUCCESS` or `FAILED` to the pre-signed URL. The custom resource provider provides the response in a JSON-formatted file and uploads it to the pre-signed S3 URL. For more information, see Uploading objects with presigned URLs in the *Amazon Simple Storage Service User Guide*.

   In the response, the custom resource provider can also include name-value pairs that the template developer can access. For example, the response can include output data if the request

succeeded or an error message if the request failed. For more information about responses, see CloudFormation custom resource request and response reference.

> **Important**
>
> If the name-value pairs contain sensitive information, you should use the `NoEcho` field to mask the output of the custom resource. Otherwise, the values are visible through APIs that surface property values (such as `DescribeStackEvents`).
> For more information about using `NoEcho` to mask sensitive information, see the Do not embed credentials in your templates best practice.

The custom resource provider is responsible for listening and responding to the request. For example, for Amazon SNS notifications, the custom resource provider must listen and respond to notifications that are sent to a specific topic ARN. CloudFormation waits and listens for a response in the pre-signed URL location.

The following sample data shows what a custom resource might include in a response:

```
{
   "Status" : "SUCCESS",
   "RequestId" : "unique id for this create request",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MyTestResource",
   "PhysicalResourceId" : "TestResource1",
   "Data" : {
      "OutputName1" : "Value1",
      "OutputName2" : "Value2",
   }
}
```

3. After getting a SUCCESS response, CloudFormation proceeds with the stack operation. If a `FAILED` response or no response is returned, the operation fails. Any output data from the custom resource is stored in the pre-signed URL location. The template developer can retrieve that data by using the Fn::GetAtt function.

> **Note**
>
> If you use AWS PrivateLink, custom resources in the VPC must have access to CloudFormation-specific S3 buckets. Custom resources must send responses to a pre-signed Amazon S3 URL. If they can't send responses to Amazon S3, CloudFormation won't receive a response and the stack operation fails. For more information, see Access CloudFormation using an interface endpoint (AWS PrivateLink).

Response timeout

The default timeout for your custom resource is 3600 seconds (1 hour). If no response is received during this time, the stack operation fails.

You can adjust the timeout value based on how long you expect the response from the custom resource will take. For example, when provisioning a custom resource that invokes a Lambda function that's expected to respond within five minutes, you can set a timeout of five minutes in the stack template by specifying the `ServiceTimeout` property. For more information, see CloudFormation custom resource request and response reference. This way, if there's an error in the Lambda function that causes it to get stuck, CloudFormation will fail the stack operation after five minutes instead of waiting the full hour.

However, be careful not to set the timeout value too low. To avoid unexpected timeouts, make sure that your custom resource has enough time to perform the necessary actions and return a response.

CloudFormation custom resource request and response reference

CloudFormation manages custom resources through a request-response protocol that communicates with your custom resource provider. Each request includes a request type (`Create`, `Update`, or `Delete`) and follows this high-level workflow:

1. A template developer defines a custom resource with a `ServiceToken` and `ServiceTimeout` in the template and initiates a stack operation.
2. CloudFormation sends a JSON request to the custom resource provider through SNS or Lambda.
3. The custom resource provider processes the request and returns a JSON response to a presigned Amazon S3 bucket URL before the timeout period expires.

4. CloudFormation reads the response and proceeds with the stack operation. If no response is received before the timeout period ends, the request is considered unsuccessful, and the stack operation fails.

For more information, see How custom resources work.

This section describes the structure, parameters, and expected responses for each request type.

> **Note**
>
> The total size of the response body can't exceed 4096 bytes.

Template setup

When defining a custom resource in a template, the template developer uses AWS::CloudFormation::CustomResource with the following properties:

`ServiceToken`

An Amazon SNS topic ARN or Lambda function ARN from the same Region as the stack.

Required: Yes

Type: String

`ServiceTimeout`

The maximum time, in seconds, before a custom resource operation times out. It must be a value between 1 and 3600. Default: 3600 seconds (1 hour).

Required: No

Type: String

Additional resource properties are supported. Resource properties will be included as `ResourceProperties` in the request. The custom resource provider must determine which properties are valid and their acceptable values.

Request object

Create

When the template developer creates a stack containing a custom resource, CloudFormation sends a request with `RequestType` set to `Create`.

Create requests contain the following fields:

`RequestType`

`Create`.

Required: Yes

Type: String

`RequestId`

A unique ID for the request.

Combining the `StackId` with the `RequestId` forms a value that you can use to uniquely identify a request on a particular custom resource.

Required: Yes

Type: String

`StackId`

The Amazon Resource Name (ARN) that identifies the stack that contains the custom resource.

Combining the `StackId` with the `RequestId` forms a value that you can use to uniquely identify a request on a particular custom resource.

Required: Yes

Type: String

`ResponseURL`

The response URL identifies a presigned S3 bucket that receives responses from the custom resource provider to CloudFormation.

Required: Yes

Type: String

`ResourceType`

The template developer-chosen resource type of the custom resource in the CloudFormation template. Custom resource type names can be up to 60 characters long and can include alphanumeric and the following characters: `_@-`.

Required: Yes

Type: String

`LogicalResourceId`

The template developer-chosen name (logical ID) of the custom resource in the CloudFormation template.

Required: Yes

Type: String

`ResourceProperties`

This field contains the contents of the `Properties` object sent by the template developer. Its contents are defined by the custom resource provider.

Required: No

Type: JSON object

Example

```
{
   "RequestType" : "Create",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
   "ResponseURL" : "pre-signed-url-for-create-response",
   "ResourceType" : "Custom::MyCustomResourceType",
   "LogicalResourceId" : "resource-logical-id",
   "ResourceProperties" : {
      "key1" : "string",
      "key2" : [ "list" ],
```

```
      "key3" : { "key4" : "map" }
   }
}
```

Update

When the template developer makes changes to the properties of a custom resource within the template and updates the stack, CloudFormation sends a request to the custom resource provider with `RequestType` set to `Update`. This means that your custom resource code doesn't have to detect changes in resources because it knows that its properties have changed when the request type is `Update`.

Update requests contain the following fields:

`RequestType`

`Update`.

Required: Yes

Type: String

`RequestId`

A unique ID for the request.

Combining the `StackId` with the `RequestId` forms a value that you can use to uniquely identify a request on a particular custom resource.

Required: Yes

Type: String

`StackId`

The Amazon Resource Name (ARN) that identifies the stack that contains the custom resource.

Combining the `StackId` with the `RequestId` forms a value that you can use to uniquely identify a request on a particular custom resource.

Required: Yes

Type: String

`ResponseURL`

The response URL identifies a presigned S3 bucket that receives responses from the custom resource provider to CloudFormation.

Required: Yes

Type: String

`ResourceType`

The template developer-chosen resource type of the custom resource in the CloudFormation template. Custom resource type names can be up to 60 characters long and can include alphanumeric and the following characters: `_@-`. You can't change the type during an update.

Required: Yes

Type: String

`LogicalResourceId`

The template developer-chosen name (logical ID) of the custom resource in the CloudFormation template.

Required: Yes

Type: String

`PhysicalResourceId`

A custom resource provider-defined physical ID that is unique for that provider.

Required: Yes

Type: String

`ResourceProperties`

This field contains the contents of the `Properties` object sent by the template developer. Its contents are defined by the custom resource provider.

Required: No

Type: JSON object

`OldResourceProperties`

Used only for `Update` requests. The resource property values that were previously declared by the template developer in the CloudFormation template.

Required: Yes

Type: JSON object

Example

```
{
   "RequestType" : "Update",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
   "ResponseURL" : "pre-signed-url-for-update-response",
   "ResourceType" : "Custom::MyCustomResourceType",
   "LogicalResourceId" : "resource-logical-id",
   "PhysicalResourceId" : "provider-defined-physical-id",
   "ResourceProperties" : {
      "key1" : "new-string",
      "key2" : [ "new-list" ],
      "key3" : { "key4" : "new-map" }
   },
   "OldResourceProperties" : {
      "key1" : "string",
      "key2" : [ "list" ],
      "key3" : { "key4" : "map" }
   }
}
```

Delete

When the template developer deletes the stack or removes the custom resource from the template and then updates the stack, CloudFormation sends a request with `RequestType` set to `Delete`.

Delete requests contain the following fields:

`RequestType`

`Delete`.

Required: Yes

Type: String

`RequestId`

A unique ID for the request.

Required: Yes

Type: String

`StackId`

The Amazon Resource Name (ARN) that identifies the stack that contains the custom resource.

Required: Yes

Type: String

`ResponseURL`

The response URL identifies a presigned S3 bucket that receives responses from the custom resource provider to CloudFormation.

Required: Yes

Type: String

`ResourceType`

The template developer-chosen resource type of the custom resource in the CloudFormation template. Custom resource type names can be up to 60 characters long and can include alphanumeric and the following characters: `_@-`.

Required: Yes

Type: String

`LogicalResourceId`

The template developer-chosen name (logical ID) of the custom resource in the CloudFormation template.

Required: Yes

Type: String

`PhysicalResourceId`

A custom resource provider-defined physical ID that is unique for that provider.

Required: Yes

Type: String

`ResourceProperties`

This field contains the contents of the `Properties` object sent by the template developer. Its contents are defined by the custom resource provider.

Required: No

Type: JSON object

Example

```
{
   "RequestType" : "Delete",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/id",
   "ResponseURL" : "pre-signed-url-for-delete-response",
   "ResourceType" : "Custom::MyCustomResourceType",
   "LogicalResourceId" : "resource-logical-id",
   "PhysicalResourceId" : "provider-defined-physical-id",
   "ResourceProperties" : {
      "key1" : "string",
      "key2" : [ "list" ],
      "key3" : { "key4" : "map" }
   }
}
```

Response object

The custom resource provider sends a response to the pre-signed URL for all request types. If the custom resource provider doesn't send a response, CloudFormation waits until the operation times out.

The response must be a JSON object with the following fields:

`Status`

Must be either SUCCESS or `FAILED`.

Required: Yes

Type: String

`RequestId`

A unique ID for the request. Copy this value exactly as it appears in the request.

Required: Yes

Type: String

`StackId`

The Amazon Resource Name (ARN) that identifies the stack that contains the custom resource. Copy this value exactly as it appears in the request.

Required: Yes

Type: String

`LogicalResourceId`

The template developer-chosen name (logical ID) of the custom resource in the CloudFormation template. Copy this value exactly as it appears in the request.

Required: Yes

Type: String

`PhysicalResourceId`

This value should be an identifier unique to the custom resource vendor, and can be up to 1 KB in size. The value must be a non-empty string and must be identical for all responses for the same resource.

When updating custom resources, the value returned for `PhysicalResourceId` determines the update behavior. If the value remains the same, CloudFormation considers it a normal update. If the value changes, CloudFormation interprets the update as a replacement and sends a delete request to the old resource. For more information, see AWS::CloudFormation::CustomResource.

Required: Yes

Type: String

`Reason`

Describes the reason for a failure response.

Required if `Status` is `FAILED`. It's optional otherwise.

Required: Conditional

Type: String

`NoEcho`

Indicates whether to mask the output of the custom resource when retrieved by using the `Fn::GetAtt` function. If set to `true`, all returned values are masked with asterisks (*****), *except for those stored in the `Metadata` section of the template*. CloudFormation does not transform, modify, or redact any information you include in the `Metadata` section. The default value is `false`.

For more information about using `NoEcho` to mask sensitive information, see the Do not embed credentials in your templates best practice.

Available for `Create` and `Update` responses only. Not supported for `Delete` responses.

Required: No

Type: Boolean

`Data`

The custom resource provider-defined name-value pairs to send with the response. You can access the values provided here by name in the template with `Fn::GetAtt`.

Available for `Create` and `Update` responses only. Not supported for `Delete` responses.

> **Important**
>
> If the name-value pairs contain sensitive information, you should use the `NoEcho` field to mask the output of the custom resource. Otherwise, the values are visible through APIs that surface property values (such as `DescribeStackEvents`).

Required: No

Type: JSON object

Success Response Examples

`Create` and `Update` Response

```
{
   "Status": "SUCCESS",
   "RequestId": "unique-request-id",
   "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
   "LogicalResourceId": "resource-logical-id",
   "PhysicalResourceId": "provider-defined-physical-id",
   "NoEcho": true,
   "Data": {
      "key1": "value1",
      "key2": "value2"
   }
}
```

`Delete` Response

```
{
   "Status": "SUCCESS",
   "RequestId": "unique-request-id",
   "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
   "LogicalResourceId": "resource-logical-id",
   "PhysicalResourceId": "provider-defined-physical-id"
}
```

Failed Response Example

```
{
   "Status": "FAILED",
   "RequestId": "unique-request-id",
   "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/name/id",
   "LogicalResourceId": "resource-logical-id",
   "PhysicalResourceId": "provider-defined-physical-id",
   "Reason": "Required failure reason string"
}
```

Amazon SNS-backed custom resources

The following topic shows you how to configure a custom resource with a service token that specifies the Amazon SNS topic that CloudFormation sends requests to. You also learn the sequence of events and messages sent and received as a result of custom resource stack creation, updates, and deletion.

With custom resources and Amazon SNS, you can enable scenarios such as adding new resources to a stack and injecting dynamic data into a stack. For example, when you create a stack, CloudFormation can send a `Create` request to a topic that's monitored by an application that's running on an Amazon EC2 instance. The Amazon SNS notification triggers the application to carry out additional provisioning tasks, such as retrieve a pool of allow-listed Elastic IP addresses. After it's done, the application sends a response (and any output data) that notifies CloudFormation to proceed with the stack operation.

When you specify an Amazon SNS topic as the target of a custom resource, CloudFormation sends messages to the specified SNS topic during stack operations involving the custom resource. To process these messages and perform the necessary actions, you must have a supported endpoint subscribed to the SNS topic.

For an introduction to custom resources and how they work, see How custom resources work. For information about Amazon SNS and how it works, see the Amazon Simple Notification Service Developer Guide.

Using Amazon SNS to create custom resources

Topics

- Step 1: Stack creation
- Step 2: Stack updates
- Step 3: Stack deletion

Step 1: Stack creation

1. The template developer creates a CloudFormation stack that contains a custom resource.

   In the template example below, we use the custom resource type name `Custom::SeleniumTester` for the custom resource with logical ID `MySeleniumTest`. Custom resource type names must be alphanumeric and can have a maximum length of 60 characters.

The custom resource type is declared with a service token, optional provider-specific properties, and optional Fn::GetAtt attributes that are defined by the custom resource provider. These properties and attributes can be used to pass information from the template developer to the custom resource provider and vice-versa. The service token specifies an Amazon SNS topic that the resource provider has configured.

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "MySeleniumTest" : {
         "Type": "Custom::SeleniumTester",
         "Version" : "1.0",
         "Properties" : {
            "ServiceToken": "arn:aws:sns:us-west-2:123456789012:CRTest",
            "seleniumTester" : "SeleniumTest()",
            "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/",
 "http://search.mysite.com" ],
            "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
         }
      }
   },
   "Outputs" : {
      "topItem" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
      },
      "numRespondents" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
      }
   }
}
```

Note

The names and values of the data accessed with `Fn::GetAtt` are returned by the custom resource provider during the provider's response to CloudFormation. If the custom resource provider is a third-party, then the template developer must obtain the names of these return values from the custom resource provider.

2. CloudFormation sends an Amazon SNS notification to the resource provider with a `"RequestType" : "Create"` that contains information about the stack, the custom resource properties from the stack template, and an S3 URL for the response.

   The SNS topic that's used to send the notification is embedded in the template in the `ServiceToken` property. To avoid using a hardcoded value, a template developer can use a template parameter so that the value is entered at the time the stack is launched.

   The following example shows a custom resource `Create` request which includes a custom resource type name, `Custom::SeleniumTester`, created with a `LogicalResourceId` of `MySeleniumTester`:

   ```
   {
      "RequestType" : "Create",
      "RequestId" : "unique-request-id",
      "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
   cd98-11ea-90d5-0a9cd3354c10",
      "ResponseURL" : "http://pre-signed-S3-url-for-response",
      "ResourceType" : "Custom::SeleniumTester",
      "LogicalResourceId" : "MySeleniumTester",
      "ResourceProperties" : {
         "seleniumTester" : "SeleniumTest()",
         "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
   search.mysite.com" ],
         "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
      }
   }
   ```

   For detailed information about the request object for `Create` requests, see the Request and response reference topic.

3. The custom resource provider processes the data sent by the template developer and determines whether the `Create` request was successful. The resource provider then uses the S3 URL sent by CloudFormation to send a response of either `SUCCESS` or `FAILED`.

   Depending on the response type, different response fields will be expected by CloudFormation. For information about the response fields for a particular request type, see the documentation for that request type in the Request and response reference section.

   In response to a create or update request, the custom resource provider can return data elements in the `Data` field of the response. These are name value pairs, and the *names*

correspond to the `Fn::GetAtt` attributes used with the custom resource in the stack template. The *values* are the data that's returned when the template developer calls `Fn::GetAtt` on the resource with the attribute name.

The following is an example of a custom resource response:

```
{
   "Status" : "SUCCESS",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1",
   "Data" : {
      "resultsPage" : "http://www.myexampledomain/test-results/guid",
      "lastUpdate" : "2012-11-14T03:30Z"
   }
}
```

For detailed information about the response object for `Create` requests, see the Request and response reference topic.

The `StackId`, `RequestId`, and `LogicalResourceId` fields must be copied verbatim from the request.

4. CloudFormation declares the stack status as `CREATE_COMPLETE` or `CREATE_FAILED`. If the stack was successfully created, the template developer can use the output values of the created custom resource by accessing them with Fn::GetAtt.

   For example, the custom resource template used for illustration used `Fn::GetAtt` to copy resource outputs into the stack outputs:

```
"Outputs" : {
   "topItem" : {
      "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
   },
   "numRespondents" : {
      "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
   }
}
```

Step 2: Stack updates

To update an existing stack, you must submit a template that specifies updates for the properties of resources in the stack, as shown in the example below. CloudFormation updates only the resources that have changes specified in the template. For more information, see Understand update behaviors of stack resources.

You can update custom resources that require a replacement of the underlying physical resource. When you update a custom resource in a CloudFormation template, CloudFormation sends an update request to that custom resource. If a custom resource requires a replacement, the new custom resource must send a response with the new physical ID. When CloudFormation receives the response, it compares the `PhysicalResourceId` between the old and new custom resources. If they're different, CloudFormation recognizes the update as a replacement and sends a delete request to the old resource, as shown in Step 3: Stack deletion.

> **Note**
>
> If you didn't make changes to the custom resource, CloudFormation won't send requests to it during a stack update.

1. The template developer initiates an update to the stack that contains a custom resource. During an update, the template developer can specify new Properties in the stack template.

   The following is an example of an `Update` to the stack template using a custom resource type:

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "MySeleniumTest" : {
         "Type": "Custom::SeleniumTester",
         "Version" : "1.0",
         "Properties" : {
            "ServiceToken": "arn:aws:sns:us-west-2:123456789012:CRTest",
            "seleniumTester" : "SeleniumTest()",
            "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/",
 "http://search.mysite.com",
               "http://mynewsite.com" ],
            "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
         }
      }
```

```
   },
   "Outputs" : {
      "topItem" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "resultsPage"] }
      },
      "numRespondents" : {
         "Value" : { "Fn::GetAtt" : ["MySeleniumTest", "lastUpdate"] }
      }
   }
}
```

2. CloudFormation sends an Amazon SNS notification to the resource provider with a `"RequestType" : "Update"` that contains similar information to the `Create` call, except that the `OldResourceProperties` field contains the old resource properties, and ResourceProperties contains the updated (if any) resource properties.

   The following is an example of an `Update` request:

```
{
   "RequestType" : "Update",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "ResponseURL" : "http://pre-signed-S3-url-for-response",
   "ResourceType" : "Custom::SeleniumTester",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1",
   "ResourceProperties" : {
      "seleniumTester" : "SeleniumTest()",
      "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
search.mysite.com",
         "http://mynewsite.com" ],
      "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
   },
   "OldResourceProperties" : {
      "seleniumTester" : "SeleniumTest()",
      "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
search.mysite.com" ],
      "frequencyOfTestsPerHour" : [ "3", "2", "4" ]
   }
}
```

For detailed information about the request object for `Update` requests, see the Request and response reference topic.

3. The custom resource provider processes the data sent by CloudFormation. The custom resource performs the update and sends a response of either `SUCCESS` or `FAILED` to the S3 URL. CloudFormation then compares the `PhysicalResourceIDs` of old and new custom resources. If they're different, CloudFormation recognizes that the update requires a replacement and sends a delete request to the old resource. The following example demonstrates the custom resource provider response to an `Update` request.

```
{
   "Status" : "SUCCESS",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester2"
}
```

   For detailed information about the response object for `Update` requests, see the Request and response reference topic.

   The `StackId`, `RequestId`, and `LogicalResourceId` fields must be copied verbatim from the request.

4. CloudFormation declares the stack status as `UPDATE_COMPLETE` or `UPDATE_FAILED`. If the update fails, the stack rolls back. If the stack was successfully updated, the template developer can access any new output values of the created custom resource with `Fn::GetAtt`.

Step 3: Stack deletion

1. The template developer deletes a stack that contains a custom resource. CloudFormation gets the current properties specified in the stack template along with the SNS topic, and prepares to make a request to the custom resource provider.
2. CloudFormation sends an Amazon SNS notification to the resource provider with a `"RequestType" : "Delete"` that contains current information about the stack, the custom resource properties from the stack template, and an S3 URL for the response.

Whenever you delete a stack or make an update that removes or replaces the custom resource, CloudFormation compares the `PhysicalResourceId` between the old and new custom resources. If they're different, CloudFormation recognizes the update as a replacement and sends a delete request for the old resource (`OldPhysicalResource`), as shown in the following example of a `Delete` request.

```
{
   "RequestType" : "Delete",
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "ResponseURL" : "http://pre-signed-S3-url-for-response",
   "ResourceType" : "Custom::SeleniumTester",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1",
   "ResourceProperties" : {
      "seleniumTester" : "SeleniumTest()",
      "endpoints" : [ "http://mysite.com", "http://myecommercesite.com/", "http://
search.mysite.com",
         "http://mynewsite.com" ],
      "frequencyOfTestsPerHour" : [ "3", "2", "4", "3" ]
   }
}
```

For detailed information about the request object for `Delete` requests, see the Request and response reference topic.

`DescribeStackResource`, `DescribeStackResources`, and `ListStackResources` display the user-defined name if it has been specified.

3. The custom resource provider processes the data sent by CloudFormation and determines whether the `Delete` request was successful. The resource provider then uses the S3 URL sent by CloudFormation to send a response of either `SUCCESS` or `FAILED`. To successfully delete a stack with a custom resource, the custom resource provider must respond successfully to a delete request.

   The following is an example of a custom resource provider response to a `Delete` request:

```
{
   "Status" : "SUCCESS",
```

```
   "RequestId" : "unique-request-id",
   "StackId" : "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10",
   "LogicalResourceId" : "MySeleniumTester",
   "PhysicalResourceId" : "Tester1"
}
```

For detailed information about the response object for `Delete` requests, see the Request and response reference topic.

The `StackId`, `RequestId`, and `LogicalResourceId` fields must be copied verbatim from the request.

4. CloudFormation declares the stack status as `DELETE_COMPLETE` or `DELETE_FAILED`.

Lambda-backed custom resources

When you associate a Lambda function with a custom resource, the function is invoked whenever the custom resource is created, updated, or deleted. CloudFormation calls a Lambda API to invoke the function and to pass all the request data (such as the request type and resource properties) to the function. The power and customizability of Lambda functions in combination with CloudFormation enable a wide range of scenarios, such as dynamically looking up AMI IDs during stack creation, or implementing and using utility functions, such as string reversal functions.

For an introduction to custom resources and how they work, see How custom resources work.

Topics

- Walkthrough: Create a delay mechanism with a Lambda-backed custom resource
- cfn-response module

Walkthrough: Create a delay mechanism with a Lambda-backed custom resource

This walkthrough shows you how to configure and launch a Lambda-backed custom resource using a sample CloudFormation template. This template creates a delay mechanism that pauses stack deployments for a specified time. This can be useful when you need to introduce deliberate delays during resource provisioning, such as when waiting for resources to stabilize before dependent resources are created.

> **Note**
>
> While Lambda-backed custom resources were previously recommended for retrieving AMI IDs, we now recommend using AWS Systems Manager parameters. This approach makes your templates more reusable and easier to maintain. For more information, see Get a plaintext value from Systems Manager Parameter Store.

Topics

- Overview
- Sample template
- Sample template walkthrough
- Prerequisites
- Launching the stack
- Cleaning up resources
- Related information

Overview

The sample stack template used in this walkthrough creates a Lambda-backed custom resource. This custom resource introduces a configurable delay (60 seconds by default) during stack creation. The delay occurs during stack updates only when the custom resource's properties are modified.

The template provisions the following resources:

- a custom resource,
- a Lambda function, and
- an IAM role that enables Lambda to write logs to CloudWatch.

It also defines two outputs:

- The actual time the function waited.
- A unique identifier generated during each execution of the Lambda function.

Note

CloudFormation is a free service but Lambda charges based on the number of requests for your functions and the time your code executes. For more information about Lambda pricing, see AWS Lambda pricing.

Sample template

You can see the Lambda-backed custom resource sample template with the delay mechanism below:

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "LambdaExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": { "Service": ["lambda.amazonaws.com"] },
            "Action": ["sts:AssumeRole"]
          }]
        },
        "Path": "/",
        "Policies": [{
          "PolicyName": "AllowLogs",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": ["logs:*"],
              "Resource": "*"
            }]
          }
        }]
      }
    },
    "CFNWaiter": {
      "Type": "AWS::Lambda::Function",
```

```
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "python3.9",
        "Timeout": 900,
        "Role": { "Fn::GetAtt": ["LambdaExecutionRole", "Arn"] },
        "Code": {
          "ZipFile": { "Fn::Join": ["\n", [
            "from time import sleep",
            "import json",
            "import cfnresponse",
            "import uuid",
            "",
            "def handler(event, context):",
            "  wait_seconds = 0",
            "  id = str(uuid.uuid1())",
            "  if event[\"RequestType\"] in [\"Create\", \"Update\"]:",
            "    wait_seconds = int(event[\"ResourceProperties\"].get(\"ServiceTimeout
\", 0))",
            "    sleep(wait_seconds)",
            "  response = {",
            "    \"TimeWaited\": wait_seconds,",
            "    \"Id\": id ",
            "  }",
            "  cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
 \"Waiter-\"+id)"
          ]]}
        }
      }
    },
    "CFNWaiterCustomResource": {
      "Type": "AWS::CloudFormation::CustomResource",
      "Properties": {
        "ServiceToken": { "Fn::GetAtt": ["CFNWaiter", "Arn"] },
        "ServiceTimeout": 60
      }
    }
  },
  "Outputs": {
    "TimeWaited": {
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "TimeWaited"] },
      "Export": { "Name": "TimeWaited" }
    },
    "WaiterId": {
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "Id"] },
```

```
      "Export": { "Name": "WaiterId" }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - "lambda.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        - PolicyName: "AllowLogs"
          PolicyDocument:
            Statement:
              - Effect: "Allow"
                Action:
                  - "logs:*"
                Resource: "*"
  CFNWaiter:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Runtime: python3.9
      Timeout: 900
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile:
          !Sub |
          from time import sleep
          import json
          import cfnresponse
          import uuid
```

```
          def handler(event, context):
            wait_seconds = 0
            id = str(uuid.uuid1())
            if event["RequestType"] in ["Create", "Update"]:
              wait_seconds = int(event["ResourceProperties"].get("ServiceTimeout", 0))
              sleep(wait_seconds)
            response = {
              "TimeWaited": wait_seconds,
              "Id": id
            }
            cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
 "Waiter-"+id)
  CFNWaiterCustomResource:
    Type: AWS::CloudFormation::CustomResource
    Properties:
      ServiceToken: !GetAtt CFNWaiter.Arn
      ServiceTimeout: 60
Outputs:
  TimeWaited:
    Value: !GetAtt CFNWaiterCustomResource.TimeWaited
    Export:
      Name: TimeWaited
  WaiterId:
    Value: !GetAtt CFNWaiterCustomResource.Id
    Export:
      Name: WaiterId
```

Sample template walkthrough

The following snippets explain relevant parts of the sample template to help you understand how the Lambda function is associated with a custom resource and understand the output.

AWS::Lambda::Function resource `CFNWaiter`

The `AWS::Lambda::Function` resource specifies the function's source code, handler name, runtime environment, and execution role Amazon Resource Name (ARN).

The `Handler` property is set to `index.handler` since it uses a Python source code. For more information on accepted handler identifiers when using inline function source codes, see AWS::Lambda::Function Code.

The `Runtime` is specified as `python3.9` since the source file is a Python code.

The `Timeout` is set to 900 seconds.

The `Role` property uses the `Fn::GetAtt` function to get the ARN of the `LambdaExecutionRole` execution role that's declared in the `AWS::IAM::Role` resource in the template.

The `Code` property defines the function code inline using a Python function. The Python function in the sample template does the following:

- Create a unique ID using the UUID
- Check if the request is a create or update request
- Sleep for the duration specified for `ServiceTimeout` during `Create` or `Update` requests
- Return the wait time and unique ID

JSON

```
...
    "CFNWaiter": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "python3.9",
        "Timeout": 900,
        "Role": { "Fn::GetAtt": ["LambdaExecutionRole", "Arn"] },
        "Code": {
          "ZipFile": { "Fn::Join": ["\n", [
            "from time import sleep",
            "import json",
            "import cfnresponse",
            "import uuid",
            "",
            "def handler(event, context):",
            "  wait_seconds = 0",
            "  id = str(uuid.uuid1())",
            "  if event[\"RequestType\"] in [\"Create\", \"Update\"]:",
            "    wait_seconds = int(event[\"ResourceProperties\"].get(\"ServiceTimeout
\", 0))",
            "    sleep(wait_seconds)",
            "  response = {",
            "    \"TimeWaited\": wait_seconds,",
            "    \"Id\": id ",
```

```
            "  }",
            "  cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
 \"Waiter-\"+id)"
          ]]}
        }
      }
    },
...
```

YAML

```
...
  CFNWaiter:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Runtime: python3.9
      Timeout: 900
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile:
          !Sub |
          from time import sleep
          import json
          import cfnresponse
          import uuid

          def handler(event, context):
            wait_seconds = 0
            id = str(uuid.uuid1())
            if event["RequestType"] in ["Create", "Update"]:
              wait_seconds = int(event["ResourceProperties"].get("ServiceTimeout", 0))
              sleep(wait_seconds)
            response = {
              "TimeWaited": wait_seconds,
              "Id": id
            }
            cfnresponse.send(event, context, cfnresponse.SUCCESS, response,
 "Waiter-"+id)
...
```

AWS::IAM::Role resource `LambdaExecutionRole`

The `AWS::IAM:Role` resource creates an execution role for the Lambda function, which includes an assume role policy which allows Lambda to use it. It also contains a policy allowing CloudWatch Logs access.

JSON

```
...
    "LambdaExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": { "Service": ["lambda.amazonaws.com"] },
            "Action": ["sts:AssumeRole"]
          }]
        },
        "Path": "/",
        "Policies": [{
          "PolicyName": "AllowLogs",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": ["logs:*"],
              "Resource": "*"
            }]
          }
        }]
      }
    },
...
```

YAML

```
...
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
```

```
          - Effect: "Allow"
            Principal:
              Service:
                - "lambda.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        - PolicyName: "AllowLogs"
          PolicyDocument:
            Statement:
              - Effect: "Allow"
                Action:
                  - "logs:*"
                Resource: "*"
...
```

AWS::CloudFormation::CustomResource resource `CFNWaiterCustomResource`

The custom resource links to the Lambda function with its ARN using `!GetAtt CFNWaiter.Arn`. It will implement a 60 second wait time for create and update operations, as set in `ServiceTimeout`. The resource will only be invoked for an update operation if the properties are modified.

JSON

```
...
    "CFNWaiterCustomResource": {
      "Type": "AWS::CloudFormation::CustomResource",
      "Properties": {
        "ServiceToken": { "Fn::GetAtt": ["CFNWaiter", "Arn"] },
        "ServiceTimeout": 60
      }
    }
  },
...
```

YAML

```
...
  CFNWaiterCustomResource:
```

```
    Type: AWS::CloudFormation::CustomResource
    Properties:
      ServiceToken: !GetAtt CFNWaiter.Arn
      ServiceTimeout: 60
...
```

`Outputs`

The `Outputs` of this template are the `TimeWaited` and the `WaiterId`. The `TimeWaited` value uses a `Fn::GetAtt` function to provide the amount of time the waiter resource actually waited. The `WaiterId` uses a `Fn::GetAtt` function to provide the unique ID that was generated and associated with the execution.

JSON

```
...
  "Outputs": {
    "TimeWaited": {
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "TimeWaited"] },
      "Export": { "Name": "TimeWaited" }
    },
    "WaiterId": {
      "Value": { "Fn::GetAtt": ["CFNWaiterCustomResource", "Id"] },
      "Export": { "Name": "WaiterId" }
    }
  }
}
...
```

YAML

```
...
Outputs:
  TimeWaited:
    Value: !GetAtt CFNWaiterCustomResource.TimeWaited
    Export:
      Name: TimeWaited
  WaiterId:
    Value: !GetAtt CFNWaiterCustomResource.Id
    Export:
      Name: WaiterId
```

```
...
```

Prerequisites

You must have IAM permissions to use all the corresponding services, such as Lambda and CloudFormation.

Launching the stack

To create the stack

1. Find the template of your preference (YAML or JSON) from the Sample template section and save it to your machine with the name `samplelambdabackedcustomresource.template`.
2. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.
3. From the **Stacks** page, choose **Create stack** at top right, and then choose **With new resources (standard)**.
4. For **Prerequisite - Prepare template**, choose **Choose an existing template**.
5. For **Specify template**, choose **Upload a template file**, and then choose **Choose file**.
6. Select the `samplelambdabackedcustomresource.template` template file you saved earlier.
7. Choose **Next**.
8. For **Stack name**, type **`SampleCustomResourceStack`** and choose **Next**.
9. For this walkthrough, you don't need to add tags or specify advanced settings, so choose **Next**.
10. Ensure that the stack name looks correct, and then choose **Create**.

It might take several minutes for CloudFormation to create your stack. To monitor progress, view the stack events. For more information, see View stack information from the CloudFormation console.

If stack creation succeeds, all resources in the stack, such as the Lambda function and custom resource, were created. You have successfully used a Lambda function and custom resource.

If the Lambda function returns an error, view the function's logs in the CloudWatch Logs console. The name of the log stream is the physical ID of the custom resource, which you can find by viewing the stack's resources. For more information, see View log data in the *Amazon CloudWatch User Guide*.

Cleaning up resources

Delete the stack to clean up all the stack resources that you created so that you aren't charged for unnecessary resources.

To delete the stack

1. From the CloudFormation console, choose the **SampleCustomResourceStack** stack.
2. Choose **Actions** and then **Delete Stack**.
3. In the confirmation message, choose **Yes, Delete**.

All the resources that you created are deleted.

Now that you understand how to create and use Lambda-backed custom resource, you can use the sample template and code from this walkthrough to build and experiment with other stacks and functions.

Related information

- CloudFormation Custom Resource Reference
- AWS::CloudFormation::CustomResource

`cfn-response` module

In your CloudFormation template, you can specify a Lambda function as the target of a custom resource. When you use the `ZipFile` property to specify your function's source code, you can load the `cfn-response` module to send responses from your Lambda function to a custom resource. The `cfn-response` module is a library that simplifies sending responses to the custom resource that invoked your Lambda function. The module has a `send` method that sends a response object to a custom resource by way of an Amazon S3 presigned URL (the `ResponseURL`).

The `cfn-response` module is available only when you use the `ZipFile` property to write your source code. It isn't available for source code that's stored in Amazon S3 buckets. For code in buckets, you must write your own functions to send responses.

> **Note**
>
> After executing the `send` method, the Lambda function terminates, so anything you write after that method is ignored.

Loading the `cfn-response` module

For Node.js functions, use the `require()` function to load the `cfn-response` module. For example, the following code example creates a `cfn-response` object with the name `response`:

```
var response = require('cfn-response');
```

For Python, use the `import` statement to load the `cfnresponse` module, as shown in the following example:

> **Note**
>
> Use this exact import statement. If you use other variants of the import statement, CloudFormation doesn't include the response module.

```
import cfnresponse
```

send method parameters

You can use the following parameters with the `send` method.

`event`

The fields in a custom resource request.

`context`

An object, specific to Lambda functions, that you can use to specify when the function and any callbacks have completed execution, or to access information from within the Lambda execution environment. For more information, see Building Lambda functions with Node.js in the *AWS Lambda Developer Guide*.

`responseStatus`

Whether the function successfully completed. Use the `cfnresponse` module constants to specify the status: `SUCCESS` for successful executions and `FAILED` for failed executions.

`responseData`

The `Data` field of a custom resource response object. The data is a list of name-value pairs.

`physicalResourceId`

Optional. The unique identifier of the custom resource that invoked the function. By default, the module uses the name of the Amazon CloudWatch Logs log stream that's associated with the Lambda function.

The value returned for a `PhysicalResourceId` can change custom resource update operations. If the value returned is the same, it's considered a normal update. If the value returned is different, CloudFormation recognizes the update as a replacement and sends a delete request to the old resource. For more information, see AWS::CloudFormation::CustomResource.

`noEcho`

Optional. Indicates whether to mask the output of the custom resource when it's retrieved by using the `Fn::GetAtt` function. If set to `true`, all returned values are masked with asterisks (*****), except for information stored in the locations specified below. By default, this value is `false`.

Important

Using the `NoEcho` attribute does not mask any information stored in the following:

- The `Metadata` template section. CloudFormation does not transform, modify, or redact any information you include in the `Metadata` section. For more information, see Metadata.
- The `Outputs` template section. For more information, see Outputs.
- The `Metadata` attribute of a resource definition. For more information, see `Metadata` attribute.

We strongly recommend you do not use these mechanisms to include sensitive information, such as passwords or secrets.

For more information about using NoEcho to mask sensitive information, see the Do not embed credentials in your templates best practice.

Examples

Node.js

In the following Node.js example, the inline Lambda function takes an input value and multiplies it by 5. Inline functions are especially useful for smaller functions because they allow you to specify the source code directly in the template, instead of creating a package and uploading it to an Amazon S3 bucket. The function uses the `cfn-response` send method to send the result back to the custom resource that invoked it.

JSON

```
"ZipFile": { "Fn::Join": ["", [
  "var response = require('cfn-response');",
  "exports.handler = function(event, context) {",
  "  var input = parseInt(event.ResourceProperties.Input);",
  "  var responseData = {Value: input * 5};",
  "  response.send(event, context, response.SUCCESS, responseData);",
  "};"
]]}
```

YAML

```
ZipFile: >
  var response = require('cfn-response');
  exports.handler = function(event, context) {
    var input = parseInt(event.ResourceProperties.Input);
    var responseData = {Value: input * 5};
    response.send(event, context, response.SUCCESS, responseData);
  };
```

Python

In the following Python example, the inline Lambda function takes an integer value and multiplies it by 5.

JSON

```
"ZipFile" : { "Fn::Join" : ["\n", [
  "import json",
  "import cfnresponse",
  "def handler(event, context):",
  "   responseValue = int(event['ResourceProperties']['Input']) * 5",
  "   responseData = {}",
  "   responseData['Data'] = responseValue",
  "   cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
 \"CustomResourcePhysicalID\")"
]]}
```

YAML

```
ZipFile: |
  import json
  import cfnresponse
  def handler(event, context):
    responseValue = int(event['ResourceProperties']['Input']) * 5
    responseData = {}
    responseData['Data'] = responseValue
    cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
 "CustomResourcePhysicalID")
```

Module source code

Topics

- Asynchronous Node.js source code
- Node.js source code
- Python source code

Asynchronous Node.js source code

The following is the response module source code for the Node.js functions if the handler is asynchronous. Review it to understand what the module does and for help with implementing your own response functions.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0
```

```
exports.SUCCESS = "SUCCESS";
exports.FAILED = "FAILED";

exports.send = function(event, context, responseStatus, responseData,
 physicalResourceId, noEcho) {

    return new Promise((resolve, reject) => {
        var responseBody = JSON.stringify({
            Status: responseStatus,
            Reason: "See the details in CloudWatch Log Stream: " +
 context.logStreamName,
            PhysicalResourceId: physicalResourceId || context.logStreamName,
            StackId: event.StackId,
            RequestId: event.RequestId,
            LogicalResourceId: event.LogicalResourceId,
            NoEcho: noEcho || false,
            Data: responseData
        });

        console.log("Response body:\n", responseBody);

        var https = require("https");
        var url = require("url");

        var parsedUrl = url.parse(event.ResponseURL);
        var options = {
            hostname: parsedUrl.hostname,
            port: 443,
            path: parsedUrl.path,
            method: "PUT",
            headers: {
                "content-type": "",
                "content-length": responseBody.length
            }
        };

        var request = https.request(options, function(response) {
            console.log("Status code: " + parseInt(response.statusCode));
            resolve(context.done());
        });

        request.on("error", function(error) {
```

```
            console.log("send(..) failed executing https.request(..): " +
 maskCredentialsAndSignature(error));
            reject(context.done(error));
        });

        request.write(responseBody);
        request.end();
    })
}

function maskCredentialsAndSignature(message) {
    return message.replace(/X-Amz-Credential=[^&\s]+/i, 'X-Amz-Credential=*****')
        .replace(/X-Amz-Signature=[^&\s]+/i, 'X-Amz-Signature=*****');
}
```

Node.js source code

The following is the response module source code for the Node.js functions if the handler is not asynchronous. Review it to understand what the module does and for help with implementing your own response functions.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: MIT-0

exports.SUCCESS = "SUCCESS";
exports.FAILED = "FAILED";

exports.send = function(event, context, responseStatus, responseData,
 physicalResourceId, noEcho) {

    var responseBody = JSON.stringify({
        Status: responseStatus,
        Reason: "See the details in CloudWatch Log Stream: " + context.logStreamName,
        PhysicalResourceId: physicalResourceId || context.logStreamName,
        StackId: event.StackId,
        RequestId: event.RequestId,
        LogicalResourceId: event.LogicalResourceId,
        NoEcho: noEcho || false,
        Data: responseData
    });

    console.log("Response body:\n", responseBody);
```

```
    var https = require("https");
    var url = require("url");

    var parsedUrl = url.parse(event.ResponseURL);
    var options = {
        hostname: parsedUrl.hostname,
        port: 443,
        path: parsedUrl.path,
        method: "PUT",
        headers: {
            "content-type": "",
            "content-length": responseBody.length
        }
    };

    var request = https.request(options, function(response) {
        console.log("Status code: " + parseInt(response.statusCode));
        context.done();
    });

    request.on("error", function(error) {
        console.log("send(..) failed executing https.request(..): " +
 maskCredentialsAndSignature(error));
        context.done();
    });

    request.write(responseBody);
    request.end();
}
```

Python source code

The following is the response module source code for Python functions:

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0

from __future__ import print_function
import urllib3
import json
import re

SUCCESS = "SUCCESS"
FAILED = "FAILED"
```

```
http = urllib3.PoolManager()


def send(event, context, responseStatus, responseData, physicalResourceId=None,
 noEcho=False, reason=None):
    responseUrl = event['ResponseURL']

    responseBody = {
        'Status' : responseStatus,
        'Reason' : reason or "See the details in CloudWatch Log Stream:
 {}".format(context.log_stream_name),
        'PhysicalResourceId' : physicalResourceId or context.log_stream_name,
        'StackId' : event['StackId'],
        'RequestId' : event['RequestId'],
        'LogicalResourceId' : event['LogicalResourceId'],
        'NoEcho' : noEcho,
        'Data' : responseData
    }

    json_responseBody = json.dumps(responseBody)

    print("Response body:")
    print(json_responseBody)

    headers = {
        'content-type' : '',
        'content-length' : str(len(json_responseBody))
    }

    try:
        response = http.request('PUT', responseUrl, headers=headers,
 body=json_responseBody)
        print("Status code:", response.status)


    except Exception as e:

        print("send(..) failed executing http.request(..):",
 mask_credentials_and_signature(e))


def mask_credentials_and_signature(message):
```

```
    message = re.sub(r'X-Amz-Credential=[^&\s]+', 'X-Amz-Credential=*****', message,
flags=re.IGNORECASE)
    return re.sub(r'X-Amz-Signature=[^&\s]+', 'X-Amz-Signature=*****', message,
flags=re.IGNORECASE)
```

Perform custom processing on CloudFormation templates with template macros

With macros, you can perform custom processing on templates, from simple actions like find-and-replace operations to extensive transformations of entire templates.

To get an idea of the breadth of possibilities, consider the `AWS::Include` and `AWS::Serverless` transforms, which are macros hosted by CloudFormation:

- AWS::Include transform enables you to insert boilerplate template snippets into your templates.
- AWS::Serverless transform takes an entire template written in the AWS Serverless Application Model (AWS SAM) syntax and transforms and expands it into a compliant CloudFormation template. For more information about serverless applications and AWS SAM, see AWS Serverless Application Model Developer Guide.

Topics

- Billing
- Macro examples
- Related resources
- Overview of CloudFormation macros
- Create a CloudFormation macro definition
- Example simple string replacement macro
- Troubleshoot the processed template

Billing

When a macro runs, the owner of the Lambda function is billed for any charges related to the execution of that function.

The `AWS::Include` and `AWS::Serverless` transforms are macros hosted by CloudFormation. There is no charge for using them.

Macro examples

In addition to the examples in this section, you can find example macros, including source code and templates, in our GitHub repository. These examples are provided 'as-is' for instructional purposes.

Related resources

- AWS::CloudFormation::Macro
- CloudFormation template Transform section
- Fn::Transform
- AWS::Serverless transform
- AWS::Include transform

Overview of CloudFormation macros

There are two major steps to processing templates using macros: creating the macro itself, and then using the macro to perform processing on your templates.

To create a macro definition, you must create the following:

- A Lambda function to perform the template processing. This Lambda function accepts either a snippet or an entire template, and any additional parameters that you define. It returns the processed template snippet or the entire template as a response.
- A resource of type AWS::CloudFormation::Macro, which enables users to call the Lambda function from within CloudFormation templates. This resource specifies the ARN of the Lambda function to invoke for this macro, and additional optional properties to assist with debugging. To create this resource within an account, author a template that includes the `AWS::CloudFormation::Macro` resource, and then create either a stack or stack set with self-managed permissions from the template. CloudFormation StackSets doesn't currently support creating or updating stack sets with service-managed permissions from templates that reference macros.

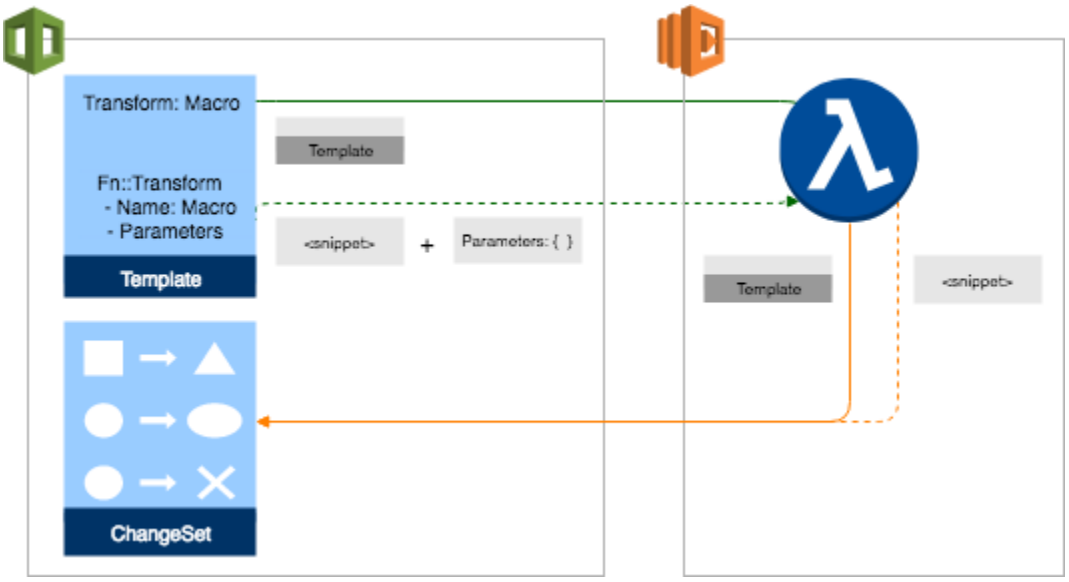
To use a macro, reference the macro in your template:

- To process a section, or part, of a template, reference the macro in an `Fn::Transform` function located relative to the template content you want to transform. When using `Fn::Transform`, you can also pass any specified parameters it requires.

- To process an entire template, reference the macro in the Transform section of the template.

Next, you typically create a change set and then execute it. (Processing macros can add multiple resources that you might not be aware of. To ensure that you're aware of all of the changes introduced by macros, we strongly advise that you use change sets.) CloudFormation passes the specified template content, along with any additional specified parameters, to the Lambda function specified in the macro resource. The Lambda function returns the processed template content, be it a snippet or an entire template.

After all macros in the template have been called, CloudFormation generates a change set that includes the processed template content. After you review the change set, execute it to apply the changes.



How to create stacks directly

To create or update a stack using a template that references macros, you typically create a change set and then execute it. A change set describes the actions CloudFormation will take based on the processed template. Processing macros can add multiple resources that you might not be aware of. To ensure that you're aware of all the changes introduced by macros, we strongly suggest you use change sets. After you review the change set, you can execute it to actually apply the changes.

A macro can add IAM resources to your template. For these resources, CloudFormation requires you to acknowledge their capabilities. Because CloudFormation can't know which resources are added before processing your template, you might need to acknowledge IAM capabilities when you create the change set, depending on whether the referenced macros contain IAM resources. That

way, when you run the change set, CloudFormation has the necessary capabilities to create IAM resources.

To create or update a stack directly from a processed template without first reviewing the proposed changes in a change set, specify the `CAPABILITY_AUTO_EXPAND` capability during a `CreateStack` or `UpdateStack` request. You should only create stacks directly from a stack template that contains macros if you know what processing the macro performs. You can't use change sets with stack set macros; you must update your stack set directly.

For more information, see CreateStack or UpdateStack in the *AWS CloudFormation API Reference*.

> **Important**
>
> If your stack set template references one or more macros, you must create the stack set directly from the processed template, without first reviewing the resulting changes in a change set. Processing macros can add multiple resources that you might not be aware of. Before you create or update a stack set from a template that references macros directly, be sure that you know what processing the macros performs.

To reduce the number of steps for launching stacks from templates that reference macros, you can use the `package` and `deploy` AWS CLI commands. For more information, see Upload local artifacts to an S3 bucket with the AWS CLI and Create a stack that includes transforms.

Considerations

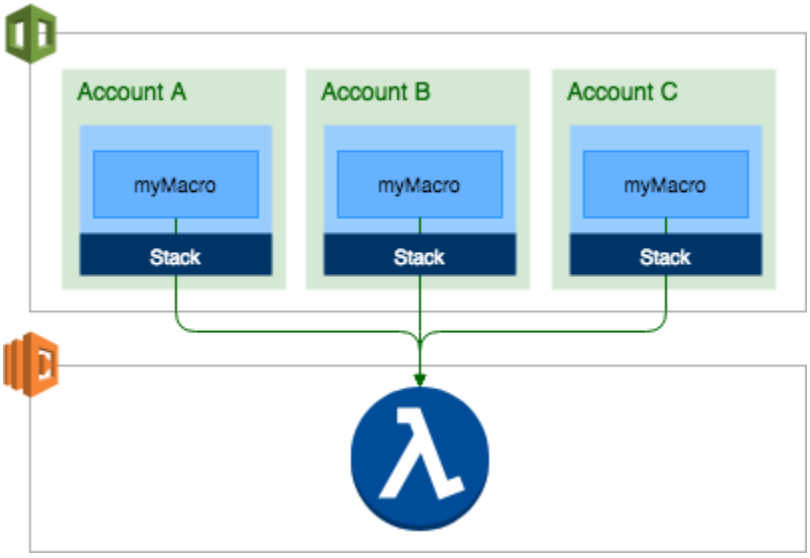
When working with macros, keep in mind the following notes and limitations:

- Macros are supported only in AWS Regions where Lambda is available. For a list of Regions where Lambda is available, see AWS Lambda endpoints and quotas.
- Any processed template snippets must be valid JSON.
- Any processed template snippets must pass validation checks for a create stack, update stack, create stack set, or update stack set operation.
- CloudFormation resolves macros first, and then processes the template. The resulting template must be valid JSON and must not exceed the template size limit.
- Because of the order in which CloudFormation processes elements in a template, a macro can't include modules in the processed template content it returns to CloudFormation. For more information, see Macro evaluation order.

- When using the update rollback feature, CloudFormation uses a copy of the original template. It rolls back to the original template even if the included snippet was changed.
- Including macros within macros doesn't work because we don't process macros recursively.
- The `Fn::ImportValue` intrinsic function isn't currently supported in macros.
- Intrinsic functions included in the template are evaluated after any macros. Therefore, the processed template content your macro returns can include calls to intrinsic functions, and they're evaluated as usual.
- StackSets doesn't currently support creating or updating stack sets with service-managed permissions from templates that reference CloudFormation macros.

Macro account scope and permissions

You can use macros only in the account in which they were created as a resource. The name of the macro must be unique within a given account. However, you can make the same functionality available in multiple accounts by enabling cross-account access on the underlying Lambda function, and then creating macro definitions referencing that function in multiple accounts. In the example below, three accounts contain macro definitions that each point to the same Lambda function.



To create a macro definition, the user must have permissions to create a stack or stack set within the specified account.

For CloudFormation to successfully run a macro included in a template, the user must have `Invoke` permissions for the underlying Lambda function. To prevent potential escalation of permissions, CloudFormation impersonates the user while running the macro.

For more information, see Managing permissions in AWS Lambda in the *AWS Lambda Developer Guide* and Actions, resources, and condition keys for AWS Lambda in the *Service Authorization Reference*.

Create a CloudFormation macro definition

When you create a macro definition, the macro definition makes the underlying Lambda function available in the specified account so that CloudFormation invokes it to process the templates.

Event mapping

When CloudFormation invokes a macro's Lambda function, it sends a request in JSON format with the following structure:

```
{
    "region" : "us-east-1",
    "accountId" : "$ACCOUNT_ID",
    "fragment" : { ... },
    "transformId" : "$TRANSFORM_ID",
    "params" : { ... },
    "requestId" : "$REQUEST_ID",
    "templateParameterValues" : { ... }
}
```

- `region`

  The Region in which the macro resides.
- `accountId`

  The account ID of the account from which the macro is invoking the Lambda function.
- `fragment`

  The template content available for custom processing, in JSON format.
  - For macros included in the `Transform` template section, this is the entire template except for the `Transform` section.
  - For macros included in an `Fn::Transform` intrinsic function call, this includes all sibling nodes (and their children) based on the location of the intrinsic function within the template except for the `Fn::Transform` function. For more information, see Macro template scope.
- `transformId`

The name of the macro invoking this function.

- `params`

  For `Fn::Transform` function calls, any specified parameters for the function. CloudFormation doesn't evaluate these parameters before passing them to the function.

  For macros included in the `Transform` template section, this section is empty.

- `requestId`

  The ID of the request invoking this function.

- `templateParameterValues`

  Any parameters specified in the Parameters section of the template. CloudFormation evaluates these parameters before passing them to the function.

Response format

CloudFormation expects the Lambda function to return a response in the following JSON format:

```
{
    "requestId" : "$REQUEST_ID",
    "status" : "$STATUS",
    "fragment" : { ... },
    "errorMessage": "optional error message for failures"
}
```

- `requestId`

  The ID of the request invoking this function. This must match the request ID provided by CloudFormation when invoking the function.

- `status`

  The status of the request (case-insensitive). Should be set to `success`. CloudFormation treats any other response as a failure.

- `fragment`

The processed template content for CloudFormation to include in the processed template, including siblings. CloudFormation replaces the template content that is passed to the Lambda function with the template fragment it receives in the Lambda response.

The processed template content must be valid JSON, and its inclusion in the processed template must result in a valid template.

If your function doesn't actually change the template content that CloudFormation passes to it, but you still need to include that content in the processed template, your function needs to return that template content to CloudFormation in its response.

- `errorMessage`

  The error message that explains why the transform failed. CloudFormation displays this error message in the **Events** pane of the **Stack details** page for your stack.

  For example:

```
Error creating change set: Transform
                           AWS account account
                           number::macro name failed with:
                           error message string.
```

Create a macro definition

To create a CloudFormation macro definition

1. Build a Lambda function that will handle the processing of template contents. It can process any part of a template, up to the entire template.
2. Create a CloudFormation template containing an `AWS::CloudFormation::Macro` resource type and specify the `Name` and `FunctionName` properties. The `FunctionName` property must contain the ARN of the Lambda function to invoke when CloudFormation runs the macro.
3. (Optional) To aid in debugging, you can also specify the `LogGroupName` and `LogRoleArn` properties when creating the `AWS::CloudFormation::Macro` resource type for your macro. These properties enable you to specify the CloudWatch Logs log group to which CloudFormation sends error logging information when invoking the macro's underlying Lambda function, and the role CloudFormation should assume when sending log entries to those logs.

4. Create a stack using the template with the macro in the account you want to use it in. Or, create a stack set with self-managed permissions using the template with the macro in the administrator account, and then create stack instances in the target accounts.
5. After CloudFormation has successfully created the stacks that contain the macro definition, the macro is available for use within those accounts. You use a macro by referencing it in a template, at the appropriate location relevant to the template contents you want to process.

Macro template scope

Macros referenced in the `Transform` section of a template can process the entire contents of that template.

Macros referenced in an `Fn::Transform` function can process the contents of any of the sibling elements (including children) of that `Fn::Transform` function in the template.

For example, in the template sample below, `AWS::Include` can process the `MyBucket` properties, based on the location of the `Fn::Transform` function that contains it. `MyMacro` can process the contents of the entire template because of its inclusion in the `Transform` section.

```
# Start of processable content for MyMacro
AWSTemplateFormatVersion: 2010-09-09
 Transform: [MyMacro]
 Resources:
    WaitCondition:
      Type: AWS::CloudFormation::WaitCondition
    MyBucket:
      Type: AWS::S3::Bucket
      # Start of processable content for AWS::Include
      Properties:
        BucketName: amzn-s3-demo-bucket1
        Tags: [{"key":"value"}]
        'Fn::Transform':
          - Name: 'AWS::Include'
              Parameters:
                Location: s3://amzn-s3-demo-bucket2/MyFileName.yaml
        CorsConfiguration: []
        # End of processable content for AWS::Include
    MyEc2Instance:
      Type: AWS::EC2::Instance
      Properties:
        ImageID: ami-1234567890abcdef0
```

```
# End of processable content for MyMacro
```

Macro evaluation order

You can reference multiple macros in a given template, including transforms hosted by CloudFormation, such as AWS::Include and AWS::Serverless.

Macros are evaluated in order, based on their location in the template, from the most deeply nested outward to the most general. Macros at the same location in the template are evaluated serially based on the order in which they're listed.

Transforms such as `AWS::Include` and `AWS::Transform` are treated the same as any other macros in terms of action order and scope.

For example, in the template sample below, CloudFormation evaluates the `PolicyAdder` macro first, because it's the most deeply-nested macro in the template. CloudFormation then evaluates `MyMacro` before evaluating `AWS::Serverless` because it's listed before `AWS::Serverless` in the `Transform` section.

```
AWSTemplateFormatVersion: 2010-09-09
 Transform: [MyMacro, AWS::Serverless]
 Resources:
    WaitCondition:
      Type: AWS::CloudFormation::WaitCondition
    MyBucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: amzn-s3-demo-bucket
        Tags: [{"key":"value"}]
        'Fn::Transform':
          - Name: PolicyAdder
        CorsConfiguration: []
    MyEc2Instance:
      Type: AWS::EC2::Instance
      Properties:
        ImageID: ami-1234567890abcdef0
```

Example simple string replacement macro

The following example walks you through the process of using macros, from defining the macro in a template, to creating a Lambda function for the macro, and then to using the macro in a template.

In this example, we create a simple macro that inserts the specified string in place of the specified target content in the processed template. And then we'll use it to insert a blank `WaitHandleCondition` in the specified location in the processed template.

Creating a macro

Before using a macro, we first have to complete two things: create the Lambda function that performs the desired template processing, and then make that Lambda function available to CloudFormation by creating a macro definition.

The following sample template contains the definition for our example macro. To make the macro available in a specific AWS account, create a stack from the template. The macro definition specifies the macro name, a brief description, and references the ARN of the Lambda function that CloudFormation invokes when this macro is used in a template. (We haven't included a `LogGroupName` or `LogRoleARN` property for error logging.)

In this example, assume that the stack created from this template is named `JavaMacroFunc`. Because the macro `Name` property is set to the stack name, the resulting macro is named `JavaMacroFunc` as well.

```
AWSTemplateFormatVersion: 2010-09-09
  Resources:
    Macro:
      Type: AWS::CloudFormation::Macro
      Properties:
        Name: !Sub '${AWS::StackName}'
        Description: Adds a blank WaitConditionHandle named WaitHandle
        FunctionName: 'arn:aws:lambda:us-east-1:012345678910:function:JavaMacroFunc'
```

Using the macro

To use our macro, we include it in a template using the `Fn::Transform` intrinsic function.

When we create a stack using the template below, CloudFormation calls our example macro. The underlying Lambda function replaces one specified string with another specified string. In this case, the result is a blank `AWS::CloudFormation::WaitConditionHandle` is inserted into the processed template.

```
Parameters:
  ExampleParameter:
```

```
    Type: String
    Default: 'SampleMacro'

Resources:
  2a:
    Fn::Transform:
      Name: "JavaMacroFunc"
      Parameters:
        replacement: 'AWS::CloudFormation::WaitConditionHandle'
        target: '$$REPLACEMENT$$'
    Type: '$$REPLACEMENT$$'
```

- The macro to invoke is specified as `JavaMacroFunc`, which is from the previous macro definition example.
- The macro is passed two parameters, `target` and `replacement`, which represent the target string and its desired replacement value.
- The macro can operate on the contents of the `Type` node because `Type` is a sibling of the `Fn::Transform` function referencing the macro.
- The resulting `AWS::CloudFormation::WaitConditionHandle` is named `2a`.
- The template also contains a template parameter, `ExampleParameter`, which the macro also has access to (but doesn't use in this case).

Lambda input data

When CloudFormation processes our example template during stack creation, it passes the following event mapping to the Lambda function referenced in the `JavaMacroFunc` macro definition.

- `region` : `us-east-1`
- `accountId` : `012345678910`
- `fragment` :

```
{
  "Type": "$$REPLACEMENT$$"
}
```

- `transformId` : `012345678910::JavaMacroFunc`
- `params` :

```
{
    "replacement": "AWS::CloudFormation::WaitConditionHandle",
    "target": "$$REPLACEMENT$$"
}
```

- `requestId` : `5dba79b5-f117-4de0-9ce4-d40363bfb6ab`
- `templateParameterValues` :

```
{
    "ExampleParameter": "SampleMacro"
}
```

`fragment` contains JSON representing the template fragment that the macro can process. This fragment consists of the siblings of the `Fn::Transform` function call, but not the function call itself. Also, `params` contains JSON representing the macro parameters. In this case, replacement and target. Similarly, `templateParameterValues` contains JSON representing the parameters specified for the template as a whole.

Lambda function code

Following is the actual code for the Lambda function underlying the `JavaMacroFunc` example macro. It iterates over the template fragment included in the response (be it in string, list, or map format), looking for the specified target string. If it finds the specified target string, the Lambda function replaces the target string with the specified replacement string. If not, the function leaves the template fragment unchanged. Then, the function returns a map of the expected properties, discussed in detail below, to CloudFormation.

```
package com.macroexample.lambda.demo;

import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class LambdaFunctionHandler implements RequestHandler<Map<String, Object>,
 Map<String, Object>> {
```

```
private static final String REPLACEMENT = "replacement";
private static final String TARGET = "target";
private static final String PARAMS = "params";
private static final String FRAGMENT = "fragment";
private static final String REQUESTID = "requestId";
private static final String STATUS = "status";
private static final String SUCCESS = "SUCCESS";
private static final String FAILURE = "FAILURE";
   @Override
   public Map<String, Object> handleRequest(Map<String, Object> event, Context
context) {
       // TODO: implement your handler
    final Map<String, Object> responseMap = new HashMap<String, Object>();
       responseMap.put(REQUESTID, event.get(REQUESTID));
       responseMap.put(STATUS, FAILURE);
    try {
        if (!event.containsKey(PARAMS)) {
         throw new RuntimeException("Params are required");
        }

        final Map<String, Object> params = (Map<String, Object>) event.get(PARAMS);
        if (!params.containsKey(REPLACEMENT) || !params.containsKey(TARGET)) {
         throw new RuntimeException("replacement or target under Params are
required");
        }

     final String replacement = (String) params.get(REPLACEMENT);
     final String target = (String) params.get(TARGET);
     final Object fragment = event.getOrDefault(FRAGMENT, new HashMap<String,
Object>());
     final Object retFragment;
     if (fragment instanceof String) {
      retFragment = iterateAndReplace(replacement, target, (String) fragment);
     } else if (fragment instanceof List) {
      retFragment = iterateAndReplace(replacement, target, (List<Object>) fragment);
     } else if (fragment instanceof Map) {
      retFragment = iterateAndReplace(replacement, target, (Map<String, Object>)
fragment);
     } else {
      retFragment = fragment;
     }
        responseMap.put(STATUS, SUCCESS);
        responseMap.put(FRAGMENT, retFragment);
```

```
        return responseMap;
    } catch (Exception e) {
     e.printStackTrace();
     context.getLogger().log(e.getMessage());
     return responseMap;
    }
   }

   private Map<String, Object> iterateAndReplace(final String replacement, final
String target, final Map<String, Object> fragment) {
    final Map<String, Object> retFragment = new HashMap<String, Object>();
    final List<String> replacementKeys = new ArrayList<>();
    fragment.forEach((k, v) -> {
     if (v instanceof String) {
      retFragment.put(k, iterateAndReplace(replacement, target, (String)v));
     } else if (v instanceof List) {
      retFragment.put(k, iterateAndReplace(replacement, target, (List<Object>)v));
     } else if (v instanceof Map ) {
      retFragment.put(k, iterateAndReplace(replacement, target, (Map<String, Object>)
v));
     } else {
      retFragment.put(k, v);
     }
    });
    return retFragment;
   }

   private List<Object> iterateAndReplace(final String replacement, final String
target, final List<Object> fragment) {
    final List<Object> retFragment = new ArrayList<>();
    fragment.forEach(o -> {
     if (o instanceof String) {
      retFragment.add(iterateAndReplace(replacement, target, (String) o));
     } else if (o instanceof List) {
      retFragment.add(iterateAndReplace(replacement, target, (List<Object>) o));
     } else if (o instanceof Map) {
      retFragment.add(iterateAndReplace(replacement, target, (Map<String, Object>)
o));
     } else {
      retFragment.add(o);
     }
    });
    return retFragment;
   }
```

```
    private String iterateAndReplace(final String replacement, final String target,
 final String fragment) {
     System.out.println(replacement + " == " + target + " == " + fragment );
     if (fragment != null AND_AND fragment.equals(target))
      return replacement;
     return fragment;
    }
}
```

Lambda function response

Following is the mapping that the Lambda function returns to CloudFormation for processing.

- `requestId` : `5dba79b5-f117-4de0-9ce4-d40363bfb6ab`
- `status` : SUCCESS
- `fragment` :

```
{
  "Type": "AWS::CloudFormation::WaitConditionHandle"
}
```

The `requestId` matches that sent from CloudFormation, and a `status` value of SUCCESS denotes that the Lambda function successfully processed the template fragment included in the request. In this response, `fragment` contains JSON representing the content to insert into the processed template in place of the original template snippet.

Resulting processed template

After CloudFormation receives a successful response from the Lambda function, it inserts the returned template fragment into the processed template.

Below is the resulting processed template for our example. The `Fn::Transform` intrinsic function call that referenced the `JavaMacroFunc` macro is no longer included. The template fragment returned by the Lambda function is included in the appropriate location, with the result that the content `"Type": "$$REPLACEMENT$$"` has been replaced with `"Type": "AWS::CloudFormation::WaitConditionHandle"`.

```
{
    "Parameters": {
```

```
        "ExampleParameter": {
            "Default": "SampleMacro",
            "Type": "String"
        }
    },
    "Resources": {
        "2a": {
            "Type": "AWS::CloudFormation::WaitConditionHandle"
        }
    }
}
```

Troubleshoot the processed template

When using a macro, the processed template can be found in the CloudFormation console.

The stage of a template indicates its processing status:

- `Original`: The template that the user originally submitted to create or update the stack or stack set.
- `Processed`: The template CloudFormation used to create or update the stack or stack set after processing any referenced macros. The processed template is formatted as JSON, even if the original template was formatted as YAML.

For troubleshooting, use the processed template. If a template doesn't reference macros, the original and processed templates are identical.

For more information, see View stack information from the CloudFormation console.

To use the AWS CLI to get the processed template, use the get-template command.

Size limitation

The maximum size for a processed stack template is 51,200 bytes when passed directly into a `CreateStack`, `UpdateStack`, or `ValidateTemplate` request, or 1 MB when passed as an S3 object using an Amazon S3 template URL. However, during processing CloudFormation updates the temporary state of the template as it serially processes the macros contained in the template. Because of this, the size of the template during processing may temporarily exceed the allowed size of a fully-processed template. CloudFormation allows some buffer for these in-process templates. However, you should design your templates and macros keeping in mind the maximum allowed size for a processed stack template.

If CloudFormation returns a `Transformation data limit exceeded` error while processing your template, your template has exceeded the maximum template size CloudFormation allows during processing.

To resolve this issue, consider doing the following:

- Restructure your template into multiple templates to avoid exceeding the maximum size for in-process templates. For example:
  - Use nested stack templates to encapsulate parts of the template. For more information, see Split a template into reusable pieces using nested stacks.
  - Create multiple stacks and use cross-stack references to exchange information between them. For more information, see Refer to resource outputs in another CloudFormation stack.
- Reduce the size of template fragment returned by a given macro. CloudFormation doesn't tamper with the contents of fragments returned by macros.

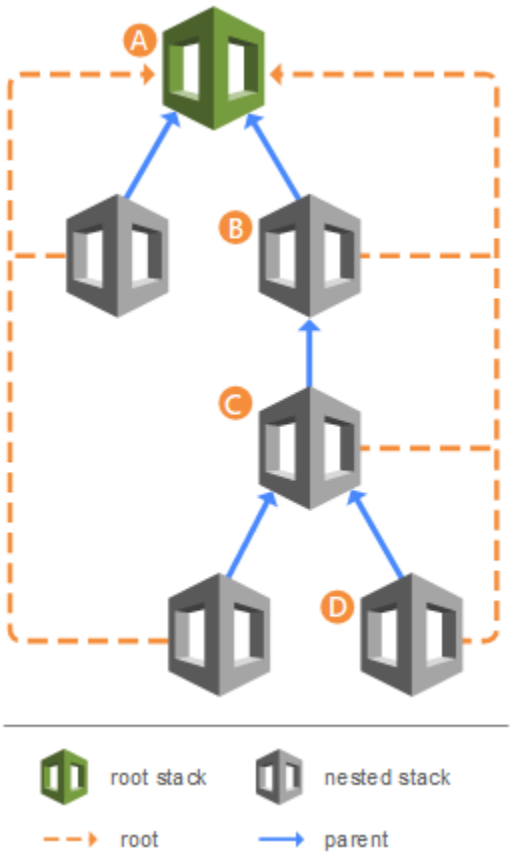
Split a template into reusable pieces using nested stacks

As your infrastructure grows, you might find yourself repeatedly creating identical resource configurations across multiple templates. To avoid this redundancy, you can separate these common configurations into dedicated templates. Then, you can use the AWS::CloudFormation::Stack resource in other templates to reference these dedicated templates, creating nested stacks.

For example, suppose you have a load balancer configuration that you use for most of your stacks. Instead of copying and pasting the same configurations into your templates, you can create a dedicated template for the load balancer. Then, you can reference this template from within other templates that require the same load balancer configuration.

Nested stacks can themselves contain other nested stacks, resulting in a hierarchy of stacks, as shown in the diagram below. The *root stack* is the top-level stack to which all nested stacks ultimately belong. Each nested stack has an immediate parent stack. For the first level of nested stacks, the root stack is also the parent stack.

- Stack A is the root stack for all the other, nested, stacks in the hierarchy.
- For stack B, stack A is both the parent stack, and the root stack.
- For stack D, stack C is the parent stack; while for stack C, stack B is the parent stack.



Topics

- Before and after example of splitting a template
- Example of a nested stack architecture
- Performing stack operations on nested stacks
- Related information
- Express mode and nested stacks

Before and after example of splitting a template

This example demonstrates how you can take a single, large CloudFormation template and reorganize it into a more structured and reusable design using nested templates. Initially, the "Before nesting stacks" template shows all the resources defined in one file. This can become messy and hard to manage as the number of resources grows. The "After nesting stacks" template splits up the resources into smaller, separate templates. Each nested stack handles a specific set of related resources, making the overall structure more organized and easier to maintain.

Before nesting stacks

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  InstanceType:
    Type: String
    Default: t2.micro
    Description: The EC2 instance type

  Environment:
    Type: String
    Default: Production
    Description: The deployment
 environment

Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-1234567890abcdef0
      InstanceType: !Ref InstanceType

  MyS3Bucket:
    Type: AWS::S3::Bucket
```

After nesting stacks

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyFirstNestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazona
ws.com/ amzn-s3-demo-bucket  /first-ne
sted-stack.yaml
      Parameters:
        # Pass parameters to the
 nested stack if needed
        InstanceType: t3.micro

  MySecondNestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazona
ws.com/ amzn-s3-demo-bucket  /second-n
ested-stack.yaml
      Parameters:
        # Pass parameters to the
 nested stack if needed
        Environment: Testing
    DependsOn: MyFirstNestedStack
```

Example of a nested stack architecture

This section demonstrates a nested stack architecture consisting of a top-level stack that references a nested stack. The nested stack deploys a Node.js Lambda function, receives a parameter value from the top-level stack, and returns an output that's exposed through the top-level stack.

Topics

- Step 1: Create a template for the nested stack on your local system
- Step 2: Create a template for the top-level stack on your local system
- Step 3: Package and deploy the templates

Step 1: Create a template for the nested stack on your local system

The following example shows the format of the nested stack template.

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Nested stack template for Lambda function deployment
Parameters:
  MemorySize:
    Type: Number
    Default: 128
    MinValue: 128
    MaxValue: 10240
    Description: Lambda function memory allocation (128-10240 MB)
Resources:
  LambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: !Sub "${AWS::StackName}-Function"
      Runtime: nodejs18.x
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile: |
          exports.handler = async (event) => {
            return {
              statusCode: 200,
              body: JSON.stringify('Hello from Lambda!')
            };
          };
      MemorySize: !Ref MemorySize
  LambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
```

```
Outputs:
  LambdaArn:
    Description: ARN of the created Lambda function
    Value: !GetAtt LambdaFunction.Arn
```

Step 2: Create a template for the top-level stack on your local system

The following example shows the format of the top-level stack template and the AWS::CloudFormation::Stack resource that references the stack you created in the previous step.

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Top-level stack template that deploys a nested stack
Resources:
  NestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: /path_to_template/nested-template.yaml
      Parameters:
        MemorySize: 256
Outputs:
  NestedStackLambdaArn:
    Description: ARN of the Lambda function from nested stack
    Value: !GetAtt NestedStack.Outputs.LambdaArn
```

Step 3: Package and deploy the templates

> **Note**
>
> When working with templates locally, the AWS CLI **package** command can help you prepare templates for deployment. It automatically handles the upload of local artifacts to Amazon S3 (including `TemplateURL`) and generates a new template file with updated references to these S3 locations. For more information, see Upload local artifacts to an S3 bucket with the AWS CLI.

Next, you can use the package command to upload the nested template to an Amazon S3 bucket.

```
aws cloudformation package \
  --s3-bucket amzn-s3-demo-bucket \
  --template /path_to_template/top-level-template.yaml \
```

```
  --output-template-file packaged-template.yaml \
  --output json
```

The command generates a new template at the path specified by `--output-template-file`. It replaces the `TemplateURL` reference with the Amazon S3 location, as shown below.

Resulting template

```
AWSTemplateFormatVersion: 2010-09-09
Description: Top-level stack template that deploys a nested stack
Resources:
  NestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/8b3bb7aa7abfc6e37e2d06b869484bed.template
      Parameters:
        MemorySize: 256
Outputs:
  NestedStackLambdaArn:
    Description: ARN of the Lambda function from nested stack
    Value:
      Fn::GetAtt:
      - NestedStack
      - Outputs.LambdaArn
```

After you run the **package** command, you can deploy the processed template using the deploy command. For nested stacks that contain IAM resources, you must acknowledge IAM capabilities by including the `--capabilities` option.

```
aws cloudformation deploy \
  --template-file packaged-template.yaml \
  --stack-name stack-name \
  --capabilities CAPABILITY_NAMED_IAM
```

Performing stack operations on nested stacks

When working with nested stacks, you must handle them carefully during operations. Certain stack operations, such as stack updates, should be initiated from the root stack rather than performed directly on the nested stacks. When you update a root stack, only nested stacks with template changes will be updated.

Additionally, the presence of the nested stacks can affect operations on the root stack. For example, if one nested stack becomes stuck in `UPDATE_ROLLBACK_IN_PROGRESS` state, the root stack will wait until that nested stack completes its rollback before continuing. Before proceeding with update operations, make sure that you have IAM permissions to cancel a stack update in case it rolls back. For more information, see Control CloudFormation access with AWS Identity and Access Management.

Use the following procedures to find the root stack and nested stacks.

To view the root stack of a nested stack

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, choose the name of the nested stack you want to view the root stack of.

   Nested stacks display **NESTED** above their stack name.
3. On the **Stack info** tab, in the **Overview** section, choose the stack name listed as **Root stack**.

To view the nested stacks that belong to a root stack

1. From the root stack whose nested stacks you want to view, choose the **Resources** tab.
2. In the **Type** column, look for resources of type **AWS::CloudFormation::Stack**.

Related information

- Nesting an existing stack
- Understand update behaviors of stack resources
- Continue rolling back from failed nested stack updates
- Nested stacks rollback failure

Express mode and nested stacks

When you use express mode on a parent stack, the setting propagates to all nested stacks in the hierarchy. You don't need to specify express mode for each nested stack separately. All resources across the hierarchy complete as soon as CloudFormation applies their configuration.

For more information about express mode, see Express mode.

Create wait conditions in a CloudFormation template

This topic explains how to create a wait condition in a template to coordinate the creation of stack resources or track the progress of a configuration process. For example, you can start the creation of another resource after an application configuration is partially complete, or you can send signals during an installation and configuration process to track its progress.

When CloudFormation creates a stack that includes a wait condition:

- It creates a wait condition just like any other resource and sets the wait condition's status to `CREATE_IN_PROGRESS`.
- CloudFormation waits until it receives the requisite number of success signals or the wait condition's timeout period has expired.
- If it receives the requisite number of success signals before the timeout period expires:
  - Wait condition status changes to `CREATE_COMPLETE`
  - Stack creation continues
- If timeout expires or a failure signal is received:
  - Wait condition status changes to `CREATE_FAILED`
  - Stack rolls back

Important

For Amazon EC2 and Auto Scaling resources, we recommend that you use a CreationPolicy attribute instead of wait conditions. Add a CreationPolicy attribute to those resources, and use the cfn-signal helper script to signal when an instance creation process has completed successfully.
For more information, see CreationPolicy attribute.

Note

If you use AWS PrivateLink, resources in the VPC that respond to wait conditions must have access to CloudFormation-specific Amazon Simple Storage Service (Amazon S3) buckets. Resources must send wait condition responses to a presigned Amazon S3 URL. If they can't send responses to Amazon S3, CloudFormation won't receive a response and the stack operation fails. For more information, see Access CloudFormation using an

interface endpoint (AWS PrivateLink) and Controlling access from VPC endpoints with bucket policies.

Topics

- Creating a wait condition in your template
- Wait condition signal syntax
- Accessing signal data

Creating a wait condition in your template

1. Wait condition handle

You start by defining a AWS::CloudFormation::WaitConditionHandle resource in the stack's template. This resource generates the presigned URL needed for sending signals. This allows you to send a signal without having to supply your AWS credentials. For example:

```
Resources:
  MyWaitHandle:
    Type: AWS::CloudFormation::WaitConditionHandle
```

2. Wait condition

Next, you define an AWS::CloudFormation::WaitCondition resource in the stack's template. The basic structure of a `AWS::CloudFormation::WaitCondition` looks like this:

```
  MyWaitCondition:
    Type: AWS::CloudFormation::WaitCondition
    Properties:
      Handle: String
      Timeout: String
      Count: Integer
```

The `AWS::CloudFormation::WaitCondition` resource has two required properties and one optional property.

- `Handle` (required) – A reference to a `WaitConditionHandle` declared in the template.
- `Timeout` (required) – The number of seconds for CloudFormation to wait for the requisite number of signals to be received. `Timeout` is a minimum-bound property, meaning the timeout

occurs no sooner than the time you specify, but can occur shortly thereafter. The maximum time that you can specify is 43200 seconds (12 hours).

- `Count` (optional) – The number of success signals that CloudFormation must receive before it sets that wait condition's status to `CREATE_COMPLETE` and resumes creating the stack. If not specified, the default value is 1.

Typically, you want a wait condition to begin immediately after the creation of a specific resource. You do this by adding the `DependsOn` attribute to a wait condition. When you add a `DependsOn` attribute to a wait condition, CloudFormation creates the resource in the `DependsOn` attribute first, and then creates the wait condition. For more information, see DependsOn attribute.

The following example demonstrates a wait condition that:

- Begins after the successful creation of the `MyEC2Instance` resource
- Uses the `MyWaitHandle` resource as the `WaitConditionHandle`
- Has a timeout of 4500 seconds
- Has the default `Count` of 1 (since no `Count` property is specified)

```
MyWaitCondition:
  Type: AWS::CloudFormation::WaitCondition
  DependsOn: MyEC2Instance
  Properties:
    Handle: !Ref MyWaitHandle
    Timeout: '4500'
```

3. Sending a signal

To signal success or failure to CloudFormation, you typically run some code or script. For example, an application running on an EC2 instance might perform some additional configuration tasks and then send a signal to CloudFormation to indicate completion.

The signal must be sent to the presigned URL generated by the wait condition handle. You use that presigned URL to signal success or failure.

To send a signal

1. To retrieve the presigned URL within the template, use the `Ref` intrinsic function with the logical name of the wait condition handle.

As shown in the following example, your template can declare an Amazon EC2 instance and pass the presigned URL to EC2 instances using the Amazon EC2 `UserData` property. This allows scripts or applications running on those instances to signal success or failure to CloudFormation.

```
 MyEC2Instance:
   Type: AWS::EC2::Instance
   Properties:
   InstanceType: t2.micro  # Example instance type
   ImageId: ami-055e3d4f0bbeb5878  # Change this as needed (Amazon Linux 2023 in
us-west-2)
   UserData:
     Fn::Base64:
       Fn::Join:
         - ""
         - - "SignalURL="
           - { "Ref": "MyWaitHandle" }
```

This results in `UserData` output similar to:

```
SignalURL=https://amzn-s3-demo-bucket.s3.amazonaws.com/....
```

Note: In the AWS Management Console and the command line tools, the presigned URL is displayed as the physical ID of the wait condition handle resource.

2. (Optional) To detect when the stack enters the wait condition, you can use one of the following methods:

   - If you create the stack with notifications enabled, CloudFormation publishes a notification for every stack event to the specified topic. If you or your application subscribe to that topic, you can monitor the notifications for the wait condition handle creation event and retrieve the presigned URL from the notification message.
   - You can also monitor the stack's events using the AWS Management Console, the AWS CLI, or an SDK.

3. To send a signal, you send an HTTP request message using the presigned URL. The request method must be PUT and the `Content-Type` header must be an empty string or omitted. The request message must be a JSON structure of the form specified in Wait condition signal syntax.

You must send the number of success signals specified by the `Count` property in order for CloudFormation to continue stack creation. If you have a `Count` that is greater than 1, the `UniqueId` value for each signal must be unique across all signals sent to a particular wait condition. The `UniqueId` is an arbitrary alphanumerical string.

A `curl` command is one way to send a signal. The following example shows a `curl` command line that signals success to a wait condition.

```
$ curl -T /tmp/a \
  "https://amzn-s3-demo-bucket.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-
west-2%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-
WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVo
%3D"
```

where the file */tmp/a* contains the following JSON structure:

```
{
   "Status" : "SUCCESS",
   "Reason" : "Configuration Complete",
   "UniqueId" : "ID1234",
   "Data" : "Application has completed configuration."
}
```

This example shows a `curl` command line that sends the same success signal except it sends the JSON structure as a parameter on the command line.

```
$ curl -X PUT \
  -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS","Reason" : "Configuration
 Complete","UniqueId" : "ID1234","Data" : "Application has completed
 configuration."}' \
  "https://amzn-s3-demo-bucket.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-
west-2%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-
WaitCondition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitConditionHandle?
Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVo
%3D"
```

Wait condition signal syntax

When you send signals to the URL generated by the wait condition handle, you must use the following JSON format:

```
{
  "Status" : "StatusValue",
  "UniqueId" : "Some UniqueId",
  "Data" : "Some Data",
  "Reason" : "Some Reason"
}
```

Properties

The `Status` field must be one of the following values:

- `SUCCESS`
- `FAILURE`

The `UniqueId` field identifies the signal to CloudFormation. If the `Count` property of the wait condition is greater than 1, the `UniqueId` value must be unique across all signals sent for a particular wait condition; otherwise, CloudFormation will consider the signal a retransmission of the previously sent signal with the same `UniqueId` and ignore it.

The `Data` field can contain any information you want to send back with the signal. You can access the `Data` value by using the Fn::GetAtt function within the template.

The `Reason` field is a string with no other restrictions on its content besides JSON compliance.

Accessing signal data

To access the data sent by valid signals, you can create an output value for the wait condition in your CloudFormation template. For example:

```
Outputs:
  WaitConditionData:
    Description: The data passed back as part of signalling the WaitCondition
    Value: !GetAtt MyWaitCondition.Data
```

You can then view this data using the describe-stacks command, or the **Outputs** tab of the CloudFormation console.

The `Fn::GetAtt` function returns the `UniqueId` and `Data` as a name/value pair within a JSON structure. For example:

```
{"Signal1":"Application has completed configuration."}
```

Create reusable resource configurations that can be included across templates with CloudFormation modules

Modules are a way for you to package resource configurations for inclusion across stack templates, in a transparent, manageable, and repeatable way. Modules can encapsulate common service configurations and best practices as modular, customizable building blocks for you to include in your stack templates. Modules enable you to include resource configurations that incorporate best practices, expert domain knowledge, and accepted guidelines (for areas such as security, compliance, governance, and industry regulations) in your templates, without having to acquire deep knowledge of the intricacies of the resource implementation.

For example, a domain expert in networking could create a module that contains built-in security groups and ingress/egress rules that adhere to security guidelines. You could then include that module in your template to provision secure networking infrastructure in your stack—without having to spend time figuring out how VPCs, subnets, security groups, and gateways work. And because modules are versioned, if security guidelines change over time, the module author can create a new version of the module that incorporates those changes.

Characteristics of using modules in your templates include:

- **Predictability** – A module must adhere to the schema it registers in the CloudFormation registry, so you know what resources it can resolve to once you include it in your template.
- **Reusability** – You can use the same module across multiple templates and accounts.
- **Traceability** – CloudFormation retains knowledge of which resources in a stack were provisioned from a module, enabling you to easily understand the source of resource changes.
- **Manageability** – Once you've registered a module, you can manage it through the CloudFormation registry, including versioning and account and regional availability.

A module can contain:

- One or more resources to be provisioned from the module, along with any associated data, such as outputs or conditions.

- Any module parameters, which enable you to specify custom values whenever the module is used.

For information about developing modules, see Developing modules in the *CloudFormation CLI User Guide*.

Topics

- Considerations when using modules
- Understand module versioning
- Use modules from the CloudFormation private registry
- Use parameters to specify module values
- Reference module resources in CloudFormation templates

Considerations when using modules

- There is no additional charge for using modules. You pay only for the resources those modules resolve to in your stacks.
- CloudFormation quotas, such as the maximum number of resources allowed in a stack, or the maximum size of the template body, apply to the processed template whether the resources included in that template come from modules or not. For more information, see Understand CloudFormation quotas.
- Tags you specify at the stack level are assigned to the individual resources derived from the module.
- Helper scripts specified at the module level don't propagate to the individual resources contained in the module when CloudFormation processes the template.
- Outputs specified in the module are propagated to outputs at the template level.

  Each output will be assigned a logical ID that's a concatenation of the module logical name and the output name as defined in the module. For more information, see Get exported outputs from a deployed CloudFormation stack.
- Parameters specified in the module aren't propagated to parameters at the template level.

  However, you can create template-level parameters that reference module-level parameters. For more information, see Use parameters to specify module values.

Understand module versioning

The CloudFormation registry acts as a repository where you can register and manage modules for use within your AWS account and Region. You can register modules from various sources, including AWS, third-party publishers, and your own custom extensions, within your account and Region. For more information, see Managing extensions with the CloudFormation registry.

Modules can have different versions, so you can specify which version of a module you want to use. This versioning capability is particularly useful when you need to update or modify a module without breaking existing stacks that depend on it.

Keep in mind the following considerations when using multiple versions of a module:

- During stack operations, CloudFormation uses whatever version of the module that's currently registered as the default version in the AWS account and Region in which the stack operation is being performed. This includes modules that are nested in other modules.

  Therefore, be aware that if you have different versions of the same module registered as the default version in different accounts or Regions, using the same template may result in different results.
- During stack operations, CloudFormation uses whatever version of the resource that's currently registered as the default version in the AWS account and Region in which the stack operation is being performed. This includes the resources generated by including modules.
- Changing the default version of a module doesn't initiate any stack update operation. However, the next time you perform a stack operation with any template containing that module, such as a stack update, CloudFormation will use the new default version in the operation.

  The one exception to this is performing a stack update with the **use previous template** option specified, as described below.
- For stack update operations, if you specify the **use previous template** option, CloudFormation uses the previous processed template for the stack update, and doesn't reprocess the module for any changes you might have made to it.
- To guarantee uniform results, if you are including modules in a stack template for use with stack sets, you should ensure that the same version of the module is set as the default version in all the accounts and Regions in which you are planning to deploy your stack instances. This includes for modules that are nested in other modules. For more information, see Managing stacks across accounts and Regions with StackSets.

Requirements for activating third-party public modules

To successfully activate a third-party public module in your account and Region, the following must be true for each third-party public extension (resource or module) included in the module:

- **Extension activation** – The extension must be activated in the account and Region you want to use it in. For more information, see Use third-party public extensions from the CloudFormation registry.
- **Alias registration** – If the extension in the module uses a type name alias, the extension must be registered in your account and Region using the same type name alias. For more information, see Use aliases to refer to extensions.
- **Version compatibility** – The extension version currently activated must be one of the supported major versions of that extension specified in the module.

If you do not have the correct third-party public extensions and extension versions activated, CloudFormation will fail the operation with an error listing the extensions and versions that need to be activated before the module can be successfully activated.

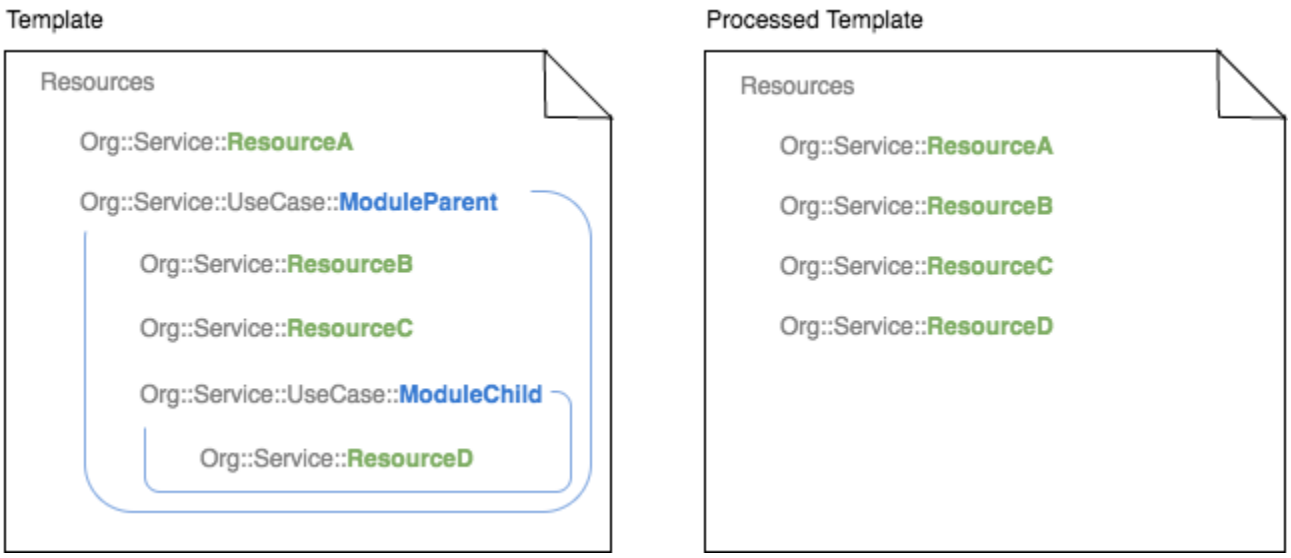
Use modules from the CloudFormation private registry

This topic explains how to use modules in CloudFormation templates. Think of modules as pre-made bundles of resources that you can add to your templates.

To use a module, the steps are as follows:

- **Register the module** – You register modules in the CloudFormation registry as private extensions. Make sure it's registered in the AWS account and Region you're working in. For more information, see CloudFormation registry concepts.
- **Include it in your template** – Add the module to the Resources section of your CloudFormation template, just like you would with other resources. You'll also need to provide any required properties for the module.
- **Create or update the stack** – When you initiate a stack operation, CloudFormation generates a processed template that resolves any included modules into the appropriate resources.
- **Preview changes** – Before making changes, you can use a change set to see what resources will be added or changed. For more information, see Update CloudFormation stacks using change sets.

Consider the following example: you have a template that contains both resources and modules. The template contains one individual resource, `ResourceA`, as well as a module, `ModuleParent`. That module contains two resources, `ResourceB` and `ResourceC`, as well as a nested module, `ModuleChild`. `ModuleChild` contains a single resource, `ResourceD`. If you create a stack from this template, CloudFormation processes the template and resolves the modules to the appropriate resources. The resulting stack has four resources: `ResourceA`, `ResourceB`, `ResourceC`, and `ResourceD`.



CloudFormation keeps track of which resources in a stack were created from modules. You can view this information on the **Events**, **Resources**, and **Drifts** tabs for a given stack, and it's also included in change set previews.

Modules are distinguishable from resources in a template because they adhere to the following four-part naming convention, as opposed to the typical three-part convention used by resources:

```
organization::service::use-case::MODULE
```

Use parameters to specify module values

In CloudFormation, you can use template parameters to customize your stacks by providing input values during stack creation or update. These parameters allow you to change certain aspects of the stack based on your needs. For more information about defining template parameters, see CloudFormation template Parameters syntax.

Similarly, modules can also have parameters. These module parameters allow you to input custom values to the module from the template (or another module) that's using it. The module can then use these custom values to set property values for the resources it contains.

You can also define template parameters that set module properties, so that you can input values that get passed to the module at the time of the stack operation.

If a module contains a nested module that has its own module parameters, you can either:

- Specify the values for the nested module's parameters directly in the parent module.
- Define corresponding module parameters in the parent module that enable the nested module's parameters to be set by the template (or module) in which the parent module is contained.

Using template parameters to specify module parameter values

The following example shows how to define template parameters that pass values to a module.

This template containing `My::S3::SampleBucket::MODULE` defines a template parameter, `BucketName`, that enables the user to specify an S3 bucket name during the stack operation.

```
# Template containing My::S3::SampleBucket::MODULE
Parameters:
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
  MyBucket:
    Type: 'My::S3::SampleBucket::MODULE'
    Properties:
      BucketName: !Ref BucketName
```

Specifying properties on resources in a child module from the parent module

The following example illustrates how to specify parameter values in a module that's nested within another module.

This first module, `My::S3::SampleBucketPrivate::MODULE`, will be the child module. It defines two parameters: `BucketName` and `AccessControl`. The values specified for these parameters are used to specify the `BucketName` and `AccessControl` properties of

the AWS::S3::Bucket resource the module contains. Below is the template fragment for My::S3::SampleBucketPrivate::MODULE.

```
# My::S3::SampleBucketPrivate::MODULE
AWSTemplateFormatVersion: 2010-09-09
Description: A sample S3 Bucket with Versioning and DeletionPolicy.
Parameters:
  BucketName:
    Description: Name for the bucket
    Type: String
  AccessControl:
    Description: AccessControl for the bucket
    Type: String
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref BucketName
      AccessControl: !Ref AccessControl
      DeletionPolicy: Retain
      VersioningConfiguration:
        Status: Enabled
```

Next, the previous module is nested within a parent module, My::S3::SampleBucket::MODULE. The parent module, My::S3::SampleBucket::MODULE, sets the child module parameters in the following ways:

- It sets the AccessControl parameter of My::S3::SampleBucketPrivate::MODULE to Private.
- For BucketName, it defines a module parameter, which will enable the bucket name to be specified in the template (or module) that contains My::S3::SampleBucket::MODULE.

```
# My::S3::SampleBucket::MODULE
AWSTemplateFormatVersion: 2010-09-09
Description: A sample S3 Bucket. With Private AccessControl.
Parameters:
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
```

```
  MyBucket:
    Type: 'My::S3::SampleBucketPrivate::MODULE'
    Properties:
      BucketName: !Ref BucketName
      AccessControl: Private
```

Specifying constraints for module parameters

Module parameters don't support constraint enforcement. To perform constraint checking on a module parameter, create a template parameter with the desired constraints. Then, reference that template parameter in your module parameter. For more information about defining template parameters, see CloudFormation template Parameters syntax.

Reference module resources in CloudFormation templates

In CloudFormation templates, you often need to set properties on one resource based on the name or property of another resource. For more information, see Referencing resources.

To reference a resource contained within a module in your CloudFormation template, you must combine two logical names:

- The logical name you gave to the module itself when you included it in your template.
- The logical name of the specific resource within that module.

You can combine these two logical names with or without using a period (.) between them. For example, if the module's logical name is `MyModule` and the resource's logical name is `MyBucket`, you can refer to that resource as either `MyModule.MyBucket` or `MyModuleMyBucket`.

To find the logical names of resources inside a module, you can consult the module's schema, which is available in the CloudFormation registry or by using the DescribeType operation. The schema lists all the resources and their logical names that are part of the module.

Once you have the full logical name, you can use CloudFormation functions like `GetAtt` and `Ref` to access property values on module resources.

For example, you have a `My::S3::SampleBucket::MODULE` module that contains an `AWS::S3::Bucket` resource with the logical name `S3Bucket`. To refer to the name of this bucket using the `Ref` function, you combine the module's name in your template (`MyBucket`) with the logical name of the resource in the module (`S3Bucket`). The full logical name is either `MyBucket.S3Bucket` or `MyBucketS3Bucket`.

Example template

The following example template creates an S3 bucket using the `My::S3::SampleBucket::MODULE` module. It also create an Amazon SQS queue and set its name to be the same as the bucket name from the module. Additionally, the template outputs the Amazon Resource Name (ARN) of the created S3 bucket.

```
# Template that uses My::S3::SampleBucket::MODULE
Parameters:
  BucketName:
    Description: Name for your sample bucket
    Type: String
Resources:
  MyBucket:
    Type: My::S3::SampleBucket::MODULE
    Properties:
      BucketName: !Ref BucketName
  exampleQueue:
    Type: AWS::SQS::Queue
    Properties:
      QueueName: !Ref MyBucket.S3Bucket
Outputs:
  BucketArn:
    Value: !GetAtt MyBucket.S3Bucket.Arn
```

Managing AWS resources as a single unit with CloudFormation stacks

A stack is a collection of AWS resources that you can manage as a single unit. In other words, you can create, update, and delete a collection of resources by creating, updating, and deleting stacks.

Creating a stack involves deploying a CloudFormation template that specifies the resources and their configurations, which CloudFormation then provisions and configures.

Updating a stack involves making changes to the template or parameters. CloudFormation compares the changes you submit with the current state of your stack and updates only the changed resources. CloudFormation might interrupt resources or replace updated resources, depending on which properties you update. For more information about resource update behaviors, see Understand update behaviors of stack resources.

CloudFormation provides two methods for updating stacks:

- **Change sets** – With change sets, you can preview the changes CloudFormation will make to your stack, and then decide whether to apply those changes. Change sets are JSON-formatted documents that summarize the changes CloudFormation will make to a stack. Use change sets when you want to make sure that CloudFormation doesn't make unintentional changes or when you want to consider several options. For example, you can use a change set to verify that CloudFormation won't replace your stack's database instances during an update.
- **Direct update** – When you directly update a stack, you submit changes and CloudFormation immediately deploys them. Use direct updates when you want to quickly deploy your updates.

Express mode lets you complete stack operations as soon as resource configuration is applied, without waiting for resources to reach a fully stabilized state. Use express mode during iterative development workflows when speed is more important than full resource stabilization. For more information, see Express mode.

Deleting a stack deletes the resources associated with it. A stack, for instance, can include all the resources required to run a web application, such as a web server, a database, and networking rules. If you no longer require that web application, you can simply delete the stack, and all of its related resources are deleted.

> **Note**
>
> You are charged for the stack resources for the time they were operating (even if you deleted the stack right away).

CloudFormation ensures all stack resources are created or deleted as appropriate. Because CloudFormation treats the stack resources as a single unit, they must all be created or deleted successfully for the stack to be created or deleted. If a resource can't be created, CloudFormation rolls the stack back and automatically deletes any resources that were created. If a resource can't be deleted, any remaining resources are retained until the stack can be successfully deleted.

Topics

- Interfaces for managing your stacks
- Agent setup guide
- Create a stack from the CloudFormation console
- View stack information from the CloudFormation console
- Update your stack template
- Understand update behaviors of stack resources
- Update CloudFormation stacks using change sets
- Validate stack deployments
- Update stacks directly
- Cancel a stack update
- Delete a stack from the CloudFormation console
- Monitor stack progress
- Roll back your CloudFormation stack on alarm breach with rollback triggers
- Detect unmanaged configuration changes to stacks and resources with drift detection
- Import AWS resources into a CloudFormation stack
- Stack refactoring
- Resource type support
- Use quick-create links to create CloudFormation stacks
- Deploy AWS CloudFormation stacks faster with express mode
- Understand stack deployment time

- Examples of CloudFormation stack operation commands for the AWS CLI and PowerShell

Interfaces for managing your stacks

You can manage your CloudFormation stacks using the following interfaces:

- **CloudFormation console** – Provides a web interface that you can use to access your stacks. You can access the CloudFormation console by signing into the AWS Management Console, using the search box on the navigation bar to search for **CloudFormation**, and then choosing **CloudFormation** from the search results.
- **AWS Command Line Interface** – Provides commands for a broad set of AWS services, including CloudFormation, and is supported on Windows, Mac, and Linux. For information about the CloudFormation commands, see cloudformation in the *AWS CLI Command Reference*.
- **AWS Tools for PowerShell** – A set of PowerShell modules that are built on the functionality exposed by the SDK for .NET. The Tools for PowerShell enable you to script operations on your AWS resources from the PowerShell command line. You can find the cmdlets for CloudFormation in the AWS Tools for PowerShell Cmdlet Reference.
- **Query API** – Provides low-level API actions that you call using HTTPS requests. If you make API calls in your application, you must write the code to handle low-level details, such as generating the hash to sign the request. For more information about the API actions for CloudFormation, see Actions in the *AWS CloudFormation API Reference*.
- **AWS SDKs** – Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and error handling. For more information, see Tools to Build on AWS.
- **AWS Cloud Development Kit (AWS CDK)** – The AWS CDK is an open-source software development framework that allows you to define AWS infrastructure using familiar programming languages like TypeScript, Python, Java, and .NET. With the CDK, you can model your application resources and then provision them using CloudFormation directly from your integrated development environment (IDE). For more information, see AWS Cloud Development Kit (AWS CDK).

Agent setup guide

The Agent Toolkit for AWS on GitHub provides the `aws-core` plugin and its bundled `aws-cloudformation` skill. The skill helps AI coding agents author and validate CloudFormation

templates, troubleshoot deployments, and generate AWS CDK code. The plugin also configures the AWS MCP Server for documentation search and AWS API access.

Work with your coding agent

For a quick-start reference, you can drop this link directly into your agent's context:

```
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/agent-setup-guide.md
```

Choose the installation method that matches your development environment.

Prerequisites

Ensure the following prerequisites are in place:

- uv installed on your system.
- Node.js (v18+) installed on your system (required for skills installation).
- (Optional) An AWS account with IAM credentials set up on your local machine. Credentials are required for tools that execute AWS API calls (template validation, deployment troubleshooting), but not for documentation search. If you do not have credentials configured, see Configuring the AWS CLI for detailed instructions.

What gets installed

Component	What it provides
`aws-cloudformation` skill	Template authoring with secure defaults, 3-layer validation pipeline (cfn-lint, cfn-guard, change sets), and root-cause diagnosis of failed stacks using CloudFormation events and CloudTrail correlation.
AWS IaC MCP Server	Documentation search (CloudFormation and CDK), template validation, compliance checks, deployment troubleshooting, and CDK best practices -- all running locally on your machine.

Claude Code

Install the `aws-core` plugin

Run the following command to install the plugin from the official Claude Code marketplace:

```
/plugin install aws-core@claude-plugins-official
```

This installs the `aws-cloudformation` skill, other core AWS skills, agent hooks, and the AWS MCP Server configuration in one step.

Install AWS IaC MCP Server

Replace `AWS_PROFILE` with your local AWS profile name.

```
claude mcp add awslabs-aws-iac-mcp \
  -e AWS_PROFILE=default \
  -e AWS_REGION=us-east-1 \
  -e FASTMCP_LOG_LEVEL=ERROR \
  --scope user \
  -- uvx awslabs.aws-iac-mcp-server@latest
```

Codex

Install CloudFormation skill

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

Install AWS IaC MCP Server

Replace `AWS_PROFILE` with your local AWS profile name.

```
codex mcp add awslabs-aws-iac-mcp \
  --env AWS_PROFILE=default \
  --env AWS_REGION=us-east-1 \
  --env FASTMCP_LOG_LEVEL=ERROR \
  -- uvx awslabs.aws-iac-mcp-server@latest
```

Cursor

Install CloudFormation skill

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

Install AWS IaC MCP Server

Add the following to `.cursor/mcp.json` under `"mcpServers"`. Replace `AWS_PROFILE` with your local AWS profile name.

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

Kiro

Install CloudFormation skill in Kiro CLI

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

Install AWS IaC MCP Server

Add the following to `~/.kiro/settings/mcp.json` under `"mcpServers"`. Replace `AWS_PROFILE` with your local AWS profile name.

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
```

```
  },
  "disabled": false
}
```

Install Kiro powers (from Kiro IDE)

Install the following Kiro power that provides specialized CloudFormation context and tools to Kiro agents on-demand:

- **CloudFormation and CDK** -- install via Kiro powers link.

GitHub Copilot

Install CloudFormation skill

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

Install AWS IaC MCP Server

Add the following to `.vscode/mcp.json` under `"mcpServers"`. Replace `AWS_PROFILE` with your local AWS profile name.

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

Windsurf

Install CloudFormation skill

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

Install AWS IaC MCP Server

Add the following to `/.codeium/windsurf/mcp_config.json` under `"mcpServers"`. Replace `AWS_PROFILE` with your local AWS profile name.

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

OpenCode

Install CloudFormation skill

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

Install AWS IaC MCP Server

Add the following to `/.config/opencode/opencode.jsonc` under `"mcpServers"`. Replace `AWS_PROFILE` with your local AWS profile name.

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

For all other agents compatible with agent skills and MCP Server configuration

For any other agent that supports the open-source agent skills format and MCP Server configuration, follow these steps:

Install CloudFormation skill

```
npx skills add aws/agent-toolkit-for-aws/skills --skill aws-cloudformation --yes --
global
```

Install AWS IaC MCP Server

Add the AWS IaC MCP Server to your agent's MCP client configuration file under `"mcpServers"`. Replace `AWS_PROFILE` with your local AWS profile name.

```
"awslabs.aws-iac-mcp": {
  "command": "uvx",
  "args": ["awslabs.aws-iac-mcp-server@latest"],
  "env": {
    "AWS_PROFILE": "default",
    "AWS_REGION": "us-east-1",
    "FASTMCP_LOG_LEVEL": "ERROR"
  }
}
```

What you can do with your agent

Once configured, your agent can help you with:

Task	Example prompt
Author a template	"Create a CloudFormation template for a VPC with public and private subnets"
Validate before deploying	"Validate my template at ./template.yaml and check for security issues"

Task	Example prompt
Troubleshoot a failure	"My stack 'my-app' in us-east-1 failed to deploy. What happened?"
Search documentation	"What properties does AWS::ECS::Service support?"
CDK guidance	"Show me CDK best practices for Lambda functions"
Compliance checks	"Check if my template complies with security best practices"

IAM permissions

The MCP server requires the following AWS permissions for full functionality:

For template validation and compliance: No AWS permissions required (local validation only).

For deployment troubleshooting:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResources",
        "cloudtrail:LookupEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

Related resources

- AWS IaC MCP Server on GitHub
- Agent Toolkit for AWS on GitHub
- CloudFormation User Guide

- AWS CDK Developer Guide
- Introducing the AWS IaC MCP Server (blog)

Create a stack from the CloudFormation console

You can create a stack template and then use it to create a stack using either the CloudFormation console or a command line tool. The console provides a wizard-driven interface with predefined options, which streamlines the stack creation process.

Topics

- Creating a stack
- Configure stack options
- Preview the configuration of your stack

Creating a stack

Follow the steps in this section to deploy your template and create a stack.

Prerequisites

- You must have created a stack template. For more information, see Working with CloudFormation templates.

To create a stack (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region to create the stack in.
3. On the **Stacks** page, choose **Create stack** at top right, and then choose **With new resources (standard)**.

   Alternatively, you can choose the **With existing resources (import resources)** option to import existing AWS resources that are described in your template. For more information on this option, see Import AWS resources into a CloudFormation stack.
4. On the **Create stack** page, do one of the following:

- To use an existing template, for **Prerequisite - Prepare template**, choose **Choose an existing template**. Then, under **Specify template**, choose either **Amazon S3 URL** or **Upload a template file** based on the template's location.
  - If you choose **Amazon S3 URL**, provide a URL to the template file in an S3 bucket.

    If your template includes nested stacks (for example, stacks described in other template documents located in subdirectories), make sure that your S3 bucket contains the necessary files and directories.

    If you have a template from a versioning-enabled bucket, you can specify a specific version of the template by appending `?versionId=`*`version-id`* to the URL. For more information about versioning-enabled buckets, see Working with objects in a versioning-enabled bucket in the *Amazon Simple Storage Service User Guide*.

    The URL must point to a template with a maximum size of 1 MB that is stored in an S3 bucket that you have read permissions to. The URL can be a maximum of 1024 characters long. Some resources may require that the bucket be in the same Region as the stack.
  - If you choose **Upload a template file**, choose **Choose File** to choose a template file from your local computer. The template file size must be 1 MB or less.

    Once you have chosen your template, CloudFormation uploads the file and displays the S3 URL. CloudFormation uploads it to an Amazon S3 bucket in your AWS account. If you already have an S3 bucket that was created by CloudFormation in your AWS account, CloudFormation adds the template to that bucket. If you don't already have an existing CloudFormation-created bucket, it creates a unique bucket for each Region where you upload a template file.

    The following are considerations when using the S3 buckets created by CloudFormation:

    - The buckets are accessible to anyone with Amazon S3 permissions in your AWS account.
    - CloudFormation creates the buckets with server-side encryption enabled by default, thereby encrypting all objects stored in the bucket.

      You can directly manage encryption options for buckets that CloudFormation has created, for example, using the Amazon S3 console at https://console.aws.amazon.com/s3/, or the AWS CLI. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets in the *Amazon Simple Storage Service User Guide*.

    - You can use your own bucket and manage its permissions by manually uploading templates to Amazon S3. When you create or update a stack, specify the Amazon S3 URL of a template file.
  - If you don't have a template ready, choose **Build with template studio**. For more information, see Template studio.
5. Choose **Next** to continue and to validate the template.

   Before continuing, CloudFormation validates your template to catch syntactic and some semantic errors, such as circular dependencies. During validation, CloudFormation first checks if the template is valid JSON. If it isn't, CloudFormation checks if the template is valid YAML. If both checks fail, CloudFormation returns a template validation error.
6. On the **Specify stack details** page, type a stack name in the **Stack name** box.

   The stack name is an identifier that helps you find a particular stack from a list of stacks. A stack name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 128 characters.
7. In the **Parameters** section, specify values for the parameters that were defined in the template.
8. Choose **Next** to continue creating the stack.
9. (Optional) On the **Configure stack options** page, change the default stack options. For more information, see Configure stack options.
10. If your template contains IAM resources, for **Capabilities**, choose **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates.
11. Choose **Next** to continue.
12. On the **Review and create** page, review the details of your stack.

    To change any of the values before launching the stack, choose **Edit** on the section that has the setting that you want to change.
13. (Optional) You can create a change set to preview the configuration of the stack before creating it. On the **Review and create** page, choose **Create change set** and follow the directions. For more information, see Preview the configuration of your stack.
14. Choose **Submit** to launch your stack.

> **Note**
>
> CloudFormation automatically validates your template before provisioning resources. If validation detects errors such as invalid property syntax or resource name conflicts, the stack creation stops before any resources are provisioned. To review validation results, open the stack's **Events** tab and choose the operation ID. On the Operation view page, choose the **Deployment validations** tab.

CloudFormation will then proceed to create all the resources defined in the template.

You can monitor the progress and status of the stack creation on the **Events** tab for your new stack. For more information, see Monitor stack progress.

To create a stack using the command line

You can use one of the following commands:

- create-stack (AWS CLI)
- New-CFNStack (AWS Tools for Windows PowerShell)

For examples of using the command line to create a stack, see Examples of CloudFormation stack operation commands for the AWS CLI and PowerShell.

Configure stack options

On the **Configure stack options** page, you can configure options for your CloudFormation stacks like tags, notification of stack events, or a stack policy.

You can set the following stack options:

Tags

You can add up to 50 tag key pairs to your stack and to any resources that CloudFormation supports tagging for. A tag is a customer-defined key and value that can be assigned to AWS resources for purposes such as cost tracking.

A **Key** consists of any alphanumeric characters or spaces. Tag keys can be up to 127 characters long.

A **Value** consists of any alphanumeric characters or spaces. Tag values can be up to 255 characters long.

After stack creation, adding, updating, or removing stack-level tags will initiate a stack update. All resources that support stack-level tag propagation will be updated accordingly.

Permissions

An existing IAM service role that CloudFormation can assume. Instead of using your account credentials, CloudFormation uses the role's credentials to create your stack. For more information, see CloudFormation service role.

> **Note**
>
> If role manager is enabled in your account, CloudFormation attaches a service role for you, and the service role selection described here is replaced by a **Customize** option. To use a different role, choose **Customize**. For more information, see IAM role creation in the *IAM User Guide*.

Stack failure options

Specifies the provision failure options for all stack deployments and change set operations. For more information, see Choose how to handle failures when provisioning resources.

The **Roll back all stack resources** option will roll back all resources specified in the template when the stack status is `CREATE_FAILED` or `UPDATE_FAILED`.

For create operations, the **Preserve successfully provisioned resources** option preserves the state of successful resources, while failed resources will stay in a failed state until the next update operation is performed.

For update and change set operations, the **Preserve successfully provisioned resources** option preserve the state of successful resources while rolling back failed resources to the last known stable state. Failed resources will be in an `UPDATE_FAILED` state. Resources without a last known stable state will be deleted upon the next stack operation.

You can also set the following advanced options for stack creation:

Express mode

Under **Deployment options**, you can select **Express** for the deployment mode. Express mode completes stack operations as soon as resource configuration is applied, without waiting for resources to reach a fully stabilized state. For more information, see Express mode.

Stack policy

Defines the resources that you want to protect from unintentional updates during a stack update. By default, all resources can be updated during a stack update.

You can enter the stack policy directly as JSON, or upload a JSON file containing the stack policy. For more information, see Prevent updates to stack resources.

Rollback configuration

You can have CloudFormation monitor the state of your stack during stack creation and updating, and roll back that operation if the stack breaches the threshold of any of the alarms you've specified. Specify the CloudWatch alarms that CloudFormation should monitor. If any of the alarms goes to `ALARM` state during the stack operation or the monitoring period, CloudFormation rolls back the entire stack operation. For more information, see Roll back your CloudFormation stack on alarm breach with rollback triggers.

Notification options

You can specify a new or existing Amazon Simple Notification Service topic where notifications about stack events are sent.

If you create an Amazon SNS topic, you must specify a name and an email address where stack event notifications are to be sent.

Stack creation options

The following options are included for stack creation, but aren't available as part of stack updates.

Timeout

Specifies the amount of time, in minutes, that CloudFormation should allot before timing out stack creation operations. If CloudFormation can't create the entire stack in the time allotted, it fails the stack creation due to timeout and rolls back the stack.

By default, there is no timeout for stack creation. However, individual resources may have their own timeouts based on the nature of the service they implement. For example, if an

individual resource in your stack times out, stack creation also times out even if the timeout you specified for stack creation hasn't yet been reached.

Termination protection

Prevents a stack from being accidentally deleted. If a user attempts to delete a stack with termination protection enabled, the deletion fails and the stack, including its status, remains unchanged. For more information, see Protect CloudFormation stacks from being deleted.

Termination protection is **Disabled** by default.

Preview the configuration of your stack

To preview how a CloudFormation stack will be configured before creating the stack, create a change set. This functionality allows you to examine various configurations and make corrections and changes to your stack before executing the change set. For more information on change sets, see Update CloudFormation stacks using change sets.

Creating a change set for a new stack

To create a change set for a new stack, select your stack template and specify the configuration of your stack as you would if you were creating a new stack, then choose to create a new change set rather than a new stack.

To create a change set for a new stack

1. On the **Review and create** page, choose **Create change set**.
2. In the **Create change set** dialog box, enter a name for the change set, and a description if desired. Choose **Create change set**.

   When you create a change set for a new stack, CloudFormation does the following:

   - Launches a new stack with a status of `REVIEW_IN_PROGRESS`.
   - Creates a change set for the new stack that reflects the stack configuration you specified in the previous steps.

   CloudFormation displays the **Change sets** page for the proposed stack. While CloudFormation creates the change set, its status is `CREATE_IN_PROGRESS`, and its execution status is `UNAVAILABLE`. When CloudFormation completes successfully creating the change set, it sets

the change set status to `CREATE_COMPLETE`, and its execution status is `AVAILABLE`. The stack status is updated to `REVIEW_IN_PROGRESS`. At this point, you can execute the change set to complete creating the new stack.

In the **Changes** pane, CloudFormation displays the proposed configuration of your stack.

If CloudFormation fails to create the change set, it sets the changes set status to `CREATE_FAILED`. Fix the error displayed in the **Status reason** field, and then create a new change set. At this stage, you can try various configurations and make corrections and changes to your stack before executing the next change set.

3. To complete creating a new stack based on the change set, choose **Execute**, specify your rollback configuration, and then choose **Execute change set**.

View stack information from the CloudFormation console

After you've created a CloudFormation stack, you can use the AWS Management Console to view its data and resources.

To view information about your CloudFormation stack

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region where the stack is located.
3. On the **Stacks** page, choose the stack name. CloudFormation displays the stack details for the selected stack.

> **Note**
>
> If the stack has been deleted, you can find it by using the **Filter status** option. For more information, see View deleted stacks from the CloudFormation console.

You can view the following stack information:

Stack info

Displays general information about the stack and its configuration, including:

Overview

Displays stack name, stack ID, root stack, and IAM role, along with status information such as stack status, drift status, and termination protection.

Tags

Displays any tags that are associated with the stack.

Stack policy

Describes the stack resources that are protected against stack updates. For you to be able to update these resources, they must be explicitly allowed during a stack update. For more information, see Prevent updates to stack resources.

Rollback configuration

Displays any CloudWatch alarms that you have specified that CloudFormation should monitor during the stack operation or the specified monitoring period. If any of the alarms goes to `ALARM` state during the stack operation or the monitoring period, CloudFormation rolls back the entire stack operation. For more information, see Roll back your CloudFormation stack on alarm breach with rollback triggers.

Notification options

Displays the Amazon Simple Notification Service topic where notifications about stack events are sent, if specified.

Events

Displays the operations that are tracked when you create, update, or delete the stack. For more information, see Monitor stack progress.

All events that are triggered by a given stack operation are assigned the same client request token, which you can use to track operations. Stack operations that are initiated from the console use the token format *Console-StackOperation-ID*, which helps you to easily identify the stack operation. For example, if you create a stack using the console, each resulting stack event would be assigned the same token in the following format: `Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002`.

Resources

Displays the resources that are part of the stack.

Outputs

Displays outputs that were declared in the stack's template. For more information, see Get exported outputs from a deployed CloudFormation stack.

Parameters

Displays the stack's parameters and their values.

For stacks that contain Systems Manager parameters, the **Resolved Value** column displays the values that are used in the stack definition for the Systems Manager parameters. For more information, see Specify existing resources at runtime with CloudFormation-supplied parameter types.

Template

Displays the stack's template.

For stacks that contain macros, choose **View original template** to view the user-submitted template, or **View processed template** to view the template after CloudFormation processes the referenced macros. CloudFormation uses the processed template to create or update your stack.

Change sets

Displays the stack's change sets.

For more information, see View a change set for a CloudFormation stack.

Git sync

Displays the stack's Git sync dashboard.

For more information, see Git sync status dashboard.

Update your stack template

To modify the resources or properties in a CloudFormation stack, you must update the stack's template. Start with the existing template for that stack and make your changes to it. If you have the template stored in a source control system, use a copy of that as your starting point. Otherwise, you can get a copy of the template from CloudFormation.

If you only want to change the parameters or settings of the stack (like a stack's Amazon SNS topic), you can reuse the existing template without getting a copy.

You can update a CloudFormation stack template by using a text editor or Template studio.

To update an existing stack template by using template studio

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, choose the name of the stack to update.
3. Start an update, and then choose **Open in template studio**.
4. Edit the template, and then choose **Save and continue**. Template studio attempts to validate the draft before saving. If validation fails, you can confirm that you want to save anyway. After saving, you return to the stack update.

To update an existing stack template by using the AWS CLI

1. To get the template for the stack you want to update, use the get-template CLI command.
2. Copy the template, paste it into a text file, modify it, and save it. Copy *only* the template. The command encloses the template in quotation marks, but don't copy the quotation marks surrounding the template. The template itself starts with an open brace and ends with the final close brace. Specify changes to the stack's resources in this file.

Keep in mind the following points as you make changes to your template:

- You can't add, modify, or delete a parameter that's used by a resource that doesn't support updates.
- For most resources, changing the logical name of a resource is equivalent to deleting that resource and replacing it with a new one. Any other resources that depend on the renamed resource also need to be updated and might cause them to be replaced. Other resources require you to update a property (not just the logical name) in order to initiate an update.
- Some resources may have constraints about what values you can set for certain properties. For example, changes to the `AllocatedStorage` property for an RDS database instance must be greater than the current value. If your update violates these rules, that part will fail.
- Updating one resource can also affect others that reference it. If you use functions like The `Ref` function or The `Fn::GetAtt` function to set a property based on another resource, CloudFormation will update the referencing resource too when the referenced one changes.

- For information about the effects of updating particular resource properties, see the AWS resource and property types reference. For each property, the effects of an update will be one of the following:
  - *Update requires*: No interruption
  - *Update requires*: Some interruptions
  - *Update requires*: Replacement
- You can verify the JSON or YAML syntax of your template by using the validate-template CLI command or by specifying your template on the console. The console performs validation automatically. However, these methods only verify the syntax of your template and don't validate the property values that you specified for a resource are valid for that resource. For more complex validations and best-practice checks, see Validating templates.

> **Note**
>
> Sometimes CloudFormation won't allow certain changes you try to make, and it will tell you the change isn't permitted. This message might occur asynchronously, however, because resources are created and updated by CloudFormation in a non-deterministic order by default.

Understand update behaviors of stack resources

When you submit an update, CloudFormation updates resources based on differences between what you submit and the stack's current template. Resources that haven't changed run without disruption during the update process. For updated resources, CloudFormation uses one of the following update behaviors:

Update with No Interruption

CloudFormation updates the resource without disrupting operation of that resource and without changing the resource's physical ID. For example, if you update certain properties on an AWS::CloudTrail::Trail resource, CloudFormation updates the trail without disruption.

Updates with Some Interruption

CloudFormation updates the resource with some interruption. For example, if you update certain properties on an AWS::EC2::Instance resource, the instance might have some interruption while CloudFormation and Amazon EC2 reconfigure the instance.

Replacement

CloudFormation recreates the resource during an update, which also generates a new physical ID. CloudFormation usually creates the replacement resource first, changes references from other dependent resources to point to the replacement resource, and then deletes the old resource. For example, if you update the `AvailabilityZone` property of an AWS::EC2::Instance resource type, CloudFormation creates a new resource and replaces the current EC2 Instance resource with the new one.

If you're adding or removing a property that requires a replacement, it will also trigger an update. The update will happen even if the real value of the property doesn't change.

The method CloudFormation uses depends on which property you update for a given resource type. The update behavior for each property is described in the AWS resource and property types reference.

Depending on the update behavior, you can decide when to modify resources to reduce the impact of these changes on your application. In particular, you can plan when resources must be *replaced* during an update. For example, if you update the `Port` property of an AWS::RDS::DBInstance resource type, CloudFormation replaces the DB instance by creating a new DB instance with the updated port setting and deletes the old DB instance. Before the update, you might plan to do the following to prepare for the database replacement:

- Take a snapshot of the current databases.
- Prepare a strategy for how applications that use that DB instance will handle an interruption while the DB instance is being replaced.
- Ensure that the applications that use that DB instance considers the updated port setting and any other updates you have made.
- Use the DB snapshot to restore the databases on the new DB instance.

This example isn't exhaustive, but it's meant to give you an idea of the things to plan for when a resource is replaced during an update.

Note

If the template includes one or more nested stacks, CloudFormation also initiates an update for every nested stack. This is necessary to determine whether the nested stacks

have been modified. CloudFormation updates only those resources in the nested stacks that have changes specified in corresponding templates.

Update CloudFormation stacks using change sets

When you need to update a stack, understanding how your changes will affect running resources before you implement them can help you update stacks with confidence. Change sets allow you to preview how proposed changes to a stack might impact your running resources, including the impact on resource properties and attributes. Whether your changes will delete or replace any critical resources, CloudFormation makes the changes to your stack only when you decide to execute the change set, allowing you to decide whether to proceed with your proposed changes or explore other changes by creating another change set. You can create and manage change sets using the CloudFormation console, AWS CLI, or CloudFormation API.

Topics

- Create a change set for a CloudFormation stack
- View a change set for a CloudFormation stack
- Using drift-aware change sets
- Execute a change set for a CloudFormation stack
- Delete a change set for a CloudFormation stack
- Example change sets for CloudFormation stacks
- Change sets for nested stacks

Important

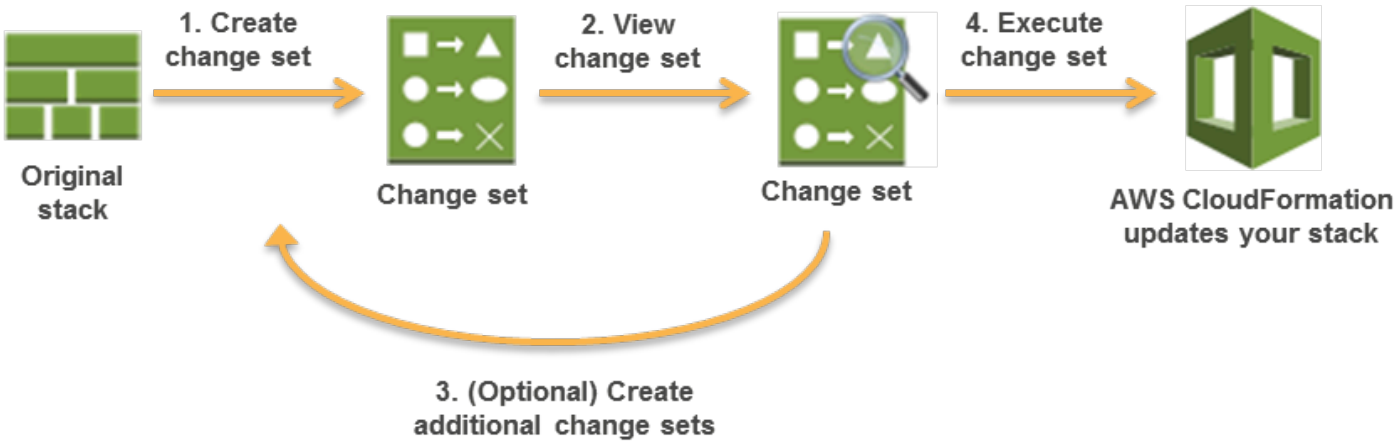
Change sets don't guarantee that CloudFormation will successfully update a stack. However, pre-deployment validation checks for common failure causes during change set creation, including property syntax errors, resource name conflicts, service quota limits, and other constraints. Failures that occur because of runtime conditions (such as custom resource logic or service-specific constraints) might still occur during execution.

Note

You can use express mode with change sets to complete stack operations faster. Specify the `--deployment-config '{"mode": "EXPRESS"}'` parameter when creating a change set. When the change set is executed, CloudFormation applies express mode, completing resource operations as soon as configuration is applied. For more information, see Express mode.

Change set overview

The following diagram summarizes how you use change sets to update a stack:



1. Create a change set by submitting changes for the stack that you want to update. You can submit a modified stack template or modified input parameter values. CloudFormation compares your stack with the changes that you submitted to generate the change set; it doesn't make changes to your stack at this point.
2. View the change set to see which stack settings and resources will change. For example, you can see which resources CloudFormation will add, modify, or delete. Additionally, you can see a before-and-after comparison of the resource properties and attributes, such as tags, that CloudFormation will modify.
3. Optional: If you want to consider other changes before you decide which changes to make, create additional change sets. Creating multiple change sets helps you understand and evaluate how different changes will affect your resources and properties. You can create as many change sets as you need.
4. Execute the change set that contains the changes that you want to apply to your stack. CloudFormation updates your stack with those changes.

> **Note**
>
> After you execute a change, CloudFormation removes all change sets that are associated with the stack because they aren't applicable to the updated stack.

You can also delete change sets to prevent executing a change set that shouldn't be applied.

Create a change set for a CloudFormation stack

To create a change set for a running stack, submit the changes that you want to make by providing a modified template, new input parameter values, or both. CloudFormation generates a change set by comparing your stack with the changes you submitted.

You can either modify a template before creating the change set or during change set creation.

Create a change set (console)

To create a change set

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, choose the running stack you want to create a change set for.
4. In the stack details pane, choose **Stack actions**, and then choose **Create a change set**.
5. On the **Create change set for *`stack-name`*** page, do one of the following to modify input parameter values, specify the location of an updated template, or modify the template:

Task	Action
To modify input parameter values	Choose **Use existing template**, and then choose **Next** to proceed to enter or modify input parameter values.
To specify the location of an updated template	If you've modified the template, choose **Replace existing template**, and then do one of the following: • For a template stored in an Amazon S3 bucket, choose **Amazon S3 URL**. Enter or paste the URL for the template, and then choose **Next**.

Task	Action
	If you have a template in a versioning-enabled bucket, you can specify a specific version of the template by appending ?versionId= *version-id* to the URL. For more information, see Working with objects in a versioning-enabled bucket in the *Amazon Simple Storage Service User Guide*. • For a template stored locally on your computer, choose **Upload a template file**. Choose **Choose File** to navigate to the file and select it, and then choose **Next**.
To modify the template	If you haven't modified the template, choose **Open in template studio**. Edit the template, and then choose **Save and continue** to return to the **Create change set for *stack-name*** page. Choose **Next**.

6. On the **Specify stack details** page, specify a name for the change set and optionally specify a description of the change set to identify its purpose in the **Overview** section. If your template contains parameters, on the **Specify stack details** page, enter or modify applicable input parameter values, and then choose **Next**.

   If you're reusing the stack's template, CloudFormation populates each parameter with the current value in the stack, with the exception of parameters declared with the `NoEcho` attribute. To use existing values for those parameters, select **Use existing value**.

   For more information about using `NoEcho` to mask sensitive information, and using dynamic parameters to manage secrets, see the Do not embed credentials in your templates best practice.

7. On the **Configure stack options** page, update the stack's tags, IAM service role, stack policy, rollback configuration, Amazon SNS notification topic (if applicable), or change sets.

   > **Note**
   >
   > Change sets for nested stacks are **Enabled** by default, which will create change sets for all nested stacks specified in your template. To create a change set for the current stack only, choose **Disabled**. For more information about change sets for nested stacks, see Change sets for nested stacks.

8. If the template includes IAM resources, for **Capabilities**, choose **I acknowledge that CloudFormation might create IAM resources**. IAM resources can modify permissions in your AWS account; review these resources to ensure that you're permitting only the actions that you intend. For more information, see Acknowledging IAM resources in CloudFormation templates.
9. Choose **Next** to continue.
10. On the **Review *stack-name*** page, review the changes for this change set.
11. Choose **Submit**.

    You're redirected to the **Changes** tab of the change set's details page. While CloudFormation generates the change set, the status of the change set is `CREATE_PENDING`. After it has created the change set, CloudFormation sets the status to `CREATE_COMPLETE`. In the **Changes** section, CloudFormation lists all of the changes that it will make to your stack. For more information, see View a change set for a CloudFormation stack.

    Choose **View details** in the **Property-level changes** column to view changes made at the property-level.

    If CloudFormation fails to create the change set (reports `FAILED` status), fix the error displayed in the **Status** field, and then recreate the change set.
12. After confirming the changes look correct, choose **Execute change set**

Create a change set for nested stacks (console)

To create a change set for nested stacks

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, select the running stack you want to create a change set for.
4. In the stack details pane, choose **Stack actions**, and then choose **Create a change set**.
5. On the **Create change set for *stack-name*** page, do one of the following to modify input parameter values, specify the location of an updated template, or modify the template:

Task	Action
To modify input parameter values	Choose **Use existing template**, and then choose **Next** to proceed to enter or modify input parameter values.
To specify the location of an updated template	If you've modified the template, choose **Replace existing template**, and then do one of the following: • For a template stored in an Amazon S3 bucket, choose **Amazon S3 URL**. Enter or paste the URL for the template, and then choose **Next**. If you have a template in a versioning-enabled bucket, you can specify a specific version of the template by appending ?versionId= *version-id* to the URL. For more informati on, see Working with objects in a versioning-enabled bucket in the *Amazon Simple Storage Service User Guide*. • For a template stored locally on your computer, choose **Upload a template file**. Choose **Choose File** to navigate to the file and select it, and then choose **Next**.
To modify the template	If you haven't modified the template, choose **Open in template studio**. Edit the template, and then choose **Save and continue** to return to the **Create change set for *stack-name*** page. Choose **Next**.

6. On the **Specify stack details** page, specify a name for the change set and optionally specify a description of the change set to identify its purpose in the **Overview** section. If your template contains parameters, on the **Specify stack details** page, enter or modify applicable input parameter values, and then choose **Next**.

   If you're reusing the stack's template, CloudFormation populates each parameter with the current value in the stack, with the exception of parameters declared with the NoEcho attribute. To use existing values for those parameters, select **Use existing value**.

For more information about using `NoEcho` to mask sensitive information, as well as using dynamic parameters to manage secrets, see the Do not embed credentials in your templates best practice.

7. On the **Configure stack options** page, update the stack's tags, IAM service role, stack policy, rollback configuration, Amazon SNS notification topic (if applicable), or change sets. For more information, see Configure stack options.

> **Note**
>
> Change sets for nested stacks are **Enabled** by default, which will create change sets for all nested stacks specified in your template. For more information about change sets for nested stacks, see Change sets for nested stacks.

8. If the template includes IAM resources, for **Capabilities**, choose **I acknowledge that CloudFormation might create IAM resources**. IAM resources can modify permissions in your AWS account; review these resources to ensure that you're permitting only the actions that you intend. For more information, see Acknowledging IAM resources in CloudFormation templates.
9. Choose **Next** to continue.
10. On the **Review *`stack-name`*** page, review the changes for this change set.
11. Choose **Submit**.

> **Note**
>
> CloudFormation property-level change sets does not resolve cross-stack references when you create change sets for nested stacks. Change sets can mark resources in a child stack for conditional replacement if they reference the output of a parent stack, and the parent stack has been modified

You're redirected to the **Changes** tab of the change set's details page. While CloudFormation generates the change set, the status of the change set is `CREATE_PENDING`. After it has created the change set, CloudFormation sets the status to `CREATE_COMPLETE`. In the **Changes** section, CloudFormation lists all of the changes that it will make to your stack. For more information, see View a change set for a CloudFormation stack.

If CloudFormation fails to create the change set (reports `FAILED` status), fix the error displayed in the **Status** field, and then recreate the change set.

12. After confirming the changes look correct, choose **Execute change set**

To create a change set (AWS CLI)

- Use the create-change-set command.

  You submit your changes as command options. You can specify new parameter values, a modified template, or both. For example, the following command creates a change set named `SampleChangeSet` for the `MyStack` stack. The change set uses the current stack's template, but with a different value for the `Purpose` parameter:

```
aws cloudformation create-change-set --stack-name MyStack \
    --change-set-name SampleChangeSet --use-previous-template \
    --parameters \
      ParameterKey="InstanceType",UsePreviousValue=true
 ParameterKey="KeyPairName",UsePreviousValue=true
 ParameterKey="Purpose",ParameterValue="production"
```

> **Note**
>
> In addition to property syntax validation and resource name conflict detection, change set creation checks for service quota limits, Recorder conflicts, and ECR repository delete readiness. These checks operate in WARN mode – CloudFormation creates the change set successfully, but warnings indicate potential issues. We recommend that you address these issues before you execute the change set.

Express mode

To create a change set that uses express mode, add the `--deployment-config` parameter:

```
aws cloudformation create-change-set --stack-name MyStack \
    --change-set-name MyExpressChangeSet \
    --use-previous-template \
    --deployment-config '{"mode": "EXPRESS"}'
```

When you execute this change set, CloudFormation uses express mode, completing resource operations as soon as configuration is applied. For more information, see Express mode.

View a change set for a CloudFormation stack

After you create a change set, you can view the proposed changes before executing them. You can use the CloudFormation console, AWS CLI, or CloudFormation API to view change sets. The CloudFormation console provides a summary of the changes and a detailed list of changes in JSON format. The AWS CLI and AWS CloudFormation API return a detailed list of changes in JSON format.

View a change set (console)

To view a change set

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, choose the name of the stack that contains the change set that you want to view.
4. In the navigation pane, choose **Change Sets** to view a list of the stack's change sets.
5. Choose the name of the change set that you want to view.

   The CloudFormation console directs you to the change set's details page, where you can see the time the change set was created, its status, the input used to generate the change set, and a summary of the changes.

   In the **Changes** section, each row represents a resource that CloudFormation will add, modify, or remove.

   - **Add** – CloudFormation creates a resource when you add a resource to the stack's template.
   - **Modify** – CloudFormation modifies a resource when you change the properties of a resource in the stack's template.
   - **Remove** – CloudFormation deletes a resource when you delete a resource from the stack's template.

> **Note**
>
> A modification can cause the resource to be interrupted or replaced (recreated). For more information about resource update behaviors, see Understand update behaviors of stack resources.

To focus on specific changes, use the filter view. For example, filter for a specific resource type, such as `AWS::EC2::Instance`. To filter for a specific resource, specify its logical or physical ID, such as `myWebServer` or `i-123abcd4`.

6. In the **Changes** section, choose **View details** in the **Property-level changes** column to view property value changes made to your resource.
7. The CloudFormation console directs you to the property-level changes page for a resource, where you can see the template configuration of the resource before executing a change set and what the template configuration will look like after executing the change set.

   The **Property-level changes** section table shows the **Path**, **Change type**, **Before value**, and **After value** for impacted properties. In the table, choose the checkbox for each change you want to highlight in the **Before** and **After** views of your template to see what changes will be made at the property-level.

   - **Add** – Added properties are highlighted green.
   - **Modify** – Modified properties are highlighted blue.
   - **Remove** – Removed properties are highlighted red.

View a change set for nested stack (console)

To view a change set for nested stacks (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, choose the name of the stack that contains the change set that you want to view.
4. In the navigation pane, choose **Change sets** to view a list of the stack's change sets.
5. Choose the name of the change set that you want to view.

The CloudFormation console directs you to the change set's details page, where you can see the time the change set was created, its status, the input used to generate the change set, and a summary of the changes.

In the **Changes** section, each row represents a resource that CloudFormation will add, modify, remove, or show the status of dynamic.

- **Add** – CloudFormation creates a resource when you add a resource to the stack's template.
- **Modify** – CloudFormation modifies a resource when you change the properties of a resource in the stack's template.
- **Remove** – CloudFormation deletes a resource when you delete a resource from the stack's template.
- **Dynamic** – CloudFormation can't determine the exact resource change action from the nested stack's template.

> **Note**
>
> A modification can cause the resource to be interrupted or replaced (recreated). For more information about resource update behaviors, see Understand update behaviors of stack resources.

To focus on specific changes, use the filter view. For example, filter for a specific resource type, such as **`AWS::CloudFormation::Stack`**. To filter for a specific resource, specify its logical or physical ID, such as **`DeadLetterQueue`** or **`NestedStack`**.

6. In the **Changes** section, choose **View nested change set** of the nested change set you want to view.

   The CloudFormation console directs you to the nested change set's details page. You can choose **Go to root change set** to view the root change set or, you can choose **View parent change set** to view the parent change set. For more information see, Change sets for nested stacks.

> **Note**
>
> CloudFormation property-level change sets does not resolve cross-stack references when you create change sets for nested stacks. Change sets can mark resources in a child stack for conditional replacement if they reference the output of a parent stack, and the parent stack has been modified

To view a change set (AWS CLI)

1. To get the ID of the change set, run the change-sets command.

   Specify the name of the stack that has the change set that you want to view, as shown in the following example:

   ```
   aws cloudformation list-change-sets --stack-name MyStack
   ```

   CloudFormation returns a list of change sets, similar to the following:

   ```
   {
       "Summaries": [
           {
               "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
   MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
               "Status": "CREATE_COMPLETE",
               "ChangeSetName": "SampleChangeSet",
               "CreationTime": "2020-11-18T20:44:05.889Z",
               "StackName": "MyStack",
               "ChangeSetId": "arn:aws:cloudformation:us-
   east-1:123456789012:changeSet/SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000"
           },
           {
               "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
   MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
               "Status": "CREATE_COMPLETE",
               "ChangeSetName": "SampleChangeSet-conditional",
               "CreationTime": "2020-11-18T21:15:56.398Z",
               "StackName": "MyStack",
   ```

```
            "ChangeSetId": "arn:aws:cloudformation:us-
east-1:123456789012:changeSet/SampleChangeSet-conditional/1a2345b6-0000-00a0-
a123-00abc0abc000"
        },
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
            "Status": "CREATE_COMPLETE",
            "ChangeSetName": "SampleChangeSet-replacement",
            "CreationTime": "2020-11-18T21:03:37.706Z",
            "StackName": "MyStack",
            "ChangeSetId": "arn:aws:cloudformation:us-
east-1:123456789012:changeSet/SampleChangeSet-replacement/1a2345b6-0000-00a0-
a123-00abc0abc000"
        }
    ]
}
```

2. Run the describe-change-set command, specifying the ID of the change set that you want to view. For example:

```
aws cloudformation describe-change-set \
  --change-set-name arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

CloudFormation returns information about the specified change set.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet-direct",
    "Parameters": [
        {
            "ParameterValue": "testing",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "ellioty-useast1",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
```

```
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
                    }
                ],
                "Action": "Modify",
                "Scope": [
                    "Tags"
                ],
                "LogicalResourceId": "MyEC2Instance",
                "Replacement": "False"
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T23:35:25.813Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-direct/9edde307-960d-4e6e-ad66-b09ea2f20255"
}
```

Use `--include-property-values` with **describe-change-set** to list the property-level changes.

The `Changes` key lists changes to resources. If you were to execute this change set, CloudFormation would update the tags of the `i-1abc23d4` EC2 instance. For a description of each field, see the Change data type in the *AWS CloudFormation API Reference*.

For additional examples of change sets, see Example change sets for CloudFormation stacks.

To view property-level changes in a change set (AWS CLI)

- The following command lists the property-level changes related to a change set for a `AWS::EC2::NetworkInterface` resource that will remove the `Ipv4Prefixes` property, modifies the `Description` for the resource, and adds a `Tag`:

```
aws cloudformation describe-change-set --include-property-values \
  --change-set-name arn:aws:cloudformation:us-east-1:123456789012:changeSet/
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841
```

The following is example output.

```
"ChangeSetName": "ExampleChangeSet",
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:803642222207:changeSet/
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841",
    "StackId": "arn:aws:cloudformation:us-east-1:803642222207:stack/ExampleStack/
ab664180-f686-11ee-9e29-12cd92393671",
    "StackName": "ExampleStack",
    "Description": null,
    "Parameters": null,
    "CreationTime": "2024-04-09T18:04:59.935000+00:00",
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": null,
    "NotificationARNs": [],
    "RollbackConfiguration": {
        "RollbackTriggers": []
    },
    "Capabilities": [],
    "Tags": null,
    "ParentChangeSetId": null,
    "IncludeNestedStacks": true,
    "RootChangeSetId": null,
    "OnStackFailure": null,
 {
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
```

```
                "Action": "Modify",
                "LogicalResourceId":
"EC2NetworkInterface00eni067fd35b649a05b7100Tpyls",
                "PhysicalResourceId": "eni-067fd35b649a05b71",
                "ResourceType": "AWS::EC2::NetworkInterface",
                "Replacement": "False",
                "Scope": [
                    "Properties",
                    "Tags"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "Ipv4Prefixes",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Ipv4Prefixes",
                            "BeforeValue": "[]",
                            "AttributeChangeType": "Remove"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    },
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "Description",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Description",
                            "BeforeValue": "",
                            "AfterValue": "Description",
                            "AttributeChangeType": "Modify"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    },
                    {
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Tags/0",
                            "AfterValue": "{\"Key\":\"Test\",\"Value\":\"Test\"}",
                            "AttributeChangeType": "Add"
                        },
```

```
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    }
                ],
                "BeforeContext": "{\"Properties\":{\"Description\":\"\",
\"PrivateIpAddress\":\"172.31.76.2\",\"PrivateIpAddresses\":[{\"PrivateIpAddress
\":\"172.31.76.2\",\"Primary\":\"true\"}],\"SecondaryPrivateIpAddressCount\":\"0\",
\"Ipv6PrefixCount\":\"0\",\"Ipv4Prefixes\":[],\"Ipv4PrefixCount\":\"0\",\"GroupSet
\":[\"sg-05a45689b1059e82d\"],\"Ipv6Prefixes\":[],\"SubnetId\":\"subnet-455e8969\",
\"SourceDestCheck\":\"true\",\"InterfaceType\":\"interface\",\"Tags\":[]},
\"UpdateReplacePolicy\":\"Retain\",\"DeletionPolicy\":\"Retain\"}",
                "AfterContext": "{\"Properties\":{\"Description\":\"Description\",
\"PrivateIpAddress\":\"172.31.76.2\",\"PrivateIpAddresses\":[{\"PrivateIpAddress
\":\"172.31.76.2\",\"Primary\":\"true\"}],\"SecondaryPrivateIpAddressCount
\":\"0\",\"Ipv6PrefixCount\":\"0\",\"Ipv4PrefixCount\":\"0\",\"GroupSet\":
[\"sg-05a45689b1059e82d\"],\"Ipv6Prefixes\":[],\"SubnetId\":\"subnet-455e8969\",
\"SourceDestCheck\":\"true\",\"InterfaceType\":\"interface\",\"Tags\":[{\"Value\":
\"Test\",\"Key\":\"Test\"}]},\"UpdateReplacePolicy\":\"Retain\",\"DeletionPolicy\":
\"Retain\"}"
            }
        }
    ],
    "ChangeSetName": "ExampleChangeSet",
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
ExampleChangeSet/9f7b541b-126b-44f7-998e-932174557841",
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/ExampleStack/
ab664180-f686-11ee-9e29-12cd92393671",
```


Using drift-aware change sets

Drift-aware change sets are enhanced CloudFormation change sets that allow you to identify and manage stack drift in a safe manner. Stacks drift from their expected template configuration when stack resources are changed outside CloudFormation, using the AWS Management Console, AWS CLI, or SDK of underlying services. Drift-aware change sets compare templates with the actual state of your stack resources and bring drifted resources in line with their template definitions. When the template definition of a resource is updated to match its actual state, drift-aware change sets reset the drift status of the resource without modifying the resource.

The benefits of using drift-aware change sets include:

- **Preview overwrites of drift** – Detect whether a deployment will revert critical out-of-band changes made during incident response.
- **Systematic drift reconciliation** – Safely align drifted resources with compliant template definitions, either by overwriting drifted properties with template values or updating the template to match actual resource state.
- **Enhanced rollback capabilities** – If a deployment fails, restore resources to their actual pre-deployment state, not the previous template state.
- **Complete visibility** – See exactly how your deployment will affect actual resource configurations before making changes.

Traditional change sets provide a comparison of your new template with the previous template for a stack, but do not account for stack drift. Drift-aware change sets solve this problem by providing a three-way comparison between:

- **Actual state** – The live configuration of your resources. CloudFormation will read the configuration of resources in your account at the time of change set creation via underlying service APIs.
- **Previous deployment state** – The configuration defined in the template from your last CloudFormation deployment.
- **Desired state** – The configuration defined in your new template.

Drift-aware change sets will update the actual state of all stack resources to match the desired state, even if a resource was not explicitly changed in the template.

Considerations

- **Resource type support** – Drift-aware change sets support a comparison of the desired state with the actual state for hundreds of resource types. For unsupported resource types, drift-aware change sets fall back to comparing the previous deployment state with the desired state. See the Resource type support limitations section for details.
- **Write-only properties** – For properties containing sensitive data (passwords, secrets), drift-aware change sets compare against previous deployment values rather than actual values.
- **AWS-managed properties** – Drift-aware change sets preserve out-of-band changes made by AWS services to managed properties, such as the desired capacity of an Auto Scaling group. See the AWS-managed properties section for details.

- **External tag keys** – Drift-aware change sets do not remove or modify tag keys that were not specified in the template, preventing conflicts with attribute-based access control (ABAC) systems.
- **Replacement of drifted resources** – Drift-aware change sets do not support drift reconciliation for immutable properties.
- **Cross-stack attachments** – Some resource types, such as `AWS::IAM::Policy`, can influence the actual state of other resource types, such as `AWS::IAM::Role`. Drift-aware change sets handle attachment of resources within a stack. If a resource is modified via attachment of a secondary resource from a different stack, drift-aware change sets will detect the modification as drift and can revert the attachment. Popular attachable resources include `AWS::IAM::Policy`, `AWS::IAM::ManagedPolicy`, `AWS::EC2::SecurityGroupIngress`, and `AWS::EC2::SecurityGroupEgress`.

AWS-managed properties

You can configure specific resource properties for active AWS management. For example, you can allow Amazon Relational Database Service (Amazon RDS) to automatically upgrade the minor engine version of an Amazon RDS table. These changes can show up as stack drift in CloudFormation. Drift-aware change sets recognize that drift is expected for AWS-managed properties and leave their actual value untouched if you have not modified the property in their template. Top examples of AWS-managed properties are:

- Enabling the `AutoMinorVersionUpgrade` property of an Amazon RDS table to allow automatic updates of engine version.
- Using the `AWS::ApplicationAutoScaling::ScalableTarget` resource to enable auto-scaling for properties such as the read/write capacity units of an Amazon DynamoDB table and the desired count of an Amazon Elastic Container Service cluster.
- Using the `AWS::AutoScaling::ScalingPolicy` for Auto Scaling groups.

Drift-aware change sets clarify the properties which were identified as AWS-managed. See the AWS CLI section for details.

Using drift-aware change sets (console)

You can create and manage drift-aware change sets through the CloudFormation console using the same workflow as traditional change sets, with additional options for deployment modes.

Creating drift-aware change sets

To create a drift-aware change set (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, select the stack that you want to update.
3. Choose **Stack actions**, and then choose **Create change set for current stack**.
4. On the **Create change set** page, for **Change set type**, choose **Drift aware change set**.
5. Choose **Create change set**.
6. On the change set details page, review the three-way comparison showing actual, previous deployment, and desired state for each affected resource.
7. If you're satisfied with the changes, choose **Execute change set**.

Reviewing drift-aware change sets

When you view a drift-aware change set in the console, you'll see enhanced information compared to traditional change sets:

- **Stack drift status** – Indicates whether the stack has drifted from its last deployment.
- **Property comparison** – Shows a JSON diff between the actual and desired state of an affected resource.
- **Drift indicators** – Clearly marks properties within the JSON diff that have drifted. Click on **View drift** to see the previous deployment value for a property.
- **Value source indicators** – Shows whether before values for a property come from actual state or previous deployment state.

Using drift-aware change sets (AWS CLI)

You can create and manage drift-aware change sets using the AWS CLI by adding the `--deployment-mode REVERT_DRIFT` parameter to the **create-change-set** command.

Creating drift-aware change sets

To create a drift-aware change set

Use the create-change-set command with the `--deployment-mode REVERT_DRIFT` parameter to create a drift-aware change set.

```
aws cloudformation create-change-set \
  --stack-name my-stack \
  --change-set-name my-drift-aware-changeset \
  --template-body file://updated-template.yaml \
  --deployment-mode REVERT_DRIFT \
  --capabilities CAPABILITY_IAM
```

Reviewing drift-aware change sets

To review the details of a drift-aware change set

Use the describe-change-set command to review the details of a drift-aware change set:

```
aws cloudformation describe-change-set \
  --change-set-name my-drift-aware-changeset \
  --stack-name my-stack
```

The response includes enhanced information for drift-aware change sets:

- `StackDriftStatus` – Shows whether the stack has drifted (`DRIFTED`, `IN_SYNC`, `NOT_CHECKED`, or `UNKNOWN`).
- `ResourceDriftStatus` – Shows the drift status for each resource (`DELETED`, `MODIFIED`, `IN_SYNC`, or `NOT_CHECKED`).
- `BeforeValueFrom` – Indicates whether the before value for a property comes from `ACTUAL_STATE` or `PREVIOUS_DEPLOYMENT_STATE`.
- `Drift` – Contains the drift details for a property including `PreviousValue`, `ActualValue`, and `DriftDetectionTimestamp`.
- `ResourceDriftIgnoredProperties` – Contains the properties of a resource for which the change set will not revert drift and the reasons for ignoring drift.

For more information, see DescribeChangeSet in the *AWS CloudFormation API Reference*.

Executing drift-aware change sets

To execute a drift-aware change set

After reviewing the change set, use the execute-change-set command to apply the changes:

```
aws cloudformation execute-change-set \
  --change-set-name my-drift-aware-changeset \
  --stack-name my-stack
```

If the deployment fails, CloudFormation will roll back resources to their actual pre-deployment state and preserve out-of-band changes made since the last deployment.

Resource type support limitations

Drift-aware change sets support a three-way comparison of actual state, previous deployment state, and desired state for all resource types that support drift detection, except the following resource types:

- `AWS::SageMaker::MlflowTrackingServer`
- `AWS::Route53Resolver::FirewallRuleGroup`
- `AWS::MediaLive::Multiplexprogram`
- `AWS::VpcLattice::ResourceConfiguration`
- `AWS::S3::StorageLensGroup`
- `AWS::Bedrock::AgentAlias`
- `AWS::MSK::Cluster`
- `AWS::RDS::DBProxy`
- `AWS::Redshift::ClusterParameterGroup`
- `AWS::QBusiness::Index`
- `AWS::NetworkManager::CoreNetwork`
- `AWS::IAM::OIDCProvider`
- `AWS::Organizations::ResourcePolicy`
- `AWS::SNS::TopicInlinePolicy`
- `AWS::Route53::KeySigningKey`
- `AWS::DataZone::PolicyGrant`
- `AWS::Transfer::Certificate`
- `AWS::SageMaker::ImageVersion`

- AWS::Neptune::DBParameterGroup
- AWS::ODB::CloudVmCluster
- AWS::RolesAnywhere::TrustAnchor
- AWS::Detective::Graph
- AWS::Maester::DocumentType
- AWS::SageMaker::ModelPackageGroup
- AWS::S3Express::BucketPolicy
- AWS::Panorama::PackageVersion
- AWS::S3Tables::TableBucketPolicy

Drift-aware change sets fall back to a comparison of previous deployment state and desired state for resources that do not support the three-way comparison.

Execute a change set for a CloudFormation stack

To make the changes described in a change set to your stack, execute the change set.

Important

After you execute a change set, CloudFormation deletes any additional change sets that are associated with the stack because they're no longer valid for the updated stack. If an update fails, you need to create a new change set.

Note

If the change set was created with express mode (`--deployment-config '{"mode": "EXPRESS"}'`), CloudFormation applies express mode when you execute the change set. Express mode completes resource operations as soon as configuration is applied, without waiting for full stabilization. For more information, see Express mode.

Stack policies and executing a change set

If you execute a change set on a stack that has a stack policy associated with it, CloudFormation enforces the policy when it updates the stack. You can't specify a temporary stack policy that

overrides the existing policy when you execute a change set. To update a protected resource, you must update the stack policy or use the direct update method. For more information, see Update stacks directly.

Execute a change set (console)

To execute a change set

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, choose the name the stack that you want to update.
4. In the navigation pane, choose **Change sets** to view a list of the stack's change sets.
5. Choose the name of the change set that you want to execute.
6. On the change set's details page, choose **Execute change set**.

   CloudFormation immediately starts updating the stack. The CloudFormation console directs you to the **Events** tab, where you can monitor the progress of the stack update. For more information, see Monitor stack progress.

Execute a change set for nested stacks (console)

To execute a change set for nested stacks

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, choose the name the stack that you want to update. You must choose the stack name associated with the root change set.
4. In the navigation pane, choose **Change sets** to view a list of the stack's change sets.
5. Choose the name of the root change set that you want to execute.
6. On the change set's details page, choose **Execute change set**.

> **Note**
>
> CloudFormation executes the changes described in your root change set and nested change sets, if **Enabled** for change sets for nested stacks was selected during the Create a change set for a CloudFormation stack process.

CloudFormation immediately starts updating the stack. The CloudFormation console directs you to the **Events** tab, where you can monitor the progress of the stack update. For more information, see Monitor stack progress.

To execute a change set (AWS CLI)

- Run the execute-change-set command.

  Specify the change set ID of the change set that you want to execute, as shown in the following example:

```
aws cloudformation execute-change-set \
    --change-set-name \
      arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

  The command in the example executes a change set with the ID `arn:aws:cloudformation:us-east-1:123456789012:changeSet/SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000`.

  After you run the command, CloudFormation starts updating the stack. To view the stack's progress, use the describe-stacks command.

Delete a change set for a CloudFormation stack

Deleting a change set removes it from the list of change sets for the stack. Deleting a change set prevents you or another user from accidentally executing a change set that shouldn't be applied. Unless you delete them, CloudFormation retains all change sets until you update the stack.

Delete a change set

To delete a change set (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, select the name of the stack that contains the change set that you want to delete.

4. In the navigation pane, choose **Change sets** to view a list of the stack's change sets.
5. Select the name of the change set that you want to delete.
6. On the change set's details page, choose **Delete change set**.

   CloudFormation immediately starts to delete the change set from the stack's list of change sets, and you're redirected to the **Stacks** page.

Delete a change set for nested stacks (console)

To delete a change set for nested stacks

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, select the stack name associated with the root change set.
4. In the navigation pane, choose **Change sets** to view a list of the stack's change sets.
5. Select the name of the change set that you want to delete.
6. On the change set's details page, choose **Delete**. By choosing **Delete change set**, you will delete the whole hierarchy of nested change sets.

   > **Note**
   >
   > The delete operation for change sets for nested stacks is asynchronous and will show a `DELETE_PENDING` status, followed by a `DELETE_IN_PROGRESS` status. Upon completion of the delete change set operation, the change sets will be removed from the list. Nested stacks in the `REVIEW_IN_PROGRESS` status will also be deleted if they were created during the change set creation.

   CloudFormation immediately starts to delete the change set from the stack's list of change sets.

   > **Note**
   >
   > If you have nested stacks that are stuck in an in-progress operation, see Troubleshooting Errors in Nested stacks rollback failure.

To delete a change set (AWS CLI)

- Run the delete-change-set command, specifying the ID of the change set that you want to delete, as shown in the following example:

```
aws cloudformation delete-change-set \
    --change-set-name \
      arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000
```

Example change sets for CloudFormation stacks

This section provides examples of the change sets that CloudFormation would create for common stack changes. They show how to edit a template directly; modify a single input parameter; plan for resource recreation (replacements), which prevents you from losing data that wasn't backed up or interrupting applications that are running in your stack; and add and remove resources. To illustrate how change sets work, we'll walk through the changes that were submitted and discuss the resulting change set. Because each example builds on and assumes that you understand the previous example, we recommend that you read them in order. For a description of each field in a change set, see the Change data type in the *AWS CloudFormation API Reference*.

You can use the console, AWS CLI, or CloudFormation DescribeChangeSet API operation to view change set details.

We generated each of the following change sets from a stack with the following sample template:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample EC2 instance template for testing change sets.",
  "Parameters" : {
    "Purpose" : {
      "Type" : "String",
      "Default" : "testing",
      "AllowedValues" : ["testing", "production"],
      "Description" : "The purpose of this instance."
    },
    "KeyPairName" : {
      "Type": "AWS::EC2::KeyPair::KeyName",
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the
 instance"
```

```
    },
    "InstanceType" : {
      "Type" : "String",
      "Default" : "t2.micro",
      "AllowedValues" : ["t2.micro", "t2.small", "t2.medium"],
      "Description" : "The EC2 instance type."
    }
  },
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyPairName" },
        "InstanceType" : { "Ref" : "InstanceType" },
        "ImageId" : "ami-8fcee4e5",
        "Tags" : [
          {
            "Key" : "Purpose",
            "Value" : { "Ref" : "Purpose" }
          }
        ]
      }
    }
  }
}
```

Directly editing a template

When you directly modify resources in the stack's template to generate a change set, CloudFormation classifies the change as a direct modification, as opposed to changes initiated by an updated parameter value. The following change set, which added a new tag to the `i-1abc23d4` instance, is an example of a direct modification. All other input values, such as the parameter values and capabilities, are unchanged, so we'll focus on the `Changes` structure.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet-direct",
    "Parameters": [
        {
            "ParameterValue": "testing",
            "ParameterKey": "Purpose"
```

```
        },
        {
            "ParameterValue": "MyKeyName",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
                    }
                ],
                "Action": "Modify",
                "Scope": [
                    "Tags"
                ],
                "LogicalResourceId": "MyEC2Instance",
                "Replacement": "False"
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T23:35:25.813Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-direct/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

In the `Changes` structure, there's only one `ResourceChange` structure. This structure describes information such as the type of resource CloudFormation will change, the action CloudFormation will take, the ID of the resource, the scope of the change, and whether the change requires a replacement (where CloudFormation creates a new resource and then deletes the old one). In the example, the change set indicates that CloudFormation will modify the `Tags` attribute of the `i-1abc23d4` EC2 instance, and doesn't require the instance to be replaced.

In the `Details` structure, CloudFormation labels this change as a direct modification that will never require the instance to be recreated (replaced). You can confidently execute this change, knowing that CloudFormation won't replace the instance.

CloudFormation shows this change as a `Static` evaluation. A static evaluation means that CloudFormation can determine the tag's value before executing the change set. In some cases, CloudFormation can determine a value only after you execute a change set. CloudFormation labels those changes as `Dynamic` evaluations. For example, if you reference an updated resource that's conditionally replaced, CloudFormation can't determine whether the reference to the updated resource will change.

Modifying an input parameter value

When you modify an input parameter value, CloudFormation generates two changes for each resource that uses the updated parameter value. In this example, we want to highlight what those changes look like and which information you should focus on. The following example was generated by changing the value of the `Purpose` input parameter only.

The `Purpose` parameter specifies a tag key value for the EC2 instance. In the example, the parameter value was changed from `testing` to `production`. The new value is shown in the `Parameters` structure.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet",
    "Parameters": [
        {
            "ParameterValue": "production",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "MyKeyName",
```

```
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Dynamic",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
                    },
                    {
                        "CausingEntity": "Purpose",
                        "ChangeSource": "ParameterReference",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
                    }
                ],
                "Action": "Modify",
                "Scope": [
                    "Tags"
                ],
                "LogicalResourceId": "MyEC2Instance",
                "Replacement": "False"
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T23:59:18.447Z",
    "Capabilities": [],
    "StackName": "MyStack",
```

```
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

The Changes structure functions similar to way it does in the Directly editing a template example. There's only one ResourceChange structure; it describes a change to the Tags attribute of the i-1abc23d4 EC2 instance.

However, in the Details structure, the change set shows two changes for the Tags attribute, even though only a single parameter value was changed. Resources that reference a changed parameter value (using the Ref intrinsic function) always result in two changes: one with a Dynamic evaluation and another with a Static evaluation. You can see these types of changes by viewing the following fields:

- For the Static evaluation change, view the ChangeSource field. In this example, the ChangeSource field equals ParameterReference, meaning that this change is a result of an updated parameter reference value. The change set must contain a similar Dynamic evaluation change.
- You can find the matching Dynamic evaluation change by comparing the Target structure for both changes, which will contain the same information. In this example, the Target structures for both changes contain the same values for the Attribute and RequireRecreation fields.

For these types of changes, focus on the static evaluation, which gives you the most detailed information about the change. In this example, the static evaluation shows that the change is the result of a change in a parameter reference value (ParameterReference). The exact parameter that was changed is indicated by the CauseEntity field (the Purpose parameter).

Determining the value of the replacement field

The Replacement field in a ResourceChange structure indicates whether CloudFormation will recreate the resource. Planning for resource recreation (replacements) prevents you from losing data that wasn't backed up or interrupting applications that are running in your stack.

The value in the Replacement field depends on whether a change requires a replacement, indicated by the RequiresRecreation field in a change's Target structure. For example, if the RequiresRecreation field is Never, the Replacement field is False. However, if there are multiple changes on a single resource and each change has a different value for the

RequiresRecreation field, CloudFormation updates the resource using the most intrusive behavior. In other words, if only one of the many changes requires a replacement, CloudFormation must replace the resource and, therefore, sets the Replacement field to True.

The following change set was generated by changing the values for every parameter (Purpose, InstanceType, and KeyPairName), which are all used by the EC2 instance. With these changes, CloudFormation will be required to replace the instance because the Replacement field is equal to True.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
    "ChangeSetName": "SampleChangeSet-multiple",
    "Parameters": [
        {
            "ParameterValue": "production",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "MyNewKeyName",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.small",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-7bef86f8",
                "Details": [
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Dynamic",
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "KeyName",
                            "RequiresRecreation": "Always"
                        },
```

```
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Dynamic",
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "InstanceType",
                            "RequiresRecreation": "Conditionally"
                        }
                    },
                    {
                        "ChangeSource": "DirectModification",
                        "Evaluation": "Dynamic",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
                        }
                    },
                    {
                        "CausingEntity": "KeyPairName",
                        "ChangeSource": "ParameterReference",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "KeyName",
                            "RequiresRecreation": "Always"
                        }
                    },
                    {
                        "CausingEntity": "InstanceType",
                        "ChangeSource": "ParameterReference",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "InstanceType",
                            "RequiresRecreation": "Conditionally"
                        }
                    },
                    {
                        "CausingEntity": "Purpose",
                        "ChangeSource": "ParameterReference",
                        "Evaluation": "Static",
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never"
```

```
                        }
                    }
                ],
                "Action": "Modify",
                "Scope": [
                    "Tags",
                    "Properties"
                ],
                "LogicalResourceId": "MyEC2Instance",
                "Replacement": "True"
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T00:39:35.974Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-multiple/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

Identify the change that requires the resource to be replaced by viewing each change (the static evaluations in the `Details` structure). In this example, each change has a different value for the `RequireRecreation` field, but the change to the `KeyName` property has the most intrusive update behavior, always requiring a recreation. CloudFormation will replace the instance because the key name was changed.

If the key name were unchanged, the change to the `InstanceType` property would have the most intrusive update behavior (`Conditionally`), so the `Replacement` field would be `Conditionally`. To find the conditions in which CloudFormation replaces the instance, view the update behavior for the `InstanceType` property of the AWS::EC2::Instance resource type.

Adding and removing resources

The following example was generated by submitting a modified template that removes the EC2 instance and adds an Auto Scaling group and launch configuration.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/1a2345b6-0000-00a0-a123-00abc0abc000",
    "Status": "CREATE_COMPLETE",
```

```
    "ChangeSetName": "SampleChangeSet-addremove",
    "Parameters": [
        {
            "ParameterValue": "testing",
            "ParameterKey": "Purpose"
        },
        {
            "ParameterValue": "MyKeyName",
            "ParameterKey": "KeyPairName"
        },
        {
            "ParameterValue": "t2.micro",
            "ParameterKey": "InstanceType"
        }
    ],
    "Changes": [
        {
            "ResourceChange": {
                "Action": "Add",
                "ResourceType": "AWS::AutoScaling::AutoScalingGroup",
                "Scope": [],
                "Details": [],
                "LogicalResourceId": "AutoScalingGroup"
            },
            "Type": "Resource"
        },
        {
            "ResourceChange": {
                "Action": "Add",
                "ResourceType": "AWS::AutoScaling::LaunchConfiguration",
                "Scope": [],
                "Details": [],
                "LogicalResourceId": "LaunchConfig"
            },
            "Type": "Resource"
        },
        {
            "ResourceChange": {
                "ResourceType": "AWS::EC2::Instance",
                "PhysicalResourceId": "i-1abc23d4",
                "Details": [],
                "Action": "Remove",
                "Scope": [],
                "LogicalResourceId": "MyEC2Instance"
```

```
            },
            "Type": "Resource"
        }
    ],
    "CreationTime": "2020-11-18T01:44:08.444Z",
    "Capabilities": [],
    "StackName": "MyStack",
    "NotificationARNs": [],
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet-addremove/1a2345b6-0000-00a0-a123-00abc0abc000"
}
```

In the `Changes` structure, there are three `ResourceChange` structures, one for each resource. For each resource, the `Action` field indicates whether CloudFormation adds or removes the resource. The `Scope` and `Details` fields are empty because they apply only to modified resources.

For new resources, CloudFormation can't determine the value of some fields until you execute the change set. For example, CloudFormation doesn't provide the physical IDs of the Auto Scaling group and launch configuration because they don't exist yet. CloudFormation creates the new resources when you execute the change set.

Viewing property-level changes

The following example shows property-level changes to the `Tag` property of an Amazon EC2 instance. The tag `Value` and `Key` will change to `Test`.

```
"ChangeSetName": "SampleChangeSet",
    "ChangeSetId": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
SampleChangeSet/38d91d27-798d-4736-9bf1-fb7c46207807",
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
SampleEc2Template/68edcdc0-f6b6-11ee-966c-126d572cdd11",
    "StackName": "SampleEc2Template",
    "Description": "A sample EC2 instance template for testing change sets.",
    "Parameters": [
        {
            "ParameterKey": "KeyPairName",
            "ParameterValue": "BatchTest"
        },
        {
            "ParameterKey": "Purpose",
            "ParameterValue": "testing"
        },
```

```
        {
            "ParameterKey": "InstanceType",
            "ParameterValue": "t2.micro"
        }
    ],
    "CreationTime": "2024-04-09T21:29:10.759000+00:00",
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": null,
    "NotificationARNs": [],
    "RollbackConfiguration": {
:...skipping...
{
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId": "MyEC2Instance",
                "PhysicalResourceId": "i-0cc7856a36315e62b",
                "ResourceType": "AWS::EC2::Instance",
                "Replacement": "False",
                "Scope": [
                    "Tags"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Tags/0/Value",
                            "BeforeValue": "testing",
                            "AfterValue": "Test",
                            "AttributeChangeType": "Modify"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    },
                    {
                        "Target": {
                            "Attribute": "Tags",
                            "RequiresRecreation": "Never",
                            "Path": "/Properties/Tags/0/Key",
                            "BeforeValue": "Purpose",
```

```
                            "AfterValue": "Test",
                            "AttributeChangeType": "Modify"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    }
                ],
                "BeforeContext": "{\"Properties\":{\"KeyName\":\"BatchTest\",\"ImageId
\":\"ami-8fcee4e5\",\"InstanceType\":\"t2.micro\",\"Tags\":[{\"Value\":\"testing\",
\"Key\":\"Purpose\"}]}}",
                "AfterContext": "{\"Properties\":{\"KeyName\":\"BatchTest\",\"ImageId
\":\"ami-8fcee4e5\",\"InstanceType\":\"t2.micro\",\"Tags\":[{\"Value\":\"Test\",\"Key
\":\"Test\"}]}}"
            }
        }
    ]
```

The `Details` structure shows the values for `Key` and `Value` before the change set is executed, and what they will be after the change set is executed.

Change sets for nested stacks

With *change sets for nested stacks* you can preview the changes to your application and infrastructure resources across the entire nested stack hierarchy and proceed with updates when you've confirmed that all the changes are as intended.

See the following sections for more details about change sets for nested stacks:

Topics

- Overview of change sets and nested stacks
- Working with change sets for nested stacks (console)
- Working with change sets for nested stacks (AWS CLI)

Overview of change sets and nested stacks

Change sets for nested stacks combines the following features together to expand the scope of previewing changes to the entire stack hierarchy:

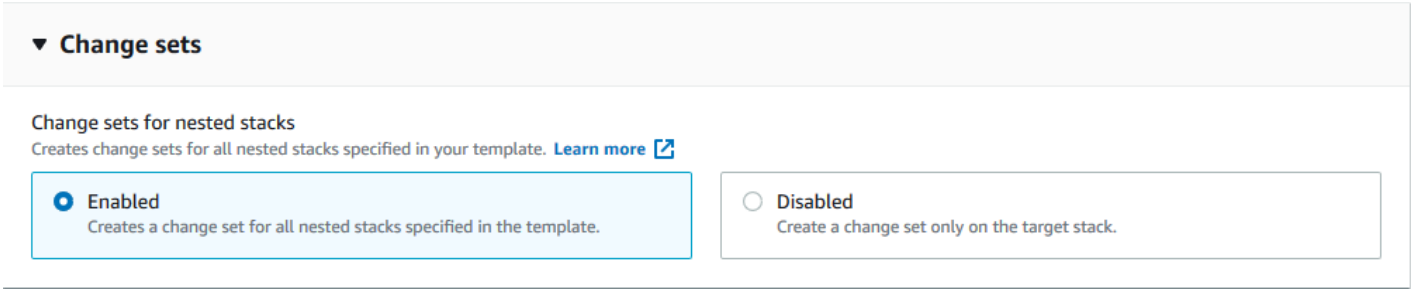
- A *change set* is a CloudFormation capability that offers a preview of how proposed changes to a stack will impact existing or newly created resources. Upon creating a change set,

CloudFormation provides a list of proposed changes by comparing your stack with the changes to the resources you submitted. For more information about change sets, see Update CloudFormation stacks using change sets.

- A *nested stack* is stack created as part of another stack. For example, you might have networking and security related resources in one nested stack and application resources in another. Partitioning application models this way helps with code maintainability and reuse. For more information about nested stacks, see Split a template into reusable pieces using nested stacks.

Working with change sets for nested stacks (console)

- **Create a change set** – Creates a change set by submitting changes from any level of the stack hierarchy. You can submit a modified stack template or modified input parameter values and CloudFormation compares your nested stack with the changes that you submitted to generate a change set. Change sets for nested stacks is enabled by default in the CloudFormation console. For more information, see Create a change set for a CloudFormation stack.



> **Note**
>
> A root change set is the change set associated with the stack from which the whole hierarchy of change sets are created. You must execute or delete change sets for nested stacks from the root change set. For more information, see Performing stack operations on nested stacks.

- **View the change set** – Visualize changes to resources inside nested stacks before executing them. You can view the proposed changes in the **Changes** section of your change set by navigating through the current stack and its nested change sets. For more information, see View a change set for a CloudFormation stack.
- **Execute the change set** – Execute the changes described in the change set that pertain to the current stack and its descendants. The execute operation must be made from the root change set. For more information, see Execute a change set for a CloudFormation stack.

- **Delete the change set** – Removes the change sets from the current stack. Deleting a change set helps to prevent you or another user from accidentally initiating a change set that shouldn't be applied. The delete operation must be executed from the root change set. For more information, see Delete a change set for a CloudFormation stack.

Working with change sets for nested stacks (AWS CLI)

create-change-set

- create-change-set – Change sets for nested stacks isn't enabled by default for the AWS CLI. To create a change set for the entire stack hierarchy, specify the `--include-nested-stacks` option. For more information, see Create a change set for a CloudFormation stack.

The following AWS CLI example creates a change set for the specified root stack.

```
aws cloudformation create-change-set \
    --stack-name my-root-stack \
    --change-set-name my-root-stack-change-set \
    --template-body file://template.yaml \
    --capabilities CAPABILITY_IAM \
    --include-nested-stacks
```

The following is example output.

```
{
    "Id":"arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:Stack/my-root-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204"
}
```

describe-change-set

- describe-change-set – Returns a list of changes that CloudFormation will make if you execute the change set. If the change set specified contains child change sets that belong to nested stacks, then `ChangeSetId` will return information about that change set. For more information, see View a change set for a CloudFormation stack.

The following AWS CLI example describes the change set for the specified root stack.

```
aws cloudformation describe-change-set \
    --change-set-name my-root-stack-change-set \
    --stack-name my-root-stack
```

The following is example output.

```
{
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId": "ChildStack",
                "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-nested-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99205",
                "ResourceType": "AWS::CloudFormation::Stack",
                "Replacement": "False",
                "ChangeSetId": "arn:aws:cloudformation:us-
west-2:123456789012:changeSet/my-nested-stack-change-set/4eca1a01-e285-
xmpl-8026-9a1967bfb4b0",
                "Scope": [
                    "Properties"
                ],
                "Details": [
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "RequiresRecreation": "Never"
                        },
                        "Evaluation": "Dynamic",
                        "ChangeSource": "Automatic"
                    }
                ]
            }
        }
    ],
    "ChangeSetName": "my-root-stack-change-set",
    "ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-
stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-root-stack/
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
    "StackName": "my-root-stack",
    "IncludeNestedStacks": true,
```

```
    "ParentChangeSetId": null,
    "RootChangeSetId": null,
    "Description": null,
    "Parameters": null,
    "CreationTime": "2020-11-18T05:20:56.651Z",
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": null,
    "NotificationARNs": [

    ],
    "RollbackConfiguration": {

    },
    "Capabilities": [
        "CAPABILITY_IAM"
    ],
    "Tags": null
}
```

The following AWS CLI example describes the change set for the specified nested stack.

```
aws cloudformation describe-change-set \
    --change-set-name my-nested-stack-change-set \
    --stack-name my-nested-stack
```

The following is example output.

```
{
    "Changes": [
        {
            "Type": "Resource",
            "ResourceChange": {
                "Action": "Modify",
                "LogicalResourceId": "function",
                "PhysicalResourceId": "my-function",
                "ResourceType": "AWS::Lambda::Function",
                "Replacement": "False",
                "ChangeSetId": null,
                "Scope": [
                    "Properties"
                ],
                "Details": [
```

```
                    {
                        "Target": {
                            "Attribute": "Properties",
                            "Name": "Timeout",
                            "RequiresRecreation": "Never"
                        },
                        "Evaluation": "Static",
                        "ChangeSource": "DirectModification"
                    }
                ]
            }
        }
    ],
    "ChangeSetName": "my-nested-stack-change-set",
    "ChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-nested-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-nested-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99205",
    "ParentChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "RootChangeSetId": "arn:aws:cloudformation:us-west-2:123456789012:changeSet/my-root-stack-change-set/4eca1a01-e285-xmpl-8026-9a1967bfb4b0",
    "IncludeNestedStacks": true,
    "StackName": "my-nested-stack",
    "Description": null,
    "Parameters": null,
    "CreationTime": "2020-11-18T05:20:56.651Z",
    "ExecutionStatus": "UNAVAILABLE",
    "Status": "CREATE_COMPLETE",
    "StatusReason": "Executable from root change set",
    "NotificationARNs": [

    ],
    "RollbackConfiguration": {

    },
    "Capabilities": [
        "CAPABILITY_IAM"
    ],
    "Tags": null
}
```

execute-change-set

- execute-change-set – Creates or updates a stack using the input information that was provided when the specified change set was created. To create a change set for the entire stack hierarchy, you must specify the `--include-nested-stacks` option during the **create-change-set** operation. For more information, see Execute a change set for a CloudFormation stack.

> **Note**
>
> **execute-change-set** must be executed from the root change set and will apply the change set on the whole hierarchy of stacks.

The following AWS CLI example executes a change set for the specified root stack.

```
aws cloudformation execute-change-set \
    --stack-name my-root-stack \
    --change-set-name my-root-stack-change-set
```

delete-change-set

- delete-change-set – Deletes the specified change set. Deleting change sets ensures that no one uses the wrong change set. Deleting change sets is asynchronous for change sets created with the `--include-nested-stacks` option. For more information, see Delete a change set for a CloudFormation stack.

> **Note**
>
> **delete-change-set** must be executed from the root change set and will delete the whole hierarchy of change sets. Nested stacks in the `REVIEW_IN_PROGRESS` status will also be deleted if they were created during the **create-change-set** operation.

The following AWS CLI example deletes the change set for the specified root stack.

```
aws cloudformation delete-change-set \
    --stack-name my-root-stack \
    --change-set-name my-root-stack-change-set
```

Validate stack deployments

With pre-deployment validation, you can identify and resolve potential deployment issues before CloudFormation provisions resources. Pre-deployment validation runs automatically on Create Stack, Update Stack, and Create Change Set operations, catching common errors in seconds.

Topics

- How pre-deployment validation works
- Considerations
- Prerequisites
- Disable pre-deployment validation
- Validate a stack deployment (console)
- Validate a stack deployment (AWS CLI)
- Validation types
- Resource limitations

How pre-deployment validation works

Pre-deployment validation involves these phases:

1. **Initiate a stack operation** – Create a stack, update a stack, or create a change set as you normally would. Pre-deployment validation is enabled by default on all three operations.
2. **Validation execution** – CloudFormation runs multiple validation checks against your template and target environment. CloudFormation supports six types of pre-deployment validation. Property syntax validation and resource name conflict detection run on all supported operations. Four additional validations run during change set creation only: S3 bucket emptiness validation, service quota checks, Recorder conflict detection, and ECR repository delete readiness.
3. **Review validation results** – CloudFormation provides detailed feedback on any issues found, including precise path pinpointing the issue location in template, eliminating manual template debugging.
4. **Resolve issues** – Address identified problems by updating your templates or resolving conflicts before proceeding with deployment.
5. **Execute with confidence** – Proceed with your deployment knowing that common failure scenarios have been validated upfront.

> **Note**
>
> For Create Stack and Update Stack operations, validation runs as part of the operation itself. If validation fails in FAIL mode, the operation stops before any resources are provisioned and the stack status reflects the validation failure. For change sets, the change set status shows `FAILED`.

Considerations

As you use pre-deployment validation, keep the following in mind:

- Pre-deployment validation focuses on common deployment failure scenarios. It doesn't guarantee that your deployment will succeed, but reduces the likelihood of common failures.
- Validation modes behave differently:
  - **FAIL mode** stops the operation before any resources are provisioned when validation detects errors. For CreateStack and UpdateStack, the operation fails and the stack status reflects the validation failure. For CreateChangeSet, the change set status shows FAILED. This applies to property syntax errors and resource naming conflicts.
  - **WARN mode** allows change set creation to succeed despite validation failures, providing warnings that developers can review and address before execution. This applies to constraint violations like S3 bucket emptiness that may be resolvable through manual intervention.
- For change set operations, validation results are tied to the specific change set. If you modify your template, create a new change set to get updated validation results. For CreateStack and UpdateStack operations, use the `describe-events` command with the operation ID to access validation results.
- S3 bucket validation only checks for object presence, not for bucket policies or other constraints that might prevent deletion.
- Pre-deployment validation adds a small amount of latency to Create Stack and Update Stack operations while validations run. This is typically a few seconds. If you need to skip validation for a specific operation, use the `DisableValidation` parameter.

Prerequisites

To use pre-deployment validation, you must have:

- The necessary IAM permissions to create stacks, update stacks, or create change sets, and read resources in your account. The validation checks that run during change set creation require the following additional permissions:
  - **Service quota check** – `cloudwatch:GetMetricData`, `lambda:GetAccountSettings`, `servicequotas:GetServiceQuota`, `ec2:DescribeSecurityGroups`, and `iam:GetAccountSummary`
  - **Recorder conflict check** – `config:ListConfigurationRecorders`
  - **S3 bucket emptiness and ECR repository checks** – `s3:ListBucketV2` and `ecr:ListImages`
- Access to the AWS Regions where your stacks are deployed.
- CloudFormation templates that you want to validate before deployment.

Disable pre-deployment validation

Pre-deployment validation runs by default. To skip validation for a specific operation, use the `DisableValidation` parameter.

Disable validation in the following situations:

- When you have already validated your template through other means (such as `cfn-lint`, `cdk validate`, or CI checks)
- When you need to minimize operation latency for time-sensitive deployments
- When a known false positive is blocking your deployment and you need to proceed

AWS CLI examples:

```
aws cloudformation create-stack --stack-name MyStack --template-body file://
template.yaml --disable-validation
```

```
aws cloudformation update-stack --stack-name MyStack --template-body file://
template.yaml --disable-validation
```

API: Set the `DisableValidation` parameter to `true` on the `CreateStack` or `UpdateStack` API call.

> **Note**
>
> Disabling validation means CloudFormation does not catch common errors until it attempts to provision resources.

Validate a stack deployment (console)

Pre-deployment validation runs automatically on CreateStack and UpdateStack operations. No additional steps are required.

To validate during stack creation or update

1. Create or update your stack as you normally would.
2. If validation detects errors, the operation stops before provisioning begins. The stack status reflects the validation failure.
3. To review validation results, open the stack's **Events** tab and choose the operation ID—or choose the link in the banner or the **Status reason** column. The Operation view page opens directly on the **Deployment validations** tab, which shows validation details, including Logical ID, Resource type, Validation type, Mode, and Status reason.
4. Fix the identified issues in your template and retry the operation.

Validate using a change set (console)

You can also use change sets to validate your template before deployment. Change sets support all six validation types, including WARN-mode validations (S3 bucket emptiness, service quota, Config Recorder conflict, and ECR repository delete readiness) that are only available during change set creation. You can use a change set to validate a new stack before you create it, or to validate an update to an existing stack.

To validate using a change set

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region where your stack is located.
3. On the **Stacks** page, choose the running stack you want to create a change set for.

4. In the stack details pane, choose **Update Stack**, and then choose **Create a change set**.
5. On the **Create change set for *stack-name*** page, upload your updated template or specify the template source.
6. Choose **Next** to proceed through the remaining change set configuration steps.
7. If the template includes IAM resources, for **Capabilities**, choose **I acknowledge that CloudFormation might create IAM resources**. IAM resources can modify permissions in your AWS account; review these resources to ensure that you're permitting only the actions that you intend. For more information, see Acknowledging IAM resources in CloudFormation templates.
8. On the **Review** page, choose **Create change set**.
9. CloudFormation creates the change set and runs validation checks. Review the validation results in the **Deployment validation** tab.
10. If validation passes or you're satisfied with the warnings, choose **Execute Change set** to deploy your changes.
11. If validation fails, fix the issues and create a new change set to re-validate your deployment.

Validate a stack deployment (AWS CLI)

Pre-deployment validation runs automatically on `create-stack` and `update-stack` commands.

Create a stack with validation

Use the following command to create a stack. Pre-deployment validation runs automatically.

```
aws cloudformation create-stack \
  --stack-name MyStack \
  --template-body file://template.yaml
```

Update a stack with validation

Use the following command to update a stack. Pre-deployment validation runs automatically.

```
aws cloudformation update-stack \
  --stack-name MyStack \
  --template-body file://updated-template.yaml
```

View validation results

Use the following command to view validation results for a stack operation.

```
aws cloudformation describe-events \
  --stack-name MyStack
```

Example output of validation errors:

```
{
   "OperationEvents":[
      {
         "EventId":"9b5c9a29-4704-4ad0-8082-afb49418d55b",
         "StackId":"arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/
c3908380-b357-11f0-a97f-0ad08f35df65",
         "OperationId":"f558b823-e1e3-4de3-a222-e6b930ddcad4",
         "OperationType":"CREATE_STACK",
         "OperationStatus":"FAILED",
         "EventType":"STACK_EVENT",
         "Timestamp":"2025-10-27T17:10:02.923Z",
         "StartTime":"2025-10-27T17:09:57.537Z",
         "EndTime":"2025-10-27T17:10:02.923Z"
      },
      {
         "EventId":"2d8c3262-3468-4283-82fb-6e780e9e4f1d",
         "StackId":"arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/
c3908380-b357-11f0-a97f-0ad08f35df65",
         "OperationId":"f558b823-e1e3-4de3-a222-e6b930ddcad4",
         "OperationType":"CREATE_STACK",
         "EventType":"VALIDATION_ERROR",
         "LogicalResourceId":"NotificationBucket",
         "PhysicalResourceId":"",
         "ResourceType":"AWS::S3::Bucket",
         "Timestamp":"2025-10-27T17:10:02.461Z",
         "ValidationFailureMode":"FAIL",
         "ValidationName":"PROPERTY_VALIDATION",
         "ValidationStatus":"FAILED",
         "ValidationStatusReason":"#/NotificationConfiguration/QueueConfigurations/0:
 required key [Event] not found",
         "ValidationPath":"/Resources/NotificationBucket/Properties/
NotificationConfiguration/QueueConfigurations/0"
      }
   ]
}
```

Validate using a change set (AWS CLI)

Change sets also surface WARN-mode validations that are not available on direct stack operations.

To validate using a change set

1. Use the create-change-set command:

```
aws cloudformation create-change-set \
  --stack-name MyStack \
  --change-set-name MyChangeSet \
  --change-set-type "CREATE" \
  --template-body file://updated-template.yaml
```

   The command returns both the change set ARN and the stack ARN.

2. Use the describe-events command with the change set ARN to review validation status and results.

```
aws cloudformation describe-events \
  --change-set-name "arn:aws:cloudformation:us-east-1:123456789012:changeSet/
MyChangeSet/94498df5-1afb-43b1-9869-9f82b2d877ac"
```

3. Address any validation errors by updating your template, then create a new change set.
4. After validation passes, execute the change set:

```
aws cloudformation execute-change-set \
  --change-set-name MyChangeSet \
  --stack-name MyStack
```

Validation types

Pre-deployment validation includes the following types of checks:

- **Property Syntax Validation** – Validates resource properties against AWS resource schemas. It checks for required properties and valid property values and identifies deprecated or unsupported property combinations.
- **Resource Name Conflict Detection** – Checks for naming conflicts with existing AWS resources. It validates that resource names meet AWS naming requirements and identifies potential conflicts before deployment attempts.

- **S3 Bucket Emptiness Validation** – Warns when attempting to delete S3 buckets that contain objects. It provides object counts to help assess deletion impact and helps prevent common S3 deletion failures.
- **Service Quota Validation** – Checks whether creating resources would exceed your AWS service quotas. This validation currently runs during change set creation only and operates in WARN mode. The operation proceeds, and you receive alerts about potential quota issues.
- **Recorder Conflict Detection** – Warns when your template adds rules to an account that doesn't have recording enabled, or defines an Recorder in an account where one is already active. This validation currently runs during change set creation only and operates in WARN mode.
- **ECR Repository Delete Readiness** – Validates that ECR repositories targeted for deletion are empty or have appropriate force-delete settings. This validation currently runs during change set creation only and operates in WARN mode.

The following table summarizes each validation type, its mode, and the operations on which it runs.

Validation type	Mode	Available on
Property syntax validation	FAIL	CreateStack, UpdateStack, CreateChangeSet
Resource name conflict (RAE)	FAIL	CreateStack, UpdateStack, CreateChangeSet
S3 bucket emptiness	WARN	CreateChangeSet
Service Quota	WARN	CreateChangeSet
Recorder conflict	WARN	CreateChangeSet
ECR Repository delete readiness	WARN	CreateChangeSet

Each validation type provides specific error messages and with error location in the template to help you resolve issues quickly.

Resource limitations

The following resource types are not supported for pre-deployment validation:

- AWS::ApiGatewayV2::ApiGatewayManagedOverrides
- AWS::ApiGatewayV2::Stage
- AWS::AppMesh::GatewayRoute
- AWS::AppMesh::Mesh
- AWS::AppMesh::Route
- AWS::AppMesh::VirtualGateway
- AWS::AppMesh::VirtualNode
- AWS::AppMesh::VirtualRouter
- AWS::AppMesh::VirtualService
- AWS::AppStream::Fleet
- AWS::AppStream::Stack
- AWS::AppStream::StackFleetAssociation
- AWS::AppStream::StackUserAssociation
- AWS::AppStream::User
- AWS::AppSync::ApiCache
- AWS::AppSync::ApiKey
- AWS::AppSync::GraphQLSchema
- AWS::AutoScalingPlans::ScalingPlan
- AWS::Budgets::Budget
- AWS::CertificateManager::Certificate
- AWS::Cloud9::EnvironmentEC2
- AWS::CloudFormation::CustomResource
- AWS::CloudFormation::Macro
- AWS::CloudFormation::WaitCondition
- AWS::CloudFormation::WaitConditionHandle
- AWS::CloudFront::StreamingDistribution
- AWS::CloudWatch::AnomalyDetector

- AWS::CloudWatch::InsightRule
- AWS::CodeBuild::Project
- AWS::CodeBuild::ReportGroup
- AWS::CodeBuild::SourceCredential
- AWS::CodeCommit::Repository
- AWS::CodeDeploy::DeploymentGroup
- AWS::CodeStar::GitHubRepository
- AWS::Config::ConfigurationRecorder
- AWS::Config::DeliveryChannel
- AWS::Config::OrganizationConfigRule
- AWS::Config::RemediationConfiguration
- AWS::DAX::Cluster
- AWS::DAX::ParameterGroup
- AWS::DAX::SubnetGroup
- AWS::DirectoryService::MicrosoftAD
- AWS::DLM::LifecyclePolicy
- AWS::DMS::Certificate
- AWS::DMS::Endpoint
- AWS::DMS::EventSubscription
- AWS::DMS::ReplicationInstance
- AWS::DMS::ReplicationSubnetGroup
- AWS::DMS::ReplicationTask
- AWS::DocDB::DBCluster
- AWS::DocDB::DBClusterParameterGroup
- AWS::DocDB::DBInstance
- AWS::DocDB::DBSubnetGroup
- AWS::DocDB::EventSubscription
- AWS::EC2::ClientVpnAuthorizationRule
- AWS::EC2::ClientVpnEndpoint

- AWS::EC2::ClientVpnRoute
- AWS::EC2::ClientVpnTargetNetworkAssociation
- AWS::EC2::NetworkInterfacePermission
- AWS::ElastiCache::CacheCluster
- AWS::ElastiCache::ReplicationGroup
- AWS::ElastiCache::SecurityGroup
- AWS::ElastiCache::SecurityGroupIngress
- AWS::ElasticLoadBalancing::LoadBalancer
- AWS::ElasticLoadBalancingV2::ListenerCertificate
- AWS::Elasticsearch::Domain
- AWS::EMR::Cluster
- AWS::EMR::InstanceFleetConfig
- AWS::EMR::InstanceGroupConfig
- AWS::FSx::FileSystem
- AWS::FSx::Snapshot
- AWS::FSx::StorageVirtualMachine
- AWS::FSx::Volume
- AWS::Glue::Classifier
- AWS::Glue::Connection
- AWS::Glue::CustomEntityType
- AWS::Glue::DataCatalogEncryptionSettings
- AWS::Glue::DataQualityRuleset
- AWS::Glue::DevEndpoint
- AWS::Glue::MLTransform
- AWS::Glue::Partition
- AWS::Glue::SecurityConfiguration
- AWS::Glue::Table
- AWS::Glue::TableOptimizer
- AWS::Glue::Workflow
- AWS::Greengrass::ConnectorDefinition

- AWS::Greengrass::ConnectorDefinitionVersion
- AWS::Greengrass::CoreDefinition
- AWS::Greengrass::CoreDefinitionVersion
- AWS::Greengrass::DeviceDefinition
- AWS::Greengrass::DeviceDefinitionVersion
- AWS::Greengrass::FunctionDefinition
- AWS::Greengrass::FunctionDefinitionVersion
- AWS::Greengrass::Group
- AWS::Greengrass::GroupVersion
- AWS::Greengrass::LoggerDefinition
- AWS::Greengrass::LoggerDefinitionVersion
- AWS::Greengrass::ResourceDefinition
- AWS::Greengrass::ResourceDefinitionVersion
- AWS::Greengrass::SubscriptionDefinition
- AWS::Greengrass::SubscriptionDefinitionVersion
- AWS::IAM::AccessKey
- AWS::IAM::UserToGroupAddition
- AWS::IoT::PolicyPrincipalAttachment
- AWS::IoT::ThingPrincipalAttachment
- AWS::IoTThingsGraph::FlowTemplate
- AWS::KinesisAnalytics::Application
- AWS::KinesisAnalytics::ApplicationOutput
- AWS::KinesisAnalytics::ApplicationReferenceDataSource
- AWS::KinesisAnalyticsV2::ApplicationCloudWatchLoggingOption
- AWS::KinesisAnalyticsV2::ApplicationOutput
- AWS::KinesisAnalyticsV2::ApplicationReferenceDataSource
- AWS::LakeFormation::DataLakeSettings
- AWS::LakeFormation::Permissions
- AWS::LakeFormation::Resource
- AWS::ManagedBlockchain::Member

- AWS::ManagedBlockchain::Node
- AWS::MediaConvert::JobTemplate
- AWS::MediaConvert::Preset
- AWS::MediaConvert::Queue
- AWS::MediaLive::Channel
- AWS::MediaLive::Input
- AWS::MediaLive::InputSecurityGroup
- AWS::MediaStore::Container
- AWS::OpsWorks::App
- AWS::OpsWorks::ElasticLoadBalancerAttachment
- AWS::OpsWorks::Instance
- AWS::OpsWorks::Layer
- AWS::OpsWorks::Stack
- AWS::OpsWorks::UserProfile
- AWS::OpsWorks::Volume
- AWS::Pinpoint::ADMChannel
- AWS::Pinpoint::APNSChannel
- AWS::Pinpoint::APNSSandboxChannel
- AWS::Pinpoint::APNSVoipChannel
- AWS::Pinpoint::APNSVoipSandboxChannel
- AWS::Pinpoint::App
- AWS::Pinpoint::ApplicationSettings
- AWS::Pinpoint::BaiduChannel
- AWS::Pinpoint::Campaign
- AWS::Pinpoint::EmailChannel
- AWS::Pinpoint::EmailTemplate
- AWS::Pinpoint::EventStream
- AWS::Pinpoint::GCMChannel
- AWS::Pinpoint::PushTemplate
- AWS::Pinpoint::Segment

- AWS::Pinpoint::SMSChannel
- AWS::Pinpoint::SmsTemplate
- AWS::Pinpoint::VoiceChannel
- AWS::PinpointEmail::ConfigurationSet
- AWS::PinpointEmail::ConfigurationSetEventDestination
- AWS::PinpointEmail::DedicatedIpPool
- AWS::PinpointEmail::Identity
- AWS::QLDB::Ledger
- AWS::RDS::DBSecurityGroup
- AWS::RDS::DBSecurityGroupIngress
- AWS::Redshift::ClusterSecurityGroup
- AWS::Redshift::ClusterSecurityGroupIngress
- AWS::Route53::RecordSet
- AWS::Route53::RecordSetGroup
- AWS::SageMaker::CodeRepository
- AWS::SageMaker::EndpointConfig
- AWS::SageMaker::Model
- AWS::SageMaker::NotebookInstance
- AWS::SageMaker::NotebookInstanceLifecycleConfig
- AWS::SageMaker::Workteam
- AWS::SDB::Domain
- AWS::ServiceCatalog::AcceptedPortfolioShare
- AWS::ServiceCatalog::LaunchRoleConstraint
- AWS::ServiceCatalog::Portfolio
- AWS::ServiceCatalog::StackSetConstraint
- AWS::ServiceDiscovery::HttpNamespace
- AWS::ServiceDiscovery::Instance
- AWS::ServiceDiscovery::PrivateDnsNamespace
- AWS::ServiceDiscovery::PublicDnsNamespace
- AWS::ServiceDiscovery::Service

- AWS::SES::ReceiptFilter
- AWS::SES::ReceiptRule
- AWS::SES::ReceiptRuleSet
- AWS::SSM::MaintenanceWindow
- AWS::SSM::MaintenanceWindowTarget
- AWS::SSM::MaintenanceWindowTask
- AWS::WAF::ByteMatchSet
- AWS::WAF::IPSet
- AWS::WAF::Rule
- AWS::WAF::SizeConstraintSet
- AWS::WAF::SqlInjectionMatchSet
- AWS::WAF::WebACL
- AWS::WAF::XssMatchSet
- AWS::WAFRegional::ByteMatchSet
- AWS::WAFRegional::GeoMatchSet
- AWS::WAFRegional::IPSet
- AWS::WAFRegional::RateBasedRule
- AWS::WAFRegional::RegexPatternSet
- AWS::WAFRegional::Rule
- AWS::WAFRegional::SizeConstraintSet
- AWS::WAFRegional::SqlInjectionMatchSet
- AWS::WAFRegional::WebACL
- AWS::WAFRegional::WebACLAssociation
- AWS::WAFRegional::XssMatchSet
- AWS::WorkSpaces::Workspace
- AWS::AmazonMQ::ConfigurationAssociation
- AWS::ApiGateway::DomainNameAccessAssociation
- AWS::AppConfig::ExtensionAssociation
- AWS::AppStream::ApplicationEntitlementAssociation
- AWS::AppStream::ApplicationFleetAssociation

- AWS::AppSync::DomainNameApiAssociation
- AWS::AppSync::SourceApiAssociation
- AWS::CleanRooms::ConfiguredTableAssociation
- AWS::CleanRooms::IdNamespaceAssociation
- AWS::CodeGuruReviewer::RepositoryAssociation
- AWS::Cognito::IdentityPoolRoleAttachment
- AWS::Cognito::UserPoolRiskConfigurationAttachment
- AWS::Cognito::UserPoolUICustomizationAttachment
- AWS::Cognito::UserPoolUserToGroupAttachment
- AWS::Connect::IntegrationAssociation
- AWS::Deadline::QueueFleetAssociation
- AWS::Deadline::QueueLimitAssociation
- AWS::EC2::EIPAssociation
- AWS::EC2::EnclaveCertificateIamRoleAssociation
- AWS::EC2::GatewayRouteTableAssociation
- AWS::EC2::IPAMResourceDiscoveryAssociation
- AWS::EC2::IpPoolRouteTableAssociation
- AWS::EC2::LocalGatewayRouteTableVPCAssociation
- AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation
- AWS::EC2::NetworkInterfaceAttachment
- AWS::EC2::RouteServerAssociation
- AWS::EC2::SecurityGroupVpcAssociation
- AWS::EC2::SubnetNetworkAclAssociation
- AWS::EC2::SubnetRouteTableAssociation
- AWS::EC2::TransitGatewayAttachment
- AWS::EC2::TransitGatewayMulticastDomainAssociation
- AWS::EC2::TransitGatewayPeeringAttachment
- AWS::EC2::TransitGatewayRouteTableAssociation
- AWS::EC2::TransitGatewayVpcAttachment
- AWS::EC2::VPCDHCPOptionsAssociation

- AWS::EC2::VPCGatewayAttachment
- AWS::EC2::VolumeAttachment
- AWS::ECS::ClusterCapacityProviderAssociations
- AWS::EKS::PodIdentityAssociation
- AWS::FSx::DataRepositoryAssociation
- AWS::FSx::S3AccessPointAttachment
- AWS::GlobalAccelerator::CrossAccountAttachment
- AWS::LakeFormation::TagAssociation
- AWS::NetworkFirewall::VpcEndpointAssociation
- AWS::NetworkManager::ConnectAttachment
- AWS::NetworkManager::CustomerGatewayAssociation
- AWS::NetworkManager::DirectConnectGatewayAttachment
- AWS::NetworkManager::LinkAssociation
- AWS::NetworkManager::SiteToSiteVpnAttachment
- AWS::NetworkManager::TransitGatewayRouteTableAttachment
- AWS::NetworkManager::VpcAttachment
- AWS::Notifications::ChannelAssociation
- AWS::Notifications::ManagedNotificationAccountContactAssociation
- AWS::Notifications::ManagedNotificationAdditionalChannelAssociation
- AWS::Notifications::OrganizationalUnitAssociation
- AWS::ResourceExplorer2::DefaultViewAssociation
- AWS::Route53Profiles::ProfileAssociation
- AWS::Route53Profiles::ProfileResourceAssociation
- AWS::Route53Resolver::FirewallRuleGroupAssociation
- AWS::Route53Resolver::ResolverQueryLoggingConfigAssociation
- AWS::Route53Resolver::ResolverRuleAssociation
- AWS::SSM::Association
- AWS::SecretsManager::SecretTargetAttachment
- AWS::SecurityHub::PolicyAssociation
- AWS::ServiceCatalog::PortfolioPrincipalAssociation

- AWS::ServiceCatalog::PortfolioProductAssociation
- AWS::ServiceCatalog::ServiceActionAssociation
- AWS::ServiceCatalog::TagOptionAssociation
- AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation
- AWS::ServiceCatalogAppRegistry::ResourceAssociation
- AWS::VpcLattice::ServiceNetworkResourceAssociation
- AWS::VpcLattice::ServiceNetworkServiceAssociation
- AWS::VpcLattice::ServiceNetworkVpcAssociation
- AWS::WAFv2::WebACLAssociation
- AWS::Wisdom::AssistantAssociation
- AWS::WorkspacesInstances::VolumeAssociation
- AWS::IAM::Policy
- AWS::SNS::TopicPolicy
- AWS::SQS::QueuePolicy
- AWS::EC2::NetworkAclEntry
- AWS::EC2::VPNGatewayRoutePropagation
- AWS::CloudFormation::Stack
- AWS::CloudWatch::MetricStream
- AWS::WorkSpaces::ConnectionAlias
- AWS::IoT::ProvisioningTemplate
- AWS::MediaPackage::Channel
- AWS::CloudFront::OriginRequestPolicy
- AWS::Route53Resolver::ResolverQueryLoggingConfig
- AWS::NetworkManager::TransitGatewayRegistration
- AWS::ImageBuilder::Image
- AWS::Config::ConformancePack
- AWS::S3::AccessPoint
- AWS::CodeStarConnections::Connection
- AWS::CloudFront::CachePolicy
- AWS::FMS::NotificationChannel

- AWS::ImageBuilder::InfrastructureConfiguration
- AWS::Detective::Graph
- AWS::EC2::CarrierGateway
- AWS::CloudWatch::CompositeAlarm
- AWS::CodeArtifact::Repository
- AWS::GroundStation::DataflowEndpointGroup
- AWS::ElasticLoadBalancingV2::Listener
- AWS::ImageBuilder::ImageRecipe
- AWS::NetworkManager::Device
- AWS::Kendra::DataSource
- AWS::Timestream::Database
- AWS::CodeGuruProfiler::ProfilingGroup
- AWS::Lambda::EventSourceMapping
- AWS::ECR::Repository
- AWS::WAFv2::IPSet
- AWS::GameLift::Alias
- AWS::IoTSiteWise::Asset
- AWS::OpsWorksCM::Server
- AWS::IoT::Authorizer
- AWS::WAFv2::RuleGroup
- AWS::NetworkManager::Site
- AWS::ResourceGroups::Group
- AWS::MediaPackage::PackagingConfiguration
- AWS::ImageBuilder::ImagePipeline
- AWS::ECS::TaskDefinition
- AWS::Macie::CustomDataIdentifier
- AWS::MediaPackage::OriginEndpoint
- AWS::Logs::LogGroup
- AWS::CodeArtifact::Domain
- AWS::Kendra::Faq

- AWS::ECS::TaskSet
- AWS::WAFv2::RegexPatternSet
- AWS::ECS::Cluster
- AWS::SSO::Assignment
- AWS::GlobalAccelerator::Listener
- AWS::ServiceCatalog::CloudFormationProvisionedProduct
- AWS::RDS::DBProxy
- AWS::EC2::FlowLog
- AWS::ImageBuilder::Component
- AWS::CloudFront::RealtimeLogConfig
- AWS::NetworkManager::GlobalNetwork
- AWS::RDS::DBProxyTargetGroup
- AWS::WAFv2::WebACL
- AWS::IVS::StreamKey
- AWS::IVS::PlaybackKeyPair
- AWS::Macie::Session
- AWS::Route53::HealthCheck
- AWS::Synthetics::Canary
- AWS::Lambda::CodeSigningConfig
- AWS::EFS::AccessPoint
- AWS::Timestream::Table
- AWS::MediaPackage::PackagingGroup
- AWS::ECS::PrimaryTaskSet
- AWS::Config::ConfigurationAggregator
- AWS::GroundStation::Config
- AWS::IoTSiteWise::AssetModel
- AWS::SES::ConfigurationSet
- AWS::ImageBuilder::DistributionConfiguration
- AWS::Config::OrganizationConformancePack
- AWS::EC2::LocalGatewayRoute

- AWS::KMS::Key
- AWS::Detective::MemberInvitation
- AWS::EKS::FargateProfile
- AWS::MediaPackage::Asset
- AWS::GlobalAccelerator::EndpointGroup
- AWS::Macie::FindingsFilter
- AWS::IoT::Certificate
- AWS::SageMaker::MonitoringSchedule
- AWS::IVS::Channel
- AWS::Kendra::Index
- AWS::EventSchemas::RegistryPolicy
- AWS::KinesisFirehose::DeliveryStream
- AWS::GlobalAccelerator::Accelerator
- AWS::EC2::PrefixList
- AWS::GameLift::GameServerGroup
- AWS::NetworkManager::Link
- AWS::EFS::FileSystem
- AWS::Route53::HostedZone
- AWS::GroundStation::MissionProfile
- AWS::KMS::Alias
- AWS::FMS::Policy
- AWS::SSO::PermissionSet
- AWS::StepFunctions::StateMachine
- AWS::QLDB::Stream
- AWS::IoTSiteWise::Gateway
- AWS::ECS::Service
- AWS::ECS::CapacityProvider
- AWS::EC2::SecurityGroup
- AWS::EC2::SecurityGroupIngress
- AWS::EC2::SecurityGroupEgress

- AWS::EC2::EC2Fleet
- AWS::IAM::Group
- AWS::IAM::Role
- AWS::IAM::User
- AWS::ApiGateway::GatewayResponse
- AWS::S3::BucketPolicy
- AWS::SNS::Topic
- AWS::SNS::Subscription
- AWS::RDS::DBInstance
- AWS::RDS::DBParameterGroup
- AWS::RDS::DBCluster
- AWS::RDS::DBClusterParameterGroup
- AWS::RDS::DBSubnetGroup
- AWS::RDS::EventSubscription
- AWS::RDS::GlobalCluster
- AWS::RDS::OptionGroup
- AWS::Neptune::DBInstance
- AWS::Neptune::DBParameterGroup
- AWS::Neptune::DBCluster
- AWS::Neptune::DBClusterParameterGroup
- AWS::Neptune::DBSubnetGroup
- AWS::Redshift::Cluster
- AWS::Redshift::ClusterParameterGroup
- AWS::Redshift::ClusterSubnetGroup
- AWS::Redshift::EndpointAccess
- AWS::Redshift::EndpointAuthorization
- AWS::Redshift::EventSubscription
- AWS::Redshift::ScheduledAction
- AWS::ElastiCache::SubnetGroup
- AWS::ApiGateway::BasePathMapping

- AWS::ApiGatewayV2::DomainName
- AWS::ApplicationAutoScaling::ScalableTarget
- AWS::Connect::ApprovedOrigin
- AWS::EKS::Cluster
- AWS::IAM::InstanceProfile
- AWS::Lambda::EventInvokeConfig

Update stacks directly

When you want to quickly deploy updates to your stack, perform a direct update. With a direct update, you submit a template or input parameters that specify updates to the resources in the stack, and CloudFormation immediately deploys them. If you want to use a template to make your updates, you can modify the current template and store it locally or in an Amazon S3 bucket.

For resource properties that don't support updates, you must keep the current values. To preview the changes that CloudFormation will make to your stack before you update it, use change sets. For more information, see Update CloudFormation stacks using change sets.

When updating a stack, CloudFormation might interrupt resources or replace updated resources, depending on which properties you update. For more information about resource update behaviors, see Understand update behaviors of stack resources.

Pre-deployment validation

When you perform a direct stack update, CloudFormation automatically validates your template against common failure scenarios before modifying any resources. If validation detects errors, the update stops and no resources are changed.

To view validation results in the console, open the stack's **Events** tab and choose the operation ID. Then choose the **Deployment validations** tab. To view results with the AWS CLI, use the describe-events command. To skip validation, use the `--disable-validation` flag in the AWS CLI or set `DisableValidation` to `true` in the API call.

To update a stack (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.

2. On the navigation bar at the top of the screen, choose your AWS Region.
3. On the **Stacks** page, select the running stack that you want to update.
4. In the stack details pane, choose **Update**.
5. If you *haven't* modified the stack template, select **Use existing template**, and then choose **Next**.

   If you have modified the template, select **Replace existing template** and specify the location of the updated template in the **Specify template** section:

   - For a template stored locally on your computer, select **Upload a template file**. Choose **Choose file** to navigate to the file and select it, and then choose **Next**.

     > **Note**
     >
     > If you upload a local template file, CloudFormation uploads it to an Amazon Simple Storage Service (Amazon S3) bucket in your AWS account. If you don't already have an S3 bucket that was created by CloudFormation, it creates a unique bucket for each Region in which you upload a template file. If you already have an S3 bucket that was created by CloudFormation in your AWS account, CloudFormation adds the template to that bucket.
     > Considerations to keep in mind about S3 buckets created by CloudFormation
     >
     > - The buckets are accessible to anyone with Amazon S3 permissions in your AWS account.
     > - CloudFormation creates the buckets with server-side encryption enabled by default, thereby encrypting all objects stored in the bucket.
     >
     >   You can directly manage encryption options for buckets that CloudFormation has created; for example, using the Amazon S3 console at https://console.aws.amazon.com/s3/, or the AWS CLI. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets in the *Amazon Simple Storage Service User Guide*.
     > - You can use your own bucket and manage its permissions by manually uploading templates to Amazon S3. When you create or update a stack, specify the Amazon S3 URL of a template file.

   - For a template stored in an Amazon S3 bucket, choose **Amazon S3 URL**. Enter or paste the URL for the template, and then choose **Next**.

If you have a template in a versioning-enabled bucket, you can specify a specific version of the template by appending `?versionId=`*`version-id`* to the URL. For more information, see Working with objects in a versioning-enabled bucket in the *Amazon Simple Storage Service User Guide*.

If any syntax issues are detected, the console provides error messages that help you correct the template.

6. If your template contains parameters, on the **Specify stack details** page you can enter or modify the parameter values, and then choose **Next**.

   CloudFormation populates each parameter with the value that's currently set in the stack with the exception of parameters declared with the `NoEcho` attribute; however, you can still use current values by checking **Use existing value**.

   For more information about using `NoEcho` to mask sensitive information, in addition to using dynamic parameters to manage secrets, see the Do not embed credentials in your templates best practice.

7. On the **Configure stack options** page, you can update the tags and permissions applied to the stack, and modify advanced options such as stack policy, rollback configuration, or update the Amazon SNS notification topic. For more information about these options, see Configure stack options.

8. If your template contains IAM resources, for **Capabilities**, choose **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates.

9. Choose **Next** to continue.

10. Review the stack information and the changes that you submitted.

    Check that you submitted the correct information, such as the correct parameter values or template URL.

    In the **Change set preview** section, check that CloudFormation will make all the changes that you expect. For example, you can check that CloudFormation adds, removes, and modifies the resources that you intended to add, remove, or modify. CloudFormation generates this preview by creating a change set for the stack. For more information, see Update CloudFormation stacks using change sets.

11. When you are satisfied with your changes, choose **Update stack**.

> **Note**
>
> At this point, you also have the option to view the change set to review your proposed updates more thoroughly. To do so, choose **View change set** instead of **Update stack**. CloudFormation displays the change set generated based on your updates. When you are ready to perform the stack update, choose **Execute**.

CloudFormation displays the stack details page for your stack, with the **Events** pane selected. Your stack now has a status of `UPDATE_IN_PROGRESS`. After CloudFormation has successfully finished updating the stack, it sets the stack status to `UPDATE_COMPLETE`.

If the stack update fails, CloudFormation; automatically rolls back changes, and sets the stack status to `UPDATE_ROLLBACK_COMPLETE`.

> **Note**
>
> You can cancel an update while it's in the `UPDATE_IN_PROGRESS` state. For more information, see Cancel a stack update.

To update a stack using the command line

You can use one of the following commands:

- update-stack (AWS CLI)
- Update-CFNStack (AWS Tools for Windows PowerShell)

For examples of using the command line to update a stack, see Examples of CloudFormation stack operation commands for the AWS CLI and PowerShell.

Express mode

To update a stack using express mode, add the `--deployment-config` parameter. Express mode completes the stack operation as soon as resource configuration is applied. For more information, see Express mode.

```
aws cloudformation update-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --deployment-config '{"mode": "EXPRESS"}'
```

Cancel a stack update

After a stack update has begun, you can cancel the stack update if the stack is still in the UPDATE_IN_PROGRESS state. After an update has finished, you can't cancel it. You can, however, update a stack again with any previous settings.

If you cancel a stack update, the stack is rolled back to the stack configuration that existed before initiating the stack update.

Topics

- To cancel a stack update (console)
- To cancel a stack update (AWS CLI)

To cancel a stack update (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region where the stack is located.
3. On the **Stacks** page, choose the stack that's currently being updated. Its status must be UPDATE_IN_PROGRESS.
4. Choose **Stack actions** and then **Cancel update stack**.
5. To continue canceling the update, choose **Cancel update**. Otherwise, choose **Cancel** to resume the update.

The stack proceeds to the UPDATE_ROLLBACK_IN_PROGRESS state. After the update cancellation is complete, the stack is set to UPDATE_ROLLBACK_COMPLETE.

To cancel a stack update (AWS CLI)

Use the command cancel-update-stack to cancel an update. For more information, see Cancel a stack update.

Delete a stack from the CloudFormation console

If you no longer need the resources in a stack, you can delete the entire stack.

When deleting a stack, CloudFormation deletes all resources in that stack unless you used a deletion policy to retain specific resources. For more information, see DeletionPolicy attribute.

To delete a stack (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region where the stack is located.
3. On the **Stacks** page, choose the stack that you want to delete. The stack must be currently running.
4. Choose **Delete**.
5. When prompted for confirmation, choose **Delete**.

> **Note**
>
> The stack deletion operation can't be stopped once the stack deletion has begun. The stack proceeds to the `DELETE_IN_PROGRESS` state.
> After the stack deletion is complete, the stack will be in the `DELETE_COMPLETE` state. Stacks in the `DELETE_COMPLETE` state aren't displayed in the CloudFormation console by default. To display deleted stacks, you must change the stack view filter as described in View deleted stacks from the CloudFormation console.

To force delete a stack (console)

A stack deletion might fail because a resource in the stack fails to delete. For example, CloudFormation will fail the deletion of a resource that another stack also depends on. Any resources that haven't been deleted will remain until you can successfully delete the stack. If the deletion fails and returns a `DELETE_FAILED` state, you can choose to retry using one of two methods.

1. On the **Stacks** page in the CloudFormation console, choose the stack that you want to force delete.

2. In the stack details pane, choose **Retry delete**.
3. Choose between the following options:

   - **Delete this stack but retain resources**: This option allows you to select the specific resources that originally failed to delete, but you want to retain during the force stack deletion.
   - **Force delete this entire stack**: This option retains all resources that failed to delete, and retains dependencies of those resources.
4. Choose **Delete** to begin the force delete process with your selections.

To review retained resources (console)

After deleting the stack, you can view the resources that were retained in the console.

1. In the stacks list, choose the **Filter status** and select **Deleted**.
2. Choose the deleted stack.
3. Choose the **Resources** tab.
4. All retained resources show the `DELETE_SKIPPED` **Status**.
5. Choose the retained resource you want to review.

To delete a stack using the command line

You can use one of the following commands:

- delete-stack (AWS CLI)
- Remove-CFNStack (AWS Tools for Windows PowerShell)

For examples of using the command line to delete a stack, see Examples of CloudFormation stack operation commands for the AWS CLI and PowerShell.

Express mode

To delete a stack using express mode, add the `--deployment-config` parameter to the `delete-stack` command. Express mode completes the delete operation as soon as CloudFormation issues the delete API call for each resource, without waiting for resources to be fully removed. For more information, see Express mode.

```
aws cloudformation delete-stack --stack-name myteststack \
    --deployment-config '{"mode": "EXPRESS"}'
```

> **Note**
>
> In the CloudFormation console, you can select **Express** for the deployment mode when deleting a stack to use express mode.

Related resources

For help troubleshooting stack deletion errors, see the Delete stack fails troubleshooting topic.

For information on protecting stacks from being accidentally deleted, see Protect CloudFormation stacks from being deleted.

View deleted stacks from the CloudFormation console

By default, the CloudFormation console doesn't display stacks with a status of `DELETE_COMPLETE`. To display information about deleted stacks, you must change the stack view.

To view deleted stacks

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region where the deleted stack is located.
3. On the **Stacks** page, choose **Deleted** from the **Filter status** drop-down.

CloudFormation lists all your stacks with a status of `DELETE_COMPLETE`.

See also

- Delete a stack from the CloudFormation console
- View stack information from the CloudFormation console

Monitor stack progress

This section describes how to monitor a stack deployment that is currently in progress. CloudFormation provides a detailed, chronological list of deployment events, showing the progress and any issues encountered during the deployment.

Topics

- View CloudFormation stack events
- View stack events by operation
- View a timeline of a CloudFormation stack deployment
- Understand CloudFormation stack creation events
- Monitor the progress of a stack update
- Continue rolling back an update
- Determine the cause of a stack failure
- Choose how to handle failures when provisioning resources

View CloudFormation stack events

You can view stack events to monitor the progress and status of your stack and resources in the stack. Stack events help you understand when resources are being created, updated, or deleted, and whether the stack deployment is proceeding as expected.

Topics

- View stack events (console)
- View stack events (AWS CLI)
- Stack status codes

View stack events (console)

To view stack events

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the stack in.

3. On the **Stacks** page of the CloudFormation console, select the stack name. CloudFormation displays the stack details for the selected stack.
4. Choose the **Events** tab to view the stack events CloudFormation has generated for your stack.

CloudFormation automatically refreshes the stack events every minute. Additionally, CloudFormation displays the **New events available** badge when new stack events occur. Choose the refresh icon to load these events into the list. By viewing stack creation events, you can understand the sequence of events that lead to your stack's creation (or failure, if you are debugging your stack).

While your stack is being created, it's listed on the **Stacks** page with a status of `CREATE_IN_PROGRESS`. After your stack has been successfully created, its status changes to `CREATE_COMPLETE`.

For more information, see Understand CloudFormation stack creation events and Monitor the progress of a stack update.

View stack events (AWS CLI)

Alternatively, you can use the describe-stack-events command while the stack is being created to view events as they're reported.

The following **describe-stack-events** command describes the *my-stack* stack events.

```
aws cloudformation describe-stack-events --stack-name my-stack
```

The following is an example response.

```
{
    "StackEvents": [
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "7b755820-7edf-11f0-ab15-0673b09f3847",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:aws-
region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2025-08-21T22:37:56.243000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
```

```
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServer-CREATE_COMPLETE-2025-08-21T22:37:54.356Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServer",
            "PhysicalResourceId": "i-099df76cb31b866a9",
            "ResourceType": "AWS::EC2::Instance",
            "Timestamp": "2025-08-21T22:37:54.356000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"UserData\":
\"IyEvYmluL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYmluL2Nmbi1pbml0IC12I
\",\"ImageId\":\"ami-0bbc328167dee8f3c\",\"InstanceType\":\"t2.micro\",
\"SecurityGroupIds\":[\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\"],\"Tags\":
[{\"Value\":\"Bootstrap Tutorial Web Server\",\"Key\":\"Name\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServer-CREATE_IN_PROGRESS-2025-08-21T22:37:31.226Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServer",
            "PhysicalResourceId": "i-099df76cb31b866a9",
            "ResourceType": "AWS::EC2::Instance",
            "Timestamp": "2025-08-21T22:37:31.226000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"UserData\":
\"IyEvYmluL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYmluL2Nmbi1pbml0IC12I
\",\"ImageId\":\"ami-0bbc328167dee8f3c\",\"InstanceType\":\"t2.micro\",
\"SecurityGroupIds\":[\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\"],\"Tags\":
[{\"Value\":\"Bootstrap Tutorial Web Server\",\"Key\":\"Name\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServer-CREATE_IN_PROGRESS-2025-08-21T22:37:29.210Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServer",
            "PhysicalResourceId": "",
```

```
            "ResourceType": "AWS::EC2::Instance",
            "Timestamp": "2025-08-21T22:37:29.210000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceProperties": "{\"UserData\":
\"IyEvYmluL2Jhc2gKeXVtIGluc3RhbGwgLXkgYXdzLWNmbi1ib290c3RyYXAKL29wdC9hd3MvYmluL2Nmbi1pbml0IC12I
\",\"ImageId\":\"ami-0bbc328167dee8f3c\",\"InstanceType\":\"t2.micro\",
\"SecurityGroupIds\":[\"my-stack-WebServerSecurityGroup-n8A43bQT1ty2\"],\"Tags\":
[{\"Value\":\"Bootstrap Tutorial Web Server\",\"Key\":\"Name\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServerSecurityGroup-
CREATE_COMPLETE-2025-08-21T22:37:28.803Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServerSecurityGroup",
            "PhysicalResourceId": "my-stack-WebServerSecurityGroup-n8A43bQT1ty2",
            "ResourceType": "AWS::EC2::SecurityGroup",
            "Timestamp": "2025-08-21T22:37:28.803000+00:00",
            "ResourceStatus": "CREATE_COMPLETE",
            "ResourceProperties": "{\"GroupDescription\":\"Allow HTTP access from my IP
 address\",\"SecurityGroupIngress\":[{\"CidrIp\":\"0.0.0.0/0\",\"Description\":\"HTTP
\",\"FromPort\":\"80\",\"ToPort\":\"80\",\"IpProtocol\":\"tcp\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServerSecurityGroup-
CREATE_IN_PROGRESS-2025-08-21T22:37:22.626Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServerSecurityGroup",
            "PhysicalResourceId": "my-stack-WebServerSecurityGroup-n8A43bQT1ty2",
            "ResourceType": "AWS::EC2::SecurityGroup",
            "Timestamp": "2025-08-21T22:37:22.626000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "Resource creation Initiated",
            "ResourceProperties": "{\"GroupDescription\":\"Allow HTTP access from my IP
 address\",\"SecurityGroupIngress\":[{\"CidrIp\":\"0.0.0.0/0\",\"Description\":\"HTTP
\",\"FromPort\":\"80\",\"ToPort\":\"80\",\"IpProtocol\":\"tcp\"}]}",
            "ClientRequestToken": "token"
        },
        {
```

```
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "WebServerSecurityGroup-
CREATE_IN_PROGRESS-2025-08-21T22:37:20.186Z",
            "StackName": "my-stack",
            "LogicalResourceId": "WebServerSecurityGroup",
            "PhysicalResourceId": "",
            "ResourceType": "AWS::EC2::SecurityGroup",
            "Timestamp": "2025-08-21T22:37:20.186000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceProperties": "{\"GroupDescription\":\"Allow HTTP access from my IP
 address\",\"SecurityGroupIngress\":[{\"CidrIp\":\"0.0.0.0/0\",\"Description\":\"HTTP
\",\"FromPort\":\"80\",\"ToPort\":\"80\",\"IpProtocol\":\"tcp\"}]}",
            "ClientRequestToken": "token"
        },
        {
            "StackId": "arn:aws:cloudformation:aws-region:123456789012:stack/my-
stack/64726230-7edf-11f0-8a36-06453a64f325",
            "EventId": "64740fe0-7edf-11f0-8a36-06453a64f325",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:aws-
region:123456789012:stack/my-stack/64726230-7edf-11f0-8a36-06453a64f325",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2025-08-21T22:37:17.819000+00:00",
            "ResourceStatus": "CREATE_IN_PROGRESS",
            "ResourceStatusReason": "User Initiated",
            "ClientRequestToken": "token"
        }
    ]
}
```

The most recent events are reported first. The following table describe the fields returned by the **describe-stack-events** command:

Field	Description
`EventId`	Event identifier.
`StackName`	Name of the stack that the event corresponds to.
`StackId`	Identifier of the stack that the event corresponds to.

Field	Description
LogicalResourceId	Logical identifier of the resource.
PhysicalResourceId	Physical identifier of the resource.
ResourceProperties	Properties of the resource.
ResourceType	Type of the resource.
Timestamp	Time when the event occurred.
ResourceStatus	The status of the resource, which can be one of the following status codes: CREATE_COMPLETE | CREATE_FAILED | CREATE_IN_PROGRESS | DELETE_COMPLETE | DELETE_FAILED | DELETE_IN_PROGRESS | DELETE_SKIPPED | IMPORT_COMPLETE | IMPORT_IN_PROGRESS | IMPORT_ROLLBACK_COMPLETE | IMPORT_ROLLBACK_FAILED | IMPORT_ROLLBACK_IN_PROGRESS | REVIEW_IN_PROGRESS | ROLLBACK_COMPLETE | ROLLBACK_FAILED | ROLLBACK_IN_PROGRESS | UPDATE_COMPLETE | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_FAILED | UPDATE_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE | UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_FAILED | UPDATE_ROLLBACK_IN_PROGRESS The DELETE_SKIPPED status applies to resources with a deletion policy attribute of retain.
DetailedStatus	The detailed status of the stack. If CONFIGURATION_COMPLETE is present, the stack resources configuration phase has completed and the stabilization of the resources is in progress.
ResourceStatusReason	More information on the status.

Stack status codes

The following table describes stack status codes:

Stack status and optional detailed status	Description
`CREATE_COMPLETE`	Successful creation of one or more stacks.
`CREATE_IN_PROGRESS`	Ongoing creation of one or more stacks.
`CREATE_FAILED`	Unsuccessful creation of one or more stacks. View the stack events to see any associated error messages. Possible reasons for a failed creation include insufficient permissions to work with all resources in the stack, parameter values rejected by an AWS service, or a timeout during resource creation.
`DELETE_COMPLETE`	Successful deletion of one or more stacks. Deleted stacks are retained and viewable for 90 days.
`DELETE_FAILED`	Unsuccessful deletion of one or more stacks. Because the delete failed, you might have some resources that are still running; however, you can't work with or update the stack. Delete the stack again or view the stack events to see any associated error messages.
`DELETE_IN_PROGRESS`	Ongoing removal of one or more stacks.
`REVIEW_IN_PROGRESS`	Ongoing creation of one or more stacks with an expected `StackId` but without any templates or resources. **Important** A stack with this status code counts against the maximum possible number of stacks.
`ROLLBACK_COMPLETE`	Successful removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation. The stack returns to the previous working state. Any resources

Stack status and optional detailed status	Description
	that were created during the create stack operation are deleted. This status exists only after a failed stack creation. It signifies that all operations from the partially created stack have been appropriately cleaned up. When in this state, only a delete operation can be performed.
ROLLBACK_FAILED	Unsuccessful removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation. Delete the stack or view the stack events to see any associate d error messages.
ROLLBACK_IN_PROGRESS	Ongoing removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation.
UPDATE_COMPLETE	Successful update of one or more stacks.
UPDATE_COMPLETE_CL EANUP_IN_PROGRESS	Ongoing removal of old resources for one or more stacks after a successful stack update. For stack updates that require resources to be replaced, CloudFormation creates the new resources first and then deletes the old resources to help reduce any interruptions with your stack. In this state, the stack has been updated and is usable, but CloudFormation is still deleting the old resources.
UPDATE_FAILED	Unsuccessful update of one or more stacks. View the stack events to see any associated error messages.
UPDATE_IN_PROGRESS	Ongoing update of one or more stacks.
UPDATE_ROLLBACK_CO MPLETE	Successful return of one or more stacks to a previous working state after a failed stack update.

Stack status and optional detailed status	Description
UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS	Ongoing removal of new resources for one or more stacks after a failed stack update. In this state, the stack has been rolled back to its previous working state and is usable, but CloudFormation is still deleting any new resources it created during the stack update.
UPDATE_ROLLBACK_FAILED	Unsuccessful return of one or more stacks to a previous working state after a failed stack update. When in this state, you can delete the stack or continue rollback. You might need to fix errors before your stack can return to a working state. Or, you can contact Support to restore the stack to a usable state.
UPDATE_ROLLBACK_IN_PROGRESS	Ongoing return of one or more stacks to the previous working state after failed stack update.
IMPORT_IN_PROGRESS	The import operation is currently in progress.
IMPORT_COMPLETE	The import operation successfully completed for all resources in the stack that support `resource import`.
IMPORT_ROLLBACK_IN_PROGRESS	Import will roll back to the previous template configuration.
IMPORT_ROLLBACK_FAILED	The import rollback operation failed for at least one resource in the stack. Results will be available for the resources CloudFormation successfully imported.
IMPORT_ROLLBACK_COMPLETE	Import successfully rolled back to the previous template configuration.

View stack events by operation

You can view stack events grouped by operation to better understand the sequence and scope of changes made to your stack. Operation-based grouping helps you track related events together, making it easier to monitor progress and troubleshoot issues when they occur.

Each stack operation (create, update, delete, rollback) is assigned a unique operation ID that groups all related events. This allows you to focus on specific operations and quickly identify the root cause of failures.

Topics

- Prerequisites
- View stack events by operation (console)
- View stack events by operation (AWS CLI)
- Stack status codes

Prerequisites

To use `DescribeEvents` API, you must have the necessary IAM permissions to: `DescribeEvents`.

View stack events by operation (console)

To view stack events grouped by operation

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the stack in.
3. On the **Stacks** page, select the stack name. CloudFormation displays the stack details for the selected stack.
4. Choose the **Events** tab to view the stack events CloudFormation has generated for your stack.
5. Events are automatically grouped by operation ID. Each operation appears as an expandable section showing the operation type, status, and timestamp.
6. Click on an **operation ID** to open a detailed view showing only the events related to that specific operation.

7. In the operation detail view, select the **Display failures only** checkbox to display only failed events for root cause analysis.

CloudFormation automatically refreshes the stack events every minute. The **New events available** badge appears when new stack events occur. Choose the refresh icon to load these events into the list.

By viewing stack events grouped by operation, you can understand the sequence of events for each operation and quickly identify which specific operation caused issues (if you are debugging your stack).

While your stack operation is running, it's listed with a status of `CREATE_IN_PROGRESS`, `UPDATE_IN_PROGRESS`, or `DELETE_IN_PROGRESS`. After your operation has completed successfully, its status changes to `CREATE_COMPLETE`, `UPDATE_COMPLETE`, or `DELETE_COMPLETE`.

For more information, see Understand CloudFormation stack creation events and Monitor the progress of a stack update.

View stack events by operation (AWS CLI)

You can use the `describe-events` command with operation ID filtering to view events for specific operations.

Get last operation IDs

The stack description available via describe-stacks API now includes LastOperations information showing recent operation IDs and their types. This enables you to quickly identify which operations occurred and their current status without parsing through event logs.

```
aws cloudformation describe-stacks --stack-name MyStack
```

The following is an example response showing the last operation was a Rollback following a failed Update operation.

```
{
    "Stacks": [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
MyStack/07580010-bb79-11f0-8f6c-0289bb5c804f",
```

```
            "StackName": "MyStack",
            "Description": "A simple CloudFormation template to create an S3 bucket.",

            "CreationTime": "2025-11-07T01:28:13.778000+00:00",
            "LastUpdatedTime": "2025-11-07T01:43:39.838000+00:00",
            "RollbackConfiguration": {},
            "StackStatus": "UPDATE_ROLLBACK_COMPLETE",
            "DisableRollback": false,
            "NotificationARNs": [],
            "Tags": [],
            "EnableTerminationProtection": false,
            "DriftInformation": {
                "StackDriftStatus": "NOT_CHECKED"
            },
            "LastOperations": [
                {
                    "OperationType": "ROLLBACK",
                    "OperationId": "d0f12313-7bdb-414d-a879-828a99b36f29"
                },
                {
                    "OperationType": "UPDATE_STACK",
                    "OperationId": "1c211b5a-4538-4dc9-bfed-e07734371e57"
                }
            ]
        }
    ]
}
```

Filter events by operation ID

The following `describe-events` command describes events for a specific operation ID:

```
aws cloudformation describe-events \
  --operation-id 1c211b5a-4538-4dc9-bfed-e07734371e57
```

To view only failed events for troubleshooting, use the `--filter FailedEvents=true` parameter:

```
aws cloudformation describe-events \
  --operation-id 1c211b5a-4538-4dc9-bfed-e07734371e57 \
  --filter FailedEvents=true
```

The new operation ID filtering capability allows you to focus on specific operations and their related events. This is particularly useful for:

- **Troubleshooting specific failures:** Isolate events from a failed operation to understand what went wrong.
- **Monitoring long-running operations:** Track progress of complex updates or large stack deployments.
- **Auditing changes:** Review all events associated with a particular update operation.
- **Root cause analysis:** Use the failure filter to quickly identify the source of deployment issues.

Stack status codes

The following table describes the fields returned by the `describe-events` command when using operation ID filtering:

Field	Description
`EventId`	Event identifier.
`OperationId`	Unique identifier for the operation that generated this event.
`StackName`	Name of the stack that the event corresponds to.
`StackId`	Identifier of the stack that the event corresponds to.
`LogicalResourceId`	Logical identifier of the resource.
`PhysicalResourceId`	Physical identifier of the resource.
`ResourceProperties`	Properties of the resource.
`ResourceType`	Type of the resource.
`Timestamp`	Time when the event occurred.

Field	Description
`ResourceStatus`	The status of the resource (`CREATE_COMPLETE` , `UPDATE_FAILED` , etc.).
`DetailedStatus`	The detailed status of the stack. If `CONFIGURATION_COMPLETE` is present, the stack resources configuration phase has completed and the stabilization of the resources is in progress.
`ResourceStatusReason`	More information on the status.

View a timeline of a CloudFormation stack deployment

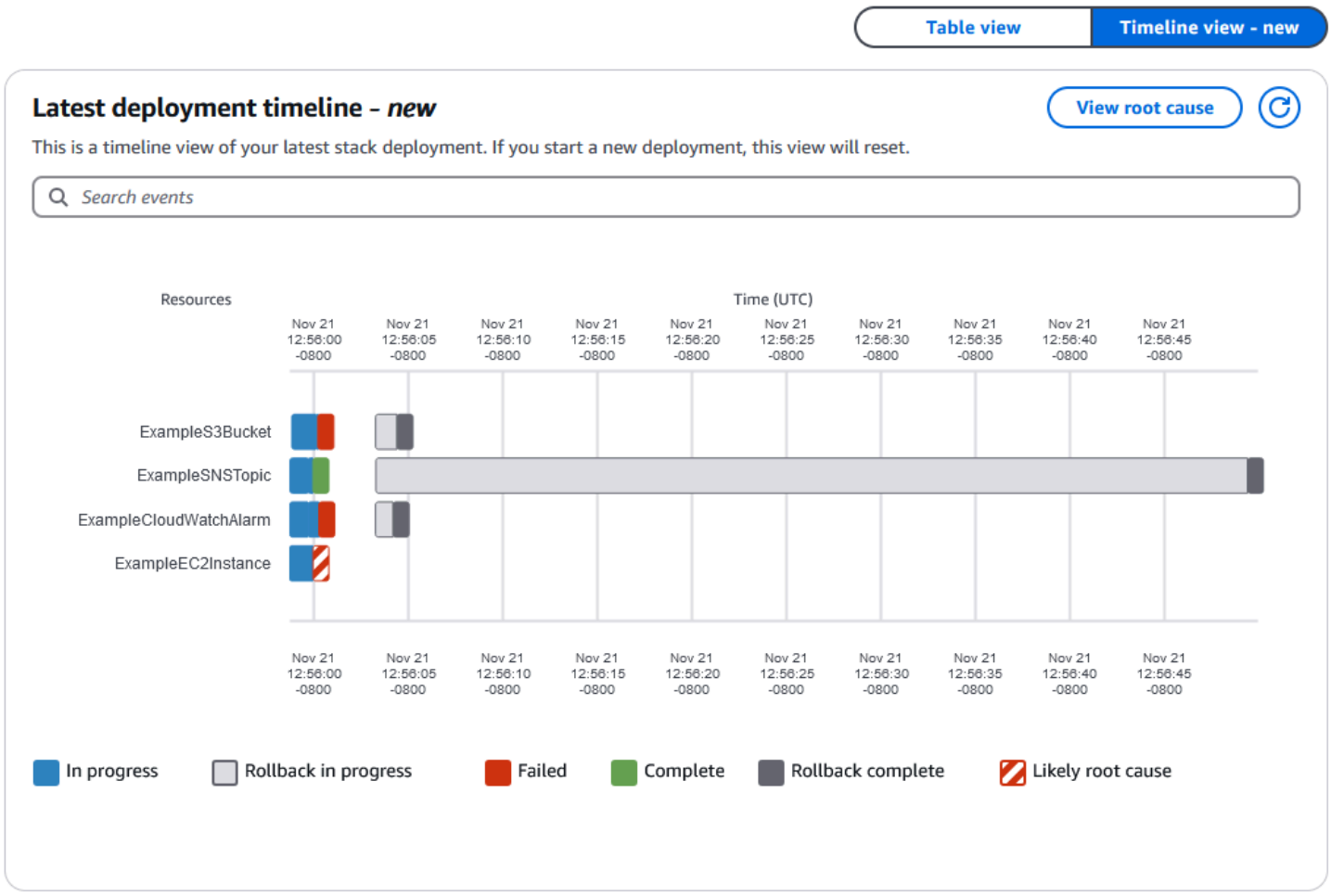
The stack deployment timeline graph provides a visual representation of a stack deployment timeline. This view shows the deployment statuses for the stack and each of its resources, and the times that each status changed. Stack deployment statuses are represented by a corresponding color.

Topics

- Understanding the stack deployment timeline graph
- Viewing the stack deployment timeline graph (console)

Understanding the stack deployment timeline graph

The following image shows the timeline graph for a stack deployment that failed due to an Amazon EC2 instance resource that failed to launch.

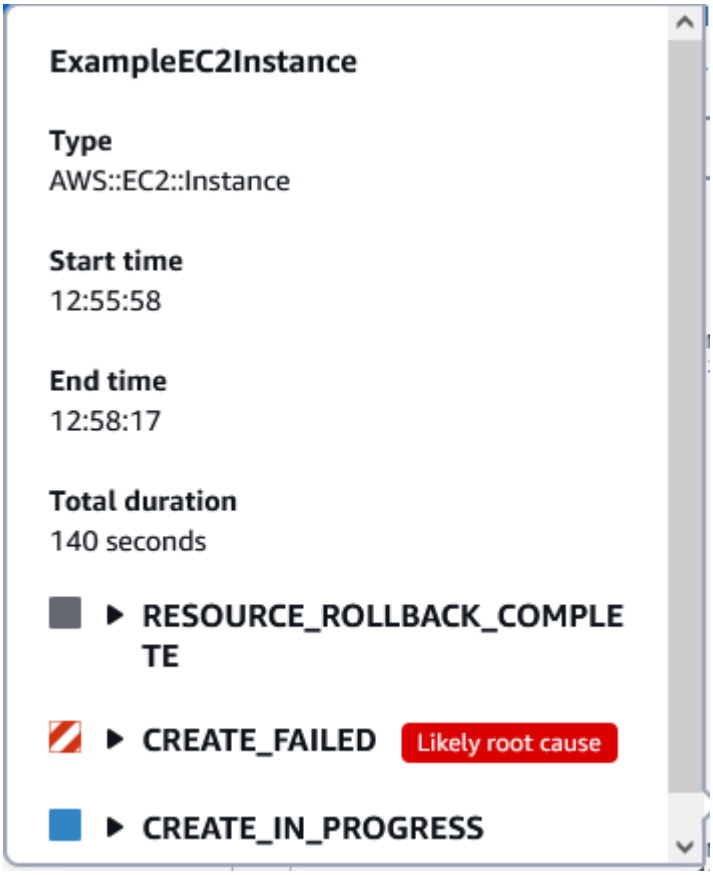


The names of the stack resources are found on the left side of the graph, and the date and time relative to the deployment times are found at the top of the graph.

Each resource starts with the **In progress** status. The status bar changes to **Complete** for each successful deployment. The status bar changes to **Failed** when a resource fails to deploy. When a resource fails to deploy and the stack deployment also fails, the resource responsible for the stack deployment failure receives the **Likely root failure** status.

After the stack deployment operation failed, the successfully deployed resource begin rolling back and change to the **Rollback in progress** status. The statuses change to **Rollback complete** after the resource finish rolling back.

Choosing each resource provides more granular detail on the deployment timeline:



Choosing a resource shows the **Type**, deployment **Start time**, deployment **End time**, and **Total duration** of the deployment. You will also find the **Start time**, **End time**, and **Duration** of each deployment status in the drop-down menus below. If the resource failed to deploy, a **Failure reason** will be provided.

For more information on stack statuses, see Stack status codes.

Viewing the stack deployment timeline graph (console)

To view a stack deployment timeline graph:

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the stack in.
3. On the **Stacks** page of the CloudFormation console, choose the stack name. CloudFormation displays the stack details for the selected stack.
4. Choose the **Events** tab to view the stack events CloudFormation has generated for your stack.
5. Choose the **Timeline graph** button to view the timeline graph for your stack.

Understand CloudFormation stack creation events

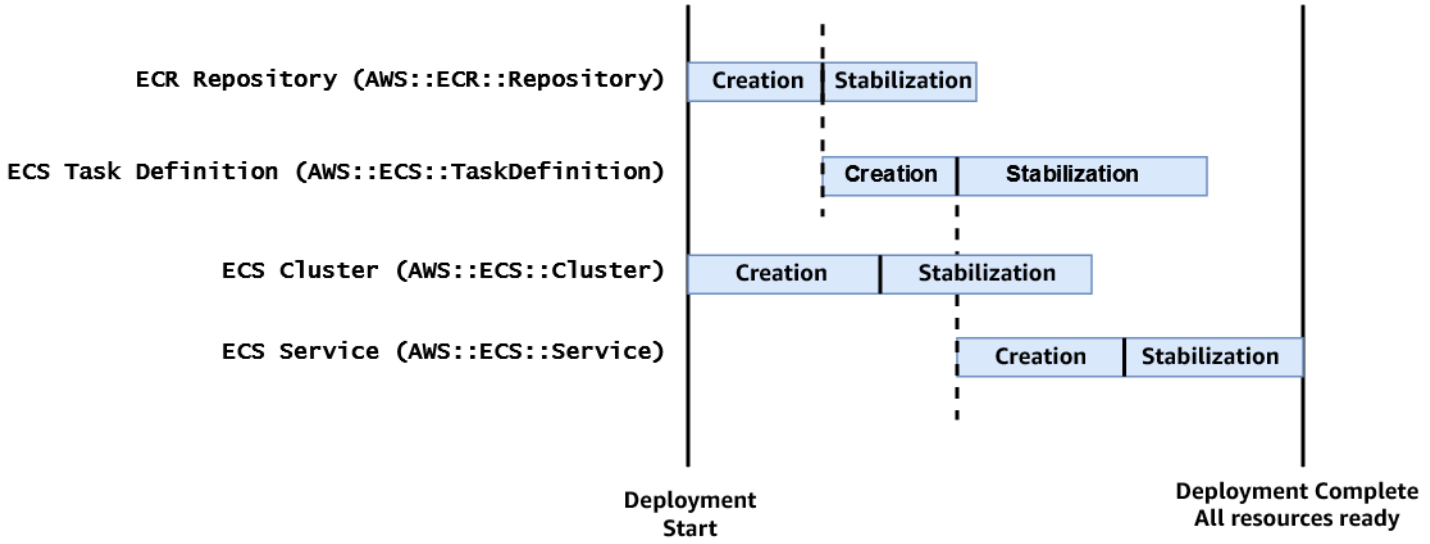
During stack deployment, several events occur to create, configure, and validate the resources defined in the stack template. Understanding these events can help you optimize your stack creation process and streamline deployments.

- **Resource creation events** – When each resource starts the creation process, a **Status** of `CREATE_IN_PROGRESS` event is set. This event indicates that the resource is being provisioned.
- **Eventual consistency check** – A significant portion of the stack creation time is spent performing an eventual consistency check against the resources created by the stack. During this phase, the service performs internal consistency checks, ensuring the resource is fully operational and meets service stabilization criteria defined by each AWS service.
- **Configuration complete event** – When each resource has finished the eventual consistency check phase of the provisioning, a **Detailed status** of `CONFIGURATION_COMPLETE` event is set.
- **Resource creation complete event** – After the resource has been created and configured as specified, and the configuration matches what is specified in the template, the **Status** of `CREATE_COMPLETE` event is set.

You can leverage the `CONFIGURATION_COMPLETE` event to streamline your stack creation process in scenarios where resource eventual consistency check is not required, such as validating a pre-production stack configuration or cross-stack provisioning. You can use this event in multiple ways. For example, you can use it as a visual signal to skip waiting for the resource or stack consistency check to finish. Or you could use it to create an automated mechanism using continuous integration and continuous delivery (CI/CD) to trigger additional actions.

> **Important**
>
> While leveraging the `CONFIGURATION_COMPLETE` event accelerates stack creation times, you should be aware of its trade-offs. First, it's only supported for a subset of resource type that support drift detection. For a list of resource types that support drift detection, see Resource type support. This approach may not be suitable for all scenarios, especially where resources require thorough eventual consistency checks to ensure full operational readiness across the cloud environment (for example, in production environments). We recommend carefully assessing your deployment requirements and the criticality of the consistency checks for each resource. Use the `CONFIGURATION_COMPLETE` event to

optimize deployment speeds without compromising the integrity and reliability of your infrastructure.
Because the CONFIGURATION_COMPLETE event is not guaranteed to be set, any scenarios that use it should be prepared to handle a CREATE_COMPLETE event when no CONFIGURATION_COMPLETE event was set.



When the stack deployment starts, both the AWS::ECR::Repository and AWS::ECS::Cluster resources start the creation process (ResourceStatus = CREATE_IN_PROGRESS. When the AWS::ECR::Repository resource type has started the eventual consistency check (DetailedStatus = CONFIGURATION_COMPLETE), then the AWS::ECS::TaskDefinition resource can start the creation process. Similarly, once the AWS::ECS::TaskDefinition resource begins the eventual consistency check, the AWS::ECS::Service resource start the creation process.

CREATE_IN_PROGRESS and CREATE_COMPLETE events

- **[Stack]:** CREATE_IN_PROGRESS
- **[Resource]:** ECR Repository CREATE_IN_PROGRESS
- **[Resource]:** ECS Cluster CREATE_IN_PROGRESS
- **[Resource]:** ECR Repository CREATE_IN_PROGRESS, CONFIGURATION_COMPLETE
- **[Resource]:** ECS Task Definition CREATE_IN_PROGRESS
- **[Resource]:** ECS Cluster CREATE_IN_PROGRESS, CONFIGURATION_COMPLETE
- **[Resource]:** ECS Task Definition CREATE_IN_PROGRESS, CONFIGURATION_COMPLETE

- **[Resource]:** ECS Service CREATE_IN_PROGRESS
- **[Resource]:** ECR Repository CREATE_COMPLETE
- **[Resource]:** ECS Cluster CREATE_COMPLETE
- **[Resource]:** ECS Service CREATE_IN_PROGRESS, CONFIGURATION_COMPLETE
- **[Stack]:** CREATE_IN_PROGRESS, CONFIGURATION_COMPLETE
- **[Resource]:** ECS Task Definition CREATE_COMPLETE
- **[Resource]:** ECS Service CREATE_COMPLETE
- **[Stack]:** CREATE_COMPLETE

Monitor the progress of a stack update

You can monitor the progress of a stack update by viewing the stack's events. The stack's **Events** tab displays each major step in the creation and update of the stack sorted by the time of each event with latest events on top. For more information, see Monitor stack progress.

Topics

- Events generated during a successful stack update
- Events generated when a resource update fails

Events generated during a successful stack update

The start of the stack update process is marked with an UPDATE_IN_PROGRESS event for the stack:

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_IN_PROGRESS
```

Next are events that mark the beginning and completion of the update of each resource that was changed in the update template. For example, updating an AWS::RDS::DBInstance resource named MyDB would result in the following entries:

```
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

The UPDATE_IN_PROGRESS event is logged when CloudFormation reports that it has begun to update the resource. The UPDATE_COMPLETE event is logged when the resource is successfully created.

When CloudFormation has successfully updated the stack, you will see the following event:

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_COMPLETE
```

Important

During stack update operations, if CloudFormation needs to replace an existing resource, it first creates a new resource and then deletes the old resource. However, there may be cases where CloudFormation can't delete the old resource (for example, if the user doesn't have permissions to delete a resource of a given type).
CloudFormation makes three attempts at deleting the old resource. If CloudFormation can't delete the old resource, it removes the old resource from the stack and continues updating the stack. When the stack update is complete, CloudFormation issues an UPDATE_COMPLETE stack event, but includes a StatusReason that states that one or more resources couldn't be deleted. CloudFormation also issues a DELETE_FAILED event for the specific resource, with a corresponding StatusReason providing more detail on why CloudFormation failed to delete the resource.
The old resource still exists and will continue to incur charges, but is no longer accessible through CloudFormation. To delete the old resource, access the old resource directly using the console or API for the underlying service.
This is also true for resources that you have removed from the stack template, and so will be deleted from the stack during the stack update.

Events generated when a resource update fails

If an update of a resource fails, CloudFormation reports an UPDATE_FAILED event that includes a reason for the failure. For example, if your update template specified a property change that's not supported by the resource such as reducing the size of AllocatedStorage for an AWS::RDS::DBInstance resource, you would see events like these:

```
2011-09-30 09:36 PDT AWS::RDS::DBInstance MyDB UPDATE_FAILED Size cannot be less than
 current size; requested: 5; current: 10
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

If a resource update fails, CloudFormation rolls back any resources that it has updated during the upgrade to their configurations before the update. Here is an example of the events you would see during an update rollback:

```
2011-09-30 09:38 PDT AWS::CloudFormation::Stack MyStack UPDATE_ROLLBACK_COMPLETE
2011-09-30 09:38 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
2011-09-30 09:37 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
2011-09-30 09:37 PDT AWS::CloudFormation::Stack MyStack UPDATE_ROLLBACK_IN_PROGRESS The
 following resource(s) failed to update: [MyDB]
```

Continue rolling back an update

Sometimes, when CloudFormation tries to roll back a stack update, it can't roll back all the changes it made during the update process. This is called the UPDATE_ROLLBACK_FAILED state. For example, you might have a stack that begins to roll back to an old database instance that was deleted outside of CloudFormation. Because CloudFormation doesn't know that the database was deleted, it assumes that the database instance still exists and attempts to roll back to it, causing the update rollback to fail.

A stack in the UPDATE_ROLLBACK_FAILED state can't be updated, but it can be rolled back to a working state (UPDATE_ROLLBACK_COMPLETE). After returning the stack to its original settings, you can try to update it again.

In most cases, you must fix the error that causes the update rollback to fail before you can continue to roll back your stack. In other cases, you can continue to roll back the update without any changes, for example when a stack operation times out.

If you can't fix the error, you can force the rollback to complete. CloudFormation skips each resource that fails to roll back and returns the stack to UPDATE_ROLLBACK_COMPLETE. For more information, see Force a rollback to complete.

> **Note**
>
> If you use nested stacks, rolling back the parent stack will attempt to roll back all the child stacks as well.

To continue rolling back an update (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region where the stack is located.
3. On the **Stacks** page, choose the stack that you want to update, choose **Stack actions**, and then choose **Continue update rollback**.

   If none of the solutions in the Troubleshooting errors worked, you can use the advanced option to skip the resources that CloudFormation can't successfully roll back. You must look up and type the logical IDs of the resources that you want to skip. Specify only resources that went into the `UPDATE_FAILED` state during the `UpdateRollback` and not during the forward update.

   > **Warning**
   >
   > CloudFormation sets the status of the specified resources to `UPDATE_COMPLETE` and continues to roll back the stack. After the rollback is complete, the state of the skipped resources will be inconsistent with the state of the resources in the stack template. Before performing another stack update, you must update the stack or resources to be consistent with each other. If you don't, subsequent stack updates might fail, and the stack will become unrecoverable.

   Specify the minimum number of resources required to successfully roll back your stack. For example, a failed resource update might cause dependent resources to fail. In this case, it might not be necessary to skip the dependent resources.

   To skip resources that are part of nested stacks, use the following format: *`NestedStackName`*`.`*`ResourceLogicalID`*. If you want to specify the logical ID of a

stack resource (`Type: AWS::CloudFormation::Stack`) in the `ResourcesToSkip` list, then its corresponding embedded stack must be in one of the following states: `DELETE_IN_PROGRESS`, `DELETE_COMPLETE`, or `DELETE_FAILED`.

To continue rolling back an update (AWS CLI)

- Use the continue-update-rollback command with the `--stack-name` option to specify the ID of the stack that you want to continue to roll back.

  To force the rollback to complete, add the `--force-rollback` option. For more information, see Force a rollback to complete.

Force a rollback to complete

Use force rollback when you can't fix the error that causes the update rollback to fail. Instead of looking up and listing each resource in `--resources-to-skip`, you let CloudFormation skip the resources that fail to roll back.

When you force a rollback, CloudFormation continues rolling back the stack. CloudFormation skips each resource that is in the `UPDATE_FAILED` state because a rollback failed, including resources in nested stacks. CloudFormation sets the status of each skipped resource to `UPDATE_COMPLETE` with the status reason `Resource skipped during UpdateRollback`. When the rollback completes, the stack status reason reports that one or more resources were skipped.

CloudFormation skips only resources whose rollback failed. Resources that failed during the forward update aren't skipped.

The following **continue-update-rollback** example forces the rollback of the *MyStack* stack to complete.

```
aws cloudformation continue-update-rollback --stack-name MyStack \
    --force-rollback
```

You can't use `--force-rollback` and `--resources-to-skip` in the same request. To skip only a specific set of resources, use `--resources-to-skip` without `--force-rollback`.

> **Warning**
>
> After the rollback completes, the skipped resources no longer match the resources in the stack template. Before you perform another stack update, you must update the stack or resources to be consistent with each other. If you don't, subsequent stack updates might fail, and the stack will become unrecoverable.

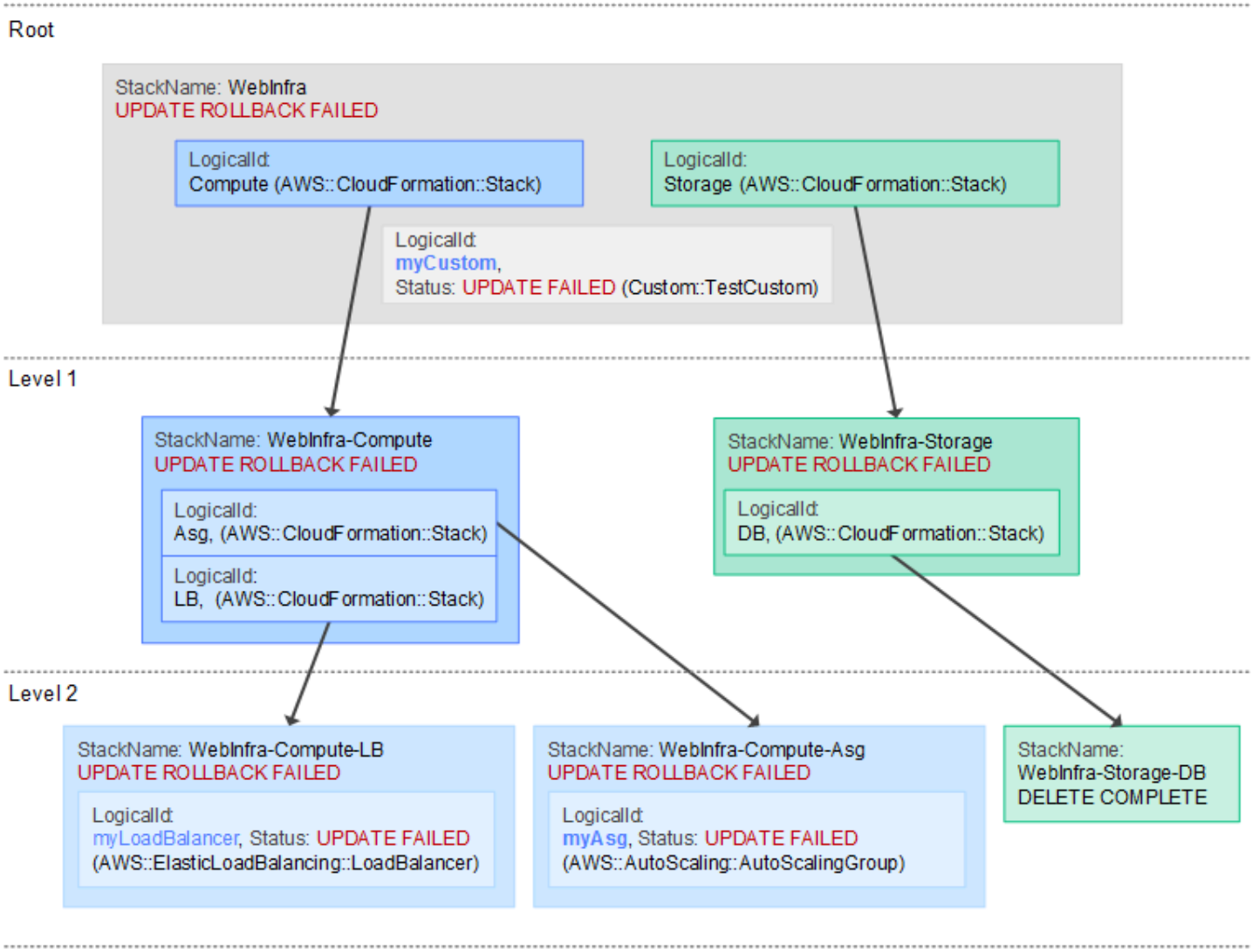
Drift detection reports skipped resources as `NOT_CHECKED`. To find out whether a skipped resource differs from the template, run DetectStackResourceDrift on the resource.

Continue rolling back from failed nested stack updates

When you have multiple stacks nested within each other, you may need to skip resources across multiple nested levels to get the full stack hierarchy back to a working state.

For example, you have a root stack called `WebInfra` that contains two smaller stacks inside it: `WebInfra-Compute` and `WebInfra-Storage`. These two stacks also have their own nested stacks within them.

If something goes wrong during an update, and the update process fails, the entire stack hierarchy may end up in the `UPDATE_ROLLBACK_FAILED` state, as shown in the following diagram.



Note

The stack names in this example are truncated for simplicity. Child stack names are typically generated by CloudFormation and contain unique random strings, so actual names might not be user-friendly.

To get the root stack into an operable state using the `continue-update-rollback` command, you must skip the resources that failed to roll back. You can either use the `--force-rollback` option to skip them across all nested levels, or use the `--resources-to-skip` option to list them.

The following **continue-update-rollback** example forces the rollback of the whole hierarchy to complete. You don't need to find the nested stack names or logical IDs.

```
aws cloudformation continue-update-rollback --stack-name WebInfra \
    --force-rollback
```

The following **continue-update-rollback** example resumes a rollback operation from a previously failed stack update. In this example, the `--resources-to-skip` option includes the following items:

- *myCustom*
- *WebInfra-Compute-Asg.myAsg*
- *WebInfra-Compute-LB.myLoadBalancer*
- *WebInfra-Storage.DB*

For the resources of the root stack, you only need to provide the logical ID, for example, *myCustom*. However, for the resources that are contained in nested stacks, you must provide both the nested stack name and its logical ID, separated by a period. For example, *WebInfra-Compute-Asg.myAsg*.

```
aws cloudformation continue-update-rollback --stack-name WebInfra \
    --resources-to-skip myCustom WebInfra-Compute-Asg.myAsg WebInfra-Compute-
LB.myLoadBalancer WebInfra-Storage.DB
```

To find the stack name of a nested stack

You can locate it within the child stack's stack ID or Amazon Resource Name (ARN).

The following ARN example refers to a stack named `WebInfra-Storage-Z2VKC706XKXT`.

```
arn:aws:cloudformation:us-east-1:123456789012:stack/WebInfra-Storage-Z2VKC706XKXT/
ea9e7f90-54f7-11e6-a032-028f3d2330bd
```

To find the logical ID of a nested stack

You can find a child stack's logical ID in the template definition of its parent. In the diagram, the `LogicalId` of the `WebInfra-Storage-DB` child stack is `DB` in its parent `WebInfra-Storage`.

In the CloudFormation console, you can also find the logical ID in the **Logical ID** column for the stack resource on the **Resources** tab or the **Events** tab. For more information, see View stack information from the CloudFormation console.

> **Note**
>
> The `RollbackStack` operation also supports express mode. You can use `--deployment-config '{"mode": "EXPRESS"}'` with the **rollback-stack** command to roll back a failed stack faster. For more information, see Express mode.

Determine the cause of a stack failure

If your stack creation fails, CloudFormation can help you to determine the event that is likely the root cause for the stack failure. Depending on the scenario and your permissions, AWS CloudTrail events may be able to provide further details about the root cause in case the provided **Status reason** in **Events** is not clear.

To determine the root cause of a stack failure

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, choose the failed stack.
3. Choose the **Events** tab.
4. Choose **Detect root cause**. CloudFormation will analyze the failure and indicate the event that is the likely the cause for the failure by adding a **Likely root cause** label to the specific event **Status**. See **Status reason** for further explanation of the status in the CloudFormation console.
5. Choose the failed **Status** with the **Likely root cause** label to learn more about the cause of the failure. Depending on the scenario and your permissions, you may be able to review a detailed CloudTrail event. These are the following potential outcomes of choosing the **Status**

   - CloudTrail events related to this issue are available and may help with resolution. View CloudTrail events.
   - We couldn't find any CloudTrail events related to this issue that could help with resolution.
   - Your current permissions do not allow access to view CloudTrail events. Learn more.
   - In the process of checking for available CloudTrail events, check back in a few minutes.
   - An error occurred while fetching the CloudTrail events. For manual inspection, visit the CloudTrail console.

6. If the provided reason in **Status reason** isn't clear, and the root cause displays a link to the CloudTrail console, open the link to view the event to find a detailed root cause.

For more information on CloudTrail events, see Understanding CloudTrail events and CloudTrail record contents.

For more information on CloudTrail event history, see Working with CloudTrail Event history.

> **Note**
>
> Nested stacks don't support **Detect root cause**.

Choose how to handle failures when provisioning resources

If your stack operation fails, you don't have to roll back resources that were already successfully provisioned and start over from the beginning every time. Instead, you can troubleshoot resources in a `CREATE_FAILED` or `UPDATE_FAILED` status, and then resume provisioning from the point where the problem occurred.

To do this, you must enable the preserve successfully provisioned resources option. This option is available for all stack deployments and change set operations.

- For stack creation, if you choose the **Preserve successfully provisioned resources** option, CloudFormation preserves the state of resources that were successfully created and leaves the failed ones in a failed state until the next update operation is performed.
- During update and change set operations, choosing **Preserve successfully provisioned resources** preserves the state of successful resources while rolling back failed resources to their last known stable state. Failed resources will be in an `UPDATE_FAILED` state. Resources without a last known stable state will be deleted upon the next stack operation.

Topics

- Overview of stack failure options
- Required conditions for pausing stack rollback
- Preserve successfully provisioned resources (console)
- Preserve successfully provisioned resources (AWS CLI)

Overview of stack failure options

Before issuing an operation from the CloudFormation console, API, or AWS CLI, specify the behavior for provisioned resource failure. Then, proceed with the deployment process of your resources without any other modifications. In the event of an operational failure, CloudFormation stops at the first failure in each independent provisioning path. CloudFormation identifies dependencies between resources to parallelize independent provisioning actions. Then it proceeds to provision resources on each independent provisioning path until it encounters a failure. A failure in one path doesn't affect other provisioning paths. CloudFormation will continue to provision the resources until completion or stop on a different failure.

Remediate any issues to continue the deployment process. CloudFormation performs the necessary updates before retrying provisioning actions on resources that couldn't be successfully provisioned earlier. You remediate issues by submitting a **Retry**, **Update**, or **Roll back** operations. For example, if you're provisioning an Amazon EC2 instance and the EC2 instance fails during a create operation, you might want to investigate the error, rather than rolling back the failed resource right away. You can review system status checks and instances status checks, and then select the **Retry** operation once the issues is resolved.

When a stack operation fails, and you've specified **Preserve successfully provisioned resources** from the **Stack failure options** menu, you can select the following options.

- **Retry** – Retries provisioning operation on failed resources and continues provisioning the template until the successful completion of the stack operation or the next failure. Select this option if the resource failed to provision due to an issue that doesn't require template modifications, such as an AWS Identity and Access Management (IAM) permission.
- **Update** – Resources that have been provisioned are updated on template updates. Resources that failed to create or update will be retried. Select this option if the resource failed to provision due to template errors, and you've modified the template. When you update a stack that's in a `FAILED` state, you must select **Preserve successfully provisioned resources** for the **Stack failure options** to continue updating your stack.
- **Roll back** – CloudFormation rolls back the stack to the last known stable state.

Required conditions for pausing stack rollback

To prevent CloudFormation from automatically rolling back and deleting the resources that were successfully created, the following conditions must be met.

1. When you create or update the stack, you must choose the option to **Preserve successfully provisioned resources**. This tells CloudFormation not to delete the resources that were created successfully, even if the overall stack operation fails.
2. The stack operation must have failed, meaning the stack status is either `CREATE_FAILED` or `UPDATE_FAILED`.

> **Note**
>
> Immutable update types aren't supported.

Preserve successfully provisioned resources (console)

Create stack

To preserve successfully provisioned resources during a create stack operation

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the **Stacks** page, choose **Create stack** at top right, and then choose **With new resources (standard)**.
3. For **Prerequisite - Prepare template**, choose **Choose an existing template**.
4. Under **Specify template**, choose to either specify the URL for the S3 bucket that contains your stack template or upload a stack template file. Then, choose **Next**.
5. On the **Specify stack details** page, enter a stack name in the **Stack name** box.
6. In the **Parameters** section, specify parameters that are defined in your stack template.

   You can use or change any parameters with default values.
7. When you're satisfied with the parameter values, choose **Next**.
8. On the **Configure stack options** page, you can set additional options for your stack.
9. For **Stack failure options**, select **Preserve successfully provisioned resources**.
10. When you're satisfied with the stack options, choose **Next**.
11. Review your stack on the **Review** page and select **Create stack**.

Results: Resources that failed to create transition the stack status to `CREATE_FAILED` to prevent the stack from rolling back when the stack operation encounters a failure. Resources that are successfully provisioned are in a `CREATE_COMPLETE` state. You can monitor the stack in the **Stack events** tab.

Update stack

To preserve successfully provisioned resources during an update stack operation

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Select the stack you want to update and then choose **Update**.
3. On the **Update stack** page, choose a stack template by using one of the following options:
   - **Use existing template**
   - **Replace current template**
   - **Open in template studio**

   Accept your settings and select **Next**.
4. On the **Specify stack details** page, specify parameters that are defined in your stack template.

   You can use or change any parameters with default values.
5. When you're satisfied with the parameter values, choose **Next**.
6. On the **Configure stack options** page, you can set additional options for your stack.
7. For the **Behavior on provisioning failure**, select **Preserve successfully provisioned resources**.
8. When you're satisfied with the stack options, choose **Next**.
9. Review your stack on the **Review** page and select **Update stack**.

Results: Resources that failed to update transition the stack status to `UPDATE_FAILED` and roll back to the last known stable state. Resources without a last known stable state will be deleted by CloudFormation upon the next stack operation. Resources that are successfully provisioned are in a `CREATE_COMPLETE` or `UPDATE_COMPLETE` state. You can monitor the stack in the **Stack events** tab.

Change set

> **Note**
>
> You can initiate a change set for a stack with a status of `CREATE_FAILED` or `UPDATE_FAILED`, but not for a status of `UPDATE_ROLLBACK_FAILED`.

To Preserve successfully provisioned resources during a change set operation

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Select the stack that contains the change set you want to initiate, and then choose the **Change sets** tab.
3. Select the change set and then choose **Execute**.
4. For **Execute change set**, select the **Preserve successfully provisioned resources** option.
5. Select **Execute change set**.

Results: Resources that failed to update transition the stack status to `UPDATE_FAILED` and roll back to the last known stable state. Resources without a last known stable state will be deleted by CloudFormation upon the next stack operation. Resources that are successfully provisioned are in a `CREATE_COMPLETE` or `UPDATE_COMPLETE` state. You can monitor the stack in the **Stack events** tab.

Preserve successfully provisioned resources (AWS CLI)

Create stack

To preserve successfully provisioned resources during a stack create operation

Specify the `--disable-rollback` option or `on-failure DO_NOTHING` enumeration during a create-stack operation.

1. Provide a stack name and template to the **create-stack** command with the `--disable-rollback` option.

```
aws cloudformation create-stack --stack-name myteststack \
```

```
    --template-body file://template.yaml \
    --disable-rollback
```

The command returns the following output.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

2. Describe the state of the stack using the **describe-stacks** command.

```
aws cloudformation describe-stacks --stack-name myteststack
```

The command returns the following output.

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template",
            "Tags": [],
            "Outputs": [],
            "StackStatusReason": “The following resource(s) failed to create:
 [MyBucket]”,
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "CREATE_FAILED",
            "DisableRollback": true
        }
    ]
}
```

Update stack

To preserve successfully provisioned resources during a stack update operation

1. Provide an existing stack name and template to the **update-stack** command with the `--disable-rollback` option.

```
aws cloudformation update-stack --stack-name myteststack \
    --template-url https://s3.amazonaws.com/amzn-s3-demo-bucket/updated.template
 --disable-rollback
```

The command returns the following output.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

2. Describe the state of the stack using either the **describe-stacks** or **describe-stack-events** command.

```
aws cloudformation describe-stacks --stack-name myteststack
```

The command returns the following output.

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template",
            "Tags": [],
            "Outputs": [],
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "UPDATE_COMPLETE",
            "DisableRollback": true
        }
    ]
}
```

Change set

> **Note**
>
> You can initiate a change set for a stack with a status of CREATE_FAILED or UPDATE_FAILED but not for a status of UPDATE_ROLLBACK_FAILED.

To preserve successfully provisioned resources during a change set operation

Specify the `--disable-rollback` option during an execute-change-set operation.

1. Provide a stack name and template to the **execute-change-set** command with the `--disable-rollback` option.

```
aws cloudformation execute-change-set --stack-name myteststack \
    --change-set-name my-change-set --template-body file://template.yaml
```

The command returns the following output.

```
{
 "Id": "arn:aws:cloudformation:us-east-1:123456789012:changeSet/my-change-set/
bc9555ba-a949-xmpl-bfb8-f41d04ec5784",
 "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

2. Initiate the change set with `--disable-rollback` option.

```
aws cloudformation execute-change-set --stack-name myteststack \
    --change-set-name my-change-set --disable-rollback
```

3. Determine the status of the stack using either the **describe-stacks** or **describe-stack-events** command.

```
aws cloudformation describe-stack-events --stack-name myteststack
```

The command returns the following output.

```
{
```

```
   "StackEvents": [
     {
        "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
        "EventId": "49c966a0-7b74-11ea-8071-024244bb0672",
        "StackName": "myteststack",
        "LogicalResourceId": " MyBucket",
        "PhysicalResourceId": "myteststack-MyBucket-abcdefghijk1",
        "ResourceType": "AWS::S3::Bucket",
        "Timestamp": "2020-04-10T21:43:17.015Z",
        "ResourceStatus": "UPDATE_FAILED"
        "ResourceStatusReason": "User XYZ is not allowed to perform
 S3::UpdateBucket on MyBucket"
     }
}
```

4. Fix permissions errors and retry the operation.

```
aws cloudformation update-stack --stack-name myteststack \
    --use-previous-template --disable-rollback
```

The command returns the following output.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

5. Describe the state of the stack using either the **describe-stacks** or **describe-stack-events** command.

```
aws cloudformation describe-stacks --stack-name myteststack
```

The command returns the following output.

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template",
            "Tags": [],
```

```
            "Outputs": [],
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "UPDATE_COMPLETE",
            "DisableRollback": true
        }
    ]
}
```

Rolling back a stack

You can use the rollback-stack command to roll back a stack with a `CREATE_FAILED` or `UPDATE_FAILED` stack status to its last stable state.

The following **rollback-stack** command rolls back the specified stack.

```
aws cloudformation rollback-stack --stack-name myteststack
```

The command returns the following output.

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

> **Note**
>
> The **rollback-stack** operation will delete a stack if it doesn't contain a last known stable state.

Express mode and rollback

When you use express mode, CloudFormation disables rollback by default. If a resource operation fails, CloudFormation does not automatically roll back previously completed changes. Instead, the stack enters a failed state and you can troubleshoot the issue before retrying.

To re-enable rollback with express mode, set `disableRollback` to `false` in the `--deployment-config` parameter when issuing the stack operation:

```
aws cloudformation create-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --deployment-config '{"mode": "EXPRESS", "disableRollback": false}'
```

> **Note**
>
> Disabling rollback isn't supported for immutable update operations. If an update requires replacing a resource and the operation fails, the failed state can't be preserved for retry.

For more information about express mode, see Express mode.

Roll back your CloudFormation stack on alarm breach with rollback triggers

With rollback triggers, you can have CloudFormation monitor the state of your application during stack creation and updating, and roll back that operation if the application breaches the threshold of the alarms you've specified. For each rollback trigger you create, you specify the CloudWatch alarm that CloudFormation should monitor. CloudFormation monitors the specified alarms during the stack create or update operation, and for the specified amount of time after all resources have been deployed. If any of the alarms goes to `ALARM` state during the stack operation or the monitoring period, CloudFormation rolls back the entire stack operation.

You can set a monitoring time from the default of 0 up to 180 minutes. During this time, CloudFormation monitors all the rollback triggers after the stack creation or update operation deploys all necessary resources. If any of the alarms goes to `ALARM` state during the stack operation or this monitoring period, CloudFormation rolls back the entire stack operation. Then, for update operations, if the monitoring period expires without any alarms going to `ALARM` state, CloudFormation proceeds to dispose of old resources as usual. If you set a monitoring time but don't specify any rollback triggers, CloudFormation still waits the specified period of time before cleaning up old resources for update operations. You can use this monitoring period to perform any manual stack validation desired, and manually cancel the stack creation or update as necessary. If you set a monitoring time of 0 minutes, CloudFormation still monitors the rollback triggers during stack creation and update operations and rolls back the operation if an alarm goes to `ALARM` state. Then, for update operations with no breaching alarms, it begins disposing of old resources immediately once the operation completes.

By default, CloudFormation only rolls back stack operations if an alarm goes to `ALARM` state, not `INSUFFICIENT_DATA` state. To have CloudFormation roll back the stack operation if an alarm goes to `INSUFFICIENT_DATA` state as well, edit the CloudWatch alarm to treat missing data as `breaching`. For more information, see Configuring how CloudWatch alarms treat missing data in the *Amazon CloudWatch User Guide*.

CloudFormation doesn't monitor rollback triggers when it rolls back a stack during an update operation.

You can add a maximum of 5 rollback triggers. To add a rollback trigger, specify the Amazon Resource Name (ARN) of the CloudWatch alarm. Currently, `AWS::CloudWatch::Alarm` and `AWS::CloudWatch::CompositeAlarm` types can be used as rollback triggers. For more information about CloudWatch alarms, see Using CloudWatch alarms in *Amazon CloudWatch User Guide*.

If a given CloudWatch alarm is missing, the entire stack operation fails and rolls back.

Be aware that access to CloudWatch requires credentials. Those credentials must have permissions to access AWS resources, such as retrieving CloudWatch metric data about your resources. For more information, see Authentication and access control for CloudWatch in *Amazon CloudWatch User Guide*.

Add rollback triggers during stack creation or updating

To add rollback triggers during stack creation or update (console)

1. When creating or updating a stack, on the **Configure stack options** page, under **Advanced options**, expand the **Rollback configuration** section.
2. Enter a monitoring time between `0` – `180` minutes. The default value is `0`.
3. Specify the ARN of the CloudWatch alarm or composite alarm you want to use as a rollback trigger, and choose **Add CloudWatch alarm ARN**.

   For example, the following is an ARN for a CloudWatch alarm or composite alarm, `arn:aws:cloudwatch:`*`us-east-1`*`:`*`123456789012`*`:alarm:`*`MyAlarmName`*.
4. Choose **Next** and review the details of your stack.
5. When you're ready, choose **Submit** to create or update the stack.

To add rollback triggers during stack creation or update (AWS CLI)

Use the create-stack or update-stack command with the `--rollback-configuration` option.

For example, the following **update-stack** command sets *MyCompositeAlarm* as a rollback trigger with a 5-minute monitoring period:

```
aws cloudformation update-stack --stack-name MyStack \
  --use-previous-template \
  --rollback-configuration \
  "RollbackTriggers=[{Arn=arn:aws:cloudwatch:us-
east-1:123456789012:alarm:MyCompositeAlarm,Type=AWS::CloudWatch::CompositeAlarm}],MonitoringTim
```

Add rollback triggers to a change set

To add rollback triggers to a change set (console)

1. When creating a change set, on the **Configure stack options** page, under **Advanced options**, expand the **Rollback configuration** section.
2. Enter a monitoring time between `0` – `180` minutes. The default value is 5.
3. Specify the ARN of the CloudWatch alarm or composite alarm you want to use as a rollback trigger, and choose **Add CloudWatch alarm ARN**.

   For example, the following is an ARN for a CloudWatch alarm or composite alarm, `arn:aws:cloudwatch:us-east-1:123456789012:alarm:MyAlarmName`.
4. Choose **Next** and review the details of your change set.
5. When you're ready, choose **Create change set** to create the change set.

To add rollback triggers to a change set (AWS CLI)

Use the create-change-set command with the `--rollback-configuration` option.

View rollback triggers for a stack

To view rollback triggers for a stack, see the **Rollback configuration** section.

1. On the **Stacks** page, choose the stack you want to view from the list on the left.
2. On the **Stack info** tab, expand the **Rollback configuration** section to view the rollback triggers.

View rollback triggers for a change set

To view rollback triggers for a change set, see the **Rollback configuration** section.

1. On the **Stacks** page, choose the stack you want to view from the list on the left.
2. Choose the **Change sets** tab, and then choose the change set you want to view.
3. Choose the **Input** tab, and view the **Rollback configuration** section.

Detect unmanaged configuration changes to stacks and resources with drift detection

Even as you manage your resources through CloudFormation, users can change those resources outside of CloudFormation. Users can edit resources directly by using the underlying service that created the resource. For example, you can use the Amazon EC2 console to update a server instance that was created as part of a CloudFormation stack. Some changes may be accidental, and some may be made intentionally to respond to time-sensitive operational events. Regardless, changes made outside of CloudFormation can complicate stack update or deletion operations. You can use drift detection to identify stack resources to which configuration changes have been made outside of CloudFormation management. You can then take corrective action so that your stack resources are again in sync with their definitions in the stack template, such as updating the drifted resources directly so that they agree with their template definition. Resolving drift helps to ensure configuration consistency and successful stack operations.

Topics

- What is drift?
- Drift detection status codes
- Considerations when detecting drift
- Detect drift on an entire CloudFormation stack
- Detect drift on individual stack resources
- Resolve drift with an import operation

What is drift?

Drift detection enables you to detect whether a stack's actual configuration differs, or has *drifted*, from its expected configuration. Use CloudFormation to detect drift on an entire stack, or on

individual resources within the stack. A resource is considered to have drifted if any of its actual property values differ from the expected property values. This includes if the property or resource has been deleted. A stack is considered to have drifted if one or more of its resources have drifted.

To determine whether a resource has drifted, CloudFormation determines the expected resource property values, as defined in the stack template and any values specified as template parameters. CloudFormation then compares those expected values with the actual values of those resource properties as they currently exist in the stack. A resource is considered to have drifted if one or more of its properties have been deleted, or had their value changed.

CloudFormation generates detailed information on each resource in the stack that has drifted.

CloudFormation detects drift on those AWS resources that support drift detection. Resources that don't support drift detection are assigned a drift status of NOT_CHECKED. For a list of AWS resources that support drift detection, see Resource type support.

In addition, CloudFormation supports drift detection on private resource types that are *provisionable*; that's, whose provisioning type is either `FULLY_MUTABLE` or `IMMUTABLE`. To perform drift detection on a resource of a private resource type, the default version of the resource type that you have registered in your account must be provisionable. For more information on resource provision type, see the `ProvisioningType` parameter of the DescribeType action in the *AWS CloudFormation API Reference* and of the DescribeType command in the *AWS CLI Command Reference*. For more information on private resources, see Managing extensions with the CloudFormation registry.

You can perform drift detection on stacks with the following statuses: `CREATE_COMPLETE`, `UPDATE_COMPLETE`, `UPDATE_ROLLBACK_COMPLETE`, and `UPDATE_ROLLBACK_FAILED`.

When detecting drift on a stack, CloudFormation does not detect drift on any nested stacks that belong to that stack. For more information, see Split a template into reusable pieces using nested stacks. Instead, you can initiate a drift detection operation directly on the nested stack.

> **Note**
>
> CloudFormation only determines drift for property values that are explicitly set, either through the stack template or by specifying template parameters. This doesn't include default values for resource properties. To have CloudFormation track a resource property for purposes of determining drift, explicitly set the property value, even if you are setting it to the default value. CloudFormation also detects drift on stack-level tags.

Drift detection status codes

The tables in this section describe the various status types used with drift detection:

- **Drift detection operation status** describes the current state of the drift operation.
- **Drift status**

  For stack sets, this describes the drift status of the stack set as a whole, based on the drift status of the stack instances that belong to it.

  For stack instances, this describes the drift status of the stack instance, based on the drift status of its associated stack.

  For stacks, this describes the drift status of the stack as a whole, based on the drift status of its resources.
- **Resource drift status** describes the drift status of an individual resource.

The following table lists the status codes CloudFormation assigns to stack drift detection operations.

Drift detection operation status	Description
`DETECTION_COMPLETE`	The stack drift detection operation has successfully completed for all resources in the stack that support drift detection.
`DETECTION_FAILED`	The stack drift detection operation has failed for at least one resource in the stack. Results will be available for resources on which CloudFormation successfully completed drift detection.
`DETECTION_IN_PROGRESS`	The stack drift detection operation is currently in progress.

The following table lists the drift status codes CloudFormation assigns to stacks.

Drift status	Description
DRIFTED	For stacks: The stack differs, or has *drifted*, from its expected template configuration. A stack is considered to have drifted if one or more of its resources have drifted. For stack instances: A stack instance is considered to have drifted if the stack associated with it has drifted. For stack sets: A stack set is considered to have drifted if one or more stack instances has drifted.
NOT_CHECKED	CloudFormation has not checked if the stack, stack set, or stack instance differs from its expected template configura tion.
IN_SYNC	The current configuration of each supported resource matches its expected template configuration. A stack, stack set, or stack instance with no resources that support drift detection will also have a status of IN_SYNC.

The following table lists the drift status codes CloudFormation assigns to stack resources.

Resource drift status	Description
DELETED	The resource differs from its expected template configura tion because the resource has been deleted.
MODIFIED	The resource differs from its expected template configura tion.
NOT_CHECKED	CloudFormation has not checked if the resource differs from its expected template configuration.
IN_SYNC	The resource's current configuration matches its expected template configuration.

The following table lists the difference-type status codes CloudFormation assigns to resource properties that differ from their expected template configuration.

Property difference types	Description
`ADD`	A value has been added to a resource property that's an array or list data type.
`REMOVE`	The property has been removed from the current resource configuration.
`NOT_EQUAL`	The current property value differs from its expected value as defined in the stack template.

Considerations when detecting drift

In order to successfully perform drift detection on a stack, a user must have the following permissions:

- Read permission for each resource that supports drift detection included in the stack. For example, if the stack includes an `AWS::EC2::Instance` resource, you must have `ec2:DescribeInstances` permission to perform drift detection on the stack.
- `cloudformation:DetectStackDrift`
- `cloudformation:DetectStackResourceDrift`
- `cloudformation:BatchDescribeTypeConfigurations`

For more information about setting permissions in CloudFormation, see Control CloudFormation access with AWS Identity and Access Management.

In certain edge cases, CloudFormation may not be able to always return accurate drift results. You should be aware of these edge cases in order to properly interpret your drift detection results.

- In certain cases, objects contained in property arrays will be reported as drift, when in actuality they're default values supplied to the property from the underlying service responsible for the resource.
- Certain resources have attachment relationships with related resources, such that a resource may actually attach or remove property values for another resource, defined in the

same or another template. For example, the `AWS::EC2::SecurityGroupIngress` and `AWS::EC2::SecurityGroupEgress` resources may be used to attach and remove values from `AWS::EC2::SecurityGroup` resources. In these cases, CloudFormation analyses the stack template for attachments before performing the drift comparison. However, CloudFormation can't perform this analysis across stacks, and so may not return accurate drift results where resources that are attached reside in different stacks.

Resources that support drift detection and allow or require attachments from other resources include:

Resource type	Attachment resource type
`AWS::SNS::Topic`	`AWS::SNS::Subscription`
`AWS::IAM::User`	`AWS::IAM::UserToGroupAddition`
`AWS::IAM::Group`	`AWS::IAM::ManagedPolicy`
`AWS::IAM::Role`	`AWS::IAM::Policy`
`AWS::IAM::User`	`AWS::IAM::RolePolicy`
`AWS::ElasticLoadBalancingV2::Listener`	`AWS::ElasticLoadBalancingV2::ListenerCertificate`
`AWS::EC2::SecurityGroup`	`AWS::EC2::SecurityGroupEgress`
	`AWS::EC2::SecurityGroupIngress`

- CloudFormation does not perform drift detection on the `KMSKeyId` property of any resources. Because AWS KMS keys can be referenced by multiple aliases, CloudFormation can't guarantee consistently accurate drift results for this property.
- There are certain resource properties that you can specify in your stack template that, by their very nature, CloudFormation will not be able to compare to the properties in the resulting stack resources. These properties therefore cannot be included in drift detection results. Such properties fall into two broad categories:
  - Property values that CloudFormation cannot map back to their initial resource property value in the stack template.

For example, CloudFormation cannot map the source code of a Lambda function back to the Code property type of the AWS::Lambda::Function resource, and therefore CloudFormation can't include it in drift detection results.

- Property values that the service that is responsible for the resource doesn't return.

  There are certain property values that, by design, are never returned by the service to which the resource belongs. These tend to contain confidential information, such as passwords or other sensitive data that shouldn't be exposed. For example, the IAM service will never return the value of the `Password` property of the AWS::IAM::User LoginProfile property type, and therefore CloudFormation can't include it in drift detection results.

- Objects in an array may be actually service defaults and not drift added manually.

- If you encounter any false positive, send us your comments using the feedback link in the CloudFormation console, or reach out to us through AWS re:Post.
- Some properties can have input values that are equal but not identical. To avoid false positives, you should ensure that your expected configuration matches the actual configuration.
  - For example, the expected configuration of resource property can be 1024 MB and the actual configuration of the same resource property can be 1GB. 1024 MB and 1GB are equal but not identical.

    When drift detection runs on this resource property, it will signal drifted results.

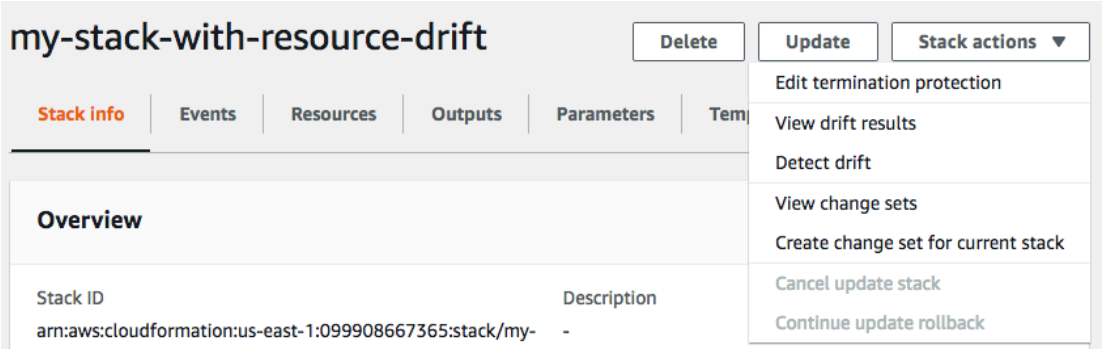
    To avoid this false positive, change the expected configuration of the resource property to 1024MB and then run drift detection.

Detect drift on an entire CloudFormation stack

Performing a drift detection operation on a stack determines whether the stack has drifted from its expected template configuration, and returns detailed information about the drift status of each resource in the stack that supports drift detection.

To detect drift on an entire stack using the AWS Management Console

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the list of stacks, select the stack on which you want to perform drift detection. In the stack details pane, choose **Stack actions**, and then choose **Detect drift**.



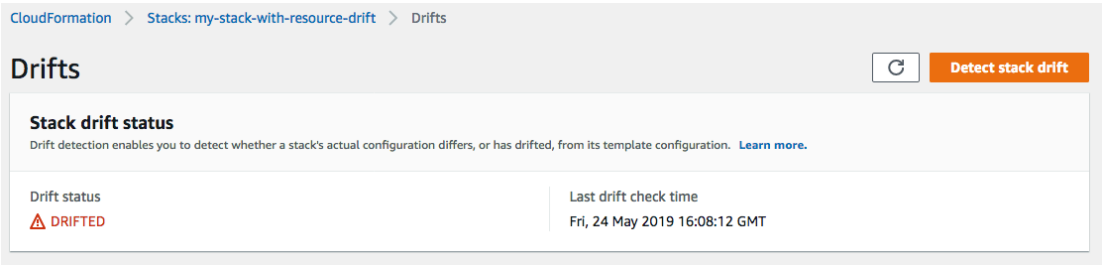
CloudFormation displays an information bar stating that drift detection has been initiated for the selected stack.

3. Wait until CloudFormation completes the drift detection operation. When the drift detection operation completes, CloudFormation updates **Drift status** and **Last drift check time** for your stack. These fields are listed in the **Overview** section of the **Stack info** pane of the stack details page.

   The drift detection operation may take several minutes, depending on the number of resources included in the stack. You can only run a single drift detection operation on a given stack at the same time. CloudFormation continues the drift detection operation even after you dismiss the information bar.

4. Review the drift detection results for the stack and its resources. With your stack selected, from the **Stack actions** menu select **View drift results**.

   CloudFormation lists the overall drift status of the stack, in addition to the last time drift detection was initiated on the stack or any of its individual resources. A stack is considered to have drifted if one or more of its resources have drifted.



In the **Resource drift status** section, CloudFormation lists each stack resource, its drift status, and the last time drift detection was initiated on the resource. The logical ID and physical ID of each resource is displayed to help you identify them. In addition, for resources with a status of **MODIFIED**, CloudFormation displays resource drift details.

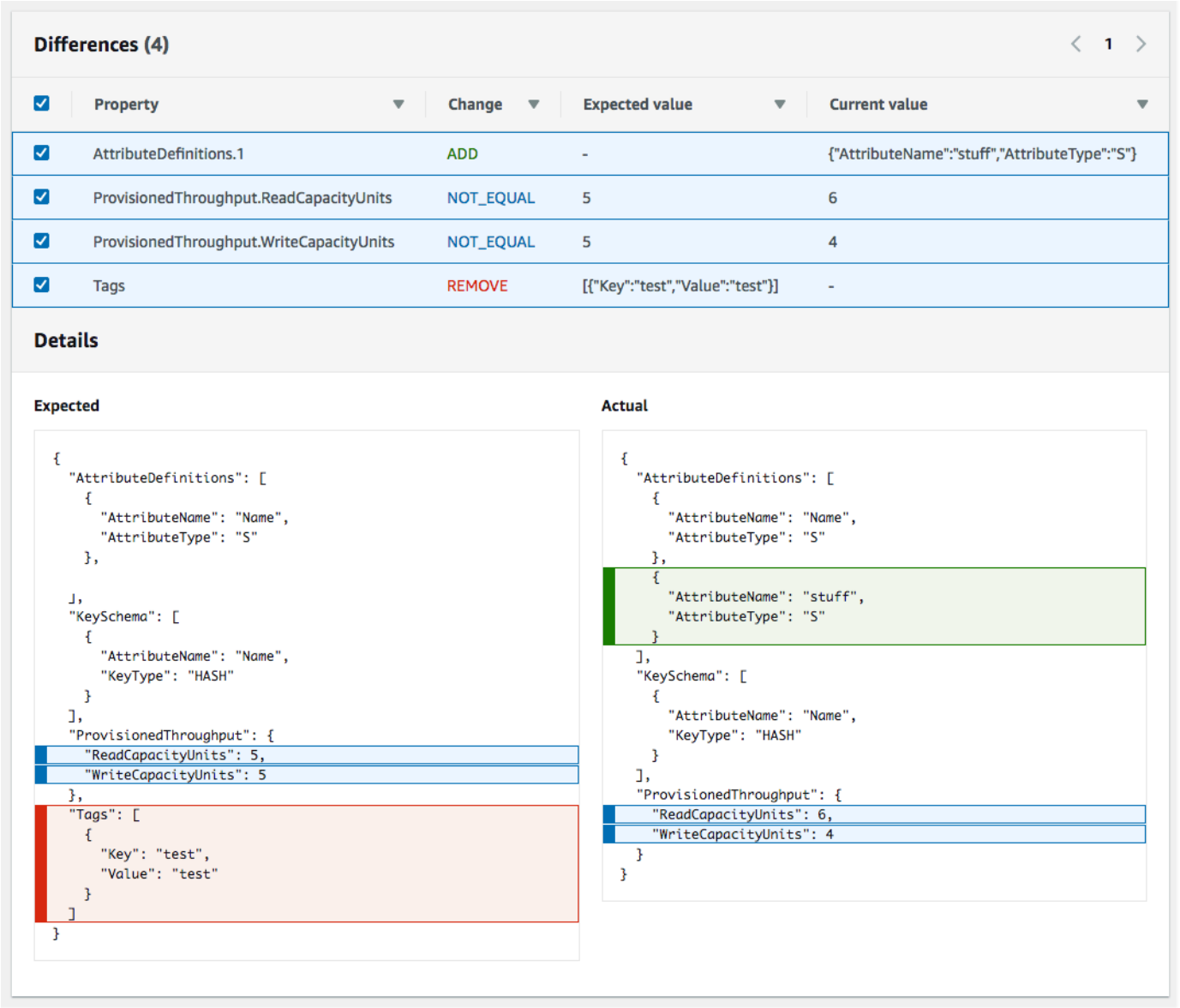
You can sort the resources based on their drift status using the **Drift status** column.

- To view the details on a modified resource.
  - With the modified resource selected, select **View drift details**.

    CloudFormation displays the drift detail page for that resource. This page lists the resource's expected and current property values, and any differences between the two.

    To highlight a difference, in the **Differences** section select the property name.

    - Added properties are highlighted in green in the **Current** column of the **Details** section.
    - Deleted properties are highlighted in red in the **Expected** column of the **Details** section.
    - Properties whose value have been changed are highlighted in yellow in the both **Expected** and **Current** columns.



To detect drift on an entire stack using the AWS CLI

> **Important**
>
> Review the **Last drift check time** for the stack and confirm that it's earlier than the timestamp shown in the resource drift results to prevent the use of stale data.

To detect drift on an entire stack using the AWS CLI, use the following AWS CLI commands:

- **detect-stack-drift** to initiate a drift detection operation on a stack.
- **describe-stack-drift-detection-status** to monitor the status of the stack drift detection operation.
- **describe-stack-resource-drifts** to review the details of the stack drift detection operation.

1. Use the **detect-stack-drift** to detect drift on an entire stack. Specify the stack name or ARN. You can also specify the logical IDs of any specific resources that you want to use as filters for this drift detection operation.

   ```
   aws cloudformation detect-stack-drift --stack-name my-stack-with-resource-drift
   ```

   Output:

   ```
   {
       "StackDriftDetectionId": "624af370-311a-11e8-b6b7-500cexample"
   }
   ```

2. Because stack drift detection operations can be long-running, use **describe-stack-drift-detection-status** to monitor the status of drift operation. This command takes the stack drift detection ID returned by the **detect-stack-drift** command.

   In the example below, we've taken the stack drift detection ID returned by the **detect-stack-drift** example above and passed it as a parameter to **describe-stack-drift-detection-status**. The parameter returns operation details that show that the drift detection operation has completed, a single stack resource has drifted, and that the entire stack is considered to have drifted as a result.

   ```
   aws cloudformation describe-stack-drift-detection-status --stack-drift-detection-
   id 624af370-311a-11e8-b6b7-500cexample
   ```

   Output:

   ```
   {
       "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-with-
   resource-drift/489e5570-df85-11e7-a7d9-50example",
       "StackDriftDetectionId": "624af370-311a-11e8-b6b7-500cexample",
       "StackDriftStatus": "DRIFTED",
       "Timestamp": "2018-03-26T17:23:22.279Z",
   ```

```
    "DetectionStatus": "DETECTION_COMPLETE",
    "DriftedStackResourceCount": 1
}
```

3. When the stack drift detection operation is complete, use the **describe-stack-resource-drifts** command to review the results, including actual and expected property values for resources that have drifted.

   The example below uses the `--stack-resource-drift-status-filters` option to request stack drift information for those resources that have been modified or deleted. The request returns information on the one resource that has been modified, including details about two of its properties whose values have been changed. No resources have been deleted.

   ```
   aws cloudformation describe-stack-resource-drifts --stack-name my-stack-with-resource-drift --stack-resource-drift-status-filters MODIFIED DELETED
   ```

   Output:

   ```
   {
       "StackResourceDrifts": [
           {
               "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
               "ActualProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,
   \"DelaySeconds\":120,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",
   \"maxReceiveCount\":12},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize
   \":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-Queue-494PBHCO76H4\"}",
               "ResourceType": "AWS::SQS::Queue",
               "Timestamp": "2018-03-26T17:23:34.489Z",
               "PhysicalResourceId": "https://sqs.us-east-1.amazonaws.com/099908667365/my-stack-with-resource-drift-Queue-494PBHCO76H4",
               "StackResourceDriftStatus": "MODIFIED",
               "ExpectedProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,
   \"DelaySeconds\":20,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",
   \"maxReceiveCount\":10},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize
   \":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-Queue-494PBHCO76H4\"}",
               "PropertyDifferences": [
   ```

```
                {
                    "PropertyPath": "/DelaySeconds",
                    "ActualValue": "120",
                    "ExpectedValue": "20",
                    "DifferenceType": "NOT_EQUAL"
                },
                {
                    "PropertyPath": "/RedrivePolicy/maxReceiveCount",
                    "ActualValue": "12",
                    "ExpectedValue": "10",
                    "DifferenceType": "NOT_EQUAL"
                }
            ],
            "LogicalResourceId": "Queue"
        }
    ]
}
```

Detect drift on individual stack resources

You can detect drift on specific resources within a stack, rather than the entire stack. This is especially useful when you only need to determine if specific resources now match their expected template configurations again.

When performing drift detection on a resource, CloudFormation also updates the overall stack drift status and the **Last drift check time**, if applicable. For example, suppose a stack has a drift status of `IN_SYNC`. You have CloudFormation perform drift detection on one or more resources contained in that stack, and CloudFormation detects that one or more of those resources has drifted. CloudFormation updates the stack drift status to `DRIFTED`. Conversely, suppose you have a stack with a drift status of `DRIFTED` because of a single drifted resource. If you set that resource back to its expected property values, and then detect drift on the resource again, CloudFormation will update both resource drift status and stack drift status to `IN_SYNC` without requiring you to detect drift on the entire stack again.

To detect drift on an individual resource using the AWS Management Console

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the list of stacks, select the stack that contains the resource. CloudFormation displays the stack details for that stack.

3. In the left navigation pane, under **Stacks**, choose **Stack actions**, and then choose **Detect drift**.
4. Under **Resource drift status**, choose the resource and then select **Detect drift for resource**.

   CloudFormation performs drift detection on the selected resource. If successful, CloudFormation updates the resource's drift status, and the overall stack drift status, if necessary. CloudFormation also updates time stamp for when drift detection was last performed on the resource, and the stack as a whole. If the resource has been modified, CloudFormation displays detailed drift information about the expected and current property values of the resource.
5. Review the drift detection results for the resource.

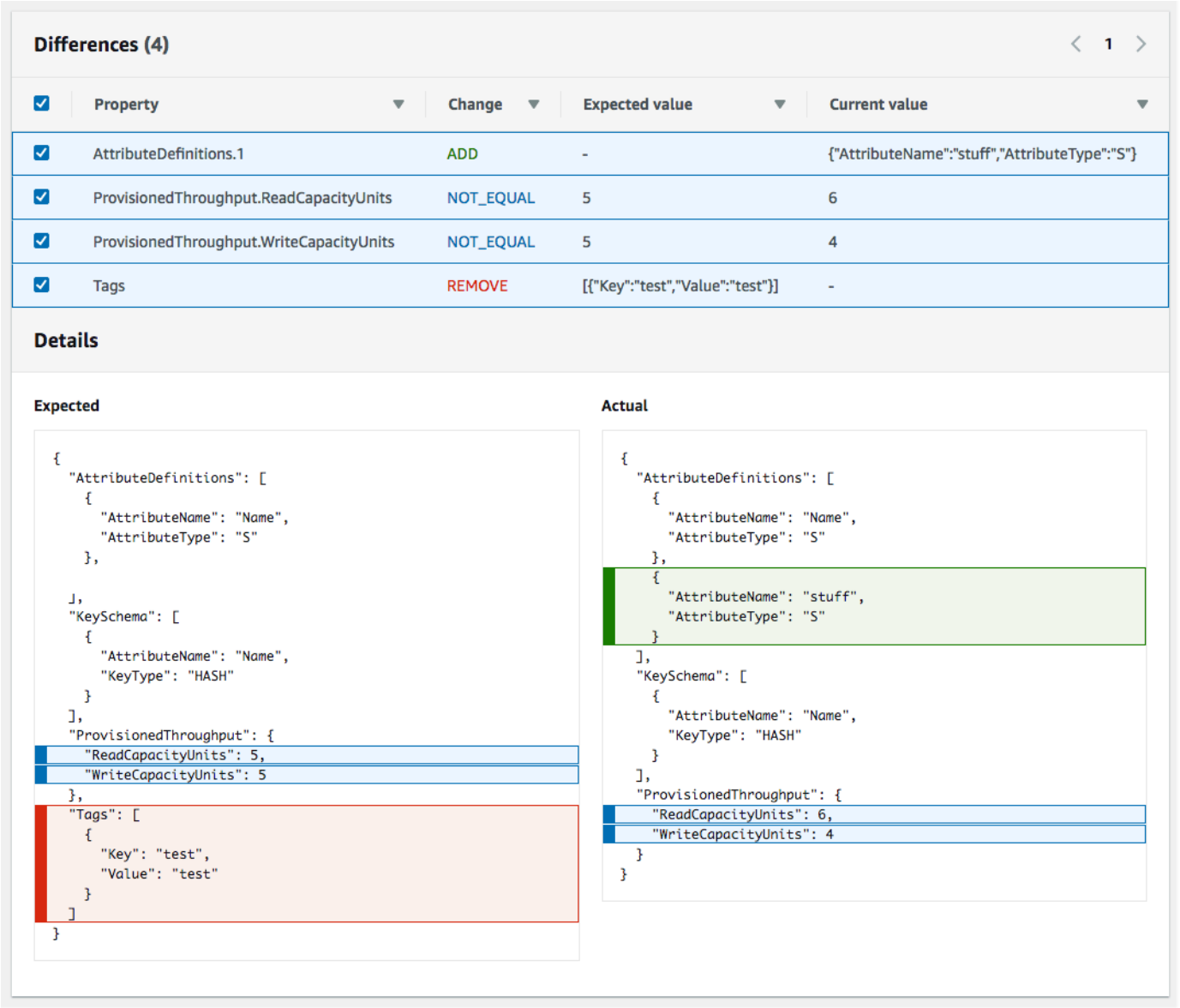
   - To view the details on a modified resource.

     - With the modified resource selected, select **View drift details**.

       CloudFormation displays the drift details for that resource, including the resource's expected and current property values, and any differences between the two.

       To highlight a difference, in the **Differences** section select the property name.

       - Added properties are highlighted in green in the **Current** column of the **Details** section.
       - Deleted properties are highlighted in red in the **Expected** column of the **Details** section.
       - Properties whose value have been changed are highlighted in yellow in the both **Expected** and **Current** columns.



To detect drift on an individual resource using the AWS CLI

- **Important**

  Review the **Last drift check time** for the stack resource and confirm that it's earlier than the timestamp shown in the resource drift results to prevent the use of stale data.

To detect drift on an individual resource using the AWS CLI, use the **detect-stack-resource-drift** command. Specify the logical ID of the resource and the stack in which it's contained.

The following example runs a drift detection operation on a specific stack resources, `my-drifted-resource`. The response returns information that confirms the resource has been modified, including details about two of its properties whose values have been changed.

```
aws cloudformation detect-stack-resource-drift \
    --stack-name my-stack-with-resource-drift \
    --logical-resource-id my-drifted-resource
```

Output:

```
{
    "StackResourceDrift": {
        "StackId": "arn:aws:cloudformation:us-east-1:099908667365:stack/my-stack-
with-resource-drift/489e5570-df85-11e7-a7d9-50example",
        "ActualProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,\"DelaySeconds
\":120,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",
\"maxReceiveCount\":12},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize
\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-
Queue-494PBHCO76H4\"}",
        "ResourceType": "AWS::SQS::Queue",
        "Timestamp": "2018-03-26T18:54:28.462Z",
        "PhysicalResourceId": "https://sqs.us-east-1.amazonaws.com/099908667365/my-
stack-with-resource-drift-Queue-494PBHCO76H4",
        "StackResourceDriftStatus": "MODIFIED",
        "ExpectedProperties": "{\"ReceiveMessageWaitTimeSeconds\":0,
\"DelaySeconds\":20,\"RedrivePolicy\":{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:099908667365:my-stack-with-resource-drift-DLQ-1BCY7HHD5QIM3\",
\"maxReceiveCount\":10},\"MessageRetentionPeriod\":345600,\"MaximumMessageSize
\":262144,\"VisibilityTimeout\":60,\"QueueName\":\"my-stack-with-resource-drift-
Queue-494PBHCO76H4\"}",
        "PropertyDifferences": [
            {
                "PropertyPath": "/DelaySeconds",
                "ActualValue": "120",
                "ExpectedValue": "20",
                "DifferenceType": "NOT_EQUAL"
```

```
            },
            {
                "PropertyPath": "/RedrivePolicy/maxReceiveCount",
                "ActualValue": "12",
                "ExpectedValue": "10",
                "DifferenceType": "NOT_EQUAL"
            }
        ],
        "LogicalResourceId": "my-drifted-resource"
    }
}
```

Resolve drift with an import operation

There may be cases where a resource's configuration has drifted from its intended configuration and you want to accept the new configuration as the intended configuration. In most cases, you would resolve the drift results by updating the resource definition in the stack template with a new configuration and then perform a stack update. However, if the new configuration updates a resource property that requires replacement, then the resource will be recreated during the stack update. If you want to retain the existing resource, you can use the resource import feature to update the resource and resolve the drift results without causing the resource to be replaced.

Resolving drift for a resource through an import operation consists of the following basic steps:

- Add a DeletionPolicy attribute, set to Retain, to the resource. This ensures the existing resource is retained rather than deleted when it's removed from the stack.
- Remove the resource from the template and run a stack update operation. This removes the resource from the stack, but doesn't delete it.
- Describe the resource's actual state in the stack template, and then import the existing resource back into the stack. This adds the resource back into the stack and resolves the property differences that were causing the drift results.

For more information on resource import, see Import AWS resources into a CloudFormation stack manually. For a list of resources that support import, see Resource type support.

In this example, we use the following template, named `templateToImport.json`.

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
         "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "BillingMode": "PROVISIONED",
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
```

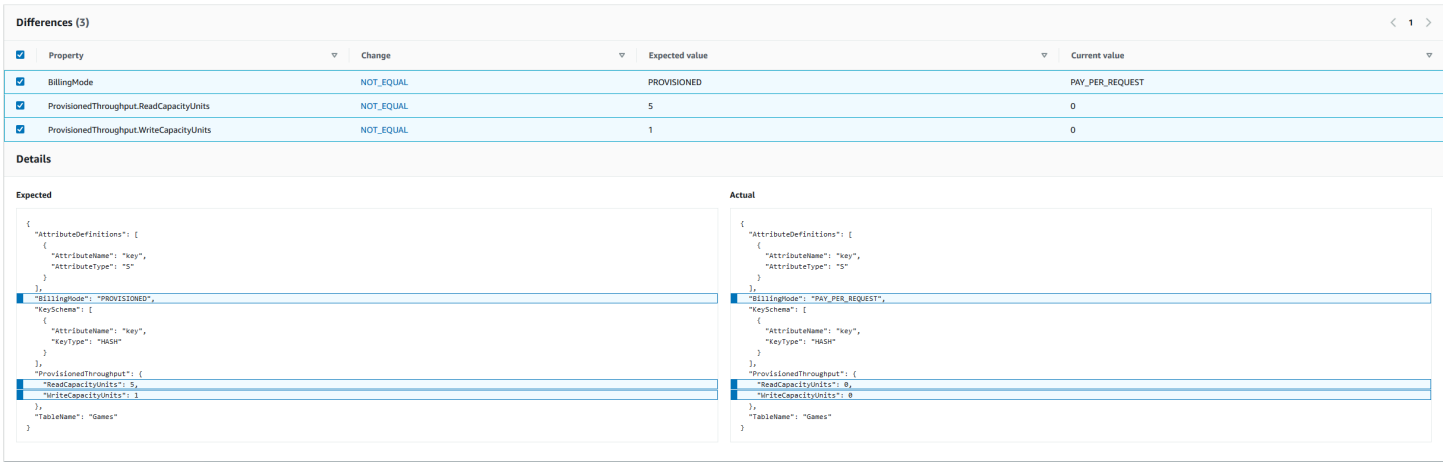
```
                ],
                "BillingMode": "PROVISIONED",
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  GamesTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Games
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
```

```
        WriteCapacityUnits: 1
```

In this example, let's assume a user changed a resource outside of CloudFormation. After running drift detect, we discovered that `GamesTable` has been modified `BillingMode` to PAY_PER_REQUEST. For more information about drift detect, see Detect unmanaged configuration changes to stacks and resources with drift detection.



Our stack is now out of date, our resources are live, but we want to preserve the intended resource configuration. We can do this by resolving drift through an import operation, without interrupting services.

Resolve drift with an import operation using the CloudFormation console

Step 1. Update stack with Retain deletion policy

To update stack using a `DeletionPolicy` attribute with the `Retain` option

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, choose the stack that has drifted.
3. Choose **Update**, and then choose **Replace current template** from the stack details pane.
4. On the **Specify template** page, provide your updated template that contains the `DeletionPolicy` attribute with the `Retain` option using one of the following methods:
   - Choose **Amazon S3 URL**, and then specify the URL for your template in the text box.
   - Choose **Upload a template file**, and then browse for your template.

Then, choose **Next**.

5. Review the **Specify stack details** page and choose **Next**.
6. Review the **Configure stack options** page and choose **Next**.
7. On the **Review *stack-name*** page, choose **Update stack**.

Results: On the **Events** page of your stack, the status is `UPDATE_COMPLETE`.

To resolve drift through an import operation, without interrupting services, specify a `Retain` DeletionPolicy for the resources you want to remove from your stack. In the following example, we've added a DeletionPolicy attribute, set to `Retain`, to the `GamesTable` resource.

Example JSON

```
  "GamesTable": {
      "Type": "AWS::DynamoDB::Table",
      "DeletionPolicy": "Retain",
      "Properties": {
          "TableName": "Games",
```

Example YAML

```
GamesTable:
  Type: AWS::DynamoDB::Table
  DeletionPolicy: Retain
  Properties:
    TableName: Games
```

Step 2. Remove drifted resources, related parameters, and outputs

To remove drifted resources, related parameters, and outputs

1. Choose **Update**, and then choose **Replace current template** from the stack details pane.
2. On the **Specify template** page, provide your updated template with its resources, related parameters, and outputs removed from the stack template using one of the following methods:

   - Choose **Amazon S3 URL**, and then specify the URL for your template in the text box.

- Choose **Upload a template file**, and then browse for your template.

Then, choose **Next**.

3. Review the **Specify stack details** page and choose **Next**.
4. Review the **Configure stack options** page and choose **Next**.
5. On the **Review *stack-name*** page, choose **Update stack**.

Results: The **Logical ID** `GamesTable` has a status of `DELETE_SKIPPED` on the **Events** page of your stack.

Wait until CloudFormation completes the stack update operation. After the stack update operation completes, remove the resource, related parameters, and outputs from the stack template. Then, import the updated template. After completing these actions, the example template now looks like the following.

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
         "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "BillingMode": "PROVISIONED",
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
```

```
                "WriteCapacityUnits":1
             }
          }
       }
    }
}
```

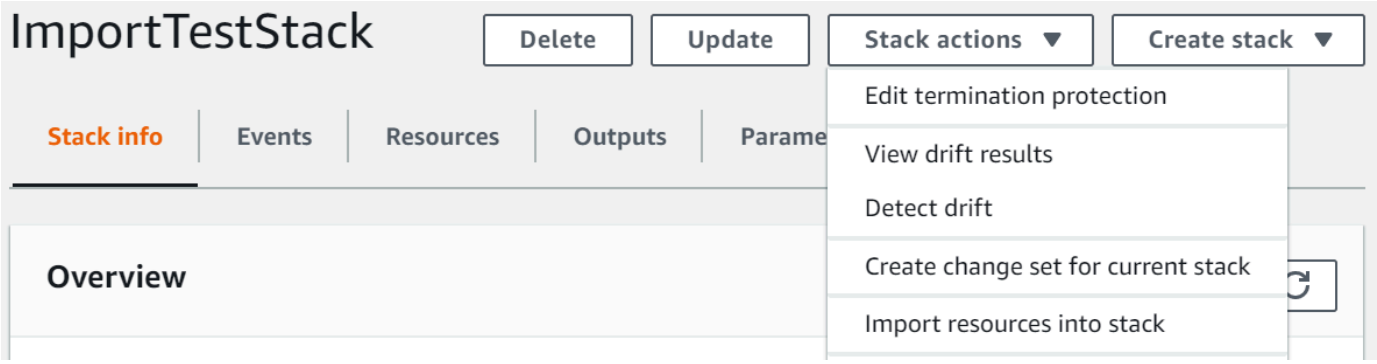
Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
```

Step 3. Update template to match the live state of your resources

To update template to match the live state of resources

1. To import the updated template, choose **Stack actions** and then choose **Import resources into stack**.



2. Review the **Import overview** page for a list of things you're required to provide during this operation, and then choose **Next**.
3. On the **Specify template** page, provide your updated template using one of the following methods:

   - Choose **Amazon S3 URL**, and then specify the URL for your template in the text box.
   - Choose **Upload a template file**, and then browse for your template.

   Then, choose **Next**.
4. On the **Identify resources** page, identify each target resource. For more information, see Resource identifiers.

   a. Under **Identifier property**, choose the type of resource identifier. For example, the `TableName` property identifies the `AWS::DynamoDB::Table` resource.

   b. Under **Identifier value**, enter the actual property value. In the example template, the `TableName` for the `GamesTable` resource is `Games`.

   c. Choose **Next**.
5. Review the **Specify stack details** page, and choose **Next**.
6. On the **Import overview** page, review the resources being imported, and then choose **Import resources**. This will import the `AWS::DynamoDB::Table` resource type back into your stack.

Results: In this example, we resolved the resource drift through an import operation, without interrupting services. You can check the progress of an import action in the CloudFormation console in the Events tab. Imported resources will have a `IMPORT_COMPLETE` status followed by a `CREATE_COMPLETE` status with **Resource import complete** as the status reason.

Wait until CloudFormation completes the stack update operation. After the stack update operation completes, update your template to match the actual, drifted state of your resources. For example, the `BillingMode` will be set to `PAY_PER_REQUEST` and `ReadCapacityUnits` and `WriteCapacityUnits` will be set to `0`.

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
         "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "BillingMode": "PROVISIONED",
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
```

```
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "BillingMode": "PAY_PER_REQUEST",
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 0,
                    "WriteCapacityUnits": 0
                }
            }
        }
    }
}
```

Example YAML

```
AWSTemplateFormatVersion: 2010-09-09
Description: Import test
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      BillingMode: PROVISIONED
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  GamesTable:
    Type: AWS::DynamoDB::Table
    DeletionPolicy: Retain
    Properties:
      TableName: Games
      AttributeDefinitions:
```

```
      - AttributeName: key
        AttributeType: S
    KeySchema:
      - AttributeName: key
        KeyType: HASH
    BillingMode: PAY_PER_REQUEST
    ProvisionedThroughput:
      ReadCapacityUnits: 0
      WriteCapacityUnits: 0
```

Import AWS resources into a CloudFormation stack

You can import existing resources into a CloudFormation stack. This is useful if you want to start using CloudFormation to manage resources that were created outside of CloudFormation, without having to delete and recreate them.

CloudFormation offers the following options for importing existing resources into a stack:

- IaC generator is a tool that automatically scans your existing resources and generates a CloudFormation template based on their current state. This template can then be used to import those resources into a stack.
- Resource import is a manual process where you describe the existing resources in your CloudFormation template and then import them into a stack. This approach requires you to manually specify the resource properties and configurations in the template.
- Auto-import is an automatic process where you describe existing resources in your CloudFormation template and CloudFormation imports the ones with matching custom names into a stack.
- Stack refactoring is a feature that simplifies reorganizing the resources in your CloudFormation stacks while still preserving the existing resource properties and data. With stack refactoring, you can move resources between stacks, split monolithic stacks into smaller components, or consolidate multiple stacks into one.

In addition to bringing existing resources under CloudFormation management, the resource import feature can be useful in the following scenarios:

- **Moving resources between stacks** – You can import resources from one stack into another, allowing you to reorganize your infrastructure as needed.

- **Nesting existing stacks** – You can import an existing stack as a nested stack within another stack, enabling modular and reusable infrastructure designs.

CloudFormation supports importing a wide range of resources. For more information, see Resource type support.

Topics

- Import AWS resources into a CloudFormation stack manually
- Import AWS resources into a CloudFormation stack automatically
- Reverting an import operation

Import AWS resources into a CloudFormation stack manually

With resource import, you can import existing AWS resources into a new or existing CloudFormation stack. During an import operation, you create a change set that imports your existing resources into a stack or creates a new stack from your existing resources. You provide the following during import.

- A template that describes the entire stack, including both the original stack resources and the resources you're importing. Each resource to import must have a DeletionPolicy attribute.
- Identifiers for the resources you're importing that CloudFormation can use to map the logical IDs in the template with the existing resources.

> **Note**
>
> CloudFormation only supports one level of nesting using resource import. This means that you can't import a stack into a child stack or import a stack that has children.

Topics

- Resource identifiers
- Resource import validation
- Resource import status codes
- Considerations during an import operation
- Additional resources

- Creating a stack from existing resources
- Importing existing resources into a stack
- Moving resources between stacks
- Nesting an existing stack

Resource identifiers

You provide two values to identify each resource you're importing.

- An identifier property. This is a resource property that can be used to identify each resource type. For example, an `AWS::S3::Bucket` resource can be identified using its `BucketName`.

  The resource property that you use to identify the resource you're importing varies with the resource type. You can find the resource property in the CloudFormation console. After you create a template that includes the resource to import, you can initiate the import process, where you'll find the identifier properties for the resources you're importing. For some resource types, there might be multiple ways to identify them, and you can select which property to use in the drop-down lists.

  Alternatively, you can get the identifier properties for the resources you're importing by calling the get-template-summary CLI command and specifying the S3 URL of the stack template as the value for the `--template-url` option.
- An identifier value. This is the resource's actual property value. For example, the actual value for the `BucketName` property might be `MyS3Bucket`.

  You can get the value of the identifier property from the service console for the resource.

Resource import validation

During an import operation, CloudFormation performs the following validations.

- The resource to import exists.
- The properties and configuration values for each resource to import adhere to the resource type schema, which defines its accepted properties, required properties, and supported property values.
- The required properties are specified in the template. Required properties for each resource type are described in the AWS resource and property types reference.

- The resource to import doesn't belong to another stack in the same Region. This check applies per Region, but some resource types aren't Region-specific, such as IAM roles, Route 53 hosted zones, or CloudFront distributions. For these types, make sure that the resource isn't already managed by a stack in another Region before importing it.

CloudFormation doesn't check that the template configuration matches the actual configuration of resource properties.

> **Important**
>
> Verify that resources and their properties defined in the template match the intended configuration of the resource import to avoid unexpected changes.

Resource import status codes

This table describes the various status types used with the resource import feature.

Import operation status	Description
`IMPORT_IN_PROGRESS`	The import operation is in progress.
`IMPORT_COMPLETE`	The import operation completed for all resources in the stack.
`IMPORT_ROLLBACK_IN_PROGRESS`	The rollback import operation is rolling back the previous template configuration.
`IMPORT_ROLLBACK_FAILED`	The import rollback operation failed.
`IMPORT_ROLLBACK_COMPLETE`	The import rolled back to the previous template configuration.

Considerations during an import operation

- After the import is complete and before performing subsequent stack operations, we recommend running drift detection on imported resources. Drift detection ensures that the

template configuration matches the actual configuration. For more information, see Detect drift on an entire CloudFormation stack.

- Import operations don't allow new resource creations, resource deletions, or changes to property configurations.
- Each resource to import must have a `DeletionPolicy` attribute for the import operation to succeed. The `DeletionPolicy` can be set to any possible value. Only the resources you're importing need a `DeletionPolicy`. Resources that are already part of the stack don't need a `DeletionPolicy`.
- You can't import the same resource into multiple stacks in the same Region.
- You can use the `cloudformation:ImportResourceTypes` IAM policy condition to control which resource types users can work with during an import operation. For more information, see Policy condition keys for CloudFormation.
- The CloudFormation stack limits apply when importing resources. For more information on limits, see Understand CloudFormation quotas.

Additional resources

To resolve stack drift with a resource import, see Resolve drift with an import operation.

Creating a stack from existing resources

This topic shows you how to create a stack from existing AWS resources by describing them in a template. To instead scan for existing resources and automatically generate a template that you can use to import existing resources into CloudFormation or replicate resources in a new account, see Generate templates from existing resources with IaC generator.

Prerequisites

Before you begin, you must have the following:

- A template that describes all of the resources that you want in the new stack. Save the template locally or in an Amazon S3 bucket.
- For each resource you want to import, include the following:
  - the properties and property values that define the resource's current configuration.
  - the unique identifier for the resource, such as the resource name. For more information, see Resource identifiers.
  - the DeletionPolicy attribute.

Topics

- Example template
- Create a stack from existing resources using the AWS Management Console
- Create a stack from existing resources using the AWS CLI

Example template

In this walkthrough, we assume you're using the following example template, called `TemplateToImport.json`, that specifies two DynamoDB tables that were created outside of CloudFormation. `ServiceTable` and `GamesTable` are the targets of the import.

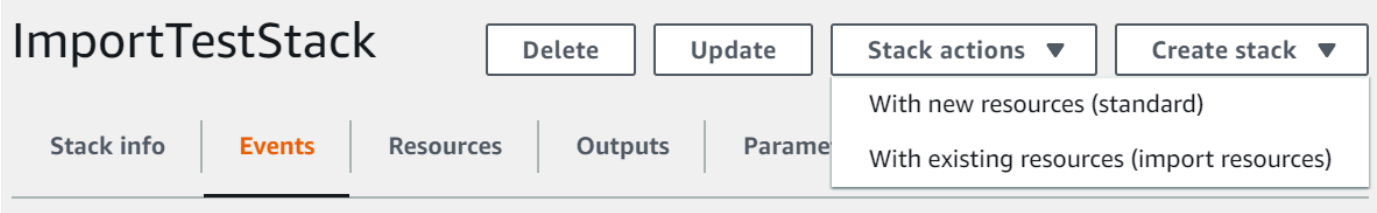
> **Note**
>
> This template is meant as an example only. To use it for your own testing purposes, replace the sample resources with resources from your account.

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Service",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
```

```
                    "WriteCapacityUnits": 1
                }
            }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

Create a stack from existing resources using the AWS Management Console

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, choose **Create stack**, and then choose **With existing resources (import resources)**.



3. Read the **Import overview** page for a list of things you're required to provide during this operation. Then, choose **Next**.
4. On the **Specify template** page, provide your template using one of the following methods, and then choose **Next**.

   - Choose **Amazon S3 URL**, and then specify the URL for your template in the text box.
   - Choose **Upload a template file**, and then browse for your template.
5. On the **Identify resources** page, identify each target resource. For more information, see Resource identifiers.

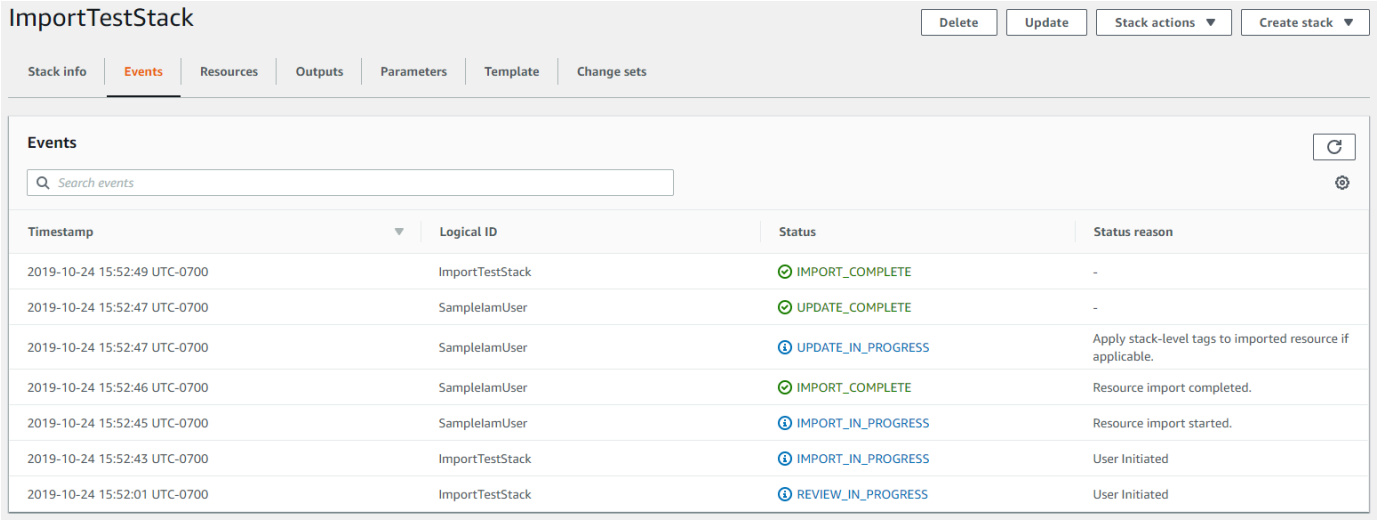
   a. Under **Identifier property**, choose the type of resource identifier. For example, the `AWS::DynamoDB::Table` resource can be identified using the `TableName` property.

   b. Under **Identifier value**, type the actual property value. For example, the `TableName` for the `GamesTable` resource in the example template is *`Games`*.

   c. Choose **Next**.
6. On the **Specify stack details** page, modify any parameters, and then choose **Next**. This automatically creates a change set.

   > **Important**
   >
   > The import operation fails if you modify existing parameters that initiate a create, update, or delete operation.
7. On the **Review *`stack-name`*** page, confirm that the correct resources are being imported, and then choose **Import resources**. This automatically executes the change set created in the last step.

   The **Events** pane of the **Stack details** page for your new stack displays.



8. (Optional) Run drift detection on the stack to make sure the template and actual configuration of the imported resources match. For more information about detecting drift, see Detect drift on an entire CloudFormation stack.
9. (Optional) If your imported resources don't match their expected template configurations, either correct the template configurations or update the resources directly. In this walkthrough, we correct the template configurations to match their actual configurations.
   a. Revert the import operation for the affected resources.
   b. Add the import targets to your template again, making sure that the template configurations match the actual configurations.
   c. Repeat steps 2 – 8 using the modified template to import the resources again.

Create a stack from existing resources using the AWS CLI

1. To learn which properties identify each resource type in the template, run the **get-template-summary** command, specifying the S3 URL of the template. For example, the `AWS::DynamoDB::Table` resource can be identified using the `TableName` property. For the `GamesTable` resource in the example template, the value of `TableName` is `Games`. You'll need this information in the next step.

```
aws cloudformation get-template-summary \
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json
```

   For more information, see Resource identifiers.

2. Compose a list of the actual resources from your template and their unique identifiers in the following JSON string format.

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifie
{"TableName":"Games"}},
{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"ServiceTable","ResourceIdentifi
{"TableName":"Service"}}]
```

   Alternatively, you can specify the JSON-formatted parameters in a configuration file.

   For example, to import `ServiceTable` and `GamesTable`, you might create a *`ResourcesToImport.txt`* file that contains the following configuration.

```
[
   {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"GamesTable",
      "ResourceIdentifier":{
         "TableName":"Games"
      }
   },
   {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"ServiceTable",
      "ResourceIdentifier":{
         "TableName":"Service"
      }
   }
]
```

3. To create a change set, use the following **create-change-set** command and replace the placeholder text. For the `--change-set-type` option, specify a value of **`IMPORT`**. For the `--resources-to-import` option, replace the sample JSON string with the actual JSON string you just created.

```
aws cloudformation create-change-set \
    --stack-name TargetStack --change-set-name ImportChangeSet \
    --change-set-type IMPORT \
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json \
```

```
    --resources-to-
import '[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceI
{"TableName":"Games"}},
{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"ServiceTable","ResourceIdentifi
{"TableName":"Service"}}]'
```

> **Note**
>
> `--resources-to-import` doesn't support inline YAML. The requirements for escaping quotes in the JSON string vary depending on your terminal. For more information, see Using quotation marks inside strings in the *AWS Command Line Interface User Guide*.

Alternatively, you can use a file URL as input for the `--resources-to-import` option, as shown in the following example.

```
--resources-to-import file://ResourcesToImport.txt
```

4. Review the change set to make sure the correct resources will be imported.

```
aws cloudformation describe-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

5. To initiate the change set and import the resources, use the following **execute-change-set** command and replace the placeholder text. On successful completion of the operation (`IMPORT_COMPLETE`), the resources are successfully imported.

```
aws cloudformation execute-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

6. (Optional) Run drift detection on the `IMPORT_COMPLETE` stack to make sure the template and actual configuration of the imported resources match. For more information on detecting drift, see Detect drift on individual stack resources.

   a. Run drift detection on the specified stack.

```
aws cloudformation detect-stack-drift --stack-name TargetStack
```

If successful, this command returns the following sample output.

```
{ "Stack-Drift-Detection-Id" : "624af370-311a-11e8-b6b7-500cexample" }
```

b. View the progress of a drift detection operation for the specified stack drift detection ID.

```
aws cloudformation describe-stack-drift-detection-status \
    --stack-drift-detection-id 624af370-311a-11e8-b6b7-500cexample
```

c. View drift information for the resources that have been checked for drift in the specified stack.

```
aws cloudformation describe-stack-resource-drifts --stack-name TargetStack
```

7. (Optional) If your imported resources don't match their expected template configurations, either correct the template configurations or update the resources directly. In this walkthrough, we correct the template configurations to match their actual configurations.

   a. Revert the import operation for the affected resources.

   b. Add the import targets to your template again, making sure that the template configurations match the actual configurations.

   c. Repeat steps 3 – 6 using the modified template to import the resources again.

Importing existing resources into a stack

This topic shows you how to import existing AWS resources into an existing stack by describing them in a template. To instead scan for existing resources and automatically generate a template that you can use to import existing resources into CloudFormation or replicate resources in a new account, see Generate templates from existing resources with IaC generator.

Prerequisites

Before you begin, you must have the following:

- A template that describes the entire stack, including both the resources that are already part of the stack and the resources to import. Save the template locally or in an Amazon S3 bucket.

  To get a copy of a running stack's template

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. From the list of stacks, choose the stack you want to retrieve the template from.
3. In the stack details pane, choose the **Template** tab, and then choose **Copy to clipboard**.
4. Paste the code into a text editor to begin adding other resources to the template.

- For each resource you want to import, include the following:
  - the properties and property values that define the resource's current configuration.
  - the unique identifier for the resource, such as the resource name. For more information, see Resource identifiers.
  - the DeletionPolicy attribute.

Topics

- Example template
- Import an existing resource into a stack using the AWS Management Console
- Import an existing resource into a stack using the AWS CLI

Example template

In this walkthrough, we assume you're using the following example template, called `TemplateToImport.json`, that specifies two DynamoDB tables. `ServiceTable` is currently part of the stack, and `GamesTable` is the table you want to import.

> **Note**
>
> This template is meant as an example only. To use it for your own testing purposes, replace the sample resources with resources from your account.

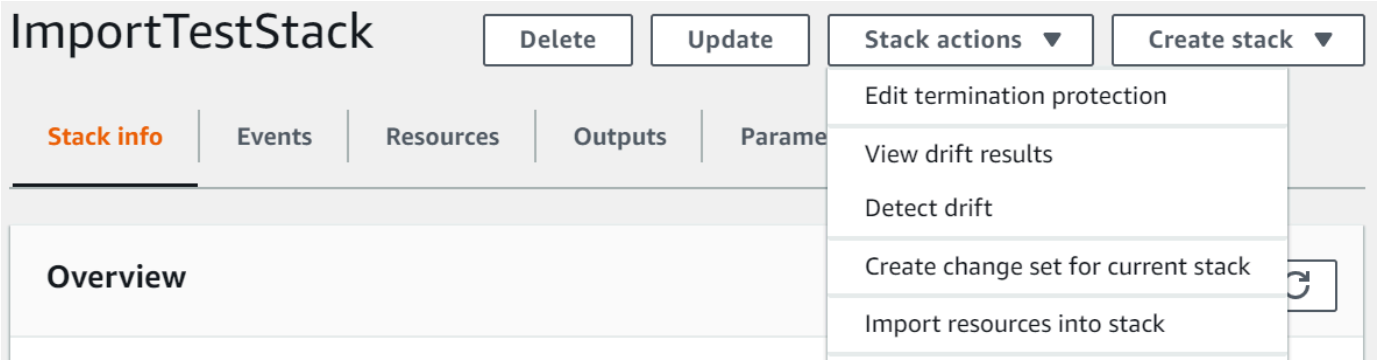
```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "TableName": "Service",
```

```
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

Import an existing resource into a stack using the AWS Management Console

> **Note**
>
> The CloudFormation console doesn't support the use of the intrinsic function `Fn::Transform` when importing resources. You can use the AWS CLI to import resources that use the `Fn::Transform` function.

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **Stacks** page, choose the stack you want to import resources into.
3. Choose **Stack actions**, and then choose **Import resources into stack**.



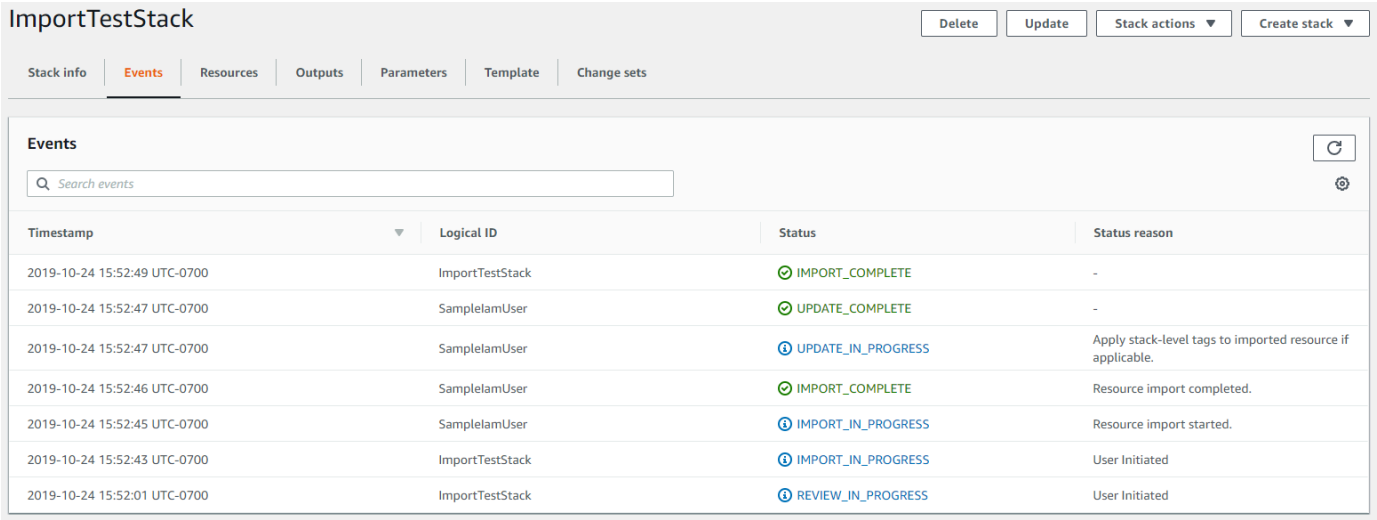
4. Review the **Import overview** page, and then choose **Next**.
5. On the **Specify template** page, provide your updated template using one of the following methods, and then choose **Next**.
   - Choose **Amazon S3 URL**, and then specify the URL for your template in the text box.
   - Choose **Upload a template file**, and then browse for your template.
6. On the **Identify resources** page, identify each target resource. For more information, see Resource identifiers.
   a. Under **Identifier property**, choose the type of resource identifier. For example, the `AWS::DynamoDB::Table` resource can be identified using the `TableName` property.
   b. Under **Identifier value**, type the actual property value. For example, the `TableName` for the `GamesTable` resource in the example template is *`Games`*.
   c. Choose **Next**.

7. On the **Specify stack details** page, update any parameters, and then choose **Next**. This automatically creates a change set.

   > **Note**
   >
   > The import operation fails if you modify existing parameters that initiate a create, update, or delete operation.

8. On the **Review *stack-name*** page, review the resources to import, and then choose **Import resources**. This automatically executes the change set created in the last step. Any stack-level tags are applied to imported resources at this time. For more information, see Configure stack options.

   The **Events** page for the stack displays.



9. (Optional) Run drift detection on the stack to make sure the template and actual configuration of the imported resources match. For more information about detecting drift, see Detect drift on an entire CloudFormation stack.
10. (Optional) If your imported resources don't match their expected template configurations, either correct the template configurations or update the resources directly. For more information about importing drifted resources, see Resolve drift with an import operation.

Import an existing resource into a stack using the AWS CLI

1. To learn which properties identify each resource type in the template, run the **get-template-summary** command, specifying the S3 URL of the template. For example, the `AWS::DynamoDB::Table` resource can be identified using the `TableName` property. For the

`GamesTable` resource in the example template, the value of `TableName` is `Games`. You'll need this information in the next step.

```
aws cloudformation get-template-summary \
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json
```

For more information, see Resource identifiers.

2. Compose a list of actual resources to import and their unique identifiers in the following JSON string format.

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifie
{"TableName":"Games"}}]
```

Alternatively, you can specify the JSON-formatted parameters in a configuration file.

For example, to import `GamesTable` , you might create a *`ResourcesToImport.txt`* file that contains the following configuration.

```
[
  {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"GamesTable",
      "ResourceIdentifier": {
        "TableName":"Games"
      }
  }
]
```

3. To create a change set, use the following **create-change-set** command and replace the placeholder text. For the `--change-set-type` option, specify a value of **`IMPORT`**. For the `--resources-to-import` option, replace the sample JSON string with the actual JSON string you just created.

```
aws cloudformation create-change-set \
    --stack-name TargetStack --change-set-name ImportChangeSet \
    --change-set-type IMPORT \
    --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/
TemplateToImport.json \
```

```
    --resources-to-
import '[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceI
{"TableName":"Games"}}]'
```

> **Note**
>
> `--resources-to-import` doesn't support inline YAML. The requirements for escaping quotes in the JSON string vary depending on your terminal. For more information, see Using quotation marks inside strings in the *AWS Command Line Interface User Guide*.

Alternatively, you can use a file URL as input for the `--resources-to-import` option, as shown in the following example.

```
--resources-to-import file://ResourcesToImport.txt
```

4. Review the change set to make sure the correct resources will be imported.

```
aws cloudformation describe-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

5. To initiate the change set and import the resources, use the following **execute-change-set** command and replace the placeholder text. Any stack-level tags are applied to imported resources at this time. For more information, see Configure stack options. On successful completion of the operation (`IMPORT_COMPLETE`), the resources are successfully imported.

```
aws cloudformation execute-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStack
```

6. (Optional) Run drift detection on the `IMPORT_COMPLETE` stack to make sure the template and actual configuration of the imported resources match. For more information about detecting drift, see Detect drift on an entire CloudFormation stack.

   a. Run drift detection on the specified stack.

   ```
   aws cloudformation detect-stack-drift --stack-name TargetStack
   ```

   If successful, this command returns the following sample output.

```
{ "Stack-Drift-Detection-Id" : "624af370-311a-11e8-b6b7-500cexample" }
```

b. View the progress of a drift detection operation for the specified stack drift detection ID.

```
aws cloudformation describe-stack-drift-detection-status \
    --stack-drift-detection-id 624af370-311a-11e8-b6b7-500cexample
```

c. View drift information for the resources that have been checked for drift in the specified stack.

```
aws cloudformation describe-stack-resource-drifts --stack-name TargetStack
```

7. (Optional) If your imported resources don't match their expected template configurations, either correct the template configurations or update the resources directly. For more information about importing drifted resources, see Resolve drift with an import operation.

Moving resources between stacks

Using the `resource import` feature, you can move resources between, or *refactor*, stacks. You need to first add a `Retain` deletion policy to the resource you want to move to ensure that the resource is preserved when you remove it from the source stack and import it to the target stack.

If you're new to importing, we recommend that you first review the introductory information in the Import AWS resources into a CloudFormation stack topic.

> **Important**
>
> Not all resources support import operations. See Resources that support import operations before you remove a resource from your stack. If you remove a resource that doesn't support import operations from your stack, you can't import the resource into another stack or bring it back into the source stack.

Refactor a stack using the AWS Management Console

1. In the source template, specify a `Retain` DeletionPolicy for the resource you want to move.

   In the following example source template, `Games` is the target of this refactor.

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
```

```
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

2. Open the CloudFormation console to perform a stack update to apply the deletion policy.

   a. On the **Stacks** page, with the stack selected, choose **Update**.

   b. Under **Prepare template**, choose **Replace current template**.

   c. Under **Specify template**, provide the updated source template with the `DeletionPolicy` attribute on `GamesTable`, and then choose **Next**.

      - Choose **Amazon S3 URL**, and then specify the URL to the updated source template in the text box.
      - Choose **Upload a template file**, and then browse for the updated source template file.

   d. On the **Specify stack details** page, no changes are required. Choose **Next**.

   e. On the **Configure stack options** page, no changes are required. Choose **Next**.

   f. On the **Review *`SourceStackName`*** page, review your changes. If your template contains IAM resources, select **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information about using IAM resources in templates, see Control CloudFormation access with AWS Identity and Access Management. Then, either update your source stack by creating a change set or update your source stack directly.

3. Remove the resource, related parameters, and outputs from the source template, and then add them to the target template.

   The source template now looks like the following.

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
```

```
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

The following example target template currently has the `PlayersTable` resource, and now also contains `GamesTable`.

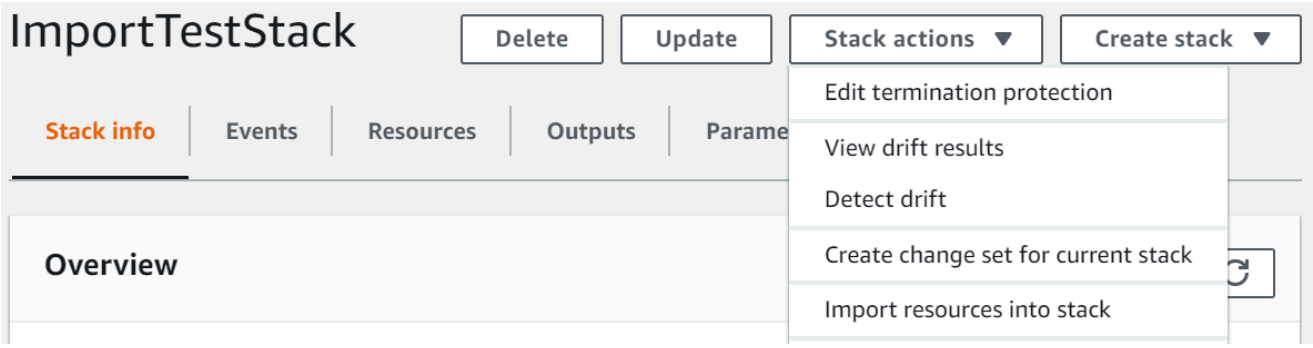
Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "PlayersTable": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "TableName": "Players",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
```

```
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

4. Repeat steps 2 – 3 to update the source stack again, this time to delete the target resource from the stack.
5. Perform an import operation to add `GamesTable` to the target stack.

a. On the **Stacks** page, with the parent stack selected, choose **Stack actions**, and then choose **Import resources into stack**.



b. Read the **Import overview** page for a list of things you're required to provide during this operation. Then, choose **Next**.

c. On the **Specify template** page, complete one of the following, and then choose **Next**.

- Choose **Amazon S3 URL**, and then specify a URL in the text box.
- Choose **Upload a template file**, and then browse for a file to upload.

d. On the **Identify resources** page, identify the resource you're moving (in this example, `GamesTable`). For more information, see Resource identifiers.

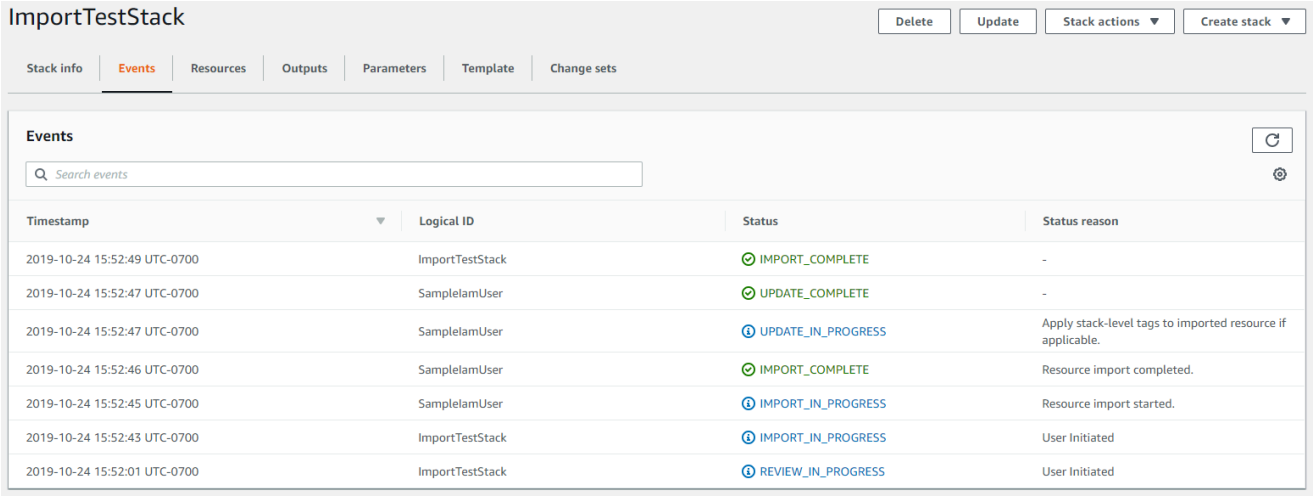
1. Under **Identifier property**, choose the type of resource identifier. For example, an `AWS::DynamoDB::Table` resource can be identified using the `TableName` property.
2. Under **Identifier value**, type the actual property value. For example, *`GamesTables`*.
3. Choose **Next**.

e. On the **Specify stack details** page, modify any parameters, and then choose **Next**. This automatically creates a change set.

> **Important**
>
> The import operation fails if you modify existing parameters that initiate a create, update, or delete operation.

f. On the **Review *`TargetStackName`*** page, confirm that the correct resource is being imported, and then choose **Import resources**. This automatically initiates the change set created in the last step. Any stack-level tags are applied to imported resources at this time.

g. The **Events** pane of the **Stack details** page for your parent stack displays.



> **Note**
>
> It's not necessary to run drift detection on the parent stack after this import operation because the `AWS::CloudFormation::Stack` resource is already managed by CloudFormation.

Refactor a stack using the AWS CLI

1. In the source template, specify a `Retain` DeletionPolicy for the resource you want to move.

   In the following example source template, `GamesTable` is the target of this refactor.

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
```

```
            ],
            "KeySchema":[
               {
                  "AttributeName":"key",
                  "KeyType":"HASH"
               }
            ],
            "ProvisionedThroughput":{
               "ReadCapacityUnits":5,
               "WriteCapacityUnits":1
            }
         }
      },
      "GamesTable": {
          "Type": "AWS::DynamoDB::Table",
          "DeletionPolicy": "Retain",
          "Properties": {
              "TableName": "Games",
              "AttributeDefinitions": [
                  {
                      "AttributeName": "key",
                      "AttributeType": "S"
                  }
              ],
              "KeySchema": [
                  {
                      "AttributeName": "key",
                      "KeyType": "HASH"
                  }
              ],
              "ProvisionedThroughput": {
                  "ReadCapacityUnits": 5,
                  "WriteCapacityUnits": 1
              }
          }
      }
   }
}
```

2. Update the source stack to apply the deletion policy to the resource.

```
aws cloudformation update-stack --stack-name SourceStackName
```

3. Remove the resource, related parameters, and outputs from the source template, and then add them to the target template.

   The source template now looks like the following.

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

   The following example target template currently has the `PlayersTable` resource, and now also contains `GamesTable`.

   Example JSON

```
{
```

```
"AWSTemplateFormatVersion": "2010-09-09",
"Description": "Import test",
"Resources": {
    "PlayersTable": {
        "Type": "AWS::DynamoDB::Table",
        "Properties": {
            "TableName": "Players",
            "AttributeDefinitions": [
                {
                    "AttributeName": "key",
                    "AttributeType": "S"
                }
            ],
            "KeySchema": [
                {
                    "AttributeName": "key",
                    "KeyType": "HASH"
                }
            ],
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 5,
                "WriteCapacityUnits": 1
            }
        }
    },
    "GamesTable": {
        "Type": "AWS::DynamoDB::Table",
        "DeletionPolicy": "Retain",
        "Properties": {
            "TableName": "Games",
            "AttributeDefinitions": [
                {
                    "AttributeName": "key",
                    "AttributeType": "S"
                }
            ],
            "KeySchema": [
                {
                    "AttributeName": "key",
                    "KeyType": "HASH"
                }
            ],
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 5,
```

```
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

4. Update the source stack to delete the `GamesTable` resource and its related parameters and outputs from the stack.

```
aws cloudformation update-stack --stack-name SourceStackName
```

5. Compose a list of actual resources to import and their unique identifiers in the following JSON string format. For more information, see Resource identifiers.

```
[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdentifie
{"TableName":"Games"}}]
```

Alternatively, you can specify the JSON-formatted parameters in a configuration file.

For example, to import `GamesTable`, you might create a *`ResourcesToImport.txt`* file that contains the following configuration.

```
[
  {
      "ResourceType":"AWS::DynamoDB::Table",
      "LogicalResourceId":"GamesTable",
      "ResourceIdentifier": {
        "TableName":"Games"
      }
  }
]
```

6. To create a change set, use the following **create-change-set** command and replace the placeholder text. For the `--change-set-type` option, specify a value of **`IMPORT`**. For the `--resources-to-import` option, replace the sample JSON string with the actual JSON string you just created.

```
aws cloudformation create-change-set \
    --stack-name TargetStackName --change-set-name ImportChangeSet \
    --change-set-type IMPORT \
```

```
    --template-body file://TemplateToImport.json \
    --resources-to-import
 "'[{"ResourceType":"AWS::DynamoDB::Table","LogicalResourceId":"GamesTable","ResourceIdenti
{"TableName":"Games"}}]'"
```

> **Note**
>
> `--resources-to-import` doesn't support inline YAML. The requirements for escaping quotes in the JSON string vary depending on your terminal. For more information, see Using quotation marks inside strings in the *AWS Command Line Interface User Guide*.

Alternatively, you can use a file URL as input for the `--resources-to-import` option, as shown in the following example.

```
--resources-to-import file://ResourcesToImport.txt
```

7. Review the change set to make sure the correct resource is being imported into the target stack.

```
aws cloudformation describe-change-set \
    --change-set-name ImportChangeSet
```

8. To initiate the change set and import the resource, use the following **execute-change-set** command and replace the placeholder text. Any stack-level tags are applied to imported resources at this time. On successful completion of the operation (`IMPORT_COMPLETE`), the resource is successfully imported.

```
aws cloudformation execute-change-set \
    --change-set-name ImportChangeSet --stack-name TargetStackName
```

> **Note**
>
> It's not necessary to run drift detection on the target stack after this import operation because the resource is already managed by CloudFormation.

Nesting an existing stack

Use the `resource import` feature to nest an existing stack within another existing stack. Nested stacks are common components that you declare and reference from within other templates. That way, you can avoid copying and pasting the same configurations into your templates and simplify stack updates. If you have a template for a common component, you can use the `AWS::CloudFormation::Stack` resource to reference this template from within another template. For more information on nested stacks, see Split a template into reusable pieces using nested stacks.

CloudFormation only supports one level of nesting using `resource import`. This means that you can't import a stack into a child stack or import a stack that has children.

If you're new to importing, we recommend that you first review the introductory information in the Import AWS resources into a CloudFormation stack manually topic.

Nested stack import validation

During a nested stack import operation, CloudFormation performs the following validations.

- The nested `AWS::CloudFormation::Stack` definition in the parent stack template matches the actual nested stack's template.
- The tags for the nested `AWS::CloudFormation::Stack` definition in the parent stack template match the tags for the actual nested stack resource.

Nest an existing stack using the AWS Management Console

1. Add the `AWS::CloudFormation::Stack` resource to the parent stack template with a `Retain` DeletionPolicy. In the following example parent stack template, `MyNestedStack` is the target of the import.

   JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
```

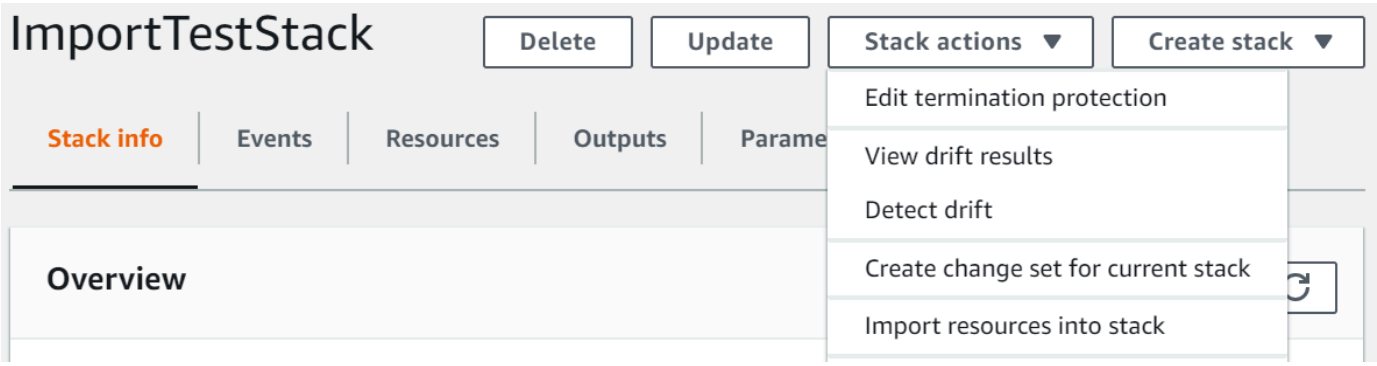
```
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
    "MyNestedStack" : {
      "Type" : "AWS::CloudFormation::Stack",
      "DeletionPolicy": "Retain",
      "Properties" : {
      "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template",
        "Parameters" : {
          "InstanceType" : "t1.micro",
          "KeyName" : "mykey"
        }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
```

```
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  MyNestedStack:
    Type: AWS::CloudFormation::Stack
    DeletionPolicy: Retain
    Properties:
      TemplateURL: >-
        https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template
      Parameters:
        InstanceType: t1.micro
        KeyName: mykey
```

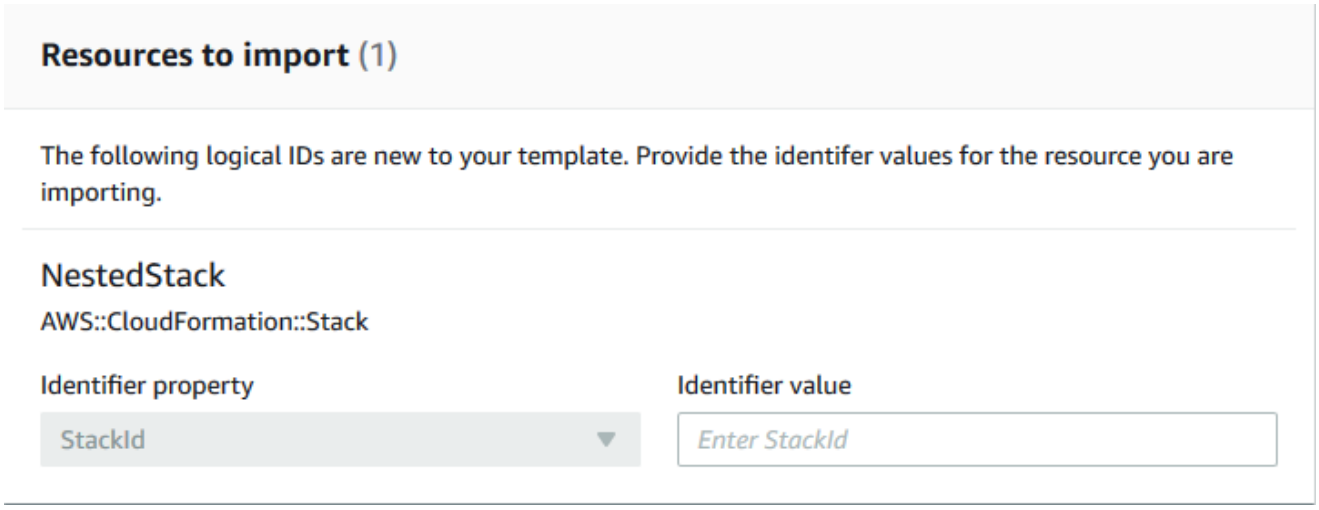
2. Open the CloudFormation console.
3. On the **Stacks** page, with the parent stack selected, choose **Stack actions**, and then choose **Import resources into stack**.



4. Read the **Import overview** page for a list of things you're required to provide during this operation. Then, choose **Next**.
5. On the **Specify template** page, provide the updated parent template using one of the following methods, and then choose **Next**.
   - Choose **Amazon S3 URL**, and then specify the URL for your template in the text box.
   - Choose **Upload a template file**, and then browse for your template.
6. On the **Identify resources** page, identify the `AWS::CloudFormation::Stack` resource.

a. Under **Identifier property**, choose the type of resource identifier. For example, an `AWS::CloudFormation::Stack` resource can be identified using the `StackId` property.

b. Under **Identifier value**, type the ARN of the stack you're importing. For example, *`arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10`*.



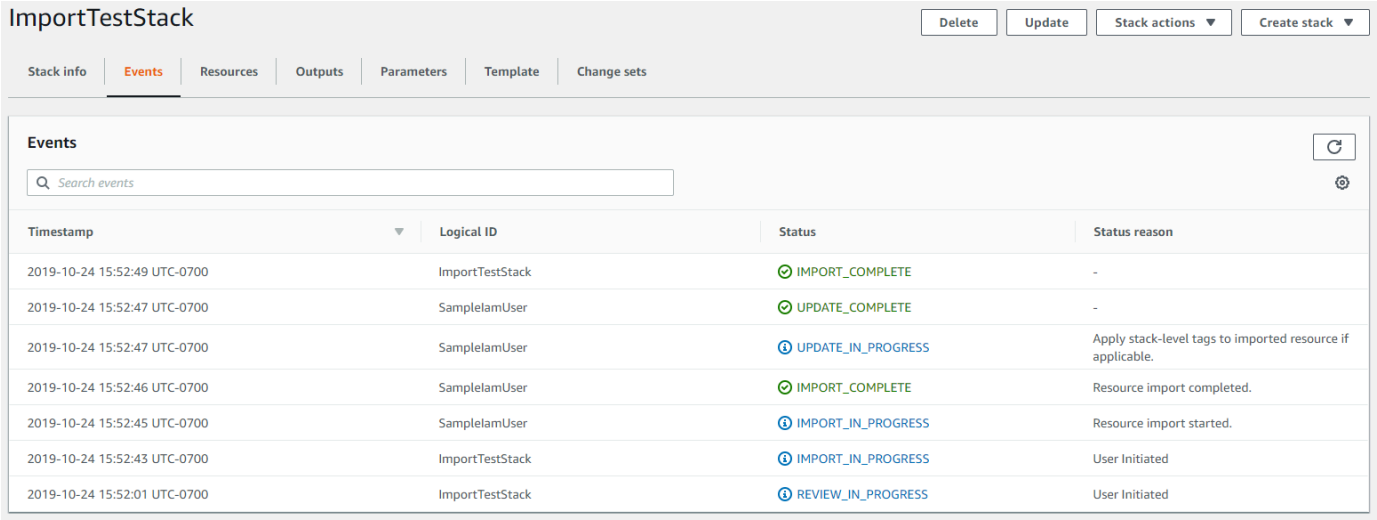
c. Choose **Next**.

7. On the **Specify stack details** page, modify any parameters, and then choose **Next**. This automatically creates a change set.

> **Important**
>
> The import operation fails if you modify existing parameters that initiate a create, update, or delete operation.

8. On the **Review *`MyParentStack`*** page, confirm that the correct resource is being imported, and then choose **Import resources**. This automatically executes the change set created in the last step. Any stack-level tags are applied to imported resources at this time.

9. The **Events** pane of the **Stack details** page for your parent stack displays.



> **Note**
>
> It's not necessary to run drift detection on the parent stack after this import operation because the `AWS::CloudFormation::Stack` resource was already managed by CloudFormation.

Nest an existing stack using the AWS CLI

1. Add the `AWS::CloudFormation::Stack` resource to the parent stack template with a `Retain` DeletionPolicy. In the following example parent template, `MyNestedStack` is the target of the import.

   JSON

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
```

```
            ],
            "KeySchema":[
               {
                  "AttributeName":"key",
                  "KeyType":"HASH"
               }
            ],
            "ProvisionedThroughput":{
               "ReadCapacityUnits":5,
               "WriteCapacityUnits":1
            }
         }
      },
    "MyNestedStack" : {
      "Type" : "AWS::CloudFormation::Stack",
      "DeletionPolicy": "Retain",
      "Properties" : {
      "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template",
        "Parameters" : {
          "InstanceType" : "t1.micro",
          "KeyName" : "mykey"
        }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ServiceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: Service
      AttributeDefinitions:
        - AttributeName: key
          AttributeType: S
      KeySchema:
        - AttributeName: key
          KeyType: HASH
      ProvisionedThroughput:
```

```
        ReadCapacityUnits: 5
        WriteCapacityUnits: 1
  MyNestedStack:
    Type: AWS::CloudFormation::Stack
    DeletionPolicy: Retain
    Properties:
      TemplateURL: >-
        https://s3.amazonaws.com/cloudformation-templates-us-east-2/
EC2ChooseAMI.template
      Parameters:
        InstanceType: t1.micro
        KeyName: mykey
```

2. Compose a JSON string as shown in the following example, with these modifications:

   - Replace *MyNestedStack* with the logical ID of the target resource as specified in the template.
   - Replace *arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10* with the ARN of the stack you want to import.

```
[{"ResourceType":"AWS::CloudFormation::Stack","LogicalResourceId":"MyNestedStack","Resource
{"StackId":"arn:aws:cloudformation:us-east-2:123456789012:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10"}}]
```

Alternatively, you can specify the parameters in a configuration file.

For example, to import MyNestedStack, you might create a *ResourcesToImport.txt* file that contains the following configuration.

JSON

```
[
  {
      "ResourceType":"AWS::CloudFormation::Stack",
      "LogicalResourceId":"MyNestedStack",
      "ResourceIdentifier": {
        "StackId":"arn:aws:cloudformation:us-west-2:12345678910:stack/
mystack/5b918d10-cd98-11ea-90d5-0a9cd3354c10"
      }
  }
```

```
]
```

YAML

```
ResourceType: AWS::CloudFormation::Stack
  LogicalResourceId: MyNestedStack
  ResourceIdentifier:
    StackId: >-
      arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10
```

3. To create a change set, use the following **create-change-set** command and replace the placeholder text. For the `--change-set-type` option, specify a value of **`IMPORT`**. For the `--resources-to-import` option, replace the sample JSON string with the actual JSON string you just created.

```
aws cloudformation create-change-set \
    --stack-name MyParentStack --change-set-name ImportChangeSet \
    --change-set-type IMPORT \
    --template-body file://TemplateToImport.json \
    --resources-to-
import '[{"ResourceType":"AWS::CloudFormation::Stack","LogicalResourceId":"MyNestedStack","
{"StackId":"arn:aws:cloudformation:us-west-2:12345678910:stack/mystack/5b918d10-
cd98-11ea-90d5-0a9cd3354c10"}}]'
```

> **Note**
>
> `--resources-to-import` doesn't support inline YAML. The requirements for escaping quotes in the JSON string vary depending on your terminal. For more information, see Using quotation marks inside strings in the *AWS Command Line Interface User Guide*.

Alternatively, you can use a file URL as input for the `--resources-to-import` option, as shown in the following example.

```
--resources-to-import file://ResourcesToImport.txt
```

If successful, this command returns the following sample output.

```
{
    "Id": "arn:aws:cloudformation:us-west-2:12345678910:changeSet/
ImportChangeSet/8ad75b3f-665f-46f6-a200-0b4727a9442e",
    "StackId": "arn:aws:cloudformation:us-west-2:12345678910:stack/
MyParentStack/4e345b70-1281-11ef-b027-027366d8e82b"
}
```

4. Review the change set to make sure the correct stack is being imported.

```
aws cloudformation describe-change-set --change-set-name ImportChangeSet
```

5. To initiate the change set and import the stack into the source parent stack, use the following **execute-change-set** command and replace the placeholder text. Any stack-level tags are applied to imported resources at this time. On successful completion of the import operation (`IMPORT_COMPLETE`), the stack is successfully nested.

```
aws cloudformation execute-change-set --change-set-name ImportChangeSet
```

> **Note**
>
> It's not necessary to run drift detection on the parent stack after this import operation because the `AWS::CloudFormation::Stack` resource is already managed by CloudFormation.

Import AWS resources into a CloudFormation stack automatically

You can now import *named resources* automatically when creating or updating CloudFormation stacks. A *named resource* is one with a custom name. For more information, see Name type in the *CloudFormation Template Reference*.

When you initiate auto-import, CloudFormation checks for existing resources that match your template and imports them during deployment. For nested stacks, create the change set from the root stack.

After the import is complete and before performing subsequent stack operations, we recommend running drift detection on imported resources. Drift detection ensures that the template

configuration matches the actual configuration. For more information, see Detect drift on an entire CloudFormation stack.

To import a resource, they need to meet the following requirements:

- The resource must have a static custom name defined in your template. Dynamic names (using ! Ref or other functions) are not currently supported.
- The resource must have a `DeletionPolicy` of `Retain` or `RetainExceptOnCreate`.
- The resource must not already belong to another CloudFormation stack.
- The resource type must support CloudFormation import operations. For more information, see Resource type support.
- The primary ID or an additional identifier for the resource type must be in the template. Primary IDs or additional identifiers with read only properties aren't supported. To find out what the primary ID or additional identifiers are for a type, look for the `primaryIdentifier` and `additionalIdentifiers` properties in the resource schema. For more information on these, see resource type definition schema.

Example auto-import

The following example uses a change set, `CreateChangeSet` to create a stack called `my-stack` based on a template file, `template.yaml`, and imports matching resources automatically.

```
aws cloudformation create-change-set \
  --stack-name my-stack \
  --change-set-name CreateChangeSet \
  --change-set-type CREATE \
  --template-body file://template.yaml \
  --import-existing-resources
```

Troubleshooting

If auto-import fails, do the following to troubleshoot:

- Verify the resource name in your template matches the name of the resource exactly
- Verify that the resource isn't already managed by another stack
- Make sure the resource type supports auto import
- Verify your template includes all the required properties for the resource type

Supported resource types

The following table lists AWS resource types that currently support auto-import.

Resource
AWS::ACMPCA::CertificateAuthorityActivation
AWS::ACMPCA::Permission
AWS::APS::ResourcePolicy
AWS::ARCZonalShift::ZonalAutoshiftConfiguration
AWS::ApiGateway::BasePathMapping
AWS::ApiGateway::DocumentationVersion
AWS::ApiGateway::DomainName
AWS::ApiGateway::Method
AWS::ApiGateway::Model
AWS::ApiGateway::Stage
AWS::ApiGatewayV2::DomainName
AWS::ApiGatewayV2::Stage
AWS::AppFlow::Connector
AWS::AppFlow::ConnectorProfile
AWS::AppFlow::Flow
AWS::AppIntegrations::EventIntegration
AWS::AppStream::AppBlockBuilder
AWS::AppStream::ApplicationEntitlementAssociation

Resource
AWS::AppStream::ApplicationFleetAssociation
AWS::AppStream::DirectoryConfig
AWS::AppStream::Entitlement
AWS::AppStream::ImageBuilder
AWS::AppStream::Stack
AWS::AppStream::StackFleetAssociation
AWS::AppStream::StackUserAssociation
AWS::AppStream::User
AWS::AppSync::DomainName
AWS::AppSync::SourceApiAssociation
AWS::ApplicationAutoScaling::ScalableTarget
AWS::Athena::CapacityReservation
AWS::Athena::DataCatalog
AWS::Athena::PreparedStatement
AWS::Athena::WorkGroup
AWS::AutoScaling::AutoScalingGroup
AWS::AutoScaling::LaunchConfiguration
AWS::AutoScaling::LifecycleHook
AWS::AutoScaling::WarmPool
AWS::Backup::BackupVault

Resource
AWS::Backup::LogicallyAirGappedBackupVault
AWS::Backup::RestoreTestingPlan
AWS::Backup::RestoreTestingSelection
AWS::Backup::TieringConfiguration
AWS::Batch::ComputeEnvironment
AWS::Batch::JobDefinition
AWS::Batch::QuotaShare
AWS::Batch::ServiceEnvironment
AWS::Bedrock::ResourcePolicy
AWS::BedrockAgentCore::ResourcePolicy
AWS::BedrockAgentCore::WorkloadIdentity
AWS::CUR::ReportDefinition
AWS::Cases::Field
AWS::Cassandra::Keyspace
AWS::Cassandra::Table
AWS::Cassandra::Type
AWS::CloudFormation::ModuleDefaultVersion
AWS::CloudFormation::Stack
AWS::CloudFormation::StackSet
AWS::CloudFront::ConnectionFunction

Resource
AWS::CloudFront::KeyValueStore
AWS::CloudFront::MonitoringSubscription
AWS::CloudFront::TrustStore
AWS::CloudTrail::ResourcePolicy
AWS::CloudTrail::Trail
AWS::CloudWatch::Alarm
AWS::CloudWatch::CompositeAlarm
AWS::CloudWatch::Dashboard
AWS::CloudWatch::LogAlarm
AWS::CloudWatch::MetricStream
AWS::CodeDeploy::Application
AWS::CodeDeploy::DeploymentConfig
AWS::CodeDeploy::DeploymentGroup
AWS::CodeGuruProfiler::ProfilingGroup
AWS::CodePipeline::CustomActionType
AWS::CodePipeline::Pipeline
AWS::CodeStarConnections::SyncConfiguration
AWS::Cognito::IdentityPoolPrincipalTag
AWS::Cognito::IdentityPoolRoleAttachment
AWS::Cognito::UserPoolDomain

Resource
AWS::Cognito::UserPoolGroup
AWS::Cognito::UserPoolIdentityProvider
AWS::Cognito::UserPoolResourceServer
AWS::Cognito::UserPoolRiskConfigurationAttachment
AWS::Cognito::UserPoolUICustomizationAttachment
AWS::Cognito::UserPoolUser
AWS::Cognito::UserPoolUserToGroupAttachment
AWS::Config::AggregationAuthorization
AWS::Config::ConfigRule
AWS::Config::ConfigurationAggregator
AWS::Config::ConformancePack
AWS::Config::OrganizationConformancePack
AWS::Config::StoredQuery
AWS::Connect::ApprovedOrigin
AWS::Connect::DataTableAttribute
AWS::Connect::IntegrationAssociation
AWS::Connect::PredefinedAttribute
AWS::ControlTower::EnabledControl
AWS::CustomerProfiles::CalculatedAttributeDefinition
AWS::CustomerProfiles::Domain

Resource
AWS::CustomerProfiles::EventStream
AWS::CustomerProfiles::EventTrigger
AWS::CustomerProfiles::Integration
AWS::CustomerProfiles::ObjectType
AWS::CustomerProfiles::Recommender
AWS::CustomerProfiles::SegmentDefinition
AWS::DMS::DataMigration
AWS::DMS::DataProvider
AWS::DMS::InstanceProfile
AWS::DMS::MigrationProject
AWS::DMS::ReplicationConfig
AWS::DataBrew::Dataset
AWS::DataBrew::Job
AWS::DataBrew::Project
AWS::DataBrew::Recipe
AWS::DataBrew::Ruleset
AWS::DataBrew::Schedule
AWS::DataZone::Domain
AWS::DataZone::Environment
AWS::DataZone::EnvironmentActions

Resource
AWS::DataZone::EnvironmentBlueprintConfiguration
AWS::DataZone::EnvironmentProfile
AWS::DataZone::FormType
AWS::DataZone::Project
AWS::Deadline::QueueFleetAssociation
AWS::Deadline::QueueLimitAssociation
AWS::Detective::MemberInvitation
AWS::Detective::OrganizationAdmin
AWS::DevOpsAgent::PrivateConnection
AWS::DocDB::GlobalCluster
AWS::DynamoDB::GlobalTable
AWS::DynamoDB::Table
AWS::EC2::EnclaveCertificateIamRoleAssociation
AWS::EC2::GatewayRouteTableAssociation
AWS::EC2::KeyPair
AWS::EC2::LocalGatewayRoute
AWS::EC2::NetworkInterfaceAttachment
AWS::EC2::NetworkPerformanceMetricSubscription
AWS::EC2::RouteServerAssociation
AWS::EC2::RouteServerPropagation

Resource
AWS::EC2::SecurityGroupVpcAssociation
AWS::EC2::SqlHaStandbyDetectedInstance
AWS::EC2::TransitGatewayMeteringPolicyEntry
AWS::EC2::TransitGatewayMulticastDomainAssociation
AWS::EC2::TransitGatewayMulticastGroupMember
AWS::EC2::TransitGatewayMulticastGroupSource
AWS::EC2::TransitGatewayRoute
AWS::EC2::TransitGatewayRouteTableAssociation
AWS::EC2::TransitGatewayRouteTablePropagation
AWS::EC2::VPCDHCPOptionsAssociation
AWS::EC2::VPCEndpointServicePermissions
AWS::EC2::VPNConnectionRoute
AWS::EC2::VolumeAttachment
AWS::ECR::PublicRepository
AWS::ECR::PullThroughCacheRule
AWS::ECR::PullTimeUpdateExclusion
AWS::ECR::Repository
AWS::ECR::RepositoryCreationTemplate
AWS::ECS::CapacityProvider
AWS::ECS::Cluster

Resource
AWS::ECS::ClusterCapacityProviderAssociations
AWS::ECS::PrimaryTaskSet
AWS::EKS::AccessEntry
AWS::EKS::Addon
AWS::EKS::Cluster
AWS::EKS::FargateProfile
AWS::EKS::IdentityProviderConfig
AWS::EMR::SecurityConfiguration
AWS::EMR::StudioSessionMapping
AWS::EMR::WALWorkspace
AWS::ElastiCache::CacheCluster
AWS::ElastiCache::ReplicationGroup
AWS::ElastiCache::ServerlessCache
AWS::ElastiCache::SubnetGroup
AWS::ElastiCache::User
AWS::ElastiCache::UserGroup
AWS::ElasticBeanstalk::Application
AWS::ElasticBeanstalk::Environment
AWS::EntityResolution::IdMappingWorkflow
AWS::EntityResolution::IdNamespace

Resource
AWS::EntityResolution::MatchingWorkflow
AWS::EntityResolution::PolicyStatement
AWS::EntityResolution::SchemaMapping
AWS::Events::ApiDestination
AWS::Events::Archive
AWS::Events::Connection
AWS::Events::Endpoint
AWS::Events::EventBus
AWS::Events::EventBusPolicy
AWS::FIS::TargetAccountConfiguration
AWS::FMS::NotificationChannel
AWS::FSx::S3AccessPointAttachment
AWS::GameLift::ContainerGroupDefinition
AWS::GameLift::GameSessionQueue
AWS::GameLift::Location
AWS::GameLift::MatchmakingConfiguration
AWS::GameLift::MatchmakingRuleSet
AWS::Glue::Crawler
AWS::Glue::Database
AWS::Glue::Job

Resource
AWS::Glue::SchemaVersionMetadata
AWS::Glue::Trigger
AWS::Glue::UsageProfile
AWS::GuardDuty::Filter
AWS::GuardDuty::Master
AWS::GuardDuty::Member
AWS::IAM::Group
AWS::IAM::GroupPolicy
AWS::IAM::InstanceProfile
AWS::IAM::Role
AWS::IAM::RolePolicy
AWS::IAM::ServerCertificate
AWS::IAM::User
AWS::IAM::UserPolicy
AWS::InternetMonitor::Monitor
AWS::IoT::AccountAuditConfiguration
AWS::IoT::Authorizer
AWS::IoT::BillingGroup
AWS::IoT::CertificateProvider
AWS::IoT::Command

Resource
AWS::IoT::CustomMetric
AWS::IoT::Dimension
AWS::IoT::DomainConfiguration
AWS::IoT::FleetMetric
AWS::IoT::JobTemplate
AWS::IoT::Logging
AWS::IoT::MitigationAction
AWS::IoT::ProvisioningTemplate
AWS::IoT::RoleAlias
AWS::IoT::ScheduledAudit
AWS::IoT::SecurityProfile
AWS::IoT::SoftwarePackage
AWS::IoT::SoftwarePackageVersion
AWS::IoT::Thing
AWS::IoT::ThingGroup
AWS::IoT::ThingType
AWS::IoT::TopicRule
AWS::IoTAnalytics::Channel
AWS::IoTAnalytics::Dataset
AWS::IoTAnalytics::Datastore

Resource
AWS::IoTAnalytics::Pipeline
AWS::IoTEvents::AlarmModel
AWS::IoTEvents::DetectorModel
AWS::IoTEvents::Input
AWS::IoTFleetWise::Campaign
AWS::IoTFleetWise::DecoderManifest
AWS::IoTFleetWise::Fleet
AWS::IoTFleetWise::ModelManifest
AWS::IoTFleetWise::SignalCatalog
AWS::IoTFleetWise::StateTemplate
AWS::IoTFleetWise::Vehicle
AWS::IoTTwinMaker::ComponentType
AWS::IoTTwinMaker::Entity
AWS::IoTTwinMaker::Scene
AWS::IoTTwinMaker::SyncJob
AWS::IoTTwinMaker::Workspace
AWS::IoTWireless::Destination
AWS::IoTWireless::NetworkAnalyzerConfiguration
AWS::IoTWireless::PartnerAccount
AWS::KMS::Alias

Resource
AWS::KafkaConnect::Connector
AWS::KafkaConnect::CustomPlugin
AWS::KafkaConnect::WorkerConfiguration
AWS::Kinesis::ResourcePolicy
AWS::Kinesis::Stream
AWS::KinesisAnalyticsV2::Application
AWS::KinesisFirehose::DeliveryStream
AWS::KinesisVideo::SignalingChannel
AWS::KinesisVideo::Stream
AWS::LakeFormation::DataCellsFilter
AWS::LakeFormation::Tag
AWS::Lambda::CapacityProvider
AWS::Lambda::Function
AWS::Lex::ResourcePolicy
AWS::Lightsail::Alarm
AWS::Lightsail::Bucket
AWS::Lightsail::Certificate
AWS::Lightsail::Container
AWS::Lightsail::Database
AWS::Lightsail::DatabaseSnapshot

Resource
AWS::Lightsail::Disk
AWS::Lightsail::DiskSnapshot
AWS::Lightsail::Distribution
AWS::Lightsail::Domain
AWS::Lightsail::Instance
AWS::Lightsail::InstanceSnapshot
AWS::Lightsail::LoadBalancer
AWS::Lightsail::LoadBalancerTlsCertificate
AWS::Lightsail::StaticIp
AWS::Location::APIKey
AWS::Location::GeofenceCollection
AWS::Location::Map
AWS::Location::PlaceIndex
AWS::Location::RouteCalculator
AWS::Location::Tracker
AWS::Location::TrackerConsumer
AWS::Logs::DeliveryDestination
AWS::Logs::DeliverySource
AWS::Logs::Destination
AWS::Logs::Integration

Resource
AWS::Logs::LogGroup
AWS::Logs::LogStream
AWS::Logs::MetricFilter
AWS::Logs::ResourcePolicy
AWS::Logs::ScheduledQuery
AWS::Logs::SubscriptionFilter
AWS::Logs::Transformer
AWS::LookoutEquipment::InferenceScheduler
AWS::LookoutVision::Project
AWS::M2::Deployment
AWS::MSK::BatchScramSecret
AWS::MSK::ClusterPolicy
AWS::MSK::Replicator
AWS::MSK::Topic
AWS::MWAA::Environment
AWS::MWAAServerless::Workflow
AWS::Macie::FindingsFilter
AWS::MediaConnect::BridgeOutput
AWS::MediaConnect::BridgeSource
AWS::MediaConnect::FlowVpcInterface

Resource
AWS::MediaLive::Multiplexprogram
AWS::MediaPackage::Asset
AWS::MediaPackage::Channel
AWS::MediaPackage::OriginEndpoint
AWS::MediaPackage::PackagingConfiguration
AWS::MediaPackage::PackagingGroup
AWS::MediaPackageV2::Channel
AWS::MediaPackageV2::ChannelGroup
AWS::MediaPackageV2::ChannelPolicy
AWS::MediaPackageV2::OriginEndpoint
AWS::MediaPackageV2::OriginEndpointPolicy
AWS::MediaTailor::Channel
AWS::MediaTailor::ChannelPolicy
AWS::MediaTailor::LiveSource
AWS::MediaTailor::PlaybackConfiguration
AWS::MediaTailor::SourceLocation
AWS::MediaTailor::VodSource
AWS::MemoryDB::ACL
AWS::MemoryDB::Cluster
AWS::MemoryDB::ParameterGroup

Resource
AWS::MemoryDB::SubnetGroup
AWS::MemoryDB::User
AWS::Neptune::DBCluster
AWS::Neptune::DBClusterParameterGroup
AWS::Neptune::DBInstance
AWS::Neptune::DBParameterGroup
AWS::Neptune::DBSubnetGroup
AWS::Neptune::EventSubscription
AWS::NeptuneGraph::Graph
AWS::NeptuneGraph::PrivateGraphEndpoint
AWS::NetworkFirewall::LoggingConfiguration
AWS::NetworkManager::ConnectPeer
AWS::NetworkManager::CoreNetwork
AWS::NetworkManager::CoreNetworkPrefixListAssociation
AWS::NetworkManager::CustomerGatewayAssociation
AWS::NetworkManager::DirectConnectGatewayAttachment
AWS::NetworkManager::LinkAssociation
AWS::NetworkManager::SiteToSiteVpnAttachment
AWS::NetworkManager::TransitGatewayRegistration
AWS::NetworkManager::VpcAttachment

Resource
AWS::Notifications::ChannelAssociation
AWS::Notifications::ManagedNotificationAccountContactAssociation
AWS::Notifications::ManagedNotificationAdditionalChannelAssociation
AWS::Notifications::NotificationHub
AWS::Notifications::OrganizationalUnitAssociation
AWS::NovaAct::WorkflowDefinition
AWS::ObservabilityAdmin::OrganizationCentralizationRule
AWS::ObservabilityAdmin::TelemetryEnrichment
AWS::ObservabilityAdmin::TelemetryPipelines
AWS::Omics::AnnotationStore
AWS::Omics::Configuration
AWS::Omics::VariantStore
AWS::OpenSearchServerless::AccessPolicy
AWS::OpenSearchServerless::Collection
AWS::OpenSearchServerless::CollectionGroup
AWS::OpenSearchServerless::CollectionIndex
AWS::OpenSearchServerless::Index
AWS::OpenSearchServerless::LifecyclePolicy
AWS::OpenSearchServerless::SecurityConfig
AWS::OpenSearchServerless::SecurityPolicy

Resource
AWS::OpenSearchServerless::VpcEndpoint
AWS::OpenSearchService::Application
AWS::OpenSearchService::Domain
AWS::PCAConnectorAD::ServicePrincipalName
AWS::PCAConnectorAD::TemplateGroupAccessControlEntry
AWS::Panorama::PackageVersion
AWS::PaymentCryptography::Alias
AWS::Pinpoint::InAppTemplate
AWS::Pipes::Pipe
AWS::Proton::EnvironmentTemplate
AWS::Proton::ServiceTemplate
AWS::QBusiness::Permission
AWS::QuickSight::ActionConnector
AWS::QuickSight::Analysis
AWS::QuickSight::CustomPermissions
AWS::QuickSight::Dashboard
AWS::QuickSight::DataSet
AWS::QuickSight::DataSource
AWS::QuickSight::Folder
AWS::QuickSight::RefreshSchedule

Resource
AWS::QuickSight::Template
AWS::QuickSight::Theme
AWS::QuickSight::Topic
AWS::QuickSight::VPCConnection
AWS::RDS::CustomDBEngineVersion
AWS::RDS::DBCluster
AWS::RDS::DBClusterParameterGroup
AWS::RDS::DBInstance
AWS::RDS::DBParameterGroup
AWS::RDS::DBProxy
AWS::RDS::DBProxyEndpoint
AWS::RDS::DBProxyTargetGroup
AWS::RDS::DBShardGroup
AWS::RDS::DBSubnetGroup
AWS::RDS::EventSubscription
AWS::RDS::GlobalCluster
AWS::RDS::OptionGroup
AWS::RUM::AppMonitor
AWS::Redshift::Cluster
AWS::Redshift::ClusterParameterGroup

Resource
AWS::Redshift::EndpointAccess
AWS::Redshift::EndpointAuthorization
AWS::Redshift::EventSubscription
AWS::Redshift::ScheduledAction
AWS::RedshiftServerless::Namespace
AWS::RedshiftServerless::Snapshot
AWS::RedshiftServerless::Workgroup
AWS::Rekognition::Collection
AWS::Rekognition::Project
AWS::Rekognition::StreamProcessor
AWS::ResourceGroups::Group
AWS::Route53::DNSSEC
AWS::Route53::KeySigningKey
AWS::Route53RecoveryReadiness::Cell
AWS::Route53RecoveryReadiness::ReadinessCheck
AWS::Route53RecoveryReadiness::RecoveryGroup
AWS::Route53RecoveryReadiness::ResourceSet
AWS::Route53Resolver::ResolverConfig
AWS::S3::AccessPoint
AWS::S3::Bucket

Resource
AWS::S3::BucketPolicy
AWS::S3::MultiRegionAccessPoint
AWS::S3::MultiRegionAccessPointPolicy
AWS::S3::StorageLens
AWS::S3::StorageLensGroup
AWS::S3Express::AccessPoint
AWS::S3Express::BucketPolicy
AWS::S3Express::DirectoryBucket
AWS::S3Files::FileSystemPolicy
AWS::S3ObjectLambda::AccessPoint
AWS::S3ObjectLambda::AccessPointPolicy
AWS::S3Outposts::BucketPolicy
AWS::S3Tables::Namespace
AWS::S3Tables::TableBucketPolicy
AWS::S3Tables::TablePolicy
AWS::S3Vectors::Index
AWS::S3Vectors::VectorBucket
AWS::S3Vectors::VectorBucketPolicy
AWS::SES::ConfigurationSet
AWS::SES::ContactList

Resource
AWS::SES::CustomVerificationEmailTemplate
AWS::SES::DedicatedIpPool
AWS::SES::EmailIdentity
AWS::SES::MultiRegionEndpoint
AWS::SES::Tenant
AWS::SMSVOICE::ConfigurationSet
AWS::SMSVOICE::OptOutList
AWS::SMSVOICE::ResourcePolicy
AWS::SMSVOICE::SenderId
AWS::SNS::TopicInlinePolicy
AWS::SQS::QueueInlinePolicy
AWS::SSM::Document
AWS::SSM::Parameter
AWS::SSM::ResourceDataSync
AWS::SSO::ApplicationAssignment
AWS::SSO::Assignment
AWS::SSO::InstanceAccessControlAttributeConfiguration
AWS::SageMaker::App
AWS::SageMaker::AppImageConfig
AWS::SageMaker::Cluster

Resource
AWS::SageMaker::Device
AWS::SageMaker::DeviceFleet
AWS::SageMaker::FeatureGroup
AWS::SageMaker::Image
AWS::SageMaker::InferenceExperiment
AWS::SageMaker::MlflowTrackingServer
AWS::SageMaker::ModelCard
AWS::SageMaker::Pipeline
AWS::SageMaker::ProcessingJob
AWS::SageMaker::Space
AWS::SageMaker::StudioLifecycleConfig
AWS::SageMaker::UserProfile
AWS::Scheduler::Schedule
AWS::Scheduler::ScheduleGroup
AWS::SecurityHub::SecurityControl
AWS::SecurityHub::Standard
AWS::SecurityLake::AwsLogSource
AWS::SecurityLake::SubscriberNotification
AWS::ServiceCatalog::PortfolioPrincipalAssociation
AWS::ServiceCatalog::PortfolioProductAssociation

Resource
AWS::ServiceCatalog::PortfolioShare
AWS::ServiceCatalog::ServiceActionAssociation
AWS::ServiceCatalog::TagOptionAssociation
AWS::ServiceCatalogAppRegistry::Application
AWS::ServiceCatalogAppRegistry::AttributeGroup
AWS::Signer::ProfilePermission
AWS::SimSpaceWeaver::Simulation
AWS::StepFunctions::StateMachine
AWS::SupportApp::SlackChannelConfiguration
AWS::SupportApp::SlackWorkspaceConfiguration
AWS::Synthetics::Canary
AWS::Synthetics::Group
AWS::Timestream::Database
AWS::Timestream::Table
AWS::Transfer::User
AWS::VerifiedPermissions::PolicyStoreAlias
AWS::VpcLattice::AccessLogSubscription
AWS::VpcLattice::AuthPolicy
AWS::VpcLattice::DomainVerification
AWS::VpcLattice::Listener

Resource
AWS::VpcLattice::ResourcePolicy
AWS::VpcLattice::Rule
AWS::VpcLattice::Service
AWS::VpcLattice::ServiceNetwork
AWS::VpcLattice::ServiceNetworkServiceAssociation
AWS::VpcLattice::ServiceNetworkVpcAssociation
AWS::VpcLattice::TargetGroup
AWS::WAFv2::LoggingConfiguration
AWS::WAFv2::WebACLAssociation
AWS::WorkspacesInstances::VolumeAssociation
AWS::XRay::ResourcePolicy

Reverting an import operation

To revert an import operation, specify a `Retain` deletion policy for the resource you want to remove from the template to ensure that it's preserved when you delete it from the stack.

Revert an import operation using the AWS Management Console

1. Specify a `Retain` DeletionPolicy for the resources you want to remove from your stack. In the following example template, `GamesTable` is the target of this revert operation.

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
```

```
            "Type":"AWS::DynamoDB::Table",
            "Properties":{
               "TableName":"Service",
               "AttributeDefinitions":[
                  {
                     "AttributeName":"key",
                     "AttributeType":"S"
                  }
               ],
               "KeySchema":[
                  {
                     "AttributeName":"key",
                     "KeyType":"HASH"
                  }
               ],
               "ProvisionedThroughput":{
                  "ReadCapacityUnits":5,
                  "WriteCapacityUnits":1
               }
            }
         },
         "GamesTable": {
             "Type": "AWS::DynamoDB::Table",
             "DeletionPolicy": "Retain",
             "Properties": {
                 "TableName": "Games",
                 "AttributeDefinitions": [
                     {
                         "AttributeName": "key",
                         "AttributeType": "S"
                     }
                 ],
                 "KeySchema": [
                     {
                         "AttributeName": "key",
                         "KeyType": "HASH"
                     }
                 ],
                 "ProvisionedThroughput": {
                     "ReadCapacityUnits": 5,
                     "WriteCapacityUnits": 1
                 }
             }
         }
```

```
    }
}
```

2. Open the CloudFormation console to perform a stack update to apply the deletion policy.

   a. On the **Stacks** page, with the stack selected, choose **Update**, and then choose **Update stack (standard)**.

   b. Under **Prepare template**, choose **Replace current template**.

   c. Under **Specify template**, provide the updated source template with the `DeletionPolicy` attribute on `GamesTable`, and then choose **Next**.

      - Choose **Amazon S3 URL**, and then specify the URL to the updated source template in the text box.
      - Choose **Upload a template file**, and then browse for the updated source template file.

   d. On the **Specify stack details** page, no changes are required. Choose **Next**.

   e. On the **Configure stack options** page, no changes are required. Choose **Next**.

   f. On the **Review *MyStack*** page, review your changes. If your template contains IAM resources, select **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates. Then, either update your source stack by creating a change set or update your source stack directly.

3. Remove the resource, related parameters, and outputs from the stack template. In this example, the template now looks like the following.

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
```

```
            ],
            "KeySchema":[
               {
                  "AttributeName":"key",
                  "KeyType":"HASH"
               }
            ],
            "ProvisionedThroughput":{
               "ReadCapacityUnits":5,
               "WriteCapacityUnits":1
            }
         }
      }
   }
}
```

4. Repeat step 2 to delete the resource (`GamesTable`) and its related parameters and outputs from the stack.

Revert an import operation using the AWS CLI

1. Specify a `Retain` DeletionPolicy for the resources you want to remove from your stack. In the following example template, `GamesTable` is the target of this revert operation.

   Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
```

```
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        },
        "GamesTable": {
            "Type": "AWS::DynamoDB::Table",
            "DeletionPolicy": "Retain",
            "Properties": {
                "TableName": "Games",
                "AttributeDefinitions": [
                    {
                        "AttributeName": "key",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "key",
                        "KeyType": "HASH"
                    }
                ],
                "ProvisionedThroughput": {
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 1
                }
            }
        }
    }
}
```

2. Update the stack to apply the deletion policy to the resource.

```
aws cloudformation update-stack --stack-name MyStack
```

3. Remove the resource, related parameters, and outputs from the stack template. In this example, the template now looks like the following.

Example JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Import test",
    "Resources": {
        "ServiceTable":{
           "Type":"AWS::DynamoDB::Table",
           "Properties":{
              "TableName":"Service",
              "AttributeDefinitions":[
                 {
                    "AttributeName":"key",
                    "AttributeType":"S"
                 }
              ],
              "KeySchema":[
                 {
                    "AttributeName":"key",
                    "KeyType":"HASH"
                 }
              ],
              "ProvisionedThroughput":{
                 "ReadCapacityUnits":5,
                 "WriteCapacityUnits":1
              }
           }
        }
    }
}
```

4. Update the stack to delete the resource (`GamesTable`) and its related parameters and outputs from the stack.

```
aws cloudformation update-stack --stack-name MyStack
```

Stack refactoring

With stack refactoring, you can reorganize resources in your CloudFormation stacks while preserving existing resource properties and data. You can move resources between stacks, split large stacks into smaller ones, or combine multiple stacks into one.

Topics

- How stack refactoring works
- Stack refactoring considerations
- Prerequisites
- Refactor stacks (console)
- Refactor stacks (AWS CLI)
- Resource limitations

How stack refactoring works

Refactoring stacks involves these phases:

1. **Assess your current infrastructure** – Review your existing CloudFormation stacks and resources to identify stack refactoring opportunities.
2. **Plan your refactor** – Define how resources should be organized. Consider your dependencies, naming conventions, and operational limits. These can affect the CloudFormation validation later.
3. **Determine the destination stacks** – Decide which stacks you will refactor resources into. You can move resources between at least 2 stacks (using the console), and a maximum of 5 stacks (using the AWS CLI). Resources can be moved between nested stacks.
4. **Update your templates** – Change your CloudFormation templates to reflect the planned change, such as moving resource definitions between templates. You can rename logical IDs during this process.
5. **Create the stack refactor** – Provide a list of stack names and templates that you want to refactor.
6. **Review the refactor impact and resolve any conflicts** – CloudFormation validates the templates you provide and checks cross-stack dependencies, resource types with tag update problems, and resource logical ID conflicts.

If the validation succeeds, CloudFormation will generate a preview of the refactor actions that will occur during execution.

If the validation fails, resolve the identified issues and retry. For conflicts, provide a resource logical ID mapping that shows the source and destination of the conflicting resources.

7. **Execute the refactor** – After confirming the changes align with your refactoring goals, complete the stack refactor.
8. **Monitor** – Track the execution status to ensure the operation completes successfully.

Stack refactoring considerations

As you refactor your stacks, keep the following in mind:

- Stack refactoring is limited to reorganizing existing resources. You cannot create or delete resources, modify resource configurations, or change or add new parameters, conditions, or mappings during refactoring. To make these changes, update your stacks first, and then perform the stack refactor.
- You can't refactor the same resource into multiple stacks.
- You can't refactor resources that refer to pseudo parameters whose values differ between the source and destination stacks, such as `AWS::StackName`.
- CloudFormation doesn't support empty stacks. If refactoring would leave a stack with no resources, you must first add at least one resource to that stack before you run create-stack-refactor. This can be a simple resource like `AWS::SNS::Topic` or `AWS::CloudFormation::WaitCondition`. For example:

```
Resources:
  MySimpleSNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: MySimpleTopic
```

- Stack refactor doesn't support stacks that have stack policies attached, regardless of what the policies allow or deny.

Prerequisites

To refactor stacks, you must have already created the revised templates.

Use the get-template command to retrieve the CloudFormation templates for the stacks that you want to refactor.

```
aws cloudformation get-template --stack-name Stack1
```

When you have the templates, use the integrated development environment (IDE) of your choice to update them to use the desired structure and resource organization.

Refactor stacks (console)

Use the following procedure to refactor stacks using the console.

To refactor stacks

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region where your stacks are located.
3. In the navigation pane on the left, choose **Stack refactors**.
4. On the **Stack refactors** page, choose **Start stack refactor**.
5. For **Description**, provide a description to help you identify your stack refactor. Then, choose **Next**.
6. For Stack 1, do the following:

    a. Choose either **Update the template for an existing stack** or **Create new stack**.

    If you chose **Update the template for an existing stack**, then select an existing stack from the list. Or choose **Enter a stack ARN** to enter the ARN of an existing stack.

    If you chose **Create new stack**, for **Stack name**, provide a name for the new stack.

    b. Under **Replace existing template with refactored template**, choose **Amazon S3 URL** or **Upload a template file** to upload the desired template for Stack 1.

    c. Choose **Next**.
7. For Stack 2, do the following:

a. Choose either **Update the template for an existing stack** or **Create new stack**.

   If you chose **Update the template for an existing stack**, then select an existing stack from the list. Or choose **Enter a stack ARN** to enter the ARN of an existing stack.

   If you chose **Create new stack**, for **Stack name**, provide a name for the new stack.

b. Under **Replace existing template with refactored template**, choose **Amazon S3 URL** or **Upload template file** to upload the desired template for Stack 2.

c. Choose **Next**.

8. On the **Specify logical resource ID renames** page, make sure CloudFormation knows how to refactor stacks by mapping any resources shown to their correct logical IDs. As part of the stack refactor, if any resource's logical IDs were changed, you need to specify how it was renamed by providing the source stack name, the original logical ID, the destination stack name and the renamed logical ID. In some instances, the CloudFormation console might automatically detect the resource mapping, and you can simply verify that the pre-filled resource mapping is correct before proceeding.
9. Choose **Next**.
10. On the **Review and execute** page, review all of your selections from the previous steps and confirm that everything is set up correctly.
11. When you are ready to refactor the stacks, choose **Execute stack refactor**.

Refactor stacks (AWS CLI)

The AWS CLI commands for stack refactoring include:

- create-stack-refactor to validate and generate a preview of planned changes.
- describe-stack-refactor to retrieve the status and details of a stack refactoring operation.
- execute-stack-refactor to complete the validated stack refactoring operation.
- list-stack-refactors to list all stack refactoring operations in your account with their current status and basic information.
- list-stack-refactor-actions to show a preview of the specific actions CloudFormation will perform on each stack and resource during the refactor execution.

Use the following procedure to refactor a stack using the AWS CLI.

To refactor stacks

1. Use the create-stack-refactor command and provide the stack names and updated templates for the stacks to refactor. Include the `--enable-stack-creation` option to allow CloudFormation to create new stacks if they don't already exist.

```
aws cloudformation create-stack-refactor \
  --stack-definitions \
    StackName=Stack1,TemplateBody@=file://template1-updated.yaml \
    StackName=Stack2,TemplateBody@=file://template2-updated.yaml \
  --enable-stack-creation
```

The command returns a `StackRefactorId` that you'll use in later steps.

```
{
    "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841"
}
```

If conflicts are detected during template validation (which you can confirm in the next step), use the create-stack-refactor command with the `--resource-mappings` option.

```
aws cloudformation create-stack-refactor \
  --stack-definitions \
    StackName=Stack1,TemplateBody@=file://template1-updated.yaml \
    StackName=Stack2,TemplateBody@=file://template2-updated.yaml \
  --enable-stack-creation \
  --resource-mappings file://resource-mapping.json
```

The following is an example `resource-mapping.json` file.

```
[
    {
        "Source": {
            "StackName": "Stack1",
            "LogicalResourceId": "MySNSTopic"
        },
        "Destination": {
            "StackName": "Stack2",
            "LogicalResourceId": "MyLambdaSNSTopic"
        }
```

```
    }
]
```

2. Use the describe-stack-refactor command to make sure the `Status` is `CREATE_COMPLETE`. This verifies that the validation is complete.

```
aws cloudformation describe-stack-refactor \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

Example output:

```
{
    "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841",
    "StackIds": [
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
    ],
    "ExecutionStatus": "AVAILABLE",
    "Status": "CREATE_COMPLETE"
}
```

3. Use the list-stack-refactor-actions command to preview the specific actions that will be performed.

```
aws cloudformation list-stack-refactor-actions \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

Example output:

```
{
    "StackRefactorActions": [
        {
            "Action": "MOVE",
            "Entity": "RESOURCE",
            "PhysicalResourceId": "MyTestLambdaRole",
            "Description": "No configuration changes detected.",
            "Detection": "AUTO",
            "TagResources": [],
            "UntagResources": [],
```

```
            "ResourceMapping": {
                "Source": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
                    "LogicalResourceId": "MyLambdaRole"
                },
                "Destination": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack2/5da91650-94b1-11f0-81cf-0a23500e151b",
                    "LogicalResourceId": "MyLambdaRole"
                }
            }
        },
        {
            "Action": "MOVE",
            "Entity": "RESOURCE",
            "PhysicalResourceId": "MyTestFunction",
            "Description": "Resource configuration changes will be validated during
 refactor execution.",
            "Detection": "AUTO",
            "TagResources": [
                {
                    "Key": "aws:cloudformation:stack-name",
                    "Value": "Stack2"
                },
                {
                    "Key": "aws:cloudformation:logical-id",
                    "Value": "MyFunction"
                },
                {
                    "Key": "aws:cloudformation:stack-id",
                    "Value": "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
                }
            ],
            "UntagResources": [
                "aws:cloudformation:stack-name",
                "aws:cloudformation:logical-id",
                "aws:cloudformation:stack-id"
            ],
            "ResourceMapping": {
                "Source": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
```

```
                    "LogicalResourceId": "MyFunction"
                },
                "Destination": {
                    "StackName": "arn:aws:cloudformation:us-
east-1:123456789012:stack/Stack2/5da91650-94b1-11f0-81cf-0a23500e151b",
                    "LogicalResourceId": "MyFunction"
                }
            }
        }
    ]
}
```

4. After reviewing and confirming your changes, use the execute-stack-refactor command to complete the stack refactoring operation.

```
aws cloudformation execute-stack-refactor \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

5. Use the describe-stack-refactor command to monitor the execution status.

```
aws cloudformation describe-stack-refactor \
  --stack-refactor-id 9c384f70-4e07-4ed7-a65d-fee5eb430841
```

Example output:

```
{
    "StackRefactorId": "9c384f70-4e07-4ed7-a65d-fee5eb430841",
    "StackIds": [
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack1/3e6a1ff0-94b1-11f0-aa6f-0a88d2e03acf",
        "arn:aws:cloudformation:us-east-1:123456789012:stack/
Stack2/5da91650-94b1-11f0-81cf-0a23500e151b"
    ],
    "ExecutionStatus": "SUCCEEDED",
    "Status": "COMPLETE"
}
```

Resource limitations

- Stack refactoring only supports resource types with a `provisioningType` of `FULLY_MUTABLE`, which you can check using the describe-type command.

- CloudFormation will validate resource eligibility during refactor creation and report any unsupported resources in the output of the describe-stack-refactor command.
- The following resources aren't supported for stack refactoring:
  - AWS::ACMPCA::Certificate
  - AWS::ACMPCA::CertificateAuthority
  - AWS::ACMPCA::CertificateAuthorityActivation
  - AWS::ApiGateway::BasePathMapping
  - AWS::ApiGateway::Method
  - AWS::AppConfig::ConfigurationProfile
  - AWS::AppConfig::Deployment
  - AWS::AppConfig::Environment
  - AWS::AppConfig::Extension
  - AWS::AppConfig::ExtensionAssociation
  - AWS::AppStream::DirectoryConfig
  - AWS::AppStream::StackFleetAssociation
  - AWS::AppStream::StackUserAssociation
  - AWS::AppStream::User
  - AWS::AutoScaling::AutoScalingGroup
  - AWS::BackupGateway::Hypervisor
  - AWS::CertificateManager::Certificate
  - AWS::CloudFormation::CustomResource
  - AWS::CloudFormation::Macro
  - AWS::CloudFormation::WaitCondition
  - AWS::CloudFormation::WaitConditionHandle
  - AWS::CodeDeploy::DeploymentGroup
  - AWS::CodePipeline::CustomActionType
  - AWS::Cognito::UserPoolRiskConfigurationAttachment
  - AWS::Cognito::UserPoolUICustomizationAttachment
  - AWS::Cognito::UserPoolUserToGroupAttachment
  - AWS::Config::ConfigRule

- AWS::Config::ConfigurationRecorder
- AWS::Config::DeliveryChannel
- AWS::DataBrew::Dataset
- AWS::DataBrew::Job
- AWS::DataBrew::Project
- AWS::DataBrew::Recipe
- AWS::DataBrew::Ruleset
- AWS::DataBrew::Schedule
- AWS::DataZone::DataSource
- AWS::DataZone::Environment
- AWS::DataZone::EnvironmentBlueprintConfiguration
- AWS::DataZone::EnvironmentProfile
- AWS::DataZone::Project
- AWS::DataZone::SubscriptionTarget
- AWS::DirectoryService::MicrosoftAD
- AWS::DynamoDB::GlobalTable
- AWS::EC2::CustomerGateway
- AWS::EC2::EIP
- AWS::EC2::LaunchTemplate
- AWS::EC2::NetworkInterfacePermission
- AWS::EC2::PlacementGroup
- AWS::EC2::SpotFleet
- AWS::EC2::VPCDHCPOptionsAssociation
- AWS::EC2::VolumeAttachment
- AWS::EMR::Cluster
- AWS::EMR::InstanceFleetConfig
- AWS::EMR::InstanceGroupConfig
- AWS::ElastiCache::CacheCluster
- AWS::ElastiCache::ReplicationGroup
- AWS::ElastiCache::SecurityGroup

- AWS::ElastiCache::SecurityGroupIngress
- AWS::ElasticBeanstalk::ConfigurationTemplate
- AWS::ElasticLoadBalancing::LoadBalancer
- AWS::ElasticLoadBalancingV2::ListenerCertificate
- AWS::Elasticsearch::Domain
- AWS::FIS::ExperimentTemplate
- AWS::Glue::Schema
- AWS::GuardDuty::IPSet
- AWS::GuardDuty::PublishingDestination
- AWS::GuardDuty::ThreatIntelSet
- AWS::IAM::AccessKey
- AWS::IAM::UserToGroupAddition
- AWS::ImageBuilder::Component
- AWS::IoT::PolicyPrincipalAttachment
- AWS::IoT::ThingPrincipalAttachment
- AWS::IoTFleetWise::Campaign
- AWS::IoTWireless::WirelessDeviceImportTask
- AWS::Lambda::EventInvokeConfig
- AWS::Lex::BotVersion
- AWS::M2::Application
- AWS::MSK::Configuration
- AWS::MSK::ServerlessCluster
- AWS::Maester::DocumentType
- AWS::MediaTailor::Channel
- AWS::NeptuneGraph::PrivateGraphEndpoint
- AWS::Omics::AnnotationStore
- AWS::Omics::ReferenceStore
- AWS::Omics::SequenceStore
- AWS::OpenSearchServerless::Collection
- AWS::OpsWorks::App

- AWS::OpsWorks::ElasticLoadBalancerAttachment
- AWS::OpsWorks::Instance
- AWS::OpsWorks::Layer
- AWS::OpsWorks::Stack
- AWS::OpsWorks::UserProfile
- AWS::OpsWorks::Volume
- AWS::PCAConnectorAD::Connector
- AWS::PCAConnectorAD::DirectoryRegistration
- AWS::PCAConnectorAD::Template
- AWS::PCAConnectorAD::TemplateGroupAccessControlEntry
- AWS::Panorama::PackageVersion
- AWS::QuickSight::Theme
- AWS::RDS::DBSecurityGroup
- AWS::RDS::DBSecurityGroupIngress
- AWS::Redshift::ClusterSecurityGroup
- AWS::Redshift::ClusterSecurityGroupIngress
- AWS::RefactorSpaces::Environment
- AWS::RefactorSpaces::Route
- AWS::RefactorSpaces::Service
- AWS::RoboMaker::RobotApplication
- AWS::RoboMaker::SimulationApplication
- AWS::Route53::RecordSet
- AWS::Route53::RecordSetGroup
- AWS::SDB::Domain
- AWS::SageMaker::InferenceComponen
- AWS::ServiceCatalog::PortfolioPrincipalAssociation
- AWS::ServiceCatalog::PortfolioProductAssociation
- AWS::ServiceCatalog::PortfolioShare
- AWS::ServiceCatalog::TagOptionAssociation
- AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation

- AWS::ServiceCatalogAppRegistry::ResourceAssociation
- AWS::StepFunctions::StateMachineVersion
- AWS::Synthetics::Canary
- AWS::VoiceID::Domain
- AWS::WAF::ByteMatchSet
- AWS::WAF::IPSet
- AWS::WAF::Rule
- AWS::WAF::SizeConstraintSet
- AWS::WAF::SqlInjectionMatchSet
- AWS::WAF::WebACL
- AWS::WAF::XssMatchSet
- AWS::WAFv2::IPSet
- AWS::WAFv2::RegexPatternSet
- AWS::WAFv2::RuleGroup
- AWS::WAFv2::WebACL
- AWS::WorkSpaces::Workspace

Resource type support

The following table lists AWS resource types that currently support import, drift detection, and infrastructure as code (IaC) generator operations. Each resource type name links to its corresponding reference topic in the CloudFormation Template Reference Guide.

A resource that supports resource import could also support auto-import. For more information, see Import AWS resources into a CloudFormation stack.

This is not an exhaustive list of AWS resources. If a specific resource type isn't listed, it likely means that it's not accessible through the AWS Cloud Control API. For more information, see Resource types that support Cloud Control API in the *Cloud Control API User Guide*. Each AWS service independently determines which resource types to make accessible through Cloud Control API.

CloudFormation also supports import and drift detection operations for private resource types that are provisionable (those with provisioning types of either FULLY_MUTABLE or IMMUTABLE). To import or perform drift detection on a private resource type, you must first register the default

version of that resource type in your account, and ensure it's provisionable. For more information, see Use third-party private extensions that have been shared with you.

Note that IaC generator only supports AWS resources that are compatible with Cloud Control API in your Region.

To programmatically access information about public and private provisionable resource types, you can use the AWS Cloud Control API. For more information, see Determining if a resource type supports Cloud Control API in the *Cloud Control API User Guide*.

To get started with import, drift detection, or IaC generator, here are some useful topics to review:

- Import AWS resources into a CloudFormation stack
- Detect unmanaged configuration changes to stacks and resources with drift detection
- Generate templates from existing resources with IaC generator

Resource	Import	Drift detection	IaC generator
AWS::ACMPCA::Certificate	Yes	No	No
AWS::ACMPCA::CertificateAuthority	Yes	Yes	Yes
AWS::ACMPCA::CertificateAuthorityActivation	Yes	No	No
AWS::ACMPCA::Permission	Yes	Yes	No
AWS::AIOps::InvestigationGroup	Yes	Yes	Yes
AWS::APS::AnomalyDetector	Yes	Yes	Yes
AWS::APS::ResourcePolicy	Yes	Yes	Yes
AWS::APS::RuleGroupsNamespace	Yes	Yes	Yes
AWS::APS::Scraper	Yes	Yes	Yes
AWS::APS::Workspace	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ARCRegionSwitch::Plan	Yes	Yes	Yes
AWS::ARCZonalShift::AutoshiftObserverNotificationStatus	Yes	Yes	Yes
AWS::ARCZonalShift::ZonalAutoshiftConfiguration	Yes	Yes	Yes
AWS::AWSExternalAnthropic::Workspace	Yes	Yes	No
AWS::AccessAnalyzer::Analyzer	Yes	Yes	Yes
AWS::AmazonMQ::Broker	Yes	Yes	Yes
AWS::AmazonMQ::Configuration	Yes	Yes	Yes
AWS::Amplify::App	Yes	Yes	Yes
AWS::Amplify::Branch	Yes	Yes	Yes
AWS::Amplify::Domain	Yes	Yes	Yes
AWS::AmplifyUIBuilder::Component	Yes	Yes	Yes
AWS::AmplifyUIBuilder::Form	Yes	Yes	Yes
AWS::AmplifyUIBuilder::Theme	Yes	Yes	Yes
AWS::ApiGateway::Account	Yes	Yes	No
AWS::ApiGateway::ApiKey	Yes	Yes	Yes
AWS::ApiGateway::Authorizer	Yes	Yes	Yes
AWS::ApiGateway::BasePathMapping	Yes	Yes	Yes
AWS::ApiGateway::BasePathMappingV2	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ApiGateway::ClientCertificate	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::Deployment	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::DocumentationPart	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::DocumentationVersion	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::DomainName	✓ Yes	✗ No	✓ Yes
AWS::ApiGateway::DomainNameAccessAssociation	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::DomainNameV2	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::GatewayResponse	✓ Yes	✗ No	✓ Yes
AWS::ApiGateway::Method	✓ Yes	✓ Yes	✗ No
AWS::ApiGateway::Model	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::RequestValidator	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::Resource	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::RestApi	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::Stage	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::UsagePlan	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGateway::UsagePlanKey	✓ Yes	✗ No	✓ Yes
AWS::ApiGateway::VpcLink	✓ Yes	✓ Yes	✓ Yes
AWS::ApiGatewayV2::Api	✓ Yes	✓ Yes	✓ Yes

Resource	Import	Drift detection	IaC generator
AWS::ApiGatewayV2::ApiMapping	Yes	Yes	Yes
AWS::ApiGatewayV2::Authorizer	Yes	Yes	Yes
AWS::ApiGatewayV2::Deployment	Yes	Yes	Yes
AWS::ApiGatewayV2::DomainName	Yes	Yes	Yes
AWS::ApiGatewayV2::Integration	Yes	No	Yes
AWS::ApiGatewayV2::IntegrationResponse	Yes	Yes	Yes
AWS::ApiGatewayV2::Model	Yes	Yes	Yes
AWS::ApiGatewayV2::Route	Yes	Yes	Yes
AWS::ApiGatewayV2::RouteResponse	Yes	Yes	Yes
AWS::ApiGatewayV2::RoutingRule	Yes	Yes	Yes
AWS::ApiGatewayV2::Stage	Yes	Yes	No
AWS::ApiGatewayV2::VpcLink	Yes	Yes	Yes
AWS::AppConfig::Application	Yes	Yes	Yes
AWS::AppConfig::ConfigurationProfile	Yes	Yes	Yes
AWS::AppConfig::Deployment	Yes	Yes	Yes
AWS::AppConfig::DeploymentStrategy	Yes	Yes	Yes
AWS::AppConfig::Environment	Yes	Yes	Yes
AWS::AppConfig::Extension	Yes	Yes	Yes
AWS::AppConfig::ExtensionAssociation	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::AppConfig::HostedConfigurationVersion	Yes	Yes	Yes
AWS::AppFlow::Connector	Yes	Yes	Yes
AWS::AppFlow::ConnectorProfile	Yes	Yes	Yes
AWS::AppFlow::Flow	Yes	Yes	Yes
AWS::AppIntegrations::Application	Yes	Yes	Yes
AWS::AppIntegrations::DataIntegration	Yes	Yes	Yes
AWS::AppIntegrations::EventIntegration	Yes	Yes	Yes
AWS::AppRunner::AutoScalingConfiguration	Yes	Yes	Yes
AWS::AppRunner::ObservabilityConfiguration	Yes	Yes	Yes
AWS::AppRunner::Service	Yes	Yes	Yes
AWS::AppRunner::VpcConnector	Yes	Yes	Yes
AWS::AppRunner::VpcIngressConnection	Yes	Yes	Yes
AWS::AppStream::AppBlock	Yes	Yes	No
AWS::AppStream::AppBlockBuilder	Yes	Yes	Yes
AWS::AppStream::Application	Yes	Yes	No
AWS::AppStream::ApplicationEntitlementAssociation	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::AppStream::ApplicationFleetAssociation	Yes	Yes	No
AWS::AppStream::DirectoryConfig	Yes	Yes	Yes
AWS::AppStream::Entitlement	Yes	Yes	No
AWS::AppStream::ImageBuilder	Yes	Yes	Yes
AWS::AppStream::Stack	Yes	Yes	No
AWS::AppStream::StackFleetAssociation	Yes	Yes	No
AWS::AppStream::StackUserAssociation	Yes	Yes	No
AWS::AppStream::User	Yes	Yes	No
AWS::AppSync::Api	Yes	Yes	Yes
AWS::AppSync::ChannelNamespace	Yes	Yes	Yes
AWS::AppSync::DataSource	Yes	Yes	Yes
AWS::AppSync::DomainName	Yes	Yes	Yes
AWS::AppSync::DomainNameApiAssociation	Yes	Yes	No
AWS::AppSync::FunctionConfiguration	Yes	Yes	Yes
AWS::AppSync::GraphQLApi	No	Yes	No
AWS::AppSync::Resolver	Yes	Yes	Yes
AWS::AppSync::SourceApiAssociation	Yes	Yes	Yes
AWS::AppTest::TestCase	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ApplicationAutoScaling::ScalableTarget	Yes	Yes	Yes
AWS::ApplicationAutoScaling::ScalingPolicy	Yes	No	Yes
AWS::ApplicationInsights::Application	Yes	Yes	Yes
AWS::ApplicationSignals::Discovery	Yes	Yes	Yes
AWS::ApplicationSignals::GroupingConfiguration	Yes	Yes	Yes
AWS::ApplicationSignals::ServiceLevelObjective	Yes	Yes	Yes
AWS::Athena::CapacityReservation	Yes	Yes	Yes
AWS::Athena::DataCatalog	Yes	Yes	Yes
AWS::Athena::NamedQuery	Yes	Yes	Yes
AWS::Athena::PreparedStatement	Yes	Yes	Yes
AWS::Athena::WorkGroup	Yes	No	Yes
AWS::AuditManager::Assessment	Yes	No	Yes
AWS::AutoScaling::AutoScalingGroup	Yes	Yes	Yes
AWS::AutoScaling::LaunchConfiguration	Yes	Yes	Yes
AWS::AutoScaling::LifecycleHook	Yes	Yes	Yes
AWS::AutoScaling::ScalingPolicy	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::AutoScaling::ScheduledAction	Yes	Yes	Yes
AWS::AutoScaling::WarmPool	Yes	Yes	Yes
AWS::B2BI::Capability	Yes	Yes	Yes
AWS::B2BI::Partnership	Yes	Yes	Yes
AWS::B2BI::Profile	Yes	Yes	Yes
AWS::B2BI::Transformer	Yes	Yes	Yes
AWS::BCM::Dashboard	Yes	Yes	No
AWS::BCMDataExports::Export	Yes	Yes	Yes
AWS::Backup::BackupPlan	Yes	Yes	Yes
AWS::Backup::BackupSelection	Yes	No	Yes
AWS::Backup::BackupVault	Yes	Yes	Yes
AWS::Backup::Framework	Yes	Yes	Yes
AWS::Backup::LogicallyAirGappedBackupVault	Yes	Yes	Yes
AWS::Backup::ReportPlan	Yes	Yes	Yes
AWS::Backup::RestoreTestingPlan	Yes	Yes	Yes
AWS::Backup::RestoreTestingSelection	Yes	Yes	Yes
AWS::Backup::TieringConfiguration	Yes	Yes	No
AWS::BackupGateway::Hypervisor	Yes	Yes	Yes
AWS::Batch::ComputeEnvironment	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Batch::ConsumableResource	Yes	Yes	Yes
AWS::Batch::JobDefinition	Yes	Yes	Yes
AWS::Batch::JobQueue	Yes	Yes	Yes
AWS::Batch::QuotaShare	Yes	Yes	No
AWS::Batch::SchedulingPolicy	Yes	Yes	Yes
AWS::Batch::ServiceEnvironment	Yes	Yes	Yes
AWS::BcmPricingCalculator::BillScenario	Yes	Yes	No
AWS::Bedrock::Agent	Yes	Yes	Yes
AWS::Bedrock::AgentAlias	Yes	Yes	Yes
AWS::Bedrock::ApplicationInferenceProfile	Yes	Yes	Yes
AWS::Bedrock::AutomatedReasoningPolicy	Yes	Yes	Yes
AWS::Bedrock::AutomatedReasoningPolicyVersion	Yes	Yes	Yes
AWS::Bedrock::Blueprint	Yes	Yes	Yes
AWS::Bedrock::DataAutomationLibrary	Yes	Yes	No
AWS::Bedrock::DataAutomationProject	Yes	Yes	Yes
AWS::Bedrock::DataSource	Yes	Yes	Yes
AWS::Bedrock::EnforcedGuardrailConfiguration	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::Bedrock::Flow	Yes	Yes	Yes
AWS::Bedrock::FlowAlias	Yes	Yes	Yes
AWS::Bedrock::FlowVersion	Yes	Yes	Yes
AWS::Bedrock::Guardrail	Yes	Yes	Yes
AWS::Bedrock::GuardrailVersion	Yes	Yes	No
AWS::Bedrock::IntelligentPromptRouter	Yes	Yes	Yes
AWS::Bedrock::KnowledgeBase	Yes	No	Yes
AWS::Bedrock::Prompt	Yes	Yes	Yes
AWS::Bedrock::PromptVersion	Yes	Yes	Yes
AWS::Bedrock::ResourcePolicy	Yes	Yes	No
AWS::BedrockAgentCore::ApiKeyCredentialProvider	Yes	Yes	No
AWS::BedrockAgentCore::BrowserCustom	Yes	Yes	Yes
AWS::BedrockAgentCore::BrowserProfile	Yes	Yes	No
AWS::BedrockAgentCore::CodeInterpreterCustom	Yes	Yes	Yes
AWS::BedrockAgentCore::ConfigurationBundle	Yes	Yes	No
AWS::BedrockAgentCore::Dataset	Yes	Yes	No
AWS::BedrockAgentCore::Evaluator	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::BedrockAgentCore::Gateway	Yes	Yes	Yes
AWS::BedrockAgentCore::GatewayTarget	Yes	Yes	Yes
AWS::BedrockAgentCore::Harness	Yes	Yes	No
AWS::BedrockAgentCore::Memory	Yes	Yes	Yes
AWS::BedrockAgentCore::OAuth2CredentialProvider	Yes	Yes	No
AWS::BedrockAgentCore::OnlineEvaluationConfig	Yes	Yes	No
AWS::BedrockAgentCore::PaymentConnector	Yes	Yes	No
AWS::BedrockAgentCore::PaymentCredentialProvider	Yes	Yes	No
AWS::BedrockAgentCore::PaymentManager	Yes	Yes	No
AWS::BedrockAgentCore::Policy	Yes	Yes	No
AWS::BedrockAgentCore::PolicyEngine	Yes	Yes	No
AWS::BedrockAgentCore::ResourcePolicy	Yes	Yes	No
AWS::BedrockAgentCore::Runtime	Yes	Yes	Yes
AWS::BedrockAgentCore::RuntimeEndpoint	Yes	Yes	Yes
AWS::BedrockAgentCore::WorkloadIdentity	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::BedrockMantle::Project	Yes	Yes	No
AWS::Billing::BillingView	Yes	Yes	Yes
AWS::BillingConductor::BillingGroup	Yes	Yes	Yes
AWS::BillingConductor::CustomLineItem	Yes	Yes	Yes
AWS::BillingConductor::PricingPlan	Yes	Yes	Yes
AWS::BillingConductor::PricingRule	Yes	Yes	Yes
AWS::Braket::SpendingLimit	Yes	Yes	No
AWS::Budgets::BudgetsAction	Yes	Yes	Yes
AWS::CE::AnomalyMonitor	Yes	Yes	Yes
AWS::CE::AnomalySubscription	Yes	Yes	Yes
AWS::CE::CostCategory	Yes	Yes	Yes
AWS::CUR::ReportDefinition	Yes	Yes	Yes
AWS::Cases::CaseRule	Yes	Yes	No
AWS::Cases::Domain	Yes	Yes	No
AWS::Cases::Field	Yes	Yes	No
AWS::Cases::Layout	Yes	Yes	No
AWS::Cases::Template	Yes	Yes	No
AWS::Cassandra::Keyspace	Yes	Yes	Yes
AWS::Cassandra::Table	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Cassandra::Type	Yes	Yes	Yes
AWS::CertificateManager::Account	Yes	Yes	No
AWS::Chatbot::CustomAction	Yes	Yes	Yes
AWS::Chatbot::MicrosoftTeamsChannelConfiguration	Yes	Yes	Yes
AWS::Chatbot::SlackChannelConfiguration	Yes	Yes	Yes
AWS::Chime::AppInstance	Yes	Yes	No
AWS::Chime::AppInstanceBot	Yes	Yes	No
AWS::CleanRooms::AnalysisTemplate	Yes	Yes	Yes
AWS::CleanRooms::Collaboration	Yes	Yes	Yes
AWS::CleanRooms::ConfiguredTable	Yes	Yes	Yes
AWS::CleanRooms::ConfiguredTableAssociation	Yes	Yes	Yes
AWS::CleanRooms::IdMappingTable	Yes	Yes	Yes
AWS::CleanRooms::IdNamespaceAssociation	Yes	Yes	Yes
AWS::CleanRooms::Membership	Yes	Yes	Yes
AWS::CleanRooms::PrivacyBudgetTemplate	Yes	Yes	Yes
AWS::CleanRoomsML::ConfiguredModelAlgorithm	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::CleanRoomsML::ConfiguredModelAlgorithmAssociation	Yes	Yes	No
AWS::CleanRoomsML::TrainingDataset	Yes	Yes	Yes
AWS::CloudFormation::GuardHook	Yes	Yes	Yes
AWS::CloudFormation::HookDefaultVersion	Yes	Yes	Yes
AWS::CloudFormation::HookTypeConfig	Yes	Yes	Yes
AWS::CloudFormation::HookVersion	Yes	Yes	Yes
AWS::CloudFormation::LambdaHook	Yes	Yes	Yes
AWS::CloudFormation::ModuleDefaultVersion	Yes	Yes	Yes
AWS::CloudFormation::ModuleVersion	Yes	Yes	No
AWS::CloudFormation::PublicTypeVersion	Yes	Yes	Yes
AWS::CloudFormation::Publisher	Yes	Yes	Yes
AWS::CloudFormation::ResourceDefaultVersion	Yes	Yes	Yes
AWS::CloudFormation::ResourceVersion	Yes	Yes	Yes
AWS::CloudFormation::Stack	Yes	No	Yes
AWS::CloudFormation::StackSet	Yes	Yes	Yes
AWS::CloudFormation::TypeActivation	Yes	Yes	Yes
AWS::CloudFront::AnycastIpList	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::CloudFront::CachePolicy	Yes	Yes	Yes
AWS::CloudFront::CloudFrontOriginAccessIdentity	Yes	Yes	Yes
AWS::CloudFront::ConnectionFunction	Yes	Yes	Yes
AWS::CloudFront::ConnectionGroup	Yes	Yes	Yes
AWS::CloudFront::ContinuousDeploymentPolicy	Yes	Yes	Yes
AWS::CloudFront::Distribution	Yes	Yes	Yes
AWS::CloudFront::DistributionTenant	Yes	Yes	Yes
AWS::CloudFront::Function	Yes	Yes	Yes
AWS::CloudFront::KeyGroup	Yes	Yes	Yes
AWS::CloudFront::KeyValueStore	Yes	Yes	Yes
AWS::CloudFront::MonitoringSubscription	Yes	Yes	No
AWS::CloudFront::OriginAccessControl	Yes	Yes	Yes
AWS::CloudFront::OriginRequestPolicy	Yes	Yes	Yes
AWS::CloudFront::PublicKey	Yes	Yes	Yes
AWS::CloudFront::RealtimeLogConfig	Yes	Yes	Yes
AWS::CloudFront::ResponseHeadersPolicy	Yes	Yes	Yes
AWS::CloudFront::TrustStore	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::CloudFront::VpcOrigin	Yes	Yes	Yes
AWS::CloudTrail::Channel	Yes	Yes	Yes
AWS::CloudTrail::Dashboard	Yes	Yes	Yes
AWS::CloudTrail::EventDataStore	Yes	Yes	Yes
AWS::CloudTrail::ResourcePolicy	Yes	Yes	No
AWS::CloudTrail::Trail	Yes	Yes	Yes
AWS::CloudWatch::Alarm	Yes	Yes	Yes
AWS::CloudWatch::AlarmMuteRule	Yes	Yes	No
AWS::CloudWatch::CompositeAlarm	Yes	Yes	Yes
AWS::CloudWatch::Dashboard	Yes	Yes	Yes
AWS::CloudWatch::LogAlarm	Yes	Yes	No
AWS::CloudWatch::MetricStream	Yes	Yes	Yes
AWS::CloudWatch::OTelEnrichment	Yes	Yes	No
AWS::CodeArtifact::Domain	Yes	Yes	Yes
AWS::CodeArtifact::PackageGroup	Yes	Yes	Yes
AWS::CodeArtifact::Repository	Yes	Yes	Yes
AWS::CodeBuild::Fleet	Yes	Yes	Yes
AWS::CodeConnections::Connection	Yes	Yes	Yes
AWS::CodeDeploy::Application	Yes	Yes	Yes
AWS::CodeDeploy::DeploymentConfig	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::CodeDeploy::DeploymentGroup	Yes	Yes	No
AWS::CodeGuruProfiler::ProfilingGroup	Yes	Yes	Yes
AWS::CodeGuruReviewer::RepositoryAssociation	Yes	Yes	Yes
AWS::CodePipeline::CustomActionType	Yes	Yes	Yes
AWS::CodePipeline::Pipeline	Yes	Yes	Yes
AWS::CodePipeline::Webhook	Yes	Yes	Yes
AWS::CodeStarConnections::Connection	Yes	Yes	Yes
AWS::CodeStarConnections::RepositoryLink	Yes	Yes	Yes
AWS::CodeStarConnections::SyncConfiguration	Yes	Yes	Yes
AWS::CodeStarNotifications::NotificationRule	Yes	Yes	Yes
AWS::Cognito::IdentityPool	Yes	Yes	Yes
AWS::Cognito::IdentityPoolPrincipalTag	Yes	Yes	Yes
AWS::Cognito::IdentityPoolRoleAttachment	Yes	No	Yes
AWS::Cognito::LogDeliveryConfiguration	Yes	Yes	No
AWS::Cognito::ManagedLoginBranding	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::Cognito::Terms	Yes	Yes	Yes
AWS::Cognito::UserPool	Yes	No	Yes
AWS::Cognito::UserPoolClient	Yes	Yes	Yes
AWS::Cognito::UserPoolDomain	Yes	Yes	No
AWS::Cognito::UserPoolGroup	Yes	Yes	Yes
AWS::Cognito::UserPoolIdentityProvider	Yes	Yes	Yes
AWS::Cognito::UserPoolResourceServer	Yes	Yes	Yes
AWS::Cognito::UserPoolRiskConfigurationAttachment	Yes	Yes	No
AWS::Cognito::UserPoolUICustomizationAttachment	Yes	Yes	No
AWS::Cognito::UserPoolUser	Yes	No	Yes
AWS::Cognito::UserPoolUserToGroupAttachment	Yes	Yes	No
AWS::Comprehend::DocumentClassifier	Yes	Yes	Yes
AWS::Comprehend::Flywheel	Yes	Yes	Yes
AWS::ComputeOptimizer::AutomationRule	Yes	Yes	No
AWS::Config::AggregationAuthorization	Yes	Yes	Yes
AWS::Config::ConfigRule	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Config::ConfigurationAggregator	Yes	Yes	Yes
AWS::Config::ConformancePack	Yes	Yes	Yes
AWS::Config::OrganizationConformancePack	Yes	Yes	Yes
AWS::Config::RemediationConfiguration	Yes	Yes	No
AWS::Config::StoredQuery	Yes	Yes	Yes
AWS::Connect::AgentStatus	Yes	Yes	Yes
AWS::Connect::ApprovedOrigin	Yes	Yes	Yes
AWS::Connect::ContactFlow	Yes	Yes	Yes
AWS::Connect::ContactFlowModule	Yes	Yes	Yes
AWS::Connect::ContactFlowModuleAlias	Yes	Yes	No
AWS::Connect::ContactFlowModuleVersion	Yes	Yes	No
AWS::Connect::ContactFlowVersion	Yes	Yes	Yes
AWS::Connect::DataTable	Yes	Yes	Yes
AWS::Connect::DataTableAttribute	Yes	Yes	No
AWS::Connect::DataTableRecord	Yes	Yes	No
AWS::Connect::EmailAddress	Yes	Yes	Yes
AWS::Connect::EvaluationForm	Yes	Yes	Yes
AWS::Connect::HoursOfOperation	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Connect::Instance	Yes	Yes	Yes
AWS::Connect::InstanceStorageConfig	Yes	Yes	Yes
AWS::Connect::IntegrationAssociation	Yes	Yes	Yes
AWS::Connect::Notification	Yes	Yes	No
AWS::Connect::PhoneNumber	Yes	Yes	Yes
AWS::Connect::PredefinedAttribute	Yes	Yes	Yes
AWS::Connect::Prompt	Yes	Yes	Yes
AWS::Connect::Queue	Yes	Yes	Yes
AWS::Connect::QuickConnect	Yes	Yes	Yes
AWS::Connect::RoutingProfile	Yes	Yes	Yes
AWS::Connect::Rule	Yes	Yes	No
AWS::Connect::SecurityKey	Yes	Yes	Yes
AWS::Connect::SecurityProfile	Yes	Yes	Yes
AWS::Connect::TaskTemplate	Yes	Yes	Yes
AWS::Connect::TrafficDistributionGroup	Yes	Yes	Yes
AWS::Connect::User	Yes	Yes	Yes
AWS::Connect::UserHierarchyGroup	Yes	Yes	Yes
AWS::Connect::UserHierarchyStructure	Yes	Yes	No
AWS::Connect::View	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Connect::ViewVersion	Yes	Yes	Yes
AWS::Connect::Workspace	Yes	Yes	No
AWS::ConnectCampaigns::Campaign	Yes	No	Yes
AWS::ConnectCampaignsV2::Campaign	Yes	Yes	Yes
AWS::ControlTower::EnabledBaseline	Yes	Yes	Yes
AWS::ControlTower::EnabledControl	Yes	Yes	Yes
AWS::ControlTower::LandingZone	Yes	Yes	Yes
AWS::CustomerProfiles::CalculatedAttributeDefinition	Yes	Yes	Yes
AWS::CustomerProfiles::Domain	Yes	Yes	Yes
AWS::CustomerProfiles::DomainObjectType	Yes	Yes	No
AWS::CustomerProfiles::EventStream	Yes	Yes	Yes
AWS::CustomerProfiles::EventTrigger	Yes	Yes	Yes
AWS::CustomerProfiles::Integration	Yes	Yes	Yes
AWS::CustomerProfiles::ObjectType	Yes	Yes	Yes
AWS::CustomerProfiles::Recommender	Yes	Yes	No
AWS::CustomerProfiles::SegmentDefinition	Yes	Yes	Yes
AWS::DMS::DataMigration	Yes	Yes	Yes
AWS::DMS::DataProvider	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::DMS::InstanceProfile	Yes	Yes	Yes
AWS::DMS::MigrationProject	Yes	Yes	Yes
AWS::DMS::ReplicationConfig	Yes	Yes	Yes
AWS::DSQL::Cluster	Yes	Yes	Yes
AWS::DataBrew::Dataset	Yes	Yes	Yes
AWS::DataBrew::Job	Yes	Yes	Yes
AWS::DataBrew::Project	Yes	Yes	Yes
AWS::DataBrew::Recipe	Yes	Yes	Yes
AWS::DataBrew::Ruleset	Yes	Yes	Yes
AWS::DataBrew::Schedule	Yes	Yes	Yes
AWS::DataPipeline::Pipeline	Yes	Yes	Yes
AWS::DataSync::Agent	Yes	Yes	Yes
AWS::DataSync::LocationAzureBlob	Yes	Yes	Yes
AWS::DataSync::LocationEFS	Yes	Yes	Yes
AWS::DataSync::LocationFSxLustre	Yes	Yes	Yes
AWS::DataSync::LocationFSxONTAP	Yes	Yes	Yes
AWS::DataSync::LocationFSxOpenZFS	Yes	Yes	Yes
AWS::DataSync::LocationFSxWindows	Yes	Yes	Yes
AWS::DataSync::LocationHDFS	Yes	Yes	Yes
AWS::DataSync::LocationNFS	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::DataSync::LocationObjectStorage	Yes	Yes	Yes
AWS::DataSync::LocationS3	Yes	Yes	Yes
AWS::DataSync::LocationSMB	Yes	Yes	Yes
AWS::DataSync::Task	Yes	Yes	Yes
AWS::DataZone::Connection	Yes	Yes	Yes
AWS::DataZone::DataSource	Yes	Yes	Yes
AWS::DataZone::Domain	Yes	Yes	Yes
AWS::DataZone::DomainUnit	Yes	Yes	Yes
AWS::DataZone::Environment	Yes	Yes	Yes
AWS::DataZone::EnvironmentActions	Yes	Yes	Yes
AWS::DataZone::EnvironmentBlueprintConfiguration	Yes	Yes	Yes
AWS::DataZone::EnvironmentProfile	Yes	Yes	Yes
AWS::DataZone::FormType	Yes	Yes	No
AWS::DataZone::GroupProfile	Yes	Yes	Yes
AWS::DataZone::Owner	Yes	Yes	Yes
AWS::DataZone::PolicyGrant	Yes	Yes	Yes
AWS::DataZone::Project	Yes	Yes	Yes
AWS::DataZone::ProjectMembership	Yes	Yes	Yes
AWS::DataZone::ProjectProfile	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::DataZone::SubscriptionTarget	Yes	Yes	Yes
AWS::DataZone::UserProfile	Yes	Yes	Yes
AWS::Deadline::Farm	Yes	Yes	Yes
AWS::Deadline::Fleet	Yes	Yes	Yes
AWS::Deadline::LicenseEndpoint	Yes	Yes	Yes
AWS::Deadline::Limit	Yes	Yes	Yes
AWS::Deadline::MeteredProduct	Yes	Yes	Yes
AWS::Deadline::Monitor	Yes	Yes	Yes
AWS::Deadline::Queue	Yes	Yes	Yes
AWS::Deadline::QueueEnvironment	Yes	Yes	Yes
AWS::Deadline::QueueFleetAssociation	Yes	Yes	Yes
AWS::Deadline::QueueLimitAssociation	Yes	Yes	Yes
AWS::Deadline::StorageProfile	Yes	Yes	Yes
AWS::Detective::Graph	Yes	Yes	Yes
AWS::Detective::MemberInvitation	Yes	Yes	Yes
AWS::Detective::OrganizationAdmin	Yes	Yes	Yes
AWS::DevOpsAgent::AgentSpace	Yes	Yes	Yes
AWS::DevOpsAgent::Association	Yes	Yes	Yes
AWS::DevOpsAgent::PrivateConnection	Yes	Yes	No
AWS::DevOpsAgent::Service	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::DevOpsGuru::LogAnomalyDetectionIntegration	Yes	Yes	Yes
AWS::DevOpsGuru::NotificationChannel	Yes	Yes	Yes
AWS::DevOpsGuru::ResourceCollection	Yes	Yes	Yes
AWS::DeviceFarm::DevicePool	Yes	Yes	Yes
AWS::DeviceFarm::InstanceProfile	Yes	Yes	Yes
AWS::DeviceFarm::NetworkProfile	Yes	Yes	Yes
AWS::DeviceFarm::Project	Yes	Yes	Yes
AWS::DeviceFarm::TestGridProject	Yes	Yes	Yes
AWS::DeviceFarm::VPCEConfiguration	Yes	Yes	Yes
AWS::DirectConnect::Connection	Yes	Yes	No
AWS::DirectConnect::DirectConnectGateway	Yes	Yes	No
AWS::DirectConnect::DirectConnectGatewayAssociation	Yes	Yes	No
AWS::DirectConnect::Lag	Yes	Yes	No
AWS::DirectConnect::PrivateVirtualInterface	Yes	Yes	No
AWS::DirectConnect::PublicVirtualInterface	Yes	Yes	No
AWS::DirectConnect::TransitVirtualInterface	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::DirectoryService::SimpleAD	Yes	Yes	Yes
AWS::DocDB::GlobalCluster	Yes	Yes	No
AWS::DocDBElastic::Cluster	Yes	Yes	Yes
AWS::DynamoDB::GlobalTable	Yes	Yes	Yes
AWS::DynamoDB::Table	Yes	Yes	Yes
AWS::EC2::CapacityManagerDataExport	Yes	Yes	Yes
AWS::EC2::CapacityReservation	Yes	Yes	Yes
AWS::EC2::CapacityReservationFleet	Yes	Yes	Yes
AWS::EC2::CarrierGateway	Yes	Yes	Yes
AWS::EC2::CustomerGateway	Yes	Yes	Yes
AWS::EC2::DHCPOptions	Yes	Yes	Yes
AWS::EC2::EC2Fleet	Yes	Yes	Yes
AWS::EC2::EIP	Yes	Yes	Yes
AWS::EC2::EIPAssociation	Yes	No	Yes
AWS::EC2::EgressOnlyInternetGateway	Yes	Yes	Yes
AWS::EC2::EnclaveCertificateIamRoleAssociation	Yes	Yes	Yes
AWS::EC2::FlowLog	Yes	Yes	Yes
AWS::EC2::GatewayRouteTableAssociation	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::EC2::Host	Yes	Yes	Yes
AWS::EC2::IPAM	Yes	Yes	Yes
AWS::EC2::IPAMAllocation	Yes	Yes	Yes
AWS::EC2::IPAMPool	Yes	Yes	Yes
AWS::EC2::IPAMPoolCidr	Yes	Yes	Yes
AWS::EC2::IPAMPrefixListResolver	Yes	Yes	No
AWS::EC2::IPAMPrefixListResolverTarget	Yes	Yes	No
AWS::EC2::IPAMResourceDiscovery	Yes	Yes	Yes
AWS::EC2::IPAMResourceDiscoveryAssociation	Yes	Yes	Yes
AWS::EC2::IPAMScope	Yes	Yes	Yes
AWS::EC2::Instance	Yes	Yes	Yes
AWS::EC2::InstanceConnectEndpoint	Yes	Yes	Yes
AWS::EC2::InternetGateway	Yes	Yes	Yes
AWS::EC2::IpPoolRouteTableAssociation	Yes	Yes	Yes
AWS::EC2::KeyPair	Yes	Yes	Yes
AWS::EC2::LaunchTemplate	Yes	Yes	Yes
AWS::EC2::LocalGatewayRoute	Yes	Yes	Yes
AWS::EC2::LocalGatewayRouteTable	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::EC2::LocalGatewayRouteTableVPCAssociation	Yes	Yes	Yes
AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation	Yes	Yes	Yes
AWS::EC2::LocalGatewayVirtualInterface	Yes	Yes	Yes
AWS::EC2::LocalGatewayVirtualInterfaceGroup	Yes	Yes	Yes
AWS::EC2::NatGateway	Yes	Yes	Yes
AWS::EC2::NetworkAcl	Yes	Yes	Yes
AWS::EC2::NetworkInsightsAccessScope	Yes	Yes	Yes
AWS::EC2::NetworkInsightsAccessScopeAnalysis	Yes	Yes	Yes
AWS::EC2::NetworkInsightsAnalysis	Yes	Yes	Yes
AWS::EC2::NetworkInsightsPath	Yes	Yes	Yes
AWS::EC2::NetworkInterface	Yes	Yes	Yes
AWS::EC2::NetworkInterfaceAttachment	Yes	Yes	Yes
AWS::EC2::NetworkPerformanceMetricSubscription	Yes	Yes	Yes
AWS::EC2::PlacementGroup	Yes	Yes	Yes
AWS::EC2::PrefixList	Yes	Yes	Yes
AWS::EC2::Route	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::EC2::RouteServer	Yes	Yes	Yes
AWS::EC2::RouteServerAssociation	Yes	Yes	Yes
AWS::EC2::RouteServerEndpoint	Yes	Yes	Yes
AWS::EC2::RouteServerPeer	Yes	Yes	Yes
AWS::EC2::RouteServerPropagation	Yes	Yes	Yes
AWS::EC2::RouteTable	Yes	Yes	Yes
AWS::EC2::SecurityGroup	Yes	Yes	Yes
AWS::EC2::SecurityGroupEgress	Yes	No	Yes
AWS::EC2::SecurityGroupIngress	Yes	No	Yes
AWS::EC2::SecurityGroupVpcAssociation	Yes	Yes	Yes
AWS::EC2::SnapshotBlockPublicAccess	Yes	Yes	Yes
AWS::EC2::SpotFleet	Yes	Yes	Yes
AWS::EC2::SqlHaStandbyDetectedInstance	Yes	Yes	No
AWS::EC2::Subnet	Yes	Yes	Yes
AWS::EC2::SubnetCidrBlock	Yes	No	Yes
AWS::EC2::SubnetNetworkAclAssociation	Yes	Yes	Yes
AWS::EC2::SubnetRouteTableAssociation	Yes	No	Yes

Resource	Import	Drift detection	IaC generator
AWS::EC2::TrafficMirrorFilter	Yes	Yes	Yes
AWS::EC2::TrafficMirrorFilterRule	Yes	Yes	Yes
AWS::EC2::TrafficMirrorSession	Yes	Yes	Yes
AWS::EC2::TrafficMirrorTarget	Yes	Yes	Yes
AWS::EC2::TransitGateway	Yes	Yes	Yes
AWS::EC2::TransitGatewayAttachment	Yes	Yes	Yes
AWS::EC2::TransitGatewayConnect	Yes	Yes	Yes
AWS::EC2::TransitGatewayConnectPeer	Yes	Yes	Yes
AWS::EC2::TransitGatewayMeteringPolicy	Yes	Yes	Yes
AWS::EC2::TransitGatewayMeteringPolicyEntry	Yes	Yes	Yes
AWS::EC2::TransitGatewayMulticastDomain	Yes	Yes	Yes
AWS::EC2::TransitGatewayMulticastDomainAssociation	Yes	Yes	Yes
AWS::EC2::TransitGatewayMulticastGroupMember	Yes	Yes	Yes
AWS::EC2::TransitGatewayMulticastGroupSource	Yes	Yes	Yes
AWS::EC2::TransitGatewayPeeringAttachment	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::EC2::TransitGatewayRoute	Yes	Yes	Yes
AWS::EC2::TransitGatewayRouteTable	Yes	Yes	Yes
AWS::EC2::TransitGatewayRouteTableAssociation	Yes	Yes	Yes
AWS::EC2::TransitGatewayRouteTablePropagation	Yes	Yes	Yes
AWS::EC2::TransitGatewayVpcAttachment	Yes	Yes	Yes
AWS::EC2::VPC	Yes	Yes	Yes
AWS::EC2::VPCBlockPublicAccessExclusion	Yes	Yes	Yes
AWS::EC2::VPCBlockPublicAccessOptions	Yes	Yes	No
AWS::EC2::VPCCidrBlock	Yes	Yes	Yes
AWS::EC2::VPCDHCPOptionsAssociation	Yes	Yes	Yes
AWS::EC2::VPCEncryptionControl	Yes	Yes	Yes
AWS::EC2::VPCEndpoint	Yes	No	Yes
AWS::EC2::VPCEndpointConnectionNotification	Yes	Yes	Yes
AWS::EC2::VPCEndpointService	Yes	Yes	Yes
AWS::EC2::VPCEndpointServicePermissions	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::EC2::VPCGatewayAttachment	Yes	No	Yes
AWS::EC2::VPCPeeringConnection	Yes	Yes	Yes
AWS::EC2::VPNConcentrator	Yes	Yes	Yes
AWS::EC2::VPNConnection	Yes	Yes	Yes
AWS::EC2::VPNConnectionRoute	Yes	Yes	Yes
AWS::EC2::VPNGateway	Yes	Yes	Yes
AWS::EC2::VerifiedAccessEndpoint	Yes	Yes	Yes
AWS::EC2::VerifiedAccessGroup	Yes	Yes	Yes
AWS::EC2::VerifiedAccessInstance	Yes	Yes	Yes
AWS::EC2::VerifiedAccessTrustProvider	Yes	Yes	Yes
AWS::EC2::Volume	Yes	Yes	Yes
AWS::EC2::VolumeAttachment	Yes	Yes	Yes
AWS::ECR::PublicRepository	Yes	Yes	Yes
AWS::ECR::PullThroughCacheRule	Yes	Yes	Yes
AWS::ECR::PullTimeUpdateExclusion	Yes	Yes	Yes
AWS::ECR::RegistryPolicy	Yes	Yes	Yes
AWS::ECR::RegistryScanningConfiguration	Yes	Yes	Yes
AWS::ECR::ReplicationConfiguration	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ECR::Repository	Yes	Yes	Yes
AWS::ECR::RepositoryCreationTemplate	Yes	Yes	Yes
AWS::ECR::SigningConfiguration	Yes	Yes	Yes
AWS::ECS::CapacityProvider	Yes	Yes	Yes
AWS::ECS::Cluster	Yes	Yes	Yes
AWS::ECS::ClusterCapacityProviderAssociations	Yes	No	Yes
AWS::ECS::Daemon	Yes	Yes	No
AWS::ECS::DaemonTaskDefinition	Yes	Yes	No
AWS::ECS::ExpressGatewayService	Yes	Yes	No
AWS::ECS::PrimaryTaskSet	Yes	Yes	No
AWS::ECS::Service	Yes	Yes	Yes
AWS::ECS::TaskDefinition	Yes	Yes	Yes
AWS::ECS::TaskSet	Yes	Yes	No
AWS::EFS::AccessPoint	Yes	Yes	Yes
AWS::EFS::FileSystem	Yes	Yes	Yes
AWS::EFS::MountTarget	Yes	Yes	Yes
AWS::EKS::AccessEntry	Yes	Yes	Yes
AWS::EKS::Addon	Yes	Yes	Yes
AWS::EKS::Capability	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::EKS::Cluster	Yes	No	Yes
AWS::EKS::FargateProfile	Yes	Yes	Yes
AWS::EKS::IdentityProviderConfig	Yes	Yes	Yes
AWS::EKS::Nodegroup	Yes	No	Yes
AWS::EKS::PodIdentityAssociation	Yes	Yes	Yes
AWS::EMR::SecurityConfiguration	Yes	Yes	Yes
AWS::EMR::Step	Yes	Yes	No
AWS::EMR::Studio	Yes	Yes	Yes
AWS::EMR::StudioSessionMapping	Yes	Yes	Yes
AWS::EMR::WALWorkspace	Yes	Yes	Yes
AWS::EMRContainers::Endpoint	Yes	Yes	No
AWS::EMRContainers::SecurityConfiguration	Yes	Yes	No
AWS::EMRContainers::VirtualCluster	Yes	Yes	Yes
AWS::EMRServerless::Application	Yes	Yes	Yes
AWS::EVS::Environment	Yes	Yes	Yes
AWS::ElastiCache::CacheCluster	Yes	Yes	No
AWS::ElastiCache::GlobalReplicationGroup	Yes	Yes	Yes
AWS::ElastiCache::ParameterGroup	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ElastiCache::ReplicationGroup	Yes	Yes	No
AWS::ElastiCache::ServerlessCache	Yes	Yes	Yes
AWS::ElastiCache::SubnetGroup	Yes	Yes	Yes
AWS::ElastiCache::User	Yes	Yes	Yes
AWS::ElastiCache::UserGroup	Yes	Yes	Yes
AWS::ElasticBeanstalk::Application	Yes	Yes	Yes
AWS::ElasticBeanstalk::ApplicationVersion	Yes	Yes	Yes
AWS::ElasticBeanstalk::ConfigurationTemplate	Yes	Yes	Yes
AWS::ElasticBeanstalk::Environment	Yes	Yes	Yes
AWS::ElasticLoadBalancing::LoadBalancer	Yes	Yes	No
AWS::ElasticLoadBalancingV2::Listener	Yes	Yes	Yes
AWS::ElasticLoadBalancingV2::ListenerRule	Yes	Yes	Yes
AWS::ElasticLoadBalancingV2::LoadBalancer	Yes	Yes	Yes
AWS::ElasticLoadBalancingV2::TargetGroup	Yes	Yes	Yes
AWS::ElasticLoadBalancingV2::TrustStore	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ElasticLoadBalancingV2::TrustStoreRevocation	Yes	Yes	Yes
AWS::ElementalInference::Dictionary	Yes	Yes	No
AWS::ElementalInference::Feed	Yes	Yes	No
AWS::EntityResolution::IdMappingWorkflow	Yes	Yes	Yes
AWS::EntityResolution::IdNamespace	Yes	Yes	Yes
AWS::EntityResolution::MatchingWorkflow	Yes	Yes	Yes
AWS::EntityResolution::PolicyStatement	Yes	Yes	Yes
AWS::EntityResolution::SchemaMapping	Yes	Yes	Yes
AWS::EventSchemas::Discoverer	Yes	Yes	Yes
AWS::EventSchemas::Registry	Yes	Yes	Yes
AWS::EventSchemas::RegistryPolicy	Yes	Yes	No
AWS::EventSchemas::Schema	Yes	Yes	Yes
AWS::Events::ApiDestination	Yes	Yes	Yes
AWS::Events::Archive	Yes	Yes	Yes
AWS::Events::Connection	Yes	Yes	Yes
AWS::Events::Endpoint	Yes	Yes	Yes
AWS::Events::EventBus	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Events::EventBusPolicy	Yes	No	Yes
AWS::Events::Rule	Yes	Yes	Yes
AWS::Evidently::Experiment	Yes	Yes	No
AWS::Evidently::Feature	Yes	No	No
AWS::Evidently::Launch	Yes	Yes	No
AWS::Evidently::Project	Yes	Yes	No
AWS::Evidently::Segment	Yes	Yes	Yes
AWS::FIS::ExperimentTemplate	Yes	Yes	Yes
AWS::FIS::TargetAccountConfiguration	Yes	Yes	Yes
AWS::FMS::NotificationChannel	Yes	Yes	Yes
AWS::FMS::Policy	Yes	No	Yes
AWS::FMS::ResourceSet	Yes	Yes	Yes
AWS::FSx::DataRepositoryAssociation	Yes	Yes	Yes
AWS::FSx::S3AccessPointAttachment	Yes	Yes	Yes
AWS::FinSpace::Environment	Yes	Yes	Yes
AWS::Forecast::Dataset	Yes	Yes	Yes
AWS::Forecast::DatasetGroup	Yes	Yes	Yes
AWS::FraudDetector::Detector	Yes	No	Yes
AWS::FraudDetector::EntityType	Yes	Yes	Yes
AWS::FraudDetector::EventType	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::FraudDetector::Label	Yes	Yes	Yes
AWS::FraudDetector::List	Yes	Yes	Yes
AWS::FraudDetector::Outcome	Yes	Yes	Yes
AWS::FraudDetector::Variable	Yes	Yes	Yes
AWS::GameLift::Alias	Yes	Yes	Yes
AWS::GameLift::Build	Yes	Yes	Yes
AWS::GameLift::ContainerFleet	Yes	Yes	Yes
AWS::GameLift::ContainerGroupDefinition	Yes	Yes	Yes
AWS::GameLift::Fleet	Yes	Yes	Yes
AWS::GameLift::GameServerGroup	Yes	Yes	Yes
AWS::GameLift::GameSessionQueue	Yes	Yes	Yes
AWS::GameLift::Location	Yes	Yes	Yes
AWS::GameLift::MatchmakingConfiguration	Yes	Yes	Yes
AWS::GameLift::MatchmakingRuleSet	Yes	Yes	Yes
AWS::GameLift::Script	Yes	Yes	Yes
AWS::GameLiftStreams::Application	Yes	Yes	No
AWS::GameLiftStreams::StreamGroup	Yes	Yes	No
AWS::GlobalAccelerator::Accelerator	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::GlobalAccelerator::CrossAccountAttachment	Yes	Yes	Yes
AWS::GlobalAccelerator::EndpointGroup	Yes	Yes	Yes
AWS::GlobalAccelerator::Listener	Yes	Yes	Yes
AWS::Glue::Catalog	Yes	Yes	No
AWS::Glue::Crawler	Yes	No	Yes
AWS::Glue::Database	Yes	Yes	Yes
AWS::Glue::IdentityCenterConfiguration	Yes	Yes	No
AWS::Glue::Integration	Yes	Yes	Yes
AWS::Glue::IntegrationResourceProperty	Yes	Yes	Yes
AWS::Glue::Job	Yes	Yes	Yes
AWS::Glue::Registry	Yes	Yes	Yes
AWS::Glue::Schema	Yes	Yes	Yes
AWS::Glue::SchemaVersion	Yes	Yes	Yes
AWS::Glue::SchemaVersionMetadata	Yes	Yes	Yes
AWS::Glue::Trigger	Yes	Yes	Yes
AWS::Glue::UsageProfile	Yes	Yes	Yes
AWS::Grafana::Workspace	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::GreengrassV2::ComponentVersion	Yes	Yes	Yes
AWS::GreengrassV2::Deployment	Yes	Yes	Yes
AWS::GroundStation::Config	Yes	Yes	Yes
AWS::GroundStation::DataflowEndpointGroup	Yes	Yes	Yes
AWS::GroundStation::DataflowEndpointGroupV2	Yes	Yes	No
AWS::GroundStation::MissionProfile	Yes	Yes	Yes
AWS::GuardDuty::Detector	Yes	Yes	Yes
AWS::GuardDuty::Filter	Yes	Yes	Yes
AWS::GuardDuty::IPSet	Yes	Yes	Yes
AWS::GuardDuty::MalwareProtectionPlan	Yes	Yes	Yes
AWS::GuardDuty::Master	Yes	Yes	Yes
AWS::GuardDuty::Member	Yes	Yes	Yes
AWS::GuardDuty::PublishingDestination	Yes	Yes	Yes
AWS::GuardDuty::ThreatEntitySet	Yes	Yes	Yes
AWS::GuardDuty::ThreatIntelSet	Yes	Yes	Yes
AWS::GuardDuty::TrustedEntitySet	Yes	Yes	Yes
AWS::HealthImaging::Datastore	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::HealthLake::FHIRDatastore	Yes	Yes	Yes
AWS::IAM::Group	Yes	Yes	Yes
AWS::IAM::GroupPolicy	Yes	Yes	No
AWS::IAM::InstanceProfile	Yes	Yes	Yes
AWS::IAM::ManagedPolicy	Yes	Yes	Yes
AWS::IAM::OIDCProvider	Yes	Yes	Yes
AWS::IAM::Role	Yes	Yes	Yes
AWS::IAM::RolePolicy	Yes	Yes	No
AWS::IAM::SAMLProvider	Yes	Yes	Yes
AWS::IAM::ServerCertificate	Yes	Yes	Yes
AWS::IAM::ServiceLinkedRole	Yes	Yes	No
AWS::IAM::User	Yes	Yes	Yes
AWS::IAM::UserPolicy	Yes	Yes	No
AWS::IAM::VirtualMFADevice	Yes	Yes	Yes
AWS::IVS::Channel	Yes	Yes	Yes
AWS::IVS::EncoderConfiguration	Yes	Yes	Yes
AWS::IVS::IngestConfiguration	Yes	Yes	Yes
AWS::IVS::PlaybackKeyPair	Yes	Yes	Yes
AWS::IVS::PlaybackRestrictionPolicy	Yes	Yes	Yes
AWS::IVS::PublicKey	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::IVS::RecordingConfiguration	Yes	Yes	Yes
AWS::IVS::Stage	Yes	Yes	Yes
AWS::IVS::StorageConfiguration	Yes	Yes	Yes
AWS::IVS::StreamKey	Yes	Yes	Yes
AWS::IVSChat::LoggingConfiguration	Yes	Yes	Yes
AWS::IVSChat::Room	Yes	Yes	Yes
AWS::IdentityStore::Group	Yes	Yes	Yes
AWS::IdentityStore::GroupMembership	Yes	Yes	Yes
AWS::ImageBuilder::Component	Yes	Yes	Yes
AWS::ImageBuilder::ContainerRecipe	Yes	No	Yes
AWS::ImageBuilder::DistributionConfiguration	Yes	Yes	Yes
AWS::ImageBuilder::Image	Yes	Yes	Yes
AWS::ImageBuilder::ImagePipeline	Yes	Yes	Yes
AWS::ImageBuilder::ImageRecipe	Yes	Yes	Yes
AWS::ImageBuilder::InfrastructureConfiguration	Yes	Yes	Yes
AWS::ImageBuilder::LifecyclePolicy	Yes	Yes	Yes
AWS::ImageBuilder::Workflow	Yes	Yes	Yes
AWS::Inspector::AssessmentTarget	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Inspector::AssessmentTemplate	Yes	Yes	Yes
AWS::Inspector::ResourceGroup	Yes	Yes	No
AWS::InspectorV2::CisScanConfiguration	Yes	Yes	Yes
AWS::InspectorV2::CodeSecurityIntegration	Yes	Yes	Yes
AWS::InspectorV2::CodeSecurityScanConfiguration	Yes	Yes	Yes
AWS::InspectorV2::Filter	Yes	Yes	Yes
AWS::Interconnect::Connection	Yes	Yes	No
AWS::InternetMonitor::Monitor	Yes	Yes	Yes
AWS::Invoicing::InvoiceUnit	Yes	Yes	Yes
AWS::IoT::AccountAuditConfiguration	Yes	Yes	Yes
AWS::IoT::Authorizer	Yes	Yes	Yes
AWS::IoT::BillingGroup	Yes	Yes	Yes
AWS::IoT::CACertificate	Yes	Yes	Yes
AWS::IoT::Certificate	Yes	Yes	Yes
AWS::IoT::CertificateProvider	Yes	Yes	Yes
AWS::IoT::Command	Yes	Yes	Yes
AWS::IoT::CustomMetric	Yes	Yes	Yes
AWS::IoT::Dimension	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::IoT::DomainConfiguration	Yes	Yes	Yes
AWS::IoT::EncryptionConfiguration	Yes	Yes	Yes
AWS::IoT::FleetMetric	Yes	Yes	Yes
AWS::IoT::JobTemplate	Yes	Yes	Yes
AWS::IoT::Logging	Yes	Yes	Yes
AWS::IoT::MitigationAction	Yes	Yes	Yes
AWS::IoT::Policy	Yes	No	Yes
AWS::IoT::ProvisioningTemplate	Yes	Yes	Yes
AWS::IoT::ResourceSpecificLogging	Yes	Yes	Yes
AWS::IoT::RoleAlias	Yes	Yes	Yes
AWS::IoT::ScheduledAudit	Yes	Yes	Yes
AWS::IoT::SecurityProfile	Yes	No	Yes
AWS::IoT::SoftwarePackage	Yes	Yes	Yes
AWS::IoT::SoftwarePackageVersion	Yes	Yes	Yes
AWS::IoT::Thing	Yes	Yes	Yes
AWS::IoT::ThingGroup	Yes	Yes	Yes
AWS::IoT::ThingType	Yes	Yes	Yes
AWS::IoT::TopicRule	Yes	Yes	Yes
AWS::IoT::TopicRuleDestination	Yes	Yes	Yes
AWS::IoTAnalytics::Channel	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::IoTAnalytics::Dataset	Yes	Yes	Yes
AWS::IoTAnalytics::Datastore	Yes	Yes	Yes
AWS::IoTAnalytics::Pipeline	Yes	Yes	Yes
AWS::IoTCoreDeviceAdvisor::SuiteDefinition	Yes	Yes	Yes
AWS::IoTEvents::AlarmModel	Yes	Yes	Yes
AWS::IoTEvents::DetectorModel	Yes	Yes	Yes
AWS::IoTEvents::Input	Yes	Yes	Yes
AWS::IoTFleetWise::Campaign	Yes	Yes	Yes
AWS::IoTFleetWise::DecoderManifest	Yes	Yes	Yes
AWS::IoTFleetWise::Fleet	Yes	Yes	Yes
AWS::IoTFleetWise::ModelManifest	Yes	No	Yes
AWS::IoTFleetWise::SignalCatalog	Yes	No	Yes
AWS::IoTFleetWise::StateTemplate	Yes	Yes	Yes
AWS::IoTFleetWise::Vehicle	Yes	Yes	Yes
AWS::IoTSiteWise::AccessPolicy	Yes	Yes	Yes
AWS::IoTSiteWise::Asset	Yes	Yes	Yes
AWS::IoTSiteWise::AssetModel	Yes	Yes	Yes
AWS::IoTSiteWise::ComputationModel	Yes	Yes	Yes
AWS::IoTSiteWise::Dashboard	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::IoTSiteWise::Dataset	Yes	Yes	Yes
AWS::IoTSiteWise::Gateway	Yes	Yes	Yes
AWS::IoTSiteWise::Portal	Yes	Yes	Yes
AWS::IoTSiteWise::Project	Yes	Yes	Yes
AWS::IoTTwinMaker::ComponentType	Yes	Yes	Yes
AWS::IoTTwinMaker::Entity	Yes	Yes	Yes
AWS::IoTTwinMaker::Scene	Yes	Yes	Yes
AWS::IoTTwinMaker::SyncJob	Yes	Yes	Yes
AWS::IoTTwinMaker::Workspace	Yes	Yes	Yes
AWS::IoTWireless::Destination	Yes	Yes	Yes
AWS::IoTWireless::DeviceProfile	Yes	Yes	Yes
AWS::IoTWireless::FuotaTask	Yes	Yes	Yes
AWS::IoTWireless::MulticastGroup	Yes	Yes	Yes
AWS::IoTWireless::NetworkAnalyzerConfiguration	Yes	Yes	Yes
AWS::IoTWireless::PartnerAccount	Yes	Yes	Yes
AWS::IoTWireless::ServiceProfile	Yes	Yes	Yes
AWS::IoTWireless::TaskDefinition	Yes	Yes	Yes
AWS::IoTWireless::WirelessDevice	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::IoTWireless::WirelessDeviceImportTask	Yes	Yes	Yes
AWS::IoTWireless::WirelessGateway	Yes	Yes	Yes
AWS::KMS::Alias	Yes	No	Yes
AWS::KMS::Key	Yes	Yes	Yes
AWS::KMS::ReplicaKey	Yes	No	Yes
AWS::KafkaConnect::Connector	Yes	No	Yes
AWS::KafkaConnect::CustomPlugin	Yes	Yes	Yes
AWS::KafkaConnect::WorkerConfiguration	Yes	Yes	Yes
AWS::Kendra::DataSource	Yes	Yes	Yes
AWS::Kendra::Faq	Yes	Yes	Yes
AWS::Kendra::Index	Yes	Yes	Yes
AWS::KendraRanking::ExecutionPlan	Yes	Yes	Yes
AWS::Kinesis::ResourcePolicy	Yes	Yes	No
AWS::Kinesis::Stream	Yes	Yes	Yes
AWS::Kinesis::StreamConsumer	Yes	Yes	Yes
AWS::KinesisAnalyticsV2::Application	Yes	Yes	Yes
AWS::KinesisFirehose::DeliveryStream	Yes	No	Yes
AWS::KinesisVideo::SignalingChannel	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::KinesisVideo::Stream	Yes	Yes	Yes
AWS::LakeFormation::DataCellsFilter	Yes	Yes	Yes
AWS::LakeFormation::PrincipalPermissions	Yes	Yes	No
AWS::LakeFormation::Tag	Yes	Yes	Yes
AWS::LakeFormation::TagAssociation	Yes	Yes	No
AWS::Lambda::Alias	Yes	Yes	Yes
AWS::Lambda::CapacityProvider	Yes	Yes	Yes
AWS::Lambda::CodeSigningConfig	Yes	Yes	Yes
AWS::Lambda::EventInvokeConfig	Yes	Yes	Yes
AWS::Lambda::EventSourceMapping	Yes	No	Yes
AWS::Lambda::Function	Yes	Yes	Yes
AWS::Lambda::LayerVersion	Yes	Yes	Yes
AWS::Lambda::LayerVersionPermission	Yes	Yes	Yes
AWS::Lambda::Permission	Yes	Yes	Yes
AWS::Lambda::Url	Yes	Yes	Yes
AWS::Lambda::Version	Yes	Yes	Yes
AWS::LaunchWizard::Deployment	Yes	Yes	Yes
AWS::Lex::Bot	Yes	Yes	Yes
AWS::Lex::BotAlias	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Lex::BotVersion	Yes	Yes	Yes
AWS::Lex::ResourcePolicy	Yes	Yes	Yes
AWS::LicenseManager::Grant	Yes	Yes	Yes
AWS::LicenseManager::License	Yes	Yes	Yes
AWS::Lightsail::Alarm	Yes	Yes	Yes
AWS::Lightsail::Bucket	Yes	Yes	Yes
AWS::Lightsail::Certificate	Yes	Yes	Yes
AWS::Lightsail::Container	Yes	Yes	Yes
AWS::Lightsail::Database	Yes	Yes	Yes
AWS::Lightsail::DatabaseSnapshot	Yes	Yes	No
AWS::Lightsail::Disk	Yes	Yes	Yes
AWS::Lightsail::DiskSnapshot	Yes	Yes	Yes
AWS::Lightsail::Distribution	Yes	Yes	Yes
AWS::Lightsail::Domain	Yes	Yes	Yes
AWS::Lightsail::Instance	Yes	Yes	Yes
AWS::Lightsail::InstanceSnapshot	Yes	Yes	Yes
AWS::Lightsail::LoadBalancer	Yes	Yes	Yes
AWS::Lightsail::LoadBalancerTlsCertificate	Yes	Yes	Yes
AWS::Lightsail::StaticIp	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Location::APIKey	Yes	Yes	Yes
AWS::Location::GeofenceCollection	Yes	Yes	Yes
AWS::Location::Map	Yes	Yes	Yes
AWS::Location::PlaceIndex	Yes	Yes	Yes
AWS::Location::RouteCalculator	Yes	Yes	Yes
AWS::Location::Tracker	Yes	Yes	Yes
AWS::Location::TrackerConsumer	Yes	Yes	Yes
AWS::Logs::AccountPolicy	Yes	Yes	Yes
AWS::Logs::Delivery	Yes	Yes	Yes
AWS::Logs::DeliveryDestination	Yes	Yes	Yes
AWS::Logs::DeliverySource	Yes	Yes	Yes
AWS::Logs::Destination	Yes	Yes	Yes
AWS::Logs::Integration	Yes	Yes	Yes
AWS::Logs::LogAnomalyDetector	Yes	Yes	Yes
AWS::Logs::LogGroup	Yes	Yes	Yes
AWS::Logs::LogStream	Yes	Yes	Yes
AWS::Logs::MetricFilter	Yes	Yes	Yes
AWS::Logs::QueryDefinition	Yes	Yes	Yes
AWS::Logs::ResourcePolicy	Yes	Yes	Yes
AWS::Logs::ScheduledQuery	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::Logs::SubscriptionFilter	Yes	Yes	Yes
AWS::Logs::Transformer	Yes	Yes	Yes
AWS::LookoutEquipment::InferenceScheduler	Yes	Yes	Yes
AWS::LookoutVision::Project	Yes	Yes	Yes
AWS::M2::Application	Yes	Yes	Yes
AWS::M2::Deployment	Yes	Yes	Yes
AWS::M2::Environment	Yes	Yes	Yes
AWS::MPA::ApprovalTeam	Yes	Yes	Yes
AWS::MPA::IdentitySource	Yes	Yes	Yes
AWS::MSK::BatchScramSecret	Yes	Yes	Yes
AWS::MSK::Cluster	Yes	Yes	Yes
AWS::MSK::ClusterPolicy	Yes	Yes	Yes
AWS::MSK::Configuration	Yes	Yes	Yes
AWS::MSK::Replicator	Yes	Yes	Yes
AWS::MSK::ServerlessCluster	Yes	Yes	Yes
AWS::MSK::Topic	Yes	Yes	No
AWS::MSK::VpcConnection	Yes	Yes	Yes
AWS::MWAA::Environment	Yes	Yes	Yes
AWS::MWAAServerless::Workflow	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::Macie::AllowList	Yes	Yes	Yes
AWS::Macie::CustomDataIdentifier	Yes	Yes	Yes
AWS::Macie::FindingsFilter	Yes	Yes	Yes
AWS::Macie::Session	Yes	Yes	Yes
AWS::ManagedBlockchain::Accessor	Yes	Yes	Yes
AWS::MediaConnect::Bridge	Yes	Yes	Yes
AWS::MediaConnect::BridgeOutput	Yes	Yes	No
AWS::MediaConnect::BridgeSource	Yes	Yes	No
AWS::MediaConnect::Flow	Yes	Yes	Yes
AWS::MediaConnect::FlowEntitlement	Yes	Yes	Yes
AWS::MediaConnect::FlowOutput	Yes	Yes	Yes
AWS::MediaConnect::FlowSource	Yes	Yes	Yes
AWS::MediaConnect::FlowVpcInterface	Yes	Yes	Yes
AWS::MediaConnect::Gateway	Yes	Yes	Yes
AWS::MediaConnect::RouterInput	Yes	Yes	Yes
AWS::MediaConnect::RouterNetworkInterface	Yes	Yes	Yes
AWS::MediaConnect::RouterOutput	Yes	Yes	Yes
AWS::MediaLive::ChannelPlacementGroup	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::MediaLive::CloudWatchAlarmTemplate	Yes	Yes	Yes
AWS::MediaLive::CloudWatchAlarmTemplateGroup	Yes	Yes	Yes
AWS::MediaLive::Cluster	Yes	Yes	Yes
AWS::MediaLive::EventBridgeRuleTemplate	Yes	Yes	Yes
AWS::MediaLive::EventBridgeRuleTemplateGroup	Yes	Yes	Yes
AWS::MediaLive::Multiplex	Yes	Yes	Yes
AWS::MediaLive::Multiplexprogram	Yes	Yes	Yes
AWS::MediaLive::Network	Yes	Yes	Yes
AWS::MediaLive::SdiSource	Yes	Yes	Yes
AWS::MediaLive::SignalMap	Yes	Yes	Yes
AWS::MediaPackage::Asset	Yes	Yes	Yes
AWS::MediaPackage::Channel	Yes	Yes	Yes
AWS::MediaPackage::OriginEndpoint	Yes	Yes	Yes
AWS::MediaPackage::PackagingConfiguration	Yes	Yes	Yes
AWS::MediaPackage::PackagingGroup	Yes	Yes	Yes
AWS::MediaPackageV2::Channel	Yes	Yes	Yes
AWS::MediaPackageV2::ChannelGroup	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::MediaPackageV2::ChannelPolicy	Yes	Yes	No
AWS::MediaPackageV2::OriginEndpoint	Yes	Yes	Yes
AWS::MediaPackageV2::OriginEndpointPolicy	Yes	Yes	No
AWS::MediaTailor::Channel	Yes	Yes	Yes
AWS::MediaTailor::ChannelPolicy	Yes	Yes	No
AWS::MediaTailor::LiveSource	Yes	Yes	Yes
AWS::MediaTailor::PlaybackConfiguration	Yes	Yes	Yes
AWS::MediaTailor::SourceLocation	Yes	Yes	Yes
AWS::MediaTailor::VodSource	Yes	Yes	Yes
AWS::MemoryDB::ACL	Yes	Yes	Yes
AWS::MemoryDB::Cluster	Yes	Yes	Yes
AWS::MemoryDB::MultiRegionCluster	Yes	Yes	Yes
AWS::MemoryDB::ParameterGroup	Yes	Yes	Yes
AWS::MemoryDB::SubnetGroup	Yes	Yes	Yes
AWS::MemoryDB::User	Yes	Yes	Yes
AWS::Neptune::DBCluster	Yes	Yes	Yes
AWS::Neptune::DBClusterParameterGroup	Yes	Yes	Yes
AWS::Neptune::DBInstance	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Neptune::DBParameterGroup	Yes	Yes	Yes
AWS::Neptune::DBSubnetGroup	Yes	Yes	Yes
AWS::Neptune::EventSubscription	Yes	Yes	Yes
AWS::NeptuneGraph::Graph	Yes	Yes	Yes
AWS::NeptuneGraph::GraphSnapshot	Yes	Yes	No
AWS::NeptuneGraph::PrivateGraphEndpoint	Yes	Yes	Yes
AWS::NetworkFirewall::Firewall	Yes	Yes	Yes
AWS::NetworkFirewall::FirewallPolicy	Yes	Yes	Yes
AWS::NetworkFirewall::LoggingConfiguration	Yes	Yes	No
AWS::NetworkFirewall::RuleGroup	Yes	Yes	Yes
AWS::NetworkFirewall::TLSInspectionConfiguration	Yes	Yes	Yes
AWS::NetworkFirewall::VpcEndpointAssociation	Yes	Yes	Yes
AWS::NetworkManager::ConnectAttachment	Yes	Yes	Yes
AWS::NetworkManager::ConnectPeer	Yes	Yes	Yes
AWS::NetworkManager::CoreNetwork	Yes	Yes	Yes
AWS::NetworkManager::CoreNetworkPrefixListAssociation	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::NetworkManager::CustomerGatewayAssociation	Yes	Yes	Yes
AWS::NetworkManager::Device	Yes	Yes	Yes
AWS::NetworkManager::DirectConnectGatewayAttachment	Yes	Yes	Yes
AWS::NetworkManager::GlobalNetwork	Yes	Yes	Yes
AWS::NetworkManager::Link	Yes	Yes	Yes
AWS::NetworkManager::LinkAssociation	Yes	Yes	Yes
AWS::NetworkManager::Site	Yes	Yes	Yes
AWS::NetworkManager::SiteToSiteVpnAttachment	Yes	Yes	Yes
AWS::NetworkManager::TransitGatewayPeering	Yes	Yes	Yes
AWS::NetworkManager::TransitGatewayRegistration	Yes	Yes	Yes
AWS::NetworkManager::TransitGatewayRouteTableAttachment	Yes	Yes	Yes
AWS::NetworkManager::VpcAttachment	Yes	Yes	Yes
AWS::Notifications::ChannelAssociation	Yes	Yes	Yes
AWS::Notifications::EventRule	Yes	Yes	Yes
AWS::Notifications::ManagedNotificationAccountContactAssociation	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Notifications::ManagedNotificationAdditionalChannelAssociation	Yes	Yes	Yes
AWS::Notifications::NotificationConfiguration	Yes	Yes	Yes
AWS::Notifications::NotificationHub	Yes	Yes	Yes
AWS::Notifications::OrganizationalUnitAssociation	Yes	Yes	Yes
AWS::NotificationsContacts::EmailContact	Yes	Yes	Yes
AWS::NovaAct::WorkflowDefinition	Yes	Yes	No
AWS::ODB::CloudAutonomousVmCluster	Yes	Yes	Yes
AWS::ODB::CloudExadataInfrastructure	Yes	Yes	Yes
AWS::ODB::CloudVmCluster	Yes	Yes	Yes
AWS::ODB::OdbNetwork	Yes	Yes	Yes
AWS::ODB::OdbPeeringConnection	Yes	Yes	Yes
AWS::OSIS::Pipeline	Yes	Yes	Yes
AWS::Oam::Link	Yes	Yes	Yes
AWS::Oam::Sink	Yes	Yes	Yes
AWS::ObservabilityAdmin::OrganizationCentralizationRule	Yes	Yes	Yes
AWS::ObservabilityAdmin::OrganizationTelemetryRule	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ObservabilityAdmin::S3TableIntegration	Yes	Yes	Yes
AWS::ObservabilityAdmin::TelemetryEnrichment	Yes	Yes	No
AWS::ObservabilityAdmin::TelemetryPipelines	Yes	Yes	Yes
AWS::ObservabilityAdmin::TelemetryRule	Yes	Yes	Yes
AWS::Omics::AnnotationStore	Yes	Yes	Yes
AWS::Omics::Configuration	Yes	Yes	No
AWS::Omics::ReferenceStore	Yes	Yes	Yes
AWS::Omics::RunGroup	Yes	Yes	Yes
AWS::Omics::SequenceStore	Yes	Yes	Yes
AWS::Omics::VariantStore	Yes	Yes	Yes
AWS::Omics::Workflow	Yes	Yes	Yes
AWS::Omics::WorkflowVersion	Yes	Yes	Yes
AWS::OpenSearchServerless::AccessPolicy	Yes	Yes	Yes
AWS::OpenSearchServerless::Collection	Yes	Yes	Yes
AWS::OpenSearchServerless::CollectionGroup	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::OpenSearchServerless::CollectionIndex	Yes	Yes	No
AWS::OpenSearchServerless::Index	Yes	Yes	Yes
AWS::OpenSearchServerless::LifecyclePolicy	Yes	Yes	Yes
AWS::OpenSearchServerless::SecurityConfig	Yes	Yes	Yes
AWS::OpenSearchServerless::SecurityPolicy	Yes	Yes	Yes
AWS::OpenSearchServerless::VpcEndpoint	Yes	Yes	Yes
AWS::OpenSearchService::Application	Yes	Yes	Yes
AWS::OpenSearchService::Domain	Yes	Yes	No
AWS::Organizations::Account	Yes	Yes	Yes
AWS::Organizations::Organization	Yes	Yes	Yes
AWS::Organizations::OrganizationalUnit	Yes	Yes	Yes
AWS::Organizations::Policy	Yes	Yes	Yes
AWS::Organizations::ResourcePolicy	Yes	Yes	Yes
AWS::PCAConnectorAD::Connector	Yes	Yes	Yes
AWS::PCAConnectorAD::DirectoryRegistration	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::PCAConnectorAD::ServicePrincipalName	Yes	Yes	Yes
AWS::PCAConnectorAD::Template	Yes	Yes	Yes
AWS::PCAConnectorAD::TemplateGroupAccessControlEntry	Yes	Yes	Yes
AWS::PCAConnectorSCEP::Challenge	Yes	Yes	Yes
AWS::PCAConnectorSCEP::Connector	Yes	Yes	Yes
AWS::PCS::Cluster	Yes	Yes	Yes
AWS::PCS::ComputeNodeGroup	Yes	Yes	Yes
AWS::PCS::Queue	Yes	Yes	Yes
AWS::Panorama::ApplicationInstance	Yes	Yes	Yes
AWS::Panorama::Package	Yes	Yes	Yes
AWS::Panorama::PackageVersion	Yes	Yes	No
AWS::PaymentCryptography::Alias	Yes	Yes	Yes
AWS::PaymentCryptography::Key	Yes	Yes	Yes
AWS::Personalize::Dataset	Yes	Yes	Yes
AWS::Personalize::DatasetGroup	Yes	Yes	Yes
AWS::Personalize::Schema	Yes	Yes	Yes
AWS::Personalize::Solution	Yes	Yes	Yes
AWS::Pinpoint::InAppTemplate	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Pipes::Pipe	Yes	Yes	Yes
AWS::Proton::EnvironmentAccountConnection	Yes	Yes	Yes
AWS::Proton::EnvironmentTemplate	Yes	Yes	Yes
AWS::Proton::ServiceTemplate	Yes	Yes	Yes
AWS::QBusiness::Application	Yes	Yes	Yes
AWS::QBusiness::DataAccessor	Yes	Yes	Yes
AWS::QBusiness::DataSource	Yes	Yes	Yes
AWS::QBusiness::Index	Yes	Yes	Yes
AWS::QBusiness::Permission	Yes	Yes	Yes
AWS::QBusiness::Plugin	Yes	Yes	Yes
AWS::QBusiness::Retriever	Yes	Yes	Yes
AWS::QBusiness::WebExperience	Yes	Yes	Yes
AWS::QLDB::Stream	Yes	Yes	Yes
AWS::QuickSight::ActionConnector	Yes	Yes	No
AWS::QuickSight::Analysis	Yes	Yes	Yes
AWS::QuickSight::CustomPermissions	Yes	Yes	Yes
AWS::QuickSight::Dashboard	Yes	Yes	Yes
AWS::QuickSight::DataSet	Yes	Yes	Yes
AWS::QuickSight::DataSource	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::QuickSight::Folder	Yes	Yes	Yes
AWS::QuickSight::RefreshSchedule	Yes	Yes	Yes
AWS::QuickSight::Template	Yes	Yes	Yes
AWS::QuickSight::Theme	Yes	Yes	Yes
AWS::QuickSight::Topic	Yes	Yes	Yes
AWS::QuickSight::VPCConnection	Yes	Yes	Yes
AWS::RAM::Permission	Yes	Yes	Yes
AWS::RAM::ResourceShare	Yes	Yes	Yes
AWS::RDS::CustomDBEngineVersion	Yes	Yes	Yes
AWS::RDS::DBCluster	Yes	Yes	Yes
AWS::RDS::DBClusterParameterGroup	Yes	Yes	Yes
AWS::RDS::DBInstance	Yes	Yes	Yes
AWS::RDS::DBParameterGroup	Yes	Yes	Yes
AWS::RDS::DBProxy	Yes	Yes	Yes
AWS::RDS::DBProxyEndpoint	Yes	Yes	Yes
AWS::RDS::DBProxyTargetGroup	Yes	Yes	Yes
AWS::RDS::DBShardGroup	Yes	Yes	Yes
AWS::RDS::DBSubnetGroup	Yes	Yes	Yes
AWS::RDS::EventSubscription	Yes	Yes	Yes
AWS::RDS::GlobalCluster	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::RDS::Integration	Yes	Yes	Yes
AWS::RDS::OptionGroup	Yes	Yes	Yes
AWS::RTBFabric::InboundExternalLink	Yes	Yes	Yes
AWS::RTBFabric::Link	Yes	Yes	Yes
AWS::RTBFabric::LinkRoutingRule	Yes	Yes	No
AWS::RTBFabric::OutboundExternalLink	Yes	Yes	Yes
AWS::RTBFabric::RequesterGateway	Yes	Yes	Yes
AWS::RTBFabric::ResponderGateway	Yes	Yes	Yes
AWS::RUM::AppMonitor	Yes	Yes	Yes
AWS::Rbin::Rule	Yes	Yes	Yes
AWS::Redshift::Cluster	Yes	Yes	Yes
AWS::Redshift::ClusterParameterGroup	Yes	Yes	Yes
AWS::Redshift::ClusterSubnetGroup	Yes	Yes	Yes
AWS::Redshift::EndpointAccess	Yes	Yes	Yes
AWS::Redshift::EndpointAuthorization	Yes	Yes	Yes
AWS::Redshift::EventSubscription	Yes	Yes	Yes
AWS::Redshift::Integration	Yes	Yes	Yes
AWS::Redshift::ScheduledAction	Yes	Yes	Yes
AWS::RedshiftServerless::Namespace	Yes	Yes	Yes
AWS::RedshiftServerless::Snapshot	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::RedshiftServerless::Workgroup	Yes	Yes	Yes
AWS::RefactorSpaces::Application	Yes	Yes	Yes
AWS::RefactorSpaces::Environment	Yes	Yes	Yes
AWS::RefactorSpaces::Route	Yes	Yes	Yes
AWS::RefactorSpaces::Service	Yes	Yes	Yes
AWS::Rekognition::Collection	Yes	Yes	Yes
AWS::Rekognition::Project	Yes	Yes	Yes
AWS::Rekognition::StreamProcessor	Yes	Yes	Yes
AWS::ResilienceHub::App	Yes	Yes	Yes
AWS::ResilienceHub::ResiliencyPolicy	Yes	Yes	Yes
AWS::ResilienceHubV2::Policy	Yes	Yes	No
AWS::ResilienceHubV2::Service	Yes	Yes	No
AWS::ResilienceHubV2::ServiceFunction	Yes	Yes	No
AWS::ResilienceHubV2::System	Yes	Yes	No
AWS::ResilienceHubV2::UserJourney	Yes	Yes	No
AWS::ResourceExplorer2::DefaultViewAssociation	Yes	Yes	No
AWS::ResourceExplorer2::Index	Yes	Yes	Yes
AWS::ResourceExplorer2::View	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ResourceGroups::Group	Yes	Yes	Yes
AWS::ResourceGroups::TagSyncTask	Yes	Yes	Yes
AWS::RolesAnywhere::CRL	Yes	Yes	Yes
AWS::RolesAnywhere::Profile	Yes	Yes	Yes
AWS::RolesAnywhere::TrustAnchor	Yes	Yes	Yes
AWS::Route53::CidrCollection	Yes	Yes	Yes
AWS::Route53::DNSSEC	Yes	Yes	Yes
AWS::Route53::HealthCheck	Yes	Yes	Yes
AWS::Route53::HostedZone	Yes	Yes	Yes
AWS::Route53::KeySigningKey	Yes	Yes	Yes
AWS::Route53GlobalResolver::AccessSource	Yes	Yes	No
AWS::Route53GlobalResolver::AccessToken	Yes	Yes	No
AWS::Route53GlobalResolver::DnsView	Yes	Yes	No
AWS::Route53GlobalResolver::FirewallDomainList	Yes	Yes	No
AWS::Route53GlobalResolver::FirewallRule	Yes	Yes	No
AWS::Route53GlobalResolver::GlobalResolver	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::Route53GlobalResolver::HostedZoneAssociation	Yes	Yes	No
AWS::Route53Profiles::Profile	Yes	Yes	Yes
AWS::Route53Profiles::ProfileAssociation	Yes	Yes	Yes
AWS::Route53Profiles::ProfileResourceAssociation	Yes	Yes	Yes
AWS::Route53RecoveryControl::Cluster	Yes	Yes	Yes
AWS::Route53RecoveryControl::ControlPanel	Yes	Yes	Yes
AWS::Route53RecoveryControl::RoutingControl	Yes	Yes	Yes
AWS::Route53RecoveryControl::SafetyRule	Yes	Yes	Yes
AWS::Route53RecoveryReadiness::Cell	Yes	Yes	Yes
AWS::Route53RecoveryReadiness::ReadinessCheck	Yes	Yes	Yes
AWS::Route53RecoveryReadiness::RecoveryGroup	Yes	Yes	Yes
AWS::Route53RecoveryReadiness::ResourceSet	Yes	Yes	Yes
AWS::Route53Resolver::FirewallDomainList	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Route53Resolver::FirewallRuleGroup	Yes	Yes	Yes
AWS::Route53Resolver::FirewallRuleGroupAssociation	Yes	Yes	Yes
AWS::Route53Resolver::OutpostResolver	Yes	Yes	Yes
AWS::Route53Resolver::ResolverConfig	Yes	Yes	Yes
AWS::Route53Resolver::ResolverDNSSECConfig	Yes	Yes	Yes
AWS::Route53Resolver::ResolverEndpoint	Yes	Yes	Yes
AWS::Route53Resolver::ResolverQueryLoggingConfig	Yes	Yes	Yes
AWS::Route53Resolver::ResolverQueryLoggingConfigAssociation	Yes	Yes	Yes
AWS::Route53Resolver::ResolverRule	Yes	Yes	Yes
AWS::Route53Resolver::ResolverRuleAssociation	Yes	Yes	Yes
AWS::S3::AccessGrant	Yes	Yes	Yes
AWS::S3::AccessGrantsInstance	Yes	Yes	Yes
AWS::S3::AccessGrantsLocation	Yes	Yes	Yes
AWS::S3::AccessPoint	Yes	Yes	Yes
AWS::S3::Bucket	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::S3::BucketPolicy	Yes	No	Yes
AWS::S3::MultiRegionAccessPoint	Yes	Yes	Yes
AWS::S3::MultiRegionAccessPointPolicy	Yes	No	Yes
AWS::S3::StorageLens	Yes	Yes	Yes
AWS::S3::StorageLensGroup	Yes	Yes	Yes
AWS::S3Express::AccessPoint	Yes	Yes	Yes
AWS::S3Express::BucketPolicy	Yes	Yes	Yes
AWS::S3Express::DirectoryBucket	Yes	Yes	Yes
AWS::S3Files::AccessPoint	Yes	Yes	No
AWS::S3Files::FileSystem	Yes	Yes	No
AWS::S3Files::FileSystemPolicy	Yes	Yes	No
AWS::S3Files::MountTarget	Yes	Yes	No
AWS::S3ObjectLambda::AccessPoint	Yes	Yes	Yes
AWS::S3ObjectLambda::AccessPointPolicy	Yes	Yes	No
AWS::S3Outposts::AccessPoint	Yes	Yes	Yes
AWS::S3Outposts::Bucket	Yes	Yes	Yes
AWS::S3Outposts::BucketPolicy	Yes	Yes	No
AWS::S3Outposts::Endpoint	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::S3Tables::Namespace	Yes	Yes	Yes
AWS::S3Tables::Table	Yes	Yes	Yes
AWS::S3Tables::TableBucket	Yes	Yes	Yes
AWS::S3Tables::TableBucketPolicy	Yes	Yes	Yes
AWS::S3Tables::TablePolicy	Yes	Yes	Yes
AWS::S3Vectors::Index	Yes	Yes	Yes
AWS::S3Vectors::VectorBucket	Yes	Yes	Yes
AWS::S3Vectors::VectorBucketPolicy	Yes	Yes	Yes
AWS::SES::ConfigurationSet	Yes	Yes	Yes
AWS::SES::ConfigurationSetEventDestination	No	Yes	Yes
AWS::SES::ContactList	Yes	Yes	Yes
AWS::SES::CustomVerificationEmailTemplate	Yes	Yes	No
AWS::SES::DedicatedIpPool	Yes	Yes	Yes
AWS::SES::EmailIdentity	Yes	Yes	Yes
AWS::SES::MailManagerAddonInstance	Yes	Yes	Yes
AWS::SES::MailManagerAddonSubscription	Yes	Yes	Yes
AWS::SES::MailManagerAddressList	Yes	Yes	Yes
AWS::SES::MailManagerArchive	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::SES::MailManagerIngressPoint	Yes	Yes	Yes
AWS::SES::MailManagerRelay	Yes	Yes	Yes
AWS::SES::MailManagerRuleSet	Yes	Yes	Yes
AWS::SES::MailManagerTrafficPolicy	Yes	Yes	Yes
AWS::SES::MultiRegionEndpoint	Yes	Yes	Yes
AWS::SES::Template	Yes	Yes	Yes
AWS::SES::Tenant	Yes	Yes	No
AWS::SES::VdmAttributes	Yes	Yes	No
AWS::SMSVOICE::ConfigurationSet	Yes	Yes	Yes
AWS::SMSVOICE::OptOutList	Yes	Yes	Yes
AWS::SMSVOICE::PhoneNumber	Yes	Yes	Yes
AWS::SMSVOICE::Pool	Yes	Yes	Yes
AWS::SMSVOICE::ProtectConfiguration	Yes	Yes	Yes
AWS::SMSVOICE::ResourcePolicy	Yes	Yes	Yes
AWS::SMSVOICE::SenderId	Yes	Yes	Yes
AWS::SNS::Subscription	Yes	No	Yes
AWS::SNS::Topic	Yes	Yes	Yes
AWS::SNS::TopicInlinePolicy	Yes	Yes	No
AWS::SQS::Queue	Yes	Yes	Yes
AWS::SQS::QueueInlinePolicy	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::SSM::Association	Yes	Yes	Yes
AWS::SSM::Document	Yes	No	Yes
AWS::SSM::MaintenanceWindow	Yes	Yes	No
AWS::SSM::MaintenanceWindowTarget	Yes	Yes	No
AWS::SSM::MaintenanceWindowTask	Yes	Yes	No
AWS::SSM::Parameter	Yes	Yes	Yes
AWS::SSM::PatchBaseline	Yes	Yes	Yes
AWS::SSM::ResourceDataSync	Yes	Yes	Yes
AWS::SSM::ResourcePolicy	Yes	Yes	Yes
AWS::SSMContacts::Contact	Yes	Yes	Yes
AWS::SSMContacts::ContactChannel	Yes	Yes	Yes
AWS::SSMContacts::Plan	Yes	Yes	Yes
AWS::SSMContacts::Rotation	Yes	Yes	Yes
AWS::SSMGuiConnect::Preferences	Yes	Yes	Yes
AWS::SSMIncidents::ReplicationSet	Yes	Yes	Yes
AWS::SSMIncidents::ResponsePlan	Yes	Yes	Yes
AWS::SSMQuickSetup::ConfigurationManager	Yes	Yes	Yes
AWS::SSMQuickSetup::LifecycleAutomation	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::SSO::Application	✓ Yes	✓ Yes	✓ Yes
AWS::SSO::ApplicationAssignment	✓ Yes	✓ Yes	✓ Yes
AWS::SSO::Assignment	✓ Yes	✓ Yes	✓ Yes
AWS::SSO::Instance	✓ Yes	✓ Yes	✓ Yes
AWS::SSO::InstanceAccessControlAttributeConfiguration	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::App	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::AppImageConfig	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::Cluster	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::DataQualityJobDefinition	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::Device	✓ Yes	✓ Yes	✗ No
AWS::SageMaker::DeviceFleet	✓ Yes	✓ Yes	✗ No
AWS::SageMaker::Domain	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::Endpoint	✗ No	✓ Yes	✗ No
AWS::SageMaker::FeatureGroup	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::Image	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::ImageVersion	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::InferenceComponent	✓ Yes	✓ Yes	✓ Yes
AWS::SageMaker::InferenceExperiment	✓ Yes	✓ Yes	✓ Yes

Resource	Import	Drift detection	IaC generator
AWS::SageMaker::MlflowApp	Yes	Yes	No
AWS::SageMaker::MlflowTrackingServer	Yes	Yes	Yes
AWS::SageMaker::Model	Yes	Yes	No
AWS::SageMaker::ModelBiasJobDefinition	Yes	Yes	Yes
AWS::SageMaker::ModelCard	Yes	Yes	Yes
AWS::SageMaker::ModelExplainabilityJobDefinition	Yes	Yes	Yes
AWS::SageMaker::ModelPackage	Yes	Yes	Yes
AWS::SageMaker::ModelPackageGroup	Yes	Yes	Yes
AWS::SageMaker::ModelQualityJobDefinition	Yes	Yes	Yes
AWS::SageMaker::MonitoringSchedule	Yes	No	Yes
AWS::SageMaker::PartnerApp	Yes	Yes	Yes
AWS::SageMaker::Pipeline	Yes	Yes	Yes
AWS::SageMaker::ProcessingJob	Yes	Yes	Yes
AWS::SageMaker::Project	Yes	Yes	Yes
AWS::SageMaker::Space	Yes	Yes	Yes
AWS::SageMaker::StudioLifecycleConfig	Yes	Yes	Yes
AWS::SageMaker::UserProfile	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Scheduler::Schedule	Yes	Yes	Yes
AWS::Scheduler::ScheduleGroup	Yes	Yes	Yes
AWS::SecretsManager::ResourcePolicy	Yes	No	Yes
AWS::SecretsManager::RotationSchedule	Yes	Yes	Yes
AWS::SecretsManager::Secret	Yes	Yes	Yes
AWS::SecretsManager::SecretTargetAttachment	Yes	No	Yes
AWS::SecurityAgent::AgentSpace	Yes	Yes	No
AWS::SecurityAgent::Application	Yes	Yes	No
AWS::SecurityAgent::Pentest	Yes	Yes	No
AWS::SecurityAgent::TargetDomain	Yes	Yes	No
AWS::SecurityHub::AggregatorV2	Yes	Yes	Yes
AWS::SecurityHub::AutomationRule	Yes	Yes	Yes
AWS::SecurityHub::AutomationRuleV2	Yes	Yes	Yes
AWS::SecurityHub::ConfigurationPolicy	Yes	Yes	Yes
AWS::SecurityHub::ConnectorV2	Yes	Yes	Yes
AWS::SecurityHub::DelegatedAdmin	Yes	Yes	Yes
AWS::SecurityHub::FindingAggregator	Yes	Yes	Yes
AWS::SecurityHub::Hub	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::SecurityHub::HubV2	Yes	Yes	Yes
AWS::SecurityHub::Insight	Yes	Yes	Yes
AWS::SecurityHub::OrganizationConfiguration	Yes	Yes	Yes
AWS::SecurityHub::PolicyAssociation	Yes	Yes	Yes
AWS::SecurityHub::ProductSubscription	Yes	Yes	Yes
AWS::SecurityHub::SecurityControl	Yes	Yes	Yes
AWS::SecurityHub::Standard	Yes	Yes	Yes
AWS::SecurityLake::AwsLogSource	Yes	Yes	Yes
AWS::SecurityLake::DataLake	Yes	Yes	Yes
AWS::SecurityLake::Subscriber	Yes	Yes	Yes
AWS::SecurityLake::SubscriberNotification	Yes	Yes	Yes
AWS::ServiceCatalog::CloudFormationProvisionedProduct	Yes	No	No
AWS::ServiceCatalog::LaunchNotificationConstraint	Yes	Yes	Yes
AWS::ServiceCatalog::LaunchRoleConstraint	Yes	Yes	No
AWS::ServiceCatalog::LaunchTemplateConstraint	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::ServiceCatalog::Portfolio	Yes	Yes	No
AWS::ServiceCatalog::PortfolioPrincipalAssociation	Yes	Yes	Yes
AWS::ServiceCatalog::PortfolioProductAssociation	Yes	Yes	Yes
AWS::ServiceCatalog::PortfolioShare	Yes	No	Yes
AWS::ServiceCatalog::ResourceUpdateConstraint	Yes	Yes	Yes
AWS::ServiceCatalog::ServiceAction	Yes	Yes	Yes
AWS::ServiceCatalog::ServiceActionAssociation	Yes	Yes	Yes
AWS::ServiceCatalog::StackSetConstraint	Yes	Yes	No
AWS::ServiceCatalog::TagOption	Yes	Yes	Yes
AWS::ServiceCatalog::TagOptionAssociation	Yes	Yes	Yes
AWS::ServiceCatalogAppRegistry::Application	Yes	Yes	Yes
AWS::ServiceCatalogAppRegistry::AttributeGroup	Yes	Yes	Yes
AWS::ServiceCatalogAppRegistry::AttributeGroupAssociation	Yes	No	Yes
AWS::ServiceCatalogAppRegistry::ResourceAssociation	Yes	No	Yes

Resource	Import	Drift detection	IaC generator
AWS::Shield::DRTAccess	Yes	Yes	Yes
AWS::Shield::ProactiveEngagement	Yes	Yes	Yes
AWS::Shield::Protection	Yes	Yes	Yes
AWS::Shield::ProtectionGroup	Yes	Yes	Yes
AWS::Signer::ProfilePermission	Yes	Yes	Yes
AWS::Signer::SigningProfile	Yes	Yes	Yes
AWS::SimSpaceWeaver::Simulation	Yes	Yes	Yes
AWS::StepFunctions::Activity	Yes	Yes	Yes
AWS::StepFunctions::StateMachine	Yes	Yes	Yes
AWS::StepFunctions::StateMachineAlias	Yes	Yes	Yes
AWS::StepFunctions::StateMachineVersion	Yes	Yes	Yes
AWS::SupportApp::AccountAlias	Yes	Yes	Yes
AWS::SupportApp::SlackChannelConfiguration	Yes	Yes	Yes
AWS::SupportApp::SlackWorkspaceConfiguration	Yes	Yes	Yes
AWS::Synthetics::Canary	Yes	Yes	Yes
AWS::Synthetics::Group	Yes	Yes	Yes
AWS::SystemsManagerSAP::Application	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::Timestream::Database	Yes	Yes	Yes
AWS::Timestream::InfluxDBCluster	Yes	Yes	No
AWS::Timestream::InfluxDBInstance	Yes	Yes	Yes
AWS::Timestream::ScheduledQuery	Yes	Yes	Yes
AWS::Timestream::Table	Yes	Yes	Yes
AWS::Transfer::Agreement	Yes	Yes	Yes
AWS::Transfer::Certificate	Yes	Yes	Yes
AWS::Transfer::Connector	Yes	Yes	Yes
AWS::Transfer::Profile	Yes	Yes	Yes
AWS::Transfer::Server	Yes	Yes	Yes
AWS::Transfer::User	Yes	Yes	Yes
AWS::Transfer::WebApp	Yes	Yes	Yes
AWS::Transfer::Workflow	Yes	Yes	Yes
AWS::UXC::AccountCustomization	Yes	Yes	No
AWS::VerifiedPermissions::IdentitySource	Yes	Yes	Yes
AWS::VerifiedPermissions::Policy	Yes	Yes	Yes
AWS::VerifiedPermissions::PolicyStore	Yes	Yes	Yes
AWS::VerifiedPermissions::PolicyStoreAlias	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::VerifiedPermissions::PolicyTemplate	Yes	Yes	Yes
AWS::VoiceID::Domain	Yes	Yes	Yes
AWS::VpcLattice::AccessLogSubscription	Yes	Yes	Yes
AWS::VpcLattice::AuthPolicy	Yes	Yes	No
AWS::VpcLattice::DomainVerification	Yes	Yes	Yes
AWS::VpcLattice::Listener	Yes	Yes	Yes
AWS::VpcLattice::ResourceConfiguration	Yes	Yes	Yes
AWS::VpcLattice::ResourceGateway	Yes	Yes	Yes
AWS::VpcLattice::ResourcePolicy	Yes	Yes	No
AWS::VpcLattice::Rule	Yes	Yes	Yes
AWS::VpcLattice::Service	Yes	Yes	Yes
AWS::VpcLattice::ServiceNetwork	Yes	Yes	Yes
AWS::VpcLattice::ServiceNetworkResourceAssociation	Yes	Yes	Yes
AWS::VpcLattice::ServiceNetworkServiceAssociation	Yes	Yes	Yes
AWS::VpcLattice::ServiceNetworkVpcAssociation	Yes	Yes	Yes
AWS::VpcLattice::TargetGroup	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::WAFv2::IPSet	Yes	Yes	Yes
AWS::WAFv2::LoggingConfiguration	Yes	Yes	Yes
AWS::WAFv2::RegexPatternSet	Yes	Yes	Yes
AWS::WAFv2::RuleGroup	Yes	Yes	Yes
AWS::WAFv2::WebACL	Yes	Yes	Yes
AWS::WAFv2::WebACLAssociation	Yes	Yes	No
AWS::Wisdom::AIAgent	Yes	Yes	Yes
AWS::Wisdom::AIAgentVersion	Yes	Yes	Yes
AWS::Wisdom::AIGuardrail	Yes	Yes	Yes
AWS::Wisdom::AIGuardrailVersion	Yes	Yes	Yes
AWS::Wisdom::AIPrompt	Yes	Yes	Yes
AWS::Wisdom::AIPromptVersion	Yes	Yes	Yes
AWS::Wisdom::Assistant	Yes	Yes	Yes
AWS::Wisdom::AssistantAssociation	Yes	Yes	Yes
AWS::Wisdom::KnowledgeBase	Yes	Yes	Yes
AWS::Wisdom::MessageTemplate	Yes	Yes	Yes
AWS::Wisdom::MessageTemplateVersion	Yes	Yes	Yes
AWS::Wisdom::QuickResponse	Yes	Yes	Yes
AWS::WorkSpaces::ConnectionAlias	Yes	Yes	No
AWS::WorkSpaces::Workspace	Yes	Yes	No

Resource	Import	Drift detection	IaC generator
AWS::WorkSpaces::WorkspacesPool	Yes	Yes	Yes
AWS::WorkSpacesThinClient::Environment	Yes	Yes	Yes
AWS::WorkSpacesWeb::BrowserSettings	Yes	Yes	Yes
AWS::WorkSpacesWeb::DataProtectionSettings	Yes	Yes	Yes
AWS::WorkSpacesWeb::IdentityProvider	Yes	Yes	Yes
AWS::WorkSpacesWeb::IpAccessSettings	Yes	Yes	Yes
AWS::WorkSpacesWeb::NetworkSettings	Yes	Yes	Yes
AWS::WorkSpacesWeb::Portal	Yes	Yes	Yes
AWS::WorkSpacesWeb::SessionLogger	Yes	Yes	Yes
AWS::WorkSpacesWeb::TrustStore	Yes	Yes	Yes
AWS::WorkSpacesWeb::UserAccessLoggingSettings	Yes	Yes	Yes
AWS::WorkSpacesWeb::UserSettings	Yes	Yes	Yes
AWS::WorkspacesInstances::Volume	Yes	Yes	Yes
AWS::WorkspacesInstances::VolumeAssociation	Yes	Yes	Yes
AWS::WorkspacesInstances::WorkspaceInstance	Yes	Yes	Yes
AWS::XRay::Group	Yes	Yes	Yes
AWS::XRay::ResourcePolicy	Yes	Yes	Yes

Resource	Import	Drift detection	IaC generator
AWS::XRay::SamplingRule	Yes	Yes	Yes
AWS::XRay::TransactionSearchConfig	Yes	Yes	Yes

Use quick-create links to create CloudFormation stacks

Quick-create links provide a streamlined method to launch CloudFormation stacks directly from URLs in the CloudFormation console. By specifying the template URL, stack name, and template parameters as URL query parameters, you can prepopulate a single **Create stack** page and expedite stack creation. This simplifies the process of creating stacks by reducing both the number of wizard pages and the amount of required user input. It also optimizes template reuse, as you can create multiple URLs that specify different values for the same template.

URL format

The quick-create link follows this URL format:

```
https://region-code.console.aws.amazon.com/cloudformation/home?region=region-code#/
stacks/create/review
   ?templateURL=TemplateURL
   &stackName=StackName
   &param_parameterName=parameterValue
```

CloudFormation supports the following URL query parameters:

Template URL

Required. The `templateURL` parameter specifies the URL of the stack template located in an Amazon S3 bucket. To avoid access issues with a presigned S3 URL, make sure that you URL-encode the URL.

Supported S3 URL formats:

- `https://s3.region-code.amazonaws.com/bucket-name/template-name`
- `https://bucket-name.s3.region-code.amazonaws.com/template-name`

- `https://s3-region-code.amazonaws.com/bucket-name/template-name` (legacy format)

Stack name

Optional. Use the `stackName` parameter to specify the name of the CloudFormation stack to be created. A stack name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 128 characters.

Template parameters

Optional. For parameters in the stack template that aren't a `NoEcho` parameter type, use the format `param_parameterName` in the URL query string. The URL parameter must include the `param_` prefix, and the parameter name segment must exactly match the parameter name in the template. For example: `param_DBName`.

CloudFormation ignores parameters that don't exist in the template, and any parameters defined with their `NoEcho` property set to `true` types (typically, user names and passwords). URL parameters override default values that are specified in the template. You can include as many parameters as needed.

> **Important**
>
> Rather than embedding sensitive information directly in your CloudFormation templates, we recommend you use dynamic parameters in the stack template to reference sensitive information that is stored and managed outside of CloudFormation, such as in the AWS Systems Manager Parameter Store or AWS Secrets Manager. For more information, see the Do not embed credentials in your templates best practice.

All query parameter names are case sensitive. Users can overwrite these values in the console before creating the stack.

Example

The following example is based on the WordPress basic single instance sample template. The query string includes the required `templateURL` parameter and the `stackName`, `DBName`, `InstanceType`, and `KeyName` parameters.

The following URL has line breaks added for clarity.

```
https://us-east-2.console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/
create/review
   ?templateURL=https://s3.us-east-2.amazonaws.com/cloudformation-templates-us-east-2/
WordPress_Single_Instance.template
   &stackName=MyWPBlog
   &param_DBName=mywpblog
   &param_InstanceType=t2.medium
```

The following URL includes the same parameters as the previous example, but the line breaks are removed. This is the actual URL format.

```
https://us-east-2.console.aws.amazon.com/cloudformation/home?
region=us-east-2#/stacks/create/review?templateURL=https://
s3.us-east-2.amazonaws.com/cloudformation-templates-us-east-2/
WordPress_Single_Instance.template&stackName=MyWPBlog&param_DBName=mywpblog&param_InstanceType=
```

Creating a stack using a quick-create link

When you open a quick-create link, you are directed to the CloudFormation console. The console opens directly to the **Quick create stack** page, with the supplied values automatically used for the parameters.

To create a stack using a quick-create link (console)

1. On the **Quick create stack** page, for **Template**, **Template URL**, confirm the template URL is correct.
2. Expand the **View template** section to verify the template.
3. For **Stack name**, verify the prepopulated stack name.
4. Review the **Parameters** section. Verify that the prepopulated parameter values are correct. Fill in any required parameters that weren't specified in the URL. Modify any prepopulated values if needed.
5. Next, you can configure the following settings:

   - **Tags** — Organize resources with key-value pairs.
   - **Permissions** — Choose the IAM service role for stack operations.
   - **Stack failure options** — Choose to roll back (default) or preserve resources.
   - **Stack policy** — Control resource update permissions.

- **Rollback configuration** — Configure CloudWatch alarm-based rollback.
- **Notification options** — Set up Amazon SNS notifications for stack events.
- **Stack creation options** — Define maximum stack creation time and enable termination protection to prevent accidental deletions.

For more information, see Configure stack options.

6. For **Capabilities**, complete any required acknowledgements. For example, if your template contains IAM resources, select **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates.
7. (Optional) You can create a change set to preview the configuration of the stack before creating it. Choose **Create change set** and follow the directions. For more information, see Preview the configuration of your stack.
8. When you're ready, choose **Create stack** to launch the stack and then monitor the stack creation progress in the **Events** tab. For more information, see Monitor stack progress.

Deploy AWS CloudFormation stacks faster with express mode

With CloudFormation express mode, you can complete stack operations as soon as CloudFormation applies the resource configuration, without waiting for resources to fully stabilize. Use express mode to create, update, and delete stacks significantly faster during iterative development workflows.

Express mode works with all existing CloudFormation templates. You don't need to make any template changes to use express mode.

Topics

- When to use express mode
- How express mode works
- How CloudFormation handles dependencies in express mode
- Getting started with express mode
- Rollback behavior with express mode
- Express mode and nested stacks
- Resource readiness considerations

- Unsupported features and limitations
- Comparison with AWS CDK hotswap deployments
- Related resources

When to use express mode

Use express mode for development and iteration workflows where speed is more important than waiting for full resource stabilization. Use express mode when:

- You iterate on template changes during development.
- You want faster feedback on whether your resource configurations are valid.
- You don't need to wait for resources to be fully operational before proceeding.

For production deployments where you need to verify that resources are fully stabilized and operational before considering the deployment complete, use the default deployment mode.

How express mode works

In the default deployment mode, CloudFormation waits for each resource to reach a fully stabilized state before reporting the resource as complete. For example, when creating an Amazon EC2 instance, CloudFormation waits until the instance is in the `running` state.

In express mode, CloudFormation typically considers a resource operation complete as soon as the API call to create, update, or delete the resource succeeds. In most cases, CloudFormation does not wait for the resource to reach its final operational state.

This means that after an express mode stack operation completes:

- Resources might still be initializing or propagating changes.
- Resources might not yet be ready to serve traffic or accept connections.
- Delete operations might still be in progress.

How CloudFormation handles dependencies in express mode

Express mode respects resource dependencies defined in your template. If a resource references another resource's ID or attribute (via `Ref` or `Fn::GetAtt`), CloudFormation confirms the referenced resource's configuration is applied before starting the dependent resource. If a

dependent resource encounters a transient failure because a referenced resource isn't ready, CloudFormation retries the operation.

Resources that have no dependencies on each other can proceed in parallel.

Express mode does not change the dependency ordering of your resources. It changes when the stack operation reports complete.

Getting started with express mode

You can use express mode with the AWS Command Line Interface (AWS CLI), the AWS Cloud Development Kit (AWS CDK) (AWS CDK), the CloudFormation console, and change sets. The following sections describe how to enable express mode with each method.

Using express mode with the AWS CLI

To use express mode with the AWS CLI, add the `--deployment-config` parameter to your stack operation commands.

Create a stack with express mode

The following command creates a stack using express mode:

```
aws cloudformation create-stack --stack-name my-stack \
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS"}'
```

Update a stack with express mode

The following command updates a stack using express mode:

```
aws cloudformation update-stack --stack-name my-stack \
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS"}'
```

Delete a stack with express mode

The following command deletes a stack using express mode:

```
aws cloudformation delete-stack --stack-name my-stack \
    --deployment-config '{"mode": "EXPRESS"}'
```

Using express mode with the AWS CDK

To use express mode with the AWS Cloud Development Kit (AWS CDK) (AWS CDK), use the `--express` flag with the `cdk deploy` command:

```
cdk deploy --express
```

Using express mode with AWS SAM

To use express mode with the AWS SAM CLI, add the `--express` flag to your deploy or sync commands:

```
sam deploy --express
```

```
sam sync --express
```

To persist the setting in your project configuration, use `--save-params`:

```
sam deploy --express --save-params
```

This saves `express = true` to your `samconfig.toml` file so subsequent deployments use Express mode automatically.

When express mode is enabled, the AWS SAM CLI passes the deployment configuration to CloudFormation. The `--disable-rollback` flag controls rollback behavior within the deployment configuration.

Using express mode with the console

When creating or updating a stack in the CloudFormation console, you can enable express mode in the **Configure stack options** step. Under **Deployment options**, select **Express** for the deployment mode.

Using express mode with change sets

You can use express mode with change sets by specifying the deployment configuration when creating the change set:

```
aws cloudformation create-change-set --stack-name my-stack \
```

```
    --change-set-name my-change-set \
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS"}'
```

When you execute the change set, CloudFormation applies the express mode deployment configuration specified during change set creation.

Rollback behavior with express mode

When you use express mode, CloudFormation disables rollback by default. If a resource operation fails, CloudFormation does not automatically roll back the changes.

To re-enable rollback when using express mode, set `disableRollback` to `false` in the `--deployment-config` parameter:

```
aws cloudformation create-stack --stack-name my-stack \
    --template-body file://my-template.yaml \
    --deployment-config '{"mode": "EXPRESS", "disableRollback": false}'
```

Express mode and nested stacks

When you use express mode on a parent stack, the setting propagates to all nested stacks in the hierarchy. You don't need to specify express mode for each nested stack separately. All resources in the root stack and its nested stacks complete as soon as their configuration is applied.

Resource readiness considerations

Express mode completes resource operations before resources reach a fully stabilized state, so any resource might require additional time to become fully ready to serve traffic or complete multi-region propagation. The following table illustrates the kinds of resource behavior you can expect, using examples across different resource types. These examples are illustrative and don't represent a complete list of supported resources.

Resource type	Operation	Readiness behavior
Amazon CloudFront distribution	Create/Update	Distribution might take several minutes to deploy globally after configuration is applied.

Resource type	Operation	Readiness behavior
Amazon EC2 instance	Create	Instance might still be initializing (running user data scripts, status checks) after the stack operation completes.
AWS Lambda function	Delete	Function resources might still be cleaning up after the delete operation reports complete.
Amazon Elastic Container Service (Amazon ECS) service	Create/Update	Service tasks might still be starting or reaching a steady state after the stack operation completes.

Unsupported features and limitations

The following features and services are not supported with express mode:

CloudFormation StackSets

CloudFormation does not support express mode with StackSets operations.

The following resources have specific behavior with express mode:

Custom resources

Custom resources (including `AWS::CloudFormation::CustomResource` and `Custom::*` resources) follow their default behavior. CloudFormation waits for the custom resource to send a response signal before considering the operation complete, even in express mode. To prevent a slow or stuck custom resource from delaying fast iteration workflows, set a maximum wait time with the `ServiceTimeout` property. For more information, see Response timeout.

Comparison with AWS CDK hotswap deployments

The following table compares express mode with AWS CDK hotswap deployments.

Feature	Express mode	AWS CDK hotswap
Deployment mechanism	Uses CloudFormation for all resource operations	Bypasses CloudFormation and calls service APIs directly
Template compatibility	Works with all existing CloudFormation templates	Supports a limited set of resource types
Stack state	Stack state remains consistent with template	Stack state might drift from template
Rollback support	Supports rollback (disabled by default)	No rollback support
Resource coverage	All CloudFormation resource types	Limited resource types (AWS Lambda, Amazon ECS, AWS Step Functions, and others)
Recommended use	Development and iteration workflows	Development workflows with supported resources only

Related resources

- Understand stack deployment time
- Choose how to handle failures when provisioning resources
- Nested stacks
- Update stacks using change sets
- CreateStack in the *AWS CloudFormation API Reference*

Understand stack deployment time

When you create, update, or delete a stack, CloudFormation provisions each resource in your template and reports that the stack operation is complete when your resources are ready to serve traffic. This topic describes how CloudFormation provisions a stack, what affects how long a stack operation takes, and how you can shorten deployment time.

Topics

- How CloudFormation provisions a stack
- What affects deployment time
- Ways to shorten deployment time
- Related resources

How CloudFormation provisions a stack

CloudFormation reads the dependencies between resources in your template. A resource that references another resource, through `Ref`, `Fn::GetAtt`, or `DependsOn`, is provisioned after the resource it references. Resources that have no dependencies on each other are provisioned in parallel.

For each resource, CloudFormation applies the resource configuration and then confirms that the resource is ready to serve traffic. CloudFormation moves forward as soon as each resource is ready, so dependent resources start as early as possible. As a result, stacks with longer dependency chains see the largest improvement. When every resource in the stack is ready, CloudFormation reports that the stack operation is complete.

This behavior applies automatically to stack operations. You don't need to change your templates or set any parameters.

What affects deployment time

Dependency chains

The longest chain of resources that depend on each other sets the minimum time for a stack operation. Resources that are not part of that chain are provisioned in parallel and usually don't add to the total time.

Resources that take longer to become ready

Workloads that include resources such as Amazon RDS database instances, Elastic Load Balancing load balancers, NAT gateways, or Amazon CloudFront distributions spend most of their deployment time waiting for those resources to become ready to serve traffic.

Outputs that reference resource attributes

If a stack output references a resource attribute, for example with `Fn::GetAtt`, CloudFormation waits until the attribute value is available before it completes the stack operation. This wait can add time to the operation.

Ways to shorten deployment time

- **Use express mode for development iteration.** Express mode completes stack operations as soon as CloudFormation applies the resource configuration, and resources continue becoming ready in the background. For more information, see Express mode.
- **Group resources by how often they change.** Place resources that take longer to become ready, such as databases, in separate stacks from resources that you update often. Updates to the frequently changed stack then don't include the longer-running resources.
- **Remove dependencies that you don't need.** Declare `DependsOn` only when a resource must wait for another resource. Fewer dependencies let CloudFormation provision more resources in parallel.
- **Reference resource attributes in outputs only when you need them.** Outputs that reference resource attributes can extend a stack operation until those values are available.

Related resources

- How CloudFormation works
- Express mode
- Update CloudFormation stacks using change sets

Examples of CloudFormation stack operation commands for the AWS CLI and PowerShell

The following command line examples demonstrate how to perform individual CloudFormation actions with the AWS CLI and PowerShell. These examples include only the most commonly used actions. For a complete list, see cloudformation in the *AWS CLI Command Reference*.

The examples in this guide use the convention of a backslash (\) to indicate that a long command line continues on the next line.

Topics

- Cancel a stack update
- Continue rolling back an update
- Create a stack
- Create a stack that includes transforms
- Delete a stack
- Describe stack events
- Describe a stack resource
- Describe stack resources
- Describe stacks
- Get a template
- List stack resources
- List stacks
- Update a stack
- Validate your template
- Upload local artifacts to an S3 bucket with the AWS CLI

Cancel a stack update

Use the cancel-update-stack command command to cancel a stack update. For more information, see Cancel a stack update.

CLI

AWS CLI

To cancel a stack update that is in progress

The following `cancel-update-stack` example cancels a stack update on the `myteststack` stack.

```
aws cloudformation cancel-update-stack \
    --stack-name myteststack
```

This command produces no output.

For more information, see Cancel a stack update in the *AWS CloudFormation User Guide*.

- For API details, see CancelUpdateStack in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Cancels an update on the specified stack.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- For API details, see CancelUpdateStack in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Cancels an update on the specified stack.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- For API details, see CancelUpdateStack in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Continue rolling back an update

Use the continue-update-rollback command to continue rolling back an update. For more information, see Continue rolling back an update.

CLI

AWS CLI

To retry an update rollback

The following `continue-update-rollback` example resumes a rollback operation from a previously failed stack update.

```
aws cloudformation continue-update-rollback \
    --stack-name my-stack
```

This command produces no output.

For more information, see Continue rolling back an update in the *AWS CloudFormation User Guide*.

- For API details, see ContinueUpdateRollback in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Continues rollback of the named stack, which should be in the state 'UPDATE_ROLLBACK_FAILED'. If the continued rollback is successful, the stack will enter state 'UPDATE_ROLLBACK_COMPLETE'.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- For API details, see ContinueUpdateRollback in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Continues rollback of the named stack, which should be in the state 'UPDATE_ROLLBACK_FAILED'. If the continued rollback is successful, the stack will enter state 'UPDATE_ROLLBACK_COMPLETE'.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- For API details, see ContinueUpdateRollback in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Create a stack

Use the create-stack command to create a stack. You must provide the stack name, the location of a valid template, and any input parameters. The parameter key names are case sensitive. If you mistype a parameter key name, CloudFormation doesn't create the stack and reports that the template doesn't contain that parameter.

The following examples show how to create a new stack with the specified name, template, and input parameters.

CLI

AWS CLI

To create an AWS CloudFormation stack

The following `create-stack` example creates a stack with the name `myteststack` using the `sampletemplate.json` template.

```
aws cloudformation create-stack \
    --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
 ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2
```

Contents of `sampletemplate.json`:

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Sample template that creates an Amazon EC2 instance in the
 first of the specified subnets.",
    "Parameters": {
        "KeyPairName": {
            "Type": "AWS::EC2::KeyPair::KeyName",
            "Description": "Name of an existing EC2 key pair for SSH access to
 the instance."
        },
        "SubnetIDs": {
            "Type": "CommaDelimitedList",
            "Description": "Comma-delimited list of subnet IDs."
        }
    },
    "Resources": {
        "MyInstance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0abcdef1234567890",
                "InstanceType": "t3.micro",
                "KeyName": {
                    "Ref": "KeyPairName"
                },
                "SubnetId": {
                    "Fn::Select": [
```

```
                        "0",
                        {
                            "Ref": "SubnetIDs"
                        }
                    ]
                }
            }
        }
    }
}
```

Output:

```
{
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/myteststack/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

For more information, see Managing AWS resources as a single unit with CloudFormation stacks in the *AWS CloudFormation User Guide*.

- For API details, see CreateStack in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Creates a new stack with the specified name. The template is parsed from the supplied content with customization parameters ('PK1' and 'PK2' represent the names of parameters declared in the template content, 'PV1' and 'PV2' represent the values for those parameters. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will not be rolled back.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Example 2: Creates a new stack with the specified name. The template is parsed from the supplied content with customization parameters ('PK1' and 'PK2' represent the names of parameters declared in the template content, 'PV1' and 'PV2' represent the values for those parameters. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will be rolled back.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Example 3: Creates a new stack with the specified name. The template is obtained from the Amazon S3 URL with customization parameters ('PK1' represents the name of a parameter declared in the template content, 'PV1' represents the value for the parameter. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will be rolled back (same as specifying -DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Example 4: Creates a new stack with the specified name. The template is obtained from the Amazon S3 URL with customization parameters ('PK1' represents the name of a parameter declared in the template content, 'PV1' represents the value for the parameter. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will be rolled back (same as specifying -DisableRollback $false). The specified notification AENs will receive published stack-related events.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- For API details, see CreateStack in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Creates a new stack with the specified name. The template is parsed from the supplied content with customization parameters ('PK1' and 'PK2' represent the names of parameters declared in the template content, 'PV1' and 'PV2' represent the values for those parameters. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will not be rolled back.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Example 2: Creates a new stack with the specified name. The template is parsed from the supplied content with customization parameters ('PK1' and 'PK2' represent the names of parameters declared in the template content, 'PV1' and 'PV2' represent the values for those parameters. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will be rolled back.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
```

```
            -OnFailure "ROLLBACK"
```

Example 3: Creates a new stack with the specified name. The template is obtained from the Amazon S3 URL with customization parameters ('PK1' represents the name of a parameter declared in the template content, 'PV1' represents the value for the parameter. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will be rolled back (same as specifying -DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Example 4: Creates a new stack with the specified name. The template is obtained from the Amazon S3 URL with customization parameters ('PK1' represents the name of a parameter declared in the template content, 'PV1' represents the value for the parameter. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. If creation of the stack fails, it will be rolled back (same as specifying -DisableRollback $false). The specified notification AENs will receive published stack-related events.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- For API details, see CreateStack in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Note

You can use the AWS CLI `--template-url` option to specify a template file location in Amazon S3 or AWS Systems Manager.
For Amazon S3, the URL must begin with `https://`. S3 static website URLs are not supported.

```
--template-url https://s3.region-code.amazonaws.com/bucket-name/template-name
```

For AWS Systems Manager, use the following format:

```
--template-url "ssm-doc://arn:aws:ssm:region-code:account-id:document/document-name"
```

Express mode

To create a stack using express mode, add the `--deployment-config` parameter. Express mode completes the stack operation as soon as CloudFormation applies the resource configuration. For more information, see Express mode.

```
aws cloudformation create-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --deployment-config '{"mode": "EXPRESS"}'
```

Create a stack that includes transforms

Use the deploy command to create a stack that includes transforms. When you create a stack from a template that includes transforms, you must use a change set. The `deploy` command combines two steps (creating a change set and executing it) into a single command.

AWS CLI

The following `deploy` command creates a stack with the specified name, template, and input parameters.

```
aws cloudformation deploy --stack-name myteststack \
  --template /path_to_template/my-template.json \
  --parameter-overrides Key1=Value1 Key2=Value2
```

Delete a stack

Use the delete-stack command to delete a stack. For more information, see Delete a stack.

CLI

AWS CLI

To delete a stack

The following `delete-stack` example deletes the specified stack.

```
aws cloudformation delete-stack \
    --stack-name my-stack
```

This command produces no output.

For more information, see Managing AWS resources as a single unit with CloudFormation stacks in the *AWS CloudFormation User Guide*.

- For API details, see DeleteStack in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Deletes the specified stack.

```
Remove-CFNStack -StackName "myStack"
```

- For API details, see DeleteStack in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Deletes the specified stack.

```
Remove-CFNStack -StackName "myStack"
```

- For API details, see DeleteStack in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

If the deletion fails and returns a `DELETE_FAILED` state, you can choose to delete the stack by force using the `--deletion-mode` option. These are the following values that can be used with `deletion-mode`:

- `STANDARD`: Deletes the stack normally. This is the default deletion mode.
- `FORCE_DELETE_STACK`: Deletes the stack and skips all resources that are failing to delete.

AWS CLI

The following `delete-stack` command force deletes the *`myteststack`* stack using the `FORCE_DELETE_STACK` value with the `deletion-mode` parameter:

```
aws cloudformation delete-stack --stack-name myteststack \
    --deletion-mode FORCE_DELETE_STACK
```

This command produces no output.

After using `FORCE_DELETE_STACK`, you can use the `list-stack-resources` command to list the resources that were skipped during the stack deletion process. The retained resources will show a DELETE_SKIPPED status. For more information, see List stack resources.

Express mode

To delete a stack using express mode, add the `--deployment-config` parameter:

```
aws cloudformation delete-stack --stack-name myteststack \
    --deployment-config '{"mode": "EXPRESS"}'
```

For more information, see Express mode.

Describe stack events

Use the describe-stack-events command to describe stack events. For more information, see Monitor stack progress.

CLI

AWS CLI

To describe stack events

The following `describe-stack-events` example displays the 2 most recent events for the specified stack.

```
aws cloudformation describe-stack-events \
    --stack-name my-stack \
    --max-items 2
```

Output:

```
{
    "StackEvents": [
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "EventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.556Z",
            "ResourceStatus": "UPDATE_COMPLETE"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "EventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.127Z",
            "ResourceStatus": "UPDATE_COMPLETE_CLEANUP_IN_PROGRESS"
        }
    ],
    "NextToken":
 "eyJOZXh0VG9XMPLiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```

For more information, see Monitor stack progress in the *AWS CloudFormation User Guide*.

- For API details, see DescribeStackEvents in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns all stack related events for the specified stack.

```
Get-CFNStackEvent -StackName "myStack"
```

- For API details, see DescribeStackEvents in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns all stack related events for the specified stack.

```
Get-CFNStackEvent -StackName "myStack"
```

- For API details, see DescribeStackEvents in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Describe a stack resource

CLI

AWS CLI

To get information about a stack resource

The following `describe-stack-resource` example displays details for the resource named `MyFunction` in the specified stack.

```
aws cloudformation describe-stack-resource \
    --stack-name MyStack \
    --logical-resource-id MyFunction
```

Output:

```
{
    "StackResourceDetail": {
        "StackName": "MyStack",
        "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/MyStack/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "LogicalResourceId": "MyFunction",
        "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
        "ResourceType": "AWS::Lambda::Function",
        "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
        "ResourceStatus": "UPDATE_COMPLETE",
```

```
        "Metadata": "{}",
        "DriftInformation": {
            "StackResourceDriftStatus": "IN_SYNC"
        }
    }
}
```

For more information, see Managing AWS resources as a single unit with CloudFormation stacks in the *AWS CloudFormation User Guide*.

- For API details, see DescribeStackResource in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns the description of a resource identified in the template associated with the specified stack by the logical ID "MyDBInstance".

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- For API details, see DescribeStackResource in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns the description of a resource identified in the template associated with the specified stack by the logical ID "MyDBInstance".

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- For API details, see DescribeStackResource in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Describe stack resources

CLI

AWS CLI

To get information about a stack resource

The following `describe-stack-resources` example displays details for the resources in the specified stack.

```
aws cloudformation describe-stack-resources \
    --stack-name my-stack
```

Output:

```
{
    "StackResources": [
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "LogicalResourceId": "bucket",
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
            "Timestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
```

```
            "Timestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        }
    ]
}
```

For more information, see Managing AWS resources as a single unit with CloudFormation stacks in the *AWS CloudFormation User Guide*.

- For API details, see DescribeStackResources in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns the AWS resource descriptions for up to 100 resources associated with the specified stack. To obtain details of all resources associated with a stack use the Get-CFNStackResourceSummary, which also supports manual paging of the results.

```
Get-CFNStackResourceList -StackName "myStack"
```

Example 2: Returns the description of the Amazon EC2 instance identified in the template associated with the specified stack by the logical ID "Ec2Instance".

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Example 3: Returns the description of up to 100 resources associated with the stack containing an Amazon EC2 instance identified by instance ID "i-123456". To obtain details of all resources associated with a stack use the Get-CFNStackResourceSummary, which also supports manual paging of the results.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Example 4: Returns the description of the Amazon EC2 instance identified by the logical ID "Ec2Instance" in the template for a stack. The stack is identified using the physical resource ID of a resource it contains, in this case also an Amazon EC2 instance with

instance ID "i-123456". A different physical resource could also be used to identify the stack depending on the template content, for example an Amazon S3 bucket.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- For API details, see DescribeStackResources in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns the AWS resource descriptions for up to 100 resources associated with the specified stack. To obtain details of all resources associated with a stack use the Get-CFNStackResourceSummary, which also supports manual paging of the results.

```
Get-CFNStackResourceList -StackName "myStack"
```

Example 2: Returns the description of the Amazon EC2 instance identified in the template associated with the specified stack by the logical ID "Ec2Instance".

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Example 3: Returns the description of up to 100 resources associated with the stack containing an Amazon EC2 instance identified by instance ID "i-123456". To obtain details of all resources associated with a stack use the Get-CFNStackResourceSummary, which also supports manual paging of the results.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Example 4: Returns the description of the Amazon EC2 instance identified by the logical ID "Ec2Instance" in the template for a stack. The stack is identified using the physical resource ID of a resource it contains, in this case also an Amazon EC2 instance with instance ID "i-123456". A different physical resource could also be used to identify the stack depending on the template content, for example an Amazon S3 bucket.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- For API details, see DescribeStackResources in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Describe stacks

CLI

AWS CLI

To describe AWS CloudFormation stacks

The following `describe-stacks` example shows summary information for the `myteststack` stack.

```
aws cloudformation describe-stacks \
    --stack-name myteststack
```

Output:

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
myteststack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
            "Tags": [],
            "Outputs": [
                {
                    "Description": "Name of S3 bucket to hold website content",
                    "OutputKey": "BucketName",
                    "OutputValue": "myteststack-s3bucket-jssofi1zie2w"
                }
            ],
            "StackStatusReason": null,
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "CREATE_COMPLETE",
            "DisableRollback": false
        }
    ]
}
```

For more information, see Managing AWS resources as a single unit with CloudFormation stacks in the *AWS CloudFormation User Guide*.

- For API details, see DescribeStacks in *AWS CLI Command Reference*.

Go

SDK for Go V2

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
```

```
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

- For API details, see DescribeStacks in *AWS SDK for Go API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns a collection of Stack instances describing all of the user's stacks.

```
Get-CFNStack
```

Example 2: Returns a Stack instance describing the specified stack

```
Get-CFNStack -StackName "myStack"
```

- For API details, see DescribeStacks in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns a collection of Stack instances describing all of the user's stacks.

```
Get-CFNStack
```

Example 2: Returns a Stack instance describing the specified stack

```
Get-CFNStack -StackName "myStack"
```

- For API details, see DescribeStacks in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

By default, the `describe-stacks` command returns parameter values. To prevent sensitive parameter values such as passwords from being returned, include a `NoEcho` property set to `TRUE` in your CloudFormation templates.

Important

Using the `NoEcho` attribute does not mask any information stored in the following:

- The `Metadata` template section. CloudFormation does not transform, modify, or redact any information you include in the `Metadata` section. For more information, see Metadata.
- The `Outputs` template section. For more information, see Outputs.
- The `Metadata` attribute of a resource definition. For more information, see `Metadata` attribute.

We strongly recommend you do not use these mechanisms to include sensitive information, such as passwords or secrets.

Important

Rather than embedding sensitive information directly in your CloudFormation templates, we recommend you use dynamic parameters in the stack template to reference sensitive information that is stored and managed outside of CloudFormation, such as in the AWS Systems Manager Parameter Store or AWS Secrets Manager.
For more information, see the Do not embed credentials in your templates best practice.

Get a template

CLI

AWS CLI

To view the template body for an AWS CloudFormation stack

The following `get-template` example shows the template for the `myteststack` stack.

```
aws cloudformation get-template \
    --stack-name myteststack
```

Output:

```
{
    "TemplateBody": {
        "AWSTemplateFormatVersion": "2010-09-09",
        "Outputs": {
            "BucketName": {
                "Description": "Name of S3 bucket to hold website content",
                "Value": {
                    "Ref": "S3Bucket"
                }
            }
        },
        "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
        "Resources": {
            "S3Bucket": {
                "Type": "AWS::S3::Bucket",
                "Properties": {
                    "AccessControl": "PublicRead"
                }
            }
        }
    }
}
```

For more information, see Working with CloudFormation templates in the *AWS CloudFormation User Guide*.

- For API details, see GetTemplate in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns the template associated with the specified stack.

```
Get-CFNTemplate -StackName "myStack"
```

- For API details, see GetTemplate in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns the template associated with the specified stack.

```
Get-CFNTemplate -StackName "myStack"
```

- For API details, see GetTemplate in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

List stack resources

CLI

AWS CLI

To list resources in a stack

The following `list-stack-resources` example displays the list of resources in the specified stack.

```
aws cloudformation list-stack-resources \
    --stack-name my-stack
```

Output:

```
{
    "StackResourceSummaries": [
        {
            "LogicalResourceId": "bucket",
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
            "LastUpdatedTimestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
```

```
            "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
            "LastUpdatedTimestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        }
    ]
}
```

For more information, see Managing AWS resources as a single unit with CloudFormation stacks in the *AWS CloudFormation User Guide*.

- For API details, see ListStackResources in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns descriptions of all the resources associated with the specified stack.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- For API details, see ListStackResources in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns descriptions of all the resources associated with the specified stack.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- For API details, see ListStackResources in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

List stacks

Use the list-stacks command to list stacks. To list only stacks with the specified status codes, include the `--stack-status-filter` option. You can specify one or more stack status codes for the `--stack-status-filter` option. For more information, see Stack status codes.

CLI

AWS CLI

To list AWS CloudFormation stacks

The following `list-stacks` example shows a summary of all stacks that have a status of `CREATE_COMPLETE`.

```
aws cloudformation list-stacks \
    --stack-status-filter CREATE_COMPLETE
```

Output:

```
{
    "StackSummaries": [
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
myteststack/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "TemplateDescription": "AWS CloudFormation Sample Template S3_Bucket:
 Sample template showing how to create a publicly accessible S3 bucket.
 **WARNING** This template creates an S3 bucket. You will be billed for the AWS
 resources used if you create a stack from this template.",
            "StackStatusReason": null,
            "CreationTime": "2013-08-26T03:27:10.190Z",
            "StackName": "myteststack",
            "StackStatus": "CREATE_COMPLETE"
        }
    ]
}
```

For more information, see Managing AWS resources as a single unit with CloudFormation stacks in the *AWS CloudFormation User Guide*.

- For API details, see ListStacks in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns summary information for all stacks.

```
Get-CFNStackSummary
```

Example 2: Returns summary information for all stacks that are currently being created.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Example 3: Returns summary information for all stacks that are currently being created or updated.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- For API details, see ListStacks in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns summary information for all stacks.

```
Get-CFNStackSummary
```

Example 2: Returns summary information for all stacks that are currently being created.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Example 3: Returns summary information for all stacks that are currently being created or updated.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- For API details, see ListStacks in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Update a stack

Use the update-stack command to directly update a stack. You specify the stack, and parameter values and capabilities that you want to update, and, if you want use an updated template, the name of the template. For more information, see Update stacks directly.

CLI

AWS CLI

To update AWS CloudFormation stacks

The following `update-stack` example updates the template and input parameters for the `mystack` stack.

```
aws cloudformation update-stack \
    --stack-name mystack \
    --template-url https://s3.amazonaws.com/sample/updated.template \
    --parameters ParameterKey=KeyPairName,ParameterValue=SampleKeyPair
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,SampleSubnetID2
```

Output:

```
{
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

The following `update-stack` example updates just the `SubnetIDs` parameter value for the `mystack` stack. If you don't specify a parameter value, the default value that is specified in the template is used.

```
aws cloudformation update-stack \
    --stack-name mystack \
    --template-url https://s3.amazonaws.com/sample/updated.template \
    --parameters ParameterKey=KeyPairName,UsePreviousValue=true
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,UpdatedSampleSubnetID2
```

Output:

```
{
```

```
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

The following `update-stack` example adds two stack notification topics to the `mystack` stack.

```
aws cloudformation update-stack \
    --stack-name mystack \
    --use-previous-template \
    --notification-arns "arn:aws:sns:use-
east-1:123456789012:mytopic1" "arn:aws:sns:us-west-2:123456789012:mytopic2"
```

Output:

```
{
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/mystack/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

For more information, see AWS CloudFormation stack updates in the *AWS CloudFormation User Guide*.

- For API details, see UpdateStack in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Updates the stack 'myStack' with the specified template and customization parameters. 'PK1' represents the name of a parameter declared in the template and 'PV1' represents its value. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Example 2: Updates the stack 'myStack' with the specified template and customization parameters. 'PK1' and 'PK2' represent the names of parameters declared in the template,

'PV1' and 'PV2' represents their requested values. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Example 3: Updates the stack 'myStack' with the specified template and customization parameters. 'PK1' represents the name of a parameter declared in the template and 'PV2' represents its value. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Example 4: Updates the stack 'myStack' with the specified template, obtained from Amazon S3, and customization parameters. 'PK1' and 'PK2' represent the names of parameters declared in the template, 'PV1' and 'PV2' represents their requested values. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Example 5: Updates the stack 'myStack', which is assumed in this example to contain IAM resources, with the specified template, obtained from Amazon S3, and customization parameters. 'PK1' and 'PK2' represent the names of parameters declared in the template, 'PV1' and 'PV2' represents their requested values. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. Stacks containing IAM resources require you to specify the -Capabilities "CAPABILITY_IAM" parameter otherwise the update will fail with an 'InsufficientCapabilities' error.

```
Update-CFNStack -StackName "myStack" `
```

```
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- For API details, see UpdateStack in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Updates the stack 'myStack' with the specified template and customization parameters. 'PK1' represents the name of a parameter declared in the template and 'PV1' represents its value. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Example 2: Updates the stack 'myStack' with the specified template and customization parameters. 'PK1' and 'PK2' represent the names of parameters declared in the template, 'PV1' and 'PV2' represents their requested values. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Example 3: Updates the stack 'myStack' with the specified template and customization parameters. 'PK1' represents the name of a parameter declared in the template and 'PV2' represents its value. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Example 4: Updates the stack 'myStack' with the specified template, obtained from Amazon S3, and customization parameters. 'PK1' and 'PK2' represent the names of parameters declared in the template, 'PV1' and 'PV2' represents their requested values.

The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Example 5: Updates the stack 'myStack', which is assumed in this example to contain IAM resources, with the specified template, obtained from Amazon S3, and customization parameters. 'PK1' and 'PK2' represent the names of parameters declared in the template, 'PV1' and 'PV2' represents their requested values. The customization parameters can also be specified using 'Key' and 'Value' instead of 'ParameterKey' and 'ParameterValue'. Stacks containing IAM resources require you to specify the -Capabilities "CAPABILITY_IAM" parameter otherwise the update will fail with an 'InsufficientCapabilities' error.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- For API details, see UpdateStack in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Note

To remove all notifications, specify for `[]` for the `--notification-arns` option.

Express mode

To update a stack using express mode, add the `--deployment-config` parameter:

```
aws cloudformation update-stack --stack-name myteststack \
    --template-body file://sampletemplate.json \
    --deployment-config '{"mode": "EXPRESS"}'
```

For more information, see Express mode.

Validate your template

Use the validate-template command to check your template file for syntax errors.

During validation, CloudFormation first checks if the template is valid JSON. If it isn't, CloudFormation checks if the template is valid YAML. If both checks fail, CloudFormation returns a template validation error.

CLI

AWS CLI

To validate an AWS CloudFormation template

The following `validate-template` example validates the `sampletemplate.json` template.

```
aws cloudformation validate-template \
    --template-body file://sampletemplate.json
```

Contents of `sampletemplate.json`:

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template
 showing how to create an S3 bucket. **WARNING** This template creates an S3
 bucket. You will be billed for the AWS resources used if you create a stack from
 this template.",
    "Resources": {
        "S3Bucket": {
            "Type": "AWS::S3::Bucket",
            "DeletionPolicy": "Retain"
        }
    },
    "Outputs": {
        "BucketName": {
            "Value": {
                "Ref": "S3Bucket"
            },
            "Description": "Name of the sample Amazon S3 bucket."
```

```
        }
    }
}
```

Output:

```
{
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template
 showing how to create an S3 bucket. **WARNING** This template creates an S3
 bucket. You will be billed for the AWS resources used if you create a stack from
 this template.",
    "Parameters": [],
    "Capabilities": []
}
```

For more information, see Working with CloudFormation templates in the *AWS CloudFormation User Guide*.

- For API details, see ValidateTemplate in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Validates the specified template content. The output details the capabilities, description and parameters of the template.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Example 2: Validates the specified template accessed via an Amazon S3 URL. The output details the capabilities, description and parameters of the template.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- For API details, see ValidateTemplate in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Validates the specified template content. The output details the capabilities, description and parameters of the template.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Example 2: Validates the specified template accessed via an Amazon S3 URL. The output details the capabilities, description and parameters of the template.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- For API details, see ValidateTemplate in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

The following is an example response that produces a validation error.

```
{
    "ResponseMetadata": {
        "RequestId": "4ae33ec0-1988-11e3-818b-e15a6df955cd"
    },
    "Errors": [
        {
            "Message": "Template format error: JSON not well-formed. (line 11, column
 8)",
            "Code": "ValidationError",
            "Type": "Sender"
        }
    ],
    "Capabilities": [],
    "Parameters": []
}
A client error (ValidationError) occurred: Template format error: JSON not well-formed.
 (line 11, column 8)
```

Note

The `validate-template` command is designed to check only the syntax of your template. It does not ensure that the property values that you have specified for a resource are valid for that resource. Nor does it determine the number of resources that will exist when the stack is created.
To check the operational validity, you need to attempt to create the stack. There is no sandbox or test area for CloudFormation stacks, so you are charged for the resources you create during testing.

Upload local artifacts to an S3 bucket with the AWS CLI

You can use the AWS CLI to upload local artifacts that are referenced by a CloudFormation template to an Amazon S3 bucket. Local artifacts are files that you reference in your template. Instead of manually uploading files to an S3 bucket and then adding their locations to your template, you can specify local artifacts in your template and use the package command to upload them quickly.

A local artifact is a path to a file or folder that the **package** command uploads to Amazon S3. For example, an artifact can be a local path to your AWS Lambda function's source code or an Amazon API Gateway REST API's OpenAPI file.

When using the **package** command:

- If you specify a file, the command directly uploads it to the S3 bucket.
- If you specify a folder, the command creates a `.zip` file for the folder, and then uploads the `.zip` file.
- If you don't specify a path, the command creates a `.zip` file for the working directory and uploads it.

You can specify an absolute or relative path, where the relative path is relative to your template's location.

After uploading the artifacts, the command returns a copy of your template, replacing references to local artifacts with the S3 location where the command uploaded the artifacts. You can then use the returned template to create or update a stack.

> **Note**
>
> You can use local artifacts only for resource properties that the **package** command supports. For more information about this command and a list of the supported resource properties, see the package documentation in the AWS CLI Command Reference.

Prerequisites

Before you begin, you must have an existing Amazon S3 bucket.

Package and deploy a template with local artifacts

The following template specifies the local artifact for a Lambda function's source code. The source code is stored in the `/home/user/code/lambdafunction` folder.

Original template

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Transform": "AWS::Serverless-2016-10-31",
  "Resources": {
    "MyFunction": {
      "Type": "AWS::Serverless::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "nodejs18.x",
        "CodeUri": "/home/user/code/lambdafunction"
      }
    }
  }
}
```

The following package command creates and uploads a `.zip` file of the function's source code folder to the root of the specified bucket.

```
aws cloudformation package \
  --s3-bucket amzn-s3-demo-bucket \
  --template /path_to_template/template.json \
  --output-template-file packaged-template.json \
  --output json
```

The command generates a new template at the path specified by `--output-template-file`. It replaces the artifact reference with the Amazon S3 location, as shown below. The `.zip` file is named using the MD5 checksum of the folder contents, rather than using the folder name itself.

Resulting template

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Transform": "AWS::Serverless-2016-10-31",
  "Resources": {
```

```
    "MyFunction": {
      "Type": "AWS::Serverless::Function",
      "Properties": {
        "Handler": "index.handler",
        "Runtime": "nodejs18.x",
        "CodeUri": "s3://amzn-s3-demo-bucket/md5 checksum"
      }
    }
  }
}
```

After you package your template's artifacts, deploy the processed template using the deploy command.

```
aws cloudformation deploy \
  --template-file packaged-template.json \
  --stack-name stack-name
```

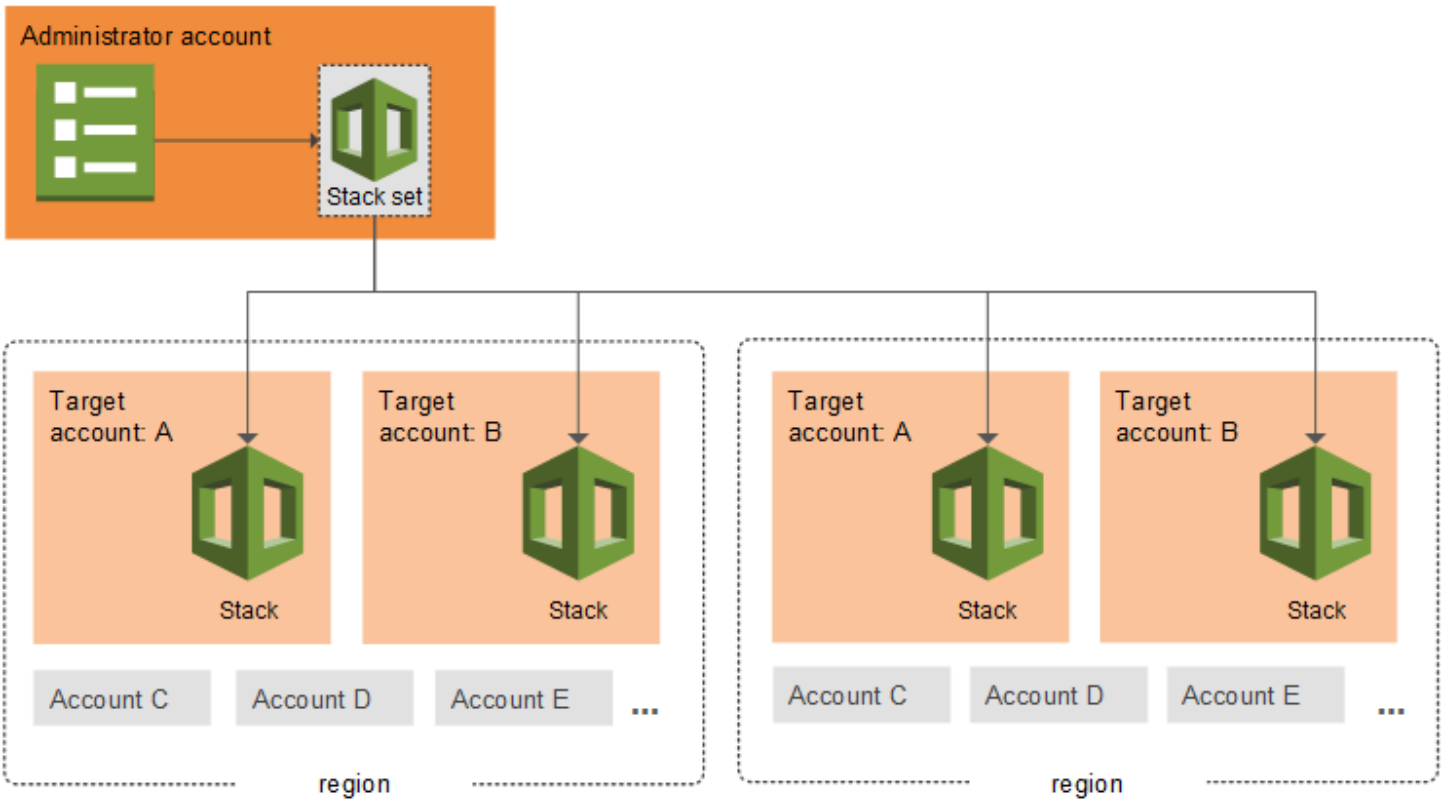
When deploying templates larger than 51,200 bytes, use the **deploy** command with the `--s3-bucket` option to upload your template to S3, as in the following example.

```
aws cloudformation deploy \
  --template-file packaged-template.json \
  --stack-name stack-name \
  --s3-bucket amzn-s3-demo-bucket
```

> **Note**
>
> For another example of using the **package** command to upload local artifacts, see Split a template into reusable pieces using nested stacks.

Managing stacks across accounts and Regions with StackSets

CloudFormation StackSets extends the capability of stacks by allowing you to create, update, or delete stacks across multiple accounts and AWS Regions with a single operation. Using an administrator account, you define and manage a CloudFormation template, and use the template as the basis for provisioning stacks into selected target accounts across specified AWS Regions.



This section helps you get started using StackSets, and answers common questions about how to work with and troubleshoot StackSet creation, updates, and deletion.

Topics

- StackSets concepts
- Prerequisites for using CloudFormation StackSets
- Get started with StackSets using a sample template
- Create CloudFormation StackSets with self-managed permissions
- Create CloudFormation StackSets with service-managed permissions

- Enable or disable automatic deployments for StackSets in AWS Organizations
- Update CloudFormation StackSets
- Add stacks to CloudFormation StackSets
- Override parameter values on stacks within your CloudFormation StackSet
- Delete stacks from CloudFormation StackSets
- Delete CloudFormation StackSets
- Prevent failed StackSets deployments using target account gates
- Choose the Concurrency Mode for CloudFormation StackSets
- Performing drift detection on CloudFormation StackSets
- Import stacks into CloudFormation StackSets
- Best practices for using CloudFormation StackSets
- CloudFormation StackSets sample templates
- Troubleshooting CloudFormation StackSets

StackSets concepts

The following terminology and concepts are central to your understanding and use of StackSets.

Topics

- Administrator and target accounts
- CloudFormation StackSets
- Permission models for StackSets
- Stack instances
- StackSet operations
- StackSet operation options
- Tags
- StackSets status codes
- Stack instance status codes

Administrator and target accounts

An *administrator account* is the AWS account in which you create StackSets. For StackSets with service-managed permissions, the administrator account is either the organization's management account or a delegated administrator account. You can manage a StackSet by signing in to the AWS administrator account that created the StackSet.

A *target account* is the account into which you create, update, or delete one or more stacks in your StackSet. Before you can use a StackSet to create stacks in a target account, set up a trust relationship between the administrator and target accounts.

CloudFormation StackSets

A *StackSet* serves as a container for multiple stacks that are deployed across specified AWS accounts and Regions. Each stack is based on the same CloudFormation template, but you can customize individual stacks using parameters.

After you've defined a StackSet, you can create, update, or delete stacks in the target accounts and AWS Regions you specify. When you create, update, or delete stacks, you can also specify operation preferences. For example, include the order of Regions you want to perform the operation, the failure tolerance threshold before stack operations stop, and the number of accounts performing stack operations concurrently.

A StackSet is a regional resource. If you create a StackSet in one AWS Region, you can only see or change it when viewing that Region.

Permission models for StackSets

You can create StackSets using either *self-managed* permissions or *service-managed* permissions.

With *self-managed* permissions, you create the IAM roles required by StackSets to deploy across accounts and Regions. These roles are necessary to establish a trusted relationship between the account you're administering the StackSet from and the account you're deploying stack instances to. Using this permissions model, StackSets can deploy to any AWS account in which you have permissions to create an IAM role.

With *service-managed* permissions, you can deploy stack instances to accounts managed by AWS Organizations. Using this permissions model, you don't have to create the necessary IAM roles;

StackSets creates the IAM roles on your behalf. With this model, you can also turn on automatic deployments to accounts that you add to your organization in the future.

AWS Organizations integrates with CloudFormation and helps you centrally manage and govern your environment as you scale and grow your AWS resources.

- Management account – the account that you use to create the organization. For more information, see Terminology and concepts for AWS Organizations.
- Delegated administrator – a compatible AWS service can register an AWS member account in the organization as an administrator for the organization's accounts in that service. For more information, see AWS services that you can use with AWS Organizations.

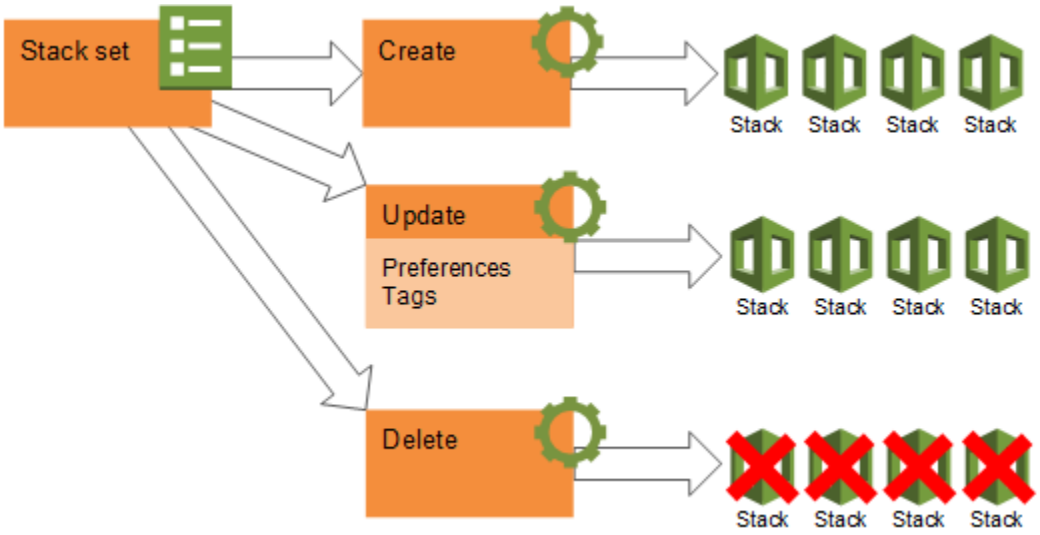
For more information about creating and managing StackSets with service-managed permissions, see the following topics:

- Activate trusted access for StackSets with AWS Organizations
- Register a delegated administrator member account
- Create CloudFormation StackSets with service-managed permissions

Stack instances

A *stack instance* is a reference to a stack in a target account within a Region. A stack instance can exist without a stack. For example, if the stack couldn't be created for some reason, the stack instance shows the reason for stack creation failure. A stack instance associates with only one StackSet.

The following figure shows the logical relationships between StackSets, stack operations, and stacks. When you update a StackSet, *all* associated stack instances update throughout all accounts and Regions.



StackSet operations

You can perform the following operations on StackSets.

Create StackSet

Creating a new StackSet includes specifying a CloudFormation template that you want to use to create stacks, specifying the target accounts in which you want to create stacks, and identifying the AWS Regions in which you want to deploy stacks in your target accounts. A StackSet ensures consistent deployment of the same stack resources, with the same settings, to all specified target accounts within the Regions you choose.

Update StackSet

When you update a StackSet, you push changes out to stacks in your StackSet. You can update a StackSet in one of the following ways. Your template updates always affect all stacks; you can't selectively update the template for some stacks in the StackSet, but not others.

- Change existing settings in the template or add new resources, such as updating parameter settings for a specific service, or adding new Amazon EC2 instances.
- Replace the template with a different template.
- Add stacks in existing or additional target accounts, across existing or additional Regions.

Delete stacks

When you delete stacks, you are removing a stack and all its associated resources from the target accounts you specify, within the Regions you specify. You can delete stacks in the following ways.

- Delete stacks from some target accounts, while leaving other stacks in other target accounts running.
- Delete stacks from some Regions, while leaving stacks in other Regions running.
- Delete stacks from your StackSet, but save them so they continue to run independently of your StackSet by choosing the **Retain Stacks** option. You can then manage retained stacks outside of your StackSet in CloudFormation.
- Delete all stacks in your StackSet, in preparation for deleting your entire StackSet.

Delete StackSet

You can delete your StackSet only when there are no stack instances in it.

StackSet operation options

The options described in this section help to control the time and number of failures allowed to perform successful StackSet operations, and prevent you from losing stack resources.

Maximum concurrent accounts

This setting, available in create, update, and delete workflows, lets you specify the maximum number or percentage of target accounts in which an operation performs at one time. A lower number or percentage means that an operation performs in fewer target accounts at one time. Operations perform in one Region at a time, in the order specified in the **Deployment order** box. For example, if you are deploying stacks to 10 target accounts within two Regions, setting **Maximum concurrent accounts** to **50** and **By percentage** deploys stacks to five accounts in the first Region, then the second five accounts within the first Region, before moving on to the next Region and beginning deployment to the first five target accounts.

When you choose **By percentage**, if the specified percentage doesn't represent a whole number of your specified accounts, CloudFormation rounds down. For example, if you are deploying stacks to 10 target accounts, and you set **Maximum concurrent accounts** to **25** and **By percentage**, CloudFormation rounds down from deploying 2.5 stacks concurrently (which would not be possible) to deploying two stacks concurrently.

Note that this setting lets you specify the *maximum* for operations. For large deployments, under certain circumstances the actual number of accounts acted upon concurrently may be lower due to service throttling.

Maximum concurrent accounts can depend on the value of **Failure tolerance** depending on your **Concurrency Mode**. If your **Concurrency Mode** is set to **Strict Failure Tolerance** then **Maximum concurrent accounts** can be at most one more than the **Failure tolerance** setting.

Concurrency mode

This setting, available in create, update, and delete workflows, lets you choose how the concurrency level behaves during StackSet operations. For more information, see Choose the Concurrency Mode for CloudFormation StackSets.

Failure tolerance

This setting, available in create, update, and delete workflows, lets you specify the maximum number or percentage of stack operation failures that can occur, per Region, beyond which CloudFormation stops an operation automatically. A lower number or percentage means that the operation performs on fewer stacks, but you are able to start troubleshooting failed operations faster. For example, if you are updating 10 stacks in 10 target accounts within three Regions, setting **Failure tolerance** to **20** and **By percentage** means that a maximum of two stack updates in a Region can fail for the operation to continue. If a third stack in the same Region fails, CloudFormation stops the operation. If a stack can't update in the first Region, the update operation continues in that Region, and then moves on to the next Region. If two stacks can't update in the second Region, the failure tolerance reaches 20%; if a third stack in the Region fails, CloudFormation stops the update operation, and doesn't go on to subsequent Regions.

When you choose **By percentage**, if the specified percentage doesn't represent a whole number of your stacks within each Region, CloudFormation rounds down. For example, if you are deploying stacks to 10 target accounts in three Regions, and you set **Failure tolerance** to **25** and **By percentage**, CloudFormation rounds down from a failure tolerance of 2.5 stacks (which would not be possible) to a failure tolerance of two stacks per Region.

Retain stacks

This setting, available in delete stack workflows, lets you keep stacks and their resources running even after removing stacks from a StackSet. When you retain stacks, CloudFormation leaves stacks in individual accounts and Regions intact. Stacks disassociate from the StackSet, but saves the stack and its resources. After a delete stacks operation is complete, you manage retained stacks in CloudFormation, in the target account (not the administrator account) that created the stacks.

Region concurrency

This setting, available in create, update, and delete workflows, lets you choose how StackSets deploy into Regions.

Sequential – Deploy StackSets operation into one Region at a time as specified by Region **Deployment order** box as long as a Region's deployment failures don't exceed a specified failure tolerance. Sequential deployment is the default selection.

Parallel – Deploy StackSets operations into all specified Regions simultaneously as long as a Region's deployment failures don't exceed a specified failure tolerance.

Tags

You can add tags during StackSet creation and update operations by specifying key and value pairs. Tags are useful for sorting and filtering StackSet resources for billing and cost allocation. For more information about how to use tags in AWS, see Organizing and tracking costs using AWS cost allocation tags in the *AWS Billing and Cost Management User Guide*. After you specify the key-value pair, choose **+** to save the tag. You can delete tags that you are no longer using by choosing the red **X** to the right of a tag.

Tags that you apply to StackSets apply to all stacks, and to the resources that your stacks create. You can also add tags at the stack-only level in CloudFormation, but those tags might not show up in StackSets.

Although StackSets doesn't add any system-defined tags, you shouldn't start the key names of any tags with the string `aws:`.

StackSets status codes

CloudFormation StackSets generates status codes for StackSet operations.

The following table describes status codes for StackSet operations.

`RUNNING`

The operation is currently in progress.

`SUCCEEDED`

The operation finished without exceeding the failure tolerance for the operation.

FAILED

The number of stacks on which the operation couldn't complete exceeded the user-defined failure tolerance. The failure tolerance value you've set for an operation applies for each Region during stack creation and update operations. If the number of failed stacks within a Region exceeds the failure tolerance, the status of the operation in the Region changes to FAILED. The status of the operation as a whole is also set to FAILED, and CloudFormation cancels the operation in any remaining Regions.

QUEUED

[Service-managed permissions] For automatic deployments that require a sequence of operations, the operation enters into a queue to perform. For example:

- Moving an account from one organizational unit (OU), OU1, to another, OU2, triggers an automatic deployment. StackSets runs a delete operation to remove the stack instance from the target OU1 account in the target Region and queues a create operation to add a stack instance to the target OU2 account in the target Region.
- Adding an account AccountA to an OU triggers an automatic deployment. StackSets runs a create operation to add a stack instance to AccountA in the target Region. If you add another account AccountB to the OU while this create operation is running, StackSets queues a second create operation. When the first create operation is complete, StackSets runs the second create operation to add a stack instance to AccountB in the target Region.

STOPPING

The operation is in the process of stopping, at the user's request.

STOPPED

The operation has stopped, at the user's request.

Stack instance status codes

CloudFormation StackSets generates status codes for stack instances.

The following table describes status codes for stack instances within StackSets.

CURRENT

The stack is up to date with the StackSet.

`OUTDATED`

The stack isn't up to date with the StackSet for one of the following reasons.

- A CreateStackSet or UpdateStackSet operation on the associated stack failed.
- The stack was part of a CreateStackSet or UpdateStackSet operation that failed, or stopped before creating or updating the stack.

`INOPERABLE`

A DeleteStackInstances operation has failed and left the stack in an unstable state. Stacks in this state are excluded from further UpdateStackSet operations. You might need to perform a DeleteStackInstances operation, with `RetainStacks` set to `true`, to delete the stack instance, and then delete the stack manually.

`CANCELLED`

The operation in the specified account and Region has been canceled. This happens because a user has stopped the StackSet operation, or because the StackSet operations exceed the failure tolerance.

`FAILED`

The operation in the specified account and Region failed. If the StackSet operation fails in enough accounts within a Region, the failure tolerance for the StackSet operation as a whole might be exceeded.

`FAILED_IMPORT`

The import of the stack instance in the specified account and Region failed and left the stack in an unstable state. Once the issues causing the failure are fixed, the import operation can be retried. If enough StackSet operations fail in enough accounts within a Region, the failure tolerance for the StackSet operation as a whole might be exceeded.

`PENDING`

The operation in the specified account and Region has yet to start.

`RUNNING`

The operation in the specified account and Region is currently in progress.

`SKIPPED_SUSPENDED_ACCOUNT`

The operation in the specified account and Region has been skipped because the account was suspended at the time of the operation.

SUCCEEDED

The operation in the specified account and Region completed successfully.

Prerequisites for using CloudFormation StackSets

StackSets extend the functionality of stacks, so you can create, update, or delete stacks across multiple accounts and Regions with a single operation.

Because StackSets perform stack operations across multiple accounts, before you can create your first StackSet you need the necessary permissions defined in your AWS accounts.

You can manage StackSets using *self-managed* or *service-managed* permissions.

- For *self-managed* StackSets, you must create and manage IAM roles in each target account and AWS Region. For more information, see Grant self-managed permissions.
- For *service-managed* StackSets, you don't need to manually create and manage IAM roles in each account; AWS handles the role creation and permissions for you. For more information, see Activate trusted access.

Topics

- Prepare to perform StackSet operations in AWS Regions that are disabled by default
- Grant self-managed permissions
- Activate trusted access for StackSets with AWS Organizations

Prepare to perform StackSet operations in AWS Regions that are disabled by default

AWS Regions introduced after March 20, 2019, such as Asia Pacific (Hong Kong), are disabled by default. You must enable these Regions for your account(s) before you can use them. For more information, see Enable or disable AWS Regions in your account in the *AWS Account Management Reference Guide*.

To create a StackSet from a StackSet's administrator account (if using self-managed permissions) or organization's management account (if using service-managed permissions) in a Region that is disabled by default, you must first enable that Region for the administrator or management account.

For CloudFormation to successfully create or update a stack instance:

- The target account must reside in a Region that's currently enabled for that target account.
- The StackSet's administrator account or organization's management account must have the same Region enabled as the target account.

> **Important**
>
> Be aware that during StackSet operations, administrator and target accounts exchange metadata regarding the accounts themselves, in addition to the StackSet and StackSet instances involved.
> In addition, if you disable a Region that contains an account in which StackSet instances reside, you are responsible for deleting any such instances or resources, if desired. In addition, be aware that metadata regarding the target account in the disabled Region will be retained in the administrator account.

Grant self-managed permissions

This topic provides instructions on how to create the IAM service roles required by StackSets to deploy across accounts and AWS Regions with *self-managed* permissions. These roles are necessary to establish a trusted relationship between the account you're administering the StackSet from and the account you're deploying stack instances to. Using this permissions model, StackSets can deploy to any AWS account in which you have permissions to create an IAM role.

To use *service-managed* permissions, see Activate trusted access instead.

Topics

- Self-managed permissions overview
- Give all users of the administrator account permissions to manage stacks in all target accounts
- Set up advanced permissions options for StackSet operations
- Set up global keys to mitigate confused deputy problems

Self-managed permissions overview

Before you create a StackSet with self-managed permissions, you must have created IAM service roles in each account.

The basic steps are:

1. Determine which AWS account is the *administrator account*.

   StackSets are created in this administrator account. A *target account* is the account in which you create individual stacks that belong to a StackSet.

2. Determine how you want to structure permissions for the StackSet.

   The simplest (and most permissive) permissions configuration is where you give *all* users and groups in the administrator account the ability to create and update *all* the StackSets managed through that account. If you need finer-grained control, you can set up permissions to specify:

   - Which users and groups can perform StackSet operations in which target accounts.
   - Which resources users and groups can include in their StackSets.
   - Which StackSet operations specific users and groups can perform.

3. Create the necessary IAM service roles in your administrator and target accounts to define the permissions you want.

   Specifically, the two required roles are:

   - **AWSCloudFormationStackSetAdministrationRole** – This role is deployed to the administrator account.
   - **AWSCloudFormationStackSetExecutionRole** – This role is deployed to all accounts where you create stack instances.

Give all users of the administrator account permissions to manage stacks in all target accounts

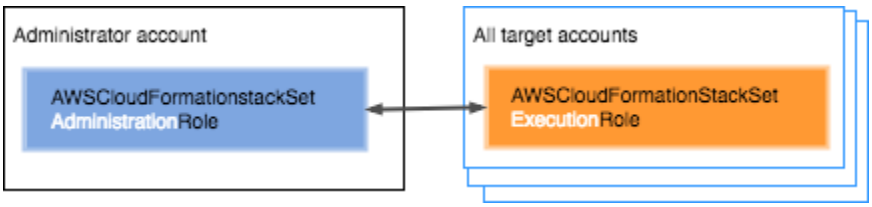
This section shows you how to set up permissions to allow all users and groups of the administrator account to perform StackSet operations in all target accounts. It guides you through creating the required IAM service roles in your administrator and target accounts. Anyone in the administrator account can then create, update, or delete any stacks across any of the target accounts.

By structuring permissions in this manner, users don't pass an administration role when creating or updating a StackSet.

Important

Even when you don't specify the `AdministrationRoleARN` parameter, the IAM principal calling `CreateStackSet` or `UpdateStackSet` must have `iam:PassRole` permission for the **AWSCloudFormationStackSetAdministrationRole** role. CloudFormation requires this permission to use the default administration role on your behalf.
The following example policy grants the required permission:

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::account-id:role/
AWSCloudFormationStackSetAdministrationRole"
}
```



Administrator account

In the administrator account, create an IAM role named **AWSCloudFormationStackSetAdministrationRole**.

You can do this by creating a stack from the CloudFormation template available from https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AWSCloudFormationStackSetAdministrationRole.yml.

Example permissions policy

The administration role created by the preceding template includes the following permissions policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWSCloudFormationStackSetExecutionRole"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Example trust policy 1

The preceding template also includes the following trust policy that grants the service permission to use the administration role and the permissions attached to the role.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Example trust policy 2

To deploy stack instances into a target account that resides in a Region that's disabled by default, you must also include the regional service principal for that Region. Each Region that's disabled by default will have its own regional service principal.

The following example trust policy grants the service permission to use the administration role in the Asia Pacific (Hong Kong) Region (`ap-east-1`), a Region that's disabled by default.

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
            "cloudformation.amazonaws.com",
            "cloudformation.ap-east-1.amazonaws.com"
         ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

For more information, see Prepare to perform StackSet operations in AWS Regions that are disabled by default. For a list of Region codes, see Regional endpoints in the *AWS General Reference Guide*.

Target accounts

In each target account, create a service role named **AWSCloudFormationStackSetExecutionRole** that trusts the administrator account. The role must have this exact name. You can do this by creating a stack from the CloudFormation template available from https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AWSCloudFormationStackSetExecutionRole.yml. When you use this template, you are prompted to provide the account ID of the administrator account with which your target account must have a trust relationship.

> **Important**
>
> Be aware that this template grants administrator access. After you use the template to create a target account execution role, you must scope the permissions in the policy statement to the types of resources that you are creating by using StackSets.

The target account service role requires permissions to perform any operations that are specified in your CloudFormation template. For example, if your template is creating an S3 bucket, then you need permissions to create new objects for S3. Your target account always

needs full CloudFormation permissions, which include permissions to create, update, delete, and describe stacks.

Example permissions policy 1

The role created by this template enables the following policy on a target account.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

Example permissions policy 2

The following example shows a policy statement with the *minimum* permissions for StackSets to work. To create stacks in target accounts that use resources from services other than CloudFormation, you must add those service actions and resources to the **AWSCloudFormationStackSetExecutionRole** policy statement for each target account.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Example trust policy

The following trust relationship is created by the template. The administrator account's ID is shown as *admin_account_id*.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

You can configure the trust relationship of an existing target account execution role to trust a specific role in the administrator account. If you delete the role in the administrator account, and create a new one to replace it, you must configure your target account trust relationships with the new administrator account role, represented by *admin_account_id* in the preceding example.

Set up advanced permissions options for StackSet operations

If you require finer-grained control over the StackSets that users and groups are creating through a single administrator account, you can use IAM roles to specify:

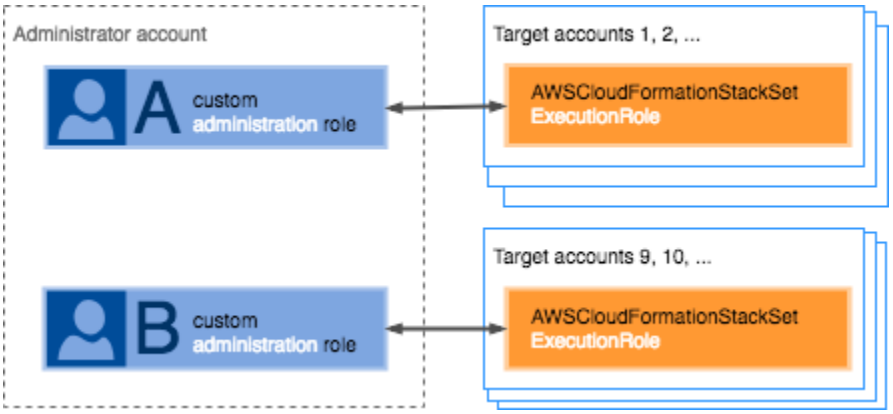
- Which users and groups can perform StackSet operations in which target accounts.
- Which resources users and groups can include in their StackSets.
- Which StackSet operations specific users and groups can perform.

Control which users can perform StackSet operations in specific target accounts

Use customized administration roles to control which users and groups can perform StackSet operations in which target accounts. You might want to control which users of the administrator account can perform StackSet operations in which target accounts. To do this, you create a

trust relationship between each target account and a specific customized administration role, rather than creating the **AWSCloudFormationStackSetAdministrationRole** service role in the administrator account itself. You then activate specific users and groups to use the customized administration role when performing StackSet operations in a specific target account.

For example, you can create Role A and Role B within your administrator account. You can give Role A permissions to access target account 1 through account 8. You can give Role B permissions to access target account 9 through account 16.



Setting up the necessary permissions involves defining a customized administration role, creating a service role for the target account, and granting users permission to pass the customized administration role when performing StackSet operations.

In general, here's how it works once you have the necessary permissions in place: When creating a StackSet, the user must specify a customized administration role. The user must have permission to pass the role to CloudFormation. In addition, the customized administration role must have a trust relationship with the target accounts specified for the StackSet. CloudFormation creates the StackSet and associates the customized administration role with it. When updating a StackSet, the user must explicitly specify a customized administration role, even if it's the same customized administration role used with this StackSet previously. CloudFormation uses that role to update the stack, subject to the requirements above.

Administrator account

Example permissions policy

For each StackSet, create a customized administration role with permissions to assume the target account execution role.

The target account execution role name must be the same in every target account. If the role name is **AWSCloudFormationStackSetExecutionRole**, StackSets uses it automatically when

creating a StackSet. If you specify a custom role name, users must provide the execution role name when creating a StackSet.

Create an IAM service role with a custom name and the following permissions policy. In the examples below, *custom_execution_role* refers to the execution role in the target accounts.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/custom_execution_role"
            ],
            "Effect": "Allow"
        }
    ]
}
```

To specify multiple accounts in a single statement, separate them with commas.

```
"Resource": [
  "arn:aws:iam::target_account_id_1:role/custom_execution_role",
  "arn:aws:iam::target_account_id_2:role/custom_execution_role"
]
```

You can specify all target accounts by using a wildcard (*) instead of an account ID.

```
"Resource": [
  "arn:aws:iam::*:role/custom_execution_role"
]
```

Example trust policy 1

You must provide a trust policy for the service role to defines which IAM principals can assume the role.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Example trust policy 2

To deploy stack instances into a target account that resides in a Region that's disabled by default, you must also include the regional service principal for that Region. Each Region that's disabled by default will have its own regional service principal.

The following example trust policy grants the service permission to use the administration role in the Asia Pacific (Hong Kong) Region (`ap-east-1`), a Region that's disabled by default.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
            "cloudformation.amazonaws.com",
            "cloudformation.ap-east-1.amazonaws.com"
         ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

For more information, see Prepare to perform StackSet operations in AWS Regions that are disabled by default. For a list of Region codes, see Regional endpoints in the *AWS General Reference Guide*.

Example pass role policy

You also need an IAM permissions policy for your IAM users that allows the user to pass the customized administration role when performing StackSet operations. For more information, see Granting a user permissions to pass a role to an AWS service.

In the example below, *customized_admin_role* refers to the administration role the user needs to pass.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::*:role/customized_admin_role"
    }
  ]
}
```

Target accounts

In each target account, create a service role that trusts the customized administration role you want to use with this account.

The target account role requires permissions to perform any operations that are specified in your CloudFormation template. For example, if your template is creating an S3 bucket, then you need permissions to create new objects in S3. Your target account always needs full CloudFormation permissions, which include permissions to create, update, delete, and describe stacks.

The target account role name must be the same in every target account. If the role name is **AWSCloudFormationStackSetExecutionRole**, StackSets uses it automatically when creating a

StackSet. If you specify a custom role name, users must provide the execution role name when creating a StackSet.

Example permissions policy

The following example shows a policy statement with the *minimum* permissions for StackSets to work. To create stacks in target accounts that use resources from services other than CloudFormation, you must add those service actions and resources to the permissions policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Example trust policy

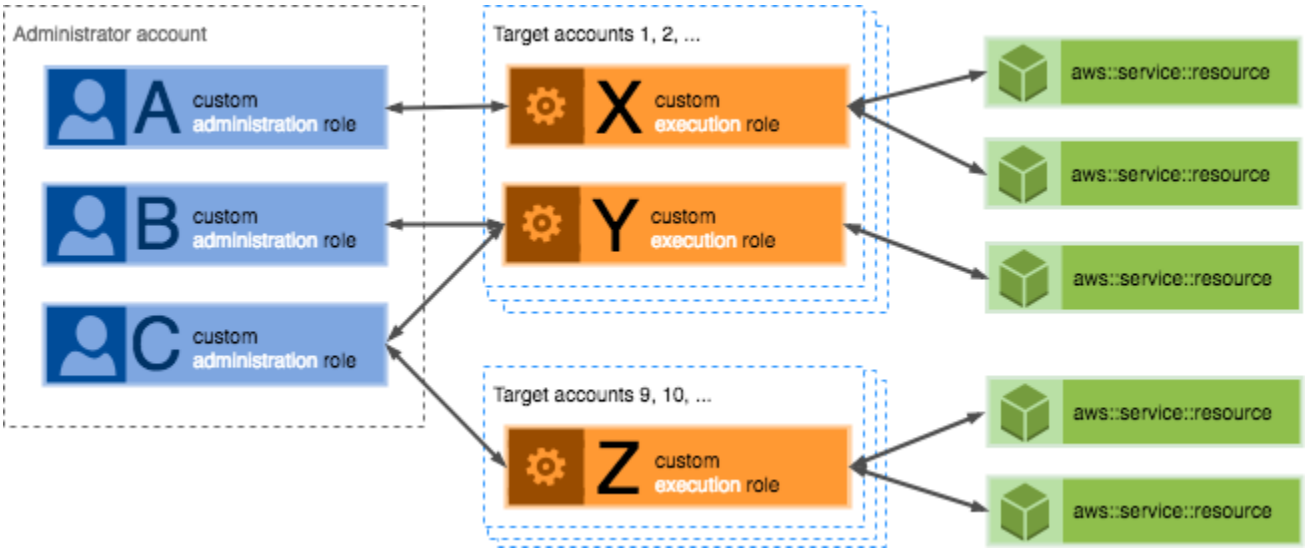
You must provide the following trust policy when you create the role to define the trust relationship.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/customized_admin_role"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Control the resources that users can include in specific StackSets

Use customized execution roles to control which stack resources users and groups can include in their StackSets. For example, you might want to set up a group that can only include Amazon S3-related resources in the StackSets they create, while another team can only include DynamoDB resources. To do this, you create a trust relationship between the customized administration role for each group and a customized execution role for each set of resources. The customized execution role defines which stack resources can be included in StackSets. The customized administration role resides in the administrator account, while the customized execution role resides in each target account in which you want to create StackSets using the defined resources. You then activate specific users and groups to use the customized administration role when performing StackSets operations.

For example you can create customized administration roles A, B, and C in the administrator account. Users and groups with permission to use Role A can create StackSets containing the stack resources specifically listed in customized execution role X, but not those in roles Y or Z, or resource not included in any execution role.



When updating a StackSet, the user must explicitly specify a customized administration role, even if it's the same customized administration role used with this StackSet previously. CloudFormation performs the update using the customized administration role specified, so long as the user has permissions to perform operations on that StackSet.

Similarly, the user can also specify a customized execution role. If they specify a customized execution role, CloudFormation uses that role to update the stack, subject to the requirements above. If the user doesn't specify a customized execution role, CloudFormation performs the

update using the customized execution role previously associated with the StackSet, so long as the user has permissions to perform operations on that StackSet.

Administrator account

Create a customized administration role in your administrator account, as detailed in Control which users can perform StackSet operations in specific target accounts. Include a trust relationship between the customized administration role and the customized execution roles that you want it to use.

Example permissions policy

The following example is a permissions policy for both the **AWSCloudFormationStackSetExecutionRole** defined for the target account, in addition to a customized execution role.

```
{
  "Version":"2012-10-17",
   "Statement": [
    {
      "Sid": "Stmt1487980684000",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWSCloudFormationStackSetExecutionRole",
        "arn:aws:iam::*:role/custom_execution_role"
      ]
    }
  ]
}
```

Target accounts

In the target accounts in which you want to create your StackSets, create a customized execution role that grants permissions to the services and resources that you want users and groups to be able to include in the StackSets.

Example permissions policy

The following example provides the minimum permissions for StackSets, along with permission to create Amazon DynamoDB tables.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamoDb:createTable"
      ],
      "Resource": "*"
    }
  ]
}
```

Example trust policy

You must provide the following trust policy when you create the role to define the trust relationship.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/customized_admin_role"
            },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

Set up permissions for specific StackSet operations

In addition, you can set up permissions for which user and groups can perform specific StackSet operations, such as creating, updating, or deleting StackSets or stack instances. For more information, see Actions, resources, and condition keys for CloudFormation in the *Service Authorization Reference*.

Set up global keys to mitigate confused deputy problems

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it shouldn't otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the aws:SourceArn and aws:SourceAccount global condition context keys in resource policies to limit the permissions that CloudFormation StackSets gives another service to the resource. If you use both global condition context keys, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:cloudformation::123456789012:*`. Whenever possible, use `aws:SourceArn`, because it's more specific. Use `aws:SourceAccount` only when you can't determine the correct ARN or ARN pattern.

When StackSets assumes the **Administration** role in your **administrator** account, StackSets populates your **administrator** account ID and StackSets Amazon Resource Name (ARN). Therefore, you can define conditions for global keys `aws:SourceAccount` and `aws:SourceArn` in the trust relationships to prevent confused deputy problems. The following example shows how you can

use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in StackSets to prevent the confused deputy problem.

Administrator account

Example Global keys for `aws:SourceAccount` and `aws:SourceArn`

When using StackSets, define the global keys `aws:SourceAccount` and `aws:SourceArn` in your **AWSCloudFormationStackSetAdministrationRole** trust policy to prevent confused deputy problems.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:cloudformation:*:111122223333:stackset/*"
        }
      }
    }
  ]
}
```

Example StackSets ARNs

Specify your associated StackSets ARNs for finer control.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": [
                        "arn:aws:cloudformation:STACKSETS-
REGION:111122223333:stackset/STACK-SET-ID-1",
                        "arn:aws:cloudformation:STACKSETS-
REGION:111122223333:stackset/STACK-SET-ID-2"
                    ]
                }
            }
        }
    ]
}
```

Activate trusted access for StackSets with AWS Organizations

This topic provides instructions on how to activate trusted access with AWS Organizations, which is required by StackSets to deploy across accounts and AWS Regions using *service-managed* permissions. To use *self-managed* permissions, see Grant self-managed permissions instead.

Before you create a StackSet with service-managed permissions, you must first complete the following tasks:

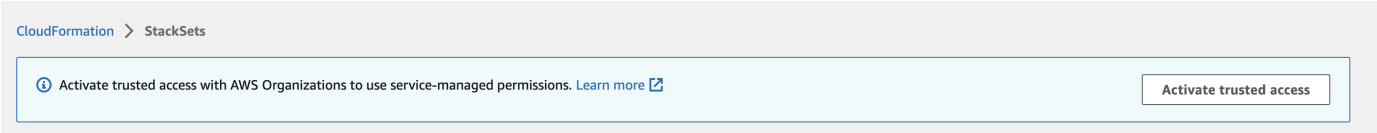
- Enable all features in AWS Organizations. With only consolidated billing features enabled, you can't create a StackSet with service-managed permissions.
- Activate trusted access with AWS Organizations. This action allows CloudFormation to create a service-linked role in the management account. After trusted access is activated, when you create a StackSet with service-managed permissions, CloudFormation creates both the necessary service-linked role and a service role named `stacksets-exec-*` in the target (member) accounts.

With trusted access activated, the management account and delegated administrator accounts can create and manage service-managed StackSets for their organization.

To activate trusted access, you must be an administrator user in the management account. An *administrator user* is a user with full permissions to your AWS account. For more information, Create an administrator user in the *AWS Account Management Reference Guide*. For recommendations for protecting the security of the management account, see Best practices for the management account in the *AWS Organizations User Guide*.

To activate trusted access

1. Sign in to AWS as an administrator of the management account and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the navigation pane, choose **StackSets**. If trusted access is deactivated, a banner displays that prompts you to activate trusted access.



3. Choose **Activate trusted access**.

   Trusted access is successfully activated when the following banner displays.



> **Note**
>
> Activate Organizations Access is the same as Enable Organizations Access, and Deactivate Organizations Access is the same as Disable Organizations Access. These terms have been updated based on marketing guidelines.

To deactivate trusted access

See CloudFormation StackSets and AWS Organizations in the *AWS Organizations User Guide*.

Before you can deactivate trusted access with AWS Organizations, you must deregister all delegated administrators. For more information, see Register a delegated administrator.

Note

For information about using API operations instead of the console to activate or deactivate trusted access, see:

- ActivateOrganizationsAccess
- DeactivateOrganizationsAccess
- DescribeOrganizationsAccess

Service-linked roles

The management account uses the **AWSServiceRoleForCloudFormationStackSetsOrgAdmin** service-linked role. You can modify or delete this role only if trusted access with AWS Organizations is deactivated.

Each target account uses a **AWSServiceRoleForCloudFormationStackSetsOrgMember** service-linked role. You can modify or delete this role only under two conditions: if trusted access with AWS Organizations is deactivated, or if the account is removed from the target organization or organizational unit (OU).

For more information, see CloudFormation StackSets and AWS Organizations in the *AWS Organizations User Guide*.

Register a delegated administrator member account

In addition to your organization's management account, member accounts with delegated administrator permissions can create and manage StackSets with service-managed permissions for the organization. StackSets with service-managed permissions are created in the management account, including StackSets created by delegated administrators. To be registered as a delegated administrator for your organization, your member account must be in the organization. For more information about joining an organization, see Inviting an AWS account to join your organization.

Your organization can have up to five registered delegated administrators at one time. Delegated administrators can choose to deploy to all accounts in your organization or specific OUs. Trusted access with AWS Organizations must be activated before delegated administrators can deploy to accounts managed by Organizations. For more information, see Activate trusted access for StackSets with AWS Organizations.

Important

Please be aware of the following:

- Delegated administrators have full permissions to deploy to accounts in your organization. The management account can't limit delegated administrator permissions to deploy to specific OUs or to perform specific StackSet operations.
- Make sure your delegated administrators have `organizations:ListDelegatedAdministrators` permissions to avoid any potential errors.

You can register delegated administrators for your organization in the following Regions: US East (Ohio), US East (N. Virginia), US West (N. California), US West (Oregon), Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), Europe (Frankfurt), Europe (Ireland), Europe (London), Europe (Paris), Europe (Stockholm), Israel (Tel Aviv), South America (São Paulo), AWS GovCloud (US-East), and AWS GovCloud (US-West).

You can register and deregister delegated administrators using the CloudFormation console, AWS CLI, or AWS SDKs.

To register a delegated administrator (console)

1. Sign in to AWS as an administrator of the management account and open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. From the navigation pane, choose **StackSets**.
3. Under **Delegated administrators**, choose **Register delegated administrator**.
4. In the **Register delegated administrator** dialog box, choose **Register delegated administrator**.

   The success message indicates that the member account has successfully been registered as a delegated administrator.

To deregister a delegated administrator (console)

1. Sign in to AWS as an administrator of the management account and open the CloudFormation console at https://console.aws.amazon.com/.
2. From the navigation pane, choose **StackSets**.

3. Under **Delegated administrators**, select the account that you want to deregister, and then choose **Deregister**.

   The success message indicates that the member account has successfully been deregistered as a delegated administrator.

   You can register this account again at any time.

To register a delegated administrator (AWS CLI)

1. Open the AWS CLI.
2. Run the `register-delegated-administrator` command.

```
$ aws organizations register-delegated-administrator \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com \
  --account-id="memberAccountId"
```

3. Run the `list-delegated-administrators` command to verify that the specified member account is successfully registered as a delegated administrator.

```
$ aws organizations list-delegated-administrators \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com
```

To deregister a delegated administrator (AWS CLI)

1. Open the AWS CLI.
2. Run the `deregister-delegated-administrator` command.

```
$ aws organizations deregister-delegated-administrator \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com \
  --account-id="memberAccountId"
```

3. Run the `list-delegated-administrators` command to verify that the specified member account is successfully deregistered as a delegated administrator.

```
$ aws organizations list-delegated-administrators \
  --service-principal=member.org.stacksets.cloudformation.amazonaws.com
```

You can register this account again at any time.

Get started with StackSets using a sample template

This tutorial will help you get started with StackSets using the AWS Management Console. It guides you through creating a StackSet using a sample template. You'll learn how to deploy stacks across multiple Regions, monitor StackSet operations, and view the results.

In this tutorial, you'll create a StackSet that enables AWS Config in your AWS account within the US West (Oregon) Region (`us-west-2`) and US East (N. Virginia) Region (`us-east-1`). With StackSets, you can create, update, or delete stacks across multiple accounts and Regions with a single operation, making it an ideal solution for managing infrastructure at scale. While this tutorial uses a single account for simplicity, it effectively demonstrates the multi-region capabilities of StackSets.

The sample template is available in the following S3 bucket: https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSConfig.yml.

> **Note**
>
> StackSets is free, but you'll be charged for the AWS resources you create with it, such as AWS Config in this tutorial. For more information, see AWS Config pricing.

Topics

- Prerequisites
- Create a StackSet with a sample template from the console
- Monitor StackSet creation
- View StackSet results
- Update your StackSet
- Add stacks to your StackSet
- Clean up
- Next steps

Prerequisites

Before you begin this tutorial, make sure you have completed the following prerequisites:

- You must have set up the required IAM roles for self-managed permissions. To create a StackSet and deploy stacks within a single account, you need the following roles in your account:
  - `AWSCloudFormationStackSetAdministrationRole`
  - `AWSCloudFormationStackSetExecutionRole`

  For detailed instructions on setting up these roles, see Grant self-managed permissions.

Create a StackSet with a sample template from the console

To create a StackSet that enables AWS Config

1. Open the CloudFormation console.
2. On the navigation bar at the top of the screen, choose the AWS Region that you want to manage the StackSet from.

   You can choose any Region that supports StackSets. The Region you select doesn't affect which Regions you can deploy to with your StackSet.
3. From the navigation pane, choose **StackSets**.
4. From the top of the **StackSets** page, choose **Create StackSet**.
5. Under **Permissions**, choose **Self-service permissions** and choose the IAM roles you created in the prerequisites.
   - For IAM admin role, choose **AWSCloudFormationStackSetAdministrationRole**.
   - For IAM execution role name, choose **AWSCloudFormationStackSetExecutionRole**.
6. Under **Prerequisite - Prepare template**, choose **Use a sample template**.
7. Under **Select a sample template**, choose the **Enable AWS Config** template. Then, choose **Next**.

   This template creates the necessary resources to enable AWS Config in your account, including a configuration recorder and delivery channel.
8. On the **Specify StackSet details** page, for **StackSet name**, enter **`my-awsconfig-stackset`**.

9. For **StackSet description**, enter **`A StackSet that enables Config across multiple Regions`**.
10. Under **Parameters**, configure the AWS Config settings as follows:

    a. For **Support all resource types**, keep the default value, **true**, to record all supported resource types.

    b. For **Include global resource types**, keep the default value, **false**, to exclude global resources like IAM roles.

    c. Leave **List of resource types if not all supported** set to **<All>**.

    d. For **The region containing the Config service-linked role resource**, replace **<DeployToAnyRegion>** with **`us-west-2`**.

    This means that the service-linked role named `AWSServiceRoleForConfig` will only be created if a stack is deployed to the US West (Oregon) Region. You'll choose the deployment Regions later in this procedure.

    e. For **Configuration recorder recording frequency**, choose **DAILY** recording.
11. Choose **Next** to continue.
12. On the **Configure StackSet options** page, choose **Add new tag** and add a tag by specifying a key and value pair:

    a. For **Key**, enter **`Stage`**.

    b. For **Value**, enter **`Test`**.

    Tags that you apply to StackSets are applied to resources that are created by your stacks.
13. For **Execution configuration**, choose **Active** to enable CloudFormation's optimized operation handling:

    - Non-conflicting operations run concurrently for faster deployment times.
    - Conflicting operations are automatically queued and processed in the order they were requested.

    While operations are running or queued, CloudFormation queues all incoming operations even if they're non-conflicting. You can't change execution settings during this time.
14. Choose **Next**.
15. On the **Set deployment options** page, for **Add stacks to StackSet**, choose **Deploy new stacks**.

16. For **Accounts**, choose **Deploy stacks in accounts**.
17. In the text box, enter your AWS account ID.
18. For **Specify regions**, select the following Regions in this order:

    1. US West (Oregon) Region (`us-west-2`)
    2. US East (N. Virginia) Region (`us-east-1`)

    Use the up arrow next to US West (Oregon) Region to move it to be the first entry in the list if needed. The order of the Regions determines their deployment order.
19. For **Deployment options**, configure the following settings:

    a. For **Maximum concurrent accounts**, keep the defaults of **Number** and **1**.

    For multi-account deployments, this setting means that CloudFormation deploys your stack in only one account at a time.

    b. For **Failure tolerance**, keep the defaults of **Number** and **0**.

    This means that a maximum of zero stack deployments can fail in one of your specified Regions before CloudFormation stops deployment in the current Region and cancels deployments in remaining Regions.

    c. For **Region concurrency**, choose **Sequential** (default).

    This setting ensures that CloudFormation completes deployments in one Region before moving to the next.

    d. For **Concurrency mode**, keep the default of **Strict failure tolerance**.

    For multi-account deployments, this reduces the account concurrency level when failures occur, staying within **Failure tolerance** +1.
20. Choose **Next**.
21. On the **Review** page, review your choices. To make changes, choose **Edit** on the related section.
22. When you are ready to create your StackSet, choose **Submit**.

Monitor StackSet creation

After you choose **Submit**, CloudFormation begins creating your StackSet and deploying stacks to the specified Regions in your account. The StackSet details page opens automatically, where you can monitor the progress of the operation.

To monitor the StackSet creation

1. On the StackSet details page, the **Operations** tab is displayed by default, showing the current operation in progress.
2. The operation status should be `RUNNING` initially. CloudFormation creates stacks in the Regions you specified according to the deployment options you configured.
3. To see more details about the operation, select the operation ID in the list.
4. On the operation details page, you can view the status of stack instances being created in each Region.
5. Wait for the operation status to change to `SUCCEEDED`, which indicates that the StackSet and all its stack instances were created successfully.

View StackSet results

After the StackSet creation is complete, you can view the deployed stack instances and verify that AWS Config has been enabled in your account across the specified Regions.

To view the StackSet results

1. On the StackSet details page, choose the **Stack instances** tab.
2. You should see a list of stack instances that were created in your account across the specified Regions. Each stack instance should have a status of `SUCCEEDED`, indicating that it was successfully deployed.
3. To verify that AWS Config is enabled in your account, you can check the AWS Config console in each of the deployed Regions.

Update your StackSet

After creating your StackSet, you might want to update it to modify parameter values or add more Regions. This section shows you how to update the AWS Config recording frequency parameter.

To update your StackSet

1. On the **StackSets** page, select your **`my-awsconfig-stackset`**.
2. With the StackSet selected, choose **Edit StackSet details** from the **Actions** menu.
3. On the **Choose a template** page, for **Prerequisite - Prepare template**, choose **Use current template**.
4. Choose **Next**.
5. On the **Specify StackSet details** page, under **Parameters**, find **Configuration recorder recording frequency** and change it from **DAILY** to **CONTINUOUS**.
6. Choose **Next**.
7. On the **Configure StackSet options** page, leave the settings as they are and choose **Next**.
8. On the **Set deployment options** page, specify your account ID and the same Regions that you used when creating the StackSet.
9. For **Deployment options**, keep the same settings as before.
10. Choose **Next**.
11. On the **Review** page, review your changes and choose **Submit**.
12. CloudFormation starts updating your StackSet. You can monitor the progress on the **Operations** tab of the StackSet details page.

Add stacks to your StackSet

You can add more stacks to your StackSet by deploying to additional Regions. This section shows you how to add stacks to a new Region.

To add stacks to your StackSet

1. On the **StackSets** page, select your **`my-awsconfig-stackset`**.
2. With the StackSet selected, choose **Add stacks to StackSet** from the **Actions** menu.
3. On the **Set deployment options** page, for **Add stacks to StackSet**, choose **Deploy new stacks**.
4. For **Accounts**, choose **Deploy stacks in accounts** and enter your account ID.
5. For **Specify regions**, select a new Region, such as **Europe (Ireland)** (`eu-west-1`).
6. For **Deployment options**, keep the same settings as before.
7. Choose **Next**.
8. On the **Specify Overrides** page, leave the property values as specified and choose **Next**.

9. On the **Review** page, review your choices and choose **Submit**.
10. CloudFormation starts creating new stacks in the specified Region. You can monitor the progress on the **Operations** tab of the StackSet details page.

Clean up

To avoid incurring charges for unwanted AWS Config resources, you should clean up by deleting the stacks from your StackSet, deleting the StackSet itself, and removing the IAM roles you created for this tutorial. Since all resources are deployed within your account, cleanup is straightforward.

To delete stacks from your StackSet

1. On the **StackSets** page, select your **`my-awsconfig-stackset`**.
2. With the StackSet selected, choose **Delete stacks from StackSet** from the **Actions** menu.
3. On the **Set deployment options** page, for **Accounts**, choose **Deploy stacks in accounts** and enter your account ID.
4. For **Specify regions**, select all the Regions where you deployed stacks.
5. For **Deployment options**, keep the default settings.
6. Make sure **Retain stacks** is *not* turned on, so that the stacks and their resources are deleted.
7. Choose **Next**.
8. On the **Review** page, review your choices and choose **Submit**.
9. CloudFormation starts deleting the stacks from your StackSet. You can monitor the progress on the **Operations** tab of the StackSet details page.

To delete your StackSet

1. After all stacks have been deleted, on the **StackSets** page, select your **`my-awsconfig-stackset`**.
2. With the StackSet selected, choose **Delete StackSet** from the **Actions** menu.
3. When prompted to confirm, choose **Delete**.

To delete the IAM service roles

Since you deployed only to your account, you only need to delete the IAM roles from this single account, making cleanup much simpler than multi-account deployments.

1. Open the IAM console.
2. From the navigation pane, choose **Roles**.
3. In the search box, enter **`AWSCloudFormationStackSet`** to find the roles you created for this tutorial.
4. Select the checkbox next to **AWSCloudFormationStackSetAdministrationRole**.
5. Choose **Delete** from the top of the page.
6. In the confirmation dialog box, enter **`delete`** and choose **Delete**.
7. Repeat the same process to delete the **AWSCloudFormationStackSetExecutionRole**.

After deleting the StackSet, an Amazon S3 bucket will remain in each AWS Region due to the `DeletionPolicy` attribute on the `AWS::S3::Bucket` resource. This preserves your AWS Config history data. If you no longer need this data, you can safely delete the bucket manually. Before you can delete a bucket, you must first empty it. Emptying a bucket deletes all objects in it.

To empty and delete the Amazon S3 buckets

1. Open the Amazon S3 console.
2. In the navigation pane on the left side of the console, choose **Buckets**.
3. In the **Buckets** list, you'll see buckets created for this StackSet in each Region where you deployed. Select the option next to the name of the bucket created for this StackSet, and then choose **Empty**.
4. On the **Empty bucket** page, confirm that you want to empty the bucket by typing **`permanently delete`** into the text field, and then choose **Empty**.
5. Monitor the progress of the bucket emptying process on the **Empty bucket: Status** page.
6. To return to your bucket list, choose **Exit**.
7. Select the option next to the name of the bucket, and then choose **Delete**.
8. When prompted for confirmation, type the name of the bucket and then choose **Delete bucket**.
9. Monitor the progress of the bucket deletion process from the **Buckets** list. When Amazon S3 completes the deletion of the bucket, it removes the bucket from the list.
10. Repeat this process for each bucket created by the StackSet in the different Regions.

Next steps

Congratulations! You've successfully created a StackSet using a sample template, deployed stacks to multiple Regions within your account, updated the StackSet, added more stacks, and cleaned up your resources. By focusing on single-account deployment, you've simplified the cleanup process while still learning the core multi-region capabilities of StackSets.

To learn more about StackSets, explore the following topics:

- Override parameter values on stacks within your CloudFormation StackSet – Learn how to override parameter values for specific accounts and Regions.
- Create CloudFormation StackSets with service-managed permissions – Explore creating StackSets for multi-account deployments with AWS Organizations.

Create CloudFormation StackSets with self-managed permissions

This topic describes how to create StackSets with *self-managed* permissions to deploy stacks across AWS accounts and Regions.

> **Note**
>
> Before you continue, create the IAM service roles required by StackSets to establish a trusted relationship between the account you're administering the StackSet from and the account you're deploying stacks to. For more information, see Grant self-managed permissions.

Topics

- Create a StackSet with self-managed permissions (console)
- Create a StackSet with self-managed permissions (AWS CLI)

Create a StackSet with self-managed permissions (console)

To create a StackSet

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region that you want to manage the StackSet from.
3. From the navigation pane, choose **StackSets**.
4. From the top of the **StackSets** page, choose **Create StackSet**.
5. Under **Permissions**, choose **Self-service permissions** and choose the IAM roles you created.
6. Under **Prerequisite - Prepare template**, choose **Template is ready**.
7. Under **Specify template**, choose to either specify the URL for the S3 bucket that contains your stack template or upload a stack template file. Then, choose **Next**.
8. On the **Specify StackSet details** page, provide a name for the StackSet, specify any parameters, and then choose **Next**.
9. Choose **Next** to continue.
10. On the **Configure StackSet options** page, under **Tags**, specify any tags to apply to resources in your stack. For more information about how tags are used in AWS, see Organizing and tracking costs using AWS cost allocation tags in the *AWS Billing and Cost Management User Guide*.
11. For **Execution configuration**, choose **Active** to enable CloudFormation's optimized operation handling:

    - Non-conflicting operations run concurrently for faster deployment times.
    - Conflicting operations are automatically queued and processed in the order they were requested.

    While operations are running or queued, CloudFormation queues all incoming operations even if they're non-conflicting. You can't change execution settings during this time.
12. If your template contains IAM resources, for **Capabilities**, choose **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates.
13. Choose **Next**.

14. On the **Set deployment options** page, for **Add stacks to StackSet**, choose **Deploy new stacks**.
15. For **Accounts**, choose **Deploy stacks in accounts**. Paste your target AWS account numbers in the text box, separating multiple numbers with commas.

    > **Note**
    >
    > You can include your administrator account ID if you want to deploy stacks in that account as well.

16. Under **Specify regions**, choose the Regions you want to deploy stacks in.
17. For **Deployment options**, do the following:

    - For **Maximum concurrent accounts**, specify how many accounts are processed concurrently.
    - For **Failure tolerance**, specify the maximum number of account failures allowed per Region. The operation will stop and won't proceed to other Regions once this limit is reached.
    - For **Region concurrency**, choose how to process Regions: **Sequential** (one Region at a time) or **Parallel** (multiple Regions concurrently).
    - For **Concurrency mode**, choose how concurrency behaves during operation execution.
      - **Strict failure tolerance** – Reduces account concurrency level when failures occur, staying within **Failure tolerance** +1.
      - **Soft failure tolerance** – Maintains your specified concurrency level (the value of **Maximum concurrent accounts**) regardless of failures.
18. Choose **Next**.
19. On the **Review** page, review your choices. To make changes, choose **Edit** on the related section.
20. When you are ready to create your StackSet, choose **Submit**.

    CloudFormation starts creating your StackSet. View the progress and status of the creation of the stacks in your StackSet in the StackSet details page that opens when you choose **Submit**.

Create a StackSet with self-managed permissions (AWS CLI)

Follow the steps in this section to use the AWS CLI to:

- Create the StackSet container.
- Deploy stack instances.

To create a StackSet

1. Use the create-stack-set command to create a new StackSet named *my-stackset*. The following example uses a template stored in an S3 bucket and includes a parameter that sets a *KeyPairName* with the value *TestKey*.

```
aws cloudformation create-stack-set \
  --stack-set-name my-stackset \
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/
MyApp.template \
  --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
```

2. After your **create-stack-set** command is finished, run the list-stack-sets command to see that your StackSet has been created. You should see your new StackSet in the results.

```
aws cloudformation list-stack-sets
```

3. Use the create-stack-instances command to deploy stacks within your StackSet. The following example deploys stacks in two AWS accounts (*account_ID_1* and *account_ID_2*) across two Regions (*us-west-2* and *us-east-1*).

   Set concurrent account processing and other deployment preferences using the `--operation-preferences` option. This example uses count-based settings. Note that `MaxConcurrentCount` must not exceed `FailureToleranceCount` + 1. For percentage-based settings, use `FailureTolerancePercentage` or `MaxConcurrentPercentage` instead.

```
aws cloudformation create-stack-instances \
  --stack-set-name my-stackset \
  --accounts account_ID_1 account_ID_2 \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

   For more information, see CreateStackInstances in the *AWS CloudFormation API Reference*.

4. Use the describe-stack-set-operation command to verify that your stacks were created successfully. For the `--operation-id` option, specify the operation ID that was returned as part of the **create-stack-instances** output.

```
aws cloudformation describe-stack-set-operation \
```

```
  --stack-set-name my-stackset \
  --operation-id operation_ID
```

Create CloudFormation StackSets with service-managed permissions

With *service-managed* permissions, you can deploy stacks to accounts managed by AWS Organizations in specific Regions. With this model, you don't need to create IAM roles in each target account and AWS Region. CloudFormation creates the IAM roles on your behalf. For more information, see Activate trusted access.

Topics

- Considerations
- Create a StackSet with service-managed permissions (console)
- Create a StackSet with service-managed permissions (AWS CLI)

Considerations

Before you create a StackSet with service-managed permissions, consider the following:

- StackSets with service-managed permissions can be initiated by either your organization's management account or delegated administrator accounts, but all operations are performed by the management account.
- CloudFormation doesn't deploy stacks to the management account, even if that account is part of your organization or belongs to an organizational unit (OU).
- Your StackSet can target your entire organization (includes all accounts) or specified OUs. When a StackSet targets a parent OU, it automatically includes any child OUs. By default, when a StackSet targets specific OUs, it includes all accounts within those OUs. However, you can target specific accounts using account filter options.
- Multiple StackSets can target the same organization or OU.
- Your can't target accounts outside your organization.
- Your authorization to deploy StackSets depends on the permissions assigned to the IAM principal (user, role, or group) you use to sign in to the management account. For an example IAM policy

that grants permissions to deploy to an organization, see Restrict stack set operations based on Region and resource types.

- Delegated administrators have full permissions to deploy to any account in your organization. The management account can't limit delegated administrator permissions to deploy to specific OUs or StackSet operations.
- Automatic deployment settings apply at the StackSet level. You can't adjust automatic deployments selectively for OUs, accounts, or Regions.
- StackSets that use service-managed permissions don't support nested stacks or templates that contain macros or transforms.

Create a StackSet with service-managed permissions (console)

To create a StackSet

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region that you want to manage the StackSet from.
3. From the navigation pane, choose **StackSets**.
4. From the top of the **StackSets** page, choose **Create StackSet**.
5. Under **Permissions**, choose **Service-managed permissions**.

   > **Note**
   >
   > If trusted access with AWS Organizations is disabled, a banner displays. Trusted access is required to create or update a StackSet with service-managed permissions. Only the administrator in the organization's management account has permissions to Activate trusted access for StackSets with AWS Organizations.

6. Under **Prerequisite - Prepare template**, choose **Template is ready**.
7. Under **Specify template**, choose to either specify the URL for the S3 bucket that contains your stack template or upload a stack template file. Then, choose **Next**.
8. On the **Specify StackSet details** page, provide a name for the StackSet, specify any parameters, and then choose **Next**.

9. On the **Configure StackSet options** page, under **Tags**, specify any tags to apply to resources in your stack. For more information about how tags are used in AWS, see Organizing and tracking costs using AWS cost allocation tags in the *AWS Billing and Cost Management User Guide*.
10. For **Execution configuration**, choose **Active** to enable CloudFormation's optimized operation handling:

    - Non-conflicting operations run concurrently for faster deployment times.
    - Conflicting operations are automatically queued and processed in the order they were requested.

    While operations are running or queued, CloudFormation queues all incoming operations even if they're non-conflicting. You can't change execution settings during this time.
11. If your template contains IAM resources, for **Capabilities**, choose **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates.
12. Choose **Next** to proceed and to activate trusted access if not already activated.
13. On the **Set deployment options** page, under **Deployment targets**, do one of the following:

    - To deploy to all accounts in your organization, choose **Deploy to organization**.
    - To deploy to all accounts in specific OUs, choose **Deploy to organizational units (OUs)**. Choose **Add an OU**, and then paste the target OU ID in the text box. Repeat for each new target OU.

    If you chose **Deploy to organizational units (OUs)**, for **Account filter type**, you can set your deployment targets to be specific individual accounts by choosing one of the following options and providing account numbers.

    - **None** (default) – Deploy stacks to all accounts in the specified OUs.
    - **Intersection** – Deploy stacks to specific individual accounts within the selected OUs.
    - **Difference** – Deploy stacks to all accounts in the selected OUs except for specific accounts.
    - **Union** – Deploy stacks to the specified OUs plus additional individual accounts.
14. Under **Automatic deployment**, choose whether to automatically deploy to accounts that are added to the target organization or OUs in the future. For more information, see Enable or disable automatic deployments for StackSets in AWS Organizations.

15. If you enabled automatic deployment, under **Account removal behavior**, choose whether stack resources are retained or deleted when an account is removed from a target organization or OU.

    > **Note**
    >
    > With **Retain stacks** selected, stacks are removed from your StackSet, but the stacks and their associated resources are retained. The resources stay in their current state, but will no longer be part of the StackSet.

16. Under **Specify regions**, choose the Regions you want to deploy stacks in.
17. For **Deployment options**, do the following:

    - For **Maximum concurrent accounts**, specify how many accounts are processed concurrently.
    - For **Failure tolerance**, specify the maximum number of account failures allowed per Region. The operation will stop and won't proceed to other Regions once this limit is reached.
    - For **Region concurrency**, choose how to process Regions: **Sequential** (one Region at a time) or **Parallel** (multiple Regions concurrently).
    - For **Concurrency mode**, choose how concurrency behaves during operation execution.
      - **Strict failure tolerance** – Reduces account concurrency level when failures occur, staying within **Failure tolerance** +1.
      - **Soft failure tolerance** – Maintains your specified concurrency level (the value of **Maximum concurrent accounts**) regardless of failures.
    - For StackSet **dependencies**, add dependent StackSet ARNs, staying within 10 dependencies maximum. For more information, see Enable or disable automatic deployments for StackSets in AWS Organizations.

18. Choose **Next** to continue.
19. On the **Review** page, verify that your StackSet will deploy to the correct accounts in the correct Regions, and then choose **Create StackSet**.

    The **StackSet details** page opens. You can view the progress and status of the creation of the stacks in your StackSet.

Create a StackSet with service-managed permissions (AWS CLI)

Follow the steps in this section to use the AWS CLI to:

- Create the StackSet container.
- Deploy stack instances.

> **Note**
>
> When acting as a delegated administrator, you must include `--call-as DELEGATED_ADMIN` in the command.

Deploy to your organization

To create a StackSet

1. Use the create-stack-set command to create a new StackSet named *my-stackset*. The following example uses a template stored in an S3 bucket, enables automatic deployments, and preserves stacks when accounts are removed. For more information, see Enable or disable automatic deployments for StackSets in AWS Organizations.

```
aws cloudformation create-stack-set \
  --stack-set-name my-stackset \
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/
MyApp.template \
  --permission-model SERVICE_MANAGED \
  --auto-deployment
 Enabled=true,RetainStacksOnAccountRemoval=true,DependsOn=ARN1,ARN2
```

2. Use the list-stack-sets command to confirm that your StackSet was created. Your new StackSet is listed in the results.

```
aws cloudformation list-stack-sets
```

   - If you set the `--call-as` option to `DELEGATED_ADMIN` while signed in to your member account, **list-stack-sets** returns all StackSets with service-managed permissions in the organization's management account.
   - If you set the `--call-as` option to `SELF` while signed in to your AWS account, **list-stack-sets** returns all self-managed StackSets in your AWS account.

- If you set the `--call-as` option to `SELF` while signed in to the organization's management account, **list-stack-sets** returns all StackSets in the organization's management account.

3. Use the create-stack-instances command to add stacks to your StackSet. For the `--deployment-targets` option, specify the organization root ID to deploy to all accounts in your organization.

   Set concurrent account processing and other deployment preferences using the `--operation-preferences` option. This example uses count-based settings. Note that `MaxConcurrentCount` must not exceed `FailureToleranceCount` + 1. For percentage-based settings, use `FailureTolerancePercentage` or `MaxConcurrentPercentage` instead.

   ```
   aws cloudformation create-stack-instances --stack-set-name my-stackset \
     --deployment-targets OrganizationalUnitIds=r-a1b2c3d4e5 \
     --regions us-west-2 us-east-1 \
     --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
   ```

   For more information, see CreateStackInstances in the *AWS CloudFormation API Reference*.

4. Using the `operation-id` that was returned as part of the **create-stack-instances** output, use the following describe-stack-set-operation command to verify that your stacks were created successfully.

   ```
   aws cloudformation describe-stack-set-operation \
     --stack-set-name my-stackset \
     --operation-id operation_ID
   ```

Deploy to organizational units (OUs)

To create a StackSet

1. Use the create-stack-set command to create a new StackSet named *`my-stackset`*. The following example uses a template stored in an S3 bucket and includes a parameter that sets a *`KeyPairName`* with the value *`TestKey`*

   ```
   aws cloudformation create-stack-set \
     --stack-set-name my-stackset \
   ```

```
  --template-url https://s3.region-code.amazonaws.com/amzn-s3-demo-bucket/
MyApp.template \
  --permission-model SERVICE_MANAGED \
  --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
```

2. Use the list-stack-sets command to confirm that your StackSet was created. Your new StackSet is listed in the results.

```
aws cloudformation list-stack-sets
```

   - If you set the `--call-as` option to `DELEGATED_ADMIN` while signed in to your member account, **list-stack-sets** returns all StackSets with service-managed permissions in the organization's management account.
   - If you set the `--call-as` option to `SELF` while signed in to your AWS account, **list-stack-sets** returns all self-managed StackSets in your AWS account.
   - If you set the `--call-as` option to `SELF` while signed in to the organization's management account, **list-stack-sets** returns all StackSets in the organization's management account.

3. Use the create-stack-instances command to add stacks to your StackSet. For the `--deployment-targets` option, specify the OU IDs to deploy to.

   Set concurrent account processing and other deployment preferences using the `--operation-preferences` option. This example uses count-based settings. Note that `MaxConcurrentCount` must not exceed `FailureToleranceCount` + 1. For percentage-based settings, use `FailureTolerancePercentage` or `MaxConcurrentPercentage` instead.

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo,ou-rcuk-slr5lh0a \
  --regions us-west-2 us-east-1 \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

   For more information, see CreateStackInstances in the *AWS CloudFormation API Reference*.

4. Using the `operation-id` that was returned as part of the **create-stack-instances** output, use the following describe-stack-set-operation command to verify that your stacks were created successfully.

```
aws cloudformation describe-stack-set-operation \
```

```
  --stack-set-name my-stackset \
  --operation-id operation_ID
```

Deploy to specific accounts in OUs

You can target specific organizational units (OUs) and use account filtering to precisely control which accounts receive stack deployments. By default, stacks deploy to all accounts within specified OUs if no account filtering is specified.

In the AWS CLI, you specify account filtering with the `--deployment-targets` option. For more information, see DeploymentTargets.

After you create the StackSet container with the **create-stack-set** command, use one of the following examples to deploy stacks to specific accounts.

Target specific accounts in an OU

The following example deploys stacks only to accounts A1 and A2 in OU1.

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets
 OrganizationalUnitIds=OU1,Accounts=A1,A2,AccountFilterType=INTERSECTION \
  --regions us-west-2 us-east-1
```

Exclude accounts from an OU

The following example deploys stacks to all accounts in OU1 except accounts A1 and A2.

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets
 OrganizationalUnitIds=OU1,Accounts=A1,A2,AccountFilterType=DIFFERENCE \
  --regions us-west-2 us-east-1
```

Enable or disable automatic deployments for StackSets in AWS Organizations

CloudFormation can automatically deploy additional stacks to new AWS Organizations accounts when they're added to your target organization or organizational units (OUs). You can enable

automatic deployments and choose whether to delete or retain stacks and their associated resources when accounts are removed from target OUs. These settings can be modified anytime for StackSets that use service-managed permissions.

How automatic deployments work

When automatic deployments are enabled, they're triggered when accounts are added to a target organization or OU, removed from a target organization or OU, or moved between target OUs.

For example, consider `StackSet1` that targets 0U1 in the `us-east-1` Region and `StackSet2` that targets 0U2 in the `us-east-1` Region. 0U1 contains `AccountA`.

If you move `AccountA` from 0U1 to 0U2 with automatic deployments enabled, CloudFormation automatically runs a delete operation to remove the `StackSet1` stack from `AccountA` and queues a create operation that adds the `StackSet2` stack to `AccountA`.

Considerations

The following are considerations when using automatic deployments:

- The automatic deployments feature is enabled at the StackSet level. You can't adjust automatic deployments selectively for OUs, accounts, or Regions.
- Overridden parameter values only apply to the accounts that are currently in the target OUs and their child OUs. Accounts added to the target OUs and their child OUs in the future will use the StackSet default values and not the overridden values.
- Automatic deployments do not consider account level targeting filters. If you target specific accounts and enable automatic deployments, your StackSet will continue to deploy to any newly added accounts within the deployed organization. To prevent deployments to newly added accounts, disable automatic deployments.
- Dependency management: Define up to 10 dependencies per StackSet, with up to 100 dependencies per account. For example, if you have five StackSets with five dependencies each, you have 25 dependencies counting towards the 100 dependency limit. You can request a limit increase through service quota console. Dependencies are removed when StackSets are deleted or Organizations are deactivated.
- It is recommended to enable managed execution when using auto-deployments. Managed execution will allow auto-deployment operations in multiple target accounts to execute concurrently within a StackSet, increasing processing velocity, especially for larger Organizations.

Enable or disable automatic deployments (console)

To enable or disable automatic deployments

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the StackSet in.
3. From the navigation pane, choose **StackSets**.
4. On the **StackSets** page, select the option next to the name of the StackSet to update.
5. Choose **Edit automatic deployment** from the **Actions** menu in the upper right corner.
6. From the dialog box that opens, do the following:
   a. For **Automatic deployment**, choose **Activated** or **Deactivated**.
   b. For **Account removal behavior**, choose **Delete stacks** or **Retain stacks**. Retained resources stay in their current state, but will no longer be part of the StackSet.
   c. For StackSet **dependencies**, add dependent StackSet ARNs, staying within 10 dependencies maximum.
7. Choose **Save**.

Enable or disable automatic deployments (AWS CLI)

To enable or disable automatic deployments

1. Use the update-stack-set command with the `--auto-deployment` option.

   The following command enables automatic deployments.

```
aws cloudformation update-stack-set --stack-set-name my-stackset \
  --use-previous-template --auto-deployment
 Enabled=true,RetainStacksOnAccountRemoval=true,DependsOn=ARN1,ARN2
```

   Alternatively, to disable automatic deployments, specify `Enabled=false` as the value for the `--auto-deployment` option, as in the following example.

```
aws cloudformation update-stack-set --stack-set-name my-stackset \
```

```
  --use-previous-template --auto-deployment Enabled=false
```

2. Using the operation ID that was returned as part of the **update-stack-set** output, run describe-stack-set-operation to verify that your StackSet was updated successfully.

```
aws cloudformation describe-stack-set-operation --operation-id operation_ID
```

Update CloudFormation StackSets

You can update your StackSet using either the CloudFormation console or the AWS CLI.

To add and remove accounts and Regions from a StackSet, see Add stacks to StackSets and Delete stacks from StackSets. To override parameter values for a stack, see Override parameters on stacks.

Topics

- Update your StackSet (console)
- Update your StackSet (AWS CLI)

Update your StackSet (console)

To update a StackSet

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the StackSet in.
3. From the navigation pane, choose **StackSets**.
4. On the **StackSets** page, select the StackSet you want to update.
5. With the StackSet selected, choose **Edit StackSet details** from the **Actions** menu.
6. On the **Choose a template** page, update the **Permissions** section as needed, or skip to the next step.
7. For **Prerequisite - Prepare template**, choose **Use current template** to use the current template, or **Replace current template** to specify an S3 URL to another template or upload a new template.
8. Choose **Next**.

9. On the **Specify StackSet details** page, for **StackSet description**, update the description for the StackSet as needed.
10. For **Parameters**, update the parameter values as needed.
11. Choose **Next**.
12. On the **Configure StackSet options** page, for **Tags**, modify the tags as needed. You can add, update, or delete tags. For more information about how tags are used in AWS, see Organizing and tracking costs using AWS cost allocation tags in the *AWS Billing and Cost Management User Guide*.
13. For **Execution configuration**, you can update the execution configuration as needed.

    > **Note**
    >
    > Remember, you can't change execution settings when operations are running or queued.

14. If your template contains IAM resources, for **Capabilities**, choose **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates.
15. Choose **Next**.
16. On the **Set deployment options** page, provide the accounts and Regions for the update.

    CloudFormation will deploy stack updates in the specified accounts within the first Region, then moves on to the next, and so on, as long as a Region's deployment failures don't exceed a specified failure tolerance.

    a. [Self-managed permissions] For **Accounts**, **Deployment locations**, choose **Deploy stacks in accounts**. Paste the target account IDs that you used to create your StackSet in the text box, separating multiple numbers with commas.

       [Service-managed permissions] Do one of the following:

       - Choose **Deploy to organizational units (OUs)**. Enter the target OUs that you used to create your StackSet.
       - Choose **Deploy to accounts**. Paste the target OU IDs or account IDs that you used to create your StackSet.

b. For **Specify regions**, specify the order in which you want CloudFormation to deploy your updates.

c. For **Deployment options**, do the following:

- For **Maximum concurrent accounts**, specify how many accounts are processed concurrently.
- For **Failure tolerance**, specify the maximum number of account failures allowed per Region. The operation will stop and won't proceed to other Regions once this limit is reached.
- For **Region concurrency**, choose how to process Regions: **Sequential** (one Region at a time) or **Parallel** (multiple Regions concurrently).
- For **Concurrency mode**, choose how concurrency behaves during operation execution.
  - **Strict failure tolerance** – Reduces account concurrency level when failures occur, staying within **Failure tolerance** +1.
  - **Soft failure tolerance** – Maintains your specified concurrency level (the value of **Maximum concurrent accounts**) regardless of failures.
- [Service-managed permissions] For StackSet **dependencies**, add dependent StackSet ARNs, staying within 10 dependencies maximum. For more information, see Enable or disable automatic deployments for StackSets in AWS Organizations.

d. Choose **Next** to continue.

17. On the **Review** page, review your choices. To make changes, choose **Edit** on the related section.
18. When you're ready to proceed, choose **Submit**.

    CloudFormation starts applying your updates to your StackSet, and displays the **Operations** tab of the StackSet details page. You can view the progress and status of update operations on the **Operations** tab.

Update your StackSet (AWS CLI)

> **Note**
>
> When acting as a delegated administrator, you must include `--call-as DELEGATED_ADMIN` in the command.

1. **To update a StackSet**

   Use the update-stack-set command to make changes to your StackSet.

   In the following examples, we're updating the StackSet by using `--parameters` option. Specifically, we change the default snapshot delivery frequency for delivery channel configuration from `TwentyFour_Hours` to `Twelve_Hours`. Because we're still using the current template, we add the `--use-previous-template` option.

   Set concurrent account processing and other deployment preferences using the `--operation-preferences` option. These examples use count-based settings. Note that `MaxConcurrentCount` must not exceed `FailureToleranceCount` + 1. For percentage-based settings, use `FailureTolerancePercentage` or `MaxConcurrentPercentage` instead.

   [Self-managed permissions] For the `--accounts` option, provide the account IDs you want your update to target.

   ```
   aws cloudformation update-stack-set --stack-set-name my-stackset \
     --use-previous-template \
     --parameters ParameterKey=MaximumExecutionFrequency,ParameterValue=Twelve_Hours \
     --accounts account_ID_1 account_ID_2 \
     --regions us-west-2 us-east-1 \
     --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
   ```

   [Service-managed permissions] For the `--deployment-targets` option, provide the organization root ID or organizational unit (OU) IDs you want your update to target.

   ```
   aws cloudformation update-stack-set --stack-set-name my-stackset \
     --use-previous-template \
     --parameters ParameterKey=MaximumExecutionFrequency,ParameterValue=Twelve_Hours \
     --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo,ou-rcuk-slr5lh0a \
     --regions us-west-2 us-east-1 \
     --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
   ```

   For more information, see UpdateStackSet in the *AWS CloudFormation API Reference*.

2. Verify that your StackSet was updated successfully by running the **describe-stack-set-operation** command to show the status and results of your update operation. For `--operation-id`, use the operation ID that was returned by your **update-stack-set** command.

```
aws cloudformation describe-stack-set-operation \
  --operation-id operation_ID
```

Add stacks to CloudFormation StackSets

When you create a StackSet, you can create the stacks for that StackSet. CloudFormation also enables you to add more stacks, for additional accounts and Regions, at any point after the StackSet is created. You can add stacks using either the CloudFormation console or the AWS CLI.

Topics

- Add stacks to a StackSet (console)
- Add stacks to a StackSet (AWS CLI)

Add stacks to a StackSet (console)

To add stacks to a StackSet

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the StackSet in.
3. From the navigation pane, choose **StackSets**. On the StackSets page, select the StackSet that you created.
4. With the StackSet selected, choose **Add stacks to StackSet** from the **Actions** menu.
5. On the **Set deployment options** page, do the following:

   a. For **Add stacks to StackSet**, choose **Deploy new stacks**.

   b. Next, do the following depending on your StackSet's permissions configuration:

   - [Self-managed permissions] For **Accounts**, **Deployment locations**, choose **Deploy stacks in accounts**. Paste your target account numbers in the text box, separating multiple numbers with commas.
   - [Service-managed permissions] For **Deployment targets**, do one of the following:
     - Choose **Deploy to organization** to deploy to all accounts in your organization.

- Choose **Deploy to organizational units (OUs)** to deploy to all accounts in specific OUs. Choose **Add another OU**, and then paste the target OU ID in the text box. Repeat for each new target OU. CloudFormation also targets any child OUs of your selected targets.

> **Note**
>
> If you add an OU that your StackSet already targets, CloudFormation creates new stacks in any accounts in the OU that don't already have stacks from your StackSet (for example, accounts that were added to the OU after your StackSet was created and with automatic deployments disabled).

c. For **Specify regions**, specify which AWS Regions to deploy to in the target accounts you specified in the previous step. By default, CloudFormation will deploy stacks in the specified accounts within the first Region, then moves on to the next, and so on, as long as a Region's deployment failures don't exceed a specified failure tolerance.

d. For **Deployment options**, do the following:

- For **Maximum concurrent accounts**, specify how many accounts are processed concurrently.
- For **Failure tolerance**, specify the maximum number of account failures allowed per Region. The operation will stop and won't proceed to other Regions once this limit is reached.
- For **Region concurrency**, choose how to process Regions: **Sequential** (one Region at a time) or **Parallel** (multiple Regions concurrently).
- For **Concurrency mode**, choose how concurrency behaves during operation execution.
  - **Strict failure tolerance** – Reduces account concurrency level when failures occur, staying within **Failure tolerance** +1.
  - **Soft failure tolerance** – Maintains your specified concurrency level (the value of **Maximum concurrent accounts**) regardless of failures.

e. Choose **Next**.

6. On the **Specify Overrides** page, leave the property values as specified. You won't be overriding any property values for the stacks you're going to create. Choose **Next**.
7. On the **Review** page, review your choices. To make changes, choose **Edit** on the related section.

8. When you're ready to proceed, choose **Submit**.

   CloudFormation starts creating your stacks. View the progress and status of the creation of the stacks in your StackSet in the StackSet details page that opens when you choose **Submit**. When complete, your new stacks should be listed on the **Stack instances** tab.

Add stacks to a StackSet (AWS CLI)

> **Note**
>
> When acting as a delegated administrator, you must include `--call-as DELEGATED_ADMIN` in the command.

To add stacks to a StackSet with self-managed permissions

Use the **create-stack-instances** CLI command. For the `--accounts` option, provide the accounts IDs for which you want to create stacks.

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --accounts account_id --regions eu-west-1 us-west-2
```

To add stacks to a StackSet with service-managed permissions

Use the **create-stack-instances** CLI command. For the `--deployment-targets` option, provide the organization (root) ID or OU IDs for which you want to create stacks. For example commands that target specific accounts, see Create a StackSet with service-managed permissions (AWS CLI).

```
aws cloudformation create-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-r1qi0wl7 --regions eu-west-1 us-
west-2
```

> **Note**
>
> If you add an OU that your StackSet already targets, CloudFormation creates new stacks in any accounts in the OU that don't already have stacks from your StackSet (for example, accounts that were added to the OU after your StackSet was created and with automatic deployments disabled).

Override parameter values on stacks within your CloudFormation StackSet

In certain cases, you might want stacks in certain Regions or accounts to have different property values than those specified in the StackSet itself. For example, you might want to specify a different value for a given parameter based on whether an account is used for development or production. For these situations, CloudFormation allows you to override parameter values in stacks by account and Region. You can override template parameter values when you first create the stacks, and you can override parameter values for existing stacks. You can only set parameters you've previously overridden in stacks back to the values specified in the StackSet.

Parameter value overrides apply to stacks in the accounts and Regions you select. During StackSet updates, any parameter values overridden for a stack aren't updated, but retain their overridden value.

You can only override parameter *values* that are specified in the StackSet; to add or delete a parameter itself, you need to update the StackSet template. If you add a parameter to a StackSet template, then before you can override that parameter value in a stack you must first update all stacks with the new parameter and value specified in the StackSet. Once all stacks have been updated with the new parameter, you can then override the parameter value in individual stacks as desired.

To learn how to override StackSet parameter values when you create stacks, see Add stacks to StackSets.

Topics

- Override parameters on stacks (console)
- Override parameters on stacks (AWS CLI)

Override parameters on stacks (console)

To override parameters for specific stacks

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the StackSet in.

3. From the navigation pane, choose **StackSets**. On the StackSets page, select your StackSet.
4. With the StackSet selected, choose **Override StackSet parameters** from the **Actions** menu.
5. On the **Set deployment options** page, provide the accounts and Regions for the stacks that you'll create overrides for.

   By default, CloudFormation will deploy stacks in the specified accounts within the first Region, then moves on to the next, and so on, provided that a Region's deployment failures don't exceed a specified failure tolerance.

   a. [Self-managed permissions] For **Deployment locations**, choose **Deploy stacks in accounts**. Paste some or all the target account IDs that you used to create your StackSet.

      [Service-managed permissions] Do one of the following:

      - Choose **Deploy to organizational units (OUs)**. Enter one or more of the target OUs that you used to create your StackSet. The overridden parameter values only apply to the accounts that are currently in the target OUs and their child OUs. Accounts added to the target OUs and their child OUs in the future will use the StackSet default values and not the overridden values.
      - Choose **Deploy to accounts**. Paste some or all the target OU IDs or account IDs that you used to create your StackSet.

   b. For **Specify regions**, add one or more of the Regions into which you have deployed stacks for this StackSet.

      If you add multiple Regions, the order of the Regions under **Specify regions** determines their deployment order.

   c. For **Deployment options**, do the following:

      - For **Maximum concurrent accounts**, specify how many accounts are processed concurrently.
      - For **Failure tolerance**, specify the maximum number of account failures allowed per Region. The operation will stop and won't proceed to other Regions once this limit is reached.
      - For **Region concurrency**, choose how to process Regions: **Sequential** (one Region at a time) or **Parallel** (multiple Regions concurrently).
      - For **Concurrency mode**, choose how concurrency behaves during operation execution.

- **Strict failure tolerance** – Reduces account concurrency level when failures occur, staying within **Failure tolerance** +1.
- **Soft failure tolerance** – Maintains your specified concurrency level (the value of **Maximum concurrent accounts**) regardless of failures.

d. Choose **Next**.

6. On the **Specify Overrides** page, select the checkboxes for the parameters to override, and then choose **Override StackSet value** from the **Edit override value** menu.
7. On the **Override StackSet parameter values** page, make your changes and then choose **Save changes**.

> **Note**
>
> To set any overridden parameters back to using the value specified in the StackSet, check all parameters and choose **Set to StackSet value** from the **Edit override value** menu. Doing so removes all overridden values once you update the stacks.

8. On the **Review** page, review your choices. To make changes, choose **Edit** on the related section.
9. When you're ready to proceed, choose **Submit**.

   CloudFormation starts updating your stacks. View the progress and status of the stacks in the StackSet details page that opens when you choose **Submit**.

Override parameters on stacks (AWS CLI)

> **Note**
>
> When acting as a delegated administrator, you must include `--call-as DELEGATED_ADMIN` in the command.

To override parameters for specific stacks

1. Use the update-stack-instances AWS CLI command and specify the `--parameter-overrides` option.

[Self-managed permissions] For the `--accounts` option, provide the account IDs for which you want to override parameter values on stacks.

```
aws cloudformation update-stack-instances --stack-set-name my-stackset \
  --parameter-overrides ParameterKey=Subnets,ParameterValue=subnet-1baa3351\
\,subnet-27b86940 \
  --accounts account_id --regions us-east-1
```

[Service-managed permissions] For the `--deployment-targets` option, provide the organization root ID, OU IDs, or AWS Organizations account IDs for which you want to override parameters on stacks. In this example, we override parameter values for stacks in all accounts in the OU with the *ou-rcuk-1x5j1lwo* ID.

The overridden parameter values only apply to the accounts that are currently in the target OU and its child OUs. Accounts added to the target OU and its child OUs in the future will use the StackSet default values and not the overridden values.

```
aws cloudformation update-stack-instances --stack-set-name my-stackset \
  --parameter-overrides ParameterKey=Subnets,ParameterValue=subnet-1baa3351\
\,subnet-27b86940 \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5j1lwo \
  --regions us-east-1
```

2. Verify that your parameter values were successfully overridden on stacks by running the **describe-stack-set-operation** command to show the status and results of your update operation. For `--operation-id`, use the operation ID that was returned by your **update-stack-instances** command.

```
aws cloudformation describe-stack-set-operation --operation-id operation_ID
```

Delete stacks from CloudFormation StackSets

You can delete stacks from StackSets using either the CloudFormation console or the AWS CLI.

Topics

- Delete stacks from your StackSet (console)
- Delete stacks from your StackSet (AWS CLI)

> **Note**
>
> Deleting stacks from a top-level organizational unit (OU) removes that OU as a StackSet target.

Delete stacks from your StackSet (console)

To delete stacks

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the StackSet in.
3. From the navigation pane, choose **StackSets**. On the StackSets page, select the StackSet.
4. With your StackSet selected, choose **Delete stacks from StackSet** from the **Actions** menu.
5. On the **Set deployment options** page, first choose the accounts and Regions where you want to delete the stacks.

   a. [Self-managed permissions] For **Accounts**, choose **Deploy stacks in accounts** or **Deploy stacks in organizational units**.

   If you choose **Deploy stacks in accounts**, paste your target account numbers in the **Account numbers** text box, separating multiple numbers with commas.

   If you choose **Deploy stacks in organizational units**, paste a target OU ID in the **Organization numbers** text box to target all accounts that are part of the specified organization.

   b. [Service-managed permissions] For **Organizational units (OUs)**, specify the target OU IDs.

   > **Important**
   >
   > CloudFormation will delete stacks from both the specified target OUs and their child OUs.

For **Account filter type**, you can refine which accounts will have stacks deleted by choosing one of the following options and providing account numbers.

- **None** (default) – Delete stacks from all accounts in the specified OUs.
- **Intersection** – Delete stacks only from specific individual accounts within the selected OUs.
- **Difference** – Delete stacks from all accounts in the selected OUs except for specific accounts.
- **Union** – Delete stacks from the specified OUs plus additional individual accounts.

c. For **Specify regions**, choose the Regions from which you want to delete stacks within the target accounts.

6. For **Deployment options**, do the following:

- For **Maximum concurrent accounts**, specify how many accounts are processed concurrently.
- For **Failure tolerance**, specify the maximum number of account failures allowed per Region. The operation will stop and won't proceed to other Regions once this limit is reached.
- For **Retain stacks**, enable this option to save the stacks and their associated resources when removing them from your StackSet. The resources stay in their current state but are no longer part of the StackSet.
- For **Region concurrency**, choose how to process Regions: **Sequential** (one Region at a time) or **Parallel** (multiple Regions concurrently).
- For **Concurrency mode**, choose how concurrency behaves during operation execution.
  - **Strict failure tolerance** – Reduces account concurrency level when failures occur, staying within **Failure tolerance** +1.
  - **Soft failure tolerance** – Maintains your specified concurrency level (the value of **Maximum concurrent accounts**) regardless of failures.

7. Choose **Next**.
8. On the **Review** page, review your choices. To make changes, choose **Edit** on the related section.
9. When you are ready to remove the stacks from your StackSet, choose **Submit**.

After stack deletion is finished, you can verify that stacks were deleted from your StackSet in the StackSet detail page, on the **Stack instances** tab.

Delete stacks from your StackSet (AWS CLI)

> **Note**
>
> When acting as a delegated administrator, you must include `--call-as DELEGATED_ADMIN` in the command.

Use the **delete-stack-instances** command with your StackSet name.

In these examples, we use the `--no-retain-stacks` option because we aren't retaining any stacks. Use `--retain-stacks` instead of `--no-retain-stacks` if you want to keep the stacks and their resources.

For `--regions`, specify the AWS Regions you want to delete stacks from, for example, `us-west-2` and `us-east-1`.

Set concurrent account processing and other preferences using the `--operation-preferences` option. These examples use count-based settings. Note that `MaxConcurrentCount` must not exceed `FailureToleranceCount` + 1. For percentage-based settings, use `FailureTolerancePercentage` or `MaxConcurrentPercentage` instead.

To delete stacks (self-managed permissions)

For the `--accounts` option, specify the IDs of the account to delete stacks from.

```
aws cloudformation delete-stack-instances --stack-set-name my-stackset \
  --accounts account_ID_1 account_ID_2 \
  --regions us-west-2 us-east-1 \
  --no-retain-stacks \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

To delete stacks (service-managed permissions)

For `--deployment-targets`, specify the organization root ID or organizational unit (OU) IDs to delete stacks from.

> **Important**
>
> CloudFormation will delete stacks from both the specified target OUs and their child OUs.

```
aws cloudformation delete-stack-instances --stack-set-name my-stackset \
  --deployment-targets OrganizationalUnitIds=ou-rcuk-1x5jlwo,ou-rcuk-slr5lh0a \
  --regions us-west-2 us-east-1 \
  --no-retain-stacks \
  --operation-preferences MaxConcurrentCount=1,FailureToleranceCount=0
```

For more information, see DeleteStackInstances in the *AWS CloudFormation API Reference*.

Optionally, after stack deletion is finished, verify that stacks were deleted from your StackSet by running the **describe-stack-set-operation** command to show the status and results of the delete stacks operation. For `--operation-id`, use the operation ID that was returned by your **delete-stack-instances** command.

```
aws cloudformation describe-stack-set-operation --stack-set-name my-stackset \
  --operation-id ddf16f54-ad62-4d9b-b0ab-3ed8e9example
```

Delete CloudFormation StackSets

To delete a StackSet, you must first delete all stacks in the StackSet. For information about how to delete all stacks, see Delete stacks from StackSets.

Topics

- Delete a StackSet (console)
- Delete a StackSet (AWS CLI)
- Delete service roles (optional)

Delete a StackSet (console)

To delete a StackSet

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region you created the StackSet in.
3. On the **StackSets** page, select the StackSet.
4. With the StackSet selected, choose **Delete StackSet** from the **Actions** menu.

5. When you are prompted to confirm that you want to delete the StackSet, choose **Delete StackSet**.

Delete a StackSet (AWS CLI)

> **Note**
>
> When acting as a delegated administrator, you must include `--call-as DELEGATED_ADMIN` in the command.

To delete a StackSet

1. Use the following **delete-stack-set** command. When you are prompted to confirm, type **y**, and then press **Enter**.

```
aws cloudformation delete-stack-set --stack-set-name my-stackset
```

2. Verify that the StackSet was deleted by running the **list-stack-sets** command. The results of the list-stack-sets command should show your stack with a status of `DELETED`.

```
aws cloudformation list-stack-sets
```

Delete service roles (optional)

If you no longer need the IAM service roles that CloudFormation requires to perform StackSet operations, we recommend that you delete the roles.

For self-managed StackSets, the roles you created. For more information about these roles, see Grant self-managed permissions.

For service-managed StackSets, the roles that were automatically created for StackSets have the suffix `CloudFormationStackSetsOrgAdmin` in the organization management account, and `CloudFormationStackSetsOrgMember` in each target account. For more information, see Service-linked roles.

To delete a service role (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. From the navigation pane, choose **Roles**, and then fill the check box next to the role that you want to delete.
3. In the **Role actions** menu at the top of the page, choose **Delete role**.
4. In the confirmation dialog box, choose **Yes, Delete**. If you are sure, you can proceed with the deletion even if the service last accessed data is still loading.

To delete a service role (AWS CLI)

- Use the following **delete-role** command. When you are prompted to confirm, type **y**, and then press **Enter**.

```
aws iam delete-role --role-name role name
```

For more information about deleting roles, see Delete roles or instance profiles in the *IAM User Guide*.

Prevent failed StackSets deployments using target account gates

An account gate is an optional feature that helps you verify that a target account meets certain requirements before CloudFormation begins StackSets operations in that account. This verification is performed through an AWS Lambda function that acts as a prerequisite check.

A common example of an account gate is verifying that there are no CloudWatch alarms active or unresolved on the target account. CloudFormation invokes the Lambda function each time you start stack operations in the target account, and only continues if the function returns a SUCCEEDED code. If the Lambda function returns a status of FAILED, CloudFormation doesn't continue with your requested operation. If you don't have an account gating Lambda function configured, CloudFormation skips the check, and continues with your operation.

If your target account fails an account gate check, the failed operation counts toward your specified failure tolerance number or percentage of stacks. For more information about failure tolerance, see StackSet operation options.

Account gating is only available for StackSets operations. This feature isn't available for other CloudFormation operations outside of StackSets.

Requirements

The following requirements must be met for account gating:

- Your Lambda function must be named `AWSCloudFormationStackSetAccountGate` to use this feature.
- The **AWSCloudFormationStackSetExecutionRole** must have permissions to invoke your Lambda function. Without these permissions, CloudFormation will skip the account gating check and proceed with stack operations.
- The Lambda `InvokeFunction` permission must be added to target accounts for account gating to work. The target account trust policy must have a trust relationship with the administrator account. The following is an example of a policy statement that grants Lambda `InvokeFunction` permissions.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "*"
        }
    ]
}
```

CloudFormation templates for creating Lambda functions

Use the following sample templates to create Lambda `AWSCloudFormationStackSetAccountGate` functions. To create a new stack using either of these templates, see Create a stack from the CloudFormation console.

Template location	Description
https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AccountGateSucceeded.yml	Creates a stack that implements a Lambda account gate function that will return a status of `SUCCEEDED` .
https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/AccountGateFailed.yml	Creates a stack that implements a Lambda account gate function that will return a status of `FAILED`.

Choose the Concurrency Mode for CloudFormation StackSets

Concurrency Mode is a parameter for StackSetOperationPreferences that allows you to choose how the concurrency level behaves during StackSet operations. You can choose between the following modes:

- **Strict Failure Tolerance**: This option dynamically lowers the concurrency level to ensure the number of failed accounts never exceeds the value of **Failure tolerance** +1. The initial actual concurrency is set to the lower of either the value of the **Maximum concurrent accounts**, or the value of **Failure tolerance** +1. The actual concurrency is then reduced proportionally by the number of failures. This is the default behavior.
- **Soft Failure Tolerance**: This option decouples **Failure tolerance** from the actual concurrency. This allows StackSet operations to run at the concurrency level set by the **Maximum concurrent accounts** value, regardless of the number of failures.

Strict Failure Tolerance lowers the deployment speed as StackSet operation failures occur because concurrency decreases for each failure. **Soft Failure Tolerance** prioritizes deployment speed while still leveraging CloudFormation safety capabilities. This allows you to review and address StackSet operation failures for common issues such as those related to existing resources, service quotas, and permissions.

For more information on StackSets stack operation failures, see Common reasons for stack operation failure.

For more information on **Maximum concurrent accounts** and **Failure tolerance**, see StackSet operation options.

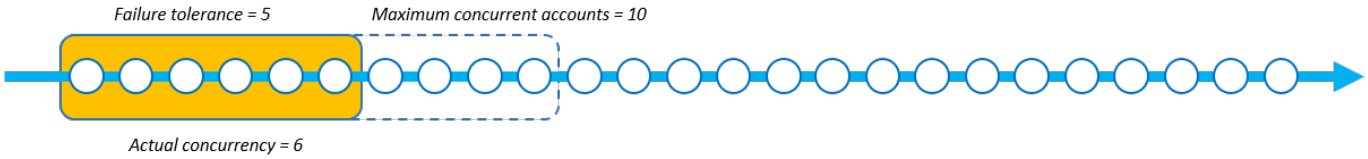
How each Concurrency Mode works

The images below provide a visual representation of how each **Concurrency Mode** works during a StackSet operation. The string of nodes represents a deployment to single AWS Region and each node is a target AWS account.

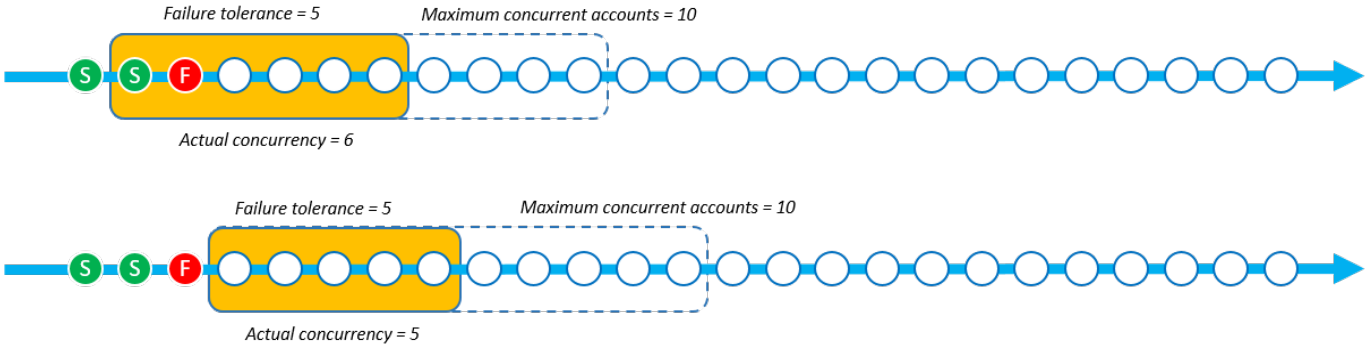
Strict Failure Tolerance

When a StackSet operation using **Strict Failure Tolerance** has the **Failure tolerance** value set to 5 and the **Maximum concurrent accounts** value set to 10, the actual concurrency is 6. The actual concurrency is 6 because this the **Failure tolerance** value of 5 +1 is lower than the value of **Maximum concurrent accounts**.

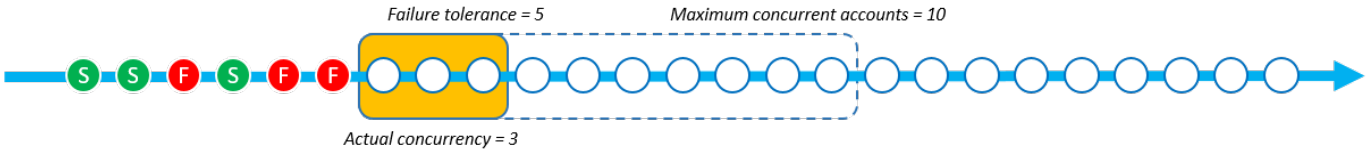
The following image shows the impact that the **Failure tolerance** value has on the **Maximum concurrent accounts** value, and the impact they both have on the actual concurrency of the StackSet operation:



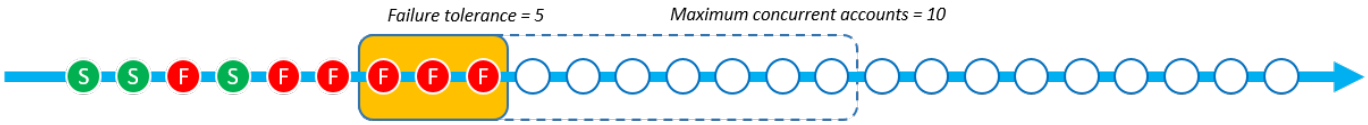
When deployment begins and there are failed stack instances, then the actual concurrency reduces to provide a safe deployment experience. The actual concurrency reduces from 6 to 5 when StackSets fails to deploy 1 stack instance.



The **Strict Failure Tolerance** mode reduces the actual concurrency proportionally to the number of failed stack instances. In the following example, the actual concurrency reduces from 5 to 3 when StackSets fails to deploy 2 more stack instances, bringing the total of failed stack instances to 3.



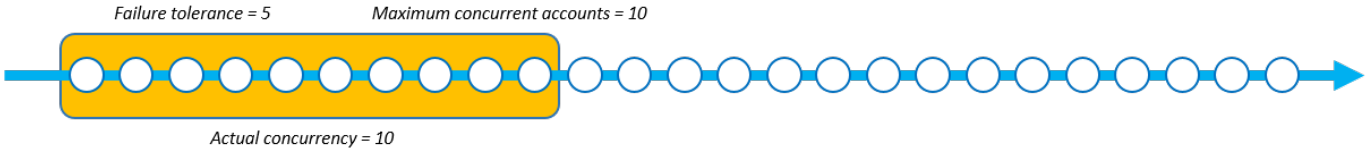
StackSets fails the StackSet operation when the number of failed stack instances equals the defined value of **Failure tolerance** +1. In the following example, StackSets fails the operation when there are 6 failed stack instances and the **Failure tolerance** value is 5.



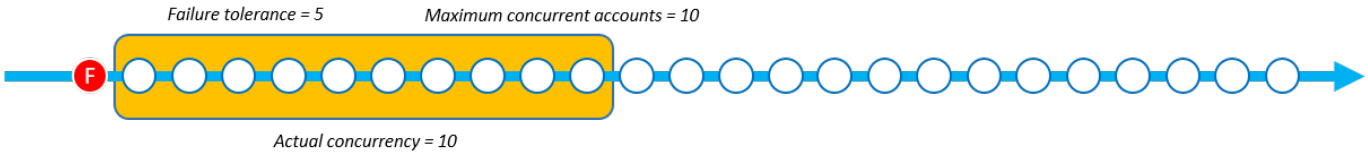
In this example, CloudFormation deployed 9 stack instances (3 successful and 6 failed) before stopping the StackSet operation.

Soft Failure Tolerance

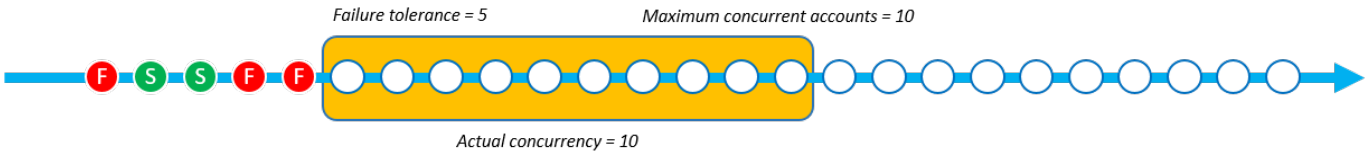
When a StackSet operation using **Soft Failure Tolerance** has the **Failure tolerance** value set to 5 and the **Maximum concurrent accounts** value set to 10, the actual concurrency is 10.



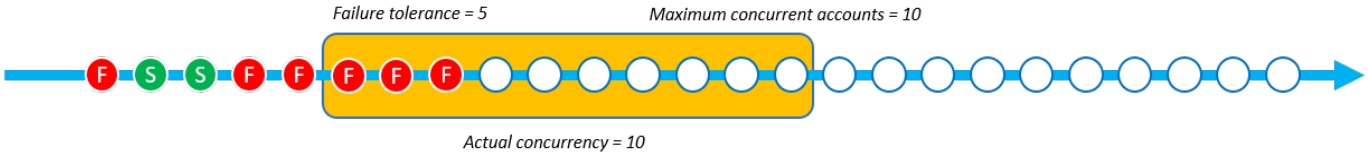
When deployment begins and there are failed stack instances, the actual concurrency doesn't change. In the following example, 1 stack operation failed, but the actual concurrency remains at 10.



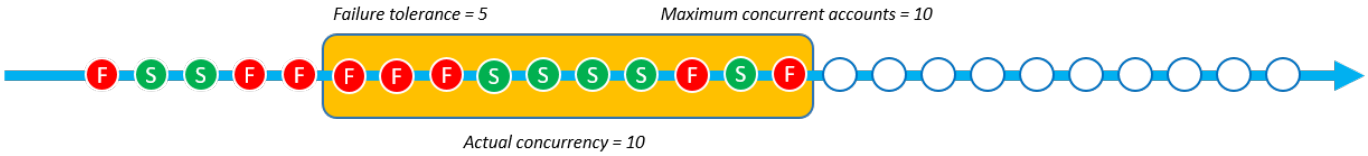
The actual concurrency remains at 10 even after 2 more stack instance failures.



StackSets fails the StackSet operation when failed stack instances exceeds the **Failure tolerance** value. In the following example, StackSets fails the operation when there are 6 failed stack instances and the **Failure tolerance** count is 5. However, the operation won't end until the remaining operations in the concurrency queue finish.



StackSets continues to deploy stack instances that are already in the concurrency queue. This means that the number of failed stack instances can be higher than **Failure tolerance**. In the following example, there are 8 failed stack instances because the concurrency queue still had 7 operations left to perform, even though the StackSet operation had reached the **Failure tolerance** of 5.



In this example, StackSets deployed 15 stack instances (7 successful and 8 failed) before stopping the stack operation.

Choosing between Strict failure tolerance and Soft failure tolerance based on deployment speed

Choosing between **Strict failure tolerance** and **Soft failure tolerance** modes depends on the preferred speed of your StackSet deployment and the permissible number of deployment failures.

The following tables show how each concurrency mode handles a StackSet operation that fails while trying to deploy 1000 total stack instances. In each scenario, the **Failure tolerance** value is set to 100 stack instances and the **Maximum concurrent accounts** value is set to 250 stack instances.

While StackSets actually queues accounts as a sliding window (see How each Concurrency Mode works), this example shows the operation in batches to demonstrate the speed of each mode.

Strict failure tolerance

This example using **Strict failure tolerance** mode lowers the actual concurrency relative to the number of failures that occur in each preceding batch. Each batch has 20 failed instances, which then lowers the actual concurrency of the following batch by 20 until the StackSet operation reaches the **Failure tolerance** value of 100.

In following table, the initial actual concurrency of the first batch is 101 stack instances. The actual concurrency is 101 because it's the lower value of either the **Maximum concurrent accounts** (250) and the **Failure tolerance** (100) +1. Each batch contains 20 failed stack instance deployments, which then lowers the actual concurrency of each following batch by 20 stack instances.

Strict failure tolerance	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Batch 6
Actual concurrency count	101	81	61	41	21	-
Failed instance count	20	20	20	20	20	-
Successful stack instance count	81	61	41	21	1	-

The operation using **Strict failure tolerance** completed 305 stack instance deployments in 5 batches by the time the StackSet operation reached the **Failure tolerance** of 100 stack instances. The StackSet operation successfully deploys 205 stack instances before it fails.

Soft failure tolerance

This example using **Soft failure tolerance** mode maintains the same actual concurrency count defined by the **Maximum concurrent accounts** value of 250 stack instances, regardless of the number of failed instances. The StackSet operations keeps the same actual concurrency until it reaches the **Failure tolerance** value of 100 instances.

In following table, the initial actual concurrency of the first batch is 250 stack instances. The actual concurrency is 250 because the **Maximum concurrent accounts** value is set to 250 and **Soft failure tolerance** mode allows StackSets to use this value as the actual concurrency, regardless of the number of failures. Even though there are 50 failures in each of the batches for this example, the actual concurrency remains unaffected.

Soft failure tolerance	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Batch 6
Actual concurrency count	250	250	-	-	-	-
Failed instance count	50	50	-	-	-	-
Successful stack instance count	200	200	-	-	-	-

Using the same **Maximum concurrent accounts** value and **Failure tolerance** value, the operation using **Soft failure tolerance** mode completed 500 stack instance deployments in 2 batches. The StackSet operation successfully deploys 400 stack instances before it fails.

Choosing your Concurrency Mode (console)

When creating or updating a StackSet, on the **Set deployment options** page, for **Concurrency mode**, choose **Strict failure tolerance** or **Soft failure tolerance**.

Choosing your Concurrency Mode (AWS CLI)

You can use the `ConcurrencyMode` parameter with the following StackSets commands:

- create-stack-instances
- delete-stack-instances
- detect-stack-set-drift
- import-stacks-to-stack-set
- update-stack-instances
- update-stack-set

These commands have an existing parameter called `--operation-preferences` that can use the `ConcurrencyMode` setting. `ConcurrencyMode` can be set to one of the following values:

- `STRICT_FAILURE_TOLERANCE`
- `SOFT_FAILURE_TOLERANCE`

The following example creates a stack instance using the `STRICT_FAILURE_TOLERANCE` `ConcurrencyMode`, with a `FailureToleranceCount` set to 10 and a `MaxConcurrentCount` set to 5.

```
aws cloudformation create-stack-instances \
  --stack-set-name example-stackset \
  --accounts 123456789012 \
  --regions eu-west-1  \
  --operation-preferences
 ConcurrencyMode=STRICT_FAILURE_TOLERANCE,FailureToleranceCount=10,MaxConcurrentCount=5
```

Note

For detailed procedures for creating and updating a StackSet, see the following topics:

- Create StackSets (self-managed permissions)
- Create StackSets (service-managed permissions)
- Update StackSets

Performing drift detection on CloudFormation StackSets

Even as you manage your stacks and the resources they contain through CloudFormation, users can change those resources outside of CloudFormation. Users can edit resources directly by using the underlying service that created the resource. By performing drift detection on a StackSet, you can determine if any of the stack instances belonging to that StackSet differ, or have *drifted*, from their expected configuration.

Topics

- How CloudFormation performs drift detection on a StackSet
- Detect drift on a StackSet (console)
- Detect drift on a StackSet (AWS CLI)
- Stopping drift detection on a StackSet

How CloudFormation performs drift detection on a StackSet

When CloudFormation performs drift detection on a StackSet, it performs drift detection on the stack associated with each stack instance in the StackSet. To do this, CloudFormation compares the current state of each resource in the stack with the expected state of that resource, as defined in the stack's template and any specified input parameters. If the current state of a resource varies from its expected state, that resource is considered to have drifted. If one or more resources in a stack have drifted, then the stack itself is considered to have drifted, and the stack instances that the stack is associated with is considered to have drifted as well. If one or more stack instances in a StackSet have drifted, the StackSet itself is considered to have drifted.

Drift detection identifies unmanaged changes; that is, changes made to stacks outside of CloudFormation. Changes made through CloudFormation to a stack directly, rather than at the StackSet level, aren't considered drift. For example, suppose you have a stack that is associated with a stack instance of a StackSet. If you use CloudFormation to update that stack to use a different template, that is not considered drift, even though that stack now has a different template than any other stacks belonging to the StackSet. This is because the stack still matches its expected template and parameter configuration in CloudFormation.

For detailed information on how CloudFormation performs drift detection on a stack, see Detect unmanaged configuration changes to stacks and resources with drift detection.

Because CloudFormation performs drift detection on each stack individually, it takes any overridden parameter values into account when determining whether a stack has drifted. For more information on overriding template parameters in stack instances, see Override parameter values on stacks within your CloudFormation StackSet.

If you perform drift detection directly on a stack that is associated with a stack instance, those drift results aren't available from the **StackSets** console page.

Detect drift on a StackSet (console)

To detect drift on a StackSet

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **StackSets** page, select the StackSet on which you want to perform drift detection.
3. From the **Actions** menu, select **Detect drifts**.

   CloudFormation displays an information bar stating that drift detection has been initiated for the selected StackSet.
4. Optional: To monitor the progress of the drift detection operation:

   a. Select the StackSet name to display the **Stackset details** page.

   b. Select the **Operations** tab, select the drift detection operation, and then select **View drift details**.

   CloudFormation displays the **Operation details** dialog box.
5. Wait until CloudFormation completes the drift detection operation. When the drift detection operation completes, CloudFormation updates **Drift status** and **Last drift check time** for your StackSet. These fields are listed on the **Overview** tab of the **StackSet details** page for the selected StackSet.

   The drift detection operation may take some time, depending on the number of stack instances included in the StackSet, and the number of resources included in the StackSet. You can only run a single drift detection operation on a given StackSet at one time. CloudFormation continues the drift detection operation even after you dismiss the information bar.
6. To review the drift detection results for the stack instances in a StackSet, select the **Stack instances** tab.

The **Stack name** column lists the name of the stack associated with each stack instance, and the **Drift status** column lists the drift status of that stack. A stack is considered to have drifted if one or more of its resources have drifted.

7. To review the drift detection results for the stack associated with a specific stack instance:

   a. Choose the **Operations** tab.

   b. Select the drift operation you want to view drift detection results for. A split panel will display the stack instance status and reason for the selected operation. For a drift operation, the status reason column shows the drift status of a stack instance.

   c. Choose the stack instance you want to view drift details for, and choose **View resource drifts**. In the **Resource drift status** table on the **Resource Drifts** page, each stack resource is listed with its drift status and the last time drift detection was initiated on the resource. The logical ID and physical ID of each resource is displayed to help you identify them.

8. You can sort the resources based on their drift status using the **Drift status** column.

   To view the details on a modified resource:

   - With the resource selected, choose **View drift details**.

     CloudFormation displays the drift detail page for that particular resource. This page lists the resource's differences. It also lists the resource's expected and current property values.

     > **Note**
     >
     > If the stack belongs to a different Region and account than the one you're currently signed into, the **Detect drift** button will be disabled and you will be unable to view the details.

Detect drift on a StackSet (AWS CLI)

To detect drift on an entire stack using the AWS CLI, use the following procedure:

To detect drift on a StackSet

1. Use the detect-stack-set-drift command to detect drift on an entire StackSet and its associated stack instances.

The following example initiates drift detection on the StackSet `stack-set-drift-example`.

```
aws cloudformation detect-stack-set-drift \
    --stack-set-name stack-set-drift-example
```

Output:

```
{
    "OperationId": "c36e44aa-3a83-411a-b503-cb611example"
}
```

2. Because StackSet drift detection operations can be a long-running operation, use the describe-stack-set-operation command to monitor the status of drift operation. This command takes the StackSet operation ID returned by the **detect-stack-set-drift** command.

   The following examples uses the operation ID from the previous example to return information on the StackSet drift detection operation. In this example, the operation is still running. Of the seven stack instances associated with this StackSet, one stack instance has already been found to have drifted, two instances are in sync , and drift detection for the remaining four stack instances is still in progress. Because one instance has drifted, the drift status of the StackSet itself is now `DRIFTED`.

   ```
   aws cloudformation describe-stack-set-operation \
       --stack-set-name stack-set-drift-example \
       --operation-id c36e44aa-3a83-411a-b503-cb611example
   ```

   Output:

   ```
   {
       "StackSetOperation": {
           "Status": "RUNNING",
           "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
   AWSCloudFormationStackSetAdministrationRole",
           "OperationPreferences": {
               "RegionOrder": []
           },
           "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
           "StackSetDriftDetectionDetails": {
               "DriftedStackInstancesCount": 1,
   ```

```
            "TotalStackInstancesCount": 7,
            "LastDriftCheckTimestamp": "2019-12-04T20:34:28.543Z",
            "InSyncStackInstancesCount": 2,
            "InProgressStackInstancesCount": 4,
            "DriftStatus": "DRIFTED",
            "FailedStackInstancesCount": 0
        },
        "Action": "DETECT_DRIFT",
        "CreationTimestamp": "2019-12-04T20:33:13.673Z",
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22example",
        "OperationId": "c36e44aa-3a83-411a-b503-cb611example"
    }
}
```

Performing the same command later, this example shows the information returned once the drift detection operation has completed. Two of the seven total stack instances associated with this StackSet have drifted, rendering the drift status of the StackSet itself as `DRIFTED`.

```
aws cloudformation describe-stack-set-operation \
    --stack-set-name stack-set-drift-example \
    --operation-id c36e44aa-3a83-411a-b503-cb611example
```

Output:

```
{
    "StackSetOperation": {
        "Status": "SUCCEEDED",
        "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
AWSCloudFormationStackSetAdministrationRole",
        "OperationPreferences": {
            "RegionOrder": []
        }
        "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
        "EndTimestamp": "2019-12-04T20:37:32.829Z",
        "StackSetDriftDetectionDetails": {
            "DriftedStackInstancesCount": 2,
            "TotalStackInstancesCount": 7,
            "LastDriftCheckTimestamp": "2019-12-04T20:36:55.612Z",
            "InSyncStackInstancesCount": 5,
            "InProgressStackInstancesCount": 0,
            "DriftStatus": "DRIFTED",
```

```
            "FailedStackInstancesCount": 0
        },
        "Action": "DETECT_DRIFT",
        "CreationTimestamp": "2019-12-04T20:33:13.673Z",
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22example",
        "OperationId": "c36e44aa-3a83-411a-b503-cb611example"
    }
}
```

3. When the StackSet drift detection operation is complete, use the **describe-stack-set**, **list-stack-instances**, **describe-stack-instance**, and **list-stack-instance-resource-drifts** commands to review the results.

   The describe-stack-set command includes the same detailed drift information returned by the **describe-stack-set-operation** command.

   ```
   aws cloudformation describe-stack-set \
       --stack-set-name stack-set-drift-example
   ```

   Output:

   ```
   {
       "StackSet": {
           "Status": "ACTIVE",
           "Description": "Demonstration of drift detection on stack sets.",
           "Parameters": [],
           "Tags": [
               {
                   "Value": "Drift detection",
                   "Key": "Feature"
               }
           ],
           "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
           "Capabilities": [],
           "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
   AWSCloudFormationStackSetAdministrationRole",
           "StackSetDriftDetectionDetails": {
               "DriftedStackInstancesCount": 2,
               "TotalStackInstancesCount": 7,
               "LastDriftCheckTimestamp": "2019-12-04T20:36:55.612Z",
               "InProgressStackInstancesCount": 0,
   ```

```
            "DriftStatus": "DRIFTED",
            "DriftDetectionStatus": "COMPLETED",
            "InSyncStackInstancesCount": 5,
            "FailedStackInstancesCount": 0
        },
        "StackSetARN": "arn:aws:cloudformation:us-east-1:123456789012:stackset/
stack-set-drift-example:bd1f4017-d4f9-432e-a73f-8c22example",
        "TemplateBody": [details omitted],
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22ebexample",
        "StackSetName": "stack-set-drift-example"
    }
}
```

You can use the list-stack-instances command to return summary information about the stack instances associated with a StackSet, including the drift status of each stack instance.

In this example, executing **list-stack-instances** on the example StackSet with the drift status filter set to `DRIFTED` enables you to identify which two stack instances have a drift status of `DRIFTED`.

```
aws cloudformation list-stack-instances \
    --stack-set-name stack-set-drift-example \
    --filters Name=DRIFT_STATUS,Values=DRIFTED
```

Output:

```
{
"Summaries": [
        {
"StackId": "arn:aws:cloudformation:eu-west-1:123456789012:stack/
StackSet-stack-set-drift-example-b0fb6083-60c0-4e39-
af15-2f071e0db90c/0e4f0940-16d4-11ea-93d8-0641cexample",
            "Status": "CURRENT",
            "Account": "012345678910",
            "Region": "eu-west-1",
            "LastDriftCheckTimestamp": "2019-12-04T20:37:32.687Z",
            "DriftStatus": "DRIFTED",
            "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
            "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
        },
```

```
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
StackSet-stack-set-drift-example-b7fde68e-e541-44c2-b33d-
ef2e2988071a/008e6030-16d4-11ea-8090-12f89example",
            "Status": "CURRENT",
            "Account": "123456789012",
            "Region": "us-east-1",
            "LastDriftCheckTimestamp": "2019-12-04T20:34:28.275Z",
            "DriftStatus": "DRIFTED",
            "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
            "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
        },

        [additional stack instances omitted]

    ]
}
```

The describe-stack-instance command also returns this information, but for a single stack instance, as in the example below.

```
aws cloudformation describe-stack-instance \
    --stack-set-name stack-set-drift-example \
    --stack-instance-account 012345678910 --stack-instance-region us-east-1
```

Output:

```
{
    "StackInstance": {
        "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
StackSet-stack-set-drift-example-b7fde68e-e541-44c2-b33d-
ef2e2988071a/008e6030-16d4-11ea-8090-12f89example",
        "Status": "CURRENT",
        "Account": "123456789012",
        "Region": "us-east-1",
        "ParameterOverrides": [],
        "DriftStatus": "DRIFTED",
        "LastDriftCheckTimestamp": "2019-12-04T20:34:28.275Z",
        "StackSetId": "stack-set-drift-example:bd1f4017-d4f9-432e-
a73f-8c22eexample",
        "LastOperationId": "c36e44aa-3a83-411a-b503-cb611example"
```

```
    }
}
```

4. Once you've identified which stack instances have drifted, you can use the information about the stack instances that is returned by the **list-stack-instances** or **describe-stack-instance** commands to run the list-stack-instance-resource-drifts command. This command returns detailed information about which resources in the stack have drifted for a particular drift operation.

   The following example uses the `--stack-instance-resource-drift-statuses` parameter to request stack drift information for the resources that have been modified or deleted in the previous drift operation example. The request returns information on the one resource that has been modified, including details about two of its properties and their changed values. No resources have been deleted.

```
aws cloudformation list-stack-instance-resource-drifts \
    --stack-set-name my-stack-set-with-resource-drift \
    --stack-instance-account 123456789012 \
    --stack-instance-region us-east-1 \
    --operation-id c36e44aa-3a83-411a-b503-cb611example \
    --stack-instance-resource-drift-statuses MODIFIED DELETED
```

   Output:

```
{
    "Summaries": [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/my-
stack-set-with-resource-drift/489e5570-df85-11e7-a7d9-50example",
            "ResourceType": "AWS::SQS::Queue",
            "Timestamp": "2018-03-26T17:23:34.489Z",
            "PhysicalResourceId": "https://sqs.us-
east-1.amazonaws.com/123456789012/my-stack-with-resource-drift-Queue-494PBHCO76H4",
            "StackResourceDriftStatus": "MODIFIED",
            "PropertyDifferences": [
                {
                    "PropertyPath": "/DelaySeconds",
                    "ActualValue": "120",
                    "ExpectedValue": "20",
                    "DifferenceType": "NOT_EQUAL"
                },
```

```
                {
                    "PropertyPath": "/RedrivePolicy/maxReceiveCount",
                    "ActualValue": "12",
                    "ExpectedValue": "10",
                    "DifferenceType": "NOT_EQUAL"
                }
            ],
            "LogicalResourceId": "Queue"
        }
    ]
}
```

Stopping drift detection on a StackSet

Because drift detection on a StackSet can be a long-running operation, there may be instances when you want to stop a drift detection operation that is currently running on a StackSet.

To stop drift detection on a StackSet (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the **StackSets** page, select the name of the StackSet.

   CloudFormation displays the **StackSets details** page for the selected StackSet.
3. On the **StackSets details** page, select the **Operations** tab, and then select the drift detection operation.
4. Select **Stop operation**.

To stop drift detection on a StackSet (AWS CLI)

- Use the stop-stack-set-operation command. You must supply both the StackSet name and the operation ID of the drift detection StackSet operation.

```
aws cloudformation stop-stack-set-operation \
    --stack-set-name stack-set-drift-example \
    --operation-id 624af370-311a-11e8-b6b7-500cexample
```

Import stacks into CloudFormation StackSets

A stack import operation can import existing stacks into new or existing StackSets, so that you can migrate existing stacks to a StackSet in one operation.

For *self-managed* StackSets, the import operation can import stacks in the administrator account or in different target accounts and AWS Regions. For *service-managed* StackSets, the import operation can import any stack in the same AWS Organizations as the management account.

The following are considerations and limitations when importing stacks into StackSets:

- The import operation can import up to 10 stacks using inline stack IDs or up to 200 stacks using an Amazon S3 object.
- The `NoEcho` property is not supported. Stacks that contain `NoEcho` won't be imported into new StackSets through StackSet import.
- Stacks can only belong to one StackSet.
- You can implement stack tags to the StackSet by specifying tags explicitly as parameters in the stack import operation.
- A stack's custom parameter overrides aren't affected during the import operation.
- Parameters of imported stack instances can't be updated using the **Edit StackSet details** option. You must use the **Override StackSet parameters**. For more information on overriding parameters, see Override parameters on stacks.
- The StackSets quotas and stack instances apply when importing stacks. For more information about quotas, see Understand CloudFormation quotas.

Topics

- Self-managed stack import for CloudFormation StackSets
- Service-managed stack import for CloudFormation StackSets
- Revert stack imports into CloudFormation StackSets

Self-managed stack import for CloudFormation StackSets

The CloudFormation stack import operation can import existing stacks into new or existing StackSets, so that you can migrate existing stacks to a StackSet in one operation. By using stack

import, you avoid downtime and outages without deleting and recreating those resources. Once the stack is imported into a StackSet, the original stack will become a stack instance of the specified stack set.

Considerations for self-managed stack imports

- The stack import operation requires an administrator account in which you create a StackSet and a target account that contains a stack.
- The target account must have permission to use the `GetTemplate` operation with the input of stack ID or ARN. Because of that, your administrator account must be granted **AWSCloudFormationStackSetAdministrationRole** or **AWSCloudFormationStackSetsExectionRole** permissions.

Topics

- Import an existing stack into a new StackSet (console)
- Import an existing stack into an existing StackSet (console)
- Import a stack into a StackSet (AWS CLI)

Import an existing stack into a new StackSet (console)

Before you begin, identify the stack that you want to import.

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the navigation pane, choose **StackSets**.
3. At the top of the **StackSets** page, choose **Create StackSet**.
4. On the **Choose a template** page, specify a template by one of the following options and choose **Next**.
   - Choose **Amazon S3 URL** and specify the URL for your template in the text box.
   - Choose **Upload a template file** and browse for your template.
   - Choose **From stack ID** and enter your stack ID.
5. On the **Specify StackSet details** page, enter the name of a StackSet you want to create and choose **Next**.

   (Optional) Enter a description of the StackSet.

6. On the **Configure StackSet options** page, review your choices and choose **Next**.
7. On the **Set deployment options** page, choose **Import stacks to stack set**.
8. Enter the stack ID of the stack you want to import in the **Stacks to import** field. For example, *arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/f449b250-b969-11e0-a185-5081d0136786*.

   (Optional) Choose **Add another stack ID** and enter the stack ID of another stack you want to import. You may add up to 10 stacks per stack import operation.
9. Review your deployment options and choose **Next**.
10. On the **Review** page, review your choices and your StackSet's properties. When you are ready to import your stack into your StackSet, choose **Submit**.

Results: The imported stack is now a stack instance of the specified StackSet. To learn more about the stack import status, see StackSets status codes.

Import an existing stack into an existing StackSet (console)

Before you begin, identify the stack that you want to import.

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the navigation pane, choose **StackSets**.
3. On the **StackSets** page, choose the StackSet that you want to import a stack into.
4. With the StackSet selected, choose **Add stacks to StackSet** from the **Actions** menu.
5. On the **Set deployment options** page, choose **Import stacks to stack set** and enter the stack ID of the stack you want to import in the **Stacks to import** field. For example, *arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/f449b250-b969-11e0-a185-5081d0136786*.

   (Optional) Choose **Add another stack ID** and enter the stack ID of another stack you want to import. You may add up to 10 stacks per stack import operation.
6. Choose **Next**.
7. On the **Specify overrides** page, review your choices and choose **Next**.
8. On the **Review** page, review your choices and your StackSet's properties. When you are ready to create your StackSet, choose **Submit**.

Results: The imported stack is now a stack instance of the specified StackSet. To learn more about the stack import status, see StackSets status codes.

Import a stack into a StackSet (AWS CLI)

To import an existing stack into a new StackSet

The following `create-stack-set` command creates a StackSet and imports the specified stack. The stack to import is identified by its ARN. Replace the placeholder text with your own information.

```
aws cloudformation create-stack-set \
  --stack-set-name MyStackSet \
  --stack-id arn:aws:cloudformation:us-east-1:123456789012:stack/
StackToImport/466df9e0-0dff-08e3-8e2f-5088487c4896 \
  --administration-role-arn arn:aws:iam::123456789012:role/
AWSCloudFormationStackSetAdministrationRole \
  --execution-role-name AWSCloudFormationStackSetExecutionRole
```

To import an existing stack into an existing StackSet

The following `import-stacks-to-stack-sets` command imports the specified stack into the *MyStackSet* StackSet. The stack to import is identified by its ARN. Replace the placeholder text with your own information.

```
aws cloudformation import-stacks-to-stack-set \
  --stack-set MyStackSet \
  --stack-ids arn:aws:cloudformation:us-east-1:123456789012:stack/StackToImport/
f449b250-b969-11e0-a185-5081d0136786
```

To specify more than one stack, use the following format for the value of the `--stack-ids` option.

```
--stack-ids "arn_1" "arn_2"
```

To clone the imported stack into other Regions and accounts

The following `create-stack-instances` command adds stack instances to your StackSet. Replace the placeholder text with your own information.

```
aws cloudformation create-stack-instances \
  --stack-set-name MyStackSet \
  --accounts '["account_ID_1","account_ID_2"]' \
  --regions '["region_1","region_2"]'
```

Service-managed stack import for CloudFormation StackSets

The CloudFormation stack import operation can import existing stacks into new or existing StackSets, so that you can migrate existing stacks to a StackSet in one operation. StackSets extends the functionality of stacks, so you can create, update, or delete stacks across multiple accounts and Regions with a single operation.

Considerations for service-managed stack imports

- The stack import operation requires a management account or delegated admin account in which you can manage the associated AWS Organizations such as enabling trust access with StackSets.
- The target accounts must be members of the AWS Organizations managed by the management account or delegated admin account.
- Target stack exists in one of the target OUs.
- The target account should be a member of AWS Organizations.
- AWS Organizations access should be in the `ACTIVATED` state for the Organizations.
- Stacks being imported should be present in any of the member accounts, and not the management account.

Topics

- Import a service-managed stack into a new StackSet (console)
- Create and import a service-managed stack into an existing StackSet (console)
- Import a service-managed stack into an existing StackSet (console)
- Importing a service-managed stack into a StackSet (AWS CLI)

Import a service-managed stack into a new StackSet (console)

Import a stack into a new StackSet using the AWS Management Console

To import a new stack into a StackSet, identify a stack that contains the resource you want to import.

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the navigation pane, choose **StackSets**.
3. At the top of the **StackSets** page, choose **Create StackSet**.
4. On the **Choose a template** page, do the following:

   a. For **StackSet permission model** choose **Service-managed permissions**.

   b. For **Prerequisite - Prepare template**, choose **Template is ready**, and choose your template by using one of the following options:

   - For **Amazon S3 URL**, enter your Amazon S3 URL in the **Amazon S3 URL** field.
   - For **Upload a template file**, choose a CloudFormation template on your local computer.

   Accept your settings and choose **Next**.
5. On the **Specify StackSet details** page, do the following:

   a. Enter a StackSet name in the **StackSet name** box.

   b. (Optional) Enter a description in the **StackSet description** section.

   On the **Configure StackSet options** page, review your choices and choose **Next**.
6. On the **Set deployment options** page, do the following:

   a. For **Add stacks to stack set**, choose **Import stacks to stack set**.

   b. For **Stacks to import**, choose your stack import method.

      i. For **Stack ID** enter your stack ID.

      ii. For **Stack URL** enter the Amazon S3 URL.
7. Under **Associate organizational units**, do the following:

   a. Choose **Associate with organization** to use root OU.

   b. Choose **Associate with organizational units (OUs)** to enter parent OU IDs for the stacks to import. For example, if `Stack 1` and `Stack 2` are under 0U1, and `Stack 3` is under 0U2, enter 0U1 and 0U2.

Accept your settings and choose **Next**.

8. Review your settings on the **Review** page and choose **Submit**.

Create and import a service-managed stack into an existing StackSet (console)

To import an existing stack into a new StackSet, identify a stack that contains the resource you want to import.

To create a StackSet and import a stack

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the navigation pane, choose **StackSets**.
3. At the top of the **StackSets** page, choose **Create StackSet**.
4. On the **Choose a template** page, do the following:

   a. For **StackSet permission model** choose **Service-managed permissions**.

   b. For **Prerequisite - Prepare template**, choose **Template is ready**, and choose your template by using one of the following options:

   - For **Amazon S3 URL**, enter your Amazon S3 URL in the **Amazon S3 URL** field.
   - For **Upload a template file**, choose a CloudFormation template on your local computer.

   Accept your settings and choose **Next**.

5. On the **Specify StackSet details** page, do the following:

   a. Enter a StackSet name in the **StackSet name** box.

   b. (Optional) Enter a description in the **StackSet description** section.

   On the **Configure StackSet options** page, review your choices and choose **Next**.

6. On the **Set deployment options** page, do the following:

   - For **Add stacks to stack set**, choose **Deploy new stacks**.

7. For the **Associate organizational units** section, do the following:

a. Choose **Associate with organization** to use root OU.

b. Choose **Associate with organizational units (OUs)** to enter parent OU IDs for the stacks to import. For example, if `Stack 1` and `Stack 2` are under 0U1, and `Stack 3` is under 0U2, enter 0U1 and 0U2.

8. For **Specify regions** and **Deployment options**, review your choices.

   Accept your settings and choose **Next**.

9. Review your settings on the **Review** page and choose **Submit**.

Import a service-managed stack into an existing StackSet (console)

Choose your StackSet and identify the stack you want to import.

To import a stack to an existing StackSet

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the navigation pane, choose **StackSets**.
3. Choose the StackSet you want to import a stack to, and then choose **Add stacks to StackSet** from the **Actions** drop-down.
4. On the **Set deployment options** page, do the following:

   a. For **Add stacks to stack set**, choose **Import stacks to stack set**.

   b. Under **Stacks to import**, do the following

      i. For **Stack ID**, enter your stack ID.

      ii. For **Stack URL**, enter the Amazon S3 URL.

   c. Under **Associate organizational units**, do the following:

      i. Choose **Associate with organization** to use root OU.

      ii. Choose **Associate with organizational units (OUs)** to enter parent OU IDs for the stacks to import. For example, if `Stack 1` and `Stack 2` are under 0U1, and `Stack 3` is under 0U2, enter 0U1 and 0U2.

   Accept your settings and choose **Next**.

5. Review the **Specify overrides** page and choose **Next**.
6. Confirm and review the **Review** page and choose **Submit**.

Importing a service-managed stack into a StackSet (AWS CLI)

Once a StackSet is created, you can import your stacks by passing the stack ID's of the stacks being imported. You may also pass the OU ID list to which you want to map it to.

CloudFormation will import user provided stacks within those OUs and use those OUs as deployment targets for the StackSet. Stack IDs presented in the input will map to the nearest OU in OU ID list input internally. If a stack doesn't belong to an existing OU ID in the input list, then the AWS CLI will return the `StackNotFoundException` error.

The `import-stacks-to-stack-set` operation creates stack instances for the stacks in the OU ID input. The following AWS CLI examples use the `import-stacks-to-stack-set` operation to import a stack into a StackSet.

- To use the `import-stacks-to-stack-sets` operation, specify `stack-ids` or `stack-ids-url` you want to import to your stack set.

```
aws cloudformation import-stacks-to-stack-set \
  --stack-set-name ServiceMangedStackSet \
  --stack-ids "arn:123456789012:us-east-1:Stack1" \
  --organizational-unit-ids ou-examplerootid111-exampleouid111
```

```
aws cloudformation import-stacks-to-stack-set \
  --stack-set-name ServiceMangedStackSet \
  --stack-ids-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/file-
name.json \
  --organizational-unit-ids ou-examplerootid111-exampleouid111
```

> **Note**
>
> The `import-stacks-to-stack-sets` operation, requires you to specify at least one organizational unit ID (OU ID) so that it can associate the stack being imported to that particular OU. This operation doesn't create stack instances for other member accounts

in the associated OUs. To update member accounts for the associated OUs, use `create-stack-instances` or `update-stack-instances`.

`create-stack-set` creates stack instances for all the accounts under the OUs with a user provided template, either from direct upload or Amazon S3. The following AWS CLI examples use the `create-stack-set` operation to import a stack into a new StackSet.

- To use the `create-stack-set` operation, specify your StackSet name and import a stack to a newly created StackSet.

```
aws cloudformation create-stack-set \
  --template-url https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/file-
name.json \
  --permission-model SERVICE_MANAGED \
  --auto-deployment Enabled=true
```

Revert stack imports into CloudFormation StackSets

If you have unwanted changes in your stack instance, you can revert the stack instance import.

When you revert a stack instance import, CloudFormation deletes the stack instance from the StackSet but retains the stack's resources.

To revert a stack import operation, complete the following procedure.

1. Specify a `DeletionPolicy` attribute of `Retain` for each resource that you want to keep after the stack instance is deleted. For more information, see Reverting an import operation.
2. Delete stack instances from your StackSet. For more information, see Delete stacks from CloudFormation StackSets.
3. Delete your StackSet. For more information, see Delete CloudFormation StackSets.

Best practices for using CloudFormation StackSets

This section describes the best practices for defining a StackSet template, creating or adding stacks to a StackSet, or updating a StackSet.

If you are new to CloudFormation, review the CloudFormation best practices topic for more recommendations that can help you use CloudFormation more effectively and securely.

Topics

- Defining the template
- Creating or adding stacks to the StackSet
- Updating stacks in a StackSet

Defining the template

- Define the template that you want to standardize in multiple accounts, within multiple regions.
- As you create the template, be sure that global resources (such as IAM roles and Amazon S3 buckets) do not have naming conflicts when they are created in more than one region in the same account.
- A StackSet has a single template and parameter set. The same stack is created in all accounts that are associated with a StackSet. As you author your templates, make them granular enough to allow you a good balance of control and standardization.
- We recommend that you store your template in an Amazon S3 bucket.

Creating or adding stacks to the StackSet

- Verify that adding stack instances to your initial StackSet works before you add larger numbers of stack instances to your StackSet.
- Choose the deployment (rollout) options that work for your use case.
  - For a more conservative deployment, set **Maximum Concurrent Accounts** to 1, and **Failure Tolerance** to 0. Set your lowest-impact region to be first in the **Region Order** list. Start with one region.
  - For a faster deployment, increase the values of **Maximum Concurrent Accounts** and **Failure Tolerance** as needed.
- Operations on StackSets depend on how many stack instances are involved, and can take significant time.

Updating stacks in a StackSet

- By default, updating a StackSet updates all stack instances. If you have 20 accounts each in two regions, you will have 40 stack instances, and all will be updated when you update the StackSet.

  For StackSets with a large number of stack instances, we recommend that to test the updated version of a template, you selectively update the stack instances in a few test accounts before updating all stack instances.
- To get more granular control over updating individual stacks within your StackSet, plan to create multiple StackSets.
- Updating a StackSet that contains a large number of stacks can take significant time. In this release, only one operation is permitted at a time on a StackSet. Plan your updates so you are not blocked from performing other operations on the StackSet.

CloudFormation StackSets sample templates

This section includes links to some sample CloudFormation templates that can help you use CloudFormation StackSets in your enterprise. Templates listed in this section enable AWS CloudTrail or AWS Config and rules within it.

> **Important**
>
> As a security best practice when allowing AWS Config access to an Amazon S3 bucket, we strongly recommend that you restrict access in the bucket policy with the `AWS:SourceAccount` condition. New templates are updated to have `AWS:SourceAccount`. If your existing bucket policy does not follow this security best practice, we strongly recommend you edit that bucket policy to include this protection. This makes sure that AWS Config is granted access on behalf of expected users only.

Description	S3 link
Enable AWS CloudTrail	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml

Description	S3 link
Enable AWS Config	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSConfig.yml
Enable AWS Config with central logging	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSConfigForOrganizations.yml
Enable Amazon Data Lifecycle Manager default policies across an AWS organization or across specific AWS accounts	• Default policy for EBS snapshots — https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/DataLifecycleManagerEBSSnapshotDefaultPolicy.yaml • Default policy for EBS-backed AMIs — https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/DataLifecycleManagerAMIDefaultPolicy.yaml https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEncryptedVolumes.yml
Configure an AWS Config rule to determine if CloudTrail is enabled	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleCloudtrailEnabled.yml
Configure an AWS Config rule to determine if root MFA is enabled	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleRootAccountMFAEnabled.yml
Configure an AWS Config rule to determine if EIPs are attached	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEipAttached.yml

Description	S3 link
Configure an AWS Config rule to determine if EBS volumes are encrypted	https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/ConfigRuleEncryptedVolumes.yml

Troubleshooting CloudFormation StackSets

This topic contains some common StackSets issues, and suggested solutions for those issues.

Topics

- Common reasons for stack operation failure
- Retrying failed stack creation or update operations
- Stack instance deletion fails
- Stack import operation fails
- Stack instance failure count for StackSets operations

Common reasons for stack operation failure

Problem: A stack operation failed, and the stack instance status is `OUTDATED`.

Cause: There can be several common causes for stack operation failure.

- Insufficient permissions in a target account for creating resources that are specified in your template.
- The CloudFormation template might have errors. Validate the template in CloudFormation and fix errors before trying to create your StackSet.
- The template could be trying to create global resources that must be unique but aren't, such as S3 buckets.
- A specified target account number doesn't exist. Check the target account numbers that you specified on the **Set deployment options** page of the wizard.
- The administrator account does not have a trust relationship with the target account.
- The maximum number of a resource that is specified in your template already exists in your target account. For example, you might have reached the limit of allowed IAM roles in a target account, but the template creates more IAM roles.

- You have reached the maximum number of stacks that are allowed in a StackSet. For the maximum number of stacks per StackSet, see Understand CloudFormation quotas.

Solution: For more information about the permissions required of target and administrator accounts before you can create StackSets, see Give all users of the administrator account permissions to manage stacks in all target accounts.

Retrying failed stack creation or update operations

Problem: A stack creation or update failed, and the stack instance status is `OUTDATED`. To troubleshoot why a stack creation or update failed, open the CloudFormation console, and view the events for the stack, which will have a status of `DELETED` (for failed create operations) or `FAILED` (for failed update operations). Browse the stack events, and find the **Status reason** column. The value of **Status reason** explains why the stack operation failed.

After you have fixed the underlying cause of the stack creation failure, and you are ready to retry stack creation, perform the following steps.

Solution: Perform the following steps to retry your stack operation.

1. In the console, select the StackSet that contains the stack on which the operation failed.
2. In the **Actions** menu, choose **Edit StackSet details** to retry creating or updating stacks.
3. On the **Specify template** page, to use the same CloudFormation template, keep the default option, **Use current template**. If your stack operation failed because the template required changes, and you want to upload a revised template, choose **Upload a template to Amazon S3** instead, and then choose **Browse** to select your updated template. When you are finished uploading your revised template, choose **Next**.
4. On the **Specify stack details** page, if you are not changing any parameters that are specific to your template, choose **Next**.
5. On the **Set deployment options** page, change defaults for **Maximum concurrent accounts** and **Failure tolerance**, if desired. For more information about these settings, see StackSet operation options.
6. On the **Review** page, review your selections, and fill the checkbox to acknowledge required IAM capabilities. Choose **Submit**.
7. If your stack is not successfully updated, repeat this procedure, after you've resolved any underlying issues that are preventing stack creation.

Stack instance deletion fails

Problem: A stack deletion has failed.

Cause: Stack deletion will fail for any stacks on which termination protection has been enabled.

Solution: Determine if termination protection has been enabled for the stack. If it has, disable termination protection and then perform the stack instance deletion again.

Stack import operation fails

Problem: A stack import operation fails to import existing stacks into new or existing StackSets. The stack instance is in an `INOPERABLE` status.

Solution: Revert the stack import operation, by completing the following tasks.

1. Use **Delete Stacks from StackSets** option and enable **RetainStacks** during configuration, then proceed to delete stack instances from your StackSet. For more information, For more information, see Delete stacks from CloudFormation StackSets.
2. You will see the stack instances of the StackSet are updated to remove the `INOPERABLE` stack instance.
3. Fix the stack instances according to the import failure error and retry the stack import operation.

Stack instance failure count for StackSets operations

The stack instance failure count alerts you if stack instances fail to provision or update. These stack instances didn't deploy because of one or more of the following reasons:

- Existing resource(s) with a similar configuration
- Missing dependencies, such as AWS Identity and Access Management (IAM) roles
- Other conflicting factors

If you want to deploy with maximum concurrency, max concurrency count is one more than the failure tolerance count, at most. For example, if the failure tolerance count is 9, then max concurrency count can’t be more than 10. This will cause the operation to return `SUCCEEDED` even if some stack instances fail to update. The new stack instance failure count enables you to determine if the operation only conditionally succeeded because the failure tolerance count is set to allow all failures.

You can use the AWS Management Console, AWS SDK, or AWS CLI to get the failure count and filter stack instances to determine which instances need to be redeployed.

Using the console

To view the number of failed stack instances:

1. Open the CloudFormation console and choose **StackSets**.
2. Choose your StackSet, and then choose the **Operations** tab.
3. Choose a status in the **Status** column to view the status details. You will find the number of failed stack instances for a particular operation in the status details.

To view the account, region, and status of stack instances for the operation:

1. In the status details, choose the failed stack instances count. *Example:* **Stack instances: `<number of failed stack instances>`**.
2. Expand the side panel by choosing the panel header. The results in the side panel are the statuses of the stack instances after the completion of the selected operation.

To view the current stack instance details for an operation:

1. Choose the **Stack Instances** tab.
2. Filter by **Last operation ID**. The results are the current statuses and status reasons from the last operation to modify the instance. You can use this filter in combination with **AWS account**, **AWS region**, **Detailed Status**, and **Drift status** to further refine your search results.

Using the AWS CLI

To get the number of failed stack instances, call `describe-stack-set-operation` or `list-stack-set-operations` and see `StatusDetails`.

```
aws cloudformation describe-stack-set-operation --stack-set-name ss1 \
    --operation-id 5550e62f-c822-4331-88fa-21c1d7bafc60
```

```
{
    "StackSetOperation": {
        "OperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60",
```

```
        "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
        "Action": "CREATE",
        "Status": "SUCCEEDED",
        "OperationPreferences": {
            "RegionOrder": [],
            "FailureToleranceCount": 10,
            "MaxConcurrentCount": 10
        },
        "AdministrationRoleARN": "arn:aws:iam::123456789012:role/
AWSCloudFormationStackSetAdministrationRole",
        "ExecutionRoleName": "AWSCloudFormationStackSetExecutionRole",
        "CreationTimestamp": "2022-10-26T17:18:53.947000+00:00",
        "EndTimestamp": "2022-10-26T17:19:35.304000+00:00",
        "StatusDetails": {
            "FailedStackInstancesCount": 3
        }
    }
}
```

```
aws cloudformation list-stack-set-operations --stack-set-name ss1
```

```
{
    "Summaries": [
        {
            "OperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60",
            "Action": "CREATE",
            "Status": "SUCCEEDED",
            "CreationTimestamp": "2022-10-26T17:18:53.947000+00:00",
            "EndTimestamp": "2022-10-26T17:19:35.304000+00:00",
            "StatusDetails": {
                "FailedStackInstancesCount": 3
            },
            "OperationPreferences": {
                "RegionOrder": [],
                "FailureToleranceCount": 10,
                "MaxConcurrentCount": 10
            }
        }
    ]
}
```

To get a historical overview for a particular operation, use `list-stack-set-operation-results` to view the status and status reason for each stack instance after the operation completed. See the following example to find the `Status` and `StatusReason`:

```
aws cloudformation list-stack-set-operation-results --stack-set-name ss1 \
  --operation-id 5550e62f-c822-4331-88fa-21c1d7bafc60 --
filters Name=OPERATION_RESULT_STATUS,Values=FAILED
```

```
{
    "Summaries": [
        {
            "Account": "123456789012",
            "Region": "us-west-2",
            "Status": "FAILED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "AccountGateResult": {
                "Status": "SKIPPED",
                "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'."
            },
            "OrganizationalUnitId": ""
        },
        {
            "Account": "123456789012",
            "Region": "us-west-1",
            "Status": "FAILED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "AccountGateResult": {
                "Status": "SKIPPED",
                "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'."
            },
            "OrganizationalUnitId": ""
        },
        {
            "Account": "123456789012",
            "Region": "us-east-1",
```

```
            "Status": "FAILED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "AccountGateResult": {
                "Status": "SKIPPED",
                "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'."
            },
            "OrganizationalUnitId": ""
        }
    ]
}
```

Use `list-stack-instances` with the `DETAILED_STATUS` and `LAST_OPERATION_ID` filters to get a list of stack instances that failed in the last operation that tried to deploy the stack instance. See the `--filters` flag in the example with `DETAILED_STATUS` and `LAST_OPERATION_ID`:

```
aws cloudformation list-stack-instances --stack-set-name ss1 \
  --filters Name=DETAILED_STATUS,Values=FAILED Name=LAST_OPERATION_ID,Values=5550e62f-
c822-4331-88fa-21c1d7bafc60
```

```
{
    "Summaries": [
        {
            "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
            "Region": "us-east-1",
            "Account": "123456789012",
            "Status": "OUTDATED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "StackInstanceStatus": {
                "DetailedStatus": "FAILED"
            },
            "OrganizationalUnitId": "",
            "DriftStatus": "NOT_CHECKED",
            "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
        },
        {
            "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
```

```
            "Region": "us-west-1",
            "Account": "123456789012",
            "Status": "OUTDATED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "StackInstanceStatus": {
                "DetailedStatus": "FAILED"
            },
            "OrganizationalUnitId": "",
            "DriftStatus": "NOT_CHECKED",
            "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
        },
        {
            "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
            "Region": "us-west-2",
            "Account": "123456789012",
            "Status": "OUTDATED",
            "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
            "StackInstanceStatus": {
                "DetailedStatus": "FAILED"
            },
            "OrganizationalUnitId": "",
            "DriftStatus": "NOT_CHECKED",
            "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
        }
    ]
}
```

To find the last operation ID to modify a stack instance, use `list-stack-instances` or `describe-stack-instance` to get the `LastOperationId`:

```
aws cloudformation describe-stack-instance --stack-set-name ss1 \
  --stack-instance-account 123456789012 --stack-instance-region us-east-2
```

```
{
    "StackInstance": {
        "StackSetId": "ss1:9101ca57-49fc-4a61-a5a6-4c97b8adb08f",
        "Region": "us-west-2",
        "Account": "123456789012",
        "ParameterOverrides": [],
```

```
        "Status": "OUTDATED",
        "StackInstanceStatus": {
            "DetailedStatus": "FAILED"
        },
        "StatusReason": "Account 123456789012 should have
 'AWSCloudFormationStackSetExecutionRole' role with trust relationship to Role
 'AWSCloudFormationStackSetAdministrationRole'.",
        "OrganizationalUnitId": "",
        "DriftStatus": "NOT_CHECKED",
        "LastOperationId": "5550e62f-c822-4331-88fa-21c1d7bafc60"
    }
}
```

Syncing stacks with source code stored in a Git repository with Git sync

With Git sync, you can manage your CloudFormation stacks with source control. You do this by configuring CloudFormation to monitor a Git repository. The repository is monitored for changes to two files:

- A CloudFormation template file that defines a stack
- A stack deployment file that contains parameters that configure the stack

With Git sync, you can use pull requests and version tracking to configure, deploy, and update your CloudFormation stacks from a centralized location. When you commit changes to the template or the deployment file, CloudFormation automatically updates the stack. If you use pull requests, CloudFormation can leave a comment on the pull request explaining what changes will be made to your stack before actually updating it. However, you need to enable this feature first.

Git sync provides a console interface that you can use to link to a repository, generate a stack deployment file, update a CloudFormation template, and submit a pull request to your repository. Git sync also provides a status dashboard that you can use to monitor, edit, and troubleshoot active Git sync stack deployments. Git sync is accessed through the CloudFormation console when you create a stack. You can also access Git sync using AWS CodeConnections. For more information, see Working with sync configurations for linked repositories in the *Developer Tools Console User Guide*.

Git sync supports GitHub, GitHub Enterprise, GitLab, Bitbucket, and GitLab self-managed repositories.

> **Note**
>
> Git sync is available in the following regions: US East (N. Virginia), US East (Ohio), US West (N. California), US West (Oregon), Canada (Central), Asia Pacific (Mumbai), Asia Pacific (Tokyo), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Europe (Ireland), Europe (London), Europe (Paris), Europe (Stockholm), Europe (Frankfurt), Europe (Milan), and South America (São Paulo).

For information about using Git sync with a multi-account strategy, see the following blog post Use CloudFormation Git sync to configure resources in customer accounts.

Topics

- How Git sync works with CloudFormation
- Prerequisites for syncing stacks to a Git repository using Git sync
- Create a stack from repository source code with Git sync
- Enable CloudFormation to post a summary of stack changes in pull requests
- Git sync status dashboard

How Git sync works with CloudFormation

This topic describes how Git sync works and introduces the key concepts required to work with it.

Topics

- How Git sync works
- Comments on pull requests
- Stack deployment file
- CloudFormation template file
- Template definition repository

How Git sync works

To use Git sync, you first must connect a Git provider to CloudFormation using the AWS CodeConnections service. In the procedures in this guide, the connection is created through the AWS CodeConnections console. Alternatively, you can create the connection with the AWS CLI. You can use any of the following Git providers:

- GitHub
- GitHub Enterprise
- GitLab
- Bitbucket
- GitLab self-managed

Next, you create a CloudFormation template that defines your stack and add it to your repository. This template file is monitored. CloudFormation updates the stack automatically when changes are committed to it.

In the CloudFormation console, you create a new stack and choose **Sync from Git** to tell CloudFormation to use Git sync. You'll specify the repository and branch you want CloudFormation to monitor, and specify the CloudFormation template in your repository that defines the stack.

During configuration, you can either provide your own stack deployment file from your repository, or have Git sync generate one for you. The stack deployment file contains parameters and values that configure the resources in your stack. This stack deployment file is monitored. CloudFormation updates the stack automatically when changes are committed to it.

Git sync creates a pull request in your repository to sync your stack with the CloudFormation template file and stack deployment file. If Git sync generates the stack deployment file for you, it's submitted to your repository by Git sync.

You then merge the pull request to your repository so that CloudFormation provisions the stack, configures it with your deployment parameters, and begins monitoring your repository for changes.

From then on, whenever you make changes to your template file or stack deployment file and commit them to your repository, CloudFormation will automatically detect the changes. If your team uses pull requests, your team members can then review and approve the changes before they're deployed. Once the pull request is accepted, CloudFormation deploys your changes.

You can monitor the status of your Git sync configuration for the stack and see a history of commits applied to the stack in the CloudFormation console. The console also provides tools for reconfiguring Git sync and troubleshooting issues.

Comments on pull requests

You can choose to have CloudFormation create a summary of the code changes in pull requests through the AWS CodeConnections service by turning on the **Enable comment on pull request** option in the console. Providing a summary of the changes in pull requests means that team members can easily review and understand the impact of the proposed modifications before merging the pull request. For more information, see Enable CloudFormation to post a summary of stack changes in pull requests.

Stack deployment file

A stack deployment file is a JavaScript Object Notation (JSON) or YAML standard formatted file that contains parameters and values that manage your CloudFormation stack. It is monitored for changes. When changes to the file are committed to the repository, the associated stack is automatically updated.

The stack deployment file contains a key-value pair and two dictionaries:

- `template-file-path`

  This is the full repository path for the CloudFormation template file. The template file declares the resources for the CloudFormation stack associated with this deployment file.

- `parameters`

  The parameters dictionary contains key-value pairs that configure the resources in the stack. A stack deployment file can have up to 50 parameters.

- `tags`

  The tags dictionary contains optional key-value pairs that you can use to identify and categorize resources in the stack. A stack deployment file can have up to 50 tags.

You can provide your own stack deployment file, or have Git sync create one for you and automatically submit a pull request to your repository. You can manage the parameters and tags by editing the stack deployment file and committing changes to the repository.

The following is an example of a Git sync stack deployment file:

```
template-file-path: fargate-srvc/my-stack-template.yaml

parameters:
    image: public.ecr.aws/lts/nginx:latest
    task_size: x-small
    max_capacity: 5
    port: 8080
    env: production
tags:
    cost-center: '123456'
    org: 'AWS'
```

CloudFormation template file

A template file contains a declaration of the AWS resources that make up a CloudFormation stack. With Git sync, the template file is stored in your Git repository and referenced by the stack deployment file. You can manage the stack by editing the template file and committing changes to the repository.

For more information, see Working with CloudFormation templates.

Template definition repository

The template definition repository is the Git repository that is linked to CloudFormation through Git sync. The repository is monitored for changes to the CloudFormation template and stack deployment file. When you commit changes to the file, the associated stack is updated automatically.

> **Important**
>
> When you configure the template definition repository in the Git sync console, select the correct *repository* and *branch* from the Git connection. Git sync only monitors the configured repository and branch for changes to the CloudFormation template and the stack deployment file.

Prerequisites for syncing stacks to a Git repository using Git sync

Before you sync a CloudFormation stack to your Git repository, verify that the following requirements are met.

Topics

- Git repository
- CloudFormation template
- Git sync service role
- IAM permissions for console users

Git repository

You must have a Git repository hosted on one of the following platforms.

- GitHub
- GitHub Enterprise
- GitLab
- Bitbucket
- GitLab self-managed

The repository can be either public or private. You will need to connect this Git repository to CloudFormation though the Connections console.

CloudFormation template

Your Git repository must contain a CloudFormation template file checked into the branch you intend to connect with Git sync. This template will be referenced by the stack deployment file.

Git sync service role

Git sync requires an IAM role. You can choose to have an IAM role created for your stack when you configure Git sync, or you can use an existing role.

> **Note**
>
> An automatically generated IAM role only applies permissions to the stack for which the role is generated. To reuse an automatically generated IAM role, you must edit the role for the new stack.

Required permissions for Git sync service role

The IAM role that you provide for Git sync requires the following permissions.

- `cloudformation:CreateChangeSet`
- `cloudformation:DeleteChangeSet`
- `cloudformation:DescribeChangeSet`
- `cloudformation:DescribeStackEvents`

- `cloudformation:DescribeStacks`
- `cloudformation:ExecuteChangeSet`
- `cloudformation:ListChangeSets`
- `cloudformation:ValidateTemplate`
- `events:PutRule`
- `events:PutTargets`

> **Note**
>
> The preceding required permissions are automatically added to IAM roles that Git sync generates.

The following example IAM role includes the prerequisite permissions for Git sync.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SyncToCloudFormation",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:GetTemplate",
                "cloudformation:ListChangeSets",
                "cloudformation:ListStacks",
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "PolicyForManagedRules",
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:PutTargets"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                "events:ManagedBy":
 ["cloudformation.sync.codeconnections.amazonaws.com"]
                }
            }
        },
        {
            "Sid": "PolicyForDescribingRule",
            "Effect": "Allow",
            "Action": "events:DescribeRule",
            "Resource": "*"
        }
    ]
}
```

Trust policy

You must provide the following trust policy when you create the role to define the trust relationship.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CfnGitSyncTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.sync.codeconnections.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

We recommend that you use the `aws:SourceArn` and `aws:SourceAccount` condition keys to protect yourself against the confused deputy problem. The source account is your account ID and the source ARN is the ARN of the connection in the AWS CodeConnections service that allows CloudFormation to connect to your Git repository.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CfnGitSyncTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudformation.sync.codeconnections.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:codeconnections:us-
east-1:123456789012:connection/EXAMPLE64-8aad-4d5d-8878-dfcab0bc441f"
        }
      }
    }
  ]
}
```

For more information about the confused deputy problem, see Cross-service confused deputy prevention.

IAM permissions for console users

To successfully set up Git sync through the CloudFormation console, end users must also be granted permissions through IAM.

The following codeconnections permissions are required to create and manage the connection to your Git repository.

- codeconnections:CreateRepositoryLink
- codeconnections:CreateSyncConfiguration
- codeconnections:DeleteRepositoryLink
- codeconnections:DeleteSyncConfiguration
- codeconnections:GetRepositoryLink
- codeconnections:GetSyncConfiguration
- codeconnections:ListRepositoryLinks
- codeconnections:ListSyncConfigurations
- codeconnections:ListTagsForResource
- codeconnections:TagResource
- codeconnections:UntagResource
- codeconnections:UpdateRepositoryLink
- codeconnections:UpdateSyncBlocker
- codeconnections:UpdateSyncConfiguration
- codeconnections:UseConnection

Console users must also have the following cloudformation permissions to view and manage stacks during the Git sync setup process.

- cloudformation:CreateChangeSet
- cloudformation:DeleteChangeSet
- cloudformation:DescribeChangeSet
- cloudformation:DescribeStackEvents
- cloudformation:DescribeStacks
- cloudformation:ExecuteChangeSet
- cloudformation:GetTemplate
- cloudformation:ListChangeSets

- `cloudformation:ListStacks`
- `cloudformation:ValidateTemplate`

> **Note**
>
> While change set permissions (`cloudformation:CreateChangeSet`, `cloudformation:DeleteChangeSet`, `cloudformation:DescribeChangeSet`, `cloudformation:ExecuteChangeSet`) might not be strictly required for console-only usage, they're recommended to enable full stack inspection and management capabilities.

The following example IAM policy includes the user permissions needed to set up Git sync through the console.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeConnectionsPermissions",
            "Effect": "Allow",
            "Action": [
                "codeconnections:CreateRepositoryLink",
                "codeconnections:CreateSyncConfiguration",
                "codeconnections:DeleteRepositoryLink",
                "codeconnections:DeleteSyncConfiguration",
                "codeconnections:GetRepositoryLink",
                "codeconnections:GetSyncConfiguration",
                "codeconnections:ListRepositoryLinks",
                "codeconnections:ListSyncConfigurations",
                "codeconnections:ListTagsForResource",
                "codeconnections:TagResource",
                "codeconnections:UntagResource",
                "codeconnections:UpdateRepositoryLink",
                "codeconnections:UpdateSyncBlocker",
                "codeconnections:UpdateSyncConfiguration",
                "codeconnections:UseConnection",
                "codeconnections:CreateForcedTargetSync",
                "codeconnections:CreatePullRequestForResource"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudFormationConsolePermissions",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStacks",
                "cloudformation:ExecuteChangeSet",
                "cloudformation:GetTemplate",
                "cloudformation:ListChangeSets",
                "cloudformation:ListStacks",
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        }
    ]
}
```

Note

When creating an IAM policy that includes the permissions `codeconnections:CreateForcedTargetSync` and `codeconnections:CreatePullRequestForResource`, you might see a warning in the IAM console stating that these actions do not exist. This warning can be ignored, and the policy will still be created successfully. These permissions are required for certain Git sync operations despite not being recognized by the IAM console.

Create a stack from repository source code with Git sync

This topic explains how to create a CloudFormation stack that syncs to a Git repository with Git sync.

Important

Before you continue, complete all prerequisites in the previous section.

Create a stack from repository source code

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region to create the stack in.
3. On the **Stacks** page, choose **Create stack**, and then choose **With new resources (standard)**.
4. On the **Create stack** page, do the following:
   a. For **Prerequisite - Prepare template**, keep **Choose an existing template** selected.
   b. For **Specify template**, choose **Sync from Git**, and then choose **Next**.
5. On the **Specify stack details** page, for **Stack name**, type a name for your stack. Stack names can include letters (A-Z and a-z), numbers (0-9), and dashes (-).
6. For **Stack deployment file**, **Deployment file creation**:
   - If you *haven't* created a stack deployment file and added it to your repository, choose **Create the file using the following parameters and place it in my repository.**
   - If you have a stack deployment file in your repository, choose **I am providing my own file in my repository.**
7. For **Template definition repository**, choose **Choose a linked Git repository** to choose a Git repository that's already linked to CloudFormation, or **Link a Git repository** to link a new one. If you choose **Link a Git repository**, do the following:
   a. For **Select repository provider**, choose one of the following:
      - **GitHub**
      - **GitHub Enterprise Server**
      - **GitLab**
      - **Bitbucket**
      - **GitLab self-managed**

b. For **Connection**, choose a connection from the list. If no options appear in the **Connection** list, choose **add a new connection** to go to the Connections console and create a connection to your repository.

8. In the **Repository** list, select the Git repository that contains your stack template file.
9. In the **Branch** list, select the branch you'd like Git sync to monitor.

> **Note**
>
> Git sync only monitors the selected branch for changes to the CloudFormation template and stack deployment files. Any changes you'd like to apply to your stack must be committed to this branch.

10. For the **Deployment file path**, specify the full path including the stack deployment file name from the root of your repository branch.

    If CloudFormation is generating the file for you, this is where the file will be committed in your repository. If you are providing the file, this is the location of the file in your repository.
11. Add an **IAM role**. The IAM role includes permissions that are required for CloudFormation to sync the stack from your Git repository. You can choose **New IAM role** to generate a new role, or choose **Existing IAM role** to select an existing role from your AWS account. If you choose to generate a new role, the required permissions are included in the role.

> **Note**
>
> If role manager is enabled in your account, CloudFormation attaches a service role for you, and the **Existing IAM role** selection described here is replaced by a **Customize** option. To use a different role, choose **Customize**. For more information, see IAM role creation in the *IAM User Guide*.

12. Enable or turn off comments on pull request:

    - To have CloudFormation post change set information in pull requests for stack updates, keep the **Enable comment on pull request** toggle switched on.
    - If you switch this toggle off, CloudFormation won't describe the differences between the current stack configuration and the proposed changes in pull requests when the repo files are updated.

13. For the **Template file path**, specify the full path from the root of your repository for the stack template file.
14. (Optional) To specify the stack parameters, choose **Add parameter**, provide a key and value for each parameter, and then choose **Next**. For more information, see Stack deployment file.

    For example, to specify a **`port=8080`** parameter in your stack deployment file, do the following:

    a. Choose **Add parameter**.

    b. For **Key**, enter **`port`**.

    c. For **Value**, enter **`8080`**.
15. (Optional) To specify stack tags, choose **Add new tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see Stack deployment file.
16. Choose **Next** to continue to **Configure stack options**. For information about configuring stack options, see Configure stack options.

    When you've completed your stack configuration, choose **Next** to continue.
17. Review your stack settings and confirm the following:

    - The stack template is configured correctly and set to **Sync from Git**.
    - The deployment file is configured correctly.
    - The template definition repository is configured correctly, in particular, that the correct **Repository** and **Branch name** are selected.
    - The preview of the deployment file is correct and contains the expected parameters and values.
18. Choose **Submit** to create the stack.

    After you choose **Submit**, a pull request is automatically created in your Git repository. You must merge this pull request into your Git repository to create your stack. After the stack is created, CloudFormation monitors your Git repository for changes.

Update your stack from your Git repository

To update the stack, make changes directly to your template file or stack deployment file in your Git repository. After you commit your changes to the monitored branch, CloudFormation automatically updates the stack. If you use pull requests, a pull request is automatically created

in your Git repository before the stack is updated. You must merge this pull request into your Git repository to update your stack.

In the CloudFormation console, you can select the stack and choose the **Git sync** tab to view information about the status of the stack and sync events. For more information, see Git sync status dashboard.

Enable CloudFormation to post a summary of stack changes in pull requests

This topic shows you how to enable CloudFormation to post a summary of stack changes in pull requests in your Git repository.

By enabling the comments on pull requests feature, you allow CloudFormation to post a comment that describes the differences between the current stack configuration and the proposed changes when the repo files are updated. This comment provides a summary of the resources that will be added, modified, or deleted, allowing you to perform a thorough code review before merging the pull request.

To enable comments on pull requests for a new stack (console)

When you create the stack, on the **Specify stack details** page, under **Template definition repository**, make sure the **Enable comment on pull request** toggle is switched on. This is the default setting for new stacks.

To enable comments on pull requests for an existing stack (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region that you created your stack in.
3. On the **Stacks** page, choose the running stack that you want to update.
4. Choose the **Git sync** tab, and then choose **Edit**.
5. On the **Edit Git sync settings** page, under **Template definition repository**, switch the **Enable comment on pull request** toggle on.
6. Choose **Update configuration**.

Git sync status dashboard

To view the status of a AWS CloudFormation Git sync deployment, select the stack in the CloudFormation console and choose the **Git sync** tab.

The Git sync tab is divided into two panels, **Git sync status** and **Latest sync events**.

Git sync status

The top panel provides the following information about the Git sync configuration for the stack.

Repository

A link to the repository that is connected to Git sync

Repository provider

The name of the repository provider

Branch

The name of the branch that Git sync is monitoring

Deployment file path

The full path to the stack deployment file for this stack

Repository sync status

The status of the most recent sync operation

Repository sync status message

The message of the most recent sync operation

Git sync status

The status of Git sync for this stack

Provisioning status

The status of the provisioning operation

In the upper-right of the panel, use the following buttons to modify or update Git sync:

- **Edit** - Edit the Git sync configuration.

- **Retry latest commit** - Update the stack according to the most recent commit to the repository.
- **Disconnect** - Disconnect Git sync from the stack.
- **Refresh** - Refresh the Git sync status panel.

Latest sync events

The **Latest sync events** panel displays a table of commits that have been applied to the stack.

You can sort the table using the arrows in the header of each column. The table can be sorted in ascending or descending order according to the following:

- **Date**
- **Commit ID**
- **Event**
- **Date**
- **Event Type**

Supported stack states

Git sync can only be configured for a stack if the stack is in one of the following supported states:

- `CREATE_COMPLETE`
- `UPDATE_COMPLETE`
- `UPDATE_ROLLBACK_COMPLETE`
- `IMPORT_COMPLETE`
- `IMPORT_ROLLBACK_COMPLETE`

The following table contains a complete list of stack status codes with descriptions:

Stack status and optional detailed status	Description
`CREATE_COMPLETE`	Successful creation of one or more stacks.
`CREATE_IN_PROGRESS`	Ongoing creation of one or more stacks.

Stack status and optional detailed status	Description
`CREATE_FAILED`	Unsuccessful creation of one or more stacks. View the stack events to see any associated error messages. Possible reasons for a failed creation include insufficient permissions to work with all resources in the stack, parameter values rejected by an AWS service, or a timeout during resource creation.
`DELETE_COMPLETE`	Successful deletion of one or more stacks. Deleted stacks are retained and viewable for 90 days.
`DELETE_FAILED`	Unsuccessful deletion of one or more stacks. Because the delete failed, you might have some resources that are still running; however, you can't work with or update the stack. Delete the stack again or view the stack events to see any associated error messages.
`DELETE_IN_PROGRESS`	Ongoing removal of one or more stacks.
`REVIEW_IN_PROGRESS`	Ongoing creation of one or more stacks with an expected `StackId` but without any templates or resources. **Important** A stack with this status code counts against the maximum possible number of stacks.
`ROLLBACK_COMPLETE`	Successful removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation. The stack returns to the previous working state. Any resources that were created during the create stack operation are deleted. This status exists only after a failed stack creation. It signifies that all operations from the partially created stack have been appropriately cleaned up. When in this state, only a delete operation can be performed.

Stack status and optional detailed status	Description
ROLLBACK_FAILED	Unsuccessful removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation. Delete the stack or view the stack events to see any associate d error messages.
ROLLBACK_IN_PROGRESS	Ongoing removal of one or more stacks after a failed stack creation or after an explicitly canceled stack creation.
UPDATE_COMPLETE	Successful update of one or more stacks.
UPDATE_COMPLETE_CL EANUP_IN_PROGRESS	Ongoing removal of old resources for one or more stacks after a successful stack update. For stack updates that require resources to be replaced, CloudFormation creates the new resources first and then deletes the old resources to help reduce any interruptions with your stack. In this state, the stack has been updated and is usable, but CloudFormation is still deleting the old resources.
UPDATE_FAILED	Unsuccessful update of one or more stacks. View the stack events to see any associated error messages.
UPDATE_IN_PROGRESS	Ongoing update of one or more stacks.
UPDATE_ROLLBACK_CO MPLETE	Successful return of one or more stacks to a previous working state after a failed stack update.
UPDATE_ROLLBACK_CO MPLETE_CLEANUP_IN_ PROGRESS	Ongoing removal of new resources for one or more stacks after a failed stack update. In this state, the stack has been rolled back to its previous working state and is usable, but CloudFormation is still deleting any new resources it created during the stack update.

Stack status and optional detailed status	Description
UPDATE_ROLLBACK_FAILED	Unsuccessful return of one or more stacks to a previous working state after a failed stack update. When in this state, you can delete the stack or continue rollback. You might need to fix errors before your stack can return to a working state. Or, you can contact Support to restore the stack to a usable state.
UPDATE_ROLLBACK_IN_PROGRESS	Ongoing return of one or more stacks to the previous working state after failed stack update.
IMPORT_IN_PROGRESS	The import operation is currently in progress.
IMPORT_COMPLETE	The import operation successfully completed for all resources in the stack that support `resource import`.
IMPORT_ROLLBACK_IN_PROGRESS	Import will roll back to the previous template configuration.
IMPORT_ROLLBACK_FAILED	The import rollback operation failed for at least one resource in the stack. Results will be available for the resources CloudFormation successfully imported.
IMPORT_ROLLBACK_COMPLETE	Import successfully rolled back to the previous template configuration.

Managing extensions with the CloudFormation registry

The CloudFormation registry serves as a centralized hub for managing extensions that can be integrated into the CloudFormation templates in your AWS account. Extensions include resource types, modules, and Hooks from AWS and third-party publishers, and your own custom extensions. The registry makes it easier to discover and provision extensions in your CloudFormation templates in the same manner you use AWS-provided resources.

This section describes how to use the CloudFormation registry to activate third-party extensions in your account, including:

- Activating public extensions
- Registering and activating private extensions

Topics

- Related documentation
- CloudFormation registry concepts
- View the available and activated extensions in the CloudFormation registry
- Use third-party public extensions from the CloudFormation registry
- Use third-party private extensions that have been shared with you
- Edit configuration data for extensions in your account
- Record resource types in AWS Config

Related documentation

If you are a developer interested in creating your own extensions, see the following documentation:

- Developing modules using the CloudFormation CLI in the *CloudFormation Command Line Interface User Guide*
- Creating resource types using the CloudFormation CLI in the *CloudFormation Command Line Interface User Guide*
- Developing custom Hooks using the CloudFormation CLI in the *CloudFormation Hooks User Guide*

Additionally, all provisionable AWS resource types available in the CloudFormation registry can be used with the AWS Cloud Control API, with their attributes and properties defined in a standard JSON schema. For more information, see the Cloud Control API User Guide. When using Cloud Control API to perform CRUDL (Create, Read, Update, Delete, List) operations on AWS resources, you can only do so on AWS resources within your own AWS account.

CloudFormation registry concepts

This topic explains key concepts to help you understand and begin using the CloudFormation registry.

Extension types

The CloudFormation registry offers the following extension types:

Hooks

Hooks are validation checks that inspect your stacks or specific resources before CloudFormation creates, updates, or deletes them. Hooks can also be invoked during create change set operations. They provide a mechanism for enforcing organizational standards and best practices by validating resource configurations against specific requirements. If a Hook detects any configurations that don't comply with its logic, it can either issue a warning or fail the provisioning process to prevent non-compliant resources from being deployed. For more information, see the CloudFormation Hooks User Guide.

For documentation on how to configure Hooks using the CloudFormation console, see the following sections of the *CloudFormation Hooks User Guide*:

- AWS Control Tower proactive controls as Hooks
- Guard Hooks
- Lambda Hooks

Modules

Modules are reusable resource configurations that can be included across multiple CloudFormation stack templates. They simplify the the creation and maintenance of CloudFormation templates by encapsulating complex or frequently used resource configurations into reusable components. This promotes consistency and standardization across your organization's infrastructure deployments.

Resource types

Resource types allow you to model and automate third-party resources or custom resources that aren't natively supported by CloudFormation. By developing resource types, you can extend CloudFormation's capabilities to provision and manage resources from various third-party services.

Public extensions

Public extensions are CloudFormation extensions that are publicly published in the registry for use by all AWS accounts. These include:

- **AWS extensions** – Extensions published by AWS are always public, and activated by default, so you don't have to take any action before using them in your account. AWS controls the versioning of these extensions, so you are always using the latest available version.
- **Third-party extensions** – These extensions are made available for general use by publishers other than AWS. To use a third-party public extension, you must first activate it in your account and Region.

Activated extensions

The CloudFormation registry in your specific AWS account includes three types of activated extensions:

- **AWS extensions** – All AWS public extensions are automatically activated.
- **Activated third-party** – These are local copies of third-party public extensions that you have explicitly activated for your account and Region. When you activate a third-party public extension, CloudFormation creates a local copy in your account's registry. For more information, see Use third-party public extensions from the CloudFormation registry.
- **Privately registered** – These are private extensions that aren't listed in the public CloudFormation registry. These may be extensions you've created yourself or ones shared with you by your organization or other third parties. To use such a private extension in your account, you must first register it. Registering the extension uploads a copy to the CloudFormation registry in your account and activates it. For more information, see Use third-party private extensions that have been shared with you.

Using privately registered extensions and activated public extensions from third-party publishers in your account is like using them in a sandbox environment. Extensions are managed with version control, so their provisioning behavior is tied to a specific version. As a result, these extensions function in the same way as public extensions, adhering to the same version-specific rules.

> **Note**
>
> Privately registered extensions and activated third-party public extensions may implement event handlers that run during create, read, update, list, and delete operations. Using these extensions in your CloudFormation stacks may incur charges to your account, in addition to any charges for the resources created. For more information, see CloudFormation pricing.

View the available and activated extensions in the CloudFormation registry

To view the available and activated extensions in the CloudFormation registry, you can use the AWS Management Console or the AWS CLI.

To view the available and activated extensions (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. From the navigation pane, under **Registry**, choose what extension category you want to view:
   - **Public extensions** displays the public extensions available in your account.
     1. For **Filter**, **Extension type**, choose your extension type: **Resource types**, **Modules**, or **Hooks**.
     2. For **Filter**, **Publisher**, choose a publisher: **AWS** or **Third party**.
   - **Activated extensions** displays the public and private extensions activated in your account.
     1. Choose your extension type: **Resource types**, **Modules**, or **Hooks**.
     2. Use the **Filter** drop-down menu to further choose the extensions to view:
        - **AWS** – lists extensions published by AWS. Extensions published by AWS are activated by default.

    - **Third-party** – lists any public extensions from publishers other than AWS that you have activated in this account.
    - **Registered** – lists any private extensions you have activated in this account.
  - **Publisher** displays any public extensions that you have published using this account.
4. Search or choose the extension name to view extension details.

To view the available and activated extensions (AWS CLI)

Use the list-types command.

Use third-party public extensions from the CloudFormation registry

To use a third-party public extension in your template, you must first *activate* the extension for the account and Region where you want to use it. Activating an extension makes it usable in stack operations in the account and Region where it's activated.

When you activate a third-party public extension, CloudFormation creates an entry in your account's extension registry for the activated extension as a private extension. This allows you to set any configuration properties the extension includes. Configuration properties define how the extension is configured for a given AWS account and Region.

In addition to setting configuration properties, you can also customize the extension in the following ways:

- Specify the execution role CloudFormation uses to activate the extension, in addition to configure logging for the extension.
- Specify whether the extension is automatically updated when a new minor or patch version becomes available.
- Specify an alias to use rather than the third-party public extension name. This can help avoid naming collisions between third-party extensions.

Topics

- Configure an execution role with IAM permissions and a trust policy for public extension access
- Automatically use new versions of extensions

- Use aliases to refer to extensions
- Commonly used AWS CLI commands for working with public extensions
- Activate a third-party public extension in your account
- Update a public third-party extension in your account
- Deactivate third-party public extensions in your account

Configure an execution role with IAM permissions and a trust policy for public extension access

When you activate a public extension from the CloudFormation registry, you can provide an execution role that gives CloudFormation the necessary permissions to invoke that extension in your AWS account and Region.

The permissions required for the execution role are defined in the handler section of the extension schema. You must create an IAM policy that grants the specific permissions needed by the extension and attach it to the execution role.

In addition to the permissions policy, the execution role must also have a trust policy that allows CloudFormation to assume the role. Follow the guidance at Create a role using custom trust policies in the *IAM User Guide* to create a role with a custom trust policy.

Trust relationship

The following shows example trust policies you can use.

You can optionally restrict the scope of the permission for cross-service confused deputy prevention by using one or more global condition context keys with the `Condition` field. For more information, see Cross-service confused deputy prevention.

- Set the `aws:SourceAccount` value to your account ID.
- Set the `aws:SourceArn` value to your extension's ARN.

Example trust policy 1

The following is an example IAM role trust policy for a resource type extension.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "resources.cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
west-2:123456789012:type/resource/Organization-Service-Resource"
                }
            }
        }
    ]
}
```

Example trust policy 2

The following is an example IAM role trust policy for a Hook extension.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "resources.cloudformation.amazonaws.com",
                    "hooks.cloudformation.amazonaws.com"
                ]
            },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
west-2:123456789012:type/hook/Organization-Service-Hook"
                }
            }
        }
    ]
}
```

Automatically use new versions of extensions

When you activate an extension, you can also specify the extension type to use the latest minor version. Your extension type updates the minor version, whenever the publisher releases a new version on your activated extension.

For example, the next time you perform a stack operation, such as creating or updating a stack, using a template that includes that extension, CloudFormation uses the new minor version.

Updating to a new extension version, either automatically or manually and doesn't affect any extension instances already provisioned in stacks.

CloudFormation treats major version updates of extensions as potentially containing breaking changes, and so requires you to manually update to a new major version of an extension.

Extensions published by AWS are activated by default for all accounts and Regions where they're available, and always use the latest version available in each AWS Region.

> **Important**
>
> Because you control if and when extensions gets updated to the latest version in your account, you could end up with different versions of the same extension deployed in different accounts and Regions.
> This might potentially lead to unexpected results when using the same template, containing that extension, across those accounts and Regions.

Use aliases to refer to extensions

You can't activate more than one extension with a given name in a given AWS account and Region. Because different publishers may offer public extensions with the same extension name, CloudFormation lets you specify an alias for any third-party public extension you activate.

If you specify an alias for the extension, CloudFormation treats the alias as the extension type name within the account and Region. You must use the alias to refer to the extension in your templates, API calls, and CloudFormation console.

Extension aliases must be unique within a given account and Region. You can activate the same public resource multiple times in the same account and Region, using different type name aliases.

> **Important**
>
> While extension aliases are only required to be unique in a given account and Region, we strongly suggest that users *not* assign the same alias to different third-party public extensions across accounts and Regions. Doing so could lead to unexpected results when using a template that contains the extension alias across multiple accounts or Regions.

Commonly used AWS CLI commands for working with public extensions

The commonly used commands for working with public extensions include:

- activate-type to activate a public third-party module or resource type in your account.
- set-type-configuration to specify the configuration data for an extension in your account and to disable and enable Hooks.
- list-types to list the extensions in your account.
- describe-type to return detailed information about a specific extension or specific extension version, including current configuration data.
- set-type-default-version to specify which version of an extension is the default version.
- deactivate-type to deactivate a public third-party module or resource type that was previously activated in your account.

Activate a third-party public extension in your account

The following topic shows you how to activate a third-party public extension in your account, which makes it usable in the account and Region it was activated in.

> **Note**
>
> Before you continue, confirm that you have created the IAM role that you'll use with this extension.

Topics

- Activate a public extension (console)
- Activate a public extension (AWS CLI)

Activate a public extension (console)

Follow the steps in this section to use the console to:

- Activate a third-party public extension
- Specify additional extension configuration data for your account

To activate a public extension for use in your account

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. From the navigation pane, under **Registry**, choose **Public extensions**.
4. Use the **Filter** to choose the extension type, and choose **Third party**. (Extensions published by AWS are activated by default.)
5. Choose the extension, then choose **Activate**.

   If multiple versions of an extension are available, you can use the **Version** menu to choose the version of the extension you want to activate. The default is the most current version.
6. For **Extension name**, you can either keep **Use default** selected, or choose **Override default**, and then enter the extension type alias you want to use with this extension. The alias must

follow the recommended format for the extension type. For more information, see Use aliases to refer to extensions.

7. If the extension you are activating is a Hook or resource type, for **Execution role ARN**, specify the IAM role for CloudFormation to assume when invoking the extension. For more information, see Configure an execution role with IAM permissions and a trust policy for public extension access.
8. For **Logging config**, specify logging configuration information for an extension, if desired. For example:

```
{
    "logRoleArn": "arn:aws:iam::account:role/rolename",
    "logGroupName": "log-group-name"
}
```

Logging configuration information isn't required but it's recommended for debugging purposes. To use logging configuration with Hooks, add the same trust policy as the execution role specified, so that the log role can write logs to your log group.

`logRoleArn` and `logGroupName` key names are case-sensitive.

9. For **Versioning**, **Automatic updates**, choose how to receive updates.

- **On** – Automatically updates to the latest minor version. Major versions are updated manually.
- **Off** – Never automatically update to the latest version. All versions are updated manually.

For more information, see Automatically use new versions of extensions.

If the extension requires additional configuration, you have the option to specify the configuration data now, or after the extension has been activated.

> **Important**
>
> If the extension you are activating is a Hook, this step is required. You must specify `ENABLED` for the `HookInvocationStatus` property. This operation enables the Hook's properties that are defined in the Hook's schema `properties` section. For more

information, see Hook configuration schema syntax reference in the *CloudFormation Hooks User Guide*.

To specify the configuration data

1. For **Configuration**, choose **Configure now**, and then choose **Activate extension**.

   CloudFormation displays the **Configure extension** page. To view the current configuration schema for the extension, make sure **View configuration schema** is activated.
2. In the **Configuration JSON** text box, enter a JSON string that represents the configuration data you want to specify for this extension. The JSON you specify must validate against the extension's configuration schema.
3. Choose **Configure extension**.

If you prefer to configure the extension after activation, you can skip this step and provide the configuration data at a later time.

1. For **Configuration**, choose **Configure later**, and then choose **Activate extension**.
2. After the extension is activated, you can configure it by navigating to the extension from the activated extensions page and providing the configuration data.

Activate a public extension (AWS CLI)

Follow the steps in this section to use the AWS CLI to:

- Activate a third-party public extension
- Specify additional extension configuration data for your account

Activate public Hooks

By activating Hooks in your account, you are authorizing a Hook to use defined permissions from your AWS account. CloudFormation removes non-required permissions before passing your permissions to the Hook. CloudFormation recommends customers or Hook users to review the Hook permissions and be aware of what permissions the Hooks are allowed to before activating Hooks in your account.

To activate a public Hook for use in your account (AWS CLI)

1. Get the ARN for your Hook and save it. You can get the ARN of a Hook using the AWS Management Console or AWS CLI. For more information see View the available and activated extensions in the CloudFormation registry.

```
export HOOK_TYPE_ARN="arn:aws:cloudformation:us-west-2:123456789012:type/hook/
Organization-Service-Hook/"
```

2. Use the set-type-configuration command to specify the configuration data. The JSON you pass for `--configuration` must validate against the Hook's configuration schema. To activate the Hook for all stack operations, you must set the `HookInvocationStatus` property to `ENABLED` in the `HookConfiguration` section.

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL", "Properties":{}}}}" \
  --type-arn $HOOK_TYPE_ARN --region us-west-2
```

   For more information on the `HookConfiguration` configuration options, see Hook configuration schema syntax reference in the *CloudFormation Hooks User Guide*.

Activate public modules and resource types

To activate a public extension for use in your account (AWS CLI)

- Use the activate-type command to activate the extension, and specify whether to auto update the extension whenever a new minor version of the extension is published.

  The example below specifies the public Amazon Resource Name (ARN) of a public extension to activate for this account. In addition, it specifies that CloudFormation updates the extension whenever a new minor version is published.

```
aws cloudformation activate-type \
  --public-type-arn public_extension_ARN \
  --execution-role-arn arn:aws:iam::123456789012:role/my-execution-role \
  --auto-update true --region us-west-2
```

  This command returns an ARN of the activated extension.

```
{
    "Arn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-
Resource-Example"
}
```

Update the version of a public extension (AWS CLI)

Use activate-type to activate the extension again.

Use the `--version-bump` option to specify whether to update the extension to the newest `MAJOR` version or newest `MINOR` version.

```
aws cloudformation activate-type --type RESOURCE \
  --type-name Example::Test::1234567890abcdef0 \
  --type-name-alias Example::Test::Alias \
  --version-bump MAJOR --region us-west-2
```

Update a public third-party extension in your account

After you activate a third-party public extension, you can update most extension details from your account.

To update a public extension in your account (console)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. From the navigation pane, under **Registry**, choose **Activated extensions**.
4. Find the extension you want to update and select it. For more information, see View the available and activated extensions in the CloudFormation registry.
5. From the **Actions** menu, choose **Edit**, and then the appropriate editing option:

   - To update the configuration schema, see Edit configuration data for extensions in your account.
   - To activate or deactivate automatic updates:

     1. Choose **Edit automatic updates**.

2. Choose **On** or **Off**, and then choose **Save**. For more information, see Automatically use new versions of extensions.

- To update the execution role:
  1. Choose **Edit execution role**.
  2. Specify the ARN of the IAM role you want CloudFormation to use when invoking this extension, and then choose **Save**. For more information, see Configure an execution role with IAM permissions and a trust policy for public extension access.
- To update the logging configuration:
  1. Choose **Edit logging config**.
  2. Edit the logging configuration JSON, and then choose **Save**.

Deactivate third-party public extensions in your account

When you no longer need an activated third-party public extension, use the following procedures to deactivate it in your account.

Topics

- Deactivate a public extension in your account (console)
- Deactivate a public extension in your account (AWS CLI)
- Disable a Hook in your account (AWS CLI)

Deactivate a public extension in your account (console)

To deactivate a public extension in your account

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. From the navigation pane, under **Registry**, choose **Activated extensions**.
4. Find the extension you want to deactivate and select it. For more information, see View the available and activated extensions in the CloudFormation registry.
5. From the **Actions** menu, choose **Deactivate**.
6. Choose **Deactivate**.

Deactivate a public extension in your account (AWS CLI)

Use the following deactivate-type command.

```
aws cloudformation deactivate-type --type MODULE \
  --type-name Example::Test::Type::MODULE \
  --region us-west-2
```

Disable a Hook in your account (AWS CLI)

Disabling a Hook prevents the Hook from running in your AWS account without removing it.

Use the set-type-configuration command and specify `HookInvocationStatus` as `DISABLED` to disable a Hook.

The following example specifies the AWS Region and the Amazon Resource Name (ARN) of the Hook that's being disabled.

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "DISABLED", "FailureMode": "FAIL", "Properties":{}}}}" \
  --type-arn "arn:aws:cloudformation:us-west-2:123456789012:type/hook/MyTestHook" --
region us-west-2
```

For more information, see Disable and enable CloudFormation Hooks in the *CloudFormation Hooks User Guide*.

Use third-party private extensions that have been shared with you

To use third-party private extensions that have been shared with you, you must first *register* them with CloudFormation, in the accounts and Regions where you want to use them. Registering the extension uploads a copy of it to the CloudFormation registry in your account, and activates it. Once you're registered a private extension, it will appear in the CloudFormation registry for that AWS account and Region, and you can use it in your stack templates.

Topics

- IAM permissions for registering a third-party private extension
- Commonly used AWS CLI commands for working with private extensions

- Register a third-party private extension in your account
- Remove third-party private extensions from your account

IAM permissions for registering a third-party private extension

As part of registering a private extension, you might specify an Amazon S3 bucket that contains the extension project package. This package contains any source files necessary for the extension you want to register. The user registering the extension must be able to access the project package in that Amazon S3 bucket. To do so, the user must have GetObject permissions for the extension package.

This is true whether you're either using the register-type command of the AWS CLI, or the submit command of the CloudFormation CLI.

For more information, see Actions, Resources, and Condition Keys for Amazon S3 in the *Service Authorization Reference*.

Commonly used AWS CLI commands for working with private extensions

The commonly used commands for working with private extensions include:

- register-type to register a private extension in your account.
- describe-type-registration to return the current status of a registration request.
- list-types to list the extensions in your account.
- describe-type to return detailed information about a specific extension or specific extension version, including current configuration data.
- set-type-configuration to specify the configuration data for an extension in your account and to disable and enable Hooks.
- set-type-default-version to specify which version of an extension is the default version.
- deregister-type to remove a private extension or extension version from your account.

Register a third-party private extension in your account

This topic covers the steps to register a third-party private extension that's shared with you so it's available for use in your account.

> **Note**
>
> Before you continue, confirm that you have the required IAM permissions to register a private extension.

To register a private extension that's shared with you (AWS CLI)

1. Locate the Amazon S3 bucket that contains the project package for the private extension you want to register in your account.
2. Use the register-type command to register the private extension in your account.

   For example, the following command registers the `My::Resource::Example` resource type in the specified AWS account.

   ```
   aws cloudformation register-type --type RESOURCE \
     --type-name My::Resource::Example \
     --schema-handler-package [s3 object path] --region us-west-2
   ```

   `RegisterType` is an asynchronous operation, and returns a registration token you can use to track the progress of your registration request.

   ```
   {
       "RegistrationToken": "f5525280-104e-4d35-bef5-8f1fexample"
   }
   ```

   If your extension calls AWS APIs as part of its functionality, you must create an IAM execution role that includes the necessary permissions to call those AWS APIs, and provision that execution role in your account. You can then specify this execution role using the `--execution-role-arn` option. CloudFormation then assumes that execution role to provide your resource type with the appropriate credentials.

   ```
   --execution-role-arn arn:aws:iam::123456789012:role/MyIAMRole
   ```

3. (Optional) Use the registration token with the describe-type-registration command to track the progress of your registration request.

   When CloudFormation completes the registration request, it sets the progress status of the request to `COMPLETE`.

The following example uses the registration token returned by the `describe-type-registration` command above to return registration status information.

```
aws cloudformation describe-type-registration \
  --registration-token f5525280-104e-4d35-bef5-8f1fexample \
  --region us-west-2
```

The command returns the following output.

```
{
    "ProgressStatus": "COMPLETE",
    "TypeArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/My-
Resource-Example",
    "Description": "Deployment is currently in DEPLOY_STAGE of status COMPLETED; ",
    "TypeVersionArn": "arn:aws:cloudformation:us-west-2:123456789012:type/resource/
My-Resource-Example/00000001"
}
```

Important

If the extension you are registering is a Hook, this next step is required. You must specify `ENABLED` for the `HookInvocationStatus` property. This operation enables the Hook's properties that are defined in the Hook's schema `properties` section. For more information, see Hook configuration schema syntax reference in the *CloudFormation Hooks User Guide*.

To specify the configuration data for a Hook (AWS CLI)

1. Get the ARN for your Hook and save it. You can get the ARN of a Hook using the AWS Management Console or AWS CLI. For more information see View the available and activated extensions in the CloudFormation registry.

   ```
   export HOOK_TYPE_ARN="arn:aws:cloudformation:us-west-2:123456789012:type/hook/
   Organization-Service-Hook/"
   ```

2. Use the set-type-configuration command to specify the configuration data. The JSON you pass for `--configuration` must validate against the Hook's configuration schema. To

activate the Hook, you must set the `HookInvocationStatus` property to `ENABLED` in the `HookConfiguration` section.

```
aws cloudformation set-type-configuration \
  --configuration "{"CloudFormationConfiguration":{"HookConfiguration":
{"HookInvocationStatus": "ENABLED", "FailureMode": "FAIL", "Properties":{}}}}" \
  --type-arn $HOOK_TYPE_ARN --region us-west-2
```

For more information, see Hook configuration schema syntax reference in the *CloudFormation Hooks User Guide*.

Remove third-party private extensions from your account

To remove a third-party private extension or extension version, use the deregister-type command.

You can deregister a specific extension version, or the extension as a whole. To deregister an extension, you must individually deregister all registered versions of that extension. If an extension has only a single registered version, deregistering that version results in the extension itself being deregistered. You can't deregister the default version of an extension, unless it's the only registered version of that extension, in which case the extension itself is deregistered as well.

Warning

Deregistering a private extension can't be undone. This action will:

- Make the extension unusable in all CloudFormation operations.
- Cause failures in future stack updates that use this extension (for modules and resource types). Although you can reregister the extension privately later, this could cause failures if CloudFormation relies on an earlier version.

Before proceeding, use the list-stacks and get-template commands to verify that no active stacks are using this extension.

Example deregister extension commands

This section provides examples that show the different ways to deregister private extensions.

Deregister by type name

Use the deregister-type command with `--type` and `--type-name` options to deregister your extension.

```
aws cloudformation deregister-type \
  --type MODULE \
  --type-name My::S3::SampleBucket::MODULE
```

Deregister by type name and version

To deregister a specific version of your extension, specify the `--version-id` option in the command.

```
aws cloudformation deregister-type \
  --type MODULE \
  --type-name My::S3::SampleBucket::MODULE \
  --version-id 00000001
```

> **Tip**
>
> To set a different version of the extension as default first, use the set-type-default-version command.

Deregister by ARN

Use the `--arn` option and specify your extension's ARN to deregister it.

```
aws cloudformation deregister-type \
  --arn arn:aws:cloudformation:us-west-2:123456789012:type/resource/Organization-
Service-Resource
```

Edit configuration data for extensions in your account

This topic provides guidance on editing configuration data for extensions in your account within a specific Region. Extensions can include configuration properties that apply to all instances of the extension for a given account and Region. These are defined by the extension author in

the extension's configuration definition. If there are any required properties in the extension's configuration definition, you must specify those properties before you can use the extension in your account and Region.

For more information about how configuration definitions are defined when developing an extension, see the following documentation.

- Hook configuration schema syntax reference
- Defining the account-level configuration of an extension

Topics

- Permissions required to use dynamic references
- Edit configuration data for an extension (console)
- Edit configuration data for an extension (AWS CLI)

Permissions required to use dynamic references

If your configuration data includes dynamic references to values stored in AWS Systems Manager or AWS Secrets Manager, any role used to provision the type (for example, when creating or updating a stack) must have the proper permissions to retrieve that value. Specifically:

- If the configuration data contains a parameter stored in AWS Systems Manager Parameter Store, the user or role used to provision the type must have permissions to call GetParameter.
- If the configuration data contains a secret stored in AWS Secrets Manager, the user or role used to provision the type must have permissions to call GetSecretValue.

For more information, see Get values stored in other services using dynamic references.

Edit configuration data for an extension (console)

Follow the steps in this section to use the console to:

- View the current configuration data for an extension
- Update extension configuration data for your account

To view the current configuration data for an extension

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. From the navigation pane, under **Registry**, choose **Activated extensions**.
4. Find the extension you want to view. For more information, see View the available and activated extensions in the CloudFormation registry.
5. Choose the extension to view the extension details.
6. On the extension details page, choose the **Configuration** tab.
7. Expand the **Configuration schema** tab to see the configuration schema defined for the extension.
8. Expand the **Configuration** tab to see the current configuration that you have set for this extension.

To update configuration data for an extension

1. On the extension details page, from the **Configuration** tab, choose **Edit configuration**.

   Alternatively, from **Actions**, choose **Edit**, and then choose **Edit configuration**.

   CloudFormation displays the **Configure extension** page. Make sure that **View configuration schema** is toggled on to see the extension's current configuration definition schema.
2. In the **Configuration JSON** text box, enter a JSON string that represents the configuration schema you want to set for this extension. It must validate against the schema defined in **Configuration schema**.
3. Choose **Configure extension**.

Edit configuration data for an extension (AWS CLI)

Follow the steps in this section to use the AWS CLI to:

- View the current configuration data for an extension
- Update extension configuration data for your account

To view the current configuration data for an extension

- Use the describe-type command to return detailed information about the extension. The `ConfigurationSchema` element of the output contains the current configuration definition of the extension in a given Region.

  Alternatively, use the batch-describe-type-configurations command to return configuration data about multiple extensions.

To update configuration data for an extension

- Use the set-type-configuration command to specify the configuration data. The JSON you pass for `--configuration` must validate against the extension's configuration schema.

  In the following example, the **set-type-configuration** command specifies the configuration data *`"{"CredentialKey": "testUserCredential"}"`* for the `--configuration` option.

```
aws cloudformation set-type-configuration --type RESOURCE \
  --type-name My::Resource::Example \
  --configuration-alias default \
  --configuration "{"CredentialKey": "testUserCredential"}" \
  --region us-west-2
```

Record resource types in AWS Config

You can specify that AWS Config automatically track your private resource types and record changes to those resources as *configuration items*. This enables you to view configuration history for these private resource types, in addition to write AWS Config Rules rules to verify configuration best practices. AWS Config is required for the Hook extension.

To have AWS Config automatically track your private resource types:

- Manage the resources through CloudFormation. This includes performing all resource create, updated, and delete operations through CloudFormation.

> **Note**
>
> If you use an IAM role to perform your stack operations, that IAM role must have permission to call the following AWS Config actions:

- PutResourceConfig
- DeleteResourceConfig

- Configure AWS Config to record all resource types. For more information, see Recording Configurations for Third-Party Resources using the AWS CLI in the *AWS Config Developer Guide*.

> **Note**
>
> AWS Config doesn't support recording of private resources containing properties defined as both required *and* write-only.
> By design, resource properties defined as write-only aren't returned in the schema used to create AWS Config configuration item. Because of this, including a property that's defined as both write-only and required will cause the configuration item creation to fail, as a required property will not be present. To view the schema that will be used to create the configuration item, you can review the `schema` property of the DescribeType action.

For more information about configuration items, see Configuration items in the *AWS Config Developer Guide*.

Preventing sensitive properties being recorded in a configuration item

Your resource type may contain properties that you consider sensitive information, such as passwords, secrets, or other sensitive data, that you don't want recorded as part of the configuration item. To prevent a property from being recorded in the configuration item, you can include that property in the `writeOnlyproperties` list in your resource type schema. Resource properties listed as `writeOnlyproperties` can be specified by the user, but won't be returned by a `read` or `list` request.

For more information, see writeOnlyProperties in the *CloudFormation CLI User Guide*.

Continuous delivery with CodePipeline

Continuous delivery is a release practice in which code changes are automatically built, tested, and prepared for release to production. With CloudFormation and CodePipeline, you can use continuous delivery to automatically build and test changes to your CloudFormation templates before promoting them to production stacks. This release process lets you rapidly and reliably make changes to your AWS infrastructure.

For example, you can create a workflow that automatically builds a test stack when you submit an updated template to a code repository. After CloudFormation builds the test stack, you can test it and then decide whether to push the changes to a production stack. For more information about the benefits of continuous delivery, see What is continuous delivery?.

Use CodePipeline to build a continuous delivery workflow by building a pipeline for CloudFormation stacks. CodePipeline has built-in integration with CloudFormation, so you can specify CloudFormation-specific actions, such as creating, updating, or deleting a stack, within a pipeline. For more information about CodePipeline, see the AWS CodePipeline User Guide.

Topics

- Walkthrough: Building a pipeline for test and production stacks
- CloudFormation configuration properties reference
- CloudFormation artifacts
- Using parameter override functions with CodePipeline pipelines

Walkthrough: Building a pipeline for test and production stacks

Imagine a release process where you submit an CloudFormation template, which CloudFormation then uses to automatically build a test stack. After you review the test stack, you can preview how your changes will modify your production stack, and then choose whether to implement them. To accomplish this workflow, you could use CloudFormation to build your test stack, delete the test stack, create a change set, and then execute the change set. However, with each action, you need to manually interact with CloudFormation. In this walkthrough, we'll build a CodePipeline pipeline that automates many of these actions, helping you achieve a continuous delivery workflow with your CloudFormation stacks.

Prerequisites

This walkthrough assumes that you have used CodePipeline and CloudFormation, and know how pipelines and AWS CloudFormation templates and stacks work. For more information about CodePipeline, see the AWS CodePipeline User Guide. You also need to have an Amazon S3 bucket in the same AWS Region you create your pipeline in.

> **Important**
>
> The sample WordPress template creates an EC2 instance that requires a connection to the Internet. Check that you have a default VPC and subnet that allow traffic to the Internet.

Walkthrough overview

This walkthrough builds a pipeline for a sample WordPress site in a stack. The pipeline is separated into three stages. Each stage must contain at least one action, which is a task the pipeline performs on your artifacts (your input). A stage organizes actions in a pipeline. CodePipeline must complete all actions in a stage before the stage processes new artifacts, for example, if you submitted new input to rerun the pipeline.

By the end of this walkthrough, you'll have a pipeline that performs the following workflow:

1. The first stage of the pipeline retrieves a source artifact (a CloudFormation template and its configuration files) from a repository.

   You'll prepare an artifact that includes a sample WordPress template and upload it to an S3 bucket.
2. In the second stage, the pipeline creates a test stack and then waits for your approval.

   After you review the test stack, you can choose to continue with the original pipeline or create and submit another artifact to make changes. If you approve, this stage deletes the test stack, and then the pipeline continues to the next stage.
3. In the third stage, the pipeline creates a change set against a production stack, and then waits for your approval.

   In your initial run, you won't have a production stack. The change set shows you all of the resources that CloudFormation will create. If you approve, this stage executes the change set and builds your production stack.

> **Note**
>
> CloudFormation is a free service. However, you are charged for the AWS resources, such as the EC2 instance, that you include in your stack at the current rate for each. For more information about AWS pricing, see the detail page for each product at http://aws.amazon.com.

Step 1: Edit the artifact and upload it to an S3 Bucket

Before you build your pipeline, you must set up your source repository and files. CodePipeline copies these source files into your pipeline's artifacts store, and then uses them to perform actions in your pipeline, such as creating a CloudFormation stack.

When you use Amazon Simple Storage Service (Amazon S3) as the source repository, CodePipeline requires you to zip your source files before uploading them to an S3 bucket. The zipped file is a CodePipeline artifact that can contain a CloudFormation template, a template configuration file, or both. We provide an artifact that contains a sample WordPress template and two template configuration files. The two configuration files specify parameter values for the WordPress template. CodePipeline uses them when it creates the WordPress stacks. One file contains parameter values for a test stack, and the other for a production stack. You'll need to edit the configuration files, for example, to specify an existing EC2 key-pair name that you own. For more information about artifacts, see CloudFormation artifacts.

After you build your artifact, you'll upload it to an S3 bucket.

To edit and upload the artifact

1. Download and open the sample artifact: https://s3.amazonaws.com/cloudformation-examples/user-guide/continuous-deployment/wordpress-single-instance.zip.

   The artifact contains three files:

   - The sample WordPress template: `wordpress-single-instance.yaml`
   - The template configuration file for the test stack.: `test-stack-configuration.json`
   - The template configuration file for the production stack: `prod-stack-configuration.json`
2. Extract all of the files, and then use any text editor to modify the template configuration files.

Open the configuration files to see that they contain key-value pairs that map to the WordPress template's parameters. The configuration files specify the parameter values that your pipeline uses when it creates the test and production stacks.

Edit the `test-stack-configuration.json` file to specify parameter values for the test stack and the `prod-stack-configuration.json` file for the production stack.

- Change the values of the `DBPassword` and `DBRootPassword` keys to passwords that you can use to log in to your WordPress database. As defined in the WordPress template, the parameter values must contain only alphanumeric characters.
- Change the value of the `KeyName` key to an existing EC2 key-pair name in the region in which you will create your pipeline.

3. Add the modified configuration files to the original artifact (`.zip`) file, replacing duplicate files.

   You now have a customized artifact that you can upload to an S3 bucket.

4. Upload the artifact to an S3 bucket that you own.

   Note the file's location. You'll specify the location of this file when you build your pipeline.

   Notes about the artifact and S3 bucket:

   - Use a bucket that is in the same AWS Region in which you will create your pipeline.
   - CodePipeline requires that the bucket is versioning enabled.
   - You can also use services that don't require you to zip your files before uploading them, like GitHub or CodeCommit, for your source repository.
   - Artifacts can contain sensitive information such as passwords. Limit access so that only permitted users can view the file. When you do, ensure that CodePipeline can still access the file. For example, if you upload your artifact to an S3 bucket, use S3 bucket policies or user policies to restrict access.

You now have an artifact that CodePipeline can pull in to your pipeline. In the next step, you'll specify the artifact's location and build the WordPress pipeline.

Step 2: Create the pipeline stack

To create the WordPress pipeline, you'll use a sample CloudFormation template. In addition to building the pipeline, the template sets up AWS Identity and Access Management (IAM) service roles for CodePipeline and CloudFormation, an S3 bucket for the CodePipeline artifact store, and an Amazon Simple Notification Service (Amazon SNS) topic to which the pipeline sends notifications, such as notifications about reviews. The sample template makes it easy to provision and configure these resources in a single CloudFormation stack.

For more details about the configuration of the pipeline, see What the pipeline does.

> **Important**
>
> The sample WordPress template creates an EC2 instance that requires a connection to the Internet. Check that your default VPC and subnet allow traffic to the Internet.

To create the pipeline stack

1. Download the sample template at https://s3.amazonaws.com/cloudformation-examples/user-guide/continuous-deployment/basic-pipeline.yml. Save it on your computer.
2. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.
3. Choose an AWS Region that supports CodePipeline and CloudFormation.

   For more information, see CodePipeline endpoints and quotas and CloudFormation endpoints and quotas in the *AWS General Reference*.
4. Choose **Create stack**.
5. Under **Specify template**, choose **Upload a template file**, and then choose the template that you just downloaded, `basic-pipeline.yml`.
6. Choose **Next**.
7. For **Stack name**, type `sample-WordPress-pipeline`.
8. In the **Parameters** section, specify the following parameter values, and then choose **Next**. When setting stack parameters, if you kept the same names for the WordPress template and its configuration files, you can use the default values. If not, specify the filenames that you used.

PipelineName

The name of your pipeline, such as `WordPress-test-pipeline`.

S3Bucket

The name of the S3 bucket where you saved your artifact (`.zip` file).

SourceS3Key

The file name of your artifact. If you saved the artifact in a folder, include it as part of the file name, such as `folder/subfolder/wordpress-single-instance.zip`.

Email

The email address to which CodePipeline sends pipeline notification, such as `myemail@example.com`.

9. For this walkthrough, you don't need to add tags or specify advanced settings, so choose **Next**.
10. Ensure that the stack name and template URL are correct, and then choose **Create stack**.
11. To acknowledge that you're aware that CloudFormation might create IAM resources, choose the check box.

It might take several minutes for AWS CloudFormation to create your stack. To monitor progress, view the stack events. For more information, see Monitor stack progress.

After your stack has been created, CodePipeline starts your new pipeline. To view its status, see the CodePipeline console. From the list of pipelines, choose **WordPress-test-pipeline**.

What the pipeline does

This section explains the pipeline's three stages, using snippets from the sample WordPress pipeline template.

Stage 1: Source

The first stage of the pipeline is a source stage in which you specify the location of your source code. Every time you push a revision to this location, CodePipeline reruns your pipeline.

The source code is located in an S3 bucket and is identified by its file name. You specified these values as input parameter values when you created the pipeline stack. To allow using the source artifact in subsequent stages, the snippet specifies the `OutputArtifacts` property, with the

name `TemplateSource`. To use this artifact in later stages, you specify `TemplateSource` as an input artifact.

```
- Name: S3Source
  Actions:
    - Name: TemplateSource
      ActionTypeId:
        Category: Source
        Owner: AWS
        Provider: S3
        Version: '1'
      Configuration:
        S3Bucket: !Ref 'S3Bucket'
        S3ObjectKey: !Ref 'SourceS3Key'
      OutputArtifacts:
        - Name: TemplateSource
```

Stage 2: TestStage

In the `TestStage` stage, the pipeline creates the test stack, waits for approval, and then deletes the test stack.

For the `CreateStack` action, the pipeline uses the test configuration file and WordPress template to create the test stack. Both files are contained in the `TemplateSource` input artifact, which is brought in from the source stage. The snippet uses the `REPLACE_ON_FAILURE` action mode. If stack creation fails, the pipeline replaces it so that you don't need to clean up or troubleshoot the stack before you can rerun the pipeline. The action mode is useful for quickly iterating on test stacks. For the `RoleArn` property, the value is an AWS CloudFormation service role that is declared elsewhere in the template.

The `ApproveTestStack` action pauses the pipeline and sends a notification to the email address that you specified when you created the pipeline stack. While the pipeline is paused, you can check the WordPress test stack and its resources. Use CodePipeline to approve or reject this action. The `CustomData` property includes a description of the action you're approving, which the pipeline adds to the notification email.

After you approve this action, CodePipeline moves to the `DeleteTestStack` action and deletes the test WordPress stack and its resources.

```
- Name: TestStage
```

```
  Actions:
    - Name: CreateStack
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      InputArtifacts:
        - Name: TemplateSource
      Configuration:
        ActionMode: REPLACE_ON_FAILURE
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref TestStackName
        TemplateConfiguration: !Sub "TemplateSource::${TestStackConfig}"
        TemplatePath: !Sub "TemplateSource::${TemplateFileName}"
      RunOrder: '1'
    - Name: ApproveTestStack
      ActionTypeId:
        Category: Approval
        Owner: AWS
        Provider: Manual
        Version: '1'
      Configuration:
        NotificationArn: !Ref CodePipelineSNSTopic
        CustomData: !Sub 'Do you want to create a change set against the production
stack and delete the ${TestStackName} stack?'
      RunOrder: '2'
    - Name: DeleteTestStack
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: DELETE_ONLY
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref TestStackName
      RunOrder: '3'
```

Stage 3: ProdStage

The `ProdStage` stage of the pipeline creates a change set against the existing production stack, waits for approval, and then executes the change set.

A change set provides a preview of all modifications AWS CloudFormation will make to your production stack before implementing them. On your first pipeline run, you won't have a running production stack. The change set shows the actions that AWS CloudFormation performed when creating the test stack. To create the change set, the `CreateChangeSet` action uses the WordPress sample template and the production template configuration from the `TemplateSource` input artifact.

Similar to the previous stage, the `ApproveChangeSet` action pauses the pipeline and sends an email notification. While the pipeline is paused, you can view the change set to check all of the proposed modifications to the production WordPress stack. Use CodePipeline to approve or reject this action to continue or stop the pipeline, respectively.

After you approve this action, the `ExecuteChangeSet` action executes the changes set, so that AWS CloudFormation performs all of the actions described in the change set. For the initial run, AWS CloudFormation creates the WordPress production stack. On subsequent runs, AWS CloudFormation updates the stack.

```
- Name: ProdStage
  Actions:
    - Name: CreateChangeSet
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      InputArtifacts:
        - Name: TemplateSource
      Configuration:
        ActionMode: CHANGE_SET_REPLACE
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref ProdStackName
        ChangeSetName: !Ref ChangeSetName
        TemplateConfiguration: !Sub "TemplateSource::${ProdStackConfig}"
        TemplatePath: !Sub "TemplateSource::${TemplateFileName}"
      RunOrder: '1'
    - Name: ApproveChangeSet
      ActionTypeId:
        Category: Approval
        Owner: AWS
        Provider: Manual
        Version: '1'
      Configuration:
```

```
        NotificationArn: !Ref CodePipelineSNSTopic
        CustomData: !Sub 'A new change set was created for the ${ProdStackName} stack.
Do you want to implement the changes?'
      RunOrder: '2'
    - Name: ExecuteChangeSet
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: CHANGE_SET_EXECUTE
        ChangeSetName: !Ref ChangeSetName
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: !Ref ProdStackName
      RunOrder: '3'
```

Step 3: View the WordPress stack

As CodePipeline runs through the pipeline, it uses CloudFormation to create test and production stacks. To see the status of these stacks and their output, use the CloudFormation console.

To view a stack

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Depending on whether your pipeline is in the test or production stage, choose the `Test-MyWordPressSite` or the `Prod-MyWordPressSite` stack.
3. To check the status of your stack, view the stack events.

If the stack is in a failed state, view the status reason to find the stack error. Fix the error, and then rerun the pipeline. If the stack is in the `CREATE_COMPLETE` state, view its outputs to get the URL of your WordPress site.

You've successfully used CodePipeline to build a continuous delivery workflow for a sample WordPress site. If you submit changes to the S3 bucket, CodePipeline automatically detects a new version, and then reruns your pipeline. This workflow makes it easier to submit and test changes before making changes to your production site.

Step 4: Clean up resources

To make sure that you are not charged for unwanted services, delete your resources.

> **Important**
>
> Delete the test and production WordPress stacks before deleting the pipeline stack. The pipeline stack contains a service role that's required to delete the WordPress stacks. If you deleted the pipeline stack first, you can associate another service role Amazon Resource Name (ARN) with the WordPress stacks, and then delete them.

To delete objects in the artifact store

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the S3 bucket that CodePipeline used as your pipeline's artifact store.

   The bucket's name follows the format: *`stackname-artifactstorebucket-id`*. If you followed this walkthrough, the bucket's name might look similar to the following example: `sample-WordPress-pipeline-artifactstorebucket-12345abcd12345`.
3. Delete all of the objects in the artifact store S3 bucket.

   When you delete the pipeline stack in the next step, this bucket must be empty. Otherwise, CloudFormation won't be able to delete the bucket.

To delete stacks

1. From the CloudFormation console, choose the stack that you want to delete.

   If the WordPress stacks that were created by the pipeline are still running, choose them first. By default, the stack names are `Test-MyWordPressSite` and `Prod-MyWordPressSite`.

   If you already deleted the WordPress stacks, choose the `sample-WordPress-pipeline` stack.
2. Choose **Actions**, and then choose **Delete Stack**.
3. In the confirmation message, choose **Yes, Delete**.

CloudFormation deletes the stack all of the stack's resources, such as the EC2 instance, notification topic, service role, and the pipeline.

Now that you understand how to build a basic CloudFormation workflow with CodePipeline, you can use the sample template and artifacts as a starting point for building your own.

See also

The following related resources can help you as you work with these parameters.

- For more information about the CloudFormation action parameters in CodePipeline, see the CloudFormation deploy action configuration reference in the *AWS CodePipeline User Guide*.
- For example template values by action provider, such as for the `Owner` field or the `configuration` fields, see the Action structure reference in the *AWS CodePipeline User Guide*.
- To download example pipeline stack templates in YAML or JSON format, see Tutorial: Create a pipeline with CloudFormation in the *AWS CodePipeline User Guide*.

CloudFormation configuration properties reference

When you build a CodePipeline pipeline, you add a `Deploy` action to the pipeline with CloudFormation as a provider. You then must specify which CloudFormation action the pipeline invokes and the action's settings. This topic describes the CloudFormation configuration properties. To specify properties, you can use the CodePipeline console, or you can create a JSON object to use for the AWS CLI, CodePipeline API, or CloudFormation templates.

Topics

- Configuration properties (console)
- Configuration properties (JSON object)
- See also

Configuration properties (console)

The CodePipeline console shows the configuration properties and indicates the properties that are required based on the action mode that you choose.

> **Note**
>
> When you create a pipeline, you can specify the **Create or update a stack** or **Create or replace a change set** action modes only. Properties in the **Advanced** section are available only when you edit a pipeline.

Action mode

The CloudFormation action that CodePipeline invokes when processing the associated stage. Choose one of the following action modes:

- **Create or replace a change set** creates the change set if it doesn't exist based on the stack name and template that you submit. If the change set exists, CloudFormation deletes it, and then creates a new one.
- **Create or update a stack** creates the stack if the specified stack doesn't exist. If the stack exists, CloudFormation updates the stack. Use this action to update existing stacks. CodePipeline won't replace the stack.
- **Delete a stack** deletes a stack. If you specify a stack that doesn't exist, the action is completed successfully without deleting a stack.
- **Execute a change set** executes a change set.
- **Replace a failed stack** creates the stack if the specified stack doesn't exist. If the stack exists and is in a failed state (reported as `ROLLBACK_COMPLETE`, `ROLLBACK_FAILED`, `CREATE_FAILED`, `DELETE_FAILED`, or `UPDATE_ROLLBACK_FAILED`), CloudFormation deletes the stack and then creates a new one. If the stack isn't in a failed state, CloudFormation updates it. Use this action to replace failed stacks without recovering or troubleshooting them. You would typically choose this mode for testing.

Stack name

The name that is associated with an existing stack or a stack you want to create. The name must be unique in the AWS Region in which you are creating the stack.

> **Note**
>
> A stack name can contain only alphanumeric characters (case sensitive) and hyphens. It must start with an alphabetic character and cannot be longer than 128 characters.

Change set name

The name of an existing change set or a new change set that you want to create for the specified stack.

Template

The location of an CloudFormation template file, which follows the format *`ArtifactName`*`::`*`TemplateFileName`*.

Template configuration

The location of a template configuration file, which follows the format *`ArtifactName`*`::`*`TemplateConfigurationFileName`*. The template configuration file can contain template parameter values, a stack policy, and tags. If you include sensitive information, such as passwords, restrict access to this file. For more information, see CloudFormation artifacts.

Capabilities

For stacks that contain certain resources, explicit acknowledgment that CloudFormation might create or update those resources. For example, you must specify `CAPABILITY_IAM` if your stack template contains AWS Identity and Access Management (IAM) resources. For more information, see CreateStack API operation request parameters.

If you have IAM resources in your stack template, you must specify this property.

You can specify more than one capability.

Role name

The name of the IAM service role that CloudFormation assumes when it operates on resources in the specified stack.

Output file name

In the **Advanced** section, you can specify an output file name, such as `CreateStackOutput.json`, that CodePipeline adds to the output artifact after it performs the specified action. The output artifact contains a JSON file with the contents of the `Outputs` section of the CloudFormation template.

If you don't specify a name, CodePipeline doesn't generate an output artifact.

Parameter overrides

Parameters are defined in your template and allow you to input custom values when you create or update a stack. You can specify a JSON object that overrides template parameter values in the template configuration file. All parameter names must be present in the stack template. For more information, see CloudFormation template Parameters syntax.

> **Note**
>
> There is a maximum size limit of 1 kilobyte for the JSON object that can be stored in the `ParameterOverrides` property.

We recommend that you use the template configuration file to specify most of your parameter values. Use parameter overrides to specify dynamic parameter values only. Dynamic parameters are unknown until you run the pipeline.

The following example defines a value for the `ParameterName` parameter by using a parameter override function. The function retrieves a value from a CodePipeline input artifact. For more information about parameter override functions, see Using parameter override functions with CodePipeline pipelines.

```
{
  "ParameterName" : { "Fn::GetParam" : ["ArtifactName", "config-file-name.json",
 "ParamName"]}
}
```

Configuration properties (JSON object)

When you specify `CloudFormation` as a provider for a stage action, define the following properties in the `Configuration` property. Use the JSON object for the AWS CLI, CodePipeline API, or CloudFormation templates. For examples, see Walkthrough: Building a pipeline for test and production stacks and CloudFormation configuration properties reference.

`ActionMode`

The CloudFormation action that CodePipeline invokes when it processes the associated stage. Specify only one of the following action modes:

- CHANGE_SET_EXECUTE executes a change set.
- CHANGE_SET_REPLACE creates the change set, if it doesn't exist, based on the stack name and template that you submit. If the change set exists, CloudFormation deletes it, and then creates a new one.
- CREATE_UPDATE creates the stack if the specified stack doesn't exist. If the stack exists, CloudFormation updates the stack. Use this action to update existing stacks. CodePipeline won't replace the stack.
- DELETE_ONLY deletes a stack. If you specify a stack that doesn't exist, the action is completed successfully without deleting a stack.
- REPLACE_ON_FAILURE creates a stack, if the specified stack doesn't exist. If the stack exists and is in a failed state (reported as ROLLBACK_COMPLETE, ROLLBACK_FAILED, CREATE_FAILED, DELETE_FAILED, or UPDATE_ROLLBACK_FAILED), CloudFormation deletes the stack and then creates a new stack. If the stack isn't in a failed state, CloudFormation updates it. Use this action to automatically replace failed stacks without recovering or troubleshooting them. You would typically choose this mode for testing.

This property is required.

`Capabilities`

For stacks that contain certain resources, explicit acknowledgment that CloudFormation might create or update those resources. For example, you must specify CAPABILITY_IAM if your stack template contains AWS Identity and Access Management (IAM) resources. For more information, see CreateStack API operation request parameters.

This property is conditional. If you have IAM resources in your stack template, you must specify this property.

You can specify multiple capabilities. The following example adds the CAPABILITY_IAM and CAPABILITY_AUTO_EXPAND properties to the template:

YAML

```
configuration:
  ActionMode: CHANGE_SET_REPLACE
  Capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
  ChangeSetName: pipeline-changeset
  RoleArn: CloudFormation_Role_ARN
  StackName: my-pipeline-stack
```

```
  TemplateConfiguration: 'my-pipeline-stack::template-configuration.json'
  TemplatePath: 'my-pipeline-stack::template-export.yml'
```

JSON

```
 "configuration": {
        "ActionMode": "CHANGE_SET_REPLACE",
        "Capabilities": "CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND",
        "ChangeSetName": "pipeline-changeset",
        "RoleArn": "CloudFormation_Role_ARN",
        "StackName": "my-pipeline-stack",
        "TemplateConfiguration": "my-pipeline-stack::template-
configuration.json",
        "TemplatePath": "my-pipeline-stack::template-export.yml"
    }
```

`ChangeSetName`

The name of an existing change set or a new change set that you want to create for the specified stack.

This property is required for the following action modes: `CHANGE_SET_REPLACE` and `CHANGE_SET_EXECUTE`. For all other action modes, this property is ignored.

`OutputFileName`

A name for the output file, such as `CreateStackOutput.json`. CodePipeline adds the file to the output artifact after it performs the specified action. The output artifact contains a JSON file with the contents of the `Outputs` section of the CloudFormation template.

This property is optional. If you don't specify a name, CodePipeline doesn't generate an output artifact.

`ParameterOverrides`

Parameters are defined in your template and allow you to input custom values when you create or update a stack. You can specify a JSON object that overrides template parameter values in the template configuration file. All parameter names must be present in the stack template. For more information, see CloudFormation template Parameters syntax.

The following example adds the `InstanceType` and `KeyName` parameter overrides to the template:

YAML

```
configuration:
  ActionMode: CHANGE_SET_REPLACE
  Capabilities: CAPABILITY_NAMED_IAM
  ChangeSetName: pipeline-changeset
  ParameterOverrides: '{"InstanceType": "t2.small","KeyName": "my-keypair"}'
  RoleArn: CloudFormation_Role_ARN
  StackName: my-pipeline-stack
  TemplateConfiguration: 'my-pipeline-stack::template-configuration.json'
  TemplatePath: 'my-pipeline-stack::template-export.yml'
```

JSON

```
"configuration": {
        "ActionMode": "CHANGE_SET_REPLACE",
        "Capabilities": "CAPABILITY_NAMED_IAM",
        "ChangeSetName": "pipeline-changeset",
        "ParameterOverrides": "{\"InstanceType\": \"t2.small\",\"KeyName\": \"my-
keypair\"}",
        "RoleArn": "CloudFormation_Role_ARN",
        "StackName": "my-pipeline-stack",
        "TemplateConfiguration": "my-pipeline-stack::template-
configuration.json",
        "TemplatePath": "my-pipeline-stack::template-export.yml"
    }
```

Note

The maximum size for the JSON object that can be stored in the `ParameterOverrides` property is 1 kilobyte.

We recommend that you use the template configuration file to specify most of your parameter values. Use parameter overrides to specify dynamic parameter values only. Dynamic parameter values are unknown until you run the pipeline.

The following example defines a value for the `ParameterName` parameter by using a parameter override function. The function retrieves a value from a CodePipeline input artifact. For more information about parameter override functions, see Using parameter override functions with CodePipeline pipelines.

```
{
  "ParameterName" : { "Fn::GetParam" : ["ArtifactName", "config-file-name.json",
 "ParamName"]}
}
```

This property is optional.

`RoleArn`

The Amazon Resource Name (ARN) of the IAM service role that CloudFormation assumes when it operates on resources in a stack.

This property is required for the following action modes: `CREATE_UPDATE`, `REPLACE_ON_FAILURE`, `DELETE_ONLY`, and `CHANGE_SET_REPLACE`. `RoleArn` is not applied when executing a change set. If you do not use CodePipeline to create the change set, make sure that the change set or stack has an associated role.

`StackName`

The name of an existing stack or a stack that you want to create.

This property is required for all action modes.

`TemplateConfiguration`

`TemplateConfiguration` is the template configuration file. You include the file in an input artifact to this action. The template configuration file name follows this format:

`Artifactname``::`*`TemplateConfigurationFileName`*

`Artifactname` is the input artifact name as it appears in CodePipeline. For example, a source stage with the artifact name of `SourceArtifact` and a `test-configuration.json` file name creates a `TemplateConfiguration` name as shown in this example:

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

The template configuration file can contain template parameter values and a stack policy. If you include sensitive information, such as passwords, restrict access to this file. For an example template configuration file, see CloudFormation artifacts.

This property is optional.

`TemplatePath`

`TemplatePath` represents the CloudFormation template file. You include the file in an input artifact to this action. The file name follows this format:

`Artifactname``::`*`TemplateFileName`*

`Artifactname` is the input artifact name as it appears in CodePipeline. For example, a source stage with the artifact name of `SourceArtifact` and a `template.yaml` file name creates a `TemplatePath` name, as shown in this example:

```
"TemplatePath": "SourceArtifact::template.yaml"
```

This property is required for the following action modes: `CREATE_UPDATE`, `REPLACE_ON_FAILURE`, and `CHANGE_SET_REPLACE`. For all other action modes, this property is ignored.

See also

The following related resources can help you as you work with these parameters.

- For more information about the CloudFormation action parameters in CodePipeline, see the CloudFormation deploy action configuration reference in the *AWS CodePipeline User Guide*.
- For example template values by action provider, such as for the `Owner` field or the `configuration` fields, see the Action structure reference in the *AWS CodePipeline User Guide*.
- To download example pipeline stack templates in YAML or JSON format, see Tutorial: Create a pipeline with CloudFormation in the *AWS CodePipeline User Guide*.

CloudFormation artifacts

CodePipeline performs tasks on artifacts as CodePipeline runs a pipeline. For CloudFormation, artifacts can include a stack template file, a template configuration file, or both. CodePipeline uses these artifacts to work with CloudFormation stacks and change sets.

If you use Amazon Simple Storage Service (Amazon S3) as a source repository, you must zip the template and template configuration files into a single file before you upload them to

an S3 bucket. For other repositories, such as GitHub and AWS CodeCommit, upload artifacts without zipping them. For more information, see Create a pipeline, stages, and actions in the *AWS CodePipeline User Guide*.

You can add as many files as you need to your repository. For example, you might want to include two different configurations for the same template: one for a test configuration and another for a production configuration.

This topic describes each artifact type.

Topics

- Stack template file
- Template configuration file
- See also

Stack template file

A stack template file defines the resources that CloudFormation provisions and configures. These files are the same template files that you use when you create or update stacks using CloudFormation. You can use YAML or JSON-formatted templates. For more information about templates, see CloudFormation template sections.

Template configuration file

A template configuration file is a JSON-formatted text file that can specify template parameter values, a stack policy, and tags. Use these configuration files to specify parameter values or a stack policy for a stack. All of the parameter values that you specify must be declared in the associated template.

If you include sensitive information—such as passwords—in this file, restrict access to it. For example, if you upload your artifact to an S3 bucket, use S3 bucket policies or user policies to restrict access.

To create a configuration file, use the following format :

```
{
  "Parameters" : {
    "NameOfTemplateParameter" : "ValueOfParameter",
```

```
    ...
  },
  "Tags" : {
    "TagKey" : "TagValue",
    ...
  },
  "StackPolicy" : {
    "Statement" : [
      StackPolicyStatement
    ]
  }
}
```

The following example specifies `TestEC2Key` for the `KeyName` parameter, adds a `Department` tag whose value is `Marketing`, and adds a stack policy that allows all update actions except for an update that deletes a resource.

```
{
  "Parameters" : {
    "KeyName" : "TestEC2Key"
  },
  "Tags" : {
    "Department" : "Marketing"
  },
  "StackPolicy" : {
    "Statement" : [
      {
        "Effect" : "Allow",
        "NotAction" : "Update:Delete",
        "Principal": "*",
        "Resource" : "*"
      }
    ]
  }
}
```

See also

The following related resources can help you as you work with these parameters.

- For more information about the CloudFormation action parameters in CodePipeline, see the CloudFormation deploy action configuration reference in the *AWS CodePipeline User Guide*.

- For example template values by action provider, such as for the `Owner` field or the `configuration` fields, see the Action structure reference in the *AWS CodePipeline User Guide*.
- To download example pipeline stack templates in YAML or JSON format, see Tutorial: Create a pipeline with CloudFormation in the *AWS CodePipeline User Guide*.

Using parameter override functions with CodePipeline pipelines

In a CodePipeline stage, you can specify parameter overrides for CloudFormation actions. Parameter overrides let you specify template parameter values that override values in a template configuration file. CloudFormation provides functions to help you to specify dynamic values (values that are unknown until the pipeline runs).

Topics

- Fn::GetArtifactAtt
- Fn::GetParam
- See also

`Fn::GetArtifactAtt`

The `Fn::GetArtifactAtt` function retrieves the value of an attribute from an input artifact, such as the S3 bucket name where the artifact is stored. Use this function to specify attributes of an artifact, such as its file name or Amazon S3 bucket name.

When you run a pipeline, CodePipeline copies and writes files to the pipeline's artifact store (an S3 bucket). CodePipeline generates the file names in the artifact store. These file names are unknown before you run the pipeline.

For example, in your pipeline, you might have a source stage where CodePipeline copies your AWS Lambda function source code to the artifact store. In the next stage, you have an CloudFormation template that creates the Lambda function, but CloudFormation requires the file name to create the function. You must use the `Fn::GetArtifactAtt` function to pass the exact S3 bucket and file names.

Syntax

Use the following syntax to retrieve an attribute value of an artifact.

```
{ "Fn::GetArtifactAtt" : [ "artifactName", "attributeName" ] }
```

`artifactName`

The name of the input artifact. You must declare this artifact as input for the associated action.

`attributeName`

The name of the artifact attribute whose value you want to retrieve. For details about each artifact attribute, see the following Attributes section.

Example

The following parameter overrides specify the `BucketName` and `ObjectKey` parameters by retrieving the S3 bucket name and file name of the `LambdaFunctionSource` artifact. This example assumes that CodePipeline copied Lambda function source code and saved it as an artifact, for example, as part of a source stage.

```
{
  "BucketName" : { "Fn::GetArtifactAtt" : ["LambdaFunctionSource", "BucketName"]},
  "ObjectKey" : { "Fn::GetArtifactAtt" : ["LambdaFunctionSource", "ObjectKey"]}
}
```

Attributes

You can retrieve the following attributes for an artifact.

`BucketName`

The name of the S3 bucket where the artifact is stored.

`ObjectKey`

The name of the `.zip` file that contains the artifact that is generated by CodePipeline, such as `1ABCyZZ.zip`.

URL

The Amazon Simple Storage Service (Amazon S3) URL of the artifact, such as `https://s3.us-west-2.amazonaws.com/artifactstorebucket-yivczw8jma0c/test/TemplateSo/1ABCyZZ.zip`.

Fn::GetParam

The Fn::GetParam function returns a value from a key-value pair in a JSON-formatted file. The JSON file must be included in an artifact.

Use this function to retrieve output values from an CloudFormation stack and use them as input for another action. For example, if you specify an output file name for an CloudFormation action, CodePipeline saves the output in a JSON file and then adds it to the output artifact's .zip file. Use the Fn::GetParam function to retrieve the output value, and use it as input for another action.

Syntax

Use the following syntax to retrieve a value from a key-value pair.

```
{ "Fn::GetParam" : [ "artifactName", "JSONFileName", "keyName" ] }
```

artifactName

The name of the artifact, which must be included as an input artifact for the associated action.

JSONFileName

The name of a JSON file that is contained in the artifact.

keyName

The name of the key whose value you want to retrieve.

Examples

The following examples demonstrate how to use the Fn::GetParam function in a parameter override.

Syntax

The following parameter override specifies the WebSiteURL parameter by retrieving the value of the URL key from the stack-output.json file that is in the WebStackOutput artifact.

```
{
  "WebSiteURL" : { "Fn::GetParam" : ["WebStackOutput", "stack-output.json", "URL"]}
```

```
}
```

CloudFormation template snippets

The following CloudFormation template snippets, from a CodePipeline pipeline, demonstrate how to pass stack outputs. These snippets show two stages of pipeline definition. The first stage creates a stack and saves its outputs in the `TestOutput.json` file in the `StackAOutput` artifact. These values are specified by the `OutputFileName` and `OutputArtifacts` properties.

The name of the source input artifact for the stages is `TemplateSource`. The file name for the stack template is `teststackA.yaml`, and the file name for the configuration file is `test-configuration.json`. In both stages, these values are specified for the `TemplateConfiguration` and `TemplatePath` properties as shown:

```
TemplateConfiguration: TemplateSource::test-configuration.json
TemplatePath: TemplateSource::teststackA.yaml
```

Example Create stack A stage

```
- Name: CreateTestStackA
  Actions:
    - Name: CloudFormationCreate
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: CREATE_UPDATE
        Capabilities: CAPABILITY_IAM
        OutputFileName: TestOutput.json
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: StackA
        TemplateConfiguration: TemplateSource::test-configuration.json
        TemplatePath: TemplateSource::teststackA.yaml
      InputArtifacts:
        - Name: TemplateSource
      OutputArtifacts:
        - Name: StackAOutput
      RunOrder: '1'
```

In a subsequent stage, stack B uses the outputs from stack A. In the `ParameterOverrides` property, the example uses the `Fn::GetParam` function to specify the `StackBInputParam` parameter. The resulting value is the value associated with the `StackAOutputName` key.

Example Create stack B stage

```
- Name: CreateTestStackB
  Actions:
    - Name: CloudFormationCreate
      ActionTypeId:
        Category: Deploy
        Owner: AWS
        Provider: CloudFormation
        Version: '1'
      Configuration:
        ActionMode: CREATE_UPDATE
        Capabilities: CAPABILITY_IAM
        RoleArn: !GetAtt [CFNRole, Arn]
        StackName: StackB
        TemplateConfiguration: TemplateSource::test-configuration.json
        TemplatePath: TemplateSource::teststackB.yaml
        ParameterOverrides: |
          {
            "StackBInputParam" : { "Fn::GetParam" : ["StackAOutput", "TestOutput.json",
 "StackAOutputName"]}
          }
      InputArtifacts:
        - Name: TemplateSource
        - Name: StackAOutput
      RunOrder: '1'
```

See also

The following related resources can help you as you work with these parameters.

- For more information about the CloudFormation action parameters in CodePipeline, see the CloudFormation deploy action configuration reference in the *AWS CodePipeline User Guide*.
- For example template values by action provider, such as for the `Owner` field or the `configuration` fields, see the Action structure reference in the *AWS CodePipeline User Guide*.
- To download example pipeline stack templates in YAML or JSON format, see Tutorial: Create a pipeline with CloudFormation in the *AWS CodePipeline User Guide*.

The CloudFormation Template Reference Guide

Detailed information about the resources and components available for creating CloudFormation templates can now be found in the CloudFormation Template Reference Guide. This guide serves as a technical reference for templates and complements the getting started content in the CloudFormation User Guide.

The *CloudFormation Template Reference Guide* provides details about:

- Resource types and their properties
- `DeletionPolicy` and other resource-level attributes
- Intrinsic functions, such as `Ref` and `Sub`
- Transforms (CloudFormation macros)
- Helper scripts for EC2 instance configuration
- Machine-readable resource type definitions

This guide is regularly updated to reflect:

- New resource types
- Property changes
- New return values
- Documentation updates

Security in CloudFormation

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to CloudFormation, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using CloudFormation. The following topics show you how to configure CloudFormation to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your CloudFormation resources.

Topics

- Protect CloudFormation stacks from being deleted
- Prevent updates to stack resources
- Data protection in CloudFormation
- Control CloudFormation access with AWS Identity and Access Management
- Logging CloudFormation API calls with AWS CloudTrail
- Infrastructure security in CloudFormation
- Resilience in CloudFormation
- Compliance validation for CloudFormation
- Configuration and vulnerability analysis in CloudFormation
- Security best practices for CloudFormation

- Access CloudFormation using an interface endpoint (AWS PrivateLink)

Protect CloudFormation stacks from being deleted

You can prevent a stack from being accidentally deleted by enabling termination protection on the stack. If a user attempts to delete a stack with termination protection enabled, the deletion fails and the stack, including its status, remains unchanged. You can enable termination protection on a stack when you create it. Termination protection on stacks is disabled by default. You can set termination protection on a stack with any status except `DELETE_IN_PROGRESS` or `DELETE_COMPLETE`.

Enabling or disabling termination protection on a stack passes the same choice on to any nested stacks belonging to that stack as well. You can't enable or disable termination protection directly on a nested stack. If a user attempts to directly delete a nested stack belonging with a stack that has termination protection enabled, the operation fails and the nested stack remains unchanged.

However, if a user performs a stack update that would delete the nested stack, CloudFormation deletes the nested stack accordingly.

Termination protection is different than disabling rollback. Termination protection applies only to attempts to delete stacks, while disabling rollback applies to auto rollback when stack creation fails.

To enable termination protection when creating a stack

On the **Specify stack options** page of the **Create stack** wizard, under **Advanced options**, expand the **Termination Protection** section and select **Enable**. For more information, see Configure stack options.

To enable or disable termination protection on an existing stack

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. Select the stack that you want.

> **Note**
>
> If **NESTED** is displayed next to the stack name, the stack is a nested stack. You can only change termination protection on the root stack to which the nested stack belongs.

4. In the stack details pane, select **Stack actions** and then **Edit termination protection**.

   CloudFormation displays the **Edit termination protection** dialog box.
5. Choose **Enable** or **Disable**, and then select **Save**.

To enable or disable termination protection on a nested stack

If **NESTED** is displayed next to the stack name, the stack is a nested stack. You can only change termination protection on the root stack to which the nested stack belongs. To change termination protection on the root stack:

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose your AWS Region.
3. Select the nested stack that you want.
4. In the **Stack info** pane, in the **Overview** section, select the stack name listed as **Root stack**.

   CloudFormation displays the stack details for the root stack.
5. Choose **Stack actions** and then choose **Edit Termination Protection**.

   CloudFormation displays the **Edit termination protection** dialog box.
6. Choose **Enable** or **Disable**, and then select **Save**.

To enable or disable termination protection using the command line

Use the update-termination-protection command.

Controlling who can change termination protection on stacks

To enable or disable termination protection on stacks, a user requires permission to the `cloudformation:UpdateTerminationProtection` action. For example, the policy below allows users to enable or disable termination protection on stacks.

For more information on specifying permissions in CloudFormation, see Control CloudFormation access with AWS Identity and Access Management.

Example A sample policy that grants permissions to change stack termination protection

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "cloudformation:UpdateTerminationProtection"
        ],
        "Resource":"*"
    }]
}
```

Prevent updates to stack resources

When you create a stack, all update actions are allowed on all resources. By default, anyone with stack update permissions can update all of the resources in the stack. During an update, some resources might require an interruption or be completely replaced, resulting in new physical IDs or completely new storage. You can prevent stack resources from being unintentionally updated or deleted during a stack update by using a stack policy. A stack policy is a JSON document that defines the update actions that can be performed on designated resources.

After you set a stack policy, all of the resources in the stack are protected by default. To allow updates on specific resources, you specify an explicit `Allow` statement for those resources in your stack policy. You can define only one stack policy per stack, but, you can protect multiple resources within a single policy. A stack policy applies to all CloudFormation users who attempt to update the stack. You can't associate different stack policies with different users.

A stack policy applies only during stack updates. It doesn't provide access controls like an AWS Identity and Access Management (IAM) policy. Use a stack policy only as a fail-safe mechanism to prevent accidental updates to specific stack resources. To control access to AWS resources or actions, use IAM.

Topics

- Example stack policy
- Defining a stack policy
- Setting a stack policy
- Updating protected resources
- Modifying a stack policy
- More example stack policies

Example stack policy

The following example stack policy prevents updates to the `ProductionDatabase` resource:

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    },
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/ProductionDatabase"
    }
  ]
}
```

When you set a stack policy, all resources are protected by default. To allow updates on all resources, we add an `Allow` statement that allows all actions on all resources. Although the `Allow` statement specifies all resources, the explicit `Deny` statement overrides it for the resource with the `ProductionDatabase` logical ID. This `Deny` statement prevents all update actions, such as replacement or deletion, on the `ProductionDatabase` resource.

The `Principal` element is required, but supports only the wild card (*), which means that the statement applies to all principals.

> **Note**
>
> During a stack update, CloudFormation automatically updates resources that depend on other updated resources. For example, CloudFormation updates a resource that references an updated resource. CloudFormation makes no physical changes, such as the resource's ID, to automatically updated resources, but if a stack policy is associated with those resources, you must have permission to update them.

Defining a stack policy

When you create a stack, no stack policy is set, so all update actions are allowed on all resources. To protect stack resources from update actions, define a stack policy and then set it on your stack. A stack policy is a JSON document that defines the CloudFormation stack update actions that CloudFormation users can perform and the resources that the actions apply to. You set the stack policy when you create a stack, by specifying a text file that contains your stack policy or typing it out. When you set a stack policy on your stack, any update not explicitly allowed is denied by default.

You define a stack policy with five elements: `Effect`, `Action`, `Principal`, `Resource`, and `Condition`. The following pseudo code shows stack policy syntax.

```
{
  "Statement" : [
    {
      "Effect" : "Deny_or_Allow",
      "Action" : "update_actions",
      "Principal" : "*",
      "Resource" : "LogicalResourceId/resource_logical_ID",
      "Condition" : {
        "StringEquals_or_StringLike" : {
          "ResourceType" : [resource_type, ...]
        }
      }
    }
  ]
}
```

`Effect`

Determines whether the actions that you specify are denied or allowed on the resource(s) that you specify. You can specify only `Deny` or `Allow`, such as:

```
"Effect" : "Deny"
```

> **Important**
>
> If a stack policy includes overlapping statements (both allowing and denying updates on a resource), a `Deny` statement always overrides an `Allow` statement. To ensure that a resource is protected, use a `Deny` statement for that resource.

Action

Specifies the update actions that are denied or allowed:

Update:Modify

Specifies update actions during which resources might experience no interruptions or some interruptions while changes are being applied. All resources maintain their physical IDs.

Update:Replace

Specifies update actions during which resources are recreated. CloudFormation creates a new resource with the specified updates and then deletes the old resource. Because the resource is recreated, the physical ID of the new resource might be different.

Update:Delete

Specifies update actions during which resources are removed. Updates that completely remove resources from a stack template require this action.

Update:*

Specifies all update actions. The asterisk is a wild card that represents all update actions.

The following example shows how to specify just the replace and delete actions:

```
"Action" : ["Update:Replace", "Update:Delete"]
```

To allow all update actions except for one, use `NotAction`. For example, to allow all update actions except for `Update:Delete`, use `NotAction`, as shown in this example:

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "NotAction" : "Update:Delete",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Principal

The `Principal` element specifies the entity that the policy applies to. This element is required but supports only the wild card (*), which means that the policy applies to all principals.

Resource

Specifies the logical IDs of the resources that the policy applies to. To specify types of resources, use the `Condition` element.

To specify a single resource, use its logical ID. For example:

```
"Resource" : ["LogicalResourceId/myEC2instance"]
```

You can use a wild card with logical IDs. For example, if you use a common logical ID prefix for all related resources, you can specify all of them with a wild card:

```
"Resource" : ["LogicalResourceId/CriticalResource*"]
```

You can also use a `NotResource` element to allow updates to all resources except for those you specify. However, using `NotResource` with `Effect: Allow` does not reliably prevent updates to the excluded resources. To protect a specific resource from updates, use an explicit `Deny` statement instead, as shown in the following example:

```
{
  "Statement" : [
    {
```

```
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/ProductionDatabase"
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Conditions

Specifies the resource type that the policy applies to. To specify the logical IDs of specific resources, use the `Resource` element.

You can specify a resource type, such as all EC2 and RDS DB instances, as shown in the following example:

```
{
  "Statement" : [
  {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "Update:*",
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "ResourceType" : ["AWS::EC2::Instance", "AWS::RDS::DBInstance"]
      }
    }
  },
  {
    "Effect" : "Allow",
    "Principal" : "*",
    "Action" : "Update:*",
    "Resource" : "*"
  }
  ]
}
```

The `Allow` statement grants update permissions to all resources and the `Deny` statement denies updates to EC2 and RDS DB instances. The `Deny` statement always overrides allow actions.

You can use a wild card with resource types. For example, you can deny update permissions to all Amazon EC2 resources—such as instances, security groups, and subnets—by using a wild card, as shown in the following example:

```
"Condition" : {
  "StringLike" : {
    "ResourceType" : ["AWS::EC2::*"]
  }
}
```

You must use the `StringLike` condition when you use wild cards.

Setting a stack policy

You can use the console or AWS CLI to apply a stack policy when you create a stack. You can also use the AWS CLI to apply a stack policy to an existing stack. After you apply a stack policy, you can't remove it from the stack, but you can use the AWS CLI to modify it.

Stack policies apply to all CloudFormation users who attempt to update the stack. You can't associate different stack policies with different users.

For information about writing stack policies, see Defining a stack policy.

To set a stack policy when you create a stack (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the navigation bar at the top of the screen, choose the AWS Region to create the stack in.
3. On the **CloudFormation Stacks** page, choose **Create stack**.
4. In the Create Stack wizard, on the **Configure stack options** page, expand the **Advanced** section and then choose **Stack policy**.
5. Specify the stack policy:

   - To write a policy directly in the console, choose **Enter stack policy** and then type the stack policy directly in the text field.

- To use a policy defined in a separate file, choose **Upload a file**, then **Choose file** to select the file containing the stack policy.

To set a stack policy when you create a stack (AWS CLI)

- Use the create-stack command with the `--stack-policy-body` option to type in a modified policy or the `--stack-policy-url` option to specify a file containing the policy.

To set a stack policy on an existing stack (AWS CLI only)

- Use the set-stack-policy command with the `--stack-policy-body` option to type in a modified policy or the `--stack-policy-url` option to specify a file containing the policy.

> **Note**
>
> To add a policy to an existing stack, you must have permission to the CloudFormation SetStackPolicy action.

Updating protected resources

To update protected resources, create a temporary policy that overrides the stack policy and allows updates on those resources. Specify the override policy when you update the stack. The override policy doesn't permanently change the stack policy.

To update protected resources, you must have permission to use the CloudFormation SetStackPolicy action. For information about setting CloudFormation permissions, see Control CloudFormation access with AWS Identity and Access Management.

> **Note**
>
> During a stack update, CloudFormation automatically updates resources that depend on other updated resources. For example, CloudFormation updates a resource that references an updated resource. CloudFormation makes no physical changes, such as the resources' ID, to automatically updated resources, but if a stack policy is associated with those resources, you must have permission to update them.

To update a protected resource (console)

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Select the stack that you want to update, choose **Stack actions**, and then choose **Update stack**.
3. If you *haven't* modified the stack template, select **Use current template**, and then click **Next**. If you have modified the template, select **Replace current template** and specify the location of the updated template in the **Specify template** section:

   - For a template stored locally on your computer, select **Upload a template file**. Choose **Choose File** to navigate to the file and select it, and then click **Next**.
   - For a template stored in an Amazon S3 bucket, select **Amazon S3 URL**. Enter or paste the URL for the template, and then click **Next**.

     If you have a template in a versioning-enabled bucket, you can specify a specific version of the template template by appending `?versionId=`*`version-id`* to the URL. For more information, see Working with objects in a versioning-enabled bucket in the *Amazon Simple Storage Service User Guide*.
4. If your template contains parameters, on the **Specify stack details** page, enter or modify the parameter values, and then choose **Next**.

   CloudFormation populates each parameter with the value that is currently set in the stack except for parameters declared with the `NoEcho` attribute. You can use current values for those parameters by choosing **Use existing value**.

   For more information about using `NoEcho` to mask sensitive information, as well as using dynamic parameters to manage secrets, see the Do not embed credentials in your templates best practice.
5. Specify an override stack policy.

   a. On the **Configure stack options** page, in the **Advanced options** section, select **Stack policy**.

   b. Select **Upload a file**.

   c. Click **Choose file** and navigate to the file that contains the overriding stack policy or type a policy.

   d. Choose **Next**.

The override policy must specify an `Allow` statement for the protected resources that you want to update. For example, to update all protected resources, specify a temporary override policy that allows all updates:

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

> **Note**
>
> CloudFormation applies the override policy only during this update. The override policy doesn't permanently change the stack policy. To modify a stack policy, see Modifying a stack policy.

6. Review the stack information and the changes that you submitted.

   Check that you submitted the correct information, such as the correct parameter values or template URL. If your template contains IAM resources, choose **I acknowledge that this template may create IAM resources** to specify that you want to use IAM resources in the template. For more information, see Acknowledging IAM resources in CloudFormation templates.

   In the **Preview your changes** section, check that CloudFormation will make all the changes that you expect. For example, check that CloudFormation adds, removes, and modifies the resources that you intended to add, remove, or modify. CloudFormation generates this preview by creating a change set for the stack. For more information, see Update CloudFormation stacks using change sets.

7. When you are satisfied with your changes, click **Update**.

> **Note**
>
> At this point, you also have the option to view the change set to review your proposed updates more thoroughly. To do so, click **View change set** instead of **Update**. CloudFormation displays the change set generated based on your updates. When you are ready to perform the stack update, click **Execute**.

CloudFormation displays the **Stack details** page for your stack. Your stack now has a status of `UPDATE_IN_PROGRESS`. After CloudFormation has successfully finished updating the stack, it sets the stack status to `UPDATE_COMPLETE`.

If the stack update fails, CloudFormation; automatically rolls back changes, and sets the stack status to `UPDATE_ROLLBACK_COMPLETE`.

To update a protected resource (AWS CLI)

- Use the update-stack command with the `--stack-policy-during-update-body` option to type in a modified policy or the `--stack-policy-during-update-url` option to specify a file containing the policy.

> **Note**
>
> CloudFormation applies the override policy only during this update. The override policy doesn't permanently change the stack policy. To modify a stack policy, see Modifying a stack policy.

Modifying a stack policy

To protect additional resources or to remove protection from resources, modify the stack policy. For example, when you add a database that you want to protect to your stack, add a `Deny` statement for that database to the stack policy. To modify the policy, you must have permission to use the SetStackPolicy action.

Use the AWS CLI to modify stack policies.

To modify a stack policy (AWS CLI)

- Use the set-stack-policy command with the `--stack-policy-body` option to type in a modified policy or the `--stack-policy-url` option to specify a file containing the policy.

You can't delete a stack policy. To remove all protection from all resources, you modify the policy to explicitly allow all actions on all resources. The following policy allows all updates on all resources:

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

More example stack policies

The following example policies show how to prevent updates to all stack resources and to specific resources, and prevent specific types of updates.

Prevent updates to all stack resources

To prevent updates to all stack resources, the following policy specifies a `Deny` statement for all update actions on all resources.

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Prevent updates to a single resource

The following policy denies all update actions on the database with the `MyDatabase` logical ID. It allows all update actions on all other stack resources with an `Allow` statement. The `Allow` statement doesn't apply to the `MyDatabase` resource because the `Deny` statement always overrides allow actions.

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "LogicalResourceId/MyDatabase"
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

⚠ Avoid using `NotResource` to protect resources

Consider using a `NotResource` element with `Effect: Allow` to protect a resource through default denial, as in the following example:

```
{
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "NotResource" : "LogicalResourceId/ProductionDatabase"
    }
  ]
}
```

This policy does *not* reliably prevent updates to the `ProductionDatabase` resource. AWS CloudFormation evaluates stack policies against both the logical resource ID and the resource type independently. A default denial blocks an update only when both evaluations result in a denied status. Because this policy excludes only the logical resource ID from the `Allow` statement, the resource type remains implicitly allowed. AWS CloudFormation then permits the update.
Always use an explicit `Deny` statement to protect resources, as shown in the previous example.

Prevent updates to all instances of a resource type

The following policy denies all update actions on the RDS DB instance resource type. It allows all update actions on all other stack resources with an `Allow` statement. The `Allow` statement doesn't apply to the RDS DB instance resources because a `Deny` statement always overrides allow actions.

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "ResourceType" : ["AWS::RDS::DBInstance"]
        }
      }
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Prevent replacement updates for an instance

The following policy denies updates that would cause a replacement of the instance with the `MyInstance` logical ID. It allows all update actions on all other stack resources with an `Allow` statement. The `Allow` statement doesn't apply to the `MyInstance` resource because the `Deny` statement always overrides allow actions.

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:Replace",
      "Principal": "*",
      "Resource" : "LogicalResourceId/MyInstance"
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Prevent updates to nested stacks

The following policy denies all update actions on the CloudFormation stack resource type (nested stacks). It allows all update actions on all other stack resources with an `Allow` statement. The `Allow` statement doesn't apply to the CloudFormation stack resources because the `Deny` statement always overrides allow actions.

```
{
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "ResourceType" : ["AWS::CloudFormation::Stack"]
```

```
        }
      }
    },
    {
      "Effect" : "Allow",
      "Action" : "Update:*",
      "Principal": "*",
      "Resource" : "*"
    }
  ]
}
```

Data protection in CloudFormation

The AWS shared responsibility model applies to data protection in AWS CloudFormation. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see Data Privacy FAQ. For information about data protection in Europe, see the General Data Protection Regulation (GDPR) Center.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with CloudFormation or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

Following the AWS shared responsibility model, CloudFormation stores your data encrypted at rest. Customers are responsible for setting encryption and storage policies for data stored in their accounts. For example, we recommend enabling at-rest encryption for templates and other data stored in S3 buckets or SNS topics. Customers similarly define encryption settings for any data storage systems provisioned by CloudFormation.

Encryption in transit

CloudFormation uses encrypted channels for service communications under the shared responsibility model.

Internetwork traffic privacy

CloudFormation service communications are securely encrypted by default between Regions or Availability Zones.

Control CloudFormation access with AWS Identity and Access Management

With AWS Identity and Access Management (IAM), you can create IAM users and control their access to specific resources in your AWS account. When you use IAM, you can control what users can do with CloudFormation, such as whether they can view stack templates, create stacks, or delete stacks.

Beyond CloudFormation-specific actions, you can manage what AWS services and resources are available to each user. That way, you can control which resources users can access when they use CloudFormation. For example, you can specify which users can create Amazon EC2 instances, terminate database instances, or update VPCs. Those same permissions apply anytime they use CloudFormation to do those actions.

Use the information in the following sections to control who can access CloudFormation. We will also explore how to authorize IAM resource creation in templates, give applications running on EC2 instances the permissions they need, and make use of temporary security credentials for enhanced security in your AWS environment.

Defining IAM identity-based policies for CloudFormation

To give access to CloudFormation, you need to create and assign IAM policies that give your IAM identities (such as users or roles) permission to call the API actions they need.

With IAM identity-based policies, you can specify allowed or denied actions and resources, as well as the conditions under which actions are allowed or denied. CloudFormation supports specific actions, resources, and condition keys.

If you're new to IAM, start by familiarizing yourself with the elements of an IAM JSON policy. For more information, see IAM JSON policy element reference in the *IAM User Guide*. To learn how to create IAM policies, complete the tutorial Create and attach your first customer managed policy in the IAM documentation.

Topics

- Policy actions for CloudFormation
- Console-specific actions for CloudFormation
- Policy resources for CloudFormation
- Policy condition keys for CloudFormation

Policy actions for CloudFormation

In the `Action` element of your IAM policy statement, you can specify any API action that CloudFormation offers. You must prefix the action name with the lowercase string `cloudformation:`. For example: `cloudformation:CreateStack`, `cloudformation:CreateChangeSet`, and `cloudformation:UpdateStack`.

To specify multiple actions in a single statement, separate them with commas, as follows:

```
"Action": [ "cloudformation:action1", "cloudformation:action2" ]
```

You can also specify multiple actions using wildcards. For example, you can specify all actions whose names begin with the word `Get`, as follows:

```
"Action": "cloudformation:Get*"
```

To see a complete list of actions associated with the `cloudformation` service prefix, see Actions, resources, and condition keys for CloudFormation and Actions, resources, and condition keys for AWS Cloud Control API in the *Service Authorization Reference*.

Examples

The following shows an example of a permissions policy that grants permissions to view CloudFormation stacks.

Example 1: A sample policy that grants view stack permissions

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "cloudformation:DescribeStacks",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStackResource",
            "cloudformation:DescribeStackResources"
        ],
        "Resource":"*"
    }]
}
```

Users who create or delete stacks need additional permissions based on their stack templates. For example, if your template describes an Amazon SQS queue, users must have permissions for both CloudFormation and Amazon SQS actions, as shown in the following sample policy.

Example 2: A sample policy that grants create and view stack actions and all Amazon SQS actions

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "sqs:*",
            "cloudformation:CreateStack",
            "cloudformation:DescribeStacks",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStackResources",
            "cloudformation:GetTemplate",
            "cloudformation:ValidateTemplate"
        ],
        "Resource":"*"
    }]
}
```

Console-specific actions for CloudFormation

Users of the CloudFormation console require additional permissions beyond those needed for the AWS Command Line Interface or CloudFormation APIs. These additional permissions support console-specific features such as template uploads to Amazon S3 buckets and drop-down lists for AWS-specific parameter types.

For all actions listed below, grant permissions to all resources; don't limit them to specific stacks or buckets.

The following action is used only by the CloudFormation console and is not documented in the API reference. The action allows users to upload templates to Amazon S3 buckets.

- `cloudformation:CreateUploadBucket`

When users upload templates, users also require the following Amazon S3 permissions:

- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetObject`
- `s3:CreateBucket`

To see values in the parameter drop-down lists for templates with AWS-specific parameter types, users need permissions to make the corresponding describe API calls. For example, the following permissions are required when these parameter types are used in the template:

- `ec2:DescribeKeyPairs` – Required for the `AWS::EC2::KeyPair::KeyName` parameter type.
- `ec2:DescribeSecurityGroups` – Required for the `AWS::EC2::SecurityGroup::Id` parameter type.
- `ec2:DescribeSubnets` – Required for the `AWS::EC2::Subnet::Id` parameter type.
- `ec2:DescribeVpcs` – Required for the `AWS::EC2::VPC::Id` parameter type.

For more information about AWS-specific parameter types, see Specify existing resources at runtime with CloudFormation-supplied parameter types.

Policy resources for CloudFormation

In an IAM policy statement, the `Resource` element specifies the object or objects that the statement covers. For CloudFormation, each IAM policy statement applies to the resources that you specify using their Amazon Resource Names (ARNs). The specific ARN format depends on the resource.

For a complete list of CloudFormation resource types and their ARNs, see Resource types defined by CloudFormation in the *Service Authorization Reference*. To learn with which actions you can specify with each resource's ARN, see Actions defined by CloudFormation.

You can specify actions for a specific stack, as shown in the following policy example. When you provide an ARN, replace the *`placeholder text`* with your resource-specific information.

Example 1: A sample policy that denies the delete and update stack actions for the specified stack

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
```

```
                "cloudformation:DeleteStack",
                "cloudformation:UpdateStack"
            ],
            "Resource": "arn:aws:cloudformation:us-
east-1:111122223333:stack/MyProductionStack/*"
        }
    ]
}
```

The policy above uses a wild card at the end of the stack name so that delete stack and update stack are denied on both the full stack ID (such as `arn:aws:cloudformation:region:account-id:stack/MyProductionStack/abc9dbf0-43c2-11e3-a6e8-50fa526be49c`) and the stack name (such as *MyProductionStack*).

To allow `AWS::Serverless` transforms to create a change set, include the `arn:aws:cloudformation:region:aws:transform/Serverless-2016-10-31` resource-level permission, as shown in the following policy.

Example 2: A sample policy that allows the create change set action for the specified transform

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateChangeSet"
            ],
            "Resource": "arn:aws:cloudformation:us-east-1:aws:transform/
Serverless-2016-10-31"
        }
    ]
}
```

Policy condition keys for CloudFormation

In an IAM policy statement, you can optionally specify conditions that control when a policy is in effect. For example, you can define a policy that allows users to create a stack only when they specify a certain template URL. You can define CloudFormation-specific conditions and AWS-wide conditions, such as `DateLessThan`, which specifies when a policy stops taking effect. For more information and a list of AWS-wide conditions, see Condition in IAM policy elements reference in *IAM User Guide*.

> **Note**
>
> Don't use the `aws:SourceIp` AWS-wide condition. CloudFormation provisions resources by using its own IP address, not the IP address of the originating request. For example, when you create a stack, CloudFormation makes requests from its IP address to launch an Amazon EC2 instance or to create an Amazon S3 bucket, not from the IP address from the `CreateStack` call or the **create-stack** command.

The following list describes the CloudFormation-specific conditions. These conditions are applied only when users create or update stacks:

`cloudformation:ChangeSetName`

An CloudFormation change set name that you want to associate with a policy. Use this condition to control which change sets users can execute or delete.

`cloudformation:ImportResourceTypes`

The template resource types that you want to associate with a policy, such as `AWS::EC2::Instance`. Use this condition to control which resource types users can work with when they import resources into a stack. This condition is checked against the resource types that users declare in the `ResourcesToImport` parameter, which is currently supported only for AWS CLI and API requests. When using this parameter, you must specify all the resource types you want users to control during import operations. For more information about the `ResourcesToImport` parameter, see the CreateChangeSet action in the *AWS CloudFormation API Reference*.

For a list of possible `ResourcesToImport`, see Resource type support.

Use the three-part resource naming convention to specify which resource types users can work with, from all resources across an organization, down to an individual resource type.

organization::*

Specify all resource types for a given organization.

organization::*service_name*::*

Specify all resource types for the specified service within a given organization.

organization::*service_name*::*resource_type*

Specify a specific resource type.

For example:

AWS::*

Specify all supported AWS resource types.

AWS::*service_name*::*

Specify all supported resources for a specific AWS service.

AWS::*service_name*::*resource_type*

Specify a specific AWS resource type, such as AWS::EC2::Instance (all EC2 instances).

cloudformation:ResourceTypes

The template resource types, such as AWS::EC2::Instance, that you want to associate with a policy. Use this condition to control which resource types users can work with when they create or update a stack. This condition is checked against the resource types that users declare in the ResourceTypes parameter, which is currently supported only for AWS CLI and API requests. When using this parameter, users must specify all the resource types that are in their template. For more information about the ResourceTypes parameter, see the CreateStack action in the *AWS CloudFormation API Reference*.

For a list of resource types, see CloudFormation Template Reference Guide.

Use the three-part resource naming convention to specify which resource types users can work with, from all resources across an organization, down to an individual resource type.

organization::*

Specify all resource types for a given organization.

organization::*service_name*::*

Specify all resource types for the specified service within a given organization.

organization::*service_name*::*resource_type*

Specify a specific resource type.

For example:

AWS::*

Specify all supported AWS resource types.

AWS::*service_name*::*

Specify all supported resources for a specific AWS service.

AWS::*service_name*::*resource_type*

Specify a specific AWS resource type, such as AWS::EC2::Instance (all EC2 instances).

Alexa::ASK::*

Specify all resource types in the Alexa Skill Kit.

Alexa::ASK::Skill

Specify the individual Alexa::ASK::Skill resource type.

Custom::*

Specify all custom resources.

For more information, see Create custom provisioning logic with custom resources.

Custom::*resource_type*

Specify a specific custom resource type.

For more information, see Create custom provisioning logic with custom resources.

cloudformation:RoleARN

The Amazon Resource Name (ARN) of an IAM service role that you want to associate with a policy. Use this condition to control which service role users can use when they work with stacks or change sets.

`cloudformation:StackPolicyUrl`

An Amazon S3 stack policy URL that you want to associate with a policy. Use this condition to control which stack policies users can associate with a stack during a create or update stack action. For more information about stack policies, see Prevent updates to stack resources.

> **Note**
>
> To ensure that users can only create or update stacks with the stack policies that you uploaded, set the S3 bucket to read only for those users.

`cloudformation:TemplateUrl`

An Amazon S3 template URL that you want to associate with a policy. Use this condition to control which templates users can use when they create or update stacks.

> **Note**
>
> To ensure that users can only create or update stacks with the templates that you uploaded, set the S3 bucket to read only for those users.

> **Note**
>
> The following CloudFormation-specific conditions apply to the API parameters of the same name:
>
> - `cloudformation:ChangeSetName`
> - `cloudformation:RoleARN`
> - `cloudformation:StackPolicyUrl`
> - `cloudformation:TemplateUrl`
>
> For example, `cloudformation:TemplateUrl` only applies to the `TemplateUrl` parameter for `CreateStack`, `UpdateStack`, and `CreateChangeSet` APIs.

For examples of IAM policies that use condition keys to control access, see Example IAM identity-based policies for CloudFormation.

Acknowledging IAM resources in CloudFormation templates

Before you can create a stack, CloudFormation validates your template. During validation, CloudFormation checks your template for IAM resources that it might create. IAM resources, such as a user with full access, can access and modify any resource in your AWS account. Therefore, we suggest that you review the permissions associated with each IAM resource before proceeding so that you don't unintentionally create resources with escalated permissions. To ensure that you've done so, you must acknowledge that the template contains those resources, giving CloudFormation the specified capabilities before it creates the stack.

You can acknowledge the capabilities of CloudFormation templates by using the CloudFormation console, AWS Command Line Interface (AWS CLI), or API:

- In the CloudFormation console, on the **Configure stack options** page of the Create Stack or Update Stack wizards, choose **I acknowledge that this template may create IAM resources**.
- In the AWS CLI, when you use the create-stack and update-stack commands, specify the `CAPABILITY_IAM` or `CAPABILITY_NAMED_IAM` value for the `--capabilities` option. If your template includes IAM resources, you can specify either capability. If your template includes custom names for IAM resources, you must specify `CAPABILITY_NAMED_IAM`.
- In the API, when you use the CreateStack and UpdateStack actions, specify `Capabilities.member.1=CAPABILITY_IAM` or `Capabilities.member.1=CAPABILITY_NAMED_IAM`. If your template includes IAM resources, you can specify either capability. If your template includes custom names for IAM resources, you must specify `CAPABILITY_NAMED_IAM`.

> **Important**
>
> If your template contains custom named IAM resources, don't create multiple stacks reusing the same template. IAM resources must be globally unique within your account. If you use the same template to create multiple stacks in different Regions, your stacks might share the same IAM resources, rather than each having a unique one. Shared resources among stacks can have unintended consequences from which you can't recover. For example, if you delete or update shared IAM resources in one stack, you will unintentionally modify the resources of other stacks.

Managing credentials for applications running on Amazon EC2 instances

If you have an application that runs on an Amazon EC2 instance and needs to make requests to AWS resources such as Amazon S3 buckets or an DynamoDB table, the application requires AWS security credentials. However, distributing and embedding long-term security credentials in every instance that you launch is a challenge and a potential security risk. Instead of using long-term credentials, like IAM user credentials, we recommend that you create an IAM role that is associated with an Amazon EC2 instance when the instance is launched. An application can then get temporary security credentials from the Amazon EC2 instance. You don't have to embed long-term credentials on the instance. Also, to make managing credentials easier, you can specify just a single role for multiple Amazon EC2 instances; you don't have to create unique credentials for each instance.

For a template snippet that shows how to launch an instance with a role, see IAM role template examples.

> **Note**
>
> Applications on instances that use temporary security credentials can call any CloudFormation actions. However, because CloudFormation interacts with many other AWS services, you must verify that all the services that you want to use support temporary security credentials. For a list of the services that accept temporary security credentials, see AWS services that work with IAM in the *IAM User Guide*.

Granting temporary access (federated access)

In some cases, you might want to grant users with no AWS credentials temporary access to your AWS account. Rather than creating and deleting long-term credentials whenever you want to grant temporary access, use AWS Security Token Service (AWS STS). For example, you can use IAM roles. From one IAM role, you can programmatically create and then distribute many temporary security credentials (which include an access key, secret access key, and security token). These credentials have a limited life, so they cannot be used to access your AWS account after they expire. You can also create multiple IAM roles in order to grant individual users different levels of permissions. IAM roles are useful for scenarios like federated identities and single sign-on.

A federated identity is a distinct identity that you can use across multiple systems. For enterprise users with an established on-premises identity system (such as LDAP or Active Directory), you can handle all authentication with your on-premises identity system. After a user has been authenticated, you provide temporary security credentials from the appropriate IAM user or role. For example, you can create an administrators role and a developers role, where administrators have full access to the AWS account and developers have permissions to work only with CloudFormation stacks. After an administrator is authenticated, the administrator is authorized to obtain temporary security credentials from the administrators role. However, for developers, they can obtain temporary security credentials from only the developers role.

You can also grant federated users access to the AWS Management Console. After users authenticate with your on-premises identity system, you can programmatically construct a temporary URL that gives direct access to the AWS Management Console. When users use the temporary URL, they won't need to sign in to AWS because they have already been authenticated (single sign-on). Also, because the URL is constructed from the users' temporary security credentials, the permissions that are available with those credentials determine what permissions users have in the AWS Management Console.

You can use several different AWS STS APIs to generate temporary security credentials. For more information about which API to use, see Compare AWS STS credentials in the *IAM User Guide*.

> **Important**
>
> You cannot work with IAM when you use temporary security credentials that were generated from the `GetFederationToken` API. Instead, if you need to work with IAM, use temporary security credentials from a role.

CloudFormation interacts with many other AWS services. When you use temporary security credentials with CloudFormation, verify that all the services that you want to use support temporary security credentials. For a list of the services that accept temporary security credentials, see AWS services that work with IAM in the *IAM User Guide*.

For more information, see the following related resources in the *IAM User Guide*:

- Common scenarios for temporary credentials
- Enable custom identity broker access to the AWS console

Example IAM identity-based policies for CloudFormation

By default, users and roles don't have permission to create or modify CloudFormation resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles. For more information, see Defining IAM identity-based policies for CloudFormation.

The following examples show policy statements that you could use to allow or deny permissions to use one or more CloudFormation actions.

Topics

- Require a specific template URL
- Deny all CloudFormation import operations
- Allow import operations for specific resource types
- Deny IAM resources in stack templates
- Allow stack creation with specific resource types
- Control access based on resource-mutating API actions
- Restrict stack set operations based on Region and resource types
- Allow all IaC generator operations

Require a specific template URL

The following policy grants permissions to use only the *https://s3.amazonaws.com/amzn-s3-demo-bucket/test.template* template URL to create or update a stack.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudformation:TemplateUrl": [
                        "https://s3.amazonaws.com/amzn-s3-demo-bucket/
test.template"
                    ]
                }
            }
        }
    ]
}
```

Deny all CloudFormation import operations

The following policy grants permissions to complete all CloudFormation operations except import operations.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllStackOperations",
      "Effect": "Allow",
      "Action": "cloudformation:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyImport",
      "Effect": "Deny",
      "Action": "cloudformation:*",
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "cloudformation:ImportResourceTypes": [
            "*"
          ]
        }
```

```
      }
    }
  ]
}
```

Allow import operations for specific resource types

The following policy grants permissions to all stack operations, in addition to import operations only on specified resources (in this example, `AWS::S3::Bucket`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowImport",
      "Effect": "Allow",
      "Action": "cloudformation:*",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEqualsIgnoreCase": {
          "cloudformation:ImportResourceTypes": [
            "AWS::S3::Bucket"
          ]
        }
      }
    }
  ]
}
```

Deny IAM resources in stack templates

The following policy grants permissions to create stacks but denies requests if the stack's template include any resource from the IAM service. The policy also requires users to specify the `ResourceTypes` parameter, which is available only for AWS CLI and API requests. This policy uses explicit deny statements so that if any other policy grants additional permissions, this policy always remain in effect (an explicit deny statement always overrides an explicit allow statement).

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect" : "Allow",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*"
    },
    {
      "Effect" : "Deny",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*",
      "Condition" : {
        "ForAnyValue:StringLikeIfExists" : {
          "cloudformation:ResourceTypes" : [ "AWS::IAM::*" ]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "cloudformation:ResourceTypes": "true"
        }
      }
    }
  ]
}
```

Allow stack creation with specific resource types

The following policy is similar to the previous example. The policy grants permissions to create a stack unless the stack's template includes any resource from the IAM service. It also requires users to specify the `ResourceTypes` parameter, which is available only for AWS CLI and API requests. This policy is simpler, but it doesn't use explicit deny statements. Other policies, granting additional permissions, could override this policy.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect" : "Allow",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*",
      "Condition" : {
        "ForAllValues:StringNotLikeIfExists" : {
          "cloudformation:ResourceTypes" : [ "AWS::IAM::*" ]
        },
        "Null":{
          "cloudformation:ResourceTypes": "false"
        }
      }
    }
  ]
}
```

Control access based on resource-mutating API actions

The following policy grants permissions to filter access by the name of a resource-mutating API action. This is used to control which APIs IAM users can use to add or remove tags on a stack or stack set. The operation that is used to add or remove tags should be added as value for the condition key. The following policy grants `TagResource` and `UntagResource` permissions to mutating operation `CreateStack`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "CreateActionConditionPolicyForTagUntagResources",
        "Effect": "Allow",
        "Action": [
            "cloudformation:TagResource",
            "cloudformation:UntagResource"
        ],
```

```
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "cloudformation:CreateAction": [
                    "CreateStack"
                ]
            }
        }
    }]
}
```

Restrict stack set operations based on Region and resource types

The following policy grants service-managed stack set permissions. A user with this policy can only perform operations on stack sets with templates containing Amazon S3 resource types (`AWS::S3::*`) or the `AWS::SES::ConfigurationSet` resource type. If signed in to the organization management account with ID `123456789012`, the user can also only perform operations on stack sets that target the OU with ID *`ou-1fsfsrsdsfrewr`*, and can only perform operations on the stack set with ID `stack-set-id` that targets the AWS account with ID *`987654321012`*.

Stack set operations fail if the stack set template contains resource types other than those specified in the policy, or if the deployment targets are OU or account IDs other than those specified in the policy for the corresponding management accounts and stack sets.

These policy restrictions only apply when stack set operations target the `us-east-1`, `us-west-2`, or `eu-west-2` AWS Regions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": [
                "arn:aws:cloudformation:*:*:stackset/*",
```

```
                "arn:aws:cloudformation:*:*:type/resource/AWS-S3-*",
                "arn:aws:cloudformation:us-west-2:111122223333:type/resource/AWS-
SES-ConfigurationSet",
                "arn:aws:cloudformation:*:111122223333:stackset-target/
*/ou-1fsfsrsdsfrewr",
                "arn:aws:cloudformation:*:111122223333:stackset-target/stack-set-
id/444455556666"
            ],
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudformation:TargetRegion": [
                        "us-east-1",
                        "us-west-2",
                        "eu-west-1"
                    ]
                }
            }
        }
    ]
}
```

Allow all IaC generator operations

The following policy allows access to CloudFormation actions related to IaC generator resource scanning and template management. The first statement grants permissions to describe, list, and start resource scans. It also allows access to additional required permissions (`cloudformation:GetResource`, `cloudformation:ListResources`, and `cloudformation:ListTypes`) that enable the IaC generator to retrieve information about resources and available resource types. The second statement grants full permissions to create, delete, describe, list, and update generated templates.

You must also grant read permissions for the target AWS services to anyone who will scan resources with IaC generator. For more information, see IAM permissions required for scanning resources.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
```

```
        {
            "Sid":"ResourceScanningOperations",
            "Effect":"Allow",
            "Action":[
                "cloudformation:DescribeResourceScan",
                "cloudformation:GetResource",
                "cloudformation:ListResources",
                "cloudformation:ListResourceScanRelatedResources",
                "cloudformation:ListResourceScanResources",
                "cloudformation:ListResourceScans",
                "cloudformation:ListTypes",
                "cloudformation:StartResourceScan"
            ],
            "Resource":"*"
        },
        {
            "Sid":"TemplateGeneration",
            "Effect":"Allow",
            "Action":[
                "cloudformation:CreateGeneratedTemplate",
                "cloudformation:DeleteGeneratedTemplate",
                "cloudformation:DescribeGeneratedTemplate",
                "cloudformation:GetResource",
                "cloudformation:GetGeneratedTemplate",
                "cloudformation:ListGeneratedTemplates",
                "cloudformation:UpdateGeneratedTemplate"
            ],
            "Resource":"*"
        }
    ]
}
```

AWS managed policies for AWS CloudFormation

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you

reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSCloudFormationFullAccess

You can attach `AWSCloudFormationFullAccess` to your users, groups, and roles.

This policy grants permissions that allow full access to CloudFormation actions and resources.

Permissions details

This policy includes the following permissions.

- `cloudformation` – Allows principals to perform all CloudFormation actions on all resources.

To view the permissions for this policy, see AWSCloudFormationFullAccess in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AWSCloudFormationReadOnlyAccess

You can attach `AWSCloudFormationReadOnlyAccess` to your users, groups, and roles.

This policy grants permissions that allow read-only access to CloudFormation resources and actions.

Permissions details

This policy includes the following permissions.

- `cloudformation` – Allows principals to perform read-only CloudFormation actions such as describing stacks, listing resources, and viewing templates, but does not allow creating, updating, or deleting stacks.

To view the permissions for this policy, see AWSCloudFormationReadOnlyAccess in the *AWS Managed Policy Reference Guide*.

CloudFormation updates to AWS managed policies

View details about updates to AWS managed policies for CloudFormation since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the CloudFormation Document history page.

Change	Description	Date
AWSCloudFormationReadOnlyAccess – Update to an existing policy	CloudFormation added new permissions to allow `cloudformation:BatchDescribe*` actions for batch describe operations.	January 30, 2026
AWSCloudFormationReadOnlyAccess – Update to an existing policy	CloudFormation added new permissions to allow `cloudformation:Detect*` actions for stack drift detection capabilities.	November 13, 2019
AWSCloudFormationReadOnlyAccess – Update to an existing policy	CloudFormation added new permissions to allow `cloudformation:EstimateTemplateCost`, `cloudformation:Get*`, and `cloudformation:ValidateTemplate` actions.	November 2, 2017
AWSCloudFormationFullAccess – New policy	CloudFormation added a new AWS managed policy that provides full access to CloudFormation actions and resources.	July 26, 2019

Change	Description	Date
AWSCloudFormationReadOnlyAccess – Update to an existing policy	CloudFormation added new permissions to allow `cloudformation:DetectStackDrift` and `cloudformation:DetectStackResourceDrift` actions for stack drift detection.	February 6, 2019
AWSCloudFormationReadOnlyAccess – New policy	CloudFormation added a new AWS managed policy that provides read-only access to CloudFormation actions and resources.	February 6, 2015
CloudFormation started tracking changes	CloudFormation started tracking changes for its AWS managed policies.	February 6, 2015

CloudFormation service role

A *service role* is an AWS Identity and Access Management (IAM) role that allows CloudFormation to make calls to resources in a stack on your behalf. You can specify an IAM role that allows CloudFormation to create, update, or delete your stack resources. By default, CloudFormation uses a temporary session that it generates from your user credentials for stack operations. If you specify a service role, CloudFormation uses that role's credentials.

Use a service role to explicitly specify the actions that CloudFormation can perform, which might not always be the same actions that you or other users can do. For example, you might have administrative privileges, but you can limit CloudFormation access to only Amazon EC2 actions.

You create the service role and its permission policy with the IAM service. For more information about creating a service role, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*. Specify CloudFormation (`cloudformation.amazonaws.com`) as the service that can assume the role.

To associate a service role with a stack, specify the role when you create the stack. For details, see Configure stack options. You can also change the service role when you update the stack in the console, or DeleteStack the stack through the API. Before you specify a service role, ensure that you have permission to pass it (`iam:PassRole`). The `iam:PassRole` permission specifies which roles you can use. For more information, see Grant a user permissions to pass a role to an AWS service in the *IAM User Guide*.

> **Important**
>
> When you specify a service role, CloudFormation always uses that role for all operations that are performed on that stack. It is not possible to remove a service role attached to a stack after the stack is created. Other users that have permissions to perform operations on this stack are able to use this role, regardless of whether those users have the `iam:PassRole` permission or not. If the role includes permissions that the user shouldn't have, you can unintentionally escalate a user's permissions. Ensure that the role grants least privilege. For more information, see Apply least-privilege permissions in the *IAM User Guide*.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it shouldn't otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the aws:SourceArn and aws:SourceAccount global condition context keys in resource policies to limit the permissions that CloudFormation gives another service to a specific resource, such as a CloudFormation extension. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

Make sure that the value of `aws:SourceArn` is an ARN of the resource that CloudFormation stores.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:cloudformation:*:123456789012:*`.

If the `aws:SourceArn` value doesn't contain the account ID, you must use both global condition context keys to limit permissions.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in CloudFormation to prevent the confused deputy problem.

Example trust policy that uses aws:SourceArn and aws:SourceAccount condition keys

For registry services, CloudFormation makes calls to AWS Security Token Service (AWS STS) to assume a service role in your account. This role is configured for `ExecutionRoleArn` in the RegisterType operation and `LogRoleArn` set in the LoggingConfig operation. For more information, see Configure an execution role with IAM permissions and a trust policy for public extension access.

This example role trust policy uses condition statements to limit the `AssumeRole` capability on the service role to only actions on the specified CloudFormation extension in the specified account. The `aws:SourceArn` and `aws:SourceAccount` conditions are evaluated independently. Any request to use the service role must satisfy both conditions.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "resources.cloudformation.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
```

```
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:cloudformation:us-east-1:123456789012:type/
resource/Organization-Service-Resource"
        }
      }
    }
  ]
}
```

Additional information

For example policies that use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys for a service role used by StackSets, see Set up global keys to mitigate confused deputy problems.

For more information, see Update a role trust policy in the *IAM User Guide*.

Forward access sessions (FAS) requests and permission evaluation

When creating, updating, and deleting CloudFormation stacks, users can optionally specify an IAM role ARN. If no role is provided, CloudFormation uses its default service mechanism to interact with other AWS services. In this scenario, the caller must have the necessary permissions for the resources being managed. Alternatively, when a user supplies their own IAM role, CloudFormation will assume that role to perform service interactions on their behalf.

Regardless of whether the user provides an IAM role, CloudFormation generates a new scoped-down FAS token for each resource operation. Consequently, FAS-related condition keys, including `aws:ViaAWSService`, are populated in both scenarios.

The use of FAS affects how IAM policies are evaluated during CloudFormation operations. When creating a stack with a template that includes resources affected by FAS-related condition keys, permission denials may occur.

Example IAM policy

Consider the following IAM policy. `Statement2` will consistently prevent the creation of an `AWS::KMS::Key` resource in CloudFormation. The restriction will be enforced consistently,

whether or not an IAM role is provided during the stack operation. This is because the `aws:ViaAWSService` condition key is always set to `true` due to the use of FAS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": [
                "kms:CreateKey"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Statement2",
            "Effect": "Deny",
            "Action": [
                "kms:CreateKey"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        }
    ]
}
```

Example stack template

For example, when a user creates a stack with the following example template, `aws:ViaAWSService` is set to `true`, and role permissions will be overridden by the FAS policy.

Stack creation will be affected by `Statement2` of the IAM policy, which denies the `CreateKey` action. This results in a permission denied error.

```
Resources:
  myPrimaryKey:
    Type: AWS::KMS::Key
    Properties:
      Description: An example multi-Region primary key
      KeyPolicy:
        Version: '2012-10-17'
        Id: key-default-1
        Statement:
          - Sid: Enable IAM User Permissions
            Effect: Allow
            Principal:
              AWS: !Join
                - ''
                - - 'arn:aws:iam::'
                  - !Ref AWS::AccountId
                  - ':root'
            Action: kms:*
            Resource: '*'
```

For more information about FAS, see Forward access sessions in the *IAM User Guide*.

> **Note**
>
> Most resources adhere to this behavior. However, if you experience unexpected success or failure when creating, updating, or deleting a resource, and your IAM policy includes FAS-related condition keys, it's likely that the resource in question belongs to a small subset of resources that don't follow this standard pattern.

Logging CloudFormation API calls with AWS CloudTrail

CloudFormation is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CloudFormation. CloudTrail captures all API calls for CloudFormation as events, including calls from the CloudFormation console and from code calls to the CloudFormation APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for CloudFormation. If you don't

configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to CloudFormation, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Topics

- CloudFormation information in CloudTrail
- Understanding CloudFormation log file entries

CloudFormation information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in CloudFormation, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail event history.

For an ongoing record of events in your AWS account, including events for CloudFormation, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions and Receiving CloudTrail log files from multiple accounts

All CloudFormation actions are logged by CloudTrail and are documented in the *AWS CloudFormation API Reference*. For example, calls to the `CreateStack`, `DeleteStack`, and `ListStacks` sections generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding CloudFormation log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateStack` operation. The operation was made by an IAM user named Alice.

> **Note**
>
> Only the input parameter key names are logged. Parameter values aren't logged.

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:02:43Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "CreateStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "templateURL": "templateURL",
```

```
    "tags": [
      {
        "key": "test",
        "value": "tag"
      }
    ],
    "stackName": "my-test-stack",
    "disableRollback": true,
    "parameters": [
      {
        "parameterKey": "password"
      },
      {
        "parameterKey": "securitygroup"
      }
    ]
  },
  "responseElements": {
    "stackId": "arn:aws:cloudformation:us-east-1:012345678910:stack/my-test-stack/
a38e6a60-b397-11e3-b0fc-08002755629e"
  },
  "requestID": "9f960720-b397-11e3-bb75-a5b75389b02d",
  "eventID": "9bf6cfb8-83e1-4589-9a70-b971e727099b"
}
```

The following example shows that Alice called the `UpdateStack` operation on the `my-test-stack` stack:

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:04:29Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "UpdateStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
```

```
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "templateURL": "templateURL",
    "parameters": [
      {
        "parameterKey": "password"
      },
      {
        "parameterKey": "securitygroup"
      }
    ],
    "stackName": "my-test-stack"
  },
  "responseElements": {
    "stackId": "arn:aws:cloudformation:us-east-1:012345678910:stack/my-test-stack/
a38e6a60-b397-11e3-b0fc-08002755629e"
  },
  "requestID": "def0bf5a-b397-11e3-bb75-a5b75389b02d",
  "eventID": "637707ce-e4a3-4af1-8edc-16e37e851b17"
}
```

The following example shows that Alice called the `ListStacks` operation.

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:03:16Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "ListStacks",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "b7d351d7-b397-11e3-bb75-a5b75389b02d",
  "eventID": "918206d0-7281-4629-b778-b91eb0d83ce5"
```

```
}
```

The following example shows that Alice called the `DescribeStacks` operation on the `my-test-stack` stack.

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
  "eventTime": "2014-03-24T21:06:15Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "DescribeStacks",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "stackName": "my-test-stack"
  },
  "responseElements": null,
  "requestID": "224f2586-b398-11e3-bb75-a5b75389b02d",
  "eventID": "9e5b2fc9-1ba8-409b-9c13-587c2ea940e2"
}
```

The following example shows that Alice called the `DeleteStack` operation on the `my-test-stack` stack.

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAABCDEFGHIJKLNMOPQ",
    "arn": "arn:aws:iam::012345678910:user/Alice",
    "accountId": "012345678910",
    "accessKeyId": "AKIDEXAMPLE",
    "userName": "Alice"
  },
```

```
  "eventTime": "2014-03-24T21:07:15Z",
  "eventSource": "cloudformation.amazonaws.com",
  "eventName": "DeleteStack",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-cli/1.2.11 Python/2.7.4 Linux/2.6.18-164.el5",
  "requestParameters": {
    "stackName": "my-test-stack"
  },
  "responseElements": null,
  "requestID": "42dae739-b398-11e3-bb75-a5b75389b02d",
  "eventID": "4965eb38-5705-4942-bb7f-20ebe79aa9aa"
}
```


Infrastructure security in CloudFormation

As a managed service, AWS CloudFormation is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access CloudFormation through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

You can call these API operations from any network location, but CloudFormation does support resource-based access policies, which can include restrictions based on the source IP address. You can also use CloudFormation policies to control access from specific Amazon Virtual Private Cloud (Amazon VPC) endpoints or specific VPCs. Effectively, this isolates network access to a given CloudFormation resource from only the specific VPC within the AWS network.

Resilience in CloudFormation

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with

low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Compliance validation for CloudFormation

Our new AWS sign-up experience is not designed for regulated workloads. If you're using our new AWS sign-up experience, but you want to use AWS for regulated workloads, you can sign up for AWS (advanced) or activate advanced features for your AWS environment.

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Configuration and vulnerability analysis in CloudFormation

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS shared responsibility model.

Security best practices for CloudFormation

CloudFormation provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

Topics

- Use IAM to control access
- Do not embed credentials in your templates
- Use AWS CloudTrail to log CloudFormation calls

Use IAM to control access

IAM is an AWS service that you can use to manage users and their permissions in AWS. You can use IAM with CloudFormation to specify what CloudFormation actions users can perform, such as viewing stack templates, creating stacks, or deleting stacks. Furthermore, anyone managing CloudFormation stacks will require permissions to resources within those stacks. For example, if users want to use CloudFormation to launch, update, or terminate Amazon EC2 instances, they must have permission to call the relevant Amazon EC2 actions.

In most cases, users require full access to manage all of the resources in a template. CloudFormation makes calls to create, modify, and delete those resources on their behalf. To separate permissions between a user and the CloudFormation service, use a service role. CloudFormation uses the service role's policy to make calls instead of the user's policy. For more information, see CloudFormation service role.

Do not embed credentials in your templates

Rather than embedding sensitive information in your CloudFormation templates, we recommend you use *dynamic references* in your stack template.

Dynamic references provide a compact, powerful way for you to reference external values that are stored and managed in other services, such as the AWS Systems Manager Parameter Store or AWS Secrets Manager. When you use a dynamic reference, CloudFormation retrieves the value of the specified reference when necessary during stack and change set operations, and passes the value to the appropriate resource. However, CloudFormation never stores the actual reference value. For more information, see Get values stored in other services using dynamic references.

AWS Secrets Manager helps you to securely encrypt, store, and retrieve credentials for your databases and other services. The AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management.

For more information on defining template parameters, see CloudFormation template Parameters syntax.

Use AWS CloudTrail to log CloudFormation calls

AWS CloudTrail tracks anyone making CloudFormation API calls in your AWS account. API calls are logged whenever anyone uses the CloudFormation API, the CloudFormation console, a back-end console, or CloudFormation AWS CLI commands. Enable logging and specify an Amazon S3 bucket to store the logs. That way, if you ever need to, you can audit who made what CloudFormation call in your account. For more information, see Logging CloudFormation API calls with AWS CloudTrail.

Access CloudFormation using an interface endpoint (AWS PrivateLink)

You can use AWS PrivateLink to create a private connection between your VPC and CloudFormation. You can access CloudFormation as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or Direct Connect connection. Instances in your VPC don't need public IP addresses to access CloudFormation.

You establish this private connection by creating an *interface endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for CloudFormation.

CloudFormation supports making calls to all of its API actions through the interface endpoint.

Considerations for CloudFormation VPC endpoints

Before you set up an interface endpoint, first make sure you have met the prerequisites in the Access an AWS service using an interface VPC endpoint topic in the *AWS PrivateLink Guide*.

The following additional prerequisites and considerations apply when setting up an interface endpoint for CloudFormation:

- If you have resources within your VPC that must respond to a custom resource request or a wait condition, make sure that they have access to the required CloudFormation-specific Amazon S3 buckets. CloudFormation has S3 buckets in each Region to monitor responses to a custom resource request or a wait condition. If a template includes custom resources or wait conditions in a VPC, the VPC endpoint policy must allow users to send responses to the following buckets:

- For custom resources, permit traffic to the `cloudformation-custom-resource-response-region` bucket. When using custom resources, AWS Region names don't contain dashes. For example, `uswest2`.
- For wait conditions, permit traffic to the `cloudformation-waitcondition-region` bucket. When using wait conditions, AWS Region names do contain dashes. For example, `us-west-2`.

  If the endpoint policy blocks traffic to these buckets, CloudFormation won't receive responses and the stack operation fails. For example, if you have a resource in a VPC in the `us-west-2` Region that must respond to a wait condition, the resource must be able to send a response to the `cloudformation-waitcondition-us-west-2` bucket.

  For a list of AWS Regions where CloudFormation is currently available, see the CloudFormation endpoints and quotas page in the *Amazon Web Services General Reference*.
- VPC endpoints currently don't support cross-Region requests — ensure that you create your endpoint in the same Region in which you plan to issue your API calls to CloudFormation.
- VPC endpoints only support Amazon-provided DNS through Route 53. If you want to use your own DNS, you can use conditional DNS forwarding. For more information, see DHCP options sets in Amazon VPC in the Amazon VPC User Guide.
- The security group attached to the VPC endpoint must allow incoming connections on port 443 from the private subnet of the VPC.

Creating an interface VPC endpoint for CloudFormation

You can create a VPC endpoint for CloudFormation using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create a VPC endpoint in the *AWS PrivateLink Guide*.

Create an interface endpoint for CloudFormation using the following service name:

- **com.amazonaws.*region*.cloudformation** – Creates an endpoint for CloudFormation API operations.

If you enable private DNS for the interface endpoint, you can make API requests to CloudFormation using its default Regional DNS name. For example, `cloudformation.us-east-1.amazonaws.com`.

In AWS Regions where FIPS-specific endpoints are supported, you can also create an interface endpoint for CloudFormation using the following service name:

- **com.amazonaws.*region*.cloudformation-fips** – Creates an endpoint for the CloudFormation API that complies with Federal Information Processing Standard (FIPS) 140-2.

For a complete list of CloudFormation endpoints, see CloudFormation endpoints and quotas in the *Amazon Web Services General Reference*.

Creating a VPC endpoint policy for CloudFormation

An endpoint policy is an IAM resource that you can attach to an interface endpoint. The default endpoint policy allows full access to CloudFormation through the interface endpoint. To control the access allowed to CloudFormation from your VPC, attach a custom endpoint policy to the interface endpoint.

An endpoint policy specifies the following information:

- The principals that can perform actions (AWS accounts, IAM users, and IAM roles).
- The actions that can be performed.
- The resources on which the actions can be performed.

For more information, see Control access to VPC endpoints using endpoint policies in the *AWS PrivateLink Guide*.

Example: VPC endpoint policy for CloudFormation actions

The following is an example of an endpoint policy for CloudFormation. When attached to an endpoint, this policy grants access to the listed CloudFormation actions for all principals on all resources. The following example denies all users the permission to create stacks through the VPC endpoint, and allows full access to all other actions on the CloudFormation service.

```
{
  "Statement": [
    {
      "Action": "cloudformation:*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
```

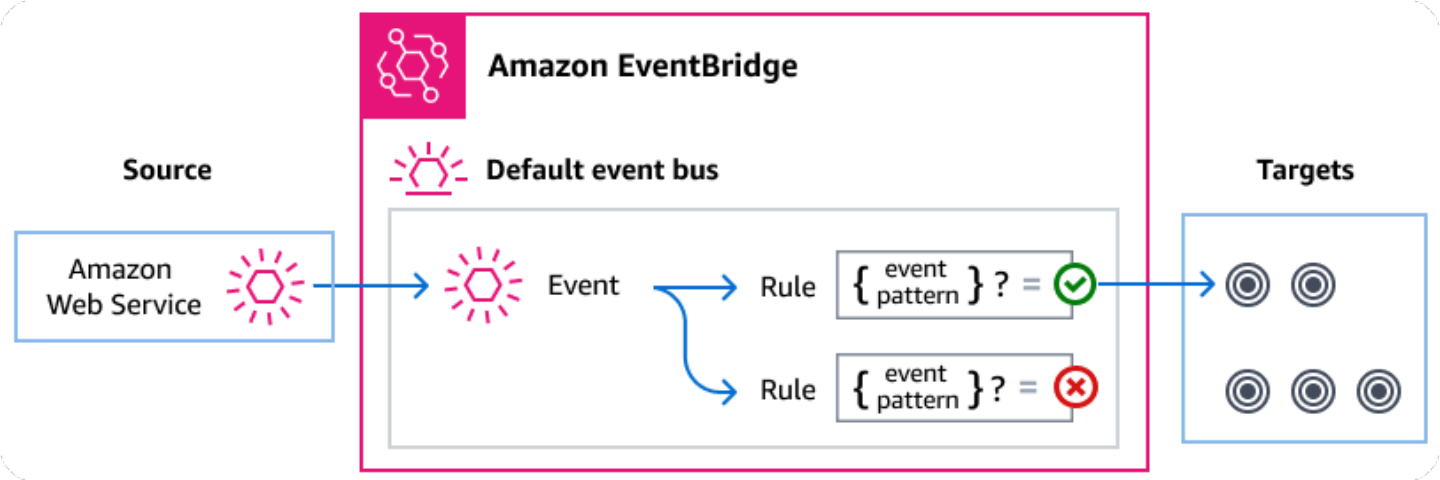
```
    },
    {
      "Action": "cloudformation:CreateStack",
      "Effect": "Deny",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

Monitoring CloudFormation and Git sync events with EventBridge

Amazon EventBridge is a serverless service that uses events to connect application components together, making it easier for you to build scalable event-driven applications. Event-driven architecture is a style of building loosely-coupled software systems that work together by emitting and responding to events. Events represent a change in a resource or environment.

As with many AWS services, CloudFormation generates and sends events to the EventBridge default event bus, which is automatically provisioned in every AWS account. An event bus is a router that receives events and delivers them to zero or more destinations, or *targets*. Rules you specify for the event bus evaluate events as they arrive. Each rule checks whether an event matches the rule's *event pattern*. If the event does match, the event bus sends the event to the specified target(s).

For more information, see Getting started with Amazon EventBridge in the *Amazon EventBridge User Guide*.



Topics

- CloudFormation and Git sync events overview
- Amazon EventBridge permissions
- Creating a custom event pattern for an EventBridge rule
- CloudFormation events detail reference

CloudFormation and Git sync events overview

CloudFormation sends events to EventBridge whenever a create, update, delete, or drift-detection operation is performed on a stack. CloudFormation also sends events to EventBridge for status changes to stack sets and stack set instances. You can use EventBridge rules to route events to your defined targets. These events are guaranteed to be delivered, and they might be delivered out of order.

Since CloudFormation events represent changes to stacks or stack sets and their resources, you can use them to initiate workflows associated with respective events. For example:

- Create stack or stack set specific tags on all resource provisioned through CloudFormation.
- Establish an association between a CloudFormation stack or stack set and an Amazon WorkSpaces Application Manager (Amazon WAM).
- Specify an association with an AppRegistry for the created stack or stack set.

The following events are generated by CloudFormation and sent to the default event bus in EventBridge. For more information, see ???.

Event type	Description	Event source
Resource Status Change	Any updates performed on a stack which changes underlying resource properties. For a complete list of supported AWS resource types, see the AWS resource and property types reference.	AWS CloudFormation
Stack Status Change	Represents a status change to a given stack. For code details, see Stack status codes.	AWS CloudFormation
Drift Detection Status Change	Represents a user-initiated drift detection update on a given stack. For a complete list of fully mutable and immutable types that support drift detection, see Resource type support	AWS CloudFormation

Event type	Description	Event source
StackSet Status Change	Represents a status change to a given stack set.	AWS CloudFormation
StackSet Stack Instance Status Change	Represents a status change to a specific StackSet stack instance. For code details, see Stack instance status codes.	AWS CloudFormation
StackSet operation status	Represents a status change to a given StackSet operation. For code details, see StackSets status codes.	AWS CloudFormation

Additionally, AWS CloudFormation Git sync sends events for status changes for repository syncs and resource syncs to EventBridge.

The following Git sync events are generated by AWS CodeConnections and sent to the default event bus in EventBridge. For more information, see ???.

Event type	Description	Event source
Repository sync status change	Represents a status change to a Git repository sync.	AWS CodeConnections
Resource sync status change	Represents a status change to a Git resource sync.	AWS CodeConnections

Amazon EventBridge permissions

CloudFormation doesn't require any additional permissions to deliver events to EventBridge. The events contain information that's already available through CloudFormation's API operations.

The targets you specify may need specific permissions or configuration. For more details on using specific services for targets, see Amazon EventBridge targets in the *Amazon EventBridge User Guide*.

Creating a custom event pattern for an EventBridge rule

You can find several predefined patterns in EventBridge for CloudFormation and Git sync events. This simplifies how an event pattern is created. Instead of writing your own event patterns, you can select field values on a form, and EventBridge generates the pattern for you. You can create a new rule using one of these predefined event patterns or create your own custom event pattern.

When a service like CloudFormation delivers an event to the default event bus, EventBridge uses the event pattern defined in your rule to determine if the event should be delivered to the rule's target(s). An event pattern matches the data in the desired CloudFormation events.

Each event pattern is a JSON object that contains:

- A `source` attribute that identifies the service the event is coming from. For example, `aws.cloudformation` or `aws.codeconnections`.
- (Optional): A `detail-type` attribute that contains an array of the event types to match.
- (Optional): A `detail` attribute containing any other event data on which to match.

  For example, the stack ID, the resources involved, status of various resources, and other data relevant to a particular type of events.

For example, the following event pattern matches against all resource status change events:

```
{
  "source": ["aws.cloudformation"],
  "detail-type": ["CloudFormation Resource Status Change"]
}
```

While the following event pattern uses event detail data to match only resource status change events where CloudFormation creates a new `AWS::S3::Bucket` or `AWS::SNS::Topic` resource:

```
{
  "source": ["aws.cloudformation"],
  "detail-type": ["CloudFormation Resource Status Change"],
  "detail": {
    "status-details": {
      "status": ["CREATE_COMPLETE"]
    },
    "resource-type": ["AWS::S3::Bucket", "AWS::SNS::Topic"]
```

```
  }
}
```

For more information on writing event patterns, see Event patterns in the *Amazon EventBridge User Guide*.

CloudFormation events detail reference

All events from AWS services have a common set of fields containing metadata about the event, such as the AWS service that's the source of the event, the time the event was generated, the account and region in which the event took place, and others. For definitions of these general fields, see AWS service event metadata in the *AWS Events Reference*.

In addition, each event has a `detail` field that contains data specific to that particular event. The reference below defines the detail fields for the various CloudFormation events.

When using EventBridge to select and manage CloudFormation events, it's useful to keep the following in mind:

- The `source` field specifies the event source.

  For example, `aws.cloudformation` or `aws.codeconnections`.
- The `detail-type` field specifies the event type.

  For example, `CloudFormation Resource Status Change` or `CloudFormation Drift Detection Status Change`.
- The `detail` field contains the data that is specific to that particular event.

  For example, the stack ID, the resources involved, status of various resources, and other data relevant to a particular type of events.

For information on constructing event patterns that enable rules to match CloudFormation events, see Event patterns in the *Amazon EventBridge User Guide*.

For more information on events and how EventBridge processes them, see EventBridge events in the *Amazon EventBridge User Guide*.

Topics

- Resource Status Change event detail
- Stack Status Change event detail
- Drift Detection Status Change event detail
- StackSet Status Change event detail
- StackSet Stack Instance Status Change event detail
- StackSet Operation Status Change event detail
- Repository Sync Status Change event detail
- Resource Sync Status Change event detail

Resource Status Change event detail

Below are the detail fields for Resource Status Change events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type": "CloudFormation Resource Status Change",
  "source": "aws.cloudformation",
  . . .,
  "detail": {
    "stack-id" : "string",
    "logical-resource-id" : "string",
    "physical-resource-id": "string",
    "status-details": {
        "status": "string",
        "status-reason": "string"
    },
     "resource-type": "string",
     "client-request-token": "string"
  }
}
```

`detail-type`

Identifies the type of event.

For resource status events, this value is `CloudFormation Resource Status Change`.

`source`

Identifies the service that generated the event. For CloudFormation events, this value is `aws.cloudformation`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For resource status events, this data includes:

`stack-id`

The unique stack ID that's associated with the stack.

`logical-resource-id`

The logical name of the resource as specified in the template.

`physical-resource-id`

The name or unique identifier that corresponds to a physical instance ID of a resource supported by CloudFormation.

`status-details`

`status`

Status of the resource.

`status-reason`

Status reason of the resource.

`resource-type`

Type of resource. For example, `AWS::S3::Bucket`.

`client-request-token`

An access token used to call the API. All events that are initiated by a given stack operation are assigned the same client request token, which you can use to track operations. Stack operations that are initiated from the console use the token format *Console-StackOperation-ID*, which helps you to easily identify the stack operation. For example, if you create a stack using the console, each resulting stack event would be assigned the same token in the following format: `Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002`.

Example: Resource Status Change event

The following is an example resource status event. This event details that CloudFormation has successfully created the requested resource, an Amazon S3 bucket, in the specified stack.

```
{
    "version":"0",
    "id":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type":"CloudFormation Resource Status Change",
    "source":"aws.cloudformation",
    "account":"111122223333",
    "time":"2017-12-22T18:43:48Z",
    "region":"us-west-1",
    "resources":[
        "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack"
    ],
    "detail":{
        "stack-id":"arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
        "logical-resource-id":"my-s3-bucket",
        "physical-resource-id":"arn:aws:s3:::my-s3-bucket-us-east-1",
        "status-details":{
            "status":"CREATE_COMPLETE",
            "status-reason":""
        },
        "resource-type":"AWS::S3::Bucket",
        "client-request-token":""
    }
}
```

Stack Status Change event detail

Below are the detail fields for Stack Status Change events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type":"CloudFormation Stack Status Change",
  "source":"aws.cloudformation",
  . . .,
  "detail":{
    "stack-id":"string",
    "status-details":{
```

```
      "status":"string",
      "status-reason":"string"
    },
    "client-request-token":"string"
  }
}
```

`detail-type`

Identifies the type of event.

For stack status events, this value is `CloudFormation Stack Status Change`.

`source`

Identifies the service that generated the event. For CloudFormation events, this value is `aws.cloudformation`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For stack status events, this data includes:

`stack-id`

The unique stack ID associated with the stack.

`status-details`

`status`

Status of the stack.

For a complete list of stack status codes, see Stack status codes.

`status-reason`

Status reason of the resource.

`client-request-token`

An access token used to call the API. All events that are initiated by a given stack operation are assigned the same client request token, which you can use to track operations. Stack operations that are initiated from the console use the token format *Console-StackOperation-ID*, which helps you to easily identify the stack operation. For example, if you create a

stack using the console, each resulting stack event would be assigned the same token in the following format: `Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002`.

Example: Stack Status event

The following is an example stack status event, where CloudFormation has successfully created the requested stack, `teststack`.

```
{
    "version":"0",
    "id":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type":"CloudFormation Stack Status Change",
    "source":"aws.cloudformation",
    "account":"111122223333",
    "time":"2017-12-22T18:43:48Z",
    "region":"us-west-1",
    "resources":[
        "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack"
    ],
    "detail":{
        "stack-id":"arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
        "status-details":{
            "status":"CREATE_COMPLETE",
            "status-reason":""
        },
        "client-request-token":""
    }
}
```

Drift Detection Status Change event detail

Below are the detail fields for stack drift detection events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type":"CloudFormation Drift Detection Status Change",
  "source":"aws.cloudformation",
  . . .,
```

```
  "detail":{
    "stack-id":"string",
    "stack-drift-detection-id":"string",
    "status-details":{
      "stack-drift-status":"string",
      "detection-status":"string"
    },
      "drift-detection-details":{
        "drifted-stack-resource-count":integer
      },
    "client-request-token":"string"
  }
}
```

`detail-type`

Identifies the type of event.

For stack drift detection events, this value is `CloudFormation Drift Detection Status Change`.

`source`

Identifies the service that generated the event. For CloudFormation events, this value is `aws.cloudformation`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For stack drift detection events, this data includes:

`stack-id`

The unique stack ID that's associated with the stack.

`stack-drift-detection-id`

The stack drift detection Id.

`status-details`

`stack-drift-status`

Drift status of the stack.

`detection-status`

Status of drift detection operation.

`drift-detection-details`

`drifted-stack-resource-count`

Number of resources drifted. When the value is `-1`, the drift detection is in progress. All other non--negative integers represent the actual number of drifted resources.

`client-request-token`

An access token used to call the API. All events that are initiated by a given stack operation are assigned the same client request token, which you can use to track operations. Stack operations that are initiated from the console use the token format *Console-StackOperation-ID*, which helps you to easily identify the stack operation. For example, if you create a stack using the console, each resulting stack event would be assigned the same token in the following format: `Console-CreateStack-7f59c3cf-00d2-40c7-b2ff-e75db0987002`.

Example: Stack Drift Detection event

The following is an example Stack Drift Detection event. This event details that CloudFormation has completed drift detection on the specified stack, and that the stack currently has a drift status of `DRIFTED` due to one drifted resource.

```
{
    "version":"0",
    "id":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type":"CloudFormation Drift Detection Status Change",
    "source":"aws.cloudformation",
    "account":"111122223333",
    "time":"2017-12-22T18:43:48Z",
    "region":"us-west-1",
    "resources": ["string"],
    "detail":{
        "stack-id":"arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
        "stack-drift-detection-id":"624af370-311a-11e8-b6b7-500cexample",
        "status-details":{
            "stack-drift-status":"DRIFTED",
            "detection-status":"DETECTION_COMPLETE"
        },
```

```
        "drift-detection-details":{
            "drifted-stack-resource-count":1
        },
    "client-request-token":""
    }
}
```

StackSet Status Change event detail

Below are the detail fields for StackSet Status Change events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type": "CloudFormation StackSet Status Change",
  "source": "aws.cloudformation",
  . . .,
  "detail": {
    "stack-set-arn" : "string",
    "status-details": {
        "status":"string"
    }
  }
}
```

`detail-type`

Identifies the type of event.

For StackSet status event events, this value is `CloudFormation StackSet Status Change`.

`source`

Identifies the service that generated the event. For CloudFormation events, this value is `aws.cloudformation`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For StackSet status event events, this data includes:

`stack-set-arn`

The Amazon Resource Name (ARN) associated with the stack set.

`status-details`

`status`

The StackSet status.

Valid values: `ACTIVE` | `DELETED`

Example: StackSet Status Change event

The following is an example StackSet Status Change event. This event details that CloudFormation has deleted the specified stack set.

```
{
  "version": "0",
  "id": "42h6hb90-hg0w-11op-b01v-0xhnh0934z09",
  "detail-type": "CloudFormation StackSet Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-23T17:06:18Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test12345:3f3a3fbe-
c937-4eb3-a87d-e36a0af3f663"
  ],
  "detail": {
    "stack-set-arn" : "arn:aws:cloudformation:us-east-1:111122223333:stackset/
test12345:3f3a3fbe-c937-4eb3-a87d-e36a0af3f663",
    "status-details": {
        "status":"DELETED"
    }
  }
}
```

StackSet Stack Instance Status Change event detail

Below are the detail fields for StackSet stack instance status events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type": "CloudFormation StackSet StackInstance Status Change",
  "source": "aws.cloudformation",
  . . .,
  "detail": {
    "stack-set-arn" : "string",
    "stack-id" : "string",
    "action" : "string",
    "status-details": {
        "status": "string",
        "status-reason": "string",
        "detailed-status": "string"
      }
    }
  }
}
```

`detail-type`

Identifies the type of event.

For StackSet stack instance status events, this value is `CloudFormation StackSet StackInstance Status Change`.

`source`

Identifies the service that generated the event. For CloudFormation events, this value is `aws.cloudformation`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For StackSet stack instance status events, this data includes:

`stack-set-arn`

The Amazon Resource Name (ARN) associated with the StackSet.

`stack-id`

The unique stack ID that's associated with the stack instance.

`action`

The type of stack set operation.

Valid values: `CREATE` | `UPDATE`| `DELETE` | `DETECT_DRIFT`

`status-details`

`status`

The StackSet instance status.

For more details, see Stack instance status codes.

Valid values: `CURRENT` | `OUTDATED`| `INOPERABLE`

`status-reason`

Status reason of the StackSet instance.

`detailed-status`

The detailed StackSet instance detailed status.

Valid values: `CANCELLED` | `FAILED` | `FAILED_IMPORT` | `INOPERABLE` | `PENDING` | `RUNNING` | `SKIPPED_SUSPENDED_ACCOUNT` | `SUCCEEDED`

Example: StackSet Stack Instance Status Change event

The following is an example StackSet Stack Instance Status Change event.

```
{
  "version": "0",
  "id": "42h6hb90-hg0w-11op-b01v-0xhnh0934z09",
  "detail-type": "CloudFormation StackSet StackInstance Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-22T19:19:23Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-
d041-44ad-94f8-b8268aca1e59"
  ],
  "detail": {
```

```
     "stack-set-arn": "arn:aws:cloudformation:us-east-1:111122223333:stackset/
test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59",
    "stack-id": "arn:aws:cloudformation:us-west-1:111122223333:stack/teststack",
    "status-details": {
        "status": "OUTDATED",
        "status-reason": "User Initiated",
        "detailed-status": "PENDING"
    }
  }
}
```

StackSet Operation Status Change event detail

Below are the detail fields for StackSet Operation Status Change events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type": "CloudFormation StackSet Operation Status Change",
  "source": "aws.cloudformation",
  . . .,
  "detail": {
    "stack-set-arn" : "string",
    "stack-set-operation-id" : "string",
    "status-details": {
        "status": "string"
      }
    }
  }
}
```

`detail-type`

Identifies the type of event.

For StackSet operation status events, this value is `CloudFormation StackSet Operation Status Change`.

`source`

Identifies the service that generated the event. For CloudFormation events, this value is `aws.cloudformation`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For StackSet operation status events, this data includes:

`stack-set-arn`

The Amazon Resource Name (ARN) associated with the StackSet.

`stack-set-operation-id`

The unique ID that's associated with the StackSet operation.

`status-details`

`status`

The StackSet operation status.

For more details, see StackSets status codes.

Valid values: `RUNNING` | `SUCCEEDED` | `FAILED` | `STOPPING` | `STOPPED` | `QUEUED`

Example: StackSet Operation Status Change event

The following is an example StackSet Operation Status Change event. The event details that CloudFormation has successfully completed the requested operation on the specified stack set.

```
{
  "version": "0",
  "id": "4de89905-fd92-6a6b-9509-23c04bcb6a21",
  "detail-type": "CloudFormation StackSet Operation Status Change",
  "source": "aws.cloudformation",
  "account": "111122223333",
  "time": "2021-09-22T05:46:24Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudformation:us-east-1:111122223333:stackset/test1234:e5f54eea-
d041-44ad-94f8-b8268aca1e59"
  ],
  "detail": {
    "stack-set-arn": "arn:aws:cloudformation:us-east-1:111122223333:stackset/
test1234:e5f54eea-d041-44ad-94f8-b8268aca1e59",
```

```
    "stack-set-operation-id": "ce69adce-2221-4483-8c4b-c51f284f25e8",
    "status-details": {
        "status": "SUCCEEDED"
    }
  }
}
```

Repository Sync Status Change event detail

Below are the detail fields for Repository Sync Status Change events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type": "Git Sync Repository Sync Status Change",
  "source": "aws.codeconnections",
  . . .,
  "detail": {
    "connectionArn" : "string",
    "providerType" : "string",
    "repositoryName": "string",
    "providerType": "string",
    "repositoryName": "string",
    "repositoryArn": "string",
    "repositoryLinkId": "string",
    "ownerId": "string",
    "commit": "string",
    "branch": "string",
    "syncType": "string",
    "status": "string",
    "previousSync": "string"
    }
  }
}
```

`detail-type`

Identifies the type of event.

For Repository Sync status events, this value is `Git Sync Repository Sync Status Change`.

`source`

Identifies the service that generated the event. For Git sync events, this value is `aws.codeconnections`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For Repository sync status events, this data includes:

`connectionArn`

The Amazon Resource Name (ARN) associated with AWS CodeConnections.

`providerType`

The Git provider connected to CloudFormation.

Valid values: `GitHub` | `GitHub Enterprise` | `GitLab` | `BitBucket`

`repositoryName`

The Git repository name.

`repositoryArn`

The ARN associated with the Git repository.

`repositoryLinkId`

The unique ID associated with repository link.

`ownerId`

The unique ID associated with repository owner.

`commit`

The unique ID associated with the repository commit.

`branch`

The unique ID associated with the repository branch.

`syncType`

The type of sync being performed.

`status`

The current repository sync status.

Valid values: `FAILED` | `INITIATED` | `IN_PROGRESS` | `SUCCEEDED`

`previousSync`

The sync status previous to the current status.

Valid values: `FAILED` | `INITIATED` | `IN_PROGRESS` | `SUCCEEDED`

Example: Repository Sync Status Change event

The following is an example Repository Sync Status Change event. The event details that AWS CodeConnections has successfully synchronized the repository.

```
{
  "version": "0",
  "id": "1b5d8feb-agbv-4cf7-a9f1-bf3703467718",
  "detail-type": "GitSync Repository Sync Status Change",
  "source": "aws.codeconnections",
  "account": "111122223333",
  "time": "2023-12-22T18:43:48Z",
  "region": "us-east-1",
  "resources": ["arn:aws:aws:codestar-connections:us-east-1:111122223333:repository-
link/550e8400-e29b-41d4-a716-446655440000",],
  "detail": {
    "connectionArn": "arn:aws:codestar-connections:us-east-1:111122223333:connection/
sample-connection-id",
    "providerType": "GitHub",
    "repositoryName": "sample-repository-name",
    "repositoryArn": "arn:aws:aws:codestar-connections:us-
east-1:111122223333:repository-link/550e8400-e29b-41d4-a716-446655440000"
    "repositoryLinkId": "550e8400-e29b-41d4-a716-446655440000"
    "ownerId": "sample-owner-id",
    "commit": "sample-commit-id",
    "branch": "main",
    "syncType": "CFN_STACK_SYNC",
    "status": "SUCCEEDED",
    "previousStatus": "IN_PROGRESS",
  }
}
```

Resource Sync Status Change event detail

Below are the detail fields for Resource Sync Status Change events.

The `source` and `detail-type` fields are included because they contain specific values for events.

```
{
  . . .,
  "detail-type": "Git Sync Resource Sync Status Change",
  "source": "aws.codeconnections",
  . . .,
  "detail": {
    "providerType" : "string",
    "commit" : "string",
    "repositoryName": "string",
    "branch": "string",
    "syncType": "string",
    "syncTarget": "string",
    "status": "string",
    "previousSync": "string"
  }
}
```

`detail-type`

Identifies the type of event.

For Repository Sync status events, this value is `Git Sync Repository Sync Status Change`.

`source`

Identifies the service that generated the event. For Git sync events, this value is `aws.codeconnections`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For resource sync status events, this data includes:

`providerType`

The Git provider connected to CloudFormation.

Valid values: `GitHub` | `GitHub Enterprise` | `GitLab` | `BitBucket`

`commit`

The unique ID associated with the repository commit.

`repositoryName`

The Git repository name.

`branch`

The unique ID associated with the repository branch.

`syncType`

The type of sync being performed.

`syncTarget`

The target stack for the resource sync.

`status`

The current repository sync status.

Valid values: `FAILED` | `INITIATED` | `IN_PROGRESS` | `SUCCEEDED`

`previousSync`

The sync status previous to the current status.

Valid values: `FAILED` | `INITIATED` | `IN_PROGRESS` | `SUCCEEDED`

Example: Resource Sync Status Change event

The following is an example resource sync status change event. The event details that AWS CodeConnections has successfully synchronized the resource.

```
{
  "version": "0",
  "id": "1b5d8feb-agbv-4cf7-a9f1-bf3703467718",
  "detail-type": "Git Sync Resource Sync Status Change",
  "source": "aws.codeconnections",
  "account": "111122223333",
  "time": "2023-12-22T18:43:48Z",
  "region": "us-east-1",
```

```
  "resources": ["arn:aws:aws:cloudformation:us-east-1:111122223333:stack/
targetStack1"],
  "detail": {
    "providerType": "GitHub",
    "commit": "sample-commit-id",
    "repositoryName": "sample-repository-name",
    "branch": "main",
    "syncType": "CFN_STACK_SYNC",
    "syncTarget": "arn:aws:aws:cloudformation:us-east-1:111122223333:stack/
targetStack1",
    "status": "SUCCEEDED",
    "previousStatus": "IN_PROGRESS"
  }
}
```

Understand CloudFormation quotas

Your AWS account has CloudFormation quotas that you might need to know when authoring templates and creating stacks. By understanding these quotas, you can avoid limitation errors that would require you to redesign your templates or stacks.

The following table shows the CloudFormation quotas.

Quotas	Description	Value	Tuning strategy
cfn-signal wait condition data	The maximum amount of data that `cfn-signal` can pass.	4,096 bytes	To pass a larger amount, send the data to an Amazon S3 bucket, and then use `cfn-signal` to pass the Amazon S3 URL to that bucket.
Custom resource response	The maximum amount of data that a custom resource provider can pass.	4,096 bytes	
Dynamic references per template	The maximum number of dynamic references allowed in a single CloudForm ation stack template.	60 dynamic reference s in a stack template	
Hooks per account	The maximum number of Hooks allowed per account per Region.	100 hooks	
Hooks per resource	The maximum number of Hooks	100 hooks	

Quotas	Description	Value	Tuning strategy
	that can be configure d per resource.		
Hook configuration size	The maximum amount of configura tion data a Hook can store.	204.8 KB	
Mappings	The maximum number of mappings that you can declare in your CloudForm ation template.	200 mappings	To specify more mappings, separate your template into multiple templates by using, for example, nested stacks.
Mapping attributes	The maximum number of mapping attributes for each mapping that you can declare in your CloudFormation template.	200 attributes	To specify more mapping attributes, separate the attribute s into multiple mappings.
Mapping name and mapping attribute name	The maximum size of each mapping name.	255 characters	
Modules	The maximum number of modules you can register in the CloudFormation registry, per account and Region.	100 modules	

Quotas	Description	Value	Tuning strategy
Module versions	The maximum number of versions you can register in the CloudFormation registry for a given module.	100 versions	To register new versions, first use DeregisterType to deregister versions you aren't using anymore.
Nested stacks	The maximum number of CloudFormation resources a nested stack can create, update, or delete per operation. For example, you can have a nested stack hierarchy with more than 2500 total resources, but you can't create, update, or delete more than 2500 of those resources in a single deployment.	2500 resources	Split the stack hierarchy into different stacks.
Outputs	The maximum number of outputs that you can declare in your CloudForm ation template.	200 outputs	
Output name	The maximum size of an output name.	255 characters	

Quotas	Description	Value	Tuning strategy
Parameters	The maximum number of parameters that you can declare in your CloudFormation template.	200 parameters	To specify more parameters, you can use mappings or lists in order to assign multiple values to a single parameter.
Parameter name	The maximum size of a parameter name.	255 characters	
Parameter value	The maximum size of a parameter value.	4,096 bytes	To use a larger parameterized value, create multiple parameters and then use the `Fn::Join` function to concatenate the multiple values into a single value.
Private resources	The maximum number of private resources that you can register in the CloudFormation registry per account, per Region.	50 private resources	
Private resource versions	The maximum number of versions that you can register in the CloudFormation registry for a given private resource.	50 private resources	To register new versions, first use DeregisterType to deregister versions you aren't using anymore.

Quotas	Description	Value	Tuning strategy
Resources	The maximum number of resources that you can declare in your CloudFormation template.	500 resources	To specify more resources, separate your template into multiple templates by using, for example, nested stacks.
Resources in concurrent stack operations	The maximum number of resources you can have involved in stack operations (create, update, or delete operations) in your Region at a given time.	Use the DescribeAccountLimits API to determine the current limit for an account in a specific Region.	
Resource name	The maximum size of a resource name.	255 characters	
Stacks	The maximum number of CloudFormation stacks that you can create.	2000 stacks	To create more stacks, delete stacks that you don't need or request an increase in the maximum number of stacks in your AWS account. For more information, see AWS service quotas in the *AWS General Reference*.
Stack name	The maximum size of a stack name.	128 characters	

Quotas	Description	Value	Tuning strategy
StackSets	The maximum number of CloudFormation stack sets you can create in your administrator account.	1000 stack sets	To create more stack sets, delete stack sets that you don't need or request an increase in the maximum number of stack sets in your AWS account. For more informati on, see AWS service quotas in the *AWS General Reference*.
Stack instances	The maximum number of stack instances you can create per stack set.	100,000 stack instances per stack set	To create more stack instances, delete stack instances that you don't need or request an increase in the maximum number of stack instances in your AWS account. For more information, see AWS service quotas in the *AWS General Reference*.

Quotas	Description	Value	Tuning strategy
StackSets instance operations	The maximum number of stack instances, across all stack sets, that you can run operations on in each Region at the same time, per administrator account.	10,000 operations	This limit applies across all stack sets involved in a Region. It includes stack instances affected by stack set creation and update operations, as well as creating, updating, or deleting stack instances directly.
StackSets queued operations	The maximum number of queued operations for a stack set at a given time.	10,000 operations	
Stacks imported using S3 object per stack set operation	The maximum number of stacks you can import using S3 object per stack set operation.	200 stacks	
Stacks imported using inline stack ids per stack set operation	The maximum number of stacks you can import using inline stack IDs per stack set operation.	10 stacks	

Quotas	Description	Value	Tuning strategy
Template body size in a request	The maximum size of a template body that you can pass in a `CreateStack`, `UpdateStack`, or `ValidateTemplate` request.	51,200 bytes	To use a larger template body, separate your template into multiple templates by using, for example, nested stacks. Or upload the template to an Amazon S3 bucket.
Template body size in an Amazon S3 object	The maximum size of a template body that you can pass in an Amazon S3 object for a `CreateStack`, `UpdateStack`, `ValidateTemplate` request with an Amazon S3 template URL.	1 MB	To use a larger template body, separate your template into multiple templates by using, for example, nested stacks. Or use minification to reduce the CloudFormation template size.
Template description	The maximum size of a template description.	1,024 bytes	
Versions per hook	The maximum number of versions you can create per Hook.	100 versions	

Feature availability

Not all features of CloudFormation may be available in every Region. For more information about AWS Regions, see Global infrastructure Region table.

- Macros are currently not available in the following Region:
  - Asia Pacific (Jakarta)
- Performing ECS blue/green deployments through CodeDeploy using CloudFormation is currently not available in the following Regions:
  - Africa (Cape Town)
  - Asia Pacific (Osaka)
  - Europe (Milan)

StackSets and macros

StackSets does not currently support creating or updating stack sets with service-managed permissions from templates that contain macros. This includes transforms, which are macros hosted by CloudFormation. For more information about macros, see Template macros.

Troubleshooting CloudFormation

When you use CloudFormation, you might encounter issues when you create, update, or delete CloudFormation stacks. The following sections can help you troubleshoot some common issues that you might encounter.

For general questions about CloudFormation, see the CloudFormation FAQs. You can also search for answers and post questions in the CloudFormation community on AWS re:Post.

Topics

- Troubleshooting guide
- Troubleshooting errors
- Contacting support
- Pre-deployment validation errors
- Express mode completion behavior

Troubleshooting guide

If CloudFormation fails to create, update, or delete your stack, you can view error messages or logs to help you learn more about the issue. The following tasks describe general methods for troubleshooting a CloudFormation issue. For information about specific errors and solutions, see the Troubleshooting errors section.

- Use operation IDs and failure filtering for targeted troubleshooting. CloudFormation assigns unique operation IDs to deployment events, making it easier to track and correlate related events. You can filter stack events by operation ID to focus on specific deployment phases and use failure filters to quickly identify the root cause of deployment failures. See View stack events by operation
- Use the CloudFormation console to view the status of your stack. In the console, you can view a list of stack events while your stack is being created, updated, or deleted. Stack events include operation IDs that help you correlate related events and track individual operation. You can click on the operation ID for a focused troubleshooting. From the operation event list, find the failure event and then view the status reason for that event. The status reason might contain an error message from CloudFormation or from a particular service that can help you troubleshoot your

problem. For more information about viewing stack events with operation IDs, see View stack events by operation.

- For Amazon EC2 issues, view the `cloud-init` and cfn logs. These logs are published on the Amazon EC2 instance in the `/var/log/` directory. These logs capture processes and command outputs while CloudFormation is setting up your instance. For Windows, view the EC2Configure service in `%ProgramFiles%\Amazon\EC2ConfigService`, EC2 Launch in `%ProgramData%\Amazon\EC2-Windows\Launch\Logs`, EC2 Launch v2 in `%ProgramData%\Amazon\EC2Launch\log`, and cfn logs in `C:\cfn\log`.

  You can also configure your CloudFormation template so that the logs are published to Amazon CloudWatch, which displays logs in the AWS Management Console so you don't have to connect to your Amazon EC2 instance. For more information, see View CloudFormation logs in the console in the Application Management Blog.

Troubleshooting errors

When you come across the following errors with your CloudFormation stack, you can use the following solutions to help you find the source of the problems and fix them.

Topics

- Delete stack fails
- Dependency error
- AWS Config and AWS Systems Manager conflicts
- Error parsing parameter when passing a list
- Insufficient IAM permissions
- Invalid value or unsupported resource property
- Quota exceeded
- Nested stacks are stuck in UPDATE_COMPLETE_CLEANUP_IN_PROGRESS, UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS, or UPDATE_ROLLBACK_IN_PROGRESS
- No updates to perform
- Resource failed to stabilize during a create, update, or delete stack operation
- Security group does not exist in VPC

- Update rollback failed
- Wait condition didn't receive the required number of signals from an Amazon EC2 instance
- Resource removed from stack but not deleted

Delete stack fails

To resolve this situation, try the following:

- Some resources must be empty before they can be deleted. For example, you must delete all objects in an Amazon S3 bucket or remove all instances in an Amazon EC2 security group before you can delete the bucket or security group.
- Ensure that you have the necessary IAM permissions to delete the resources in the stack. In addition to CloudFormation permissions, you must be allowed to use the underlying services, such as Amazon S3 or Amazon EC2.
- When stacks are in the `DELETE_FAILED` state because CloudFormation couldn't delete a resource, rerun the deletion with the `RetainResources` parameter and specify the resource that CloudFormation can't delete. CloudFormation deletes the stack without deleting the retained resource. Retaining resources is useful when you can't delete a resource, such as an S3 bucket that contains objects that you want to keep, but you still want to delete the stack.

  After you delete the stack, you can manually delete retained resources by using their associated AWS service.

  Alternatively, you may consider using the `FORCE_DELETE_STACK` option with the `DeletionMode` parameter. For more information on force deleting a stack, see Delete a stack from the CloudFormation console.
- You can't delete stacks that have termination protection enabled. If you attempt to delete a stack with termination protection enabled, the deletion fails and the stack--including its status--remains unchanged. Disable termination protection on the stack, then perform the delete operation again.

  This includes nested stacks whose root stacks have termination protection enabled. Deactivate termination protection on the root stack, then perform the delete operation again. It's strongly recommended that you don't delete nested stacks directly, but only delete them as part of deleting the root stack and all its resources.

  For more information, see Protect CloudFormation stacks from being deleted.

- For all other issues, if you have AWS Support, you can create a Support case. See Contacting support.

Dependency error

To resolve a dependency error, add a `DependsOn` attribute to resources that depend on other resources in your template. In some cases, you must explicitly declare dependencies so that CloudFormation can create or delete resources in the correct order. For example, if you create an Elastic IP and a VPC with an Internet gateway in the same stack, the Elastic IP must depend on the Internet gateway attachment. For more information, see DependsOn attribute.

AWS Config and AWS Systems Manager conflicts

AWS Config and AWS Systems Manager can automate infrastructure management tasks which can cause conflict with the deployment of a CloudFormation stack. Do the following to avoid any potential conflicts:

- Review the configuration of AWS Config and Systems Manager in the associated AWS account and AWS Region.
- Check for active rules or automation documents that might be triggered during a CloudFormation deployment. These may potentially cause conflicts or resource dependencies that conflict with your deployment.
- Check your CloudFormation template for any resources managed by AWS Config and Systems Manager. Check for potential overlaps or interdependencies and consider adjusting the template or automation configuration to avoid conflicts.
- Temporarily disable or suspend any related AWS Config rules or Systems Manager automations during CloudFormation deployment. Remember to restore the original configurations after the successful deployment to maintain the desired level of automation and compliance.
- Review the CloudFormation logs and error messages for any reference to AWS Config and Systems Manager-related issues to help pinpoint the source of the conflict.

For more information on AWS Config rules, see Evaluating Resources with AWS Config Rules.

For more information on Systems Manager automations, see AWS Systems Manager Automation.

Error parsing parameter when passing a list

When you use the AWS CLI or CloudFormation to pass in a list, add the escape character (\) before each comma. The following sample shows how you specify an input parameter when using the AWS CLI.

```
ParameterKey=CIDR,ParameterValue='10.10.0.0/16\,10.10.0.0/24\,10.10.1.0/24'
```

Insufficient IAM permissions

When you work with a CloudFormation stack, you not only need permissions to use CloudFormation, you must also have permission to use the underlying services that are described in your template. For example, if you're creating an Amazon S3 bucket or starting an Amazon EC2 instance, you need permissions to Amazon S3 or Amazon EC2. Review your IAM policy and verify that you have the necessary permissions before you work with CloudFormation stacks. For more information see, Control CloudFormation access with AWS Identity and Access Management.

Invalid value or unsupported resource property

When you create or update a CloudFormation stack, your stack can fail due to invalid input parameters, unsupported resource property names, or unsupported resource property values. For input parameters, verify that the resource exists. For example, when you specify an Amazon EC2 key pair or VPC ID, the resource must exist in your account and in the region in which you are creating or updating your stack. You can use AWS-specific parameter types to ensure that you use valid values.

For resource property names and values, update your template to use valid names and values. For more information, see the AWS resource and property types reference.

Quota exceeded

Verify that you didn't reach a resource quota. For example, the default maximum number of Amazon EC2 On-Demand instances that you can launch is 5. If try to create more Amazon EC2 On-Demand instances than your account quota, the instance creation fails and you receive the error `Status=start_failed`. To view the default AWS quotas by service, see AWS service quotas in the *AWS General Reference*.

For CloudFormation quotas and tweaking strategies, see Understand CloudFormation quotas.

Also, during an update, if a resource is replaced, CloudFormation creates new resource before it deletes the old one. This replacement might put your account over the resource quota, which would cause your update to fail. You can delete excess resources or request a quota increase.

Nested stacks are stuck in UPDATE_COMPLETE_CLEANUP_IN_PROGRESS, UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS, or UPDATE_ROLLBACK_IN_PROGRESS

A nested stack failed to roll back. Because of potential resource dependencies between nested stacks, CloudFormation doesn't start cleaning up nested stack resources until all nested stacks have been updated or have rolled back. When a nested stack fails to roll back, CloudFormation cancels all operations, regardless of the state that the other nested stacks are in. A nested stack that completed updating or rolling back but didn't receive a signal from CloudFormation to start cleaning up because another nested failed to roll back is in an `UPDATE_COMPLETE_CLEANUP_IN_PROGRESS` or `UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS` state. A nested stack that failed to update but didn't receive a signal to start rolling back is in an `UPDATE_ROLLBACK_IN_PROGRESS` state.

A nested stack might fail to roll back because of changes that were made outside of CloudFormation, when the stack template doesn't accurately reflect the state of the stack. A nested stack might also fail if an Auto Scaling group in a nested stack had an insufficient resource signal timeout period when the group was created or updated.

To fix the stack, contact AWS Support.

No updates to perform

To update a CloudFormation stack, you must submit template or parameter value changes to CloudFormation. However, CloudFormation won't recognize some template changes as an update, such as changes to a deletion policy, update policy, condition declaration, or output declaration. If you need to make such changes without making any other change, you can add or modify a metadata attribute for any of your resources. The metadata attribute can be any arbitrary value, as CloudFormation does not interpret its content. For more information, see Metadata attribute.

Resource failed to stabilize during a create, update, or delete stack operation

A resource didn't respond because the operation exceeded the CloudFormation timeout period or an AWS service was interrupted. For service interruptions, check that the relevant AWS service is running, and then retry the stack operation.

If the AWS services have been running successfully, check if your stack contains one of the following resources:

- `AWS::AutoScaling::AutoScalingGroup` for create, update, and delete operations
- `AWS::CertificateManager::Certificate` for create operations
- `AWS::CloudFormation::Stack` for create, update, and delete operations
- `AWS::ElasticSearch::Domain` for update operations
- `AWS::RDS::DBCluster` for create and update operations
- `AWS::RDS::DBInstance` for create, update, and delete operations
- `AWS::Redshift::Cluster` for update operations

Operations for these resources might take longer than the default timeout period. The timeout period depends on the resource and credentials that you use. To extend the timeout period, specify a service role when you perform the stack operation. If you're already using a service role, or if your stack contains a resource that isn't listed, contact AWS Support.

If your stack is in the `UPDATE_ROLLBACK_FAILED` state, see Update Rollback Failed.

Security group does not exist in VPC

Verify that the security group exists in the VPC that you specified. If the security group exists, ensure that you specify the security group ID and not the security group name. For example, the `AWS::EC2::SecurityGroupIngress` resource has a `SourceSecurityGroupName` and `SourceSecurityGroupId` properties. For VPC security groups, you must use the `SourceSecurityGroupId` property and specify the security group ID.

Update rollback failed

A dependent resource can't return to its original state, causing the rollback to fail (`UPDATE_ROLLBACK_FAILED` state). For example, you might have a stack that's rolling back to

an old database instance that was deleted outside of CloudFormation. Because CloudFormation doesn't know the database was deleted, it assumes that the database instance still exists and attempts to roll back to it, causing the update rollback to fail.

Depending on the cause of the failure, you can manually fix the error and continue the rollback. By continuing the rollback, you can return your stack to a working state (the `UPDATE_ROLLBACK_COMPLETE` state), and then try to update the stack again. The following list describes solutions to common errors that cause update rollback failures:

- Failed to receive the required number of signals

  Use the signal-resource command to manually send the required number of successful signals to the resource that's waiting for them, and then continue rolling back the update. For example, during an update rollback, instances in an Auto Scaling group might fail to signal success within the specified timeout duration. Manually send success signals to the Auto Scaling group. When you continue the update rollback, CloudFormation sees your signals and proceeds with the rollback.

- Changes to a resource were made outside of CloudFormation

  Manually sync resources so that they match the original stack's template, and then continue rolling back the update. For example, if you manually deleted a resource that CloudFormation is attempting to roll back to, you must manually create that resource with the same name and properties it had in the original stack.

- Insufficient permissions

  Check that you have sufficient IAM permissions to modify resources, and then continue the update rollback. For example, your IAM policy might allow you to create an S3 bucket, but not modify the bucket. Add the modify actions to your policy.

- Invalid security token

  CloudFormation requires a new set of credentials. No change is required. Continue rolling back the update, which refreshes the credentials.

- Limitation error

  Delete resources that you don't need or request a quota increase, and then continue rolling back the update. For example, if your account quota for the number of EC2 On-Demand instances is 5 and the update rollback exceeds that quota, it will fail.

- Resource didn't stabilize

  A resource didn't respond because the operation might have exceeded the CloudFormation timeout period or an AWS service might have been interrupted. No change is required. After the resource operation is complete or the AWS service is back in operation, continue rolling back the update.

To continue rolling back an update, you can use the CloudFormation console or AWS command line interface (AWS CLI). For more information, see Continue rolling back an update.

If none of these solutions work, you can skip the resources that CloudFormation can't successfully roll back. You can skip them in one of the following ways:

- Force the rollback to complete. CloudFormation skips each resource that fails to roll back, including resources in nested stacks. For more information, see Force a rollback to complete.
- Skip a specific set of resources. For more information, see the `ResourcesToSkip` parameter for the ContinueUpdateRollback API operation in the *AWS CloudFormation API Reference*.

CloudFormation sets the status of the skipped resources to `UPDATE_COMPLETE` and continues to roll back the stack. After the rollback is complete, the state of the skipped resources will be inconsistent with the state of the resources in the stack template. Before you perform another stack update, you must modify the resources or update the stack to be consistent with each other. If you don't, subsequent stack updates might fail and make your stack unrecoverable.

Wait condition didn't receive the required number of signals from an Amazon EC2 instance

To resolve this situation, try the following:

- Ensure that the AMI you're using has the CloudFormation helper scripts installed. If the AMI doesn't include the helper scripts, you can also download them to your instance. For more information, see the CloudFormation helper scripts reference in the *CloudFormation Template Reference Guide*.
- Verify that the `cfn-signal` command was successfully run on the instance. You can view logs, such as `/var/log/cloud-init.log` or `/var/log/cfn-init.log`, to help you debug the instance launch. You can retrieve the logs by logging in to your instance, but you must disable rollback on failure or else CloudFormation deletes the instance after your stack fails to create.

You can also publish the logs to Amazon CloudWatch. For Windows, you can view cfn logs in `C:\cfn\log` and EC2Config service logs in `%ProgramFiles%\Amazon\EC2ConfigService`.

- Verify that the instance has a connection to the Internet. If the instance is in a VPC, the instance should be able to connect to the Internet through a NAT device if it's is in a private subnet or through an Internet gateway if it's in a public subnet. To test the instance's Internet connection, try to access a public web page, such as `http://aws.amazon.com`. For example, you can run the following command on the instance. It should return an HTTP 200 status code.

  ```
  curl -I https://aws.amazon.com
  ```

  For information about configuring a NAT device, see NAT in the *Amazon VPC User Guide*.

Resource removed from stack but not deleted

During a stack update, CloudFormation has removed a resource from a stack but not deleted the resource. The resource still exists, but is no longer accessible through CloudFormation. This may occur during stack updates where:

- CloudFormation needs to replace an existing resource, so it first creates a new resource, then attempts to delete the old resource.
- You have removed the resource from the stack template, so CloudFormation attempts to delete the resource from the stack.

However, there may be cases where CloudFormation can't delete the resource. For example, if the user doesn't have permissions to delete a resource of a given type.

CloudFormation attempts to delete the old resource three times. If CloudFormation can't delete the old resource, it removes the old resource from the stack and continues updating the stack. When the stack update is complete, CloudFormation issues an `UPDATE_COMPLETE` stack event, but includes a `StatusReason` that states that one or more resources couldn't be deleted. CloudFormation also issues a `DELETE_FAILED` event for the specific resource, with a corresponding `StatusReason` providing more detail on why CloudFormation failed to delete the resource.

To resolve this situation, delete the resource directly using the console or API for the underlying service.

Contacting support

If you have AWS Support, you can create a technical support case at https://console.aws.amazon.com/support/home#/. Before you contact support, gather the following information:

- The ID of the stack. You can find the stack ID in the **Overview** tab of the CloudFormation console. For more information, see View stack information from the CloudFormation console.

  > **Important**
  >
  > Don't make changes to the resources in the stack outside of CloudFormation. Making changes to the resources in your stack outside of CloudFormation might put your stack in an unrecoverable state.

- Any stack error messages. For information about viewing stack error messages, see the Troubleshooting guide section.
- For Amazon EC2 issues, gather the `cloud-init` and cfn logs. These logs are published on the Amazon EC2 instance in the `/var/log/` directory. These logs capture processes and command outputs while your instance is setting up. For Windows, gather the EC2Configure service and cfn logs in `%ProgramFiles%\Amazon\EC2ConfigService` and `C:\cfn\log`.

You can also search for answers and post questions in the CloudFormation community on AWS re:Post.

Pre-deployment validation errors

Pre-deployment validation catches common errors before CloudFormation provisions resources. When validation fails, the stack operation stops immediately.

Property syntax validation failed

Cause: Your template contains resource property values that don't match the expected schema – for example, an invalid ARN pattern, a type mismatch (string instead of integer), an invalid enum value, or a missing required property.

Resolution: Review the `ValidationStatusReason` in the event details. It includes the exact property path where the error occurs. Fix the property value in your template and retry the operation.

Resource name conflict

Cause: Your template specifies a resource name that already exists in your account, and the target resource type does not support using an existing resource with that name.

Resolution: Either change the resource name in your template or remove the existing resource that is causing the conflict.

Service quota warning

Cause: Creating or updating resources would exceed your current AWS service quotas.

Resolution: Request a quota increase through the Service Quotas console, or modify your template to stay within current limits.

Bypass validation

If you determine a validation error is a false positive or you need to proceed despite the warning, use the `--disable-validation` AWS CLI flag or set `DisableValidation` to `true` in the API call.

Express mode completion behavior

When you use express mode, stack operations complete as soon as CloudFormation applies the resource configuration. This means that resources might not be fully operational when the stack operation reports success.

> **Note**
>
> If a resource attribute is referenced in a stack output, CloudFormation waits for the resource to propagate so that it can read the attribute value, even in express mode.

Common issues

The following issues can occur when you use express mode.

Resources not yet available

Your application might depend on a resource being fully operational immediately after deployment—for example, an Amazon EC2 instance passing health checks or a CloudFront distribution being globally deployed. If so, wait for the resource to reach its final state before relying on it.

Delete operations still in progress

After a stack delete completes in express mode, some resources might not yet be fully removed. If you immediately try to create a new stack with the same resource names, you might encounter name conflicts.

Dependent resource failures

Because express mode doesn't wait for resource stabilization, downstream resources that depend on a previous resource being fully operational might fail. If this occurs, consider using the default deployment mode for that stack.

For more information about express mode, see Express mode.

Troubleshoot unsuccessful CloudFormation stack deployments with AWS assistant

The AWS assistant is a generative artificial intelligence (AI) powered conversational assistant that can help you understand, build, extend, and operate AWS applications. For information about the AWS assistant, see the AWS assistant User Guide.

This document describes how to use the AWS assistant to help you troubleshoot unsuccessful CloudFormation deployments. This feature is designed to help you quickly identify and resolve issues encountered during stack deployment.

Powered by Amazon Bedrock: The AWS assistant is built on Amazon Bedrock and includes automated abuse detection implemented in Amazon Bedrock to enforce safety, security, and the responsible use of AI.

Topics

- Features
- How it works
- Pricing
- Regional availability
- Limitations

Features

The troubleshooting stack deployments with the AWS assistant solution provides the following features:

- **Error analysis** – Provides easy-to-understand analysis of CloudFormation stack deployments errors.
- **Resolution suggestions** – Offers step-by-step instructions to resolve identified issues.

How it works

When a CloudFormation stack creation or update fails, an error message is displayed in the AWS Management Console.

You can choose the **Diagnose with AWS assistant** button next to the error message.

The AWS assistant analyzes the error and provides an explanation of the issue.

For additional assistance, choose **Help me resolve** to receive detailed resolution steps.

Pricing

For information about pricing, see Amazon Q Developer pricing.

Regional availability

The AWS assistant is available in most AWS Regions. For a list of all the Regions where the AWS assistant is currently available, see Supported AWS Regions for AWS assistant in the *AWS assistant User Guide*.

Limitations

- This feature is only accessible through the AWS Management Console.
- This feature covers the top 20 most common errors. It does not cover all possible errors.
- Complex troubleshooting cases might still require Support intervention.

Document history for the AWS CloudFormation User Guide

The following table describes important changes to the AWS CloudFormation User Guide content after May 2018. To receive notifications about documentation updates, you can subscribe to an RSS feed.

Important

The table rows describing updates to the template reference content from May 2018 onward have moved to the new *AWS CloudFormation Template Reference*. For these updates and any future changes, see the Document history for the AWS CloudFormation Template Reference in the *AWS CloudFormation Template Reference*.

Change	Description	Date
PDF guide available	You can now download the *AWS CloudFormation User Guide* as a PDF.	May 30, 2025
Moved template reference content to a new guide	CloudFormation published the *CloudFormation Template Reference Guide*. For details, see The CloudFormation Template Reference Guide.	May 30, 2025
IaC generator supports partial scanning	You can now choose specific resource types to scan for, making it easier to generate Infrastructure-as-Code (IaC) templates from your existing resources. For more informati on, see Start a resource scan with CloudFormation IaC generator.	March 27, 2025

Stack refactoring	Stack refactoring simplifies reorganizing the resources in your CloudFormation stacks while still preserving the existing resource properties and data. For more informati on, see Stack refactoring.	February 6, 2025
Troubleshoot stack deployments with Amazon Q Developer	You can now use Amazon Q Developer to troublesh oot common errors when deploying CloudFormation stacks. For more information, see Troubleshoot unsuccess ful CloudFormation stack deployments with Amazon Q Developer.	November 22, 2024
Stack deployment timeline graph	You can now see a visual representation of your stack deployment. The stack deployment timeline graph shows the stack deploymen t status, individual resource deployment statuses, and the times the deployment statuses changed. For more information, see View a timeline of a CloudFormation stack deployment.	November 11, 2024

Visualize your scanned resources and generated templates	You can now streamline your Infrastructure as Code (IaC) generator workflows by visualizing scan summary details and previewing the generated templates before deploying your infrastructure stack. For more information, see View the scan summary in the CloudFormation console and Create a CloudFormation stack from scanned resources.	August 22, 2024
Amazon EventBridge integration with AWS CloudFormation Git sync	AWS CloudFormation Git sync now publishes sync status changes as events to Amazon EventBridge. For more information, see Repository Sync Status Change event detail and Resource Sync Status Change event detail.	July 29, 2024
Force delete stuck stacks	Two new options to force delete stacks is available for stack deletion operations that are stuck. You can now choose to force delete the stack but retain the resource, or the force delete the entire stack. For more information, see Delete a stack from the CloudFormation console.	May 22, 2024

AWS CloudTrail event stack operation root causes	CloudFormation improves the troubleshooting experience for stack operations with a new AWS CloudTrail deep-link integration. This feature directly links stack operation events in the CloudFormation console to relevant CloudTrail events. For more information, see Determine the cause of a stack failure.	May 15, 2024
Property level change sets	Property level change sets allow you to preview the changes that CloudForm ation deployments will make to the property values of resources. For more informati on, see View a change set for a CloudFormation stack.	April 12, 2024
CloudFormation introduces the `CONFIGURATION_COMP LETE` event	You can now use the `CONFIGURATION_COMP LETE` event to enable faster workflows involving the creation of resources. For more information, see Understand CloudFormation stack creation events.	March 11, 2024

Generate AWS CloudFormation templates and AWS CDK applications from existing AWS resources	You can now generate a template using resources provisioned in your account that are not already managed by CloudFormation. For more information, see Generate templates from existing resources with IaC generator.	February 2, 2024
StackSets concurrency mode	Concurrency Mode is a parameter for `StackSetOperationPreferences` that allows you to choose how the concurrency level behaves during stack set operations. For more information, see Choose the Concurrency Mode for CloudFormation StackSets.	November 9, 2023

Detailed StackSet drift information	The following APIs allow you to see which stack instances have drifted from the StackSet template and which resources have drifted. ListStackInstanceResourceDrifts Returns drift information for resources in a stack instance. StackInstanceResourceDriftsSummary The structure containing summary information about resource drifts for a stack instance.	July 24, 2023

CloudFormation StackSets APIs to control AWS Organizations trust access	CloudFormation StackSets provides customers with the following APIs for managing AWS Organizations trust access: ActivateOrganizationsAccess Activate trusted access with AWS Organizations. With trusted access between StackSets and Organizations activated, the management account has permissions to create and manage StackSets for your organization. DeactivateOrganizationsAccess Deactivates trusted access with AWS Organizations. If trusted access is deactivated, the management account does not have permissions to create and manage service-managed StackSets for your organization. DescribeOrganizationsAccess Retrieves information about the account's `OrganizationAccess` status. This API can be called either by the management account or	June 5, 2023

	the delegated administrator by using the `CallAs` parameter. This API can also be called without the `CallAs` parameter by the management account.	
`DescribeStackSet` API	The `DescribeStackSet` API has a new parameter to the list of Regions where a given stack set is deployed. For more information, see DescribeStackSet.	February 1, 2023
Managing StackSets events with CloudFormation and Amazon EventBridge	CloudFormation StackSets launch event notifications via Amazon EventBridge. You can trigger event-driven actions after creating, updating, or deleting your CloudFormation stack sets. For more information, see Monitoring CloudFormation and Git sync events with EventBridge.	November 16, 2022

Improved insights on stack instances for stack set operations	CloudFormation StackSets provides more detailed information on stack instances for stack set operations: DescribeStackSetOperation You can now use `DescribeStackSetOperation` to provide the count of failed stack instances for stack set operations during deployment. ListStackInstances You can now use the filtering option `LastOperationID` to list stack instances for stack set operations.	November 4, 2022
Managing events with CloudFormation and Amazon EventBridge	Receive notifications when specific CloudFormation events occur. For more information, see Monitoring CloudFormation and Git sync events with EventBridge.	July 20, 2022

Account level	CloudFormation announces the general availability of *account filter type*, a feature that allows customers to limit deployment targets to individual accounts or include additional accounts with provided OUs. For more information, see Account level targets for service-managed StackSets.	July 7, 2022
CloudFormation registry	CloudFormation announces the general availability of *Hooks*, a feature that allows customers to invoke custom logic to automate actions or inspect resource configura tions prior to a create, update or delete stack operation. For more information, see the CloudFormation Hooks User Guide.	February 10, 2022

Stack failure options	You can iteratively develop your applications when provisioning failures are encountered by starting from the point of failure without rolling back successfully provisioned resources. By specifying stack failure options, you can troubleshoot resources in a `CREATE_FAILED` or `UPDATE_FAILED` status. You can provision failure options for all stack deployments and change set operations. For more information, see Choose how to handle failures when provisioning resources.	August 30, 2021
Import stacks to stack set	You can now import existing stacks into new or existing stack sets. For more information, see Importing stacks into CloudFormation StackSets.	July 28, 2021
Increased quota	You can now declare a defaulted maximum of `2000` stacks in your AWS account. For more information, see Understand CloudFormation quotas.	July 15, 2021

Publish public third-party extensions	You can now use public extensions provided by third-party publishers, just as you would extensions from AWS. For more information, see Use third-party public extensions from the CloudFormation registry.	June 21, 2021
Reference macros in stack set templates	StackSets now supports creating or updating stack sets with self-managed permissions from templates that reference macros. For more information about macros, see Perform custom processing on CloudFormation templates with template macros.	April 14, 2021
Use the latest value of a Systems Manager parameter in a dynamic reference	You can now have CloudFormation use the latest version of an Systems Manager parameter whenever you create or update a stack. You are no longer required to specify a specific version. For more details, see Get a plaintext value from Systems Manager Parameter Store.	April 13, 2021

Modules support using period delimiters in resource names	You can now use a period as a delimiter in specifying the fully-qualified logical name for a resource contained in a module. For more informati on, see Reference module resources in CloudFormation templates.	April 8, 2021
CloudFormation StackSets now supports parallel region deployment	You can now choose to deploy StackSets into Regions sequentially or in parallel. For more information, see Stack set operation options.	April 6, 2021
CloudFormation StackSets now supports delegated administrator with AWS Organizations	In addition to the organizat ion's management account, delegated administrator accounts can create and manage stack sets with service-managed permissio ns for their organization. For more information, see Register a delegated administrator member account and Create CloudFormation StackSets with service-managed permissions.	February 18, 2021
CloudFormation StackSets Region availability	CloudFormation StackSets is now available in the Asia Pacific (Osaka) Region. For more information, see Managing stacks across accounts and Regions with StackSets.	February 10, 2021

Modules	Modules are a way for you to package resource configurations for inclusion across stack templates, in a transparent, manageable, and repeatable way. Modules can encapsulate common service configurations and best practices as modular, customizable building blocks for you to include in your stack templates. For more information, see Create reusable resource configurations that can be included across templates with CloudFormation modules.	November 24, 2020
Change sets for nested stacks	With change sets for nested stacks you can preview the changes to your application and infrastructure resources across the entire nested stack hierarchy and proceed with updates when you've confirmed that all the changes are as intended. For more information, see Change sets for nested stacks.	November 18, 2020

Increased quotas	The following AWS CloudFormation quotas have been updated. • You can now declare a maximum of 200 mappings in your AWS CloudForm ation template. • You can now declare a maximum of 200 mapping attributes for each mapping in your AWS CloudForm ation template. • You can now declare a maximum of 200 outputs in your AWS CloudFormation template. • You can now declare a maximum of 200 parameters in your AWS CloudFormation template. • You can now declare a maximum of 500 resources in your AWS CloudForm ation template. • You can now pass a template body with a maximum size of 1 MB in an Amazon S3 object.	October 22, 2020

Drift detection for private resources	CloudFormation now supports drift detection operations on an expanded list of AWS resources, as well as private resources that are defined as provisonable in the CloudFormation registry. For more information, see Resource type support.	October 1, 2020
Updated permissions required for registering resource providers	Registering a resource provider in your account now requires you have permission to access the schema handler package uploaded to an S3 bucket for that resource provider. For more informati on, see IAM permissions for registering a third-party private extension.	August 7, 2020
Resource import supports provisionable private resource types	Import operations now support private resource types that are *provisionable*; that is, whose provisioning type is either `FULLY_MUTABLE` or `IMMUTABLE`. For more information, see Resources that support import operations.	June 3, 2020

ECS blue/green deployments through CodeDeploy	You can now use CloudFormation to perform ECS blue/green deployments through CodeDeploy. Blue/green deployments are a safe deployment strategy provided by AWS CodeDeploy for minimizing interruptions caused by changing application versions. For more information, see Performing ECS blue/green deployments through CodeDeploy using CloudFormation.	May 19, 2020
CloudFormation StackSets Region availability	CloudFormation StackSets is now available in the AWS GovCloud (US-West) Region.	May 18, 2020
AWS CloudFormation StackSets integrates with AWS Organizations	You can now use StackSets to centrally manage deployments to all the accounts in your organization or specific organizational units (OUs) in AWS Organizations. You can enable automatic deployments to any new accounts added to your organization or OUs. The permissions needed to deploy across accounts will automatically be handled by StackSets. For more information, see Managing stacks across accounts and Regions with StackSets.	February 11, 2020

Drift Detection for StackSets	You can now run drift detection on a stack set and all the stack instances it includes. For more informati on, see Performing drift detection on CloudFormation StackSets.	November 19, 2019
CloudFormation registry now available	You can now use the CloudFormation console to view private and public resources that are available for use in your account. For more information, see View the available and activated extensions in the CloudForm ation registry.	November 18, 2019

CloudFormation registry API actions	The following API actions for managing types in the CloudFormation registry are now available. DeregisterType Removes a type or type version from active use in the CloudFormation registry. DescribeType Returns detailed informati on about a registered type. DescribeTypeRegistration Returns information about a type's registration, including its current status and type and version identifiers. ListTypeRegistrations Returns a list of registrat ion request identifiers for the specified type. ListTypes Returns summary information about types that have been registered with CloudFormation.	November 18, 2019

	ListTypeVersions Returns summary information about the versions of a type. RegisterType Registers a type with the CloudFormation registry. Registering a type makes it available for use in CloudFormation templates in your AWS account. SetTypeDefaultVersion Specify the default version of a type. The default version of a type will be used in CloudFormation operations. For more information about the CloudFormation registry, see Managing extensions with the CloudFormation registry	
Resource import added	If you created an AWS resource outside of CloudFormation managemen t, you can bring this existing resource into CloudForm ation management using `resource import`. For more information, see Import AWS resources into a CloudForm ation stack with a resource import.	November 11, 2019

Stack set limit increases	You can now create a maximum of 100 stack sets in your administrator account, create a maximum of 2000 stack instances per stack set, and run a maximum of 3500 stack instance operations in each region at the same time, per administrator account. For more information, see Understand CloudFormation quotas.	August 2, 2019
Limit for resources in concurrent stack operations	CloudFormation now enforces an account limit for the number of resources in concurrent stack operation s. This limit is determined by region. For more information, see Understand CloudForm ation quotas.	April 30, 2019
Stack instance operation limit	For StackSets, you can now have a maximum of 1500 stack instance operations running in a given region at the same time, per administr ator account. For more information, see Understand CloudFormation quotas.	December 13, 2018

The CAPABILITY_AUTO_EXPAND capability is now available	You can now use the CAPABILITY_AUTO_EXPAND capability to create or update a stack directly from a stack template that contains macros, without first reviewing the resulting changes in a change set first. For more information, see CreateStack or UpdateStack in *AWS CloudFormation API Reference*.	December 7, 2018
Stack drift detection added	You can now detect whether a stack's actual configuration has drifted from its expected template configuration as defined within CloudFormation. You can detect drift on an entire stack, or individual stack resources. For more information, see Detect unmanaged configuration changes to stacks and resources with drift detection.	November 13, 2018

`secretsmanager` dynamic reference now available	You can now use the `secretsmanager` dynamic reference to retrieve entire secrets or secret values that are stored in AWS Secrets Manager. Secrets can be database credentials, passwords, third-party API keys, and even arbitrary text. Using the `secretsmanager` dynamic reference guarantees that neither Secrets Manager nor CloudFormation logs or persists any resolved secret value. For more informati on, see Get a secret or secret value from Secrets Manager.	November 9, 2018
Macros now available	You can now use macros to perform custom processing on templates, from simple actions like find-and-replace operations to extensive transformations of entire templates. For more information, see Perform custom processing on CloudFormation templates with template macros.	September 6, 2018

CloudFormation now supports VPC endpoints powered by PrivateLink	You can use a VPC endpoint to create a private connection between your VPC and CloudFormation without requiring access over the Internet, through a NAT instance, a VPN connection, or Direct Connect. For more information, see Access CloudFormation using an interface endpoint (AWS PrivateLink).	August 22, 2018
Dynamic references support secure strings	You can now use new dynamic references to specify values that are stored and managed in other services, including Systems Manager Parameter Store `SecureString` type parameters, in your stack templates. For more information, see Get a secure string value from Systems Manager Parameter Store.	August 16, 2018
Stack sets now support customized execution roles	You can now use customized execution roles in target accounts to control the stack resources that users or groups can include in their stack sets. For more information, see Set up advanced permissions options for stack set operations.	May 30, 2018

Selective updates of stack instances	You can now use the optional `Accounts` and `Regions` parameters to specify the accounts and regions in which to update stack instances during a stack set update operation. For more information, see UpdateStackSet in the *CloudFormation API Reference*.	May 30, 2018
CloudFormation now creates S3 buckets with encryption enabled	For Amazon S3 buckets that CloudFormation creates to store uploaded stack templates, server-side encryption is now enabled by default, thereby encrypting all objects stored in those buckets. For more information, see Create a stack from the CloudFormation console.	May 24, 2018
FIPS endpoints added	CloudFormation now offers new endpoints which use FIPS 140-2 validated cryptographic modules in the following public US regions: US-East-1, US-East-2, US-West-1, and US-West-2. See CloudFormation endpoints and quotas in the *Amazon Web Services General Reference* for the new FIPS-compliant endpoint URLs.	May 17, 2018

For updates to the *CloudFormation Hooks User Guide*, see Document history for the CloudFormation Hooks User Guide in the *CloudFormation Hooks User Guide*.

Archived updates

The following table describes important changes in each release of the AWS CloudFormation User Guide before May 2018.

Change	Release Date	Description	API Version
Updated resources	July 22, 2019	Use the `encryptionOptions` property to specify an AWS owned key or a customer managed key for Amazon MQ brokers.	2010-05-15
Stack set naming convention	April 10, 2018	CloudFormation stacks created using stack sets now follow a new naming convention, in which the stack name contains the stack set name.	2010-05-15
New resources	April 10, 2018	AWS::AppSync::ApiKey Use the `AWS::AppSync::ApiKey` resource to create a unique key that you can distribute to clients who are executing GraphQL operations with AWS AppSync. AWS::AppSync::DataSource Use the `AWS::AppSync::DataSource` resource to create data sources for resolvers in AWS AppSync. AWS::AppSync::GraphQLApi Use the `AWS::AppSync::GraphQLApi` resource to create a new AWS AppSync GraphQL API. AWS::AppSync::GraphQLSchema Use the `AWS::AppSync::GraphQLSchema` resource to create the data model for your AWS AppSync GraphQL API.	2010-05-15

Change	Release Date	Description	API Version
		AWS::AppSync::Resolver Use the `AWS::AppSync::Resolver` resource to define the logical GraphQL resolver that you will attach to fields in a schema.	
Updated resource	April 10, 2018	AWS::Config::ConfigurationAggregator Use the `OrganizationAggregationSource` property type to specify the regions of AWS Config data to aggregate into an AWS Config configuration aggregator and the IAM role to use to retrieve AWS Organizations details.	2010-05-15
New resources	April 4, 2018	AWS::Config::AggregationAuthorization Use the `AWS::Config::AggregationAuthorization` resource to grant permission to an aggregator account to collect your AWS Config data. AWS::Config::ConfigurationAggregator Use the `AWS::Config::ConfigurationAggregator` resource to create a configuration aggregator for AWS Config.	2010-05-15
Stack sets now support customized administrator roles	March 29, 2018	Use customized administrator roles to control which users or groups can manage specific stack sets within the same administrator account. For more information, see Set up advanced permissions options for stack set operations.	2010-05-15
New resource	March 29, 2018	AWS::EC2::LaunchTemplate Use the `AWS::EC2::LaunchTemplate` resource to create a launch template for an Amazon EC2 instance.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	March 29, 2018	AWS::AutoScaling::AutoScalingGroup Use the `LaunchTemplate` property to specify the launch template to use to launch instances. AWS::EC2::SpotFleet In the SpotFleetRequestConfigData property type, use the `LaunchTemplateConfigs` property to describe a launch template and overrides.	2010-05-15
New `Fn::Cidr` intrinsic function	March 6, 2018	Returns the specified Cidr address block. For more information, see Fn::Cidr.	2010-05-15

Change	Release Date	Description	API Version
New resources	March 6, 2018	AWS::ApiGateway::VpcLink Use the `AWS::ApiGateway::VpcLink` resource to specify an API Gateway VPC link for a `AWS::ApiGateway::RestApi` to access resources in an Amazon Virtual Private Cloud (VPC). AWS::GuardDuty::Master Use the `AWS::GuardDuty::Master` resource to create a GuardDuty primary account. AWS::GuardDuty::Member Use the `AWS::GuardDuty::Member` resource to create a GuardDuty member account. AWS::SES::ConfigurationSet Use the `AWS::SES::ConfigurationSet` resource to create groups of rules that you can apply to the emails you send. AWS::SES::ConfigurationSetEventDestination Use the `AWS::SES::ConfigurationSetEventDestination` resource to specify a configuration set event destination. AWS::SES::ReceiptFilter Use the `AWS::SES::ReceiptFilter` resource to specify whether to accept or reject mail originating from an IP address or range of IP addresses. AWS::SES::ReceiptRule Use the `AWS::SES::ReceiptRule` resource to specify which actions Amazon SES should take	2010-05-15

Change	Release Date	Description	API Version
		when it receives mail on behalf of one or more email addresses or domains that you own. AWS::SES::ReceiptRuleSet Use the `AWS::SES::ReceiptRuleSet` resource to specify an empty rule set for Amazon SES. AWS::SES::Template Use the `AWS::SES::Template` resource to specify the content of the email, composed of a subject line, an HTML part, and a text-only part.	

Change	Release Date	Description	API Version
Updated resources	March 6, 2018	AWS::AutoScaling::AutoScalingGroup Use the `AutoScalingGroupName` property to specify the name of the Auto Scaling group. AWS::ApiGateway::RestApi Use the `ApiKeySourceType` property to specify the source of the API key for metering requests according to a usage plan. Use the `MinimumCompressionSize` property to specify a nullable integer that's used to enable compression or disable compression on an API. AWS::ApplicationAutoScaling::ScalingPolicy In the TargetTrackingScalingPolicyConfiguration property type, use the `DisableScaleIn` property to specify whether scale in by the target tracking policy is disabled. AWS::EC2::SpotFleet In the LaunchSpecifications property type, use the `TagSpecifications` property to specify the tags to apply during SpotFleet creation. AWS::Elasticsearch::Domain Use the `Arn` attribute to have `Fn::GetAtt` return the Amazon Resource Name (ARN) of the domain. The `DomainArn` attribute of `Fn::GetAtt` has been deprecated. AWS::RDS::DBCluster Use the `DBClusterIdentifier` property to specify the DB cluster identifier.	2010-05-15

Change	Release Date	Description	API Version
		AWS::RDS::DBCluster Use the `DBClusterIdentifier` property to specify the DB cluster identifier. AWS::Redshift::Cluster Use the `ClusterIdentifier` property to specify the unique identifier of the cluster. AWS::Route53::HealthCheck In the HealthCheckConfig property type, use the `Regions` property to specify the regions from which you want Route 53 health checkers to check the specified endpoint. AWS::SSM::Document Use the `Tags` property to specify the CloudFormation resource tags to apply to the document.	
Updated resource	February 19, 2018	AWS::CodeBuild::Project Use the `Triggers` property to configure a webhook for the project to begin to automatically rebuild the source code every time a code change is pushed to the repository. This is available only for GitHub projects in CloudFormation. It's not available for GitHub Enterprise projects.	2010-05-15
Updated resource	February 8, 2018	AWS::DynamoDB::Table Use the `SSESpecification` property to specify the settings to enable server-side encryption.	2010-05-15

Change	Release Date	Description	API Version
Updated resource	February 5, 2018	AWS::CodeBuild::Project In the Source `CodeBuild Project Source` property type: • Use the `GitCloneDepth` property to specify the depth of history to download. • Use the `InsecureSsl` property to specify whether to ignore SSL warnings while connecting to your GitHub Enterprise project repository.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	January 23, 2018	AWS::AutoScaling::LifecycleHook Use the `LifecycleHookName` property to specify the name of the lifecycle hook. AWS::DynamoDB::Table The `AttributeDefinitions` property now requires replacement when updated. AWS::EC2::Instance Use the `CreditSpecification` property to specify the credit option for CPU usage of a T2 instance. Use the `ElasticGpuSpecifications` property to specify Elastic GPUs, GPU resources that you can attach to your instance to accelerate the graphics performance of your applications. AWS::EC2::VPC The `InstanceTenancy` property now requires no interruption when updated from `"dedicated"` to `"default"` . AWS::ECS::Service Use the `HealthCheckGracePeriodSeconds` property to specify the period of time, in seconds, that the Amazon ECS service scheduler ignores unhealthy Elastic Load Balancing target health checks after a task has first started. AWS::IoT::TopicRule In the DynamoDBAction property type, the `RangeKeyField` and `RangeKeyValue` properties are no longer required.	2010-05-15

Change	Release Date	Description	API Version
		AWS::KinesisAnalytics::ApplicationOutput In the ApplicationOutput property type, use the `LambdaOutput` property to identify a Lambda function as the destination when configuring application output. AWS::Kinesis::Stream Use the `StreamEncryption` property to enable or update server-side encryption using an AWS KMS key for a specified stream. AWS::Lambda::Function Use the `ReservedConcurrentExecutions` property to specify the maximum of concurrent executions you want reserved for the function. AWS::RDS::DBSubnetGroup Use the `DBSubnetGroupName` property to specify the name for the DB Subnet Group. AWS::S3::Bucket Use the `BucketEncryption` property to specify default encryption for a bucket using server-side encryption with Amazon S3-managed keys SSE-S3 or AWS KMS keys (SSE-KMS) bucket. In the ReplicationRule property type, use the `SourceSelectionCriteria` property to specify additional filters in identifying source objects that you want to replicate. In the ReplicationDestination property type:	

Change	Release Date	Description	API Version
		• Use the `AccessControlTranslation` property to specify replica ownership of the AWS account that owns the destination bucket. • Use the `Account` property to specify destination bucket owner account ID. • Use the `EncryptionConfiguration` property to specify encryption-related information for a bucket that is a destination for replicated objects. AWS::SSM::Association Use the `AssociationName` property to specify the name of the association between an SSM document and EC2 instances that contain a configuration agent to process the document.	
Rollback triggers added to the CloudFormation console.	January 15, 2018	Rollback triggers enable you to have CloudFormation monitor the state of your application during stack creation and updating, and to roll back that operation if the application breaches the threshold of any of the alarms you've specified. For more information, see Monitor and Roll Back Stack Operations.	2010-05-15
Updated resource	January 12, 2018	AWS::SSM::Parameter Use the `AllowedPattern` property to specify a regular expression used to validate the parameter value.	2010-05-15

Change	Release Date	Description	API Version
New resources	December 5, 2017	AWS::Inspector::AsssmentTarget Use the `AWS::Inspector::AsssmentTarget` resource to create an Amazon Inspector assessment target. AWS::Inspector::AssessmentTemplate Use the `AWS::Inspector::AssessmentTemplate` resource to create an Amazon Inspector assessment template. AWS::Inspector::ResourceGroup Use the `AWS::Inspector::ResourceGroup` resource to create an Amazon Inspector resource group, which defines tags that identify AWS resources that make up an Amazon Inspector assessment target. AWS::ServiceDiscovery::Instance Use the `AWS::ServiceDiscovery::Instance` resource to specify information about an instance that Amazon Route 53 creates. AWS::ServiceDiscovery::PrivateDnsNamespace Use the `AWS::ServiceDiscovery::PrivateDnsNamespace` resource to specify information about a private namespace for Amazon Route 53. AWS::ServiceDiscovery::PublicDnsNamespace Use the `AWS::ServiceDiscovery::PublicDnsNamespace` resource to specify information about a public namespace for Amazon Route 53.	2010-05-15

Change	Release Date	Description	API Version
		AWS::ServiceDiscovery::Service Use the `AWS::ServiceDiscovery::Service` resource to define a template for up to five records and an optional health check that you want Amazon Route 53 to create when you register an instance.	
Updated resource	December 5, 2017	AWS::KinesisAnalytics::Application In the Input property type, use the `InputProcessingConfiguration` property to transform records as they're received from the stream.	2010-05-15
Updated resource	December 1, 2017	AWS::CodeBuild::Project Use the `BadgeEnabled` property to generate a publicly accessible URL for a project's build badge. Use the `Cache` property to configure cache settings for build dependencies. Use the `VpcConfig` property to enable CodeBuild to access resources in an Amazon VPC. In the EnvironmentVariable property type, use the `Type` property to specify the type of environment variable.	2010-05-15
New resource	November 30, 2017	AWS::Cloud9::EnvironmentEC2 Use the `AWS::Cloud9::EnvironmentEC2` resource to create an Amazon EC2 development environment in AWS Cloud9.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	November 29, 2017	AWS::ECS::TaskDefinition Use the Cpu property to specify the number of cpu units needed for the task. Use the ExecutionRoleArn property to specify the ARN of the execution role. Use the Memory property to specify the amount (in MiB) of memory needed for the task. Use the RequiresCompatibilities property to specify the launch type the task requires. AWS::ECS::Service Use the LaunchType property to specify the launch type on which to run your service. Use the NetworkConfiguration property to specify the network configuration for the service. Use the PlatformVersion property to specify the platform version on which to run your service.	2010-05-15
New resources	November 28, 2017	AWS::GuardDuty::Detector Use the AWS::GuardDuty::Detector resource to create a single Amazon GuardDuty detector. AWS::GuardDuty::IPSet Use the AWS::GuardDuty::IPSet resource to create an Amazon GuardDutyIP set. AWS::GuardDuty::ThreatIntelSet Use the AWS::GuardDuty::ThreatIntelSet resource to create a ThreatIntelSet.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	November 28, 2017	AWS::CodeDeploy::Application Use the `ComputePlatform` property to specify an AWS Lambda compute platform for CodeDeploy to deploy an application to. AWS::CodeDeploy::DeploymentGroup In the DeploymentStyle property type, use the `DeploymentType` property to specify a blue/green deployment on a Lambda compute platform. AWS::EC2::SpotFleet In the SpotFleetRequestConfigData property type, the `SpotPrice` property is now optional. AWS::Lambda::Alias Use the `RoutingConfig` property to specify two different versions of an AWS Lambda function, allowing you to dictate what percentage of traffic will invoke each version.	2010-05-15
New `CodeDeployLambdaAliasUpdate` update policy	November 28, 2017	Use the `CodeDeployLambdaAliasUpdate` update policy to perform an CodeDeploy deployment when the version changes on an `AWS::Lambda::Alias` resource. For more information, see UpdatePolicy Attribute.	2010-05-15
New SSM parameter types	November 21, 2017	Use SSM parameter types to use existing parameters from Systems Manager Parameter Store. **Note**: CloudFormation doesn't currently support the `SecureString` type. For more information, see SSM Parameter Types.	2010-05-15

Change	Release Date	Description	API Version
New `ResolvedValue` field for `Parameter` data type	November 21, 2017	The `ResolvedValue` field returns the value that's used in the stack definition for an `SSM` parameter. For more information, see the Parameter data type in the *CloudFormation API Reference*.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	November 20, 2017	AWS::ApiGateway::ApiKey Use the `CustomerId` property to specify an AWS Marketplace customer identifier. Use the `GenerateDistinctId` property to specify whether the key identifier is distinct from the created API key value. AWS::ApiGateway::Authorizer Use the `AuthType` property to specify a customer-defined field that's used in Swagger imports and exports without functional impact. AWS::ApiGateway::DomainName Use the `EndpointConfiguration` property to specify the endpoint types of an API Gateway domain name. Use the `RegionalCertificateArn` property to reference a certificate for use by the regional endpoint for a domain name. AWS::ApiGateway::Method In the Integration and IntegrationResponse property types, use the `ContentHandling` property to specify how to handle request payload content type conversions. AWS::ApiGateway::RestApi Use the `EndpointConfiguration` property to specify the endpoint types of an API Gateway REST API.	2010-05-15

Change	Release Date	Description	API Version
		AWS::ApplicationAutoScaling::ScalableTarget Use the `ScheduledActions` property to specify scheduled actions for an Application Auto Scaling scalable target. AWS::ECR::Repository Use the `LifecyclePolicy` property to specify a lifecycle policy for an Amazon ECR repository. AWS::ECS::TaskDefinition In the ContainerDefinition property type, use the `LinuxParameters` property to specify Linux-specific options for an Amazon ECS container. AWS::ElastiCache::ReplicationGroup Use the `AtRestEncryptionEnabled` property to enable encryption at rest. Use the `AuthToken` property to specify a password that's used to access a password-protected server. Use the `TransitEncryptionEnabled` property to enable in-transit encryption. AWS::ElasticLoadBalancingV2::TargetGroup Use the `TargetGroupName` attribute with the `Fn::GetAtt` function to get the name of an Elastic Load Balancing target group. AWS::Elasticsearch::Domain Use the `VPCOptions` property to specify a VPC configuration for the OpenSearch Service domain.	

Change	Release Date	Description	API Version
		AWS::EMR::Cluster Use the EbsRootVolumeSize property to specify the size of the EBS root volume for an Amazon EMR cluster. AWS::RDS::DBInstance Use the SourceRegion and KmsKeyId properties to create an encrypted read replica from a cross-region source DB instance. AWS::Route53::HostedZone Use the QueryLoggingConfig property to specify a configuration for DNS query logging.	
New NoEcho field for custom resource Response objects	November 20, 2017	You can now use the optional NoEcho field to mask the output of a custom resource. For more information, see Custom Resource Response Objects. The corresponding noEcho parameter is supported by the send method. For more information, see cfn-response Module.	2010-05-15
Stack instance overrides added for stack sets.	November 17, 2017	CloudFormation StackSets allows you to override parameter values in stack instances by account and region. You can override parameter values when you create the stack instances, or when updating existing stack instances. For more information, see Override Parameters on Stack Instances.	2010-05-15

Change	Release Date	Description	API Version
Updated resource	November 15, 2017	AWS::StepFunctions::StateMachine You can use `AWS::StepFunctions::StateMachine` to specify a `StateMachineName` when creating a state machine, and both `DefinitionString` and `RoleArn` can be updated without replacing the state machine.	2010-05-15
StackSets now supports a maximum of 500 stack instances per stack set.	November 6, 2017	You can now create up to a maximum of 500 stack instances per stack set. For more information about CloudFormation limits, see Understand CloudFormation quotas.	2010-05-15
New resources	November 2, 2017	AWS::CloudFront::CloudFrontOriginAccessIdentity Use the `AWS::CloudFront::CloudFrontOriginAccessIdentity` resource to specify the Amazon CloudFront origin access identity to associate with the origin of a CloudFront distribution. AWS::CloudFront::StreamingDistribution Use the `AWS::CloudFront::StreamingDistribution` resource to specify an Adobe Real-Time Messaging Protocol (RTMP) streaming distribution for CloudFront.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	November 2, 2017	AWS::ApiGateway::Deployment The `StageName` property has been deprecated on the StageDescription property type. AWS::ApiGateway::Method Use the `OperationName` property to assign a friendly name to an API Gateway method. Use the `RequestValidatorId` property to associate a request validator with a method. AWS::AutoScaling::AutoScalingGroup Use the `LifecycleHookSpecificationList` property to specify actions to perform when Auto Scaling launches or terminates instances. AWS::CloudFront::Distribution Use the `Tags` property to specify an arbitrary set of tags (key–value pairs) to associate with a CloudFront distribution. In the CacheBehavior and DefaultCacheBehavior property types, use the `LambdaFunctionAssociations` property to specify Lambda function associations for a CloudFront distribution. In the CustomOriginConfig property type, use the `OriginKeepaliveTimeout` property to specify a custom keep-alive timeout, and use the `OriginReadTimeout` property to specify a custom origin read timeout. In the DistributionConfig property type, use the `IPV6Enabled` property to specify whether	2010-05-15

Change	Release Date	Description	API Version
		CloudFront responds to IPv6 DNS requests with an IPv6 address for your distribution. AWS::CodeDeploy::DeploymentGroup In the LoadBalancerInfo property type, use the `TargetGroupInfoList` property to specify information about a target group in Elastic Load Balancing to use in a deployment. AWS::EC2::SecurityGroup, AWS::EC2::SecurityGroupEgress, and AWS::EC2::SecurityGroupIngress Use the `Description` property to specify the description of a security group rule. AWS::EC2::Subnet The `Ipv6CidrBlock` property now supports `No interruption` updates. AWS::EC2::VPNGateway Use the `AmazonSideAsn` property to specify a private Autonomous System Number (ASN) for the Amazon side of a BGP session. AWS::EC2::VPNConnection Use the `VpnTunnelOptionsSpecifications` property to configure tunnel options for a VPN connection. AWS::ElasticBeanstalk::ConfigurationTemplate and AWS::ElasticBeanstalk::Environment In the ConfigurationOptionSetting and OptionSetting property types, use the `ResourceName` property to specify a resource name for a time-based scaling configuration option.	

Change	Release Date	Description	API Version
		AWS::EMR::Cluster Use the `CustomAmiId` property to specify a custom Amazon Linux AMI for a cluster. AWS::KinesisFirehose::DeliveryStream Use the `Arn` attribute with the `Fn::GetAtt` function to get the Amazon Resource Name (ARN) of the delivery stream. AWS::KMS::Key Use the `Tags` property to specify an arbitrary set of tags (key–value pairs) to associate with a customer managed key. AWS::OpsWorks::Layer and AWS::OpsWorks::Stack Use the `Tags` property to specify an arbitrary set of tags (key–value pairs) to associate with an OpsWorks layer or stack. AWS::RDS::OptionGroup In the OptionConfiguration property type, use the `OptionVersion` property to specify a version for the option. AWS::S3::Bucket Use the `AnalyticsConfigurations` property to configure an analysis filter for an Amazon S3 bucket.	

Change	Release Date	Description	API Version
New resources	October 24, 2017	AWS::Glue::Classifier Use the `AWS::Glue::Classifier` resource to create an AWS Glue classifier. AWS::Glue::Connection Use the `AWS::Glue::Connection` resource to specify an AWS Glue connection to a data source. AWS::Glue::Crawler Use the `AWS::Glue::Crawler` resource to specify an AWS Glue crawler. AWS::Glue::Database Use the `AWS::Glue::Database` resource to create an AWS Glue database. AWS::Glue::DevEndpoint Use the `AWS::Glue::DevEndpoint` resource to specify a development endpoint for remotely debugging ETL scripts. AWS::Glue::Job Use the `AWS::Glue::Job` resource to specify an AWS Glue job in the data catalog. AWS::Glue::Partition Use the `AWS::Glue::Partition` resource to create an AWS Glue partition, which represents a slice of table data. AWS::Glue::Table Use the `AWS::Glue::Table` resource to create an AWS Glue table.	2010-05-15

Change	Release Date	Description	API Version
		AWS::Glue::Trigger Use the `AWS::Glue::Trigger` resource to specify triggers that run AWS Glue jobs.	
New resources	October 11, 2017	AWS::SSM::MaintenanceWindow Use the `AWS::SSM::MaintenanceWindow` resource to create an AWS Systems Manager Maintenance Window. AWS::SSM::MaintenanceWindowTarget Use the `AWS::SSM::MaintenanceWindowTarget` resource to register a target with a Maintenance Window. AWS::SSM::MaintenanceWindowTask Use the `AWS::SSM::MaintenanceWindowTask` resource to define a Maintenance Window task. AWS::SSM::PatchBaseline Use the `AWS::SSM::PatchBaseline` resource to define a Systems Manager patch baseline.	2010-05-15
New resource	October 10, 2017	AWS::ElasticLoadBalancingV2::ListenerCertificate Use the `AWS::ElasticLoadBalancingV2::ListenerCertificate` resource to specify certificates for an Elastic Load Balancing listener.	2010-05-15
New resource	September 27, 2017	AWS::Athena::NamedQuery Use the `AWS::Athena::NamedQuery` resource to create an Amazon Athena query.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	September 27, 2017	AWS::EC2::NatGateway Use the `Tags` property to specify resource tags for a NAT gateway. AWS::ElasticBeanstalk::Application Use the `ResourceLifecycleConfig` property to define lifecycle settings for resources that belong to the application, and the service role that Elastic Beanstalk assumes in order to apply lifecycle settings. AWS::ElasticBeanstalk::ConfigurationTemplate and AWS::ElasticBeanstalk::Environment Use the `PlatformArn` property to specify a custom platform for Elastic Beanstalk. AWS::ElasticLoadBalancingV2::TargetGroup In the TargetDescription property type, use the `AvailabilityZone` property to specify the Availability Zone where the IP address is to be registered. AWS::Events::Rule In the Target property type, use the following properties for input transformation of events and setting Amazon ECS task and Kinesis stream targets. • `EcsParameters` • `InputTransformer` • `KinesisParameters` • `RunCommandParameters`	2010-05-15

Change	Release Date	Description	API Version
		AWS::KinesisFirehose::DeliveryStream Use the `DeliveryStreamType` property to specify the stream type and the `KinesisStreamSourceConfiguration` property to specify the stream and role ARNs for a Kinesis stream used as the source for a delivery stream. AWS::RDS::DBInstance For the `Engine` property, if you have specified `oracle-se` or `oracle-se1`, you can update to `oracle-se2` without the database instance being replaced. AWS::S3::Bucket Use the `AccelerateConfiguration` property to configure the transfer acceleration state for an Amazon S3 bucket.	
Termination protection added for stacks.	September 26, 2017	Enabling termination protection on a stack prevents it from being accidentally deleted. A user can't delete a stack with termination protection enabled. For more information, see Protecting a Stack From Being Deleted.	2010-05-15
Changed default `umask` value from version 1.4-22 onwards	September 14, 2017	The default `umask` parameter value for the cfn-hup.conf configuration file is now `022`. For more information, see cfn-hup .	

Change	Release Date	Description	API Version
Updated resources	September 7, 2017	AWS::ElasticLoadBalancingV2::LoadBalancer Use the SubnetMappings property to specify the IDs of the subnets to attach to the load balancer. Use the Type property to specify the type of load balancer to create. AWS::ElasticLoadBalancingV2::TargetGroup Use the TargetType property to specify the registration type of the targets in this target group.	2010-05-15
Rollback triggers added to the CloudFormation API	August 31, 2017	Rollback triggers enable you to have CloudFormation monitor the state of your application during stack creation and updating, and to roll back that operation if the application breaches the threshold of any of the alarms you've specified. For more information, see RollbackConfiguration in the *CloudFormation API Reference*.	2010-05-15
New umask parameter for cfn-hup.conf file	August 31, 2017	Use the umask parameter in the cfn-hup.conf configuration file to control file permissions used by the cfn-hup daemon (version 1.4-21). For more information, see cfn-hup.	
Updated resources for VPC Sizing support	August 29, 2017	AWS::EC2::VPCCidrBlock Use the CidrBlock property to associate an IPv4 CIDR block with a VPC. AWS::EC2::VPC Use the CidrBlockAssociations attribute with the Fn::GetAtt function to get a list of IPv4 CIDR block association IDs associated with the VPC.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	August 23, 2017	AWS::S3::Bucket In the Rule property type, use the TagFilters property to specify tags to use in identifying a subset of objects for an Amazon S3 bucket. Use the MetricsConfiguration property to specify a metrics configuration for the CloudWatch request metrics from an Amazon S3 bucket. AWS::IoT::TopicRule In the Action property type, use the DynamoDBv2Action property to describe an AWS IoT action that writes data to a DynamoDB table. In the Action property type, the DynamoDBAction property now supports the HashKeyType and RangeKeyType properties. AWS::Lambda::Permission Use the EventSourceToken property to specify a unique token that must be supplied by the principal invoking the function.	2010-05-15
New pseudo parameters	August 23, 2017	Use the AWS::Partition pseudo parameter to return the partition that a resource is in. Use the AWS::URLSuffix pseudo parameter to return the suffix for a domain. For more information, see Pseudo Parameters Reference.	2010-05-15

Change	Release Date	Description	API Version
New resources for DAX support	August 22, 2017	AWS::DAX::Cluster Use the `AWS::DAX::Cluster` resource to create a DAX cluster for use with Amazon DynamoDB. AWS::DAX::ParameterGroup Use the `AWS::DAX::ParameterGroup` resource to create a parameter group for use with Amazon DynamoDB. AWS::DAX::SubnetGroup Use the `AWS::DAX::SubnetGroup` resource to create a subnet group for use with DAX (DynamoDB Accelerator).	2010-05-15

Change	Release Date	Description	API Version
New resources	August 18, 2017	AWS::ApiGateway::DocumentationPart and AWS::ApiGateway::DocumentationPart Use the `AWS::ApiGateway::DocumentationPart` and `AWS::ApiGateway::DocumentationVersion` resources to create documentation for your API Gateway API. AWS::ApiGateway::GatewayResponse Use the `AWS::ApiGateway::GatewayResponse` resource to create a custom response for your API Gateway API. AWS::ApiGateway::RequestValidator Use the `AWS::ApiGateway::RequestValidator` resource to set up validation rules for incoming requests to your API Gateway API. AWS::EC2::NetworkInterfacePermission Use the `AWS::EC2::NetworkInterfacePermission` resource to grant an AWS account permission to a network interface.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	August 18, 2017	AWS::ApiGateway::Stage Use the DocumentationVersion property to specify a versioned snapshot of the API documentation. AWS::AutoScaling::ScalingPolicy Use the TargetTrackingConfiguration property to specify an Auto Scaling target tracking scaling policy configuration. AWS::CloudTrail::Trail Use the EventSelectors property for Amazon S3 Data Events support. AWS::CodeDeploy::DeploymentGroup Use the LoadBalancerInfo and DeploymentStyle properties to specify an Elastic Load Balancing load balancer for an in-place deployment. Use the AutoRollbackConfiguration property to configure automatic rollback for the deployment. AWS::EC2::SpotFleet In the SpotFleetRequestConfigData property type, use the ReplaceUnhealthyInstances property to indicate whether the Spot fleet should replace unhealthy instances and the Type property to specify the type of request. AWS::EC2::Subnet Use the AssignIpv6AddressOnCreation and Ipv6CidrBlock properties to create a subnet with an IPv6 CIDR block.	2010-05-15

Change	Release Date	Description	API Version
		AWS::KinesisFirehose::DeliveryStream Use the `ExtendedS3DestinationConfiguration` property to configure a destination in Amazon S3. Use the `ProcessingConfiguration` subproperty within each destination configuration to invoke Lambda functions that transform incoming source data and deliver the transformed data to destinations. AWS::RDS::DBCluster and AWS::RDS::DBInstance The default `DeletionPolicy` is now `Snapshot` for `AWS::RDS::DBCluster` resources and for `AWS::RDS::DBInstance` resources that don't specify the `DBClusterIdentifier` property. For more information, see DeletionPolicy Attribute. AWS::S3::Bucket In the Rule property type, use the `AbortIncompleteMultipartUpload` property to specify a lifecycle rule that aborts incomplete multipart uploads to an Amazon S3 bucket. AWS::SQS::Queue Use the `KmsMasterKeyId` and `KmsDataKeyReusePeriodSeconds` properties to configure server-side encryption for Amazon SQS. Added the `Arn` attribute to the `Fn::GetAtt` intrinsic function for the following resources: • AWS::CloudTrail::Trail. Also added `SnsTopicArn` .	

Change	Release Date	Description	API Version
		• AWS::CloudWatch::Alarm • AWS::DynamoDB::Table • AWS::ECS::Cluster • AWS::IoT::Policy • AWS::IoT::TopicRule • AWS::Logs::Destination	
Support for stack tags in CodePipeline artifacts	August 18, 2017	You can now specify tags for stacks in template configuration files for use as artifacts for CodePipeline pipelines. Specified tags are applied to stacks created using the template configuration file. For more information, see CloudFormation Artifacts.	2010-05-15
Create encrypted file systems	August 14, 2017	AWS::EFS::FileSystem Use the `Encrypted` property to encrypt an Amazon EFS file system during creation. Use the `KmsKeyId` property to optionally specify a custom customer managed key to use to protect the encrypted file system.	2010-05-15

Change	Release Date	Description	API Version
New resources for AWS Batch support	August 8, 2017	AWS::Batch::ComputeEnvironment Use the `AWS::Batch::ComputeEnvironment` resource to define your AWS Batch compute environment. AWS::Batch::JobDefinition Use the `AWS::Batch::JobDefinition` resource to specify the parameters for an AWS Batch job definition. AWS::Batch::JobQueue Use the `AWS::Batch::JobQueue` resource to define your AWS Batch job queue.	2010-05-15
New resources for Amazon Managed Service for Apache Flink support	July 28, 2017	AWS::KinesisAnalytics::Application Use the `AWS::KinesisAnalytics::Application` resource to create an Amazon Managed Service for Apache Flink application. AWS::KinesisAnalytics::ApplicationOutput Use the `AWS::KinesisAnalytics::ApplicationOutput` resource to add an external destination to your Amazon Managed Service for Apache Flink application. AWS::KinesisAnalytics::ApplicationReferenceDataSource Use the `AWS::KinesisAnalytics::ApplicationReferenceDataSource` resource to add a reference data source to an existing Amazon Managed Service for Apache Flink application.	2010-05-15

Change	Release Date	Description	API Version
Use StackSets to centrally manage stacks across accounts and regions	July 25, 2017	StackSets enables you to create, update, or delete stacks across multiple accounts and regions in a single operation. Using an administrator account, you define and manage a CloudFormation template, and use the template as the basis for provisioning stacks into selected target accounts across specified regions. For more information, see Managing stacks across accounts and Regions with StackSets.	2010-05-15
View stack events by client request token	July 14, 2017	In the console, stack operations display the client request token on the **Events** tab. All events triggered by a given stack operation are assigned the same client request token, which you can use to track operations. For more information, see Viewing CloudFormation Stack Data and Resources on the AWS Management Console and StackEvent in the *AWS CloudFormation API Reference*.	2010-05-15
Use stack quick-create links	July 14, 2017	Use quick-create links to get stacks up and running quickly. You can specify the template URL, stack name, and template parameters to prepopulate a single **Create Stack Wizard** page. For more information, see Creating Quick-Create Links for Stacks.	2010-05-15

Change	Release Date	Description	API Version
New resources for AWS Database Migration Service support	July 12, 2017	AWS::DMS::Certificate Use the `AWS::DMS::Certificate` resource to create an SSL certificate that encrypts connections between AWS DMS endpoints and the replication instance. AWS::DMS::Endpoint Use the `AWS::DMS::Endpoint` resource to create an AWS DMS endpoint. AWS::DMS::EventSubscription Use the `AWS::DMS::EventSubscription` resource to get notifications for AWS DMS events through the Amazon Simple Notification Service. AWS::DMS::ReplicationInstance Use the `AWS::DMS::ReplicationInstance` resource to create an AWS DMS replication instance. AWS::DMS::ReplicationSubnetGroup Use the `AWS::DMS::ReplicationSubnetGroup` resource to create an AWS DMS replication subnet group. AWS::DMS::ReplicationTask Use the `AWS::DMS::ReplicationTask` resource to create an AWS DMS replication task.	2010-05-15

Change	Release Date	Description	API Version
New resources	July 5, 2017	AWS::CloudWatch::Dashboard Use the `AWS::CloudWatch::Dashboard` resource to specify a custom CloudWatch dashboard for your CloudWatch console. AWS::ApiGateway::DomainName Use the `AWS::ApiGateway::DomainName` resource to specify a custom, friendly URL for your API that's deployed to Amazon API Gateway. AWS::EC2::EgressOnlyInternetGateway Use the `AWS::EC2::EgressOnlyInternetGateway` resource to create an egress-only internet gateway for your VPC. InstanceFleetConfig Use the `InstanceFleetConfig` resource to configure a Spot Instance fleet for an Amazon EMR cluster.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	July 5, 2017	AWS::ApiGateway::RestApi Use the `BinaryMediaTypes` property to specify supported binary media types. AWS::ApplicationAutoScaling::ScalingPolicy Use the `TargetTrackingScalingPolicyConfiguration` property to specify a target tracking scaling policy configuration. AWS::CloudTrail::Trail Use the `TrailName` property to specify a custom name for an AWS CloudTrail resource. Use the `Tags` property to specify resource tags. AWS::CodeDeploy::DeploymentGroup Use the `AlarmConfiguration` property to configure alarms for the deployment group. Use the `TriggerConfigurations` property to configure notification triggers for the deployment group. AWS::EMR::Cluster Use the `CoreInstanceFleet` property and the `MasterInstanceFleet` property in the JobFlowInstancesConfig property type to configure the Spot Instance fleet for an Amazon EMR cluster. AWS::DynamoDB::Table Use the `TimeToLiveSpecification` property to specify the Time to Live (TTL) settings for an Amazon DynamoDB table.	2010-05-15

Change	Release Date	Description	API Version
		Use the `Tags` property to specify resource tags for a DynamoDB table. AWS::EC2::Instance The `IamInstanceProfile` property now supports `No interruption` updates. AWS::EC2::Route Use the `EgressOnlyInternetGatewayId` property to specify an egress-only Internet gateway for an EC2 route. AWS::Kinesis::Stream Use the `RetentionPeriodHours` property to specify the number of hours that data records stored in shards remain accessible. AWS::RDS::DBCluster Use the `ReplicationSourceIdentifier` property to create a DB cluster as a Read Replica of another DB cluster or an Amazon RDS MySQL DB instance. AWS::Redshift::Cluster Use the `LoggingProperties` property to create audit log files and store them in Amazon S3.	
New resources	June 6, 2017	AWS::EMR::SecurityConfiguration Use the `AWS::EMR::SecurityConfiguration` resource to create a security configuration, which is stored in the service and can be specified when a cluster is created.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	June 6, 2017	AWS::AutoScaling::LifecycleHook The NotificationTargetARN and RoleARN properties are now optional. AWS::CloudWatch::Alarm You can now use the EvaluateLowSampleCountPercentile , ExtendedStatistic , and TreatMissingData properties when creating AWS::CloudWatch::Alarm resources. AWS::EC2::SpotFleet CloudFormation supports mutable changes to Spot fleet properties. The following properties of the SpotFleetRequestConfigData property support Replacement updates: • AllocationStrategy • IamFleetRole • LaunchSpecifications • SpotPrice • TerminateInstancesWithExpiration • ValidFrom • ValidUntil The following properties of the SpotFleetRequestConfigData property support No interruption updates: • ExcessCapacityTerminationPolicy • TargetCapacity	2010-05-15

Change	Release Date	Description	API Version
		AWS::EMR::InstanceGroupConfig CloudFormation now supports Auto Scaling for Amazon EMR task instance groups. AWS::Events::Rule The `RoleArn` property is deprecated on the `Rule` resource. Use the `RoleArn` property on the `Target` property type to specify the IAM role to use for a target. AWS::Kinesis::Stream The `ShardCount` property now supports `No interruption` updates. AWS::Lambda::Function Use the `TracingConfig` property to configure tracing settings for Lambda functions. AWS::Redshift::Cluster, AWS::Redshift::ClusterParameterGroup, AWS::Redshift::ClusterSecurityGroup, and AWS::Redshift::ClusterSubnetGroup Use the `Tags` property to specify resource tags. AWS::RDS::DBCluster Added the `ReadEndpoint.Address` attribute to the `Fn::GetAtt` intrinsic function. AWS::S3::Bucket Added the `Arn` attribute to the `Fn::GetAtt` intrinsic function.	

Change	Release Date	Description	API Version
New resources	May 11, 2017	The following new resources support using AWS WAF with Elastic Load Balancing (ELB) Application Load Balancers. AWS::WAFRegional::ByteMatchSet Use the `AWS::WAFRegional::ByteMatchSet` resource to identify a part of a web request that you want to inspect. AWS::WAFRegional::IPSet Use the `AWS::WAFRegional::IPSet` resource to specify which web requests to permit or block based on the IP addresses from which the requests originate. AWS::WAFRegional::Rule Use the `AWS::WAFRegional::Rule` resource to specify a combination of `IPSet`, `ByteMatchSet` , and `SqlInjectionMatchSet` objects that identify the web requests to allow, block, or count. AWS::WAFRegional::SizeConstraintSet Use the `AWS::WAFRegional::SizeConstraintSet` resource to specify a size constraint used to check the size of a web request and which parts of the request to check. AWS::WAFRegional::SqlInjectionMatchSet Use the `AWS::WAFRegional::SqlInjectionMatchSet` resource to allow, block, or count requests that contain malicious SQL code in a specific part of web requests.	2010-05-15

Change	Release Date	Description	API Version
		AWS::WAFRegional::WebACL Use the `AWS::WAFRegional::WebACL` resource to identify the web requests that you want to allow, block, or count. AWS::WAFRegional::WebACLAssociation Use the `AWS::WAFRegional::WebACLAssociation` resource to associate a web access control group (ACL) with a resource. AWS::WAFRegional::XssMatchSet Use the `AWS::WAFRegional::XssMatchSet` resource to specify the parts of web requests that you want AWS WAF to inspect for cross-site scripting attacks and the name of the header to inspect.	

Change	Release Date	Description	API Version
New resources	April 28, 2017	AWS::Cognito::IdentityPool Use the `AWS::Cognito::IdentityPool` resource to create an Amazon Cognito identity pool. AWS::Cognito::IdentityPoolRoleAttachment Use the `AWS::Cognito::IdentityPoolRoleAttachment` resource to manage the role configuration for an Amazon Cognito identity pool. AWS::Cognito::UserPool Use the `AWS::Cognito::UserPool` resource to create an Amazon Cognito user pool. AWS::Cognito::UserPoolClient Use the `AWS::Cognito::UserPoolClient` resource to create a user pool client. AWS::Cognito::UserPoolGroup Use the `AWS::Cognito::UserPoolGroup` resource to create a user group in an Amazon Cognito user pool. AWS::Cognito::UserPoolUser Use the `AWS::Cognito::UserPoolUser` resource to create an Amazon Cognito user pool user. AWS::Cognito::UserPoolUserToGroupAttachment Use the `AWS::Cognito::UserPoolUserToGroupAttachment` resource to attach a user to an Amazon Cognito user pool group.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	April 28, 2017	SourceDetails Use the `MaximumExecutionFrequency` subproperty of the `AWS::Config::ConfigRule` resource to run evaluations for a custom rule using a periodic trigger. AWS::EC2::Volume We now support Elastic Volumes for Amazon Elastic Block Store (Amazon EBS) in CloudFormation. We now support `No interruption` updates on three properties: `VolumeType` , `Size`, and `Iops`. AWS::EC2::SecurityGroup Use the `GroupName` property to specify a name for your Amazon EC2 security group. AWS::ECS::Service There are three new properties for `AWS::ECS::Service` : `PlacementConstraints` , `PlacementStrategies` , and `ServiceName` . AWS::ECS::TaskDefinition Use the `PlacementConstraints` property to define placement constraints for tasks in the service. AWS::ElastiCache::ReplicationGroup Added the `ConfigurationEndPoint.Address` attribute and the `ConfigurationEndPoint.Port` attribute to the `Fn::GetAtt` intrinsic function.	2010-05-15

Change	Release Date	Description	API Version
		AWS::ElasticLoadBalancingV2::LoadBalancer Use the `IpAddressType` property to specify the type of IP addresses that are used by the load balancer's subnets. AWS::EMR::Cluster CloudFormation now supports Auto Scaling for Amazon EMR clusters. AWS::IAM::ManagedPolicy Use the `ManagedPolicyName` property to specify a custom name for your IAM managed policy. AWS::Lambda::Function Use the `Tags` property to add tags to your Lambda function. AWS::OpsWorks::Instance Added the following attributes to the `Fn::GetAtt` intrinsic function: `AvailabilityZone`, `PrivateDnsName`, `PrivateIp`, and `PublicDnsName`. AWS::OpsWorks::UserProfile Use the `SshUsername` property to specify a user's SSH name. Added the `SshUsername` attribute to the `Fn::GetAtt` intrinsic function. AWS::Redshift::Cluster Use the `IamRoles` property to provide a list of one or more AWS Identity and Access Management roles	

Change	Release Date	Description	API Version
		that the Amazon Redshift cluster can use to access other AWS services.	
Edit templates in YAML and JSON using CloudFormation Designer	April 6, 2017	When you create CloudFormation templates using Designer, you can now edit your template in both YAML and JSON in the integrated editor. You can also convert JSON templates to YAML and vice-versa, depending on your preferred template authoring language. For more information, see What Is CloudFormation Designer?.	2010-05-15
New resource	April 6, 2017	AWS::SSM::Parameter Use the `AWS::SSM::Parameter` resource to create an SSM parameter in Parameter Store.	2010-05-15
`AWS::Include` transform	March 28, 2017	Use the `AWS::Include` transform to reference reusable snippets stored in an Amazon S3 bucket. For more information, see AWS::Include Transform.	2010-05-15
Peer your Amazon VPC with another account	March 28, 2017	You can now use CloudFormation to peer your Amazon VPC with a VPC in another AWS account. For more information, see Peer with an Amazon VPC in Another AWS Account.	2010-05-15
New resource	March 28, 2017	AWS::ApiGateway::UsagePlanKey Use the `AWS::ApiGateway::UsagePlanKey` resource to associate a usage plan key and determine which users the usage plan is applied to.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	March 28, 2017	AWS::EC2::VPCPeeringConnection Use the `PeerOwnerId` property and the `PeerRoleArn` property to peer with a VPC in another AWS account. For more information, see Peer with an Amazon VPC in Another AWS Account. AWS::IAM::InstanceProfile Use the `InstanceProfileName` property to configure an instance profile. AWS::Lambda::Function Use the `DeadLetterConfig` property to configure how AWS Lambda handles events that it can't process. Node.js v0.10 is no longer supported for the `Runtime` property. AWS::Route53::HealthCheck There are seven new resource subproperty types for the HealthCheckConfig `HealthCheckConfig` property: `AlarmIdentifier`, `ChildHealthChecks`, `EnableSNI`, `HealthThreshold`, `InsufficientDataHealthStatus`, `Inverted`, and `MeasureLatency`. AWS::SQS::Queue Use the `ContentBasedDeduplication` and `FifoQueue` properties to create First-In-First-Out (FIFO) Amazon Simple Queue Service queues.	2010-05-15

Change	Release Date	Description	API Version
		AWS::S3::Bucket You can now specify IPv6 domain names for your Amazon S3 buckets.	
New resources	February 10, 2017	AWS::StepFunctions::Activity Use the `AWS::StepFunctions::Activity` resource to create an AWS Step Functions activity. AWS::StepFunctions::StateMachine Use the `AWS::StepFunctions::StateMachine` resource to create a Step Functions state machine.	2010-05-15
New intrinsic function	January 17, 2017	Use the `Fn::Split` function to split a string into a list of string values. For more information, see Fn::Split.	2010-05-15
Console support for listing imports	January 17, 2017	Use the CloudFormation console to see all of the stacks that are importing an exported output value. For more information, see Listing Stacks That Import an Exported Output Value.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	January 17, 2017	AWS::AutoScaling::AutoScalingGroup The `LoadBalancerNames` property can be updated without replacing the Auto Scaling group. AWS::ECS::TaskDefinition Added the `NetworkMode` and `MemoryReservation` properties. AWS::RDS::DBCluster CloudFormation supports updates to the `Tags` property. AWS::RDS::DBInstance Added the `Timezone` property. FirehoseAction Added the `Separator` property. AWS::OpsWorks::Instance Added the `PublicIp` attribute for the `Fn::GetAtt` intrinsic function.	2010-05-15

Change	Release Date	Description	API Version
New resources	December 01, 2016	AWS::CodeBuild::Project Use the `AWS::CodeBuild::Project` resource to create an AWS CodeBuild project that defines how CodeBuild builds your source code. AWS::SSM::Association Use the `AWS::SSM::Association` resource to associate an Amazon EC2 Systems Manager document with EC2 instances. AWS::EC2::SubnetCidrBlock Use the `AWS::EC2::SubnetCidrBlock` resource to associate a single IPv6 CIDR block with an Amazon VPC subnet. AWS::EC2::VPCCidrBlock Use the `AWS::EC2::VPCCidrBlock` resource to associate a single Amazon-provided IPv6 CIDR block with an Amazon VPC.	2010-05-15

Change	Release Date	Description	API Version
Updated resources for IPv6 support	December 01, 2016	AWS::EC2::Instance Added the `Ipv6AddressCount` and `Ipv6Addresses` properties. AWS::EC2::NetworkAclEntry Added the `Ipv6CidrBlock` property. AWS::EC2::NetworkInterface Added the `Ipv6AddressCount` and `Ipv6Addresses` properties. AWS::EC2::Route Added the `DestinationIpv6CidrBlock` property. AWS::EC2::SecurityGroupEgress Added the `CidrIpv6` property. AWS::EC2::SecurityGroupIngress Added the `CidrIpv6` property. AWS::EC2::SpotFleet Added the `Ipv6AddressCount` and `Ipv6Addresses` properties for the launch specification network interfaces. AWS::EC2::Subnet Added the `Ipv6CidrBlocks` attribute for the `Fn::GetAtt` function. AWS::EC2::VPC Added the `Ipv6CidrBlocks` attribute for the `Fn::GetAtt` function.	2010-05-15

Change	Release Date	Description	API Version
		AWS::SSM::Document Added the `DocumentType` property.	
Resource specification	November 22, 2016	Use the CloudFormation resource specification to builds tools that help you create CloudFormation templates . The specification is a machine-readable, JSON-formatted text file. For more information, see CloudFormation Resource Specification.	2010-05-15
New resources	November 22, 2016	AWS::OpsWorks::UserProfile Use the `AWS::OpsWorks::UserProfile` resource to configure SSH access for users who require access to instances in an OpsWorks stack. AWS::OpsWorks::Volume Use the `AWS::OpsWorks::Volume` resource to register an Amazon Elastic Block Store volume with an OpsWorks stack.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	November 22, 2016	AWS::OpsWorks::App Added the DataSources property. AWS::OpsWorks::Instance Added the BlockDeviceMappings, AgentVersion, ElasticIps, Hostname, Tenancy, and Volumes properties. AWS::OpsWorks::Layer Added the CustomJson and VolumeConfigurations properties. AWS::OpsWorks::Stack Added the ElasticIps, EcsClusterArn, RdsDbInstances, CloneAppIds, ClonePermissions, and SourceStackId properties. AWS::RDS::DBInstance Added the CopyTagsToSnapshot property.	2010-05-15
List imports	November 22, 2016	List imports of an exported output value to track which CloudFormation stacks are importing the value. For more information, see Listing Stacks That Import an Exported Output Value.	2010-05-15
Transforms	November 17, 2016	Specify the AWS Serverless Application Model (AWS SAM) that CloudFormation uses to process AWS SAM syntax for serverless applications. For more information, see Transform.	2010-05-15

Change	Release Date	Description	API Version
New resource	November 17, 2016	AWS::SNS::Subscription Use the `AWS::SNS::Subscription` resource to subscribe an endpoint to an Amazon Simple Notification Service topic.	2010-05-15
Updated resource	November 17, 2016	AWS::Lambda::Function Use the `Environment` property to specify key-value pairs (environment variables) that your AWS Lambda function can access. Use the `KmsKeyArn` property to specify an KMS key that AWS Lambda uses to encrypt and decrypt environment variables.	2010-05-15
New CLI commands	November 17, 2016	Uploading Local Artifacts to an S3 Bucket Use the `package` command to upload local artifacts that are referenced in a CloudFormation template to an S3 bucket. Quickly Deploying Templates with Transforms Use the `deploy` command to combine the create and execute change set actions into a single command. This command is useful for quickly creating or updating stacks that contain transforms.	2010-05-15

Change	Release Date	Description	API Version
Updated resource	November 03, 2016	AWS::CloudFront::Distribution For the DistributionConfig property, use the `HttpVersion` property to specify the latest HTTP version that viewers can use to communicate with Amazon CloudFront. For the ForwardedValues property, use the `QueryStringCacheKeys` property to specify the query string parameters that CloudFront uses to determine which content to cache.	2010-05-15
List stack exports	November 03, 2016	Use the CloudFormation console, API, or AWS CLI to see a list of all the exported output values for a region. For more information, see Exporting Stack Output Values.	2010-05-15
Continuous delivery with stacks	November 03, 2016	Use AWS CodePipeline to build continuous delivery workflows with CloudFormation stacks. For more information, see Continuous Delivery with CodePipeline.	2010-05-15
Skip resources during rollback	November 03, 2016	If you have a stack in the `UPDATE_ROLLBACK_FAILED` state, use the `ResourcesToSkip` parameter for the `ContinueUpdateRollback` action to skip resources that CloudFormation can't rollback. For more information, see the Troubleshooting section in Update Rollback Failed.	2010-05-15
Change sets enhancement	November 03, 2016	You can create a new stack using a change set.	2010-05-15

Change	Release Date	Description	API Version
Updated resource	October 12, 2016	AWS::ElastiCache::CacheCluster Update the `CacheNodeType` property without replacing the cluster. AWS::ElastiCache::ReplicationGroup You can create a Redis (cluster mode enabled) replication group that can contain multiple node groups (shards), each with a primary cluster and read replicas. AWS::ElastiCache::SubnetGroup Use the `CacheSubnetGroupName` property to specify a name for an Amazon ElastiCache subnet group.	2010-05-15
New resources	October 06, 2016	AWS::ApiGateway::UsagePlan Use the `AWS::ApiGateway::UsagePlan` resource to specify a usage plan for deployed Amazon API Gateway APIs. AWS::CodeCommit::Repository Use the `AWS::CodeCommit::Repository` resource to create an CodeCommit repository that's hosted by Amazon Web Services.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	October 06, 2016	AWS::ApiGateway::Authorizer Use the `ProviderARNs` property to use Amazon Cognito user pools as Amazon API Gateway API authorizers. AWS::ApiGateway::Deployment The `StageName` property is no longer required. AWS::ElasticLoadBalancingV2::TargetGroup For the `GetAtt` function, use the `LoadBalancerArns` attribute to retrieve the Amazon Resource Names (ARNs) of the load balancers that route traffic to the target group. AWS::RDS::DBInstance Use the `Domain` and `DomainIAMRoleName` properties to use Windows Authentication when users connect to the RDS DB instance. AWS::EC2::SecurityGroupEgress Use the `DestinationPrefixListId` property to specify the AWS service prefix of an Amazon VPC endpoint.	2010-05-15
Cross-stack reference enhancement	October 06, 2016	Use intrinsic functions to customize the `Name` value of an export or to refer to a value in the `ImportValue` function.	2010-05-15
CloudFormation service role	September 26, 2016	Use an AWS Identity and Access Management (IAM) service role for CloudFormation stack operations. CloudFormation uses the role's credentials to make calls to stack resources on your behalf. For more information, see CloudFormation service role.	2010-05-15

Change	Release Date	Description	API Version
New feature	September 19, 2016	You can use the `Export` output field and the `Fn::ImportValue` intrinsic function to have one stack refer to resource outputs in another stack. For more information, see Outputs, Fn::ImportValue, and Walkthrough: Refer to Resource Outputs in Another CloudFormation Stack.	2010-05-15
YAML support	September 19, 2016	You can use the YAML format to author CloudFormation templates. YAML also allows you to, for example, add comments to your templates or use the short form for intrinsic functions. For more information, see CloudFormation template format.	2010-05-15
New intrinsic function	September 19, 2016	Use the `Fn::Sub` function to substitute variables in an input string with values that you specify. For more information, see Fn::Sub.	2010-05-15
New resources	September 19, 2016	AWS::KMS::Alias Use the `AWS::KMS::Alias` resource to create an alias for an AWS KMS key.	

Change	Release Date	Description	API Version
Updated resources	September 19, 2016	AWS::EC2::SpotFleet For the `LaunchSpecifications` property, use the `SpotPrice` property to specify a bid price for a specific instance type. AWS::ECS::Cluster Use the `ClusterName` property to specify a name for an Amazon Elastic Container Service cluster. AWS::ECS::TaskDefinition Use the `TaskRoleArn` property to specify an AWS Identity and Access Management role that Amazon Elastic Container Service containers use to make AWS calls on your behalf. Use the `Family` property to register a task definitio n to a specific family. AWS::Elasticsearch::Domain Use the `ElasticsearchVersion` property to specify which version of Elasticsearch to use.	2010-05-15
New resources	August 11, 2016	Use the following Elastic Load Balancing Application Load Balancer resources to distribute incoming applicati on traffic to multiple targets, such as EC2 instances, in multiple Availability Zones: • AWS::ElasticLoadBalancingV2::Listener • AWS::ElasticLoadBalancingV2::ListenerRule • AWS::ElasticLoadBalancingV2::LoadBalancer • AWS::ElasticLoadBalancingV2::TargetGroup	2010-05-15

Change	Release Date	Description	API Version
Updated resource	August 11, 2016	AWS::AutoScaling::AutoScalingGroup Use the `TargetGroupARNs` property to associate the Auto Scaling group with one or more Application Load Balancer target groups. AWS::ECS::Service For the load `LoadBalancers` property, use the `TargetGroupArn` property to associate an Amazon Elastic Container Service service with an Application Load Balancer target group.	2010-05-15
New resources	August 09, 2016	CloudFormation added the following resources: AWS::ApplicationAutoScaling::ScalableTarget and AWS::ApplicationAutoScaling::ScalingPolicy Use an Application Auto Scaling scaling policy to define when and how a target resource scales. AWS::CertificateManager::Certificate Provision an AWS Certificate Manager certificate that you can use with other AWS services to enable secure connections.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	August 09, 2016	CloudFormation updated the following resources: AWS::CloudFront::Distribution For the distribution configuration `ViewerCertificate` property, you can specify an AWS Certificate Manager certificate. For the distribution configuration `Origin` property, you can specify custom headers and the SSL protocols for custom origins. AWS::EFS::FileSystem You can specify the performance mode for an Amazon Elastic File System file system.	2010-05-15
New resources	July 20, 2016	AWS IoT Use AWS IoT to declare an AWS IoT policy, an X.509 certificate, an association between a policy and a principal (an X.509 certificate or other credential), an AWS IoT thing, an association between a principal and a thing, or an AWS IoT rule. • AWS::IoT::Certificate • AWS::IoT::Policy • AWS::IoT::PolicyPrincipalAttachment • AWS::IoT::Thing • AWS::IoT::ThingPrincipalAttachment • AWS::IoT::TopicRule	2010-05-15

Change	Release Date	Description	API Version
Updated resources	July 20, 2016	CloudFormation updated the following resources: AWS::IAM::Group, AWS::IAM::Role, AWS::IAM::User Use the name properties to specify a custom name for AWS Identity and Access Management (IAM) resources. AWS::ApiGateway::Method For the `Integration` property, you can use the `PassthroughBehavior` property to specify when Amazon API Gateway passes requests to the targeted back end. AWS::ApiGateway::Model and AWS::ApiGateway::RestApi You can specify JSON objects for the `Schema` and `Body` properties.	2010-05-15
Auto Scaling group UpdatePolicy	June 9, 2016	For the `UpdatePolicy` attribute, use the `AutoScalingReplacingUpdate` property to specify whether an Auto Scaling group and the instances it contains are replaced when you update the Auto Scaling group. During a replacement, CloudFormation retains the old Auto Scaling group until it creates the new one successfully so that CloudFormation can roll back to the old Auto Scaling group if the update fails. For more information, see UpdatePolicy Attribute.	2010-05-15

Change	Release Date	Description	API Version
New resource	June 9, 2016	CloudFormation added the following resources: AWS::EC2::FlowLog Creates an Amazon Elastic Compute Cloud flow log that captures IP traffic for a specified network interface, subnet, or VPC. AWS::KinesisFirehose::DeliveryStream Creates a delivery stream that delivers real-time streaming data to a destination, such as Amazon Simple Storage Service, Amazon Redshift, or Amazon OpenSearch Service.	2010-05-15
Updated resources	June 9, 2016	CloudFormation updated the following resources: AWS::Kinesis::Stream Use the `Name` property to specify a name for an Amazon Kinesis stream. AWS::Lambda::Function For the `Code` property, you can use the `ZipFile` property and cfn response module for `nodejs4.3` runtime environments. AWS::SNS::Topic CloudFormation enabled updates for the Amazon Simple Notification Service topic resource.	2010-05-15
New resource	April 25, 2016	Use the AWS::EC2::Host resource to allocate a fully dedicated physical server for launching EC2 instances.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	April 25, 2016	AWS::EC2::Instance Use the `Affinity` and `HostId` properties to launch instances onto an Amazon Elastic Compute Cloud dedicated host. AWS::ECS::Service Use the `DeploymentConfiguration` property to configure how many tasks can run during a deployment. AWS::ECS::TaskDefinition CloudFormation added support for additional Amazon Elastic Container Service container definitio n properties. AWS::GameLift::Fleet Use the `MaxSize` and `MinSize` properties to specify the maximum and minimum number of EC2 instances allowed in your Amazon GameLift Servers fleet. AWS::Lambda::Function Use the `FunctionName` property to specify a name for your AWS Lambda function. You can also use Python 2.7 to specify an inline function.	2010-05-15

Change	Release Date	Description	API Version
New resources	April 18, 2016	Amazon API Gateway Use the Amazon API Gateway resources to publish, maintain, and monitor APIs at any scale. You can create APIs that clients can call to access your back-end services, such as applications running EC2 instances or code running on AWS Lambda. • AWS::ApiGateway::Account • AWS::ApiGateway::ApiKey • AWS::ApiGateway::Authorizer • AWS::ApiGateway::BasePathMapping • AWS::ApiGateway::ClientCertificate • AWS::ApiGateway::Deployment • AWS::ApiGateway::Method • AWS::ApiGateway::Model • AWS::ApiGateway::Resource • AWS::ApiGateway::RestApi • AWS::ApiGateway::Stage AWS::Events::Rule Create an Amazon CloudWatch Events rule that monitors changes to AWS resources in your account (events). If an incoming event matches the conditions that you described in the rule, Amazon CloudWatch Events sends messages to and activates your specified targets, such as AWS Lambda functions or Amazon Simple Notification Service topics.	2010-05-15

Change	Release Date	Description	API Version
		AWS::WAF::SizeConstraintSet and AWS::WAF::XssMatchSet Use the two AWS WAF rules to check the size of a web request or to prevent cross-site scripting attacks.	
New resources	March 31, 2016	Use the AWS::Lambda::Alias resource to create aliases for your AWS Lambda functions and the AWS::Lambda::Version resource to create versions of your functions.	2010-05-15
Updated resources	March 31, 2016	CloudFormation updated the following resources: AWS::EMR::Cluster and AWS::EMR::InstanceGroupConfig Use the `EbsConfiguration` property to configure Amazon Elastic Block Store storage volumes for your Amazon EMR clusters or instance groups. AWS::Lambda::Function Use the `VpcConfig` property to enable AWS Lambda functions to access resources in a VPC. AWS::S3::Bucket For the Amazon Simple Storage Service life cycle rules, you can specify multiple transition rules that specify when objects transition to a specified storage class.	2010-05-15
Change sets	March 29, 2016	Before updating stacks, use change sets to see how your changes might affect your running resources. For more information, see Updating Stacks Using Change Sets.	2010-05-15

Change	Release Date	Description	API Version
New resources	March 15, 2016	Use the AWS::GameLift::Alias, AWS::GameLift::Build, and AWS::GameLift::Fleet resources to deploy multiplayer game servers in AWS.	2010-05-15
New resources	February 26, 2016	CloudFormation added the following resources: AWS::ECR::Repository Create Amazon Elastic Container Registry repositories where users can push and pull Docker images. AWS::EC2::NatGateway Use the network address translator (NAT) gateway to enable EC2 instances in a private subnet to connect to the Internet. AWS::Elasticsearch::Domain Create Amazon OpenSearch Service domains that run legacy Elasticsearch OSS clusters. AWS::EMR::Cluster, AWS::EMR::InstanceGroupConfig, AWS::EMR::Step Use the Amazon EMR resources to assist you analyze and process vast amounts of data. You can create clusters and then run jobs on them.	2010-05-15

Change	Release Date	Description	API Version
Updated resources	February 26, 2016	CloudFormation updated the following resources: AWS::CloudTrail::Trail Use the `IsMultiRegionTrail` property to specify whether to create an AWS CloudTrail trail in the region in which you create a stack or in all regions. AWS::Config::ConfigurationRecorder For the recording group, use the `IncludeGlobalResourceTypes` property to record all global resource types. AWS::RDS::DBCluster Use the `KmsKeyId` and `StorageEncrypted` properties to encrypt database instances in the cluster.	2010-05-15
Retain resources	February 26, 2016	For stacks in the `DELETE_FAILED` state, use the `RetainResources` parameter to retain resources that CloudFormation can't delete. For more information, see Delete Stack Fails.	2010-05-15
Update stack tags	February 26, 2016	You can add, modify, or remove stack tags when you update a stack. For more information, see CloudFormation Stacks Updates.	2010-05-15
Continue rolling back failed update rollbacks	January 25, 2016	For a stack in the `UPDATE_ROLLBACK_FAILED` state, you can continue rolling back the update to get your stack in a working state. That way, you can return the stack to its original settings and try to update it again. For more information, see Continue Rolling Back an Update.	2010-05-15

Change	Release Date	Description	API Version
New sample templates available for the Asia Pacific (Seoul) region.	January 7, 2016	The following collection of CloudFormation sample templates are for the ap-northeast-2 region: • Sample Solutions • Application Frameworks • Services For more information, see Working with CloudFormation templates.	2010-05-15
New resources	December 28, 2015	CloudFormation added the following resources: AWS::DirectoryService::MicrosoftAD Use the Microsoft Active Directory resource to create a Microsoft Active Directory directory in AWS. AWS::Logs::Destination and AWS::Logs::LogStream Use the Amazon CloudWatch Logs resources to create a destination for real-time processing of log data or to create log streams, respectively. AWS::WAF::ByteMatchSet, AWS::WAF::IPSet, AWS::WAF::Rule, AWS::WAF::SqlInjectionMatchSet, and AWS::WAF::WebACL Use the AWS WAF resources to control and monitor web requests to your content.	2010-05-15

Change	Release Date	Description	API Version
Resource updates	December 28, 2015	CloudFormation updated the following resources: AWS::CloudFront::Distribution For the distribution configuration, use the `WebACLId` property to associate an AWS WAF web access control list (ACL) with an Amazon CloudFront distribution. For the cache behavior and default cache behavior, you can specify a default and maximum Time to Live (TTL) value. AWS::DynamoDB::Table You can create, update, or delete a global secondary index without replacing your Amazon DynamoDB table. AWS::S3::Bucket Use the `ReplicationConfiguration` property to specify which objects to replicate and where they are stored. Use the properties in the `NotificationConfiguration` property to specify filters so that Amazon Simple Storage Service sends notifications for objects that you specify.	2010-05-15
Parameter grouping and sorting	December 3, 2015	Use the AWS::CloudFormation::Interface metadata key to group and sort parameters in the CloudFormation console when users create or update a stack with your template.	2010-05-15
Update policy attribute	December 3, 2015	For an Auto Scaling update policy attribute, use the `MinSuccessfulInstancesPercent` property to specify the percentage of instances that must signal success for a successful update.	2010-05-15

Change	Release Date	Description	API Version
New resources	December 3, 2015	CloudFormation added the following resources: AWS::CodePipeline::Pipeline and AWS::CodePipeline::CustomActionType Use the CodePipeline resources to create a pipeline that describes how software changes go through a release process. AWS::Config::ConfigurationRecorder, AWS::Config::DeliveryChannel, and AWS::Config::ConfigRule Use the AWS Config resources to monitor configuration changes to specific AWS resources. AWS::KMS::Key Use the AWS Key Management Service (AWS KMS) resource to create customer managed keys in AWS KMS that users can use to encrypt small amounts of data. AWS::SSM::Document Use the Amazon EC2 Systems Manager to create a document that specifies on-instance configurations.	2010-05-15

Change	Release Date	Description	API Version
Resources update	December 3, 2015	CloudFormation updated the following resources: AWS::AutoScaling::LaunchConfiguration Specify whether EBS volumes are encrypted. AWS::AutoScaling::ScalingPolicy You can use two different policy types (simple and step scaling) to specify how an Auto Scaling group scales when an Amazon CloudWatch (CloudWatch) alarm is breached. AWS::CloudTrail::Trail Use the CloudWatch properties to send logs to a CloudWatch log group. You can add tags to a trail and specify an AWS KMS key that you want to use to encrypt logs. AWS::CodeDeploy::Application, AWS::CodeDeploy::DeploymentConfig, and AWS::CodeDeploy::DeploymentGroup Use the `ApplicationName` , `DeploymentConfigName` , and `DeploymentGroupName` properties to specify custom names for CodeDeploy resources. AWS::DynamoDB::Table Use the `StreamSpecification` property to specify settings for capturing changes to items stored in an Amazon DynamoDB (DynamoDB) table. AWS::EC2::Instance Use the `SsmAssociations` property to associate an Amazon EC2 Systems Manager document with an instance.	2010-05-15

Change	Release Date	Description	API Version
		AWS::EC2::SpotFleet Use the `AllocationStrategy` property to specify how to allocate target capacity across Spot pools. Use the `ExcessCapacityTerminationPolicy` property to specify how instances are terminated if the target capacity is below the size of the Spot fleet. AWS::Redshift::Cluster Use the `KmsKeyId` property to specify an AWS KMS key to encrypt data in an Amazon Redshift cluster. AWS::WorkSpaces::Workspace Use the encryption properties to encrypt data stored on volumes.	
Resource update	November 4, 2015	For the AWS::EC2::Volume resource, use the `AutoEnableIO` property to automatically resume I/O operations if a volume's data becomes inconsistent.	2010-05-15

Change	Release Date	Description	API Version
New resources	October 1, 2015	CloudFormation added the following resources: AWS::CodeDeploy::Application, AWS::CodeDeploy::DeploymentGroup, and AWS::CodeDeploy::DeploymentConfig Use the CodeDeploy resources to create and apply deployments to EC2 or on-premises instances. AWS::DirectoryService::SimpleAD Use the Simple Active Directory resource to create an Directory Service Simple AD, which is a Microsoft Active Directory-compatible directory. AWS::EC2::PlacementGroup Use a placement group to create a cluster of instances in a low-latency network. AWS::EC2::SpotFleet Use a Spot fleet to launch a collection of Spot instances that run interruptible tasks. AWS::Lambda::EventSourceMapping Use the event source mapping resource to specify a stream as an event source for an AWS Lambda (Lambda) function. AWS::Lambda::Permission Use a Lambda permission to add a statement to a Lambda function's policy. AWS::Logs::SubscriptionFilter Use the subscription filter to define which log events are delivered to your Kinesis stream.	2010-05-15

Change	Release Date	Description	API Version
		AWS::RDS::DBCluster and AWS::RDS::DBClusterParameterGroup Use the cluster and cluster parameter group resources to create an Amazon Aurora DB cluster. AWS::WorkSpaces::Workspace Use WorkSpaces to create cloud-based desktop experiences.	
Resource updates	October 1, 2015	CloudFormation updated the following resources: AWS::ElastiCache::ReplicationGroup Use the `Fn::GetAtt` intrinsic function to get a list of read-only replica addresses and ports. AWS::OpsWorks::Stack Use the `AgentVersion` property to specify a particular OpsWorks agent. AWS::OpsWorks::App Use the `Environment` property to specify environment variables for an OpsWorks app. AWS::S3::Bucket For the NotificationConfiguration property, you can configure notification settings for Lambda functions and Amazon Simple Queue Service (Amazon SQS) queues.	2010-05-15
IAM condition keys	October 1, 2015	For AWS Identity and Access Management (IAM) policies, use CloudFormation-specific condition keys to specify when an IAM policy takes effect. For more information, see Controlling Access with AWS Identity and Access Management.	2010-05-15

Change	Release Date	Description	API Version
CloudFormation Designer	October 1, 2015	Use CloudFormation Designer to create and modify templates using a drag-and-drop interface.	2010-05-15
New resource	August 24, 2015	Use the AWS::EC2::VPCEndpoint resource to establish a private connection between your VPC and another AWS service.	2010-05-15
Resource updates	August 24, 2015	CloudFormation updated the following resources: AWS::ElasticBeanstalk::Environment Use the `Tags` property to specify tags (key-value pairs) for an AWS Elastic Beanstalk (Elastic Beanstalk) environment. AWS::Lambda::Function For the Code property, use the `ZipFile` property to write the source code of your Lambda function directly in a template. Currently, you can use the `ZipFile` property only for `nodejs` runtime environments. You can still point to a file in an S3 bucket for all runtime environments, such as `java8` and `nodejs`. AWS::OpsWorks::Instance Use the `EbsOptimized` property to indicate whether an instance is optimized for Amazon Elastic Block Store (Amazon EBS) I/O. AWS::RDS::DBInstance For the `SourceDBInstanceIdentifier` property, you can specify a database instance in another region to create a cross-region read replica.	2010-05-15

Change	Release Date	Description	API Version
Amazon S3 template URL	August 24, 2015	For versioning-enabled buckets, you can specify a version ID in an Amazon S3 template URL when you create or update a stack, such as `https://s3.amazonaws.com/templates/myTemplate.template?versionId=123ab1cdeKdOW5IH4GAcYbEngcpTJTDW` .	2010-05-15
New resource	August 3, 2015	Use the AWS::EFS::FileSystem resource to create an Amazon Elastic File System (Amazon EFS) file system and the AWS::EFS::MountTarget resource to create a mount point for a file system.	2010-05-15
Permission requirement change	June 11, 2015	When you create or update an AWS::RDS::DBInstance resource, you must now also have permission to call the `ec2:DescribeAccountAttributes` action.	2010-05-15

Change	Release Date	Description	API Version
New resources	June 11, 2015	CloudFormation added the following resources: AWS::DataPipeline::Pipeline Use data pipelines to automate the movement and transformation of data. Amazon Elastic Container Service resources Use the AWS::ECS::Service, AWS::ECS::Cluster, and AWS::ECS::TaskDefinition resources to create Docker containers on a cluster of EC2 instances. AWS::ElastiCache::ReplicationGroup Use replication groups to create a collection of nodes with one primary read-write cluster and a maximum of five secondary read-only clusters. AWS::IAM::ManagedPolicy Use managed policies to create policies in your AWS account that you can use to apply permissions to IAM users, groups, and roles. AWS::Lambda::Function Use Lambda functions to run code in response to events. AWS::RDS::OptionGroup Use option groups to help you create and manage Amazon Relational Database Service (Amazon RDS) databases.	2010-05-15

Change	Release Date	Description	API Version
Resource updates	June 11, 2015	CloudFormation updated the following resources: AWS::EC2::Subnet Use the `MapPublicIpOnLaunch` property to automatically assign public IP addresses to instances in a subnet. AWS::ElastiCache::CacheCluster Use the `SnapshotName` property to restore snapshot data into a new Redis cache cluster. AWS::IAM::User For the `LoginProfile` property, use the `PasswordResetRequired` property so that users are required to set a new password when they log in to the AWS Management Console. AWS::OpsWorks::Layer Use the `LifecycleEventConfiguration` property to configure lifecycle events for an OpsWorks layer. AWS::S3::Bucket For the `LifecycleConfiguration` property, use the `NoncurrentVersionExpirationInDays` and `NoncurrentVersionTransition` properties to specify lifecycle rules for non-current object versions.	2010-05-15

Change	Release Date	Description	API Version
New parameter types	May 19, 2015	Whenever you use the CloudFormation console to create or update a stack, you can search for AWS-specific parameter type values by ID, name, or Name tag value. CloudFormation also added support for the following AWS-specific parameter types. For more information, see Parameters. • `AWS::EC2::AvailabilityZone::Name` • `List<AWS::EC2::AvailabilityZone::Name>` • `AWS::EC2::Instance::Id` • `List<AWS::EC2::Instance::Id>` • `AWS::EC2::Image::Id` • `List<AWS::EC2::Image::Id>` • `AWS::EC2::SecurityGroup::GroupName` • `List<AWS::EC2::SecurityGroup::GroupName>` • `AWS::EC2::Volume::Id` • `List<AWS::EC2::Volume::Id>` • `AWS::Route53::HostedZone::Id` • `List<AWS::Route53::HostedZone::Id>`	2010-05-15

Change	Release Date	Description	API Version
New resources	April 16, 2015	CloudFormation added the following resources: AWS::AutoScaling::LifecycleHook Use Auto Scaling lifecycle hooks to control the state of an instance after it is launched or terminated. AWS::RDS::EventSubscription Use event subscriptions to get notifications about Amazon RDS events.	2010-05-15

Change	Release Date	Description	API Version
Resource updates	April 16, 2015	CloudFormation updated the following resources: AWS::AutoScaling::AutoScalingGroup Use the `NotificationConfigurations` property to specify multiple notifications. AWS::AutoScaling::LaunchConfiguration Use the `PlacementTenancy` property to specify the tenancy of instances. Use the `ClassicLinkVPCId` and `ClassicLinkVPCSecurityGroups` properties to link EC2-Classic instances to a ClassicLink-enabled VPC. AWS::AutoScaling::ScalingPolicy Use the `MinAdjustmentStep` property to specify the minimum number of instances that are added or removed during a scaling event. AWS::CloudFront::Distribution For viewer certificates, use the `MinimumProtocolVersion` property to specify a minimum protocol version. For cache behaviors, use the `CachedMethods` property to specify which methods Amazon CloudFront (CloudFront) caches responses for. For origins, use the `OriginPath` to specify a path that CloudFront uses to request content. AWS::ElastiCache::CacheCluster For Memcached cache clusters, use the `AZMode` and `PreferredAvailabilityZones` properties to specify nodes in multiple Availability Zones (AZs).	2010-05-15

Change	Release Date	Description	API Version
		AWS::EC2::Volume Use the KmsKeyId property to specify a customer managed key for encrypted volumes. AWS::OpsWorks::Instance Use the TimeBasedAutoScaling property to automatically scale instances based on a schedule that you specify. AWS::OpsWorks::Layer Use the LoadBasedAutoScaling property to specify load-based scaling policies. For volume configurations, use the VolumeType and Iops properties to specify a volume type and the number of I/O operations per second, respectively. AWS::RDS::DBInstance Use the CharacterSetName property to specify a character set for supported database engines. Use the StorageEncrypted property to indicate whether database instances will be encrypted and the KmsKeyId to specify a customer managed key for encrypted database instances. AWS::Route53::HealthCheck Use the HealthCheckTags property to associate tags with health checks. AWS::Route53::HostedZone Use the VPCs property to create private hosted zones.	

Change	Release Date	Description	API Version
		Use the `HostedZoneTags` property to associate tags with hosted zones.	
New template section	April 16, 2015	Add the Metadata section to your templates to include arbitrary JSON objects that describe your templates, such as the design or implementation details.	2010-05-15
Resource update	April 8, 2015	For the AWS::CloudFormation::CustomResource resource, you can specify Lambda function Amazon Resource Names (ARNs) in the `ServiceToken` property.	2010-05-15
Amazon RDS update	December 24, 2014	CloudFormation added two new properties for RDS DB instances. You can associate an option group with a DB instance and specify the DB instance storage type. For more information, see AWS::RDS::DBInstance.	2010-05-15
Elastic Load Balancing update	December 24, 2014	You can use the `ConnectionSettings` property to specify how long connections can remain idle. For more information, see AWS::ElasticLoadBalancing::LoadBalancer.	2010-05-15
Route 53 update	November 6, 2014	You can now provision and manage Route 53 hosted zones , health checks, failover record sets , and geolocation record sets .	2010-05-15
Auto Scaling rolling update enhancement	November 6, 2014	During an update, you can use the `WaitOnResourceSignals` flag to instruct CloudFormation to wait for instances to signal success. That way, CloudFormation won't update the next batch of instances until the current batch is ready. For more information, see UpdatePolicy Attribute.	2010-05-15

Change	Release Date	Description	API Version
New VPC Fn:GetAtt attributes	November 6, 2014	Given a VPC ID, you can retrieve the default security group and network ACL for that VPC. For more information, see Fn::GetAtt.	2010-05-15
New AWS-specific parameter types	November 6, 2014	You can specify AWS-specific parameter types in your CloudFormation templates. In the CloudFormation console, these parameter types provide a drop-down list of valid values. With the API or AWS CLI, CloudFormation can quickly validate values for these parameter types before creating or updating a stack. For more information, see Parameters.	2010-05-15
CreationPolicy attribute	November 6, 2014	With the CreationPolicy attribute, you can instruct CloudFormation to wait until applications are ready on EC2 instances before proceeding with stack creation. You can use a creation policy instead of a wait condition and wait condition handle. For more information, see CreationPolicy Attribute.	2010-05-15
Amazon CloudFront forwarded values	September 29, 2014	For cache behaviors, you can forward headers to the origin. See ForwardedValues.	2010-05-15
AWS OpsWorks update	September 29, 2014	For Chef 11.10, you can use the `ChefConfiguration` property to enable Berkshelf. You can also use the AWS OpsWorks built-in security groups with your AWS OpsWorks stacks. For more information, see AWS::OpsWorks::Stack.	2010-05-15
Elastic Load Balancing tagging support	September 29, 2014	AWS CloudFormation tags Elastic Load Balancing load balancers with stack-level tags. You can also add your own tags to a load balancer. See AWS::ElasticLoadBalancing::LoadBalancer.	2010-05-15

Change	Release Date	Description	API Version
Amazon Simple Notification Service topic policy update	September 29, 2014	You can now update Amazon SNS topic policies. For more information, see AWS::SNS::TopicPolicy.	2010-05-15
RDS DB instance update	September 5, 2014	You can specify whether a DB instance is Internet-facing by using the `PubliclyAccessible` property in the AWS::RDS::DBInstance resource.	2010-05-15
UpdatePolicy attribute update	September 05, 2014	You can specify an update policy for an Auto Scaling group that has an associated scheduled action. For more information, see UpdatePolicy Attribute.	2010-05-15
Amazon CloudWatch support	July 10, 2014	You can use CloudFormation to provision and manage Amazon CloudWatch Logs (CloudWatch Logs) log groups and metric filters. For more information, see AWS::Logs::LogGroup or AWS::Logs::MetricFilter.	2010-05-15

Change	Release Date	Description	API Version
Amazon CloudFront distribution configuration update	June 17, 2014	You can specify additional CloudFront distribution configuration properties: • Custom error responses define custom error messages for 4xx and 5xx HTTP status codes. • Price class defines the maximum price that you want to pay for the CloudFront service. • Restrictions define who can view your content. • Viewer certificate specifies the certificate to use when viewers use HTTPS. • For cache behaviors, you can specify allowed HTTP methods and indicate whether to forward cookies. For more information, see AWS::CloudFront::Distribution.	2010-05-15
EC2 instance update	June 17, 2014	You can specify whether an instance stops or terminates when you invoke the instance's operating system shutdown command. For more information, see AWS::EC2::Instance.	2010-05-15
EBS volume update	June 17, 2014	You can use encrypted EBS volumes with supported instance types. For more information, see AWS::EC2::Volume.	2010-05-15
New Amazon VPC peering connection	June 17, 2014	You can use CloudFormation to create an Amazon Virtual Private Cloud (Amazon VPC) peering connection, which establishes a network connection between two VPCs. For more information, see AWS::EC2::VPCPeeringConnection.	2010-05-15

Change	Release Date	Description	API Version
Amazon EC2 Auto Scaling group update	June 17, 2014	You can specify an existing cluster placement group in which to launch instances for an Amazon EC2 Auto Scaling group. For more information, see AWS::AutoScaling::AutoScalingGroup.	2010-05-15
AWS CloudTrail support	June 17, 2014	CloudFormation supports AWS CloudTrail, which can capture API calls made from your AWS account and publish the logs at a location you designate. For more information, see AWS::CloudTrail::Trail.	2010-05-15
Update stack enhancements	May 12, 2014	CloudFormation supports additional features for updating stacks: • You can update CloudFormation stack parameters without resubmitting the stack's template. • You can add or remove Amazon SNS notification topics for an CloudFormation stack. For more information, see CloudFormation Stacks Updates.	2010-05-15
Amazon Kinesis support	May 6, 2014	You can use CloudFormation to create Amazon Kinesis streams that capture and transport data records from data sources. For more information, see AWS::Kinesis::Stream.	2010-05-15

Change	Release Date	Description	API Version
New S3 bucket properties	May 5, 2014	CloudFormation supports additional S3 bucket properties: • Cross-origin resource sharing (CORS) defines cross-origin resource sharing of objects in a bucket. • Lifecycle defines how Amazon S3 manages objects during their lifetime. • Access logging policy captures information about requests made to your bucket. • Notifications define which events to report and which Amazon SNS topic to send messages to. • Versioning enables multiple variants of all objects in a bucket. • Redirect and routing rules govern redirect behavior for requests made to a bucket's website endpoint. For more information, see AWS::S3::Bucket.	2010-05-15
Amazon EC2 Auto Scaling support	May 5, 2014	CloudFormation supports metrics collection for an Auto Scaling group. For more information, see AWS::AutoScaling::AutoScalingGroup.	2010-05-15
`Fn::If` update	May 5, 2014	You can use the `Fn::If` intrinsic function in the output section of a template. For more information, see Condition Functions.	2010-05-15
API logging with AWS CloudTrail	April 2, 2014	You can use AWS CloudTrail (CloudTrail) to log CloudFormation requests. With CloudTrail you can get a history of CloudFormation API calls for your account. For more information, see Logging CloudFormation API Calls with AWS CloudTrail.	2010-05-15

Change	Release Date	Description	API Version
Elastic Load Balancing update	March 20, 2014	You can specify an access logging policy to capture information about requests made to your load balancer. You can also specify a connection draining policy that describes how to handle in-flight requests when instances are deregistered or become unhealthy. For more information, see AWS::ElasticLoadBalancing::LoadBalancer.	2010-05-15
OpsWorks support	March 3, 2014	You can use CloudFormation to provision and manage OpsWorks stacks. For more information, see AWS::OpsWorks::Stack or AWS OpsWorks Template Snippets.	2010-05-15
Amazon S3 template size limit increase	February 18, 2014	You can specify template sizes up to 460,800 bytes in Amazon S3.	2010-05-15
Amazon Redshift support	February 10, 2014	You can use CloudFormation to provision and manage Amazon Redshift clusters. For more information, see Amazon Redshift Template Snippets or AWS::Redshift::Cluster.	2010-05-15
S3 buckets and bucket policies update	February 10, 2014	You can update some properties of the S3 bucket and bucket policy resources. For more information, see AWS::S3::Bucket or AWS::S3::BucketPolicy.	2010-05-15
Elastic Beanstalk environments and application versions update	February 10, 2014	You can update Elastic Beanstalk environment configurations and application versions. For more information, see AWS::ElasticBeanstalk::Environment, AWS::ElasticBeanstalk::ConfigurationTemplate, or AWS::ElasticBeanstalk::ApplicationVersion.	2010-05-15

Change	Release Date	Description	API Version
Amazon SQS update	January 29, 2014	You can specify a dead letter queue for an Amazon SQS queue. For more information, see AWS::SQS::Queue.	2010-05-15
Auto Scaling scheduled actions	January 27, 2014	You can scale the number of EC2 instances in an Auto Scaling group based on a schedule. By using a schedule, you can scale applications in response to predictable load changes. For more information, see AWS::AutoScaling::ScheduledAction.	2010-05-15
DynamoDB secondary indexes	January 27, 2014	You can create local and global secondary indexes for DynamoDB databases. By using secondary indexes, you can efficiently access data with attributes other than the primary key. For more information, see AWS::DynamoDB::Table.	2010-05-15
Auto Scaling update	January 2, 2014	You can specify an instance ID for an Auto Scaling group or launch configuration. You can also specify additional Auto Scaling block device properties. For more information, see AWS::AutoScaling::AutoScalingGroup or AWS::AutoScaling::LaunchConfiguration.	2010-05-15
Amazon SQS update	January 2, 2014	You can update SQS queues and specify additional properties. For more information, see AWS::SQS::Queue.	2010-05-15
Limit increases	January 2, 2014	You can specify up to 60 parameters and 60 outputs in your CloudFormation templates.	2010-05-15
New console	December 19, 2013	The new CloudFormation console adds features like auto-refreshing stack events and alphabetical ordering of stack parameters.	2010-05-15
Cross-zone load balancing	December 19, 2013	With cross-zone load balancing, you can route traffic to back-end instances across all Availability Zones (AZs). For more information, see AWS::ElasticLoadBalancing::LoadBalancer.	2010-05-15

Change	Release Date	Description	API Version
AWS Elastic Beanstalk environment tiers	December 19, 2013	You can specify whether AWS Elastic Beanstalk provisions resources to support a web server or to handle background processing tasks. For more information, see AWS::ElasticBeanstalk::Environment.	2010-05-15
Resource names	December 19, 2013	You can assign names (physical IDs) to the following resources: • ElastiCache clusters • Elastic Load Balancing load balancers • RDS DB instances For more information, see Name Type.	2010-05-15
VPN support	November 22, 2013	You can enable a virtual private gateway (VGW) to propagate routes to the routing tables of a VPC. For more information, see AWS::EC2::VPNGatewayRoutePropagation.	2010-05-15
Conditionally create resources and assign properties	November 8, 2013	Using input parameters, you can control the creation and settings of designated stack resources by defining conditions in your CloudFormation templates. For example, you can use conditions to create stack resources for a production environment. Using the same template, you can create similar stack resources with lower capacity for a test environment. For more information, see Condition Functions.	2010-05-15

Change	Release Date	Description	API Version
Prevent accidental updates to stack resources	November 8, 2013	You can prevent stack updates that might result in unintentional changes to stack resources. For example, if you have a stack with a database layer that should rarely be updated, you can set a stack policy that prevents most users from updating that database layer. For more information, see Prevent Updates to Stack Resources.	2010-05-15
Name resources	November 8, 2013	Instead of using CloudFormation-generated physical IDs, you can assign names to certain resources. The following CloudFormation resources support naming • Amazon CloudWatch alarms • DynamoDB tables • AWS Elastic Beanstalk applications and environments • Amazon S3 buckets • Amazon SNS topics • Amazon SQS queues For more information, see Name Type.	2010-05-15
Assign custom resource types	November 8, 2013	In your templates, you can specify your own resource type for CloudFormation custom resources (`AWS::CloudFormation::CustomResource`). By using your own custom resource type name, you can quickly identify the type of custom resources that you have in your stack. For example, you can specify `"Type": "Custom::` *`MyCustomResource`* `"`. For more information, see AWS::CloudFormation::Custom Resource.	2010-05-15

Change	Release Date	Description	API Version
Add pseudo parameter	November 8, 2013	You can now refer to the AWS AccountID inside CloudFormation templates by referring to the `AWS::AccountID` pseudo parameter. For more information, see Pseudo Parameters Reference.	2010-05-15
Specify stacks in IAM policies	November 8, 2013	You can allow or deny IAM users, groups, or roles to operate on specific CloudFormation stacks. For example, you can deny the delete stack action on a specific stack ID. For more information, see Controlling Access with AWS Identity and Access Management.	2010-05-15
Federation support	October 14, 2013	CloudFormation supports temporary security credentials from IAM roles, which enable scenarios such as federation and single sign-on to the AWS Management Console. You can also make calls to CloudFormation from EC2 instances without embedding long-term security credentials by using IAM roles. For more information about CloudFormation and IAM, see Controlling Access with AWS Identity and Access Management.	2010-05-15
Amazon RDS read replica support	September 24, 2013	You can now create Amazon RDS read replicas from a source DB instance. For more information, see the `SourceDBInstanceIdentifier` property in the AWS::RDS::DBInstance resource.	2010-05-15
Associate public IP address with instances in an Auto Scaling group	September 19, 2013	You can now associate public IP addresses with instances in an Auto Scaling group. For more information, see AWS::AutoScaling::LaunchConfiguration.	2010-05-15

Change	Release Date	Description	API Version
Additional VPC support	September 17, 2013	CloudFormation adds several enhancements to support VPC and VPN functionality • You can associate a public IP address and multiple private IP addresses to Amazon EC2 network interfaces. For more information, see AWS::EC2::NetworkInterface. You can also associate a primary private IP address to an elastic IP address (EIP). • You can enable DNS support and specify DNS host names. For more information, see AWS::EC2::VPC. • You can specify a static route between a virtual private gateway to your VPN gateway. For more information, see AWS::EC2::VPNConnectionRoute.	2010-05-15
Redis and VPC security groups support for Amazon ElastiCache	September 3, 2013	You can now specify Redis as the cache engine for an Amazon ElastiCache (ElastiCache) cluster. You can also now assign VPC security groups to ElastiCache clusters. For more information, see AWS::ElastiCache::CacheCluster.	2010-05-15
Parallel stack creation, update and deletion, and nested stack updates	August 12, 2013	AWS CloudFormation now creates, updates, and deletes resources in parallel, improving the operations' performance. If you update a top-level template, AWS CloudFormation automatically updates nested stacks that have changed. For more information, see CloudFormation Stacks Updates.	2010-05-15
VPC security groups can now be set in RDS DB instances	February 28, 2013	You can now assign VPC security groups to an RDS DB instance with CloudFormation. For more information, see the VPCSecurityGroups property in AWS::RDS::DBInstance.	2010-05-15

Change	Release Date	Description	API Version
Rolling deployments for Amazon EC2 Auto Scaling groups	February 20, 2013	CloudFormation now supports update policies on Amazon EC2 Auto Scaling groups, which describe how instances in the Amazon EC2 Auto Scaling group are replaced or modified when the Amazon EC2 Auto Scaling group adds or removes instances. You can modify these settings at stack creation or during a stack update. For more information and an example, see UpdatePolicy Attribute.	2010-05-15
Cancel and rollback action for stack updates	February 20, 2013	CloudFormation supports the ability to cancel a stack update. The stack must be in the UPDATE_IN_PROGRESS state when the update request is made. More information is available in the following topics: • Canceling a Stack Update • cancel-update-stack • CancelUpdateStack in the *CloudFormation API Reference*	2010-05-15
EBS-optimized instances for Amazon EC2 Auto Scaling groups	February 20, 2013	You can now provision EBS-optimized instances in Amazon EC2 Auto Scaling groups for dedicated throughput to Amazon Elastic Block Store (Amazon EBS) in autoscaled instances. The implementation is similar to that of the previously released support for optimized Amazon EBS EC2 instances. For more information, see the new `EbsOptimized` property in AWS::AutoScaling::LaunchConfiguration.	2010-05-15

Change	Release Date	Description	API Version
New documentation	December 21, 2012	AWS::EC2::Instance now provides a `BlockDeviceMappings` property to allow you to set block device mappings for your EC2 instance. With this change, two new types have been added: • BlockDeviceMapping • Block Device	2010-05-15
New documentation	December 21, 2012	New sections have been added to describe the procedures for creating and viewing stacks using the recently redesigned AWS Management Console. You can find them here: • Creating a Stack from the CloudFormation console • View stack information from the CloudFormation console	2010-05-15
New documentation	November 15, 2012	Information about custom resources is provided in the following topics: • Custom Resources • AWS::CloudFormation::CustomResource • Custom Resource Reference	2010-05-15
Updated documentation	November 15, 2012	CloudFormation now supports specifying provisioned I/O operations per second (IOPS) for RDS DB instances. You can set this value from 1000–10,000 in 1000 IOPS increments by using the new Iops property in AWS::RDS::DBInstance . For more information about specifying IOPS for RDS DB instances, see Provisioned IOPS in the *Amazon Relational Database Service User Guide*.	2010-05-15

Change	Release Date	Description	API Version
New and updated documentation	August 27, 2012	Topics have been reorganized to more clearly provide specific information about using the AWS Management Console and using the CloudFormation command line interface (CLI). Information about tagging CloudFormation stacks has been added, including new guides and updated reference topics: • New topic: Configure stack options. • New information about tags in the *CloudFormation API reference*: CreateStack, Stack, and Tag. New information about working with Windows stacks: • Microsoft Windows Amazon Machine Images (AMIs) and CloudFormation templates • Bootstrapping CloudFormation Windows stacks New topic: Using regular expressions in CloudFormation templates.	2010-05-15

Change	Release Date	Description	API Version
New feature	April 25, 2012	CloudFormation now provides full support for Virtual Private Cloud (VPC) security with Amazon EC2. You can now create and populate an entire VPC with every type of VPC resource (subnets, gateways, network ACLs, route tables, and so forth) using a single CloudFormation template. Documentation for the following resource types has been updated: • AWS::EC2::SecurityGroup • AWS::EC2::SecurityGroupIngress • AWS::EC2::SecurityGroupEgress • AWS::EC2::Instance • AWS::AutoScaling::AutoScalingGroup • AWS::EC2::EIP • AWS::EC2::EIPAssociation • AWS::ElasticLoadBalancing::LoadBalancer New resource types have been added to the documentation: • AWS::EC2::VPC • AWS::EC2::InternetGateway • AWS::EC2::DHCPOptions • AWS::EC2::DHCPOptions • AWS::EC2::RouteTable • AWS::EC2::NetworkAcl • AWS::EC2::NetworkAclEntry • AWS::EC2::Subnet • AWS::EC2::VPNGateway	2010-05-15

Change	Release Date	Description	API Version
		• AWS::EC2::CustomerGateway	
New feature	April 13, 2012	CloudFormation now allows you to add or remove elements from a stack when updating it. CloudFormation Stacks Updates has been updated, and a new section has been added to the walkthrough: Change the Stack's Resources, which describes how to add and remove resources when updating the stack.	2010-05-15

Change	Release Date	Description	API Version
New feature	February 2, 2012	CloudFormation now provides support for resources in an existing Amazon Virtual Private Cloud (Amazon VPC). With this release, you can: • Launch an EC2 Dedicated instance into an existing Amazon VPC. For more information, see AWS::EC2::Instance. • Set the `SourceDestCheck` attribute of an EC2 instance that resides in an existing Amazon VPC. For more information, see AWS::EC2::Instance. • Create Elastic IP addresses in an existing Amazon VPC. For more information, see AWS::EC2::EIP. • Use CloudFormation to create Amazon VPC security groups and ingress/egress rules in an existing VPC. For more information, see AWS::EC2::SecurityGroup. • Associate an Auto Scaling group with an existing Amazon VPC by setting the `VPCZoneIdentifier` property of your `AWS::AutoScaling::AutoScalingGroup` resource. For more information, see AWS::AutoScaling::AutoScalingGroup. • Attach an Elastic Load Balancing load balancer to a Amazon VPC subnet and create security groups for the load balancer. For more information, see AWS::ElasticLoadBalancing::LoadBalancer. • Create an RDS DB instance in an existing Amazon VPC. For more information, see AWS::RDS::DBInstance.	2010-05-15

Change	Release Date	Description	API Version
New feature	February 2, 2012	You can now update properties for the following resources in an existing stack: • AWS::EC2::SecurityGroupIngress • AWS::EC2::SecurityGroupEgress • AWS::EC2::EIPAssociation • AWS::RDS::DBSubnetGroup • AWS::RDS::DBSecurityGroup • AWS::RDS::DBSecurityGroupIngress • AWS::Route53::RecordSetGroup For a complete list of updatable resources and details about what to consider when updating a stack, see CloudFormation Stacks Updates.	2010-05-15
Restructured guide	February 2, 2012	Reorganized existing sections into new sections: Working with CloudFormation Templates and **Managing Stacks**. Moved Template Reference to the top level of the Table of Contents. Moved Estimating the Cost of Your CloudFormation Stack to the Getting Started section.	2010-05-15

Change	Release Date	Description	API Version
New content	February 2, 2012	Added new sections: • Walkthrough: Updating a Stack is a tutorial that walks through the process of updating a LAMP stack. • Deploying Applications on Amazon EC2 with AWS CloudFormation describes how to use CloudFormation helper scripts to deploy applications using metadata stored in your template. • CloudFormation Helper Scripts Reference provides reference material for the CloudFormation helper scripts (cfn-init, cfn-get-metadata, cfn-signal, and cfn-hup).	2010-05-15
New feature	May 26, 2011	CloudFormation now provides the `list-stacks` command, which enables you to list stacks filtered by stack status. Deleted stacks can be listed for up to 90 days after they have been deleted. For more information, see Describing and Listing Your Stacks.	2010-05-15
New features	May 26, 2011	The `describe-stack-resources` and `get-template` commands now enable you to get information from stacks that have been deleted for 90 days after they have been deleted. For more information, see Listing Resources and Retrieving a Template.	2010-05-15
New link	March 1, 2011	CloudFormation endpoint information is now located in the AWS General Reference. For more information, go to Regions and Endpoints in AWS General Reference.	2010-05-15
Initial release	February 25, 2011	The initial public release of CloudFormation.	2010-05-15