aws

User Guide

AWS Transfer Family



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

AWS Transfer Family: 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 AWS Transfer Family? 1
How AWS Transfer Family works 4
Blog posts relevant for Transfer Family 5
Prerequisites 9
Regions, endpoints and quotas 9
Sign up for AWS 10
Configure storage 10
Configure an Amazon S3 bucket 11
Configure an Amazon EFS file system 15
Create an IAM role and policy 19
Create a user role 20
How session policies work 23
Example read/write access policy 27
Transfer Family tutorials 30
Get started with server endpoints 31
Prerequisites 31
Step 1: Sign in to the AWS Transfer Family console 32
Step 2: Create an SFTP-enabled server 32
Step 3: Add a service managed user 33
Step 4: Transfer a file using a client 34
Create a decryption Workflow 36
Step 1: Configure an execution role 37
Step 2: Create a managed workflow 38
Step 3: Add the workflow to a server and create a user 39
Step 4: Create a PGP key pair 41
Step 5: Store the PGP private key in AWS Secrets Manager 42
Step 6: Encrypt a file 43
Step 7: Run the workflow and view the results 43
Create and use SFTP connectors 45
Connector egress types 45
Step 1: Create the necessary supporting resources 46
Step 2: Create and test an SFTP connector 52
Step 3: Send and retrieve files using the SFTP connector 58
Procedures to create a Transfer Family server to use as your remote SFTP server 62

Use a custom identity provider 65
Prerequisites 65
Step 1: Create a CloudFormation stack 66
Step 2: Check the API Gateway method configuration for your server 67
Step 3: View the Transfer Family server details 68
Step 4: Test that your user can connect to the server 70
Step 5: Test the SFTP connection and file transfer 70
Step 6: Limit access to the bucket 71
Update Lambda if using Amazon EFS 73
Set up an AS2 configuration 74
Step 1: Create certificates for AS2 76
Step 2: Import certificates as Transfer Family certificate resources 79
Step 3: Create profiles for you and your trading partner 81
Step 4: Create a Transfer Family server that uses the AS2 protocol 81
Step 5: Create an agreement between you and your partner 85
Step 6: Create a connector between you and your partner 86
Step 7: Test exchanging files over AS2 by using Transfer Family 88
Set up a basic web app 90
Prerequisites 90
Step 1: Create the necessary supporting resources 90
Step 2: Create a Transfer Family web app 91
Step 3: Configure Cross-origin resource sharing (CORS) for your bucket 92
Step 4: Add a user to your Transfer Family web app 94
Step 5: Register a location in Amazon S3 and create an access grant 95
Step 6: Access your Transfer Family web app as a user 96
Next steps 98
Integrate Okta as your identity provider for web apps 98
Set up web app with selective multi-bucket access 101
Prerequisites 101
Step 1: Create a Transfer Family web app 101
Step 2: Configure IAM roles for S3 access 102
Step 3: Set up S3 Access Grants 105
Step 4: Register S3 bucket locations 105
Step 5: Create Access Grants 105
Step 6: Configure CORS policy for S3 buckets 106
Step 7: Test the configuration 107

Conclusion ... 107
Transfer Family for SFTP, FTPS, FTP ... 109
Identity provider options ... 109
AWS Transfer Family endpoint type matrix ... 111
Configure a Transfer Family server endpoint ... 115
Create an SFTP-enabled server ... 117
Create an FTPS-enabled server ... 125
Create an FTP-enabled server ... 133
Create a server in a VPC ... 140
Working with custom hostnames ... 165
Working with Network Load Balancers ... 169
FTP/FTPS NLB considerations ... 173
Transfer files over server endpoint ... 174
Available SFTP/FTPS/FTP Commands ... 177
Find your Amazon VPC endpoint ... 178
Avoid `setstat` errors ... 180
Use OpenSSH ... 35
Use WinSCP ... 182
Use Cyberduck ... 35
Use FileZilla ... 185
Use a Perl client ... 186
Use LFTP ... 187
Post upload processing ... 187
SFTP messages ... 188
Manage users ... 190
Amazon EFS vs. Amazon S3 ... 191
Logical directories ... 191
Active Directory group quotas ... 193
Service-managed users ... 193
Custom identity provider ... 203
Directory service for MS AD ... 239
Directory service for Entra ID ... 250
Use logical directories ... 257
Understanding chroot and directory structure ... 258
Rules for using logical directories ... 258
Implementing logical directories ... 260

Configure logical directories examples ... 263
Configure logical directories for Amazon EFS ... 265
Custom AWS Lambda response ... 265
Access FSx for NetApp ONTAP file systems ... 266
Overview ... 267
Prerequisites ... 267
How FSx storage works with Transfer Family ... 268
Creating an S3 access point for FSx ... 269
Using S3 access point aliases with FSx ... 272
Configuring Transfer Family for FSx storage ... 274
Managing users for FSx storage ... 275
Configuring file transfer clients ... 277
Troubleshooting FSx storage ... 278
Transfer Family web apps ... 280
AWS Regions for Transfer Family web apps ... 281
Browser compatibility for AWS Transfer Family web apps ... 281
How to create a Transfer Family web app ... 281
Configure your identity provider ... 284
Configure IAM roles ... 286
Configure a Transfer Family web app ... 288
Create a Transfer Family web app ... 289
IP addressing ... 291
Create a Transfer Family web app in a VPC ... 292
Create a Transfer Family web app ... 292
Post-creation steps ... 294
Cross-origin resource sharing (CORS) policy ... 294
Restricting access to a specific VPC endpoint ... 295
Assign or add users or groups to a Transfer Family web app ... 296
Set up Cross-origin resource sharing (CORS) for your bucket ... 299
Configure Amazon S3 Access Grants ... 301
Use a custom URL ... 307
Logging for Transfer Family web apps ... 310
Enabling Amazon S3 data events ... 310
Authentication log examples ... 313
Viewing log entries ... 320
Troubleshooting your web apps ... 321

Troubleshoot network errors ... 321
Troubleshoot configured bucket not appearing ... 322
Troubleshoot custom URL errors ... 322
Troubleshoot miscellaneous errors ... 323
Duplicate S3 buckets appearing in web app ... 323
End user instructions ... 324
Web app quotas ... 324
IAM Identity Center users ... 324
Third-party end users ... 326
Transfer Family end user interface ... 326
Available actions ... 327
SFTP connectors ... 330
Creating SFTP connectors ... 330
Choosing SFTP connector egress type ... 331
Choosing IP addressing mode ... 332
Store credentials in Secrets Manager ... 333
Create an SFTP connector with service-managed egress ... 335
Create an SFTP connector with VPC-based egress ... 346
Test an SFTP connector ... 357
VPC connectivity ... 358
Using SFTP connectors ... 359
Transfer files ... 360
List contents of remote directories ... 362
Move and delete files on the remote server ... 364
Monitoring SFTP connectors ... 366
Use the connector API to query the status of file transfer requests ... 366
View SFTP connector events in Amazon EventBridge ... 366
View SFTP connector logs in Amazon CloudWatch ... 366
Monitoring VPC egress type connectors ... 367
Managing SFTP connectors ... 369
Update SFTP connectors ... 369
View SFTP connector details ... 370
Quotas for SFTP connectors ... 371
Quotas for SFTP connectors ... 371
Scaling your SFTP connectors ... 373
Reference architectures ... 374

Blog posts ... 374
Workshops ... 374
Solutions ... 375
VPC reference architectures ... 375
Transfer Family for AS2 ... 378
AS2 use cases ... 379
AS2 CloudFormation templates ... 383
Customizing AS2 templates ... 383
Testing your AS2 deployment ... 383
Best practices for AS2 template deployment ... 384
Configure AS2 ... 385
AS2 configurations ... 386
AS2 quotas ... 387
AS2 features and capabilities ... 391
Manage AS2 certificates ... 392
Import AS2 certificates ... 393
AS2 certificate rotation ... 395
Create AS2 profiles ... 397
Create an AS2 server ... 398
Create an AS2 server using the Transfer Family console ... 398
Create an AS2 server using a template ... 401
Create an AS2 agreement ... 404
Configure AS2 connectors ... 406
Create an AS2 connector ... 407
AS2 connector algorithms ... 410
Basic authentication for AS2 connectors ... 411
Enable Basic authentication for AS2 connectors ... 412
View connector details ... 415
Transfer AS2 messages ... 418
Receive AS2 messages ... 419
Configure HTTPS for AS2 ... 420
Transfer files with AS2 connectors ... 426
File names and locations ... 428
Status codes ... 431
Sample JSON files ... 432
Custom HTTP Headers for AS2 ... 434

Template Overview ... 434
How It Works ... 435
Key Features ... 435
Implementation Details ... 435
Deployment and Usage ... 436
Monitor AS2 ... 437
AS2 Status codes ... 439
AS2 error codes ... 440
Certificate expiration monitoring ... 453
DaysUntilExpiry metric ... 453
Best practices for certificate monitoring ... 453
Example alarm configurations ... 454
Managing file-processing workflows ... 455
Create a workflow ... 457
Configure and run a workflow ... 458
View workflow details ... 460
Use predefined steps ... 463
Copy file ... 464
Decrypt file ... 469
Tag file ... 475
Delete file ... 476
Named variables for workflows ... 477
Example tag and delete workflow ... 477
Use custom file-processing steps ... 482
Using multiple Lambda functions consecutively ... 483
Accessing a file after custom processing ... 484
Example events sent to AWS Lambda upon file upload ... 485
Example Lambda function for a custom workflow step ... 486
IAM permissions for a custom step ... 487
IAM policies for workflows ... 487
Workflow trust relationships ... 489
Example execution role: Decrypt, copy, and tag ... 490
Example execution role: Run function and delete ... 492
Exception handling for a workflow ... 492
Monitor workflow execution ... 493
CloudWatch logging for a workflow ... 493

CloudWatch metrics for workflows ... 497
Create workflow from template ... 497
Remove a workflow from a Transfer Family server ... 501
Restrictions and limits ... 503
Managing servers ... 505
View a list of servers ... 505
Delete a server ... 506
View SFTP server details ... 507
View AS2 server details ... 508
IPv6 support ... 510
IPv6 limitations ... 510
Configuring IPv6 for servers ... 511
Use ALB for dual-stack AS2 public server ... 511
Edit server details ... 513
Edit the file transfer protocols ... 516
Edit the server endpoint ... 518
Edit logging ... 519
Edit the security policy ... 520
Change the managed workflow ... 522
Change the display banners for your server ... 523
Put your server online or offline ... 523
Edit identity provider configuration ... 524
Changing to service-managed identity provider ... 524
Changing to AWS Directory Service ... 525
Changing to custom identity provider ... 525
User preservation during identity provider transitions ... 530
Important considerations when changing identity providers ... 530
Managing server host keys ... 530
When to import host keys ... 532
Adding an additional server host key ... 532
Storing host keys in Secrets Manager ... 534
Deleting a server host key ... 538
Rotating the server host keys ... 538
Additional server host key information ... 540
Monitor usage within console ... 541
Managing access controls ... 547

Creating an S3 bucket access policy ... 548
Creating a session policy ... 549
Example session policy ... 550
Nested substitutions for session policies ... 553
Dynamic permission management ... 553
Understanding Transfer Family permission architecture ... 553
Two approaches to permission management ... 554
Implementing session policies ... 555
Implementation by user type ... 555
Example: Simplifying role management with session policies ... 556
CloudTrail logging ... 559
Enabling CloudTrail logging ... 560
Enabling Amazon S3 data events ... 561
Example log entry for creating a server ... 561
Data access logs ... 563
Example log entry for successful data access ... 563
Common data access operations ... 565
CloudWatch logging ... 567
Types of CloudWatch logging for Transfer Family ... 567
Creating logging for servers ... 569
Creating logging for servers ... 572
Updating logging for a server ... 573
Viewing the server configuration ... 576
Managing logging for workflows ... 578
Configuring a role for CloudWatch ... 582
Viewing Transfer Family log streams ... 584
Creating Amazon CloudWatch alarms ... 588
Logging S3 API calls to S3 access logs ... 589
Examples to limit confused deputy problem ... 589
CloudWatch log structure for Transfer Family ... 591
JSON structured logs for Transfer Family ... 591
Legacy logs for Transfer Family ... 595
Example CloudWatch log entries ... 598
Example transfer sessions log entries ... 599
Example log entries for SFTP connectors ... 600
Example log entries for VPC Lattice connectors ... 602

Example log entries for Key exchange algorithm failures ... 603
Using CloudWatch metrics ... 604
Transfer Family dimensions ... 609
User notifications ... 609
CloudWatch queries ... 609
Managing events using EventBridge ... 612
Transfer Family events ... 613
SFTP, FTPS, and FTP server events ... 613
SFTP connector events ... 616
AS2 events ... 617
Sending Transfer Family events ... 618
Creating event patterns ... 619
Testing event patterns for Transfer Family events ... 621
Permissions ... 621
Additional resources ... 621
Events detail reference ... 621
Server events ... 622
Connector events ... 629
AS2 events ... 637
Security ... 643
VPC connectivity security benefits ... 644
VPC Lattice security model ... 644
Security best practices for VPC connectivity ... 645
Security policies for servers ... 645
Cryptographic algorithms ... 647
Security policy details ... 658
Security policies for SFTP connectors ... 676
Cryptographic algorithms ... 676
SFTP connector security policy details ... 677
Post-Quantum security policies ... 679
About post-quantum hybrid key exchange in SSH ... 680
How to use it ... 680
How to test it ... 682
Data protection ... 685
Data encryption in Transfer Family ... 686
Encryption at rest ... 687

Key management ... 688
Algorithm overview ... 689
SSH authentication ... 689
PGP algorithms ... 690
Generate SSH keys ... 692
Rotate SSH keys ... 697
Generate PGP keys ... 700
Manage PGP keys ... 701
Supported PGP clients ... 705
Identity and access management ... 706
Audience ... 707
Authenticating with identities ... 707
Managing access using policies ... 709
How AWS Transfer Family works with IAM ... 710
Identity-based policy examples ... 715
Tag-based policy examples ... 717
Troubleshooting identity and access ... 721
IAM condition keys ... 723
Compliance validation ... 725
Resilience ... 725
VPC endpoints for APIs ... 726
Controlling access using VPC endpoint policies ... 727
Create VPC endpoint ... 728
Infrastructure security ... 728
NLB and NAT considerations ... 728
VPC connectivity infrastructure security ... 730
Web application firewall ... 731
Cross-service confused deputy prevention ... 732
Transfer Family user roles ... 734
Transfer Family workflow roles ... 735
Transfer Family connector roles ... 737
Transfer Family logging/invocation roles ... 738
AWS managed policies ... 740
AWSTransferConsoleFullAccess ... 740
AWSTransferFullAccess ... 740
AWSTransferLoggingAccess ... 741

AWSTransferReadOnlyAccess ... 741
AWSTransferServiceRolePolicy ... 741
Policy updates ... 741
Using service-linked roles ... 742
Service-linked role permissions ... 743
Creating a service-linked role ... 744
Editing a service-linked role ... 744
Deleting a service-linked role ... 744
Supported Regions ... 745
Transfer Family Terraform modules ... 746
SFTP servers ... 746
SFTP connectors ... 746
AS2 ... 747
B2B Data Interchange ... 747
Troubleshooting Transfer Family ... 748
Authentication issues ... 748
Authentication failures—SSH/SFTP ... 749
Managed AD mismatched realms issue ... 749
Active Directory group limits exceeded ... 750
Miscellaneous authentication issues ... 750
Troubleshoot Amazon API Gateway issues ... 751
Troubleshoot testing your identity provider ... 753
Duplicate Amazon S3 buckets in web app ... 323
SFTP connector issues ... 754
Troubleshoot adding trusted host keys for your SFTP connector ... 754
Key negotiation fails ... 755
SFTP connector throttling ... 755
Optimizing SFTP connector performance ... 756
Troubleshoot VPC connectivity issues ... 757
Miscellaneous SFTP connector issues ... 760
SFTP connectivity issues ... 760
Troubleshoot SFTP connectivity issues ... 761
Troubleshoot SFTP client issues ... 761
Troubleshoot file upload issues ... 761
Troubleshoot VPC egress type SFTP connector issues ... 762
Custom identity provider issues ... 764

Troubleshoot API Gateway integration errors ... 765
Troubleshoot Lambda function timeouts ... 766
Troubleshoot consistent Lambda timeout issues ... 767
Troubleshoot `KeyError` exceptions ... 767
Workflow issues ... 768
Troubleshoot managed workflows issues ... 768
Troubleshoot workflow decryption issues ... 771
EFS issues ... 775
Troubleshoot Amazon EFS issues ... 775
Storage and encryption issues ... 777
Troubleshoot policies for encrypted Amazon S3 buckets ... 777
Troubleshoot `ResourceNotFound` exception ... 778
Monitoring and alerting issues ... 779
Troubleshoot missing or incomplete CloudWatch metrics ... 779
Troubleshoot missing EventBridge events ... 780
Cross-region transfer issues ... 781
Troubleshoot cross-region transfer permission issues ... 781
Troubleshoot cross-region transfer performance issues ... 783
Terraform deployment issues ... 784
Troubleshoot Terraform resource creation failures ... 784
Troubleshoot Terraform state management issues ... 785
WAF integration issues ... 786
Troubleshoot WAF blocking legitimate traffic ... 786
Troubleshoot WAF integration with custom identity providers ... 787
Service-managed user issues ... 788
Troubleshoot service-managed users ... 788
AS2 issues ... 789
Troubleshoot AS2 issues ... 789
AS2 certificate issues ... 789
AS2 MDN receipt issues ... 790
Certificate expiration monitoring issues ... 790
API refererence ... 792
Making API requests ... 793
Transfer Family required request headers ... 794
Transfer Family request inputs and signing ... 794
Error responses ... 794

Available libraries 797
Identity Providers 797
Conventions 798
ID Formats 798
ARN Formats 799
DNS and Endpoints 800
API Endpoints 800
Server Endpoints 801
Document history 802

What is AWS Transfer Family?

AWS Transfer Family is a secure transfer service that enables you to transfer files into and out of AWS storage services. Transfer Family is part of the AWS Cloud platform. AWS Transfer Family offers fully managed support for the transfer of files over SFTP, AS2, FTPS, FTP, and web browser-based transfers directly into and out of AWS storage services. You can seamlessly migrate, automate, and monitor your file transfer workflows by maintaining existing client-side configurations for authentication, access, and firewalls—so nothing changes for your customers, partners, and internal teams, or their applications. See Getting started with AWS to learn more and to start building cloud applications with Amazon Web Services.

AWS Transfer Family supports transferring data from or to the following AWS storage services.

- Amazon Simple Storage Service (Amazon S3) storage. For information about Amazon S3, see Getting started with Amazon Simple Storage Service.
- Amazon Elastic File System (Amazon EFS) Network File System (NFS) file systems. For information about Amazon EFS, see What is Amazon Elastic File System?

AWS Transfer Family supports transferring data over the following protocols:

- Secure File Transfer Protocol (SFTP): version 3

  The official IETF document is here: SSH File Transfer Protocol draft-ietf-secsh-filexfer-02.txt.
- File Transfer Protocol Secure (FTPS)
- File Transfer Protocol (FTP)
- Applicability Statement 2 (AS2)
- Browser-based transfers

> **Note**
>
> For FTP and FTPS data connections, the port range that Transfer Family uses to establish the data channel is 8192–8200.

File transfer protocols are used in data exchange workflows across different industries such as financial services, healthcare, advertising, and retail, among others. Transfer Family simplifies the migration of file transfer workflows to AWS.

The following are some common use cases for using Transfer Family with Amazon S3:

- Data lakes in AWS for uploads from third parties such as vendors and partners.
- Subscription-based data distribution with your customers.
- Internal transfers within your organization.

The following are some common use cases for using Transfer Family with Amazon EFS:

- Data distribution
- Supply chain
- Content management
- Web serving applications

The following are some common use cases for using Transfer Family with AS2:

- Workflows with compliance requirements that rely on having data protection and security features built into the protocol
- Supply chain logistics
- Payments workflows
- Business-to-business (B2B) transactions
- Integrations with enterprise resource planning (ERP) and customer relationship management (CRM) systems

The following are some common use cases for using Transfer Family web apps:

- Simplified access to data in Amazon S3 to a wider and diverse range of business users
- Centralized data access management for your workforce
- Visualization of Amazon S3 Access Grants through a managed interface

With Transfer Family, you get access to a file transfer protocol-enabled server in AWS (or a managed file transfer web interface), without the need to run any server infrastructure. You can

use this service to migrate your file transfer-based workflows to AWS while maintaining your end users' clients and configurations as is. For servers, you first associate your hostname with the server endpoint, then add your users and provision them with the right level of access. After you do this, your users' transfer requests are serviced directly out of your Transfer Family server endpoint.

For Transfer Family web apps, determine your configuration settings and apply optional customizations. After you do this, your users can log in and directly transfer data to and from Amazon S3.

Transfer Family provides the following benefits:

- A fully managed service that scales in real time to meet your needs.
- You don't need to modify your applications or run any file transfer protocol infrastructure.
- With your data in durable Amazon S3 storage, you can use native AWS services for processing, analytics, reporting, auditing, and archival functions.
- With Amazon EFS as your data store, you get a fully managed elastic file system for use with AWS Cloud services and on-premises resources. Amazon EFS is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files. This helps eliminate the need to provision and manage capacity to accommodate growth.
- A fully managed, serverless File Transfer Workflow service that makes it easy to set up, run, automate, and monitor processing of files uploaded using AWS Transfer Family.
- There are no upfront costs, and you pay only for the use of the service.

In the following sections, you can find a description of the different features of Transfer Family, a getting started tutorial, detailed instructions on how to set up the different protocol enabled servers, how to use different types of identity providers, and the service's API reference.

To get started with Transfer Family, see the following:

- How AWS Transfer Family works
- Prerequisites
- Getting started with AWS Transfer Family server endpoints
- Transfer Family web apps

How AWS Transfer Family works

AWS Transfer Family is a fully managed AWS service that you can use to transfer files into and out of Amazon Simple Storage Service (Amazon S3) storage or Amazon Elastic File System (Amazon EFS) file systems over the following protocols or web browser:

- Secure File Transfer Protocol (SFTP): version 3

  The official IETF document is here: SSH File Transfer Protocol draft-ietf-secsh-filexfer-02.txt.
- File Transfer Protocol Secure (FTPS)
- File Transfer Protocol (FTP)
- Applicability Statement 2 (AS2)
- Browser-based transfers

AWS Transfer Family supports up to 3 Availability Zones and is backed by an auto scaling, redundant fleet for your connection and transfer requests. For an example on how to build for higher redundancy and minimize network latency by using Latency-based routing, see the blog post Minimize network latency with your AWS transfer for SFTP servers.

Transfer Family Managed File Transfer Workflows (MFTW) is a fully managed, serverless File Transfer Workflow service that makes it easy to set up, run, automate, and monitor processing of files uploaded using AWS Transfer Family. Customers can use MFTW to automate various processing steps such as copying, tagging, scanning, filtering, compressing/decompressing, and encrypting/decrypting the data that's transferred using Transfer Family. This provides end to end visibility for tracking and auditability. For more details, see AWS Transfer Family managed workflows.

AWS Transfer Family supports any standard file transfer protocol client. Some commonly used clients are the following:

- OpenSSH – A Macintosh and Linux command line utility.
- WinSCP – A Windows-only graphical client.
- Cyberduck – A Linux, Macintosh, and Microsoft Windows graphical client.
- FileZilla – A Linux, Macintosh, and Windows graphical client.

AWS offers the following Transfer Family workshops.

- Build a file transfer solution that leverages AWS Transfer Family for managed SFTP/FTPS endpoints and Amazon Cognito and DynamoDB for user management. You can view the details for this workshop here.
- Build a Transfer Family endpoint with AS2 enabled, and a Transfer Family AS2 connector You can view the details for this workshop here.
- Build a solution that provides prescriptive guidance and a hands on lab on how you can build a scalable and secure file transfer architecture on AWS without needing to modify existing applications or manage server infrastructure. You can view the details for this workshop here.

Blog posts relevant for Transfer Family

The following table lists the blog posts that contain useful information for Transfer Family customers. The table is in reverse chronological order, so that the most recent posts are at the beginning of the table.

Blog post title and link	Date
Deploy Okta as a custom identity provider for AWS Transfer Family	August 11, 2025
Automating paper-to-electronic healthcare claims processing with AWS	May 29, 2025
How to use AWS Transfer Family and GuardDuty for malware protection	April 30, 2025
How FICO modernizes file transfers with ETL automation using AWS Transfer Family	April 24, 2025
Announcing AWS Transfer Family web apps for fully managed Amazon S3 file transfers	December 1, 2024
Six tips to improve the security of your AWS Transfer Family server	September 24, 2024

Blog post title and link	Date
Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family	August 12, 2024
Architecting secure and compliant managed file transfers with AWS Transfer Family SFTP connectors and PGP encryption	May 16, 2024
Using Amazon Cognito as an identity provider with AWS Transfer Family and Amazon S3	May 14, 2024
How Transfer Family can help you build a secure, compliant managed file transfer solution	January 3, 2024
Detect malware threats using AWS Transfer Family	July 20, 2023
Extending SAP workloads with AWS Transfer Family	July 13, 2023
Encrypt and decrypt files with PGP and AWS Transfer Family	June 21, 2023
Authenticating to AWS Transfer Family with Azure Active Directory and AWS Lambda	December 15, 2022
Customize file delivery notifications using AWS Transfer Family managed workflows	October 14, 2022
Building a cloud-native file transfer platform using AWS Transfer Family workflows	January 5, 2022
Enabling user self-service key management with AAWS Transfer Family and AWS Lambda.	December 17, 2021

Blog post title and link	Date
Enhance data access control with AWS Transfer Family and Amazon S3	October 5, 2021
Improve throughput for internet facing file transfers using AWS Global Accelerator and AWS Transfer Family services	June 7, 2021
Securing AWS Transfer Family with AWS Web Application Firewall and Amazon API Gateway	May 5, 2021
Securing AWS Transfer Family with AWS Web Application Firewall and Amazon API Gateway	January 15, 2021
AWS Transfer Family support for Amazon Elastic File System	January 7, 2021
Enable password authentication for AWS Transfer Family using AWS Secrets Manager	November 5, 2020
Centralize data access using AWS Transfer Family and AWS Storage Gateway	June 22, 2020
Using Amazon EFS for AWS Lambda in your serverless applications	June 18, 2020
Use IP allow list to secure your AWS Transfer Family servers	April 8, 2020
Minimize network latency with your AWS transfer for SFTP servers	February 19, 2020
Lift and Shift migration of SFTP servers to AWS	February 12, 2020
Simplify your AWS SFTP Structure with chroot and logical directories	September 26, 2019

Blog post title and link	Date
Using Okta as an identity provider with AWS Transfer Family	May 30, 2019

Prerequisites

The following sections describe the prerequisites required to use the AWS Transfer Family service. At a minimum, you need to create an Amazon Simple Storage Service (Amazon S3) bucket and provide access to that bucket through an AWS Identity and Access Management (IAM) role. Your role also needs to establish a trust relationship. This trust relationship allows Transfer Family to assume the IAM role to access your bucket so that it can service your users' file transfer requests.

For information about IPv6 support for AWS Transfer Family servers, see IPv6 support for Transfer Family servers in the Managing servers chapter.

Topics

- Supported AWS Regions, endpoints and quotas for Transfer Family servers
- Sign up for AWS
- Configure storage to use with AWS Transfer Family servers
- Create an IAM role and policy

Supported AWS Regions, endpoints and quotas for Transfer Family servers

To connect programmatically to an AWS service, you use an endpoint. For example, the endpoint for customers in US East (Ohio) region (`us-east-2`), is `transfer.us-east-2.amazonaws.com`. Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account. In this guide, you can find quotas in AS2 quotas and Quotas for SFTP connectors.

For more information about supported AWS Regions, endpoints, and service quotas, see AWS Transfer Family endpoints and quotas in the *Amazon Web Services General Reference*.

For Transfer Family web apps, the supported regions are listed in AWS Regions for Transfer Family web apps. For quotas that pertain to Transfer Family web apps, see Web app quotas.

Sign up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including AWS Transfer Family. You are charged only for the services that you use.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://signin.aws.amazon.com/signup?request_type=register.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

For information about pricing and to use AWS Pricing Calculator to get an estimate of the cost to use Transfer Family, see AWS Transfer Family pricing.

For information about AWS Region availability, see the AWS Transfer Family endpoints and quotas in the *AWS General Reference*.

Configure storage to use with AWS Transfer Family servers

This topic describes the storage options that you can use with AWS Transfer Family. You can use either Amazon S3 or Amazon EFS as storage for your Transfer Family servers.

Contents

- Configure an Amazon S3 bucket
  - Amazon S3 access points

- Amazon S3 HeadObject behavior
  - Grant ability to only write and list files
  - Large number of zero-byte objects causing latency issues
- Configure an Amazon EFS file system
  - Amazon EFS file ownership
  - Set up Amazon EFS users for Transfer Family
    - Configure Transfer Family users on Amazon EFS
    - Create an Amazon EFS root user
  - Supported Amazon EFS commands

Configure an Amazon S3 bucket

AWS Transfer Family accesses your Amazon S3 bucket to service your users' transfer requests, so you need to provide an Amazon S3 bucket as part of setting up your file transfer protocol-enabled server. You can use an existing bucket, or you can create a new one.

> **Note**
>
> You don't have to use a server and Amazon S3 bucket that are in the same AWS Region, but we recommend this as a best practice.

When you set up your users, you assign them each an IAM role. This role determines the level of access that they have to your Amazon S3 bucket.

For information on creating a new bucket, see How do I create an S3 bucket? in the *Amazon Simple Storage Service User Guide*.

> **Note**
>
> You can use Amazon S3 Object Lock to prevent objects from being overwritten for a fixed amount of time or indefinitely. This works the same way with Transfer Family as with other services. If an object exists and is protected, writing to that file or deleting it is not allowed. For more details on Amazon S3 Object Lock, see Using Amazon S3 Object Lock in the *Amazon Simple Storage Service User Guide*.

Amazon S3 access points

AWS Transfer Family supports Amazon S3 Access Points, a feature of Amazon S3 that allows you to easily manage granular access to shared data sets. You can use S3 Access Point aliases anywhere you use an S3 bucket name. You can create hundreds of access points in Amazon S3 for users who have different permissions to access shared data in an Amazon S3 bucket.

For example, you can use access points to allow three different teams to have access to the same shared dataset where one team can read data from S3, a second team can write data to S3, and the third team can read, write, and delete data from S3. To implement a granular access control as mentioned above, you can create an S3 access point that contains a policy that gives asymmetrical access to different teams. You can use S3 access points with your Transfer Family server to achieve a fine-grained access control, without creating a complex S3 bucket policy that spans hundreds of use cases. To learn more about how to use S3 access points with a Transfer Family server, refer to the Enhance data access control with AWS Transfer Family and Amazon S3 blog post.

> **Note**
>
> AWS Transfer Family does not currently support Amazon S3 Multi-Region Access Points.

Amazon S3 HeadObject behavior

> **Note**
>
> When you create or update a Transfer Family server, you can optimize performance for your Amazon S3 directories, which eliminates `HeadObject` calls.

In Amazon S3, buckets and objects are the primary resources, and objects are stored in buckets. Amazon S3 can mimic a hierarchical file system, but can sometimes behave differently than a typical file system. For example, directories are not a first-class concept in Amazon S3 but instead are based on object keys. AWS Transfer Family infers a directory path by splitting an object's key by the forward slash character (**/**), treating the last element as the file name, then grouping file names which have the same prefix together under the same path. Zero-byte objects are created to represent a folder's path when you create an empty directory using `mkdir` or by using the Amazon S3 console. The key for these objects ends in a trailing forward slash. These zero-byte objects are

described in Organizing objects in the Amazon S3 console using folders in the *Amazon S3 User Guide*.

When you run an `ls` command, and some results are Amazon S3 zero-byte objects (these objects have keys that end in the forward slash character), Transfer Family issues a `HeadObject` request for each of these objects (see HeadObject in the *Amazon Simple Storage Service API Reference* for details). This can result in the following problems when using Amazon S3 as your storage with Transfer Family.

Grant ability to only write and list files

In some cases, you might want to offer only write access to your Amazon S3 objects. For example, you might want to provide access to write (or upload) and list objects in a bucket, but not to read (download) objects. To perform `ls` and `mkdir` commands by using file transfer clients, you must have the Amazon S3 `ListObjects` and `PutObject` permissions. However, when Transfer Family needs to make a `HeadObject` call to either write or list files, the call fails with an error of **Access denied**, because this call requires the `GetObject` permission.

> **Note**
>
> When you create or update a Transfer Family server, you can optimize performance for your Amazon S3 directories, which eliminates `HeadObject` calls.

In this case, you can grant access by adding an AWS Identity and Access Management (IAM) policy condition that adds the `GetObject` permission only for objects that end in a slash (/). This condition prevents `GetObject` calls on files (so that they can't be read), but allows the user to list and traverse folders. The following example policy offers only write and list access to your Amazon S3 buckets. To use this policy, replace *`amzn-s3-demo-bucket`* with the name of your bucket.

> **Note**
>
> To address WinSCP's upload behavior, make sure to add the `"arn:aws:s3:::amzn-s3-demo-bucket/*.filepart"` line as listed in the following example policy. This line ensures proper handling of .filepart objects to prevent failures.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListing",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
            "Sid": "AllowReadWrite",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
         },
      {
            "Sid": "DenyIfNotFolder",
            "Effect": "Deny",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "NotResource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*/",
                "arn:aws:s3:::amzn-s3-demo-bucket/*.filepart"
            ]
         }
      ]
}
```

Note

This policy doesn't allow users to append files. In other words, a user who is assigned this policy can't open files to add content to them, or to modify them. Additionally, if your use case requires a `HeadObject` call before uploading a file, this policy won't work for you.

Large number of zero-byte objects causing latency issues

If your Amazon S3 buckets contain a large number of these zero-byte objects, Transfer Family issues a lot of `HeadObject` calls, which can result in processing delays. The recommended solution for this issue is to enable **Optimized Directories** to reduce latency.

For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds. You set this option in the **Configure additional details** screen during the server creation or update procedure. These procedures are detailed under the Configuring an SFTP, FTPS, or FTP server endpoint topic.

> **Note**
>
> GUI clients may issue an `ls` command outside your control, so it is important to enable this setting if you can.

If you don't or can't optimize your directories, an alternate solution to this problem is to delete all of your zero-byte objects. Note the following:

- Empty directories will no longer exist. Directories only exist as a result of their names being in the key of an object.
- Doesn’t prevent someone from calling `mkdir` and breaking things all over again. You could mitigate this by crafting a policy which prevents directory creation.
- Some scenarios make use of these 0-byte objects. For example, you have a structure like **/inboxes/customer1000** and the inbox directory gets cleaned every day.

Finally, one more possible solution is to limit the number of objects visible through a policy condition to reduce the number of `HeadObject` calls. For this to be a workable solution, you need to accept that you might only be able to view a limited set of all of your sub-directories.

Configure an Amazon EFS file system

AWS Transfer Family accesses Amazon Elastic File System (Amazon EFS) to service your users' transfer requests. So you must provide an Amazon EFS file system as part of setting up your file transfer protocol-enabled server. You can use an existing file system, or you can create a new one.

Note the following:

- When you use a Transfer Family server and an Amazon EFS file system, the server and the file system must be in the same AWS Region.
- The server and the file system don't need to be in the same account. If the server and file system are not in the same account, the file system policy must give explicit permission to the user role.

  For information about how to set up multiple accounts, see Managing the AWS accounts in your organization in the *AWS Organizations User Guide*.
- When you set up your users, you assign them each an IAM role. This role determines the level of access that they have to your Amazon EFS file system.
- For details on mounting an Amazon EFS file system, see Mounting Amazon EFS file systems.

For more details on how AWS Transfer Family and Amazon EFS work together, see Using AWS Transfer Family to access files in your Amazon EFS file system in the *Amazon Elastic File System User Guide*.

Amazon EFS file ownership

Amazon EFS uses the Portable Operating System Interface (POSIX) file permission model to represent file ownership.

In POSIX, users in the system are categorized into three distinct permission classes: When you allow a user to access files stored in an Amazon EFS file system using AWS Transfer Family, you must assign them a “POSIX profile.” This profile is used to determine their access to files and directories in the Amazon EFS file system.

- User (u): Owner of the file or directory. Usually, the creator of a file or directory is also the owner.
- Group (g): Set of users that need identical access to files and directories that they share.
- Others (o): All other users that have access to the system except for the owner and group members. This permission class is also referred to as "Public."

In the POSIX permission model, every file system object (files, directories, symbolic links, named pipes, and sockets) is associated with the previously mentioned three sets of permissions. Amazon EFS objects have a Unix-style mode associated with them. This mode value defines the permissions for performing actions on that object.

Additionally, on Unix-style systems, users and groups are mapped to numeric identifiers, which Amazon EFS uses to represent file ownership. For Amazon EFS, objects are owned by a single owner and a single group. Amazon EFS uses the mapped numeric IDs to check permissions when a user attempts to access a file system object.

Set up Amazon EFS users for Transfer Family

Before you set up your Amazon EFS users, you can do either of the following:

- You can create users and set up their home folders in Amazon EFS. See Configure Transfer Family users on Amazon EFS for details.
- If you are comfortable adding a root user, you can Create an Amazon EFS root user.

> **Note**
>
> Transfer Family servers do not support Amazon EFS access points to set POSIX permissions. Transfer Family users' POSIX profiles (described in the preceding section) offer the ability to set POSIX permissions. These permissions are set at a user level, for granular access, based on UID, GID, and secondary GIDs.

Configure Transfer Family users on Amazon EFS

Transfer Family maps the users to the UID/GID and directories you specify. If the UID/GID/directories do not already exist in EFS, then you should create them before assigning them in Transfer to a user. The details for creating Amazon EFS users is described in Working with users, groups, and permissions at the Network File System (NFS) Level in the *Amazon Elastic File System User Guide*.

Steps to set up Amazon EFS users in Transfer Family

1. Map the EFS UID and GID for your user in Transfer Family using the `PosixProfile` fields.
2. If you want the user to start in a specific folder upon login, you can specify the EFS directory under the `HomeDirectory` field.

You can automate the process, by using a CloudWatch rule and Lambda function. For an example Lambda function that interacts with EFS, see Using Amazon EFS for AWS Lambda in your serverless applications.

Additionally, you can configure logical directories for your Transfer Family users. For details, see the Configure logical directories for Amazon EFS section in the Using logical directories to simplify your Transfer Family directory structures topic.

Create an Amazon EFS root user

If your organization is comfortable for you to enable root user access via SFTP/FTPS for the configuration of your users, you can create a user who's UID and GID are 0 (root user), then use that root user to create folders and assign POSIX ID owners for rest of the users. The advantage of this option is that there is no need to mount the Amazon EFS file system.

Perform the steps described in Adding Amazon EFS service-managed users, and for both the User ID and Group ID, enter 0 (zero).

> **Tip**
>
> Don't let this superuser account exist for longer than necessary. Or, if you do keep the root user account, make sure that you keep it well protected.

Supported Amazon EFS commands

The following commands are supported for Amazon EFS for AWS Transfer Family.

- `cd`
- `ls/dir`
- `pwd`
- `put`
- `get`
- `rename`
- `chown`: Only root (that is, users with uid=0) can change ownership and permissions of files and directories.
- `chmod`: Only root can change ownership and permissions of files and directories.
- `chgrp`: Supported either for root or for the file's owner who can only change a file's group to be one of their secondary groups.
- `ln -s/symlink`

- `mkdir`
- `rm/delete`
- `rmdir`
- `chmtime`

Create an IAM role and policy

This topic describes the types of policies and roles that can be used with AWS Transfer Family, and walks through the process of creating a user role. It also describes how session policies work and provides an example user role.

AWS Transfer Family uses the following types of roles:

- **User role** – Allows service-managed users to access the necessary Transfer Family resources. AWS Transfer Family assumes this role in the context of a Transfer Family user ARN.
- **Access role** – Provides access to only the Amazon S3 files that are being transferred. For inbound AS2 transfers, the access role uses the Amazon Resource Name (ARN) for the agreement. For outbound AS2 transfers, the access role uses the ARN for the connector.
- **Invocation role** – For use with Amazon API Gateway as the server's custom identity provider. Transfer Family assumes this role in the context of a Transfer Family server ARN.
- **Logging role** – Used to log entries into Amazon CloudWatch. Transfer Family uses this role to log success and failure details along with information about file transfers. Transfer Family assumes this role in the context of a Transfer Family server ARN. For outbound AS2 transfers, the logging role uses the connector ARN.
- **Execution role** – Allows a Transfer Family user to call and launch workflows. Transfer Family assumes this role in the context of a Transfer Family workflow ARN.

In addition to these roles, you can also use *session policies*. A session policy is used to limit access when necessary. Note that these policies are stand-alone: that is, you don't add these polices to a role. Rather, you add a session policy directly to a Transfer Family user.

> **Note**
>
> When you are creating a service-managed Transfer Family user, you can select **Auto-generate policy based on home folder**. This is a useful shortcut if you want to limit user access to their own folders. Also, you can view details about session policies and an

example in How session policies work. You can also find more information about session policies in Session policies in the *IAM User Guide*.

Topics

- Create a user role
- How session policies work
- Example read/write access policy

Create a user role

When you create a user, you make a number of decisions about user access. These decisions include which Amazon S3 buckets or Amazon EFS file systems that the user can access, what portions of each Amazon S3 bucket and which files in the file system are accessible, and what permissions the user has (for example, `PUT` or `GET`).

To set access, you create an identity-based AWS Identity and Access Management (IAM) policy and role that provide that access information. As part of this process, you provide access for your user to the Amazon S3 bucket or Amazon EFS file system that is the target or source for file operations. To do this, take the following high-level steps, described in detail later:

Create a user role

1. Create an IAM policy for AWS Transfer Family. This is described in To create an IAM policy for AWS Transfer Family.
2. Create an IAM role and attach the new IAM policy. For an example, see Example read/write access policy.
3. Establish a trust relationship between AWS Transfer Family and the IAM role. This is described in To establish a trust relationship.

The following procedures describe how to create an IAM policy and role.

To create an IAM policy for AWS Transfer Family

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**, and then choose **Create policy**.
3. On the **Create Policy** page, choose the **JSON** tab.

4. In the editor that appears, replace the contents of the editor with the IAM policy that you want attach to the IAM role.

   You can grant read/write access or restrict users to their home directory. For more information, see Example read/write access policy.
5. Choose **Review policy** and provide a name and description for your policy, and then choose **Create policy**.

Next, you create an IAM role and attach the new IAM policy to it.

To create an IAM role for AWS Transfer Family

1. In the navigation pane, choose **Roles**, and then choose **Create role**.

   On the **Create role** page, make sure that **AWS service** is chosen.
2. Choose **Transfer** from the service list, and then choose **Next: Permissions**. This establishes a trust relationship between AWS Transfer Family and AWS.
3. In the **Attach permissions policies** section, locate and choose the policy that you just created, and choose **Next: Tags**.
4. (Optional) Enter a key and value for a tag, and choose **Next: Review**.
5. On the **Review** page, enter a name and description for your new role, and then choose **Create role**.

Next, you establish a trust relationship between AWS Transfer Family and AWS.

To establish a trust relationship

> **Note**
>
> In our examples, we use both `ArnLike` and `ArnEquals`. They are functionally identical, and therefore you may use either when you construct your policies. Transfer Family documentation uses `ArnLike` when the condition contains a wildcard character, and `ArnEquals` to indicate an exact match condition.

1. In the IAM console, choose the role that you just created.
2. On the **Summary** page, choose **Trust relationships**, and then choose **Edit trust relationship**.

3. In the **Edit Trust Relationship** editor, make sure **service** is `"transfer.amazonaws.com"`. The access policy is shown following.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the confused deputy problem. The source account is the owner of the server and the source ARN is the ARN of the user. For example:

```
"Condition": {
    "StringEquals": {
        "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
        "aws:SourceArn": "arn:aws:transfer:region:account_id:user/*"
    }
}
```

You can also use the `ArnLike` condition if you are looking to restrict to a particular server instead of any server in the user account. For example:

```
"Condition": {
    "ArnLike": {
        "aws:SourceArn": "arn:aws:transfer:region:account-id:user/server-id/*"
    }
}
```

> **Note**
>
> In the examples above, replace each *`user input placeholder`* with your own information.

For details on the confused deputy problem and more examples, see Cross-service confused deputy prevention.

4. Choose **Update Trust Policy** to update the access policy.

You have now created an IAM role that allows AWS Transfer Family to call AWS services on your behalf. You attached to the role the IAM policy that you created to give access to your user. In the Getting started with AWS Transfer Family server endpoints section, this role and policy are assigned to your user or users.

See also

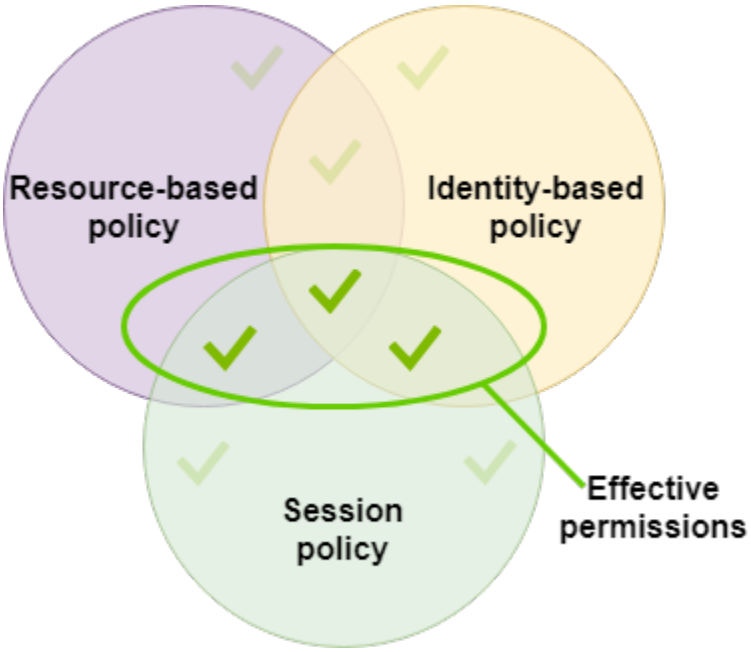
- For more general information about IAM roles, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
- To learn more about identity-based policies for Amazon S3 resources, see Identity and access management in Amazon S3 in the *Amazon Simple Storage Service User Guide*.
- To learn more about identity-based policies for Amazon EFS resources, see Using IAM to control file system data access in the *Amazon Elastic File System User Guide*.

How session policies work

When an administrator creates a role, the role often includes broad permissions to cover multiple use cases or team members. If an administrator configures a console URL, they can reduce permissions for the resulting session by using a *session policy*. For example, if you create a role with read/write access, you can set up a URL that limits users' access to only their home directories.

Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or user. Session policies are useful for locking down users so that they have access only to portions of your bucket where object prefixes contain their username. The following diagram shows that the session policy's permissions are the intersection of the

session policies and the resource-based policies plus the intersection of the session policies and identity-based policies.



For more details, see Session policies in the *IAM User Guide*.

In AWS Transfer Family, a session policy is supported only when you are transferring to or from Amazon S3. The following example policy is a session policy that limits users' access to their home directories only. Note the following:

- The `GetObjectACL` and `PutObjectACL` statements are only required if you need to enable Cross Account Access. That is, your Transfer Family server needs to access a bucket in a different account.
- The maximum length of a session policy is 2048 characters. For more details, see the Policy request parameter for the `CreateUser` action in the *API reference*.
- If your Amazon S3 bucket is encrypted using AWS Key Management Service (AWS KMS), you must specify additional permissions in your policy. For details, see Data protection and encryption.
- To use session policies to create access permissions based on user attributes without creating separate IAM roles for each user, see Dynamic permission management approaches.

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowListingOfUserFolder",
            "Action": [
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::${transfer:HomeBucket}"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "${transfer:HomeFolder}/*",
                        "${transfer:HomeFolder}"
                    ]
                }
            }
        },
        {
            "Sid": "HomeDirObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion",
                "s3:GetObjectVersion",
                "s3:GetObjectACL",
                "s3:PutObjectACL"
            ],
            "Resource": "arn:aws:s3:::${transfer:HomeDirectory}/*"
        }
    ]
}
```

Note

The preceding policy example assumes that users have their home directories set to include a trailing slash, to signify that it is a directory. If, on the other hand, you set a user's

HomeDirectory without the trailing slash, then you should include it as part of your policy.

In the previous example policy, note the use of the `transfer:HomeFolder`, `transfer:HomeBucket`, and `transfer:HomeDirectory` policy parameters. These parameters are set for the `HomeDirectory` that is configured for the user, as described in HomeDirectory and Implementing your API Gateway method. These parameters have the following definitions:

- The `transfer:HomeBucket` parameter is replaced with the first component of `HomeDirectory`.
- The `transfer:HomeFolder` parameter is replaced with the remaining portions of the `HomeDirectory` parameter.
- The `transfer:HomeDirectory` parameter has the leading forward slash (/) removed so that it can be used as part of an S3 Amazon Resource Name (ARN) in a `Resource` statement.

Note

If you are using logical directories—that is, the user's homeDirectoryType is `LOGICAL`—these policy parameters (`HomeBucket`, `HomeDirectory`, and `HomeFolder`) are not supported.

For example, assume that the `HomeDirectory` parameter that is configured for the Transfer Family user is `/home/bob/amazon/stuff/`.

- `transfer:HomeBucket` is set to /home.
- `transfer:HomeFolder` is set to `/bob/amazon/stuff/`.
- `transfer:HomeDirectory` becomes home/bob/amazon/stuff/.

The first `"Sid"` allows the user to list all directories starting from `/home/bob/amazon/stuff/`.

The second `"Sid"` limits the user'put and `get` access to that same path, `/home/bob/amazon/stuff/`.

Example read/write access policy

Grant read/write access to Amazon S3 bucket

The following example policy for AWS Transfer Family grants read/write access to objects in your Amazon S3 bucket.

Note the following:

- Replace `amzn-s3-demo-bucket` with the name of your Amazon S3 bucket.
- The `GetObjectACL` and `PutObjectACL` statements are only required if you need to enable Cross Account Access. That is, your Transfer Family server needs to access a bucket in a different account.
- The `GetObjectVersion` and `DeleteObjectVersion` statements are only required if versioning is enabled on the Amazon S3 bucket that is being accessed.

> **Note**
>
> If you have *ever* enabled versioning for your bucket, then you need these permissions, as you can only suspend versioning in Amazon S3, and not turn it off completely. For details, see Unversioned, versioning-enabled, and versioning-suspended buckets.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListingOfUserFolder",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Sid": "HomeDirObjectAccess",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionTagging",
                "s3:GetObjectACL",
                "s3:PutObjectACL"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Grant file system access to files in Amazon EFS file system

> **Note**
>
> In addition to the policy, you must also make sure your POSIX file permissions are granting the appropriate access. For more information, see Working with users, groups, and permissions at the Network File System (NFS) Level in the *Amazon Elastic File System User Guide*.

The following example policy grants root file system access to files in your Amazon EFS file system.

> **Note**
>
> In the following examples, replace *`region`* with your region, *`account-id`* with the account the file is in, and *`file-system-id`* with the ID of your Amazon Elastic File System (Amazon EFS).

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "RootFileSystemAccess",
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientRootAccess",
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:123456789012:file-
system/file-system-id"
        }
    ]
}
```

The following example policy grants user file system access to files in your Amazon EFS file system.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "UserFileSystemAccess",
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:123456789012:file-
system/file-system-id"
        }
    ]
}
```

Transfer Family tutorials

The AWS Transfer Family user guide provides detailed walkthroughs for several use cases.

- Getting started with AWS Transfer Family server endpoints - Create an SFTP Transfer Family server with a service-managed user and transfer files using a client.
- Setting up and using SFTP connectors - Set up an SFTP connector to transfer files between Amazon S3 storage and an SFTP server.
- Setting up an Amazon API Gateway method as a custom identity provider - Configure an Amazon API Gateway method as a custom identity provider for uploading files to an AWS Transfer Family server.
- Setting up a managed workflow for decrypting a file - Create a managed workflow to decrypt files, upload encrypted files to Amazon S3, and view decrypted content.
- Setting up an AS2 configuration - Set up an AS2 Transfer Family server, including certificate import, profile and agreement creation, optional AS2 connector configuration, and testing.
- The following tutorials guide you through creating and configuring Transfer Family web applications. Choose the tutorial that best fits your needs:

  Tutorial: Setting up a basic Transfer Family web app - Learn how to create and configure a Transfer Family web application.

  Tutorial: Setting up AWS Transfer Family web app with selective multi-bucket access - Configure a Transfer Family web application with granular Amazon S3 bucket permissions for a single user.

Topics

- Getting started with AWS Transfer Family server endpoints
- Setting up a managed workflow for decrypting a file
- Setting up and using SFTP connectors
- Setting up an Amazon API Gateway method as a custom identity provider
- Setting up an AS2 configuration
- Tutorial: Setting up a basic Transfer Family web app
- Tutorial: Setting up AWS Transfer Family web app with selective multi-bucket access

Getting started with AWS Transfer Family server endpoints

Use this tutorial to get started with AWS Transfer Family (Transfer Family). You'll learn how to create an SFTP-enabled server with publicly accessible endpoint using Amazon S3 storage, add a user with service-managed authentication, and transfer a file with Cyberduck.

Topics

- Prerequisites
- Step 1: Sign in to the AWS Transfer Family console
- Step 2: Create an SFTP-enabled server
- Step 3: Add a service managed user
- Step 4: Transfer a file using a client

Prerequisites

Before you begin, be sure to complete the requirements in Prerequisites. As part of this setup, you create an Amazon Simple Storage Service (Amazon S3) bucket and an AWS Identity and Access Management (IAM) user role.

There are permissions required for using the AWS Transfer Family console, and there are permissions required for configuring other AWS services that Transfer Family uses, such as Amazon Simple Storage Service, AWS Certificate Manager, Amazon Elastic File System, and Amazon Route 53. For example, for users that are transferring files into and out of AWS using Transfer Family, **AmazonS3FullAccess** grants permissions to setup and use an Amazon S3 bucket. Some of the permissions in this policy are needed to create Amazon S3 buckets.

To use the Transfer Family console, you require the following:

- **AWSTransferConsoleFullAccess** grants permissions for your SFTP user to create Transfer Family resources.
- **IAMFullAccess** (or specifically a policy that allows creation of IAM roles) is only needed if you want Transfer Family to automatically create a logging role for your server in Amazon CloudWatch Logs or a user role for a user logging into a server.
- To create and delete VPC server types, you need to add the actions **ec2:CreateVpcEndpoint** and **ec2:DeleteVpcEndpoints** to your policy. For information about limiting VPC endpoint access for security purposes, see Limiting VPC endpoint access for Transfer Family servers.

> **Note**
>
> The **AmazonS3FullAccess** and **IAMFullAccess** polices are, themselves, not needed for general usage of AWS Transfer Family. They are presented here as a simple way to make sure that all of the permissions that you need are covered. Additionally, these are AWS managed policies, which are standard policies that are available to all AWS customers. You can view the individual permissions in these policies and determine a minimal set that you need for your purposes.

Step 1: Sign in to the AWS Transfer Family console

To sign in to Transfer Family

1. Sign in to the AWS Management Console and open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. For **Account ID or alias**, enter the ID for your AWS account.
3. For **IAM user name**, enter the name of the user role that you created for Transfer Family.
4. For **Password**, enter your AWS account password.
5. Choose **Sign in**.

Step 2: Create an SFTP-enabled server

Secure Shell (SSH) File Transfer Protocol (SFTP) is a network protocol used for secure transfer of data over the internet. The protocol supports the full security and authentication functionality of SSH. It is widely used to exchange data, including sensitive information between business partners in a variety of industries such as financial services, healthcare, retail, and advertising.

To create an SFTP-enabled server

1. Select **Servers** from the Navigation pane then choose **Create server**.
2. In **Choose protocols**, select **SFTP**, and then choose **Next**.
3. In **Choose an identity provider**, choose **Service managed** to store user identities and keys in Transfer Family, and then choose **Next**.
4. In **Choose an endpoint**, do the following:

a. For **Endpoint type**, choose the **Publicly accessible** endpoint type.

b. For **Custom hostname**, choose **None**.

c. Choose **Next**.

5. In **Choose a domain**, choose **Amazon S3**.

6. In **Configure additional details**, for **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server. Our latest security policy is the default: for details, see Security policies for AWS Transfer Family servers.

> **Note**
>
> Only if you are adding a managed workflow for your server, choose **Create a new role** for **CloudWatch logging**. To log server events, you do not need to create an IAM role.

7. In **Review and create**, choose **Create server**. You are taken to the **Servers** page.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations, but you'll need to create a user first. For details on creating users, see Managing users for server endpoints.

Step 3: Add a service managed user

To add a user to the SFTP-enabled server

1. On the **Servers** page, select the server that you want to add a user to.

2. Choose **Add user**.

3. In the **User configuration** section, for **Username**, enter the username. This username must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A-Z, 0–9, underscore '_', hyphen '-', period '.' and at sign '@'. The username can't start with a hyphen '-', period '.' or at sign '@'.

4. For **Access**, choose the IAM role that you created in Create an IAM role and policy. This IAM role includes an IAM policy that contains permissions to access your Amazon S3 bucket, as well as a trust relationship with the AWS Transfer Family service. The procedure outlined in To establish a trust relationship shows how to establish the proper trust relationship.

5. For **Policy**, choose **None**.

6. For **Home directory**, choose the Amazon S3 bucket where you want to store the data that you transfer using AWS Transfer Family. Enter the path to the home directory. This is the directory that your users see when they log in using their client.

   We recommend using a directory path that contains the username so that you have the option to use a session policy. A session policy limits a user's access in the Amazon S3 bucket to that user's home directory. For more information about using session policies, see How session policies work.

   If you prefer, you can keep this parameter blank to use your Amazon S3 bucket's `root` directory. If you choose this option, make sure that your IAM role provides access to the`root` directory.
7. Select the **Restricted** check box to prevent your users from accessing anything outside of their home directory. This also prevents users from seeing the Amazon S3 bucket name or folder name.
8. For **SSH public key**, enter the public SSH key portion of the SSH key pair in `ssh-rsa <string>` format.

   Your key must be validated by the service before you can add your new user. For more information about how to generate an SSH key pair, see Generate SSH keys for service-managed users.
9. (Optional) For **Key** and **Value**, enter one or more tags as key-value pairs, and choose **Add tag**.
10. Choose **Add** to add your new user to the server that you chose.

    The new user appears in the **Users** section of the **Server details** page.

Step 4: Transfer a file using a client

You transfer files over the AWS Transfer Family service by specifying the transfer operation in a client. AWS Transfer Family supports several clients. For details, see Transferring files over a server endpoint using a client

This section contains procedures for using Cyberduck and OpenSSH.

Topics

- Use Cyberduck
- Use OpenSSH

Use Cyberduck

To transfer files over AWS Transfer Family using Cyberduck

1. Open the Cyberduck client.
2. Choose **Open Connection**.
3. In the **Open Connection** dialog box, choose **SFTP (SSH File Transfer Protocol)**.
4. For **Server**, enter your server endpoint. The server endpoint is located on the **Server details** page, see View SFTP, FTPS, and FTP server details.
5. For **Port number**, enter **22** for SFTP.
6. For **Username**, enter the name for the user that you created in Managing users for server endpoints.
7. For **SSH Private Key**, choose or enter the SSH private key.
8. Choose **Connect**.
9. Perform your file transfer.

   Depending on where your files are, do one of the following:

   - In your local directory (the source), choose the files that you want to transfer, and drag and drop them into the Amazon S3 directory (the target).
   - In the Amazon S3 directory (the source), choose the files that you want to transfer, and drag and drop them into your local directory (the target).

Use OpenSSH

Use the instructions that follow to transfer files from the command line using OpenSSH.

> **Note**
>
> This client works only with an SFTP-enabled server.

To transfer files over AWS Transfer Family using the OpenSSH command line utility

1. On Linux or Macintosh, open a command terminal.

2. At the prompt, enter the following command: `% sftp -i transfer-key sftp_user@service_endpoint`

   In the preceding command, `sftp_user` is the username and `transfer-key` is the SSH private key. Here, `service_endpoint` is the server's endpoint as shown in the AWS Transfer Family console for the selected server.

   An `sftp` prompt should appear.

3. (Optional) To view the user's home directory, enter the following command at the `sftp` prompt: `sftp> pwd`

4. On the next line, enter the following text: `sftp> cd /amzn-s3-demo-bucket/home/sftp_user`

   In this getting-started exercise, this Amazon S3 bucket is the target of the file transfer.

5. On the next line, enter the following command: `sftp> put filename.txt`

   The `put` command transfers the file into the Amazon S3 bucket.

   A message like the following appears, indicating that the file transfer is in progress, or complete.

   ```
   Uploading filename.txt to /amzn-s3-demo-bucket/home/sftp_user/
   filename.txt

   some-file.txt 100% 127 0.1KB/s 00:00
   ```

Setting up a managed workflow for decrypting a file

This tutorial illustrates how to set up a managed workflow that contains a decrypt step. The tutorial also shows how to upload an encrypted file to an Amazon S3 bucket and then view the decrypted file in that same bucket.

> **Note**
>
> The AWS storage blog has a post that describes how to simply decrypt files without writing any code using Transfer Family Managed workflows, Encrypt and decrypt files with PGP and AWS Transfer Family.

Topics

- Step 1: Configure an execution role
- Step 2: Create a managed workflow
- Step 3: Add the workflow to a server and create a user
- Step 4: Create a PGP key pair
- Step 5: Store the PGP private key in AWS Secrets Manager
- Step 6: Encrypt a file
- Step 7: Run the workflow and view the results

Step 1: Configure an execution role

Create an AWS Identity and Access Management (IAM) execution role that Transfer Family can use to launch a workflow. The process of creating an execution role is described in IAM policies for workflows.

> **Note**
>
> As part of creating an execution role, make sure to establish a trust relationship between the execution role and Transfer Family, as described in To establish a trust relationship.

The following execution role policy contains all the required permissions to start the workflow that you create in this tutorial. To use this example policy, replace the *`user input placeholders`* with your own information. Replace `amzn-s3-demo-bucket` with the name of the Amazon S3 bucket where you upload your encrypted files.

> **Note**
>
> Not every workflow requires every permission that's listed in this example. You can restrict permissions based on the types of steps in your specific workflow. The permissions needed for each predefined step type are described in Use predefined steps. The permissions needed for a custom step are described in IAM permissions for a custom step.

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "WorkflowsS3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "s3:PutObjectTagging",
                "s3:ListBucket",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Sid": "DecryptSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:us-east-1:123456789012:secret:aws/
transfer/*"
        }
    ]
}
```

Step 2: Create a managed workflow

Now you need to create a workflow that contains a decrypt step.

To create a workflow that contains a decrypt step

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Workflows**, and then choose **Create workflow**.

3. Enter the following details:

   - Enter a description, for example **`Decrypt workflow example`**.
   - In the **Nominal steps** section, choose **Add step**.
4. For **Choose step type**, choose **Decrypt file**, and then choose **Next**.
5. In the **Configure parameters** dialog box, specify the following:

   - Enter a descriptive step name, for example, **`decrypt-step`**. Spaces are not allowed in step names.
   - For the **Destination for decrypted files**, choose Amazon S3.
   - For the **Destination bucket name**, choose the same Amazon S3 bucket that you specified as the `amzn-s3-demo-bucket` in the IAM policy that you created in Step 1.
   - For the **Destination key prefix**, enter the name of the prefix (folder) where you want to store your decrypted files in your destination bucket, for example, **`decrypted-files/`**.

     > **Note**
     >
     > Make sure to add a trailing slash (**/**) to your prefix.

   - For this tutorial, leave **Overwrite existing** cleared. When this setting is cleared, if you try to decrypt a file with the identical name of an existing file, the workflow processing stops, and the new file is not processed.

   Choose **Next** to move to the review screen.
6. Review the details for the step. If everything is correct, choose **Create step**.
7. Your workflow needs only the single decrypt step, so there are no additional steps to configure. Choose **Create workflow** to create the new workflow.

Note the workflow ID for your new workflow. You will need this ID for the next step. This tutorial uses *`w-1234abcd5678efghi`* as the example workflow ID.

Step 3: Add the workflow to a server and create a user

Now that you have a workflow with a decrypt step, you must associate it with a Transfer Family server. This tutorial shows how to attach the workflow to an existing Transfer Family server. Alternatively, you can create a new server to use with your workflow.

After you attach the workflow to a server, you must create a user that can SFTP into the server and trigger the workflow to run.

To configure a Transfer Family server to run a workflow

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**, and then choose a server from the list. Make sure that this server supports the SFTP protocol.
3. On the details page for the server, scroll down to the **Additional details** section, and then choose **Edit**.
4. On the **Edit additional details** page, in the **Managed workflows** section, choose your workflow, and choose a corresponding execution role.
   - For **Workflow for complete file uploads**, choose the workflow that you created in Step 2: Create a managed workflow, for example, **`w-1234abcd5678efghi`**.
   - For **Managed workflows execution role**, choose the IAM role that you created in Step 1: Configure an execution role.
5. Scroll to the bottom of the page, and choose **Save** to save your changes.

Note the ID for the server that you are using. The name of the AWS Secrets Manager secret that you use to store your PGP keys is based in part on the server ID.

To add a user that can trigger the workflow

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**, and then choose the server that you're using for the decrypt workflow.
3. On the server details page, scroll down to the **Users** section, and choose **Add user**.
4. For your new user, enter the following details:
   - For **Username**, enter **`decrypt-user`**.
   - For **Role**, choose a user role that can access your server.
   - For **Home directory**, choose the Amazon S3 bucket that you used earlier, for example, `amzn-s3-demo-bucket`.
   - For **SSH public keys**, paste in a public key that corresponds to a private key that you have. For details, see Generate SSH keys for service-managed users.

5. Choose **Add** to save your new user.

Note the name of your Transfer Family user for this server. The secret is partially based on the name of the user. For simplicity, this tutorial uses a default secret that can be used by any user of the server.

Step 4: Create a PGP key pair

Use one of the supported PGP clients to generate a PGP key pair. This process is described in detail in Generate PGP keys.

To generate a PGP key pair

1. For this tutorial, you can use gpg (GnuPG) version 2.0.22 client to generate a PGP key pair that uses RSA as the encryption algorithm. For this client, run the following command, and provide an email address and a passphrase. You can use any name or email address that you like. Make sure that you remember the values that you use, because you will need to enter them later in the tutorial.

   ```
   gpg --gen-key
   ```

   > **Note**
   >
   > If you're using GnuPG version 2.3.0 or newer, you must run `gpg --full-gen-key`. When prompted for the type of key to create, choose RSA or ECC. If you choose **ECC**, you can choose from NIST, BrainPool and Curve25519 for the elliptic curve.

2. Export the private key by running the following command. Replace *user@example.com* with the email address that you used when you generated the key.

   ```
   gpg --output workflow-tutorial-key.pgp --armor --export-secret-key user@example.com
   ```

   This command exports the private key to the **workflow-tutorial-key.pgp** file. You can name the output file anything that you like. You can also delete the private key file after you have added it to AWS Secrets Manager.

Step 5: Store the PGP private key in AWS Secrets Manager

You need to store the private key in Secrets Manager, in a very specific way, so that the workflow can find the private key when the workflow runs a decrypt step on an uploaded file.

> **Note**
>
> When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see AWS Secrets Manager Pricing.

To store a PGP private key in Secrets Manager

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at https://console.aws.amazon.com/secretsmanager/.
2. In the left navigation pane, choose **Secrets**.
3. On the **Secrets** page, choose **Store a new secret**.
4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
5. In the **Key/value pairs** section, choose the **Key/value** tab.
   - **Key** – Enter **`PGPPrivateKey`**.
   - **value** – Paste the text of your private key into the value field.
6. Choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.
   - **Key** – Enter **`PGPPassphrase`**.
   - **value** – Enter the passphrase that you used when you generated your PGP key pair in Step 4: Create a PGP key pair.
7. Choose **Next**.
8. On the **Configure secret** page, enter a name and description for your secret. You can create a secret for a specific user or one that can be used by all users. If your server ID is *`s-11112222333344445`*, you name the secret as follows.
   - To create a default secret for all users, name the secret **`aws/transfer/`*`s-11112222333344445`*`/@pgp-default`**.
   - To create a secret only for the user that you created earlier, name the secret **`aws/transfer/`*`s-11112222333344445`*`/decrypt-user`**.

9. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
10. On the **Review** page, choose **Store** to create and store the secret.

For more information about adding your PGP private key to Secrets Manager, see Use AWS Secrets Manager to store your PGP key.

Step 6: Encrypt a file

Use the gpg program to encrypt a file for use in your workflow. Run the following command to encrypt a file:

```
gpg -e -r marymajor@example.com --openpgp testfile.txt
```

Before running this command, note the following:

- For the `-r` argument, replace *marymajor@example.com* with the email address that you used when you created the PGP key pair.
- The `--openpgp` flag is optional. This flag makes the encrypted file conform to the OpenPGP RFC4880 standard.
- This command creates a file named **`testfile.txt.gpg`** in the same location as **`testfile.txt`**.

> **Important**
>
> When encrypting files for use with AWS Transfer Family workflows, always ensure you specify a non-anonymous recipient using the `-r` parameter. Anonymous encryption (without specifying a recipient) can cause decryption failures in the workflow because the system won't be able to identify which key to use for decryption. Debugging information for this issue is available at Troubleshoot anonymous recipient encryption issues.

Step 7: Run the workflow and view the results

To run the workflow, you connect to the Transfer Family server with the user that you created in Step 3. Then you can look in the Amazon S3 bucket that you specified in Step 2.5, configure destination parameters to see the decrypted file.

To run the decrypt workflow

1. Open a command terminal.
2. Run the following command, replacing *your-endpoint* with your actual endpoint, and *transfer-key* with your user's SSH private key:

   ```
   sftp -i transfer-key decrypt-user@your-endpoint
   ```

   For example, if the private key is stored in `~/.ssh/decrypt-user`, and your endpoint is `s-11112222333344445.server.transfer.us-east-2.amazonaws.com`, the command is as follows:

   ```
   sftp -i  ~/.ssh/decrypt-user decrypt-user@s-11112222333344445.server.transfer.us-
   east-2.amazonaws.com
   ```

3. Run the pwd command. If successful, this command will return the following:

   ```
   Remote working directory: /amzn-s3-demo-bucket/decrypt-user
   ```

   Your directory reflects the name of your Amazon S3 bucket.
4. Run the following command to upload the file and trigger the workflow to run:

   ```
   put testfile.txt.gpg
   ```

5. For the destination of the decrypted files, you specified the `decrypted-files/` folder when you created the workflow. Now, you can navigate to that folder and list the contents.

   ```
   cd ../decrypted-files/
   ls
   ```

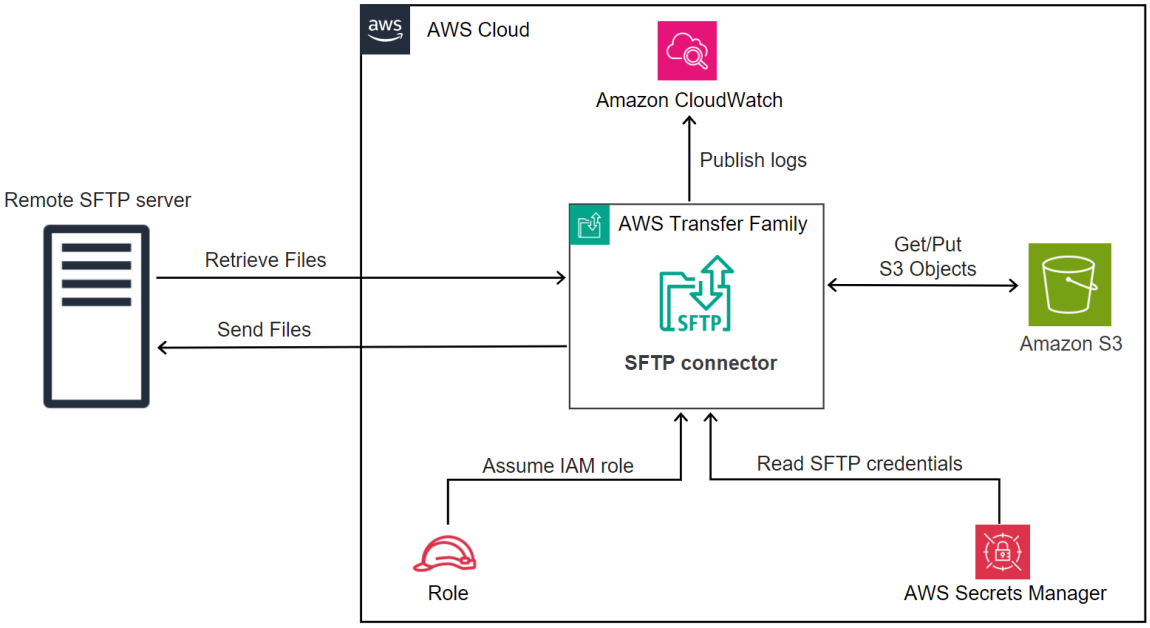
   If successful, the `ls` command lists the `testfile.txt` file. You can download this file and verify that it is the same as the original file that you encrypted earlier.

Setting up and using SFTP connectors

The purpose of a connector is to establish a relationship between your AWS storage and a partner's SFTP server. You can send files from Amazon S3 to an external, partner-owned destination. You can also use an SFTP connector to retrieve files from a partner's SFTP server.

This tutorial illustrates how to set up an SFTP connector with both service managed and VPC_LATTICE egress types, and then transfer files between Amazon S3 storage and an SFTP server.

An SFTP connector retrieves SFTP credentials from AWS Secrets Manager to authenticate into a remote SFTP server and establish a connection. The connector sends files to or retrieves files from the remote server, and stores the files in Amazon S3. You can choose between service managed egress (using AWS managed infrastructure) or VPC egress (routing through your VPC using Cross-VPC Resource Access). An IAM role is used to allow access to the Amazon S3 bucket and to the credentials stored in Secrets Manager. And you can log to Amazon CloudWatch.



The following blog post provides a reference architecture to build an MFT workflow using SFTP connectors, including encryption of files using PGP before sending them to a remote SFTP server using SFTP connectors: Architecting secure and compliant managed file transfers with AWS Transfer Family SFTP connectors and PGP encryption.

Connector egress types

SFTP connectors support two egress types that determine how your connector routes traffic to remote SFTP servers:

- **SERVICE_MANAGED** (default): Uses AWS Transfer Family managed infrastructure with static IP addresses for outbound connections.
- **VPC**: Routes traffic through your VPC using Cross-VPC Resource Access, enabling private endpoint connectivity and use of your own NAT gateways.

This tutorial covers both egress types. Choose VPC egress type when you need to:

- Connect to private SFTP servers in your VPC (private IP addresses)
- Connect to on-premises SFTP servers via Direct Connect or VPN
- Route public endpoint traffic through your VPC for security controls
- Use your own Elastic IP addresses for outbound connections

Topics

- Step 1: Create the necessary supporting resources
- Step 2: Create and test an SFTP connector
- Step 3: Send and retrieve files using the SFTP connector
- Procedures to create a Transfer Family server to use as your remote SFTP server

Step 1: Create the necessary supporting resources

You can use SFTP connectors to copy files between Amazon S3 and any remote SFTP server. For this tutorial, we are using an AWS Transfer Family server as our remote SFTP server. We need to create and configure the following resources:

- Create Amazon S3 buckets to store files in your AWS environment, and to send and retrieve files from the remote SFTP server: Create Amazon S3 buckets.
- Create an AWS Identity and Access Management role for accessing Amazon S3 storage and our secret in Secrets Manager: Create an IAM role with the necessary permissions.
- Create a Transfer Family server that uses the SFTP protocol, and a service-managed user that uses the SFTP connector to transfer files to or from the SFTP server: Create a Transfer Family SFTP server and a user.
- Create an AWS Secrets Manager secret that stores the credentials used by the SFTP connector to log in to the remote SFTP server: Create and store a secret in AWS Secrets Manager.

For VPC egress type connectors, you also need:

- A VPC with appropriate subnets and security groups
- A Resource Gateway (minimum 2 Availability Zones required): Create a Resource Gateway (VPC egress type only).
- A Resource Configuration pointing to your SFTP server: Create a Resource Configuration (VPC egress type only). For more information, see Resource configurations in the *Amazon VPC Lattice User Guide*.

Create Amazon S3 buckets

To create an Amazon S3 bucket

1. Sign in to the AWS Transfer Family console at https://console.aws.amazon.com/s3/.
2. Choose a Region and enter a name.

   For this tutorial, our bucket is in **`US East (N. Virginia) us-east-1`**, and the name is **`sftp-server-storage-east`**.
3. Accept the defaults and choose **Create bucket**.

For complete details about creating Amazon S3 buckets, see How do I create an S3 bucket? in the *Amazon Simple Storage Service User Guide*.

Create an IAM role with the necessary permissions

For the access role, create a policy with the following permissions.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Sid": "AllowListingOfUserFolder",
        "Action": [
            "s3:ListBucket",
            "s3:GetBucketLocation"
        ],
```

```
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ]
    },
    {
        "Sid": "HomeDirObjectAccess",
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:GetObjectVersion",
            "s3:GetObjectACL",
            "s3:PutObjectACL"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
        "Sid": "GetConnectorSecretValue",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue"
        ],
        "Resource": "arn:aws:secretsmanager:us-west-2:111122223333:secret:aws/
transfer/SecretName-6RandomCharacters"
    }
  ]
}
```

Replace items as follows:

- For `amzn-s3-demo-bucket`, the tutorial uses **`sftp-server-storage-east`**.
- For *`region`*, the tutorial uses **`us-east-1`**.
- For *`account-id`*, use your AWS account ID.
- For *`SecretName-6RandomCharacters`*, we are **`using sftp-connector1`** for the name (you will have your own six random characters for your secret).

You must also make sure that this role contains a trust relationship that allows the connector to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see To establish a trust relationship.

> **Note**
>
> To see the details for the role that we are using for the tutorial, see Combined user and access role.

Create and store a secret in AWS Secrets Manager

We need to store a secret in Secrets Manager to store user credentials for your SFTP connector. You can use a password, SSH private key, or both. For the tutorial, we are using a private key.

> **Note**
>
> When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see AWS Secrets Manager Pricing.

Before you begin the procedure to store the secret, retrieve and format your private key. The private key must correspond to the public key that is configured for the user on the remote SFTP server. For our tutorial, the private key must correspond to the public key that is stored for our test user on the Transfer Family SFTP server that we are using as remote server.

To do this, run the following command:

```
jq -sR . path-to-private-key-file
```

For example, if your private key file is located in `~/.ssh/sftp-testuser-privatekey`, the command is as follows.

```
jq -sR . ~/.ssh/sftp-testuser-privatekey
```

This outputs the key in the correct format (with embedded newline characters) to standard output. Copy this text somewhere, as you need to paste it in the following procedure (in step 6).

To store user credentials in Secrets Manager for an SFTP connector

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at https://console.aws.amazon.com/secretsmanager/.
2. In the left navigation pane, choose **Secrets**.
3. On the **Secrets** page, choose **Store a new secret**.
4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
5. In the **Key/value pairs** section, choose the **Key/value** tab.
   - **Key** — Enter **`Username`**.
   - **value** — Enter the name of our user, **`sftp-testuser`**.
6. To enter the key, we recommend that you use the **Plaintext** tab.
   a. Choose **Add row**, then enter **`PrivateKey`**.
   b. Choose the **Plaintext** tab. The field now contains the following text:

   ```
   {"Username":"sftp-testuser","PrivateKey":""}
   ```

   c. Paste in the text for your private key (saved earlier) between the empty double quotes (""). Your screen should look as follows (key data is grayed out).



7. Choose **Next**.

8. On the **Configure secret** page, enter a name for your secret. For this tutorial, we name the secret **`aws/transfer/sftp-connector1`**.
9. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
10. On the **Review** page, choose **Store** to create and store the secret.

Create a Resource Gateway (VPC egress type only)

For VPC egress type connectors, you need to create a Resource Gateway in your VPC. The Resource Gateway serves as the entry point for Cross-VPC Resource Access.

To create a Resource Gateway

1. Run the following command to create a Resource Gateway (replace the VPC ID and subnet IDs with your values):

```
aws vpc-lattice create-resource-gateway \
    --name my-sftp-resource-gateway \
    --vpc-identifier vpc-12345678 \
    --subnet-ids subnet-12345678 subnet-87654321
```

> **Note**
>
> Resource Gateways require subnets in at least 2 Availability Zones.

2. Note the Resource Gateway ID from the response for use in the next step.

Create a Resource Configuration (VPC egress type only)

Create a Resource Configuration that points to your SFTP server. This can be a private IP address for servers in your VPC, or a public DNS name for external servers. For more information about Resource Configurations, see Resource configurations in the *Amazon VPC Lattice User Guide*.

To create a Resource Configuration

1. For a private SFTP server, run:

```
aws vpc-lattice create-resource-configuration \
    --name my-sftp-resource-config \
```

```
    --port-ranges 22 \
    --type SINGLE \
    --resource-gateway-identifier rgw-12345678 \
    --resource-configuration-definition ipResource={ipAddress="10.0.1.100"}
```

2. For a public SFTP server (DNS name only), run:

```
aws vpc-lattice create-resource-configuration \
    --name my-public-sftp-resource-config \
    --port-ranges 22 \
    --type SINGLE \
    --resource-gateway-identifier rgw-12345678 \
    --resource-configuration-definition dnsResource={domainName="sftp.example.com"}
```

> **Note**
>
> Public endpoints must use DNS names, not IP addresses.

3. Note the Resource Configuration ARN from the response for use when creating the connector.

Step 2: Create and test an SFTP connector

In this section, we create an SFTP connector that uses all of the resources that we created earlier. For more details, see Creating SFTP connectors.

To create an SFTP connector

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **SFTP Connectors**, then choose **Create SFTP connector**.
3. For **Egress type**, choose one of the following:

   - **Service managed** (default): Uses AWS Transfer Family managed infrastructure with static IP addresses for outbound connections.
   - **VPC Lattice**: Routes traffic through your VPC using Cross-VPC Resource Access. Choose this option for private endpoint connectivity or to use your own NAT gateways.

> **Important**
>
> You cannot change the egress type after creating the connector. Choose carefully based on your connectivity requirements.

4. In the **Connector configuration** section, provide the following information:

   - For the **URL**, enter the URL of the remote SFTP server. For the tutorial, we enter the URL of the Transfer Family server that we are using as the remote SFTP server.

     ```
     sftp://s-1111aaaa2222bbbb3.server.transfer.us-east-1.amazonaws.com
     ```

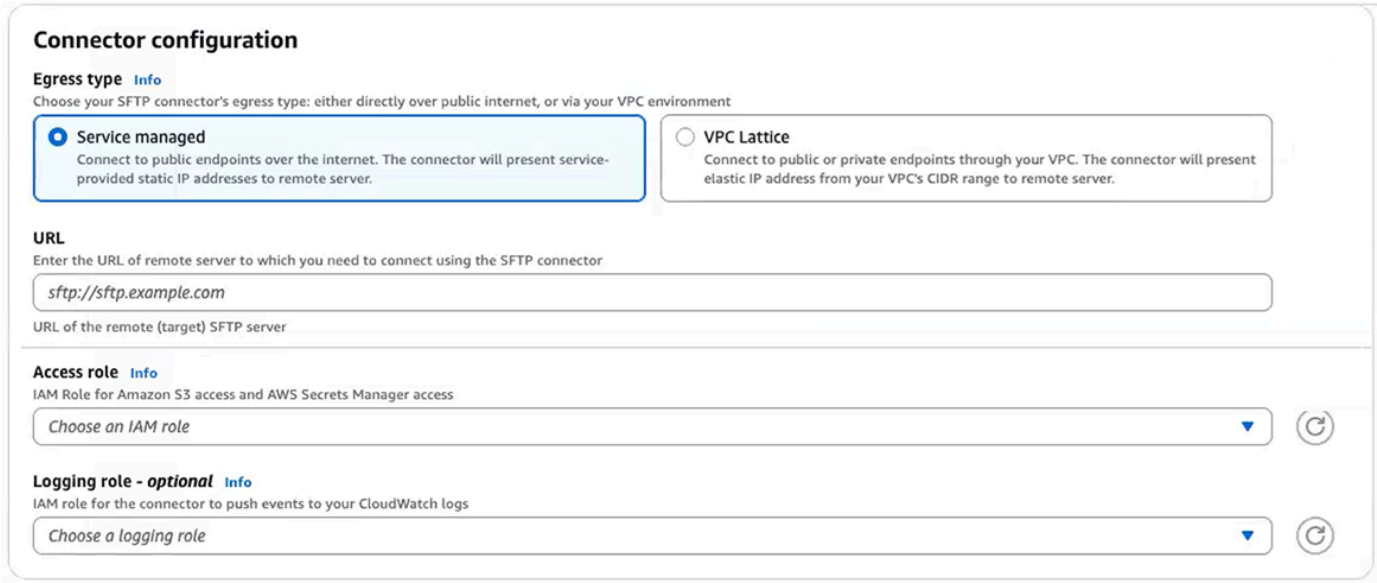
     Replace *1111aaaa2222bbbb3* with your Transfer Family server ID.

   - For the **Access role**, enter the role we created earlier, **`sftp-connector-role`**.
   - For **Resource Configuration ARN** (VPC Lattice egress type only), enter the ARN of the Resource Configuration you created earlier:

     ```
     arn:aws:vpc-lattice:us-east-1:account-id:resourceconfiguration/rcfg-12345678
     ```

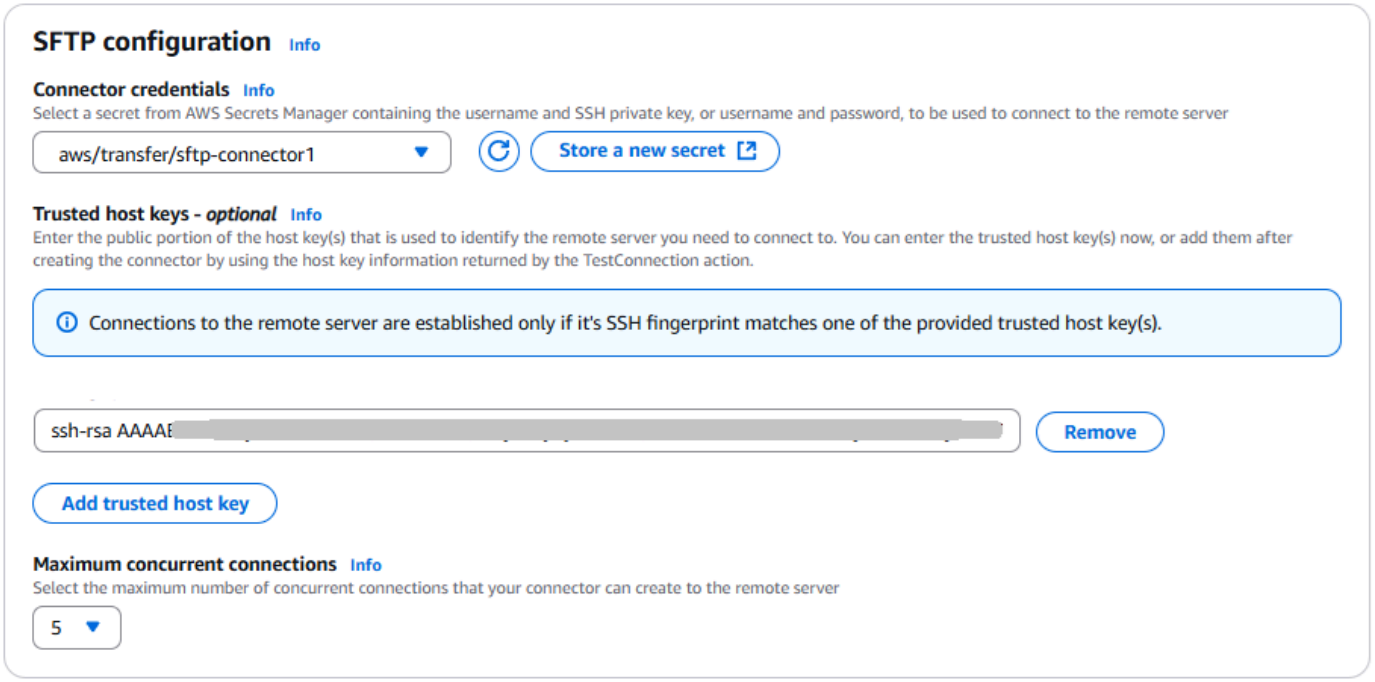
   - For the **Logging role**, choose a role that includes a trust policy with `transfer.amazonaws.com` in the Principal element.

     Tip: In addition to adding Transfer Family as a trusted entity, you can add the **AWSTransferLoggingAccess** AWS managed policy to the role. This policy is described in detail in AWSTransferLoggingAccess.



5. In the **SFTP Configuration** section, provide the following information:

   - For **Connector credentials**, choose the name of your Secrets Manager resource that contains SFTP credentials. For the tutorial, choose **`aws/transfer/sftp-connector1`**.
   - For **Trusted host keys** , paste in the public portion of the host key. You can retrieve this key by running `ssh-keyscan` for your SFTP server. For details on how to format and store the trusted host key, see the SftpConnectorConfig data type documentation.
   - For **Maximum concurrent connections**, select an integer value from 1 to 5: the default value is 5.



6. After you have confirmed all of your settings, choose **Create connector** to create the SFTP connector.

You can also create connectors using the AWS Command Line Interface.

- To create an SFTP connector with service-managed egress run the following command:

```
aws transfer create-connector \
    --url "sftp://s-1111aaaa2222bbbb3.server.transfer.us-east-1.amazonaws.com" \
    --access-role "arn:aws::iam::account-id:role/sftp-connector-role" \
    --sftp-config UserSecretId="aws/transfer/sftp-connector1",TrustedHostKeys="ssh-
rsa AAAAB3NzaC..."
```

- To create an SFTP connector with VPC-based egress run the following command:

```
aws transfer create-connector \
   --url "sftp://my.sftp.server.com:22" \
   --access-role "arn:aws::iam::account-id:role/sftp-connector-role" \
   --sftp-config UserSecretId="aws/transfer/sftp-connector1",TrustedHostKeys="ssh-rsa
 AAAAB3NzaC..." \
   --egress-config VpcLattice={ResourceConfigurationArn="arn:aws:vpc-lattice:us-
east-1:account-id:resourceconfiguration/rcfg-12345678",PortNumber=22}
```

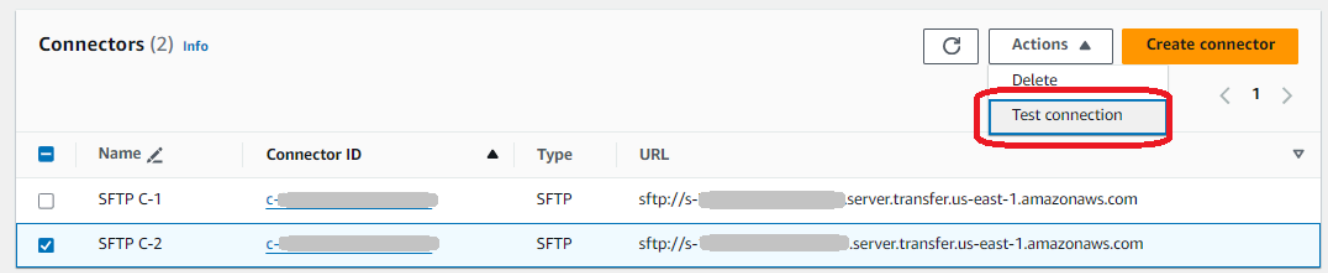
After you create an SFTP connector, we recommend that you test it before you attempt to transfer any files using your new connector.

> **Note**
>
> For VPC egress type connectors, DNS resolution may take several minutes after creation. During this time, the connector status will be `PENDING` and `TestConnection` will return "Connector not available". Wait for the status to become `ACTIVE` before attempting file transfers.

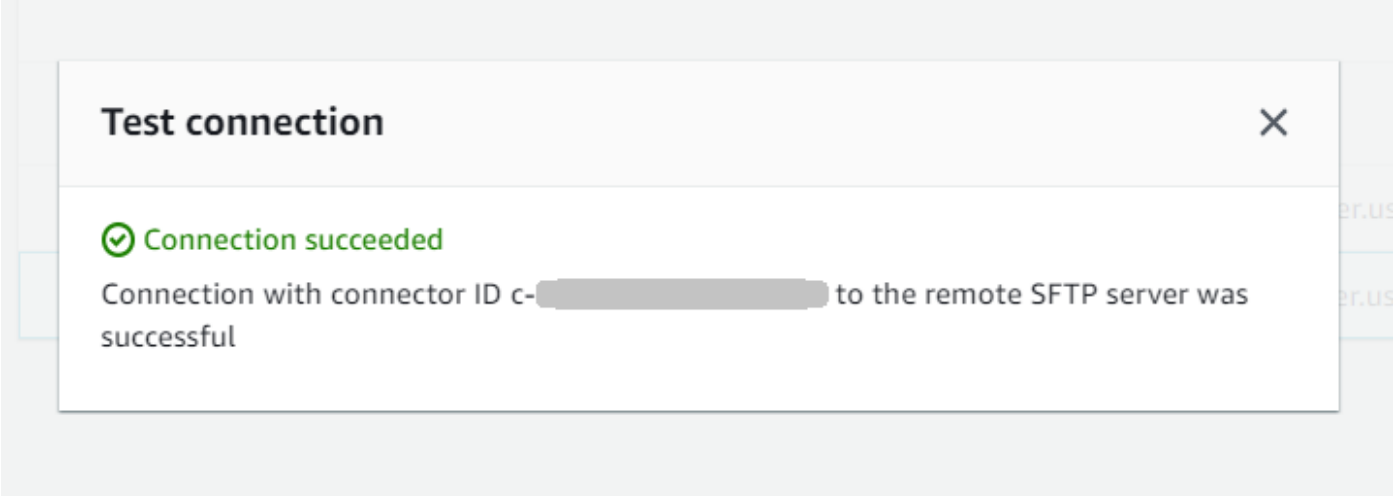
Test a connector using the console

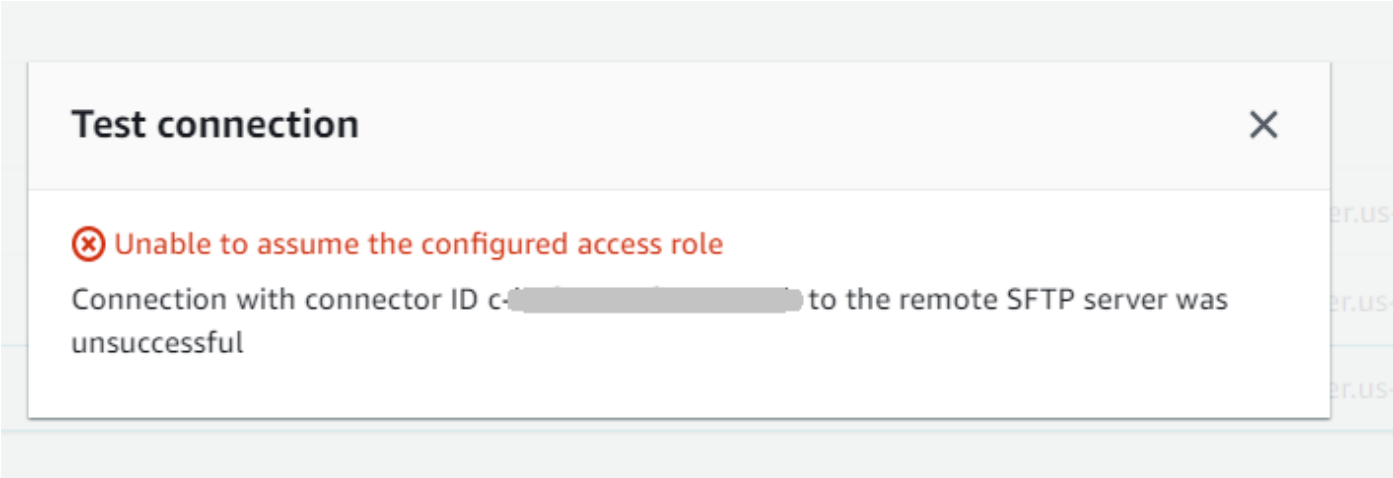
To test an SFTP connector

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **SFTP Connectors**, and select a connector.
3. From the **Actions** menu, choose **Test connection**.



The system returns a message, indicating whether the test passes or fails. If the test fails, the system provides an error message based on the reason the test failed.





Test a connector using the CLI

To test a connector using the AWS Command Line Interface, run the following command at a command prompt (replace *connector-id* with your actual connector ID):

```
aws transfer test-connection --connector-id c-connector-id
```

If the test is successful, the following lines are returned:

```
{
   "Status": "OK",
   "StatusMessage": "Connection succeeded"
}
```

If the test is unsuccessful, you receive a descriptive error message, for example:

```
{
```

```
    "Status": "ERROR",
    "StatusMessage": "Unable to assume the configured access role"
}
```

When you describe a VPC egress type connector, the response includes the new fields:

```
{
   "Connector": {
      "AccessRole": "arn:aws:iam::219573224423:role/sftp-connector-role",
      "Arn": "arn:aws:transfer:us-east-1:219573224423:connector/c-5dfa309ccabf40759",
      "ConnectorId": "c-5dfa309ccabf40759",
      "Status": "ACTIVE",
      "EgressConfig": {
        "ResourceConfigurationArn": "arn:aws:vpc-lattice:us-
east-1:025066256552:resourceconfiguration/rcfg-079259b27a357a190"
      },
      "EgressType": "VPC",
      "ServiceManagedEgressIpAddresses": null,
      "SftpConfig": {
         "TrustedHostKeys": [ "ssh-rsa AAAAB3NzaC..." ],
         "UserSecretId": "aws/transfer/sftp-connector1"
      },
      "Url": "sftp://my.sftp.server.com:22"
   }
}
```

Note that `ServiceManagedEgressIpAddresses` is null for VPC egress type connectors since traffic routes through your VPC instead of AWS managed infrastructure.

Step 3: Send and retrieve files using the SFTP connector

For simplicity, we assume that you already have files in your Amazon S3 bucket.

> **Note**
>
> The tutorial is using Amazon S3 buckets for both source and destination storage locations. If your SFTP server doesn't use Amazon S3 storage, then wherever you see `sftp-server-storage-east` in the following commands, you can replace the path with a path to file locations accessible from your SFTP server.

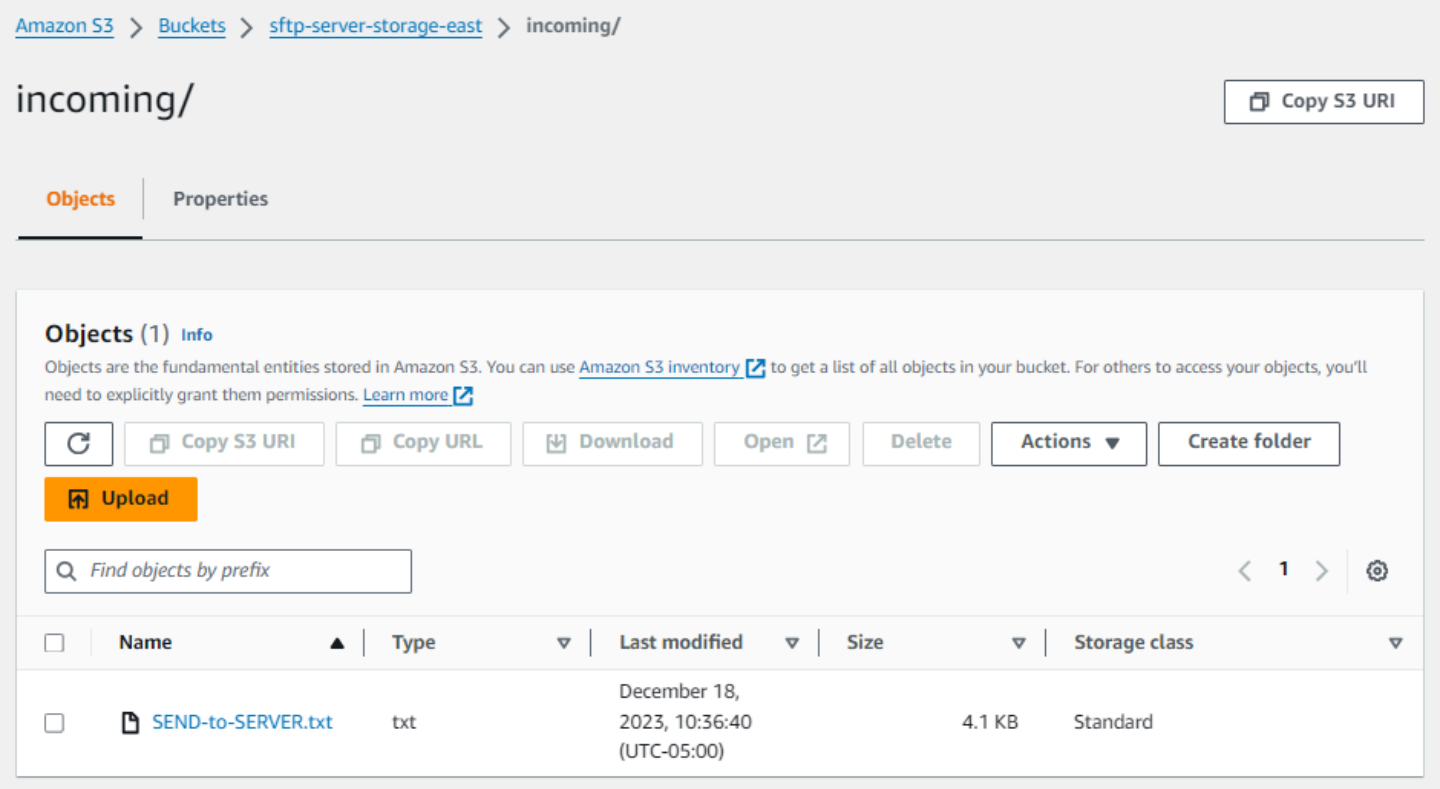
- We send a file named `SEND-to-SERVER.txt` from Amazon S3 storage to the SFTP server.
- We retrieve a file named `RETRIEVE-to-S3.txt` from the SFTP server to Amazon S3 storage.

> **Note**
>
> In the following commands, replace *`connector-id`* with your connector ID.

First, we send a file from our Amazon S3 bucket to the remote SFTP server. From a command prompt, run the following command:

```
aws transfer start-file-transfer --connector-id c-connector-id --send-file-paths "/
sftp-server-storage-east/SEND-to-SERVER.txt" /
   --remote-directory-path "/sftp-server-storage-east/incoming"
```

Your `sftp-server-storage-east` bucket should now look like this.



If you don't see the file as expected, check your CloudWatch logs.

To check your CloudWatch logs

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/
2. Select **Log groups** from the left navigation menu.
3. Enter your connector ID in the search bar to find your logs.
4. Select the Log stream that is returned from the search.
5. Expand the most recent log entry.

If successful, the log entry looks like the following:

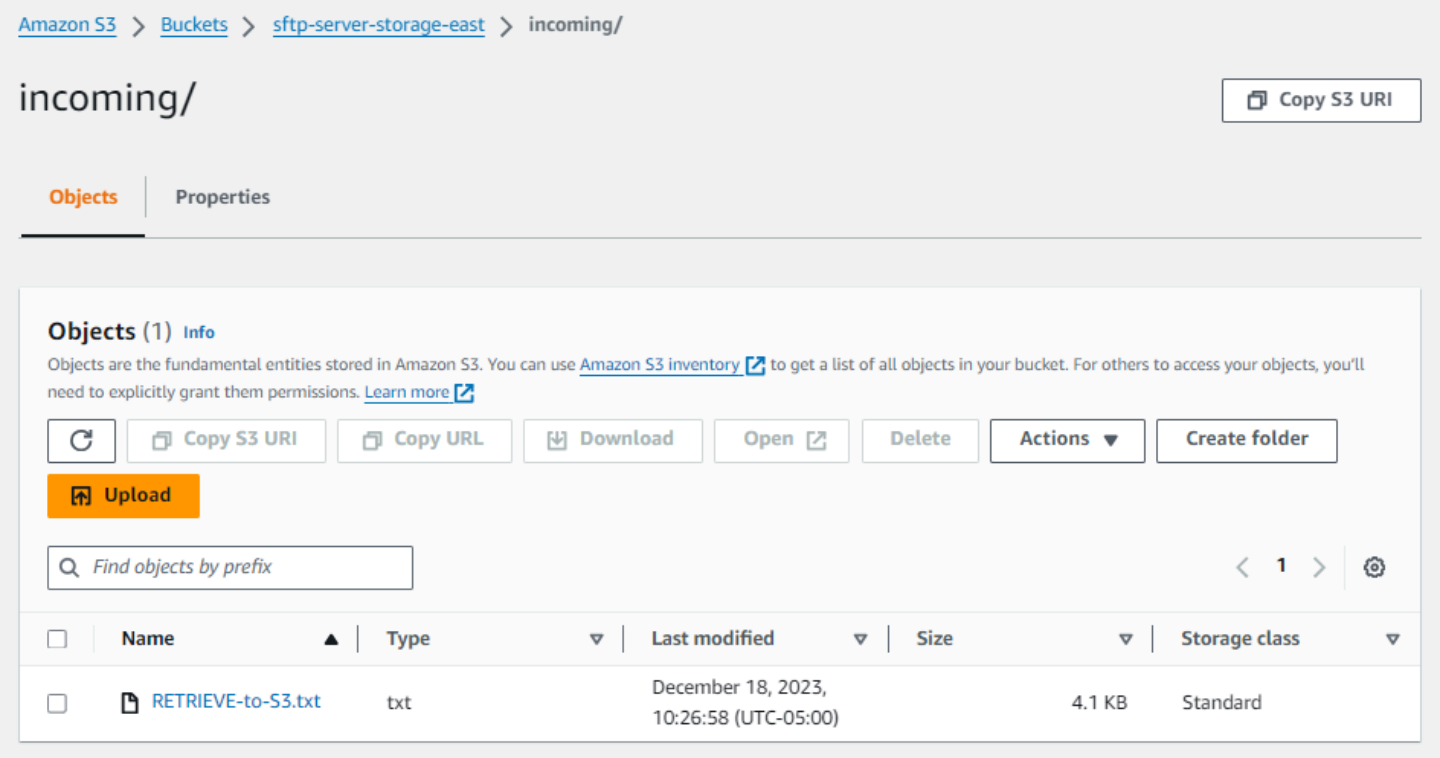
```
{
    "operation": "SEND",
    "timestamp": "2023-12-18T15:26:57.346283Z",
    "connector-id": "connector-id",
    "transfer-id": "transfer-id",
    "file-transfer-id": "transfer-id/file-transfer-id",
    "url": "sftp://server-id.server.transfer.us-east-1.amazonaws.com",
    "file-path": "/sftp-server-storage-east/SEND-to-SERVER.txt",
    "status-code": "COMPLETED",
    "start-time": "2023-12-18T15:26:56.915864Z",
    "end-time": "2023-12-18T15:26:57.298122Z",
    "account-id": "account-id",
    "connector-arn": "arn:aws:transfer:us-east-1:account-id:connector/connector-id",
    "remote-directory-path": "/sftp-server-storage-east/incoming"
}
```

If the file transfer failed, the log entry contains an error message that specifies the issue. Common causes for errors are problems with the IAM permissions and incorrect file paths.

Next, we retrieve a file from the SFTP server into an Amazon S3 bucket. From a command prompt, run the following command:

```
aws transfer start-file-transfer --connector-id c-connector-id --retrieve-file-paths
 "/sftp-server-storage-east/RETRIEVE-to-S3.txt" --local-directory-path "/sftp-server-
storage-east/incoming"
```

If the transfer succeeds, your Amazon S3 bucket contains the transferred file, as shown here.



If successful, the log entry looks like the following:

```
{
    "operation": "RETRIEVE",
    "timestamp": "2023-12-18T15:36:40.017800Z",
    "connector-id": "c-connector-id",
    "transfer-id": "transfer-id",
    "file-transfer-id": "transfer-id/file-transfer-id",
    "url": "sftp://s-server-id.server.transfer.us-east-1.amazonaws.com",
    "file-path": "/sftp-server-storage-east/RETRIEVE-to-S3.txt",
    "status-code": "COMPLETED",
    "start-time": "2023-12-18T15:36:39.727626Z",
    "end-time": "2023-12-18T15:36:39.895726Z",
    "account-id": "account-id",
    "connector-arn": "arn:aws:transfer:us-east-1:account-id:connector/c-connector-id",
    "local-directory-path": "/sftp-server-storage-east/incoming"
}
```

Troubleshooting VPC egress type connectors

If you're experiencing issues with VPC egress type connectors, check the following:

- **Connector status is PENDING**: DNS resolution for VPC connectors can take several minutes. Wait for the status to become ACTIVE before attempting connections.
- **Connection timeouts**: Verify that security groups allow traffic on port 22 between your Resource Gateway subnets and the target SFTP server.
- **Resource Configuration errors**: Ensure your Resource Configuration points to the correct IP address or DNS name, and that the Resource Gateway is in the same VPC as your SFTP server (for private endpoints). For more information, see Resource configurations in the *Amazon VPC Lattice User Guide*.
- **Public endpoint issues**: For public endpoints, ensure you're using a DNS name, not an IP address, in your Resource Configuration. Verify that your VPC has a NAT Gateway for outbound internet access.
- **AZ availability**: Resource Gateways require subnets in at least 2 Availability Zones. Not all AZs support VPC Lattice - check the supported AZs in your region.

Cost considerations for VPC egress type:

- VPC Lattice charges $0.006/GB for data processing as the resource provider (billed directly by VPC Lattice)
- AWS Transfer Family absorbs the $0.01/GB resource consumer costs (first 1 PB)
- For public endpoints via VPC, additional NAT Gateway and data transfer charges may apply
- No additional Transfer Family charges beyond the standard $0.40/GB data processing fee

Procedures to create a Transfer Family server to use as your remote SFTP server

Following, we outline the steps to create a Transfer Family server that serves as your remote SFTP server for this tutorial. Note the following:

- We use a Transfer Family server to represent a remote SFTP server. Typical SFTP connector users have their own remote SFTP server. See Create a Transfer Family SFTP server and a user.
- Because we're using a Transfer Family server, we're also using a service-managed SFTP user. And, for simplicity, we combined the permissions that this user needs to access the Transfer Family server with permissions they need to use our connector. Again, most SFTP connector use cases have a separate SFTP user that is not associated with a Transfer Family server. See Create a Transfer Family SFTP server and a user.

- For the tutorial, because we are using Amazon S3 storage for our remote SFTP server, we need to create a second bucket, **`sftp-server-storage-east`**, so that we can transfer files from one bucket to another.

Create a Transfer Family SFTP server and a user

Most users won't need to create a Transfer Family SFTP server and a user, as you already have an SFTP server with users, and you can use this server to transfer files to and from. However, for this tutorial, for simplicity, we are using a Transfer Family server to function as the remote SFTP server.

Follow the procedure described in Create an SFTP-enabled server to create a server, and Step 3: Add a service managed user to add a user. These are the user details that we are using for the tutorial:

- Create your service-managed user, `sftp-testuser`.
  - Set the home directory to `/sftp-server-storage-east/sftp-testuser`
  - When you create the user, you store a public key. Later, when you create the secret in Secrets Manager, you need to provide the corresponding private key.
- Role: `sftp-connector-role`. For the tutorial, we are using the same IAM role for both our SFTP user and for accessing the SFTP connector. When you create connectors for your organization, you might have separate user and access roles.
- Server host key: You need to use the server host key when you create the connector. You can retrieve this key by running `ssh-keyscan` for your server. For example, if your server ID is `s-1111aaaa2222bbbb3`, and its endpoint is in `us-east-1`, the following command retrieves the server host key:

  ```
  ssh-keyscan s-1111aaaa2222bbbb3.server.transfer.us-east-1.amazonaws.com
  ```

  Copy this text somewhere, as you need to paste it in the Step 2: Create and test an SFTP connector procedure.

Combined user and access role

For the tutorial, we are using a single, combined role. We use this role both for our SFTP user, as well as for access to the connector. The following example contains the details for this role, in case you want to perform the tasks in the tutorial.

The following example grants the necessary permissions to access our two buckets in Amazon S3, and the secret named `aws/transfer/sftp-connector1` stored in Secrets Manager. For the tutorial, this role is named `sftp-connector-role`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListingOfUserFolder",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::sftp-server-storage-east",
                "arn:aws:s3:::sftp-server-storage-east"
            ]
        },
        {
            "Sid": "HomeDirObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion",
                "s3:GetObjectVersion",
                "s3:GetObjectACL",
                "s3:PutObjectACL"
            ],
            "Resource": [
                "arn:aws:s3:::sftp-server-storage-east/*",
                "arn:aws:s3:::sftp-server-storage-east/*"
            ]
        },
        {
            "Sid": "GetConnectorSecretValue",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
```

```
            "Resource": "arn:aws:secretsmanager:us-east-1:111122223333:secret:aws/
transfer/sftp-connector1-6RandomCharacters"
        }
    ]
}
```

For complete details about creating roles for Transfer Family, follow the procedure described in Create a user role to create a role.

Setting up an Amazon API Gateway method as a custom identity provider

This tutorial illustrates how to set up an Amazon API Gateway method and use it as a custom identity provider to upload files to an AWS Transfer Family server. This tutorial uses the Basic stack template, and other basic functionality as an example only.

Topics

- Prerequisites
- Step 1: Create a CloudFormation stack
- Step 2: Check the API Gateway method configuration for your server
- Step 3: View the Transfer Family server details
- Step 4: Test that your user can connect to the server
- Step 5: Test the SFTP connection and file transfer
- Step 6: Limit access to the bucket
- Update Lambda if using Amazon EFS

Prerequisites

Before you create the Transfer Family resources in CloudFormation, create your storage and your user role.

To specify storage and create a user role

1. Depending on which storage you are using, see the following documentation:

- To create an Amazon S3 bucket, see How do I create an S3 bucket? in the *Amazon Simple Storage Service User Guide*.
- To create an Amazon EFS file system, see Configure an Amazon EFS file system.

2. To create a user role, see Create an IAM role and policy

You enter the details for your storage and your user role when you create your CloudFormation stack in the next section.

Step 1: Create a CloudFormation stack

To create an AWS CloudFormation stack from the provided template

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Select **Create stack**, and choose **With new resources (standard)**.
3. In the **Prerequisite - Prepare template** pane, choose **Choose an existing template**.
4. Copy this link, Basic stack template, and paste it into the **Amazon S3 URL** field.
5. Click **Next**.
6. Specify parameters, including a name for your stack. Be sure to do the following:
   - Replace the default values for **UserName** and **UserPassword**.
   - For **UserHomeDirectory**, enter the details for the storage (either an Amazon S3 bucket or an Amazon EFS filesystem) that you created earlier.
   - Replace the default **UserRoleArn** with the user role that you created earlier. The AWS Identity and Access Management (IAM) role must have the appropriate permissions. For an example IAM role and bucket policy, see Step 6: Limit access to the bucket.
   - If you want to authenticate using a public key instead of a password, enter your public key in the **UserPublicKey1** field. The first time that you connect to the server using SFTP, you then provide the private key instead of a password.
7. Choose **Next**, and then choose **Next** again on the **Configure stack options** page.
8. Review the details for the stack that you are creating, and then choose **Create stack**.

> **Note**
>
> At the bottom of the page, under **Capabilities**, you must acknowledge that CloudFormation might create IAM resources.

Step 2: Check the API Gateway method configuration for your server

> **Note**
>
> To improve security, you can configure a web application firewall. AWS WAF is a web application firewall that lets you monitor the HTTP and HTTPS requests that are forwarded to an Amazon API Gateway. For details, see Add a web application firewall.

> **Do not enable API Gateway caching**
>
> Do not enable caching for your API Gateway method when using it as a custom identity provider for Transfer Family. Caching is inappropriate and invalid for authentication requests because:
>
> - Each authentication request is unique and requires a live response, not a cached response
> - Caching provides no benefits since Transfer Family never sends duplicate or repeated requests to the API Gateway
> - Enabling caching will cause the API Gateway to respond with mismatched data, resulting in invalid responses to authentication requests

To check the API Gateway method configuration for your server and deploy it

1. Open the API Gateway console at https://console.aws.amazon.com/apigateway/.
2. Choose the **Transfer Custom Identity Provider basic template API** that the CloudFormation template generated.
3. In the **Resources** pane, choose **GET**, and then choose **Method Request**.

4. For **Actions**, choose **Deploy API**. For **Deployment stage**, choose **prod**, and then choose **Deploy**.

   After the API Gateway method is successfully deployed, view its performance in the **Stage Editor** section.

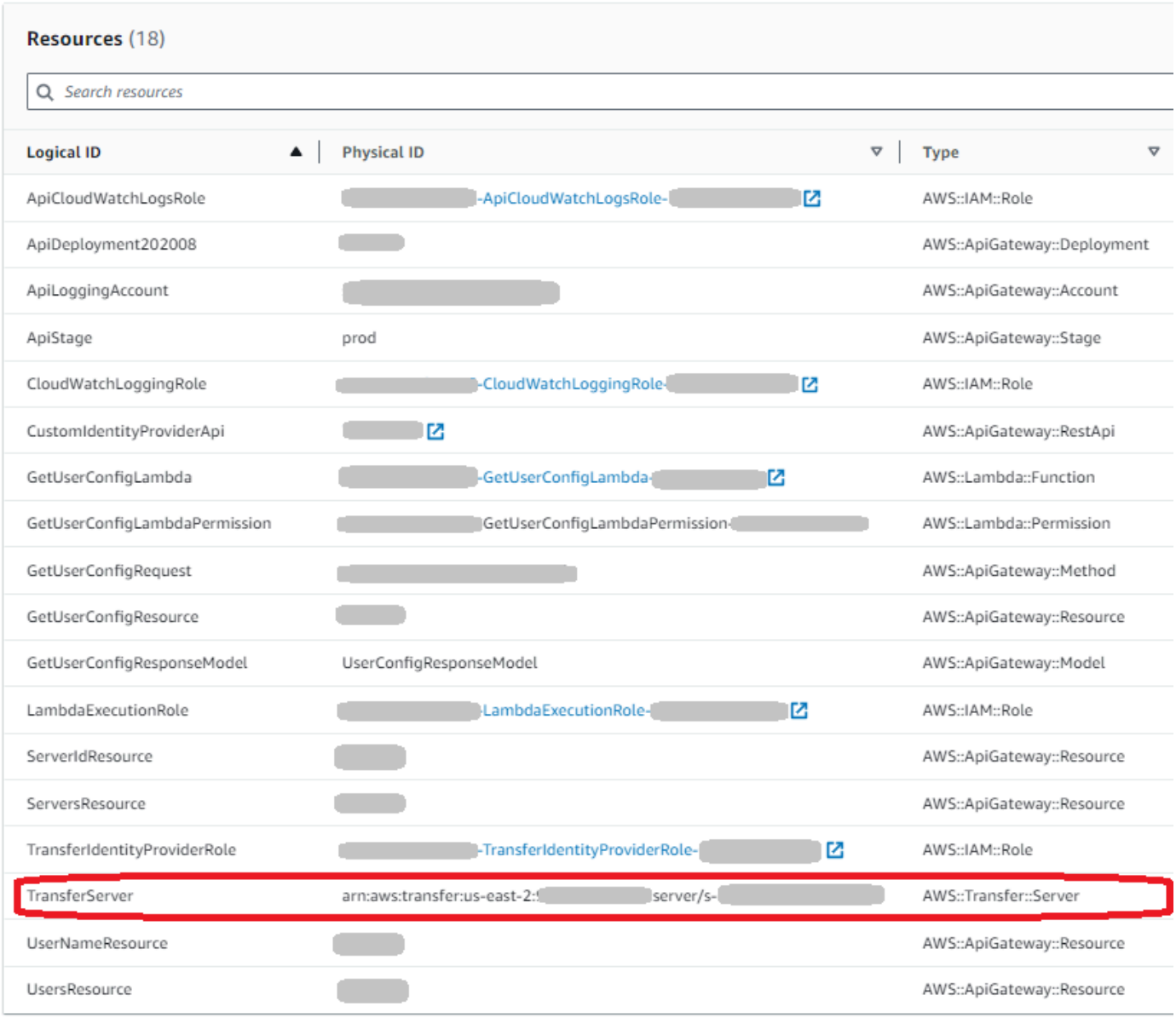
   > **Note**
   >
   > Copy the **Invoke URL** address that appears at the top of the page. You will need it for the next step.

Step 3: View the Transfer Family server details

When you use the template to create an CloudFormation stack, a Transfer Family server is automatically created.

To view your Transfer Family server details

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Choose the stack that you created.
3. Choose the **Resources** tab.



The server ARN is shown in the **Physical ID** column for the **TransferServer** row. The server ID is contained in the ARN, for example **s-11112222333344445**.

4. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/, and on the **Servers** page, choose the new server.

   The server ID matches the ID displayed for the **TransferServer** resource in CloudFormation.

Step 4: Test that your user can connect to the server

To test that your user can connect to the server, using the Transfer Family console

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. On the **Servers** page, choose your new server, choose **Actions**, and then choose **Test**.
3. Enter the text for your sign-in credentials into the **Username** field, and into the **Password** field. These are the values that you set when you deployed the CloudFormation stack.
4. For **Server Protocol**, select **SFTP**, and for **Source IP**, enter **`127.0.0.1`**.
5. Choose **Test**.

   If user authentication succeeds, the test returns a `StatusCode: 200` HTML response and a JSON object containing the details of the user's roles and permissions. For example:

   ```
   {
       "Response": "{\"Role\": \"arn:aws:iam::123456789012:role/my-user-role\",
   \"HomeDirectory\": \"/${transfer:HomeBucket}/\"}",
       "StatusCode": 200,
       "Message": "",
       "Url": "https://1a2b3c4d5e.execute-api.us-east-2.amazonaws.com/prod/servers/
   s-1234abcd5678efgh0/users/myuser/config"
   }
   ```

   If the test fails, add one of the API Gateway AWS managed policies to the role that you are using for your API.

Step 5: Test the SFTP connection and file transfer

To test the SFTP connection

1. On a Linux or macOS device, open a command terminal.
2. Enter one of the following commands, depending on whether you are using a password or a key pair for authentication.

   - If you are using a password, enter this command:

     ```
     sftp -o PubkeyAuthentication=no myuser@server-
     ID.server.transfer.region-code.amazonaws.com
     ```

When prompted, enter your password.

- If you are using a key pair, enter this command:

  ```
  sftp -i private-key-file myuser@server-ID.server.transfer.region-code.amazonaws.com
  ```

> **Note**
>
> For these `sftp` commands, insert the code for the AWS Region where your Transfer Family server is located. For example, if your server is in US East (Ohio), enter **`us-east-2`**.

3. At the `sftp>` prompt, make sure that you can upload (`put`), download (`get`), and view directories and files (`pwd` and `ls`).

Step 6: Limit access to the bucket

You can limit who can access a specific Amazon S3 bucket. The following example shows the settings to use in your CloudFormation stack and in the policy that you select for your user.

In this example, we set the following parameters for the CloudFormation stack:

- **CreateServer**: `true`
- **UserHomeDirectory**: `/amzn-s3-demo-bucket1`
- **UserName**: `myuser`
- **UserPassword**: `MySuperSecretPassword`

> **Important**
>
> This is an example password. When you configure your API Gateway method, make sure that you enter a strong password.

- **UserPublicKey1**: *your-public-key*
- **UserRoleArn**: `arn:aws:iam::`*role-id*`:role/myuser-api-gateway-role`

The **UserPublicKey1** is a public key that you have generated as part of a public/private key pair.

The *role-id* is unique to the user role that you create. The policy attached to the `myuser-api-gateway-role` is the following:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObjectAcl",
                "s3:GetObject",
                "s3:DeleteObjectVersion",
                "s3:DeleteObject",
                "s3:PutObjectAcl",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        }
    ]
}
```

To connect to the server using SFTP, enter one of the following commands at the prompt.

- If you are using a password to authenticate, run the following command:

  `sftp -o PubkeyAuthentication=no` *myuser*`@`*transfer-server-ID*`.server.transfer.`*region-id*`.amazonaws.com`

  When prompted, enter your password.
- If you are using a key pair to authenticate, run the following command:

```
sftp -i private-key-file myuser@transfer-server-
ID.server.transfer.region-id.amazonaws.com
```

> **Note**
>
> For these `sftp` commands, use the ID for the AWS Region where your Transfer Family server is located. For example, if your server is in US East (Ohio), use `us-east-2`.

At the `sftp` prompt, you are directed to your home directory, which you can view by running the `pwd` command. For example:

```
sftp> pwd
Remote working directory: /amzn-s3-demo-bucket1
```

The user cannot view any directories above the home directory. For example:

```
sftp> pwd
Remote working directory: /amzn-s3-demo-bucket1
sftp> cd ..
sftp> ls
Couldn't read directory: Permission denied
```

Update Lambda if using Amazon EFS

If you selected Amazon EFS as the storage option for your Transfer Family server, you need to edit the lambda function for your stack.

To add a Posix profile to your Lambda function

1. Open the Lambda console at https://console.aws.amazon.com/lambda/.
2. Select the Lambda function that you created earlier. The Lambda function has the format of ***stack-name*-GetUserConfigLambda-*lambda-identifier***, where *stack-name* is the CloudFormation stack name and *lambda-identifier* is the identifier for the function.
3. In the **Code** tab, select **index.js** to display the code for the function.
4. In the `response`, add the following line between `Policy` and `HomeDirectory`:

```
PosixProfile: {"Uid": uid-value, "Gid": gid-value},
```

Where the *uid-value* and *gid-value* are integers, 0 or greater, that represent the User ID and Group ID respectively.

For example, after you add the Posix profile, the response field might look like the following:

```
response = {
     Role: 'arn:aws:iam::123456789012:role/api-gateway-transfer-efs-role', // The
user will be authenticated if and only if the Role field is not blank
     Policy: '', // Optional JSON blob to further restrict this user's permissions
     PosixProfile: {"Gid": 65534, "Uid": 65534},
     HomeDirectory: '/fs-fab2c234' // Not required, defaults to '/'
   };
```

Setting up an AS2 configuration

This tutorial walks through how to set up an Applicability Statement 2 (AS2) configuration with AWS Transfer Family. After you complete the steps described here, you will have an AS2-enabled server that's ready for accepting AS2 messages from a sample trading partner. You will also have a connector that can be used to send AS2 messages to the sample trading partner.

Note

Some portions of the example setup use the AWS Command Line Interface (AWS CLI). If you haven't already installed the AWS CLI, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

1. Create certificates for yourself and your trading partner. If you have existing certificates that you can use, you can skip this section.

   This process is described in Step 1: Create certificates for AS2.
2. Import the certificates that you created in step 1.

   This process is described in Step 2: Import certificates as Transfer Family certificate resources.
3. To set up your trading partners, create a local profile and a partner profile.

This process is described in Step 3: Create profiles for you and your trading partner.

4. Create an AWS Transfer Family server that uses the AS2 protocol. Optionally, you can add an Elastic IP address to the server to make it internet-facing.

   This process is described in Step 4: Create a Transfer Family server that uses the AS2 protocol.

   > **Note**
   >
   > You must create a Transfer Family server for inbound transfers only. If you're performing only outbound transfers, you don't need a Transfer Family server.

5. Create an agreement between you and your trading partner.

   This process is described in Step 5: Create an agreement between you and your partner.

   > **Note**
   >
   > You must create an agreement for inbound transfers only. If you're performing only outbound transfers, you don't need an agreement.

6. Create a connector between you and your trading partner.

   This process is described in Step 6: Create a connector between you and your partner.

   > **Note**
   >
   > You must create a connector for outbound transfers only. If you're performing only inbound transfers, you don't need a connector.

7. Test an AS2 file exchange.

   This process is described in Step 7: Test exchanging files over AS2 by using Transfer Family.

After you complete these steps, you can do the following:

- Send files to a remote AS2-enabled partner server with the Transfer Family `start-file-transfer` AWS Command Line Interface (AWS CLI) command.

- Receive files from a remote AS2-enabled partner server on port 5080 through your virtual private cloud (VPC) endpoint.

Step 1: Create certificates for AS2

Both parties in an AS2 exchange need X.509 certificates. You can create these certificates in any way that you like. This topic describes how to use OpenSSL from the command line to create a root certificate, and then sign subordinate certificates. Both parties must generate their own certificates.

> **Note**
>
> The key length for AS2 certificates must be at least 2048 bits, and at most 4096.

To transfer files with a partner, take note of the following:

- You can attach certificates to profiles. The certificates contain public or private keys.
- Your trading partner sends you their public keys, and you send them yours.
- Your trading partner encrypts messages with your public key and signs them with their private key. Conversely, you encrypt messages with your partner's public key and sign them with your private key.

> **Note**
>
> If you prefer to manage keys with a GUI, Portecle is one option that you can use.

To generate example certificates

> **Important**
>
> **Do not send your partner your private keys.** In this example, you generate a set of self-signed public and private keys for one party. If you are going to act as both trading partners for testing purposes, you can repeat these instructions to generate two sets of

keys: one for each trading partner. In this case, you do not need to generate two root certificate authorities (CAs).

1. Run the following command to generate an RSA private key with a 2048-bit-long modulus.

```
/usr/bin/openssl genrsa -out root-ca-key.pem 2048
```

2. Run the following command to create a self-signed certificate with your `root-ca-key.pem` file.

```
/usr/bin/openssl req \
-x509 -new -nodes -sha256 \
-days 1825 \
-subj "/C=US/ST=MA/L=Boston/O=TransferFamilyCustomer/OU=IT-dept/CN=ROOTCA" \
-key root-ca-key.pem \
-out root-ca.pem
```

The `-subj` argument consists of the following values.

	Name	Description
C	Country code	A two-letter code for the country in which your organization is located.
ST	State, region, or province	The state, region, or province in which your organization is located. (In this case, *region* does not refer to your AWS Region.)
L	Locality name	The city in which your organization is located.
O	Organization name	The full legal name of your organization, including

	Name	Description
		suffixes, such as LLC, Corp, and so on.
OU	Organizational unit name	The division in your organization that deals with this certificate.
CN	Common name or fully qualified domain name (FQDN)	In this case, we're creating a root certificate, so the value is ROOTCA. In these examples, we are using CN to describe the purpose of the certificate.

3. Create a signing key and an encryption key for your local profile.

```
/usr/bin/openssl genrsa -out signing-key.pem 2048
/usr/bin/openssl genrsa -out encryption-key.pem 2048
```

> **Note**
>
> Some AS2-enabled servers, such as OpenAS2, require that you use the same certificate for both signing and encryption. In this case, you can import the same private key and certificate for both purposes. To do so, run this command instead of the two previous commands:
>
> ```
> /usr/bin/openssl genrsa -out signing-and-encryption-key.pem 2048
> ```

4. Run the following commands to create Certificate Signing Requests (CSRs) for the root key to sign.

```
/usr/bin/openssl req -new -key signing-key.pem -subj \
"/C=US/ST=MA/L=Boston/O=TransferFamilyCustomer/OU=IT-dept/CN=Signer" -out signing-
key-csr.pem
```

```
/usr/bin/openssl req -new -key encryption-key.pem -subj \
```

```
"/C=US/ST=MA/L=Boston/O=TransferFamilyCustomer/OU=IT-dept/CN=Encrypter" -out
 encryption-key-csr.pem
```

5. Next, you must create a `signing-cert.conf` file and an `encryption-cert.conf` file.

   - Use a text editor to create the `signing-cert.conf` file with the following contents:

     ```
     authorityKeyIdentifier=keyid,issuer
     keyUsage = digitalSignature, nonRepudiation
     ```

   - Use a text editor to create the `encryption-cert.conf` file with the following contents:

     ```
     authorityKeyIdentifier=keyid,issuer
     keyUsage = dataEncipherment
     ```

6. Finally, you create the signed certificates by running the following commands.

   ```
   /usr/bin/openssl x509 -req -sha256 -CAcreateserial -days 1825 -in signing-key-
   csr.pem -out signing-cert.pem -CA \
   root-ca.pem -CAkey root-ca-key.pem -extfile signing-cert.conf
   ```

   ```
   /usr/bin/openssl x509 -req -sha256 -CAcreateserial -days 1825 -in encryption-key-
   csr.pem -out encryption-cert.pem \
   -CA root-ca.pem -CAkey root-ca-key.pem -extfile encryption-cert.conf
   ```

Step 2: Import certificates as Transfer Family certificate resources

This procedure explains how to import certificates by using the AWS CLI. If you want to use the Transfer Family console instead, see the section called “Import AS2 certificates”.

To import the signing and encryption certificates that you created in step 1, run the following `import-certificate` commands. If you're using the same certificate for encryption and signing, import the same certificate twice (once with the `SIGNING` usage and again with the `ENCRYPTION` usage).

> **Note**
>
> If you have a file that contains both a certificate and its chain, you can provide that file to the `import-certificate` command using only the `certificate` parameter. For example:

```
aws transfer import-certificate --usage ENCRYPTION --certificate
 file://combined-cert-and-chain-file.pem
```

If you use the `certificate` parameter to upload both the certificate and its chain, don't use the `certificate-chain` parameter.
If you combine the certificate and its chain, your key is formatted using conventional PEM standards, which includes a newline "\n" every 64 characters. The certificate that is stored is functionally equivalent to the one you uploaded, with the sole difference that the `DescribeCertificate` response via the AWS CLI will contain these newline characters.

```
aws transfer import-certificate --usage SIGNING --certificate file://signing-cert.pem \
            --private-key file://signing-key.pem --certificate-chain file://root-ca.pem
```

This command returns your signing `CertificateId`. In the next section, this certificate ID is referred to as *my-signing-cert-id*.

```
aws transfer import-certificate --usage ENCRYPTION --certificate file://encryption-
cert.pem \
            --private-key file://encryption-key.pem --certificate-chain file://root-
ca.pem
```

This command returns your encryption `CertificateId`. In the next section, this certificate ID is referred to as *my-encrypt-cert-id*.

Next, import your partner's encryption and signing certificates by running the following commands.

```
aws transfer import-certificate --usage ENCRYPTION --certificate file://partner-
encryption-cert.pem \
--certificate-chain file://partner-root-ca.pem
```

This command returns your partner's encryption `CertificateId`. In the next section, this certificate ID is referred to as *partner-encrypt-cert-id*.

```
aws transfer import-certificate --usage SIGNING --certificate file://partner-signing-
cert.pem \
```

```
--certificate-chain file://partner-root-ca.pem
```

This command returns your partner's signing `CertificateId`. In the next section, this certificate ID is referred to as *partner-signing-cert-id*.

Step 3: Create profiles for you and your trading partner

This procedure explains how to create AS2 profiles by using AWS CLI. If you want to use the Transfer Family console instead, see the section called "Create AS2 profiles".

Create your local AS2 profile by running the following command. This command references the certificates that contain your public and private keys.

```
aws transfer create-profile --as2-id MYCORP --profile-type LOCAL --certificate-ids \
my-signing-cert-id my-encrypt-cert-id
```

This command returns your profile ID. In the next section, this ID is referred to as *my-profile-id*.

Now create the partner profile by running the following command. This command uses only your partner's public key certificates. To use this command, replace the *user input placeholders* with your own information; for example, your partner's AS2 name and certificate IDs.

```
aws transfer create-profile --as2-id PARTNER-COMPANY --profile-type PARTNER --
certificate-ids \
partner-signing-cert-id partner-encrypt-cert-id
```

This command returns your partner's profile ID. In the next section, this ID is referred to as *partner-profile-id*.

> **Note**
>
> In the previous commands, replace *MYCORP* with the name of your organization, and *PARTNER-COMPANY* with the name of your trading partner's organization.

Step 4: Create a Transfer Family server that uses the AS2 protocol

This procedure explains how to create an AS2-enabled server by using the Transfer Family AWS CLI.

> **Note**
>
> Many of the example steps use commands that load parameters from a file. For more details about using files to load parameters, see How to load parameters from a file.

If you want to use the console instead, see Create an AS2 server using the Transfer Family console.

Similar to how you create an SFTP or FTPS AWS Transfer Family server, you create an AS2-enabled server by using the `--protocols AS2` parameter of the `create-server` AWS CLI command. Currently, Transfer Family supports only VPC endpoint types and Amazon S3 storage with the AS2 protocol.

When you create your AS2-enabled server for Transfer Family by using the `create-server` command, a VPC endpoint is automatically created for you. This endpoint exposes TCP port 5080 so that it can accept AS2 messages.

If you want to expose your VPC endpoint publicly to the internet, you can associate Elastic IP addresses with your VPC endpoint.

To use these instructions, you need the following:

- The ID of your VPC (for example, **vpc-abcdef01**).
- The IDs of your VPC subnets (for example, **subnet-abcdef01**, **subnet-subnet-abcdef01**, **subnet-021345ab**).
- One or more IDs of the security groups that allow incoming traffic on TCP port 5080 from your trading partners (for example, **sg-1234567890abcdef0** and **sg-abcdef01234567890**).
- (Optional) The Elastic IP addresses that you want to associate with your VPC endpoint.
- If your trading partner is not connected to your VPC through a VPN, you need an internet gateway. For more information, see Connect to the internet using an internet gateway in the *Amazon VPC User Guide*.

To create an AS2-enabled server

1. Run the following command. Replace each *`user input placeholder`* with your own information.

```
aws transfer create-server --endpoint-type VPC \
```

```
--endpoint-details VpcId=vpc-abcdef01,SubnetIds=subnet-abcdef01,subnet-
abcdef01,subnet-
021345ab,SecurityGroupIds=sg-abcdef01234567890,sg-1234567890abcdef0 --protocols AS2
 \
--protocol-details As2Transports=HTTP
```

2. (Optional) You can make the VPC endpoint public. You can attach Elastic IP addresses to a Transfer Family server only through an `update-server` operation. The following commands stop the server, update it with Elastic IP addresses, and then start it again.

```
aws transfer stop-server --server-id your-server-id
```

```
aws transfer update-server --server-id your-server-id --endpoint-details \
AddressAllocationIds=eipalloc-abcdef01234567890,eipalloc-
1234567890abcdef0,eipalloc-abcd012345cccccc
```

```
aws transfer start-server --server-id your-server-id
```

This `start-server` command automatically creates a DNS record for you that contains the public IP address for your server. To give your trading partner access to the server, you provide them with the following information. In this case, *your-region* refers to your AWS Region.

`s-`*your-server-id*`.server.transfer.`*your-region*`.amazonaws.com`

The full URL that you provide to your trading partner is as follows:

`http://s-`*your-server-id*`.server.transfer.`*your-region*`.amazonaws.com:5080`

3. To test whether your AS2-enabled server is accessible, use the following commands. Make sure that your server can be accessed either through your VPC endpoint's private DNS address, or through your public endpoint (if you associated an Elastic IP address with your endpoint).

   If your server is configured correctly, the connection will succeed. However, you will receive an HTTP status code 400 (Bad Request) response because you aren't sending a valid AS2 message.

   - For a public endpoint (if you associated an Elastic IP address in the previous step), run the following command, substituting your server ID and Region.

     ```
     curl -vv -X POST http://s-your-server-id.transfer.your-region.amazonaws.com:5080
     ```

- If you are connecting within your VPC, look up your VPC endpoint's private DNS name by running the following commands.

  ```
  aws transfer describe-server --server-id s-your-server-id
  ```

  This `describe-server` command returns your VPC endpoint ID in the `VpcEndpointId` parameter. Use this value to run the following command.

  ```
  aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-your-vpc-endpoint-id
  ```

  This `describe-vpc-endpoints` command returns a `DNSEntries` array, with several `DnsName` parameters. Use the Regional DNS name (the one that does not include the Availability Zone) in the following command.

  ```
  curl -vv -X POST http://vpce-your-vpce.vpce-svc-your-vpce-svc.your-
  region.vpce.amazonaws.com:5080
  ```

  For example, the following command shows sample values for the placeholders in the previous command.

  ```
  curl -vv -X POST http://vpce-0123456789abcdefg-fghij123.vpce-
  svc-11111aaaa2222bbbb.us-east-1.vpce.amazonaws.com:5080
  ```

4. (Optional) Configure a logging role. Transfer Family logs the status of messages sent and received in a structured JSON format to Amazon CloudWatch logs. To provide Transfer Family with access to the CloudWatch logs in your account, you must configure a logging role on your server.

   Create an AWS Identity and Access Management (IAM) role that trusts `transfer.amazonaws.com`, and attach the `AWSTransferLoggingAccess` managed policy. For details, see Create an IAM role and policy. Note the Amazon Resource Name (ARN) of the IAM role that you just created, and associate it with the server by running the following `update-server` command:

   ```
   aws transfer update-server --server-id your-server-id --logging-role
    arn:aws:iam::your-account-id:role/logging-role-name
   ```

> **Note**
>
> Even though the logging role is optional, we highly recommend setting it up so that you can see the status of your messages and troubleshoot configuration issues.

Step 5: Create an agreement between you and your partner

This procedure explains how to create AS2 agreements by using the AWS CLI. If you want to use the Transfer Family console instead, see the section called “Create an AS2 agreement”.

Agreements bring together the two profiles (local and partner), their certificates, and a server configuration that allows inbound AS2 transfers between two parties. You can list your items by running the following commands.

```
aws transfer list-profiles --profile-type LOCAL
aws transfer list-profiles --profile-type PARTNER
aws transfer list-servers
```

This step requires an Amazon S3 bucket and IAM role with read/write access to and from the bucket. The instructions for creating this role are the same as for the Transfer Family SFTP, FTP, and FTPS protocols and are available in Create an IAM role and policy.

To create an agreement, you need the following items:

- The Amazon S3 bucket name (and object prefix, if specified) to store your AS2 files. We recommend that you specify separate directories for different file types.
- The ARN of the IAM role with access to the bucket
- Your Transfer Family server ID
- Your profile ID and your partner's profile ID

Save the following code into a file, for example, `agreementDetails.json`. Replace each *`user input placeholder`* with your own information.

```
{
    "Description": "ExampleAgreementName",
```

```
    "ServerId": "your-server-id",
    "LocalProfileId": "your-profile-id",
    "PartnerProfileId": "your-partner-profile-id",
    "AccessRole": "arn:aws:iam::111111111111:role/TransferAS2AccessRole",
    "Status": "ACTIVE",
    "PreserveFilename": "ENABLED",
    "EnforceMessageSigning": "ENABLED",
    "CustomDirectories": {
        "FailedFilesDirectory": "/amzn-s3-demo-destination-bucket/AS2-failed",
        "MdnFilesDirectory": "/amzn-s3-demo-destination-bucket/AS2-mdn",
        "PayloadFilesDirectory": "/amzn-s3-demo-destination-bucket/AS2-payload",
        "StatusFilesDirectory": "/amzn-s3-demo-destination-bucket/AS2-status",
        "TemporaryFilesDirectory": "/amzn-s3-demo-destination-bucket/AS2-temp"
    }
}
```

Note

To use a single base directory instead of separate directories, remove the `CustomDirectories` line and its individual directory lines from the previous code and use the following parameter instead:
`"BaseDirectory": "/amzn-s3-demo-destination-bucket/AS2-inbox"`
Do not use both a base directory and separate directory parameters, or the command will fail.

Then, run the following command.

```
aws transfer create-agreement --cli-input-json file://agreementDetails.json
```

If successful, this command returns the ID for the agreement. You can then view the details of the agreement with the following command.

```
aws transfer describe-agreement --agreement-id agreement-id --server-id your-server-id
```

Step 6: Create a connector between you and your partner

This procedure explains how to create AS2 connectors by using the AWS CLI. If you want to use the Transfer Family console instead, see the section called “Configure AS2 connectors”.

You can use the `StartFileTransfer` API operation to send files that are stored in Amazon S3 to your trading partner's AS2 endpoint by using a connector. You can find the profiles that you created earlier by running the following command.

```
aws transfer list-profiles
```

When you create the connector, you must provide your partner's AS2 server URL. Copy the following text to a file named `testAS2Config.json`.

```
{
"Compression": "ZLIB",
"EncryptionAlgorithm": "AES256_CBC",
"LocalProfileId": "your-profile-id",
"MdnResponse": "SYNC",
"MdnSigningAlgorithm": "DEFAULT",
"MessageSubject": "Your Message Subject",
"PartnerProfileId": "partner-profile-id",
"PreserveContentType": "FALSE",
"SigningAlgorithm": "SHA256"
}
```

> **Note**
>
> For `EncryptionAlgorithm`, do not specify the `DES_EDE3_CBC` algorithm unless you must support a legacy client that requires it, as it is a weak encryption algorithm.

Then run the following command to create the connector.

```
aws transfer create-connector --url "http://partner-as2-server-url" \
--access-role your-IAM-role-for-bucket-access \
--logging-role arn:aws:iam::your-account-id:role/service-role/AWSTransferLoggingAccess
 \
--as2-config file:///path/to/testAS2Config.json
```

Step 7: Test exchanging files over AS2 by using Transfer Family

Receive a file from your trading partner

If you associated a public Elastic IP address with your VPC endpoint, Transfer Family automatically created a DNS name that contains your public IP address. The subdomain is your AWS Transfer Family server ID (of the format `s-1234567890abcdef0`). Provide your server URL to your trading partner in the following format.

```
http://s-1234567890abcdef0.server.transfer.us-east-1.amazonaws.com:5080
```

If you didn't associate a public Elastic IP address with your VPC endpoint, look up the hostname of the VPC endpoint that can accept AS2 messages over HTTP POST from your trading partners on port 5080. To retrieve the VPC endpoint details, use the following command.

```
aws transfer describe-server --server-id s-1234567890abcdef0
```

For example, assume the preceding command returns a VPC endpoint ID of `vpce-1234abcd5678efghi`. Then, you would use the following command to retrieve the DNS names.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-1234abcd5678efghi
```

This command returns all the details for the VPC endpoint that you need to run the following command.

The DNS name is listed in the `DnsEntries` array. Your trading partner must be within your VPC to access your VPC endpoint (for example through AWS PrivateLink or a VPN). Provide your VPC endpoint URL to your partner in the following format.

```
http://vpce-your-vpce-id.vpce-svc-your-vpce-svc-id.your-region.vpce.amazonaws.com:5080
```

For example, the following URL shows sample values for the placeholders in the previous commands.

```
http://vpce-0123456789abcdefg-fghij123.vpce-svc-11111aaaa2222bbbb.us-
east-1.vpce.amazonaws.com:5080
```

In this example, successful transfers are stored at the location that's specified in the `base-directory` parameter that you specified in Step 5: Create an agreement between you and your partner. If we successfully receive files named `myfile1.txt` and `myfile2.txt`, the files are stored as */path-defined-in-the-agreement/processed/original_filename.messageId.original_extension*. Here, the files are stored as `/amzn-s3-demo-destination-bucket/AS2-inbox/processed/myfile1.`*`messageId`*`.txt` and `/amzn-s3-demo-destination-bucket/AS2-inbox/processed/myfile2.`*`messageId`*`.txt`.

If you configured a logging role when you created your Transfer Family server, you can also check your CloudWatch logs for the status of AS2 messages.

Send a file to your trading partner

You can use Transfer Family to send AS2 messages by referencing the connector ID and the paths to the files, as illustrated in the following `start-file-transfer` AWS Command Line Interface (AWS CLI) command:

```
aws transfer start-file-transfer --connector-id c-1234567890abcdef0 \
--send-file-paths "/amzn-s3-demo-source-bucket/myfile1.txt" "/amzn-s3-demo-source-bucket/myfile2.txt"
```

To get the details for your connectors, run the following command:

```
aws transfer list-connectors
```

The `list-connectors` command returns the connector IDs, URLs, and Amazon Resource Names (ARNs) for your connectors.

To return the properties of a particular connector, run the following command with the ID that you want to use:

```
aws transfer describe-connector --connector-id your-connector-id
```

The `describe-connector` command returns all of the properties for the connector, including its URL, roles, profiles, Message Disposition Notices (MDNs), tags, and monitoring metrics.

You can confirm that the partner successfully received the files by viewing the JSON and MDN files. These files are named according to the conventions described in File names and locations. If you

configured a logging role when you created the connector, you can also check your CloudWatch logs for the status of AS2 messages.

Tutorial: Setting up a basic Transfer Family web app

This tutorial walks through how to set up a Transfer Family web app. Transfer Family web apps enable a simple interface for transferring data to and from Amazon S3 over a web browser. For detailed documentation for this feature, see Transfer Family web apps.

Web app tutorial: prerequisites

- Create either an account instance or organization instance of AWS IAM Identity Center. For details, see Configure your identity provider for Transfer Family web apps.

  If you are not using IAM Identity Center as your identity provider, Integrate Okta as your identity provider for web apps illustrates how to use an alternative (in this case Okta) identity provider.
- You need an Amazon S3 bucket to use for interacting with your Transfer Family web app. For details, see Configure an Amazon S3 bucket

> **Note**
>
> This tutorial assumes that you are using the IAM Identity Center directory for your identity provider, If that is not the case, see Configure your identity provider for Transfer Family web apps before proceeding with this tutorial.

After you complete the tutorial, your user can log in and interact with the web app that you create.

Step 1: Create the necessary supporting resources

You need to add a user to your IAM Identity Center directory. You also need two roles: one to use as an identity bearer role for your web app, and a second to use for configuring an Amazon S3 access grant. For the tutorial, we allow AWS services to create these roles for us.

To add a user

1. Sign in to the AWS Management Console and open the AWS IAM Identity Center console at https://console.aws.amazon.com/singlesignon/.

2. From the left navigation pane, choose **Users**.
3. Choose **Add user** and specify the user details.

   Specify a username, email address, and other required information. You can choose to either send an email to the user with instructions for setting up their password, or you can generate a one-time password to share with them.
4. Choose **Next** and optionally assign the new user to one or more groups.
5. Choose **Next** and review your choices.

   If everything looks good, choose **Add user** to create the new user with the details that you specified.

   For the tutorial, the example user is **Bob Stiles**, username **bobstiles** and email address **bobstiles@example.com**.

Step 2: Create a Transfer Family web app

To create a Transfer Family web app

1. Sign in to the AWS Management Console and open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Web apps**.
3. Choose **Create web app**.

   For authentication access, note that the service automatically finds the AWS IAM Identity Center instance that you set up as a prerequisite.
4. In the **Permission type** pane, select **Create and use a new service role**. The service creates the identity bearer role for you. An identity bearer role includes an authenticated user's identity in its sessions.
5. In the **Web app units** pane, accept the default value of 1, or adjust to a higher value if needed.
6. Add a tag to help you organize your web apps. For the tutorial, enter **Name** for the key and **Tutorial web app** for the value.

> **Tip**
>
> You can edit the web app name directly from the **Web apps** list page after you create it.

7. Choose **Next** to open the **Design web app** page. On this screen, provide the following information.

   You can optionally provide a title for your web app. You can also upload image files for your logo and favicon.

   - For the page title, customize the title for the browser tab that your users see when they connect to the web app. If you don't enter anything for the page title, it defaults to **Transfer Web App**.
   - For the logo, upload an image file. The maximum upload size for your logo image is 2 MB.
   - For the favicon, upload an image file. The maximum upload size for your favicon is 2 MB.
8. Choose **Next**, then choose **Create web app**.

To provide a branded experience, you can provide a custom URL for your users to access your Transfer Family web app. For details, see Update your access endpoint with a custom URL.

Step 3: Configure Cross-origin resource sharing (CORS) for your bucket

You must set up cross-origin resource sharing (CORS) for all buckets that are used by your web app. A *CORS configuration* is a document that defines rules that identify the origins that you will allow to access your bucket. For more information about CORS, see Configuring cross-origin resource sharing (CORS).

To set up Cross-origin resource sharing (CORS) for your Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Buckets** from the left navigation panel and search for your bucket in the search dialog, then choose the **Permissions** tab.
3. In **Cross-origin resource sharing (CORS)**, choose **Edit** and paste in the following code. Replace *`AccessEndpoint`* with the actual access endpoint for your web app. Make sure not to enter trailing slashes, because doing so causes errors when users attempt to log on to your web app.

- Incorrect example: `https://webapp-c7bf3423.transfer-webapp.us-east-2.on.aws/`
- Correct example: `https://webapp-c7bf3423.transfer-webapp.us-east-2.on.aws`

If you are reusing a bucket for multiple web apps, append their web app access endpoints to the `AllowedOrigins` list.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "PUT",
      "POST",
      "DELETE",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://AccessEndpoint"
    ],
    "ExposeHeaders": [
      "last-modified",
       "content-length",
      "etag",
      "x-amz-version-id",
      "content-type",
      "x-amz-request-id",
      "x-amz-id-2",
      "date",
      "x-amz-cf-id",
      "x-amz-storage-class",
      "access-control-expose-headers"
     ],
    "MaxAgeSeconds": 3000
  }
]
```

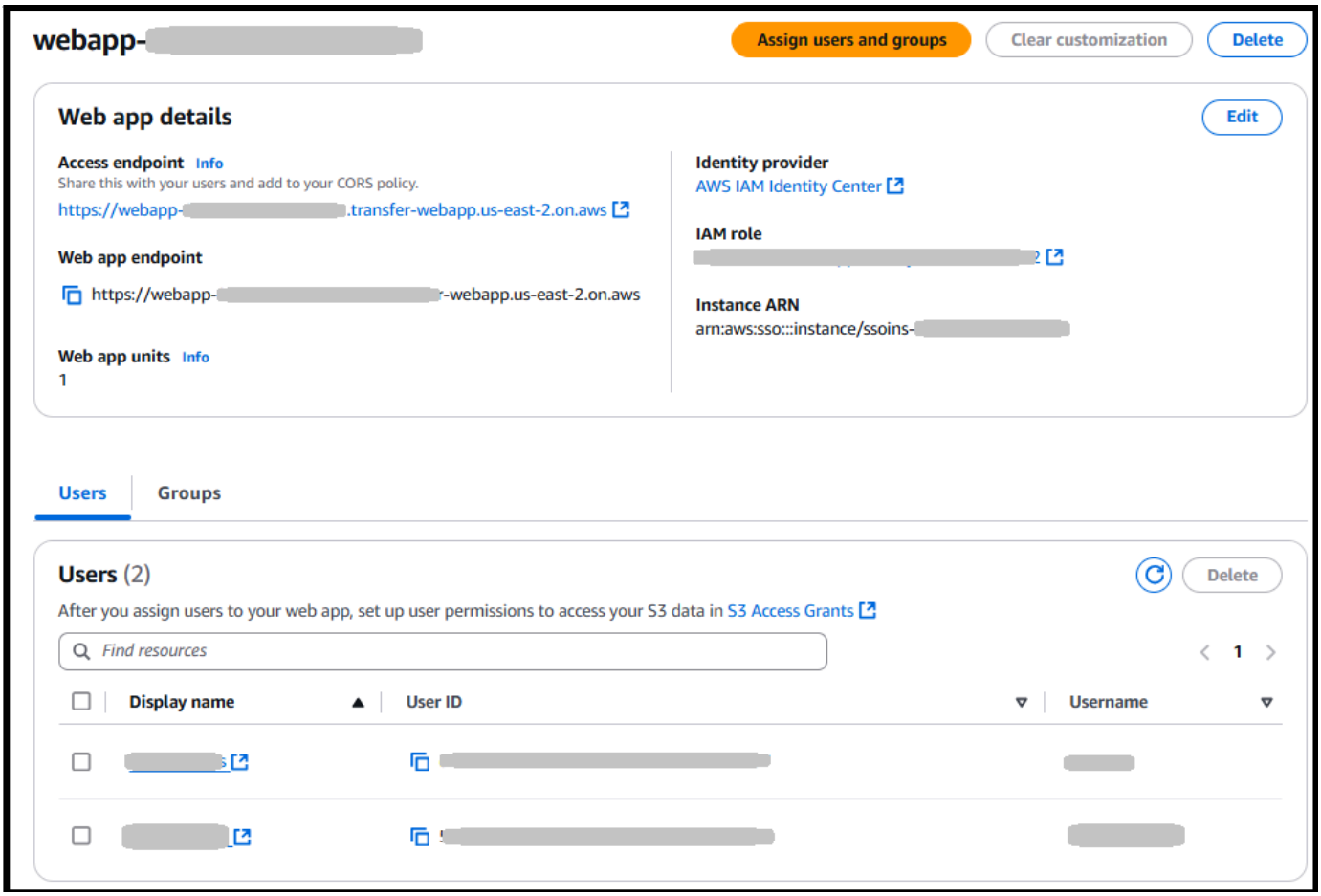
4. Choose **Save changes** to update the CORS.

Step 4: Add a user to your Transfer Family web app

Add the user that you previously created in IAM Identity Center.

To assign users to a Transfer Family web app

1. Navigate to the web app that you created earlier.
2. Choose **Assign users and groups**.



3. To assign the user that you previously created in IAM Identity Center, select **Assign existing users and groups** and select **Next**.

   a. Search for the user by the display name. Note that no users appear until you begin entering your search criteria. To add **`Bob Stiles`**, enter **bob** in the search box. If you can't find your user, navigate to the IAM Identity Center management console, find the user, then copy and paste their display name here.

   b. Choose the **`Bob Stiles`** user, then choose **Assign**.

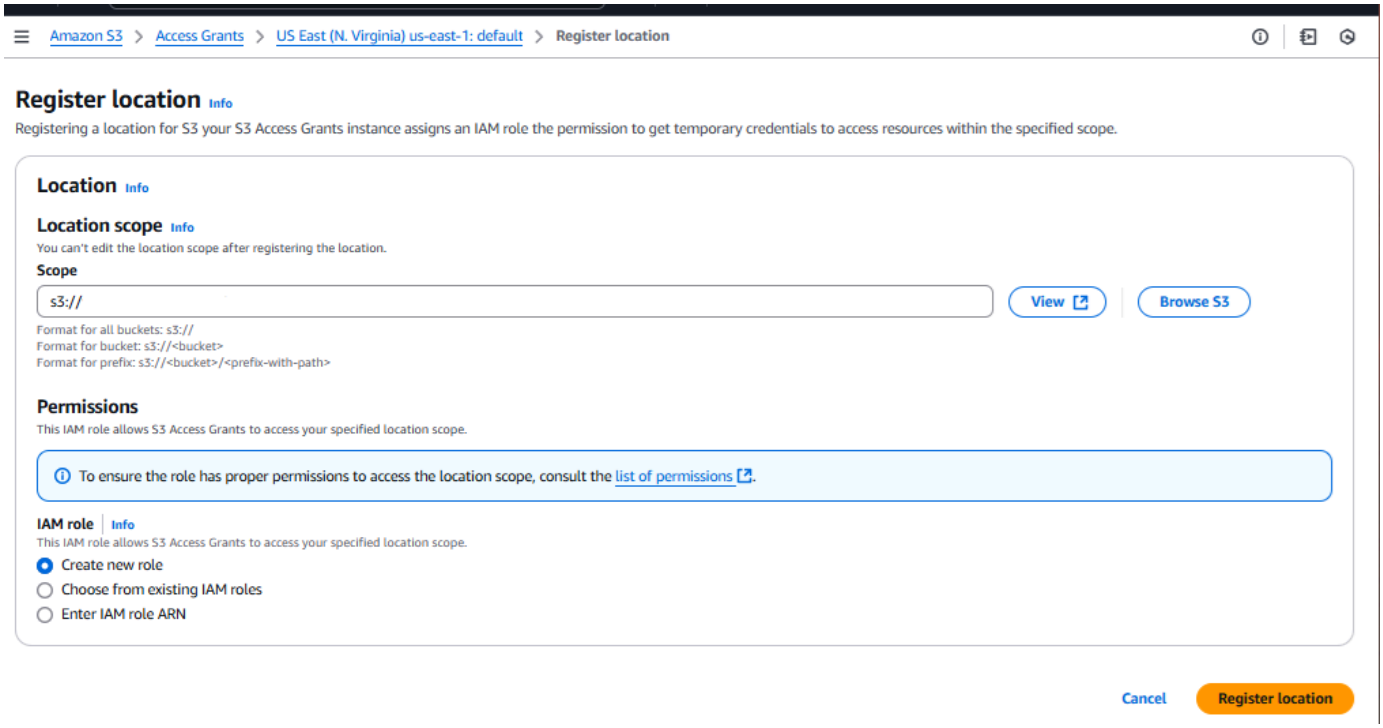
Step 5: Register a location in Amazon S3 and create an access grant

After you assign a user to your web app, you need to register a bucket and create an access grant for that user.

> **Note**
>
> You must have an S3 Access Grants instance before you can proceed. For details, see Create an S3 Access Grants instance in the *Amazon Simple Storage Service User Guide*.

To register a location and create an access grant

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Access Grants** from the left navigation pane.
3. Choose **View details** to see the details for your S3 Access Grants instance.
4. Select the **Locations** tab, then choose **Register location**.
5. Provide the following information.
   - For the **Scope**, browse for a bucket or enter the name of your bucket, and optionally a prefix. Note that the scope begins with the string **`s3://`**.
   - For the IAM role, choose **Create new role** to have Amazon S3 create a role. This role allows S3 Access Grants to access your specified location scope.



Choose **Register location** to continue.

6. Select the **Grants** tab, then choose **Create Grant** and provide the following details.

   - For **Location**, select **Browse locations** and choose the location that you registered in the previous step.
   - For **Subprefix**, enter ***** to indicate that the access grant applies to the entire bucket.
   - For **Permissions**, select **Read** and **Write**.
   - For **Grantee type**, choose **Directory identity from IAM Identity Center**.
   - For **Directory identity type**, select **User**.
   - In **IAM Identity Center user/ ID**, copy and paste the user ID for **`Bob Stiles`**. This ID is available in the **Users** pane in your Transfer Family web app.

7. Choose **Create Grant**.

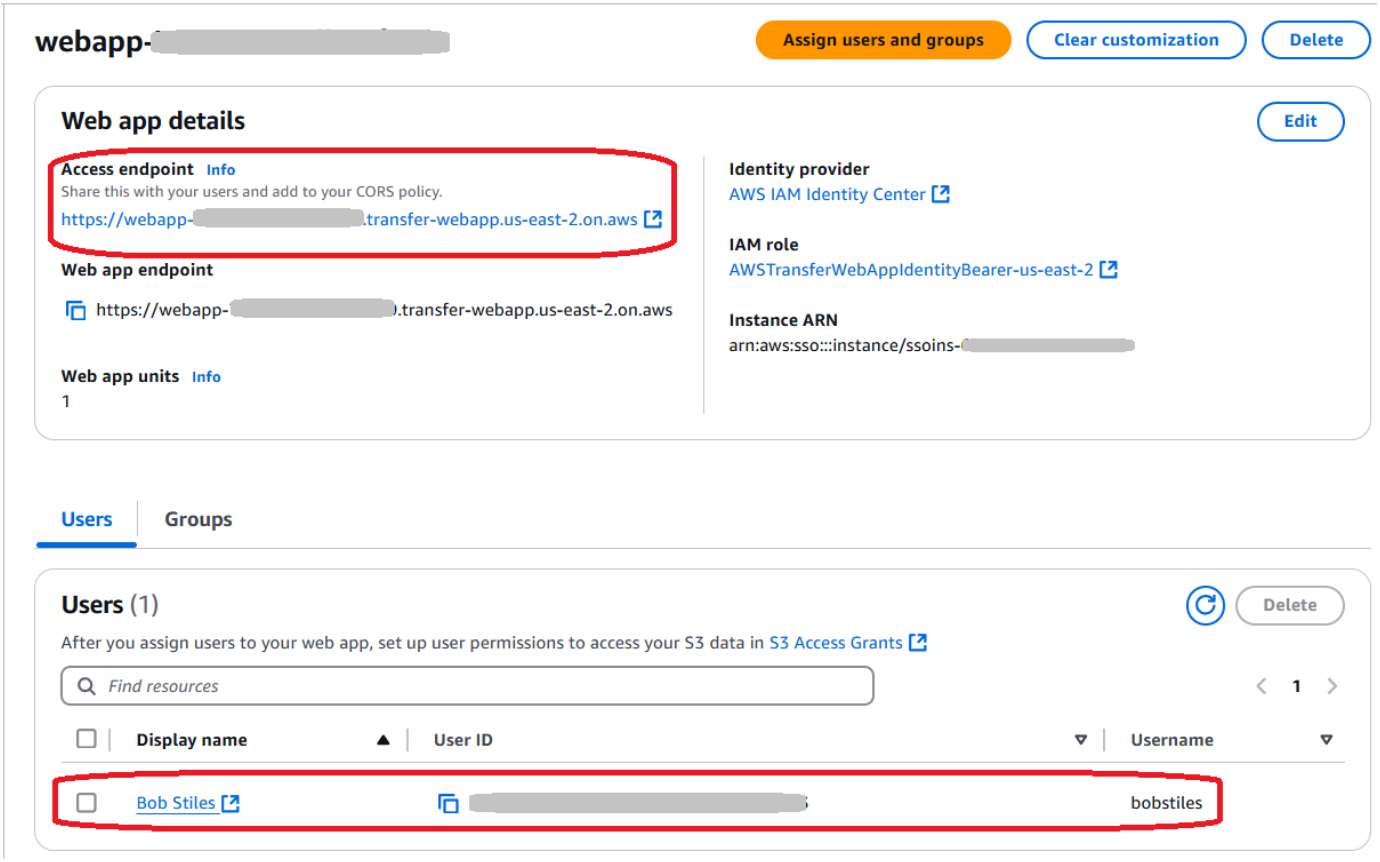
The access grant is created.

Step 6: Access your Transfer Family web app as a user

Now, we navigate to the web app's URL and log in as the user that we assigned earlier.

To log onto the Transfer Family web app

1. Navigate to your web app
2. Choose the **Access endpoint** from the **Web app details** pane.



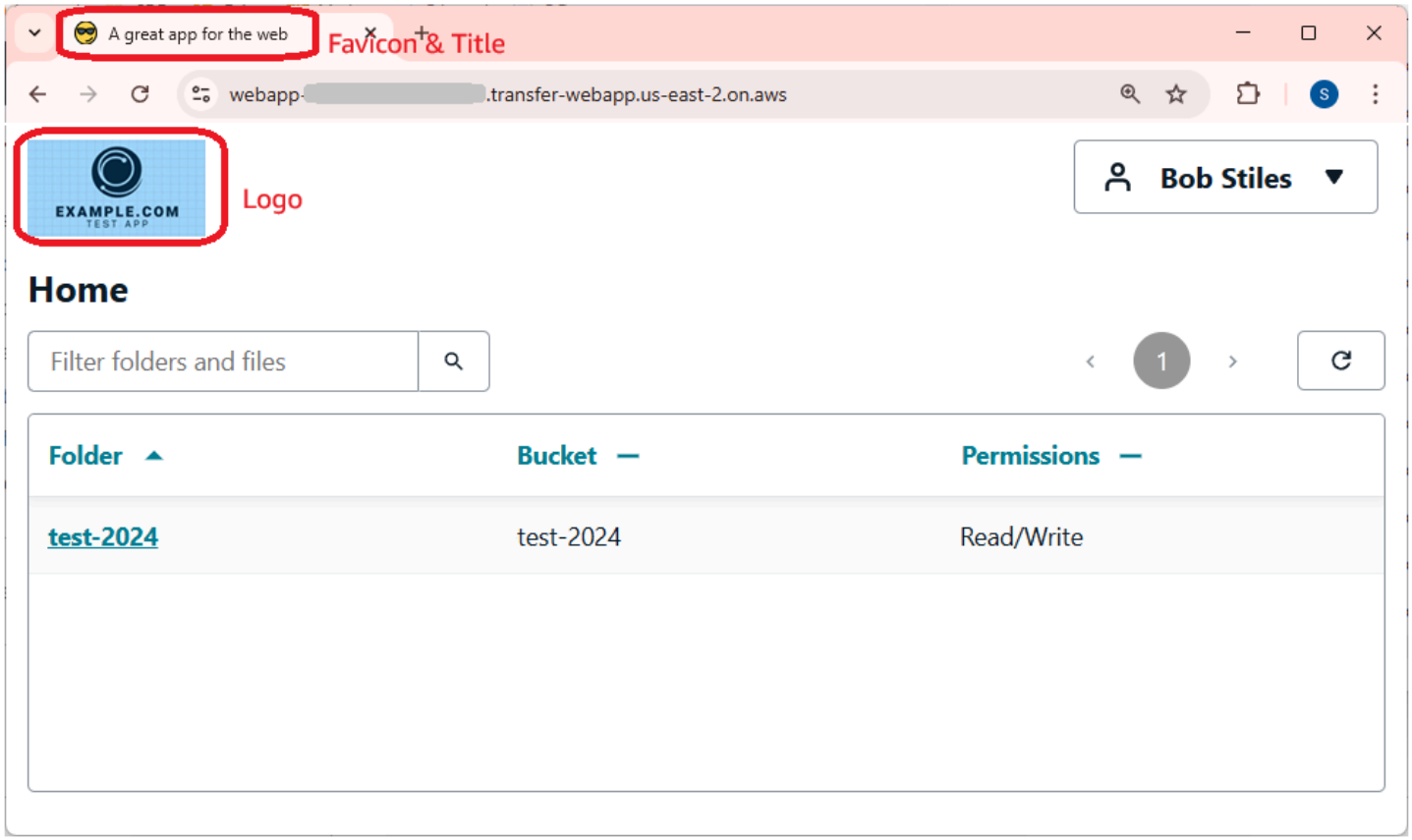
3. On the sign in screen, enter the user you created, **`bobstiles`**, then select **Next**.
4. Enter the password that the system assigned to this user when created and select **Next**.
5. If your organization requires multi-factor authentication (MFA), you need to set it up now. If not, skip ahead to step 6.

   a. You are presented with a screen to register your MFA device. Choose one of the available options and select **Next**.

   b. Perform the necessary steps to configure MFA for this user: the steps depend upon the MFA option you chose.

   c. You may have to set a new password for your user: if required, do so now. The system may also require that you sign in again, using the new MFA credentials that you configured.

Your use should see a screen similar to the following. Note that this screenshot includes customization for the favicon and logo.



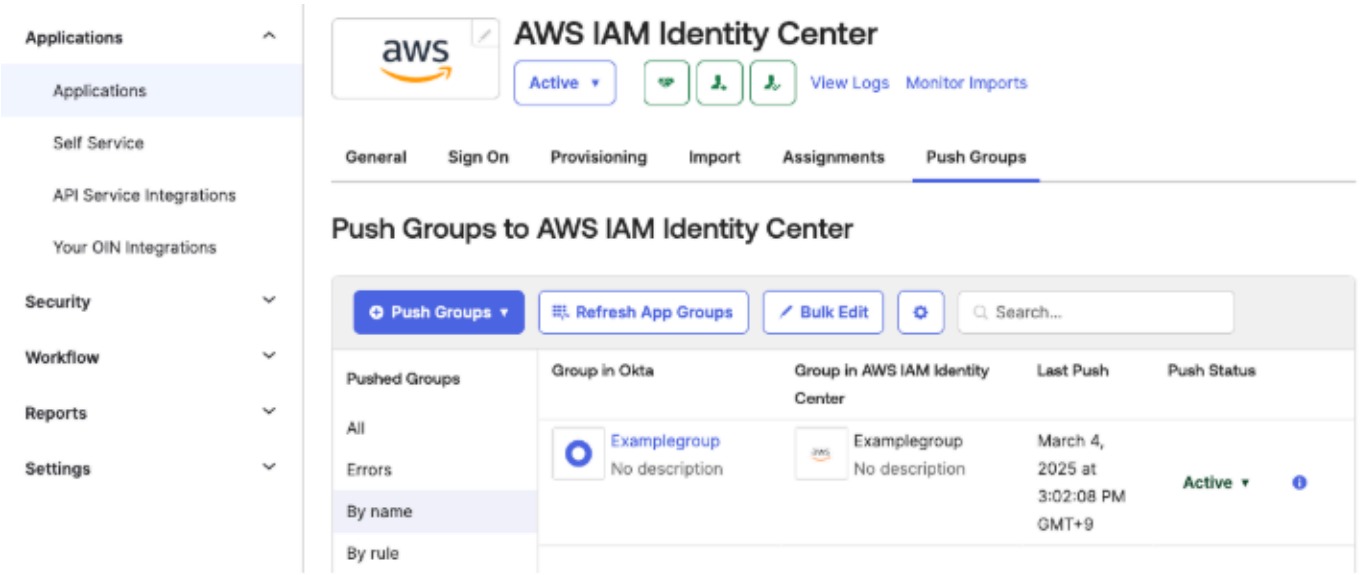
Next steps

You have successfully set up a basic Transfer Family web app with standard S3 bucket access. If you need more granular control over bucket permissions, such as allowing users to download from one bucket and upload to another, see Tutorial: Setting up AWS Transfer Family web app with selective multi-bucket access.

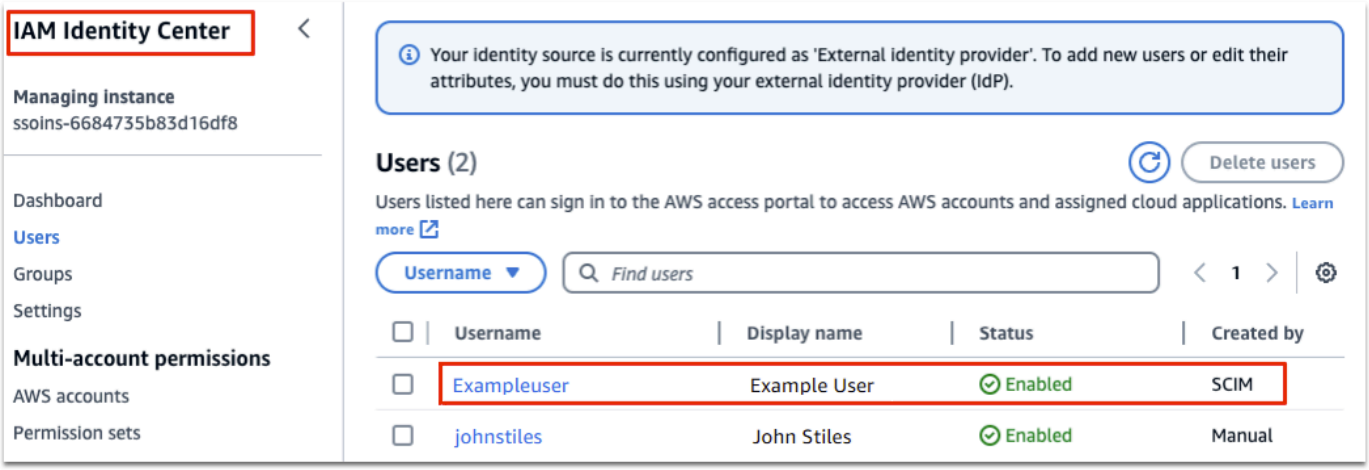
Integrate Okta as your identity provider for web apps

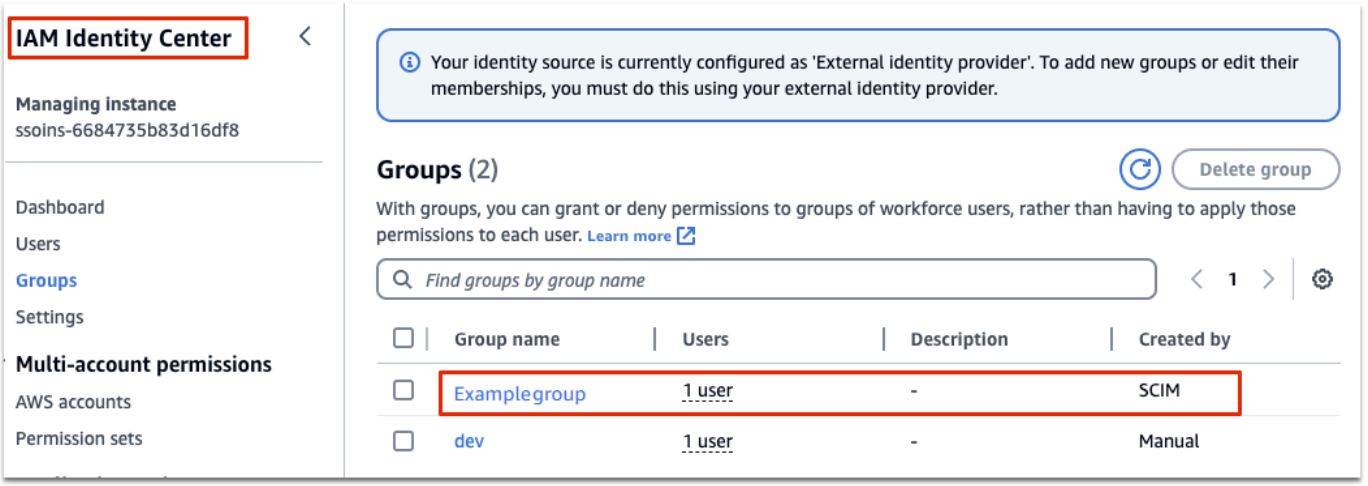
You can integrate an external identity provider with Transfer Family web apps. This section describes how to set up Okta as your identity provider.

1. In Okta, create a user, group, and application. For details on how to do this, see Configure SAML and SCIM with Okta and IAM Identity Center.

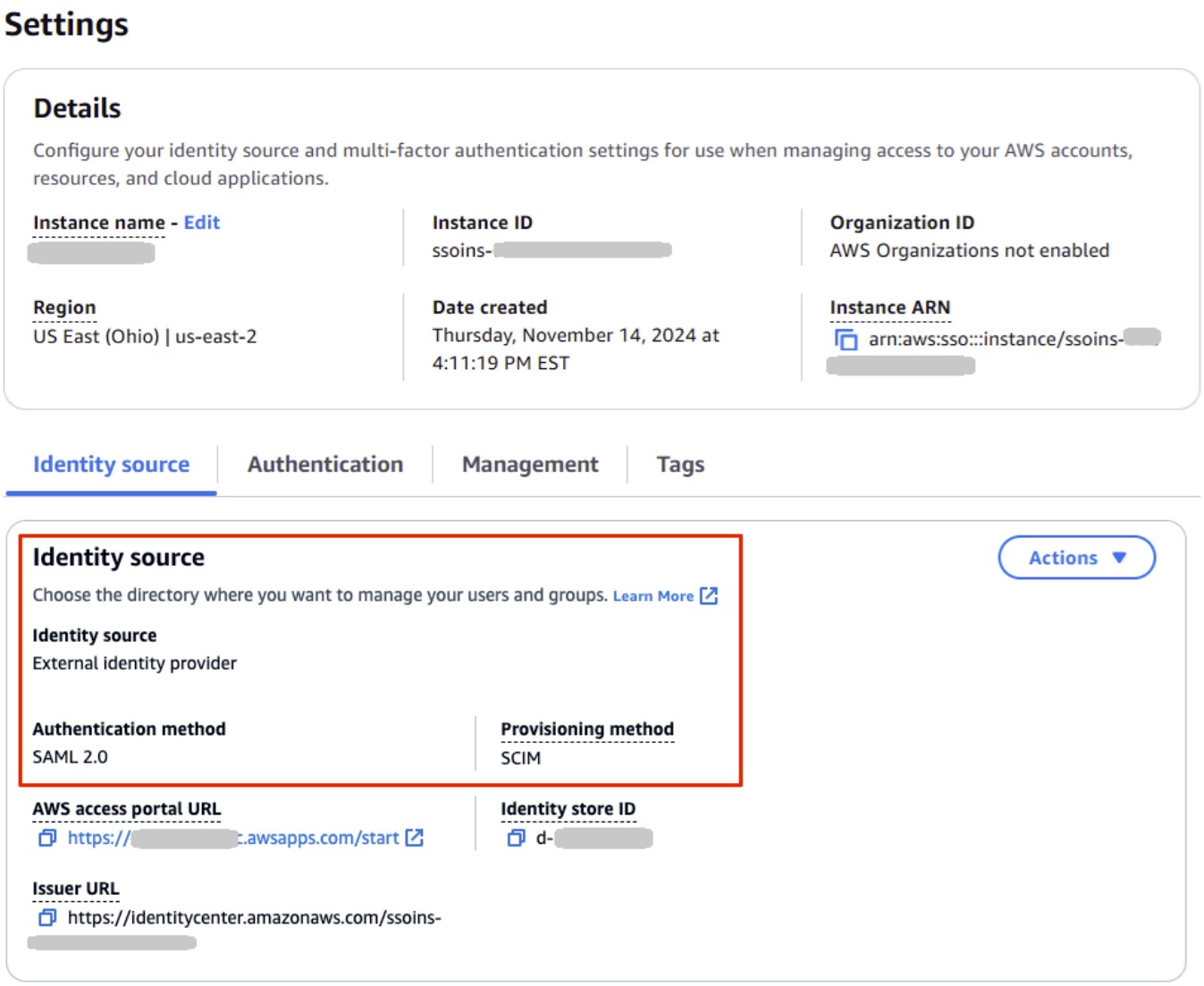


2. Connect Okta and import user and group from Okta to AWS IAM Identity Center. Follow steps 1–4 in [Configure SAML and SCIM with Okta and IAM Identity Center](#).





3. Confirm that the identity source in IAM Identity Center is SAML 2.0.



4. Assign your user and group, as described in Step 4: Add a user to your Transfer Family web app.
5. To avoid having your users need to use MFA when logging into your web app, perform the following steps in Okta.
   a. From the Okta admin console, access **[Applications] - [Applications]**, and select the AWS IAM Identity Center application.
   b. On the **Sign on** tab, select **[User authentication] - Edit**.
   c. Select **Password only**.

After you perform all of the other steps for the tutorial, your user should be able to access your Transfer Family web app by navigating to the web app's access endpoint in a web browser.

Tutorial: Setting up AWS Transfer Family web app with selective multi-bucket access

This tutorial guides you through configuring a Transfer Family web app with specific Amazon S3 bucket permissions for a single user. You'll learn how to set up a solution that allows users to download from one bucket and upload to another while maintaining security. This is an advanced tutorial that builds on the concepts covered in the basic tutorial. If you're new to AWS Transfer Family web apps, consider starting with Tutorial: Setting up a basic Transfer Family web app.

Prerequisites

Before you begin this tutorial, you need:

- IAM Identity Center configured in the same region as your AWS Transfer Family web app. Note that AWS allows only one instance of IAM Identity Center per AWS account for all Regions.
- At least one user configured in IAM Identity Center.
- Two S3 buckets: one for downloads and one for uploads.

> **Note**
>
> This tutorial shares many prerequisites with the basic web app tutorial. For more information about setting up IAM Identity Center and creating users, see Tutorial: Setting up a basic Transfer Family web app.

Step 1: Create a Transfer Family web app

To create a Transfer Family web app

1. Sign in to the AWS Management Console and open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Web apps**.
3. Choose **Create web app**.

For authentication access, note that the service automatically finds the AWS IAM Identity Center instance that you set up as a prerequisite.

4. In the **Permission type** pane, select **Create and use a new service role**. The service creates the identity bearer role for you. An identity bearer role includes an authenticated user's identity in its sessions.
5. In the **Web app units** pane, accept the default value of 1, or adjust to a higher value if needed.
6. Add a tag to help you organize your web apps. For the tutorial, enter **Name** for the key and **Tutorial web app** for the value.

   > **Tip**
   >
   > You can edit the web app name directly from the **Web apps** list page after you create it.

7. Choose **Next** to open the **Design web app** page. On this screen, provide the following information.

   You can optionally provide a title for your web app. You can also upload image files for your logo and favicon.

   - For the page title, customize the title for the browser tab that your users see when they connect to the web app. If you don't enter anything for the page title, it defaults to **Transfer Web App**.
   - For the logo, upload an image file. The maximum upload size for your logo image is 2 MB.
   - For the favicon, upload an image file. The maximum upload size for your favicon is 2 MB.
8. Choose **Next**, then choose **Create web app**.

Step 2: Configure IAM roles for S3 access

You need to create two IAM roles: one with download-only access to the first bucket and another with upload-only access to the second bucket.

Trust policy for both roles

Use the following trust policy for both IAM roles:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessGrantsTrustPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "access-grants.s3.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity",
                "sts:SetContext"
            ]
        }
    ]
}
```

IAM policy for download bucket

Create an IAM role with the following policy for read-only access to your download bucket:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ObjectLevelReadPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetObjectAcl",
                "s3:GetObjectVersionAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        }
```

```
    ]
}
```

> **Important**
>
> Replace **amzn-s3-demo-bucket1** with the actual name of your download bucket.

IAM policy for upload bucket

Create another IAM role with the following policy for write access to your upload bucket:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ObjectLevelWritePermissions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectVersionAcl",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion",
                "s3:AbortMultipartUpload",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket2/*",
                "arn:aws:s3:::amzn-s3-demo-bucket2"
            ]
        }
    ]
}
```

> **Important**
>
> Replace **amzn-s3-demo-bucket2** with the actual name of your upload bucket.

Step 3: Set up S3 Access Grants

1. Open the S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access Grants**.
3. Click **Create S3 Access Grants instance**.
4. Select the option **Add IAM Identity Center instance** and enter the identity center instance ARN.
5. Click **Next**, then click **Cancel** to finish creating the S3 Access Grants instance without proceeding with additional steps.

This step creates the S3 Access Grants instance. You'll now register locations and create access grants.

Step 4: Register S3 bucket locations

Register both S3 buckets as locations with S3 Access Grants:

1. In the S3 Access Grants console, navigate to **Locations** and click **Register location**.
2. Under **Location scope**, choose the specific S3 bucket for downloads (amzn-s3-demo-bucket1).
3. When prompted to choose an IAM role, select the download IAM role you created earlier.
4. Complete the registration process.
5. Repeat the process to register the upload bucket (amzn-s3-demo-bucket2), selecting the upload IAM role when prompted.

Step 5: Create Access Grants

Create two grants, one for each registered location:

1. In the S3 Access Grants console, navigate to **Grants** and click **Create grant**.
2. In **Location**, click **Browse location** and select the download bucket location (amzn-s3-demo-bucket1).
3. In **Subprefix** (optional), enter `*` to allow access to the entire bucket, or specify a path like `folder1/folder2/*` to restrict access to a specific prefix.

Using * will set the grant scope to `s3://bucket-name/*`, allowing access to the entire bucket. To allow access to only a specific prefix, enter a path like `folder1/folder2/*`, which will set the grant scope to `s3://bucket-name/folder1/folder2/*`.

4. Under **Permissions and access**, select **Read** for the download bucket.
5. In **Grantee type**, choose **Directory identity from IAM Identity Center**.
6. For **IAM principal type**, select **User** and enter the User ID of your IAM Identity Center user.
7. Complete the grant creation process.
8. Repeat the process to create a grant for the upload bucket (amzn-s3-demo-bucket2), but select **Read-write** for the permissions.

Step 6: Configure CORS policy for S3 buckets

Configure a CORS policy for both S3 buckets to allow access through your AWS Transfer Family WebApp:

1. Open the S3 console and navigate to your download bucket (amzn-s3-demo-bucket1).
2. Select the **Permissions** tab.
3. Scroll down to the **Cross-origin resource sharing (CORS)** section and click **Edit**.
4. Add the following CORS configuration, replacing *`WebAppEndpoint`* with your actual WebApp endpoint URL:

   You can find your web app endpoint URL in the AWS Transfer Family console under WebApps. It will look similar to **https://webapp-**********.transfer-webapp.us-west-2.on.aws**.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "PUT",
      "POST",
      "DELETE",
      "HEAD"
    ],
```

```
    "AllowedOrigins": [
      "https://WebAppEndpoint"
    ],
    "ExposeHeaders": [
      "last-modified",
      "content-length",
      "etag",
      "x-amz-version-id",
      "content-type",
      "x-amz-request-id",
      "x-amz-id-2",
      "date",
      "x-amz-cf-id",
      "x-amz-storage-class",
      "access-control-expose-headers"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

5. Click **Save changes**.
6. Repeat the process for your upload bucket (amzn-s3-demo-bucket2).

Step 7: Test the configuration

1. Open your AWS Transfer Family web app URL. You can find this URL in the AWS Transfer Family console under WebApps in the **Access endpoint** field.
2. Log in using the IAM Identity Center user credentials you configured with access grants.
3. After logging in, you should see both S3 locations on the home page.
4. Navigate to the download bucket (amzn-s3-demo-bucket1) and verify that you can download files but not upload.
5. Navigate to the upload bucket (amzn-s3-demo-bucket2) and verify that you can upload files.

Conclusion

You have successfully configured AWS Transfer Family WebApp with selective S3 bucket access for a single user. This setup allows the user to download from one bucket and upload to another while maintaining security through IAM roles and S3 Access Grants.

This approach can be extended to multiple users by creating additional grants in S3 Access Grants for each user, allowing for granular control over bucket access permissions. For information about the basic web app setup, see Tutorial: Setting up a basic Transfer Family web app.

Configuring an SFTP, FTPS, or FTP server endpoint

This topic provides details for creating and using AWS Transfer Family server endpoints that use one or more of the SFTP, FTPS, and FTP protocols.

Topics

- Identity provider options
- AWS Transfer Family endpoint type matrix
- Configuring an SFTP, FTPS, or FTP server endpoint
- FTP and FTPS Network Load Balancer considerations
- Transferring files over a server endpoint using a client
- Managing users for server endpoints
- Using logical directories to simplify your Transfer Family directory structures
- Access your FSx for NetApp ONTAP file systems with Transfer Family

Identity provider options

AWS Transfer Family provides several methods for authenticating and managing users. The following table compares the available identity providers that you can use with Transfer Family.

Action	AWS Transfer Family service managed	AWS Managed Microsoft AD	Amazon API Gateway	AWS Lambda
Supported protocols	SFTP	SFTP, FTPS, FTP	SFTP, FTPS, FTP	SFTP, FTPS, FTP
Key-based authentication	Yes	No	Yes	Yes
Password authentication	No	Yes	Yes	Yes
AWS Identity and Access	Yes	Yes	Yes	Yes

Action	AWS Transfer Family service managed	AWS Managed Microsoft AD	Amazon API Gateway	AWS Lambda
Management (IAM) and POSIX				
Logical home directory	Yes	Yes	Yes	Yes
Parameterized access (username-based)	Yes	Yes	Yes	Yes
Ad hoc access structure	Yes	No	Yes	Yes
AWS WAF	No	No	Yes	No

Notes:

- IAM is used to control access for Amazon S3 backing storage, and POSIX is used for Amazon EFS.
- *Ad hoc* refers to the ability to send the user profile at runtime. For example, you can land users in their home directories by passing the username as a variable.
- For details about AWS WAF, see Add a web application firewall.
- There is a blog post that describes using a Lambda function integrated with Microsoft Entra ID (formerly Azure AD) as your Transfer Family identity provider. For details, see Authenticating to AWS Transfer Family with Azure Active Directory and AWS Lambda.
- We provide several CloudFormation templates to help you quickly deploy a Transfer Family server that uses a custom identity provider. For details, see Lambda function templates.

In the following procedures, you can create an SFTP-enabled server, FTPS-enabled server, FTP-enabled server, or AS2-enabled server.

Next step

- Create an SFTP-enabled server
- Create an FTPS-enabled server
- Create an FTP-enabled server
- Configuring AS2

AWS Transfer Family endpoint type matrix

When you create a Transfer Family server, you choose the type of endpoint to use. The following table describes characteristics for each type of endpoint.

Endpoint type matrix

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
Supported protocols	SFTP	SFTP, FTPS, AS2	SFTP, FTP, FTPS, AS2	SFTP
Access	From over the internet. This endpoint type doesn't require any special configuration in your VPC.	Over the internet and from within VPC and VPC-connected environments, such as an on-premises data center over Direct Connect or VPN.	From within VPC and VPC-connected environments, such as an on-premises data center over Direct Connect or VPN.	From within VPC and VPC-connected environments, such as an on-premises data center over Direct Connect or VPN.
Static IP address	You can't attach a static IP address. AWS provides IP addresses that are subject to change.	You can attach Elastic IP addresses to the endpoint. These can be AWS-owned IP addresses or your own	Private IP addresses attached to the endpoint don't change.	Private IP addresses attached to the endpoint don't change.

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
		IP addresses (Bring your own IP addresses). Elastic IP addresses attached to the endpoint don't change. Private IP addresses attached to the server also don't change.		

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
Source IP allow list	This endpoint type does not support allow lists by source IP addresses. The endpoint is publicly accessible and listens for traffic over port 22. **Note** For VPC-hosted endpoints, SFTP Transfer Family servers can operate over port 22 (the default), 2222, 2223, or 22000.	To allow access by source IP address, you can use security groups attached to the server endpoints and network ACLs attached to the subnet that the endpoint is in.	To allow access by source IP address, you can use security groups attached to the server endpoints and network access control lists (network ACLs) attached to the subnet that the endpoint is in.	To allow access by source IP address, you can use security groups attached to the server endpoints and network ACLs attached to the subnet that the endpoint is in.

Characteristic	Public	VPC - Internet	VPC - Internal	VPC_Endpoint (deprecated)
Client firewall allow list	You must allow the DNS name of the server. Because IP addresses are subject to change, avoid using IP addresses for your client firewall allow list.	You can allow the DNS name of the server or the Elastic IP addresses attached to the server.	You can allow the private IP addresses or the DNS name of the endpoints.	You can allow the private IP addresses or the DNS name of the endpoints.
IP address type	IPv4 (default) or dual-stack (IPv4 and IPv6)	IPv4 only (dual-stack not supported)	IPv4 (default) or dual-stack (IPv4 and IPv6)	IPv4 only (dual-stack not supported)

Note

The `VPC_ENDPOINT` endpoint type is now deprecated and cannot be used to create new servers. Instead of using `EndpointType=VPC_ENDPOINT`, use the VPC endpoint type (`EndpointType=VPC`), which you can use as either **Internal** or **Internet Facing**, as described in the preceding table.

- For details about the deprecation, see Discontinuing the use of VPC_ENDPOINT.
- For information about managing VPC endpoint permissions, see Limiting VPC endpoint access for Transfer Family servers.

Consider the following options to increase the security posture of your AWS Transfer Family server:

- Use a VPC endpoint with internal access, so that the server is accessible only to clients within your VPC or VPC-connected environments such as an on-premises data center over Direct Connect or VPN.
- To allow clients to access the endpoint over the internet and protect your server, use a VPC endpoint with internet-facing access. Then, modify the VPC's security groups to allow traffic only from certain IP addresses that host your users' clients.
- If you require password-based authentication and you use a custom identity provider with your server, it's a best practice that your password policy prevents users from creating weak passwords and limits the number of failed login attempts.
- AWS Transfer Family is a managed service, and so it doesn't provide shell access. You cannot directly access the underlying SFTP server to run OS native commands on Transfer Family servers.
- Use a Network Load Balancer in front of a VPC endpoint with internal access. Change the listener port on the load balancer from port 22 to a different port. This can reduce, but not eliminate, the risk of port scanners and bots probing your server, because port 22 is most commonly used for scanning. For details, see the blog post Network Load Balancers now support Security groups.

> **Note**
>
> If you use a Network Load Balancer, the AWS Transfer Family CloudWatch logs show the IP address for the NLB, rather than the actual client IP address.

Configuring an SFTP, FTPS, or FTP server endpoint

You can create a file transfer server by using the AWS Transfer Family service. The following file transfer protocols are available:

- Secure Shell (SSH) File Transfer Protocol (SFTP) – File transfer over SSH. For details, see the section called “Create an SFTP-enabled server”.

> **Note**
>
> We provide an AWS CDK example for creating an SFTP Transfer Family server. The example uses TypeScript, and is available on GitHub here.

- File Transfer Protocol Secure (FTPS) – File transfer with TLS encryption. For details, see the section called "Create an FTPS-enabled server".
- File Transfer Protocol (FTP) – Unencrypted file transfer. For details, see the section called "Create an FTP-enabled server".
- Applicability Statement 2 (AS2) – File transfer for transporting structured business-to-business data. For details, see the section called "Configure AS2". For AS2, you can quickly create an CloudFormation stack for demonstration purposes. This procedure is described in Use a template to create a demo Transfer Family AS2 stack.

You can create a server with multiple protocols.

> **Note**
>
> If you have multiple protocols enabled for the same server endpoint and you want to provide access by using the same username over multiple protocols, you can do so as long as the credentials specific to the protocol have been set up in your identity provider. For FTP, we recommend maintaining separate credentials from SFTP and FTPS. This is because, unlike SFTP and FTPS, FTP transmits credentials in clear text. By isolating FTP credentials from SFTP or FTPS, if FTP credentials are shared or exposed, your workloads using SFTP or FTPS remain secure.

When you create a server, you choose a specific AWS Region to perform the file operation requests of users who are assigned to that server. Along with assigning the server one or more protocols, you also assign one of the following identity provider types:

- **Service managed by using SSH keys**. For details, see Working with service-managed users.
- **AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD)**. This method allows you integrate your Microsoft Active Directory groups to provide access to your Transfer Family servers. For details, see Using AWS Directory Service for Microsoft Active Directory.
- **A custom identity provider**. Transfer Family offers several options for using a custom identity provider, as described in the Working with custom identity providers topic.

You also assign the server an endpoint type (publicly accessible or VPC hosted) and a hostname by using the default server endpoint, or a custom hostname by using the Amazon Route 53 service or

by using a Domain Name System (DNS) service of your choice. A server hostname must be unique in the AWS Region where it's created.

Additionally, you can assign an Amazon CloudWatch logging role to push events to your CloudWatch logs, choose a security policy that contains the cryptographic algorithms that are enabled for use by your server, and add metadata to the server in the form of tags that are key-value pairs.

> **Important**
>
> You incur costs for instantiated servers and for data transfer. For information about pricing and to use AWS Pricing Calculator to get an estimate of the cost to use Transfer Family, see AWS Transfer Family pricing.

Create an SFTP-enabled server

Secure Shell (SSH) File Transfer Protocol (SFTP) is a network protocol used for secure transfer of data over the internet. The protocol supports the full security and authentication functionality of SSH. It's widely used to exchange data, including sensitive information between business partners in a variety of industries such as financial services, healthcare, retail, and advertising.

Note the following

- SFTP servers for Transfer Family operate over port 22. For VPC-hosted endpoints, SFTP Transfer Family servers can also operate over port 2222, 2223 or 22000. For details, see Create a server in a virtual private cloud.
- Public endpoints cannot restrict traffic via security groups. To use security groups with your Transfer Family server, you must host your server's endpoint inside a virtual private cloud (VPC) as described in Create a server in a virtual private cloud.

See also

- We provide an AWS CDK example for creating an SFTP Transfer Family server. The example uses TypeScript, and is available on GitHub here.
- For a walkthrough of how to deploy a Transfer Family server inside of a VPC, see Use IP allow list to secure your AWS Transfer Family servers.

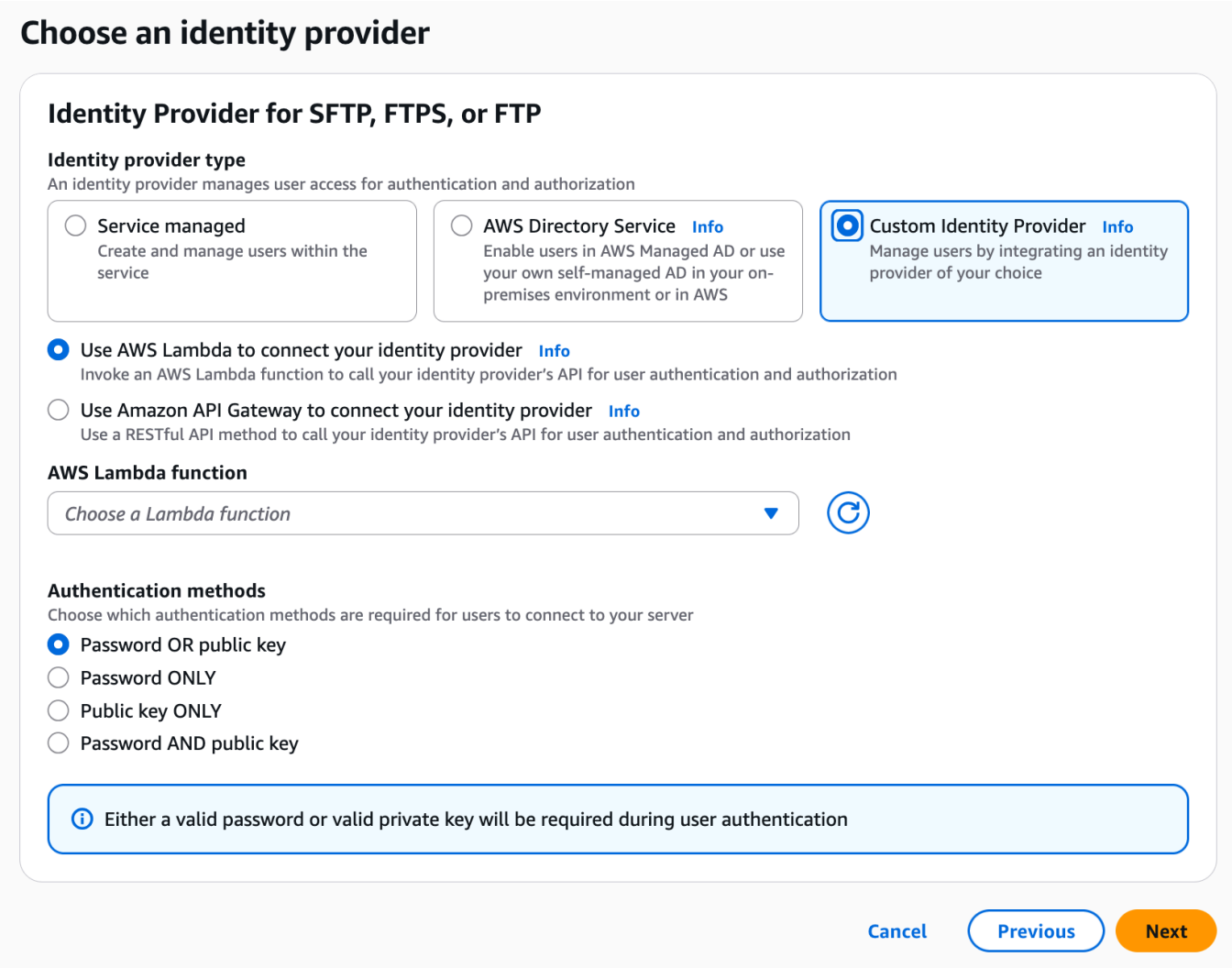
To create an SFTP-enabled server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/ and select **Servers** from the navigation pane, then choose **Create server**.
2. In **Choose protocols**, select **SFTP**, and then choose **Next**.
3. In **Choose an identity provider**, choose the identity provider that you want to use to manage user access. You have the following options:

   - **Service managed** – You store user identities and keys in AWS Transfer Family.
   - **AWS Directory Service for Microsoft Active Directory** – You provide an Directory Service directory to access the endpoint. By doing so, you can use credentials stored in your Active Directory to authenticate your users. To learn more about working with AWS Managed Microsoft AD identity providers, see Using AWS Directory Service for Microsoft Active Directory.

     > **Note**
     >
     > - Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
     > - To set up a server with Directory Service as your identity provider, you need to add some Directory Service permissions. For details, see Before you start using AWS Directory Service for Microsoft Active Directory.

   - **Custom identity provider** – Choose either of the following options:
     - **Use AWS Lambda to connect your identity provider** – You can use an existing identity provider, backed by a Lambda function. You provide the name of the Lambda function. For more information, see Using AWS Lambda to integrate your identity provider.
     - **Use Amazon API Gateway to connect your identity provider** – You can create an API Gateway method backed by a Lambda function for use as an identity provider. You provide an Amazon API Gateway URL and an invocation role. For more information, see Using Amazon API Gateway to integrate your identity provider.



4. Choose **Next**.
5. In **Choose an endpoint**, do the following:
   a. For **Endpoint type**, choose the **Publicly accessible** endpoint type. For a **VPC hosted** endpoint, see Create a server in a virtual private cloud.
   b. For **IP address type**, choose **IPv4** (default) for backwards compatibility or **Dual-stack** to enable both IPv4 and IPv6 connections to your endpoint.

> **Note**
>
> Dual-stack mode allows your Transfer Family endpoint to communicate with both IPv4 and IPv6 enabled clients. This enables you to gradually transition from IPv4 to IPv6 based systems without needing to switch all at once.

c. (Optional) For **Custom hostname**, choose **None**.

You get a server hostname provided by AWS Transfer Family. The server hostname takes the form *serverId*.server.transfer.*regionId*.amazonaws.com.

For a custom hostname, you specify a custom alias for your server endpoint. To learn more about working with custom hostnames, see Working with custom hostnames.

d. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure that the endpoint complies with Federal Information Processing Standards (FIPS).

> **Note**
>
> FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*. For more information about FIPS, see Federal Information Processing Standard (FIPS) 140-2 .

e. Choose **Next**.

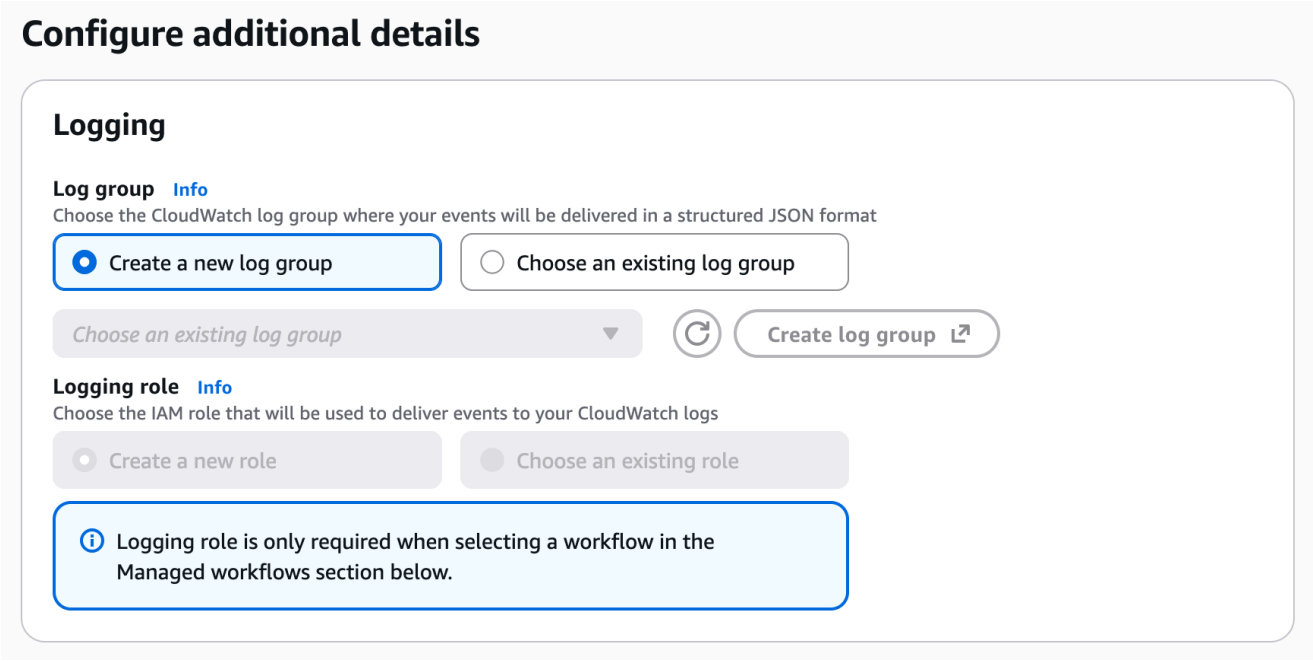
6. On the **Choose domain** page, choose the AWS storage service that you want to use to store and access your data over the selected protocol:

   - Choose **Amazon S3** to store and access your files as objects over the selected protocol.
   - Choose **Amazon EFS** to store and access your files in your Amazon EFS file system over the selected protocol.

   Choose **Next**.

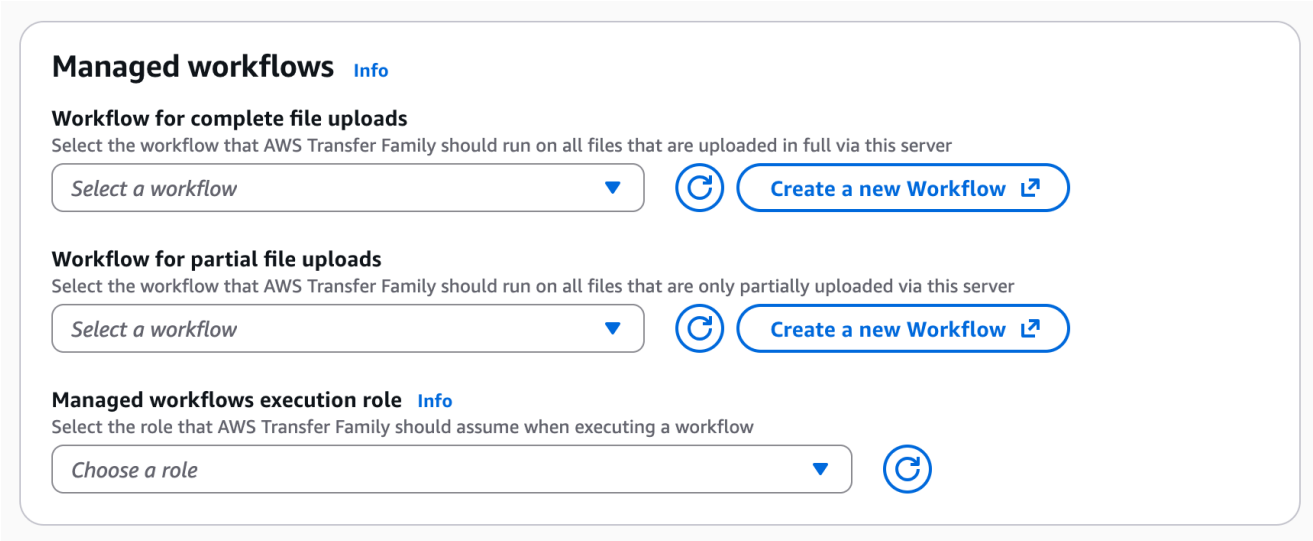
7. In **Configure additional details**, do the following:

   a. For logging, specify an existing log group or create a new one (the default option). If you choose an existing log group, you must select one that is associated with your AWS account.



If you choose **Create log group**, the CloudWatch console (https://console.aws.amazon.com/cloudwatch/) opens to the **Create log group** page. For details, see Create a log group in CloudWatch Logs.

b. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial upload. To learn more about processing your files by using managed workflows, see AWS Transfer Family managed workflows.



c. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server. Our latest security policy is the default: for details, see Security policies for AWS Transfer Family servers.

d. For **Server Host Key**, choose where Transfer Family stores your host key material. Choose **AWS Secrets Manager** (the default) or **Service managed**. For more information about storage types, see Storing SFTP server host keys in AWS Secrets Manager.

   Providing host key material is optional. If you don't enter an RSA, ED25519, or ECDSA private key, Transfer Family generates a host key for you. To identify your server when clients connect to it over SFTP, you can enter your own private key. You can also add a description to differentiate among multiple host keys.

   After you create your server, you can add additional host keys. Having multiple host keys is useful if you want to rotate keys or if you want to have different types of keys, such as an RSA key and also an ECDSA key.

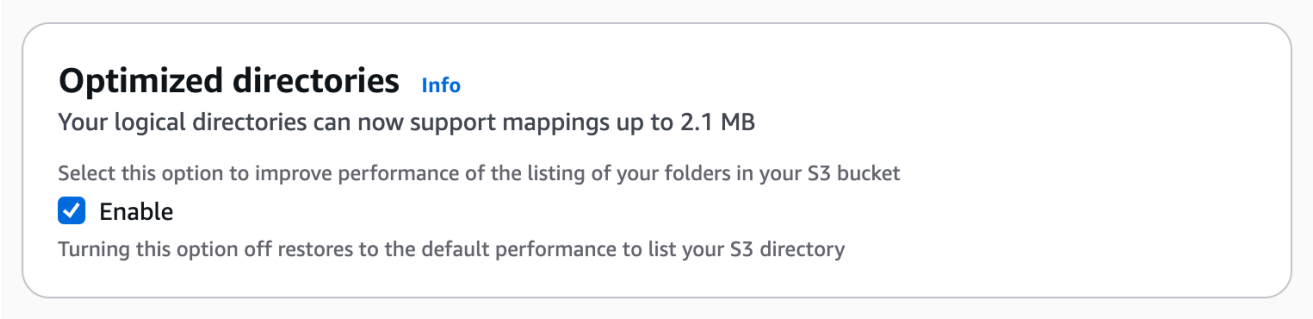
   > **Note**
   >
   > When you choose **AWS Secrets Manager** storage, Transfer Family stores the host key in a Secrets Manager secret. Transfer Family uses the AWSServiceRoleForTransfer service-linked role to access the secret. If this role doesn't already exist in your account, Transfer Family creates it for you. For more information, see Using service-linked roles for AWS Transfer Family.

e. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.

f. Choose **Next**.

g. You can optimize performance for your Amazon S3 directories. For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds.

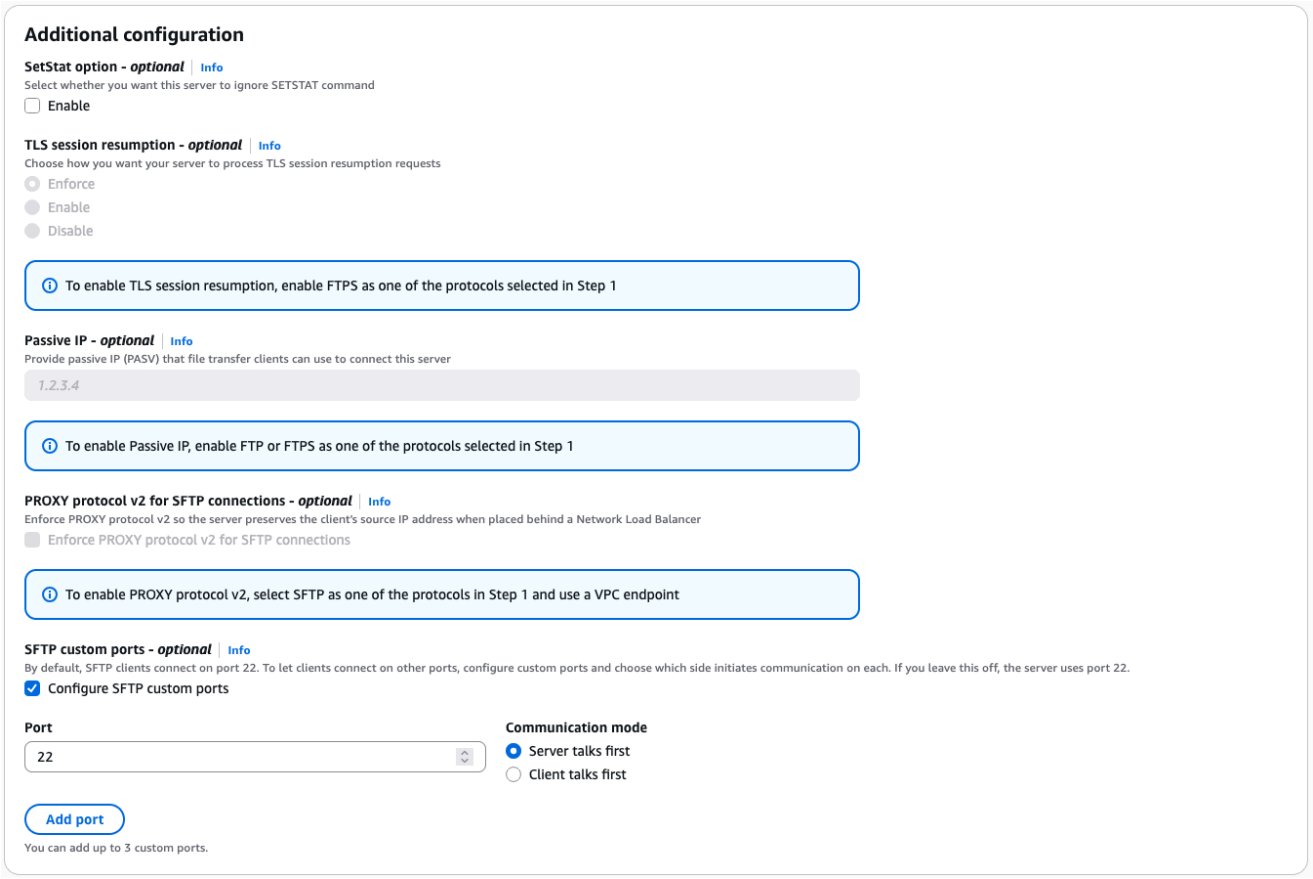
   When you create your server using the console, optimized directories is enabled by default. If you create your server using the API, this behavior is not enabled by default.



h. (Optional) Configure AWS Transfer Family servers to display customized messages such as organizational policies or terms and conditions to your end users. For **Display banner**, in the **Pre-authentication display banner** text box, enter the text message that you want to display to your users before they authenticate.

i. (Optional) You can configure the following additional options.

- **SetStat option**: enable this option to ignore the error that is generated when a client attempts to use `SETSTAT` on a file you are uploading to an Amazon S3 bucket. For additional details, see the `SetStatOption` documentation in the ProtocolDetails.
- **TLS session resumption**: this option is only available if you have enabled FTPS as one of the protocols for this server.
- **Passive IP**: this option is only available if you have enabled FTPS or FTP as one of the protocols for this server.
- **PROXY protocol configuration**: turn on this option to preserve the client's source IP address if your SFTP server sits behind a Network Load Balancer (NLB) with PROXY protocol v2 (PPv2) enabled. For more information, see Working with Network Load Balancers and `ProxyConfig` in the ProtocolDetails API reference.
- **SFTP custom ports**: For an SFTP server that uses a `PUBLIC` endpoint, you can choose the ports that the server listens on. Use the `SftpPorts` property to configure up to three ports. Each port can be any integer from 2000 to 65535, or 22. For each port, you can also set a `CommunicationMode` that controls which side sends data first. Use `CLIENT_TALK_FIRST` for clients that require TCP piggy-back ACKs. A piggy-back ACK carries data in the final ACK of the TCP three-way handshake. Use `SERVER_TALK_FIRST` if you are unsure of your clients' behavior or your clients do not depend on this. For more information, see the `SftpPorts` documentation in the ProtocolDetails topic.



8. In **Review and create**, review your choices.

   - If you want to edit any of them, choose **Edit** next to the step.

     > **Note**
     >
     > You must review each step after the step that you chose to edit.

   - If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations, but you'll need to create a user first. For details on creating users, see Managing users for server endpoints.

Create an FTPS-enabled server

File Transfer Protocol over SSL (FTPS) is an extension to FTP. It uses Transport Layer Security (TLS) and Secure Sockets Layer (SSL) cryptographic protocols to encrypt traffic. FTPS allows encryption of both the control and data channel connections either concurrently or independently.

> **Note**
>
> For important considerations about Network Load Balancers, see Avoid placing NLBs and NATs in front of AWS Transfer Family servers.

To create an FTPS-enabled server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/ and select **Servers** from the navigation pane, then choose **Create server**.
2. In **Choose protocols**, select **FTPS**.

   For **Server certificate**, choose a certificate stored in AWS Certificate Manager (ACM) which will be used to identify your server when clients connect to it over FTPS and then choose **Next**.

   To request a new public certificate, see Request a public certificate in the *AWS Certificate Manager User Guide*.

   To import an existing certificate into ACM, see Importing certificates into ACM in the *AWS Certificate Manager User Guide*.

   To request a private certificate to use FTPS through private IP addresses, see Requesting a Private Certificate in the *AWS Certificate Manager User Guide*.

   Certificates with the following cryptographic algorithms and key sizes are supported:

   - 2048-bit RSA (RSA_2048)
   - 4096-bit RSA (RSA_4096)
   - Elliptic Prime Curve 256 bit (EC_prime256v1)
   - Elliptic Prime Curve 384 bit (EC_secp384r1)
   - Elliptic Prime Curve 521 bit (EC_secp521r1)

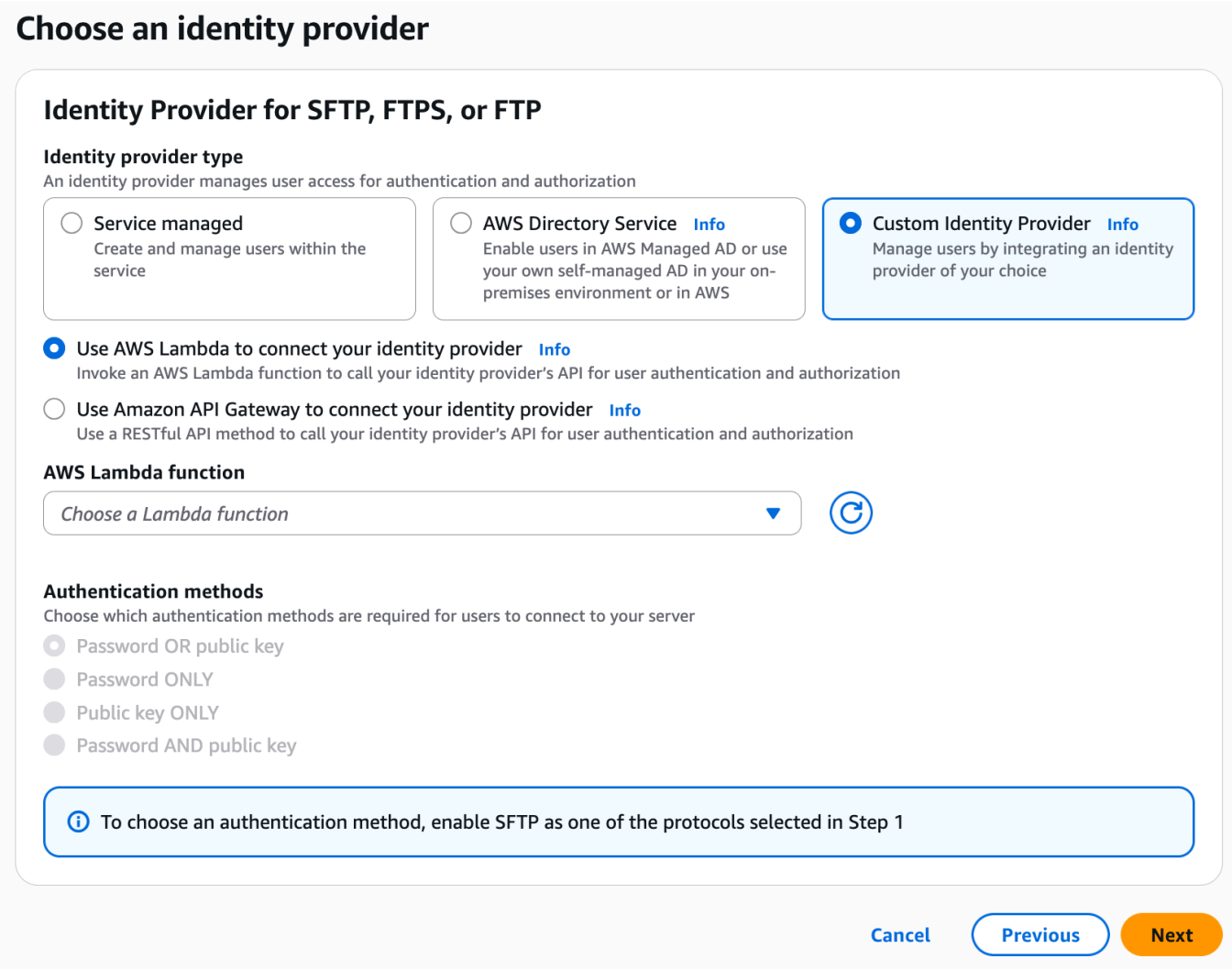
> **Note**
>
> The certificate must be a valid SSL/TLS X.509 version 3 certificate with FQDN or IP address specified and contain information about the issuer.

3. In **Choose an identity provider**, choose the identity provider that you want to use to manage user access. You have the following options:

   - **AWS Directory Service for Microsoft Active Directory** – You provide an Directory Service directory to access the endpoint. By doing so, you can use credentials stored in your Active Directory to authenticate your users. To learn more about working with AWS Managed Microsoft AD identity providers, see Using AWS Directory Service for Microsoft Active Directory.

     > **Note**
     >
     > - Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
     > - To set up a server with Directory Service as your identity provider, you need to add some Directory Service permissions. For details, see Before you start using AWS Directory Service for Microsoft Active Directory.

   - **Custom identity provider** – Choose either of the following options:
     - **Use AWS Lambda to connect your identity provider** – You can use an existing identity provider, backed by a Lambda function. You provide the name of the Lambda function. For more information, see Using AWS Lambda to integrate your identity provider.
     - **Use Amazon API Gateway to connect your identity provider** – You can create an API Gateway method backed by a Lambda function for use as an identity provider. You provide an Amazon API Gateway URL and an invocation role. For more information, see Using Amazon API Gateway to integrate your identity provider.



4. Choose **Next**.
5. In **Choose an endpoint**, do the following:

> **Note**
>
> FTPS servers for Transfer Family operate over Port 21 (Control Channel) and Port Range 8192–8200 (Data Channel).

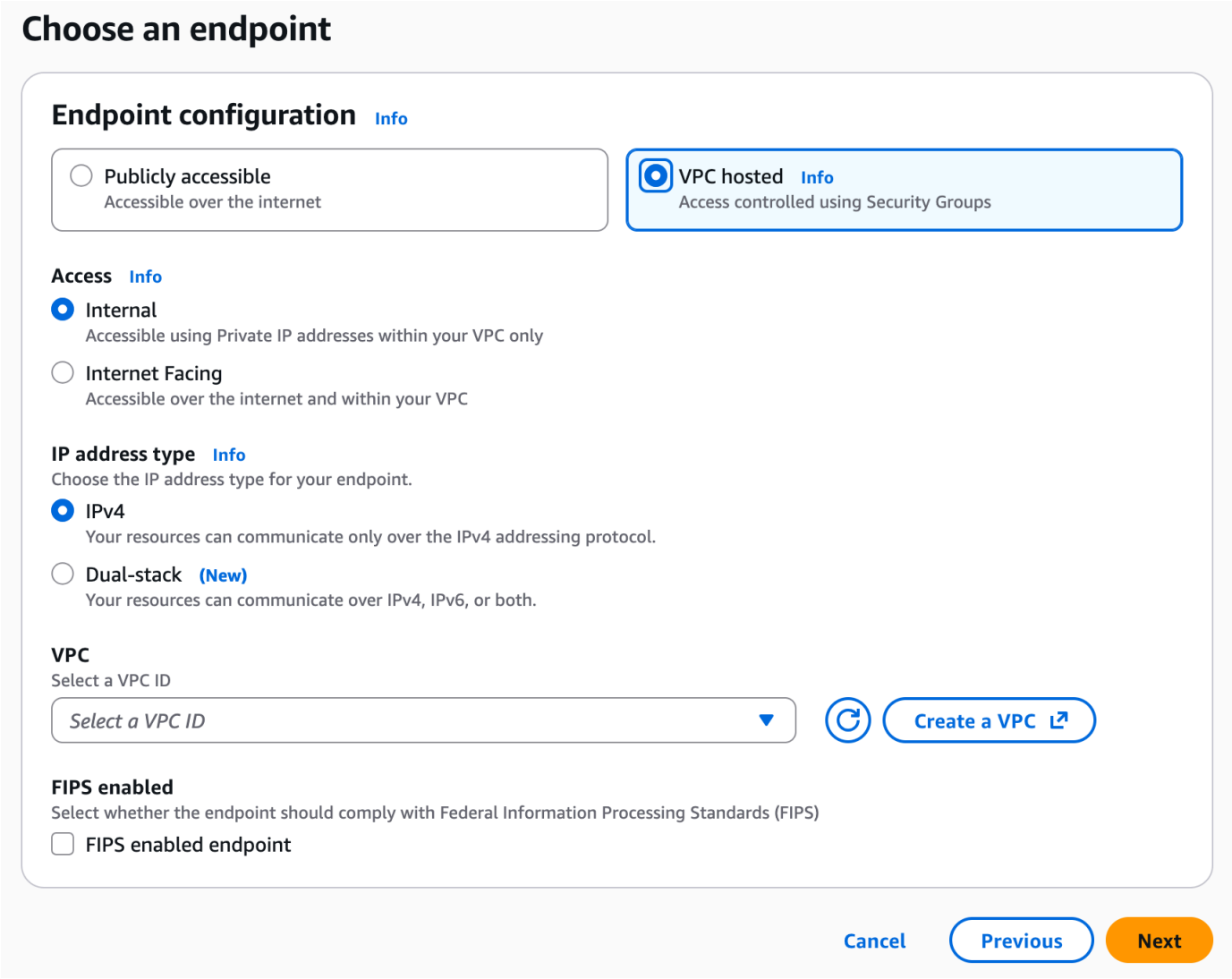
a. For **Endpoint type**, choose the **VPC hosted** endpoint type to host your server's endpoint. For information about setting up your VPC hosted endpoint, see Create a server in a virtual private cloud.

> **Note**
>
> Publicly accessible endpoints are not supported.

b. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure that the endpoint complies with Federal Information Processing Standards (FIPS).

> **Note**
>
> FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*. For more information about FIPS, see Federal Information Processing Standard (FIPS) 140-2 .

c. Choose **Next**.



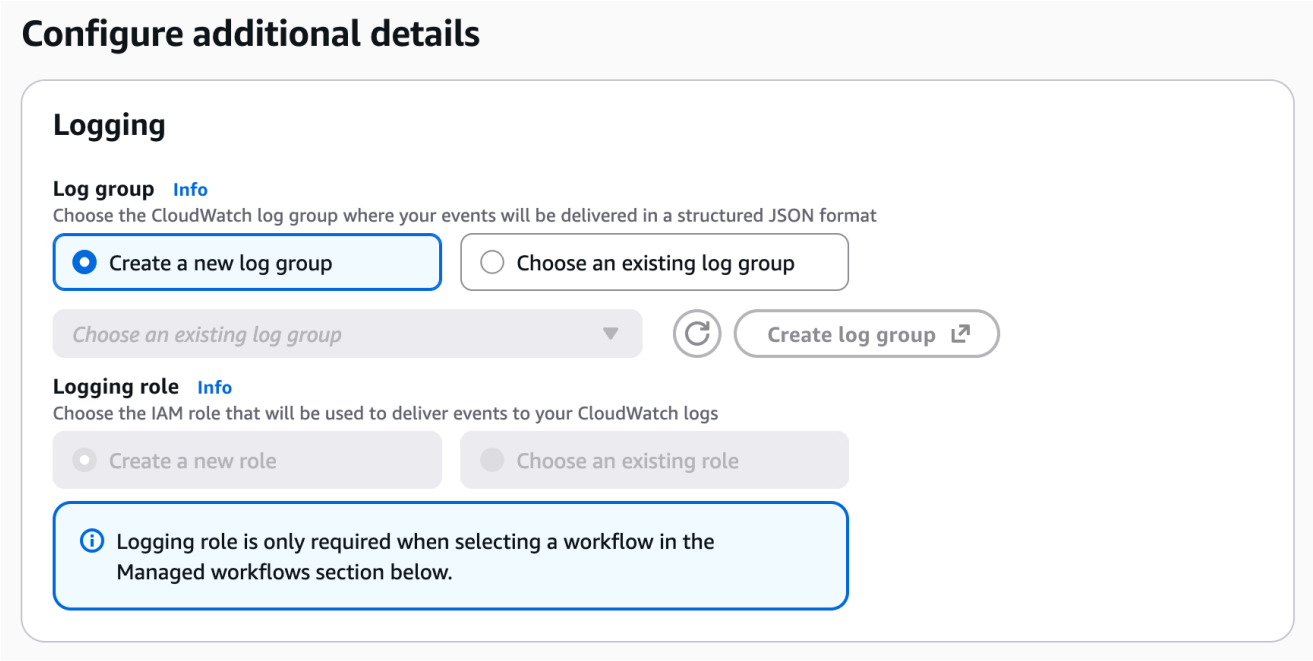
6. On the **Choose domain** page, choose the AWS storage service that you want to use to store and access your data over the selected protocol:

   - Choose **Amazon S3** to store and access your files as objects over the selected protocol.
   - Choose **Amazon EFS** to store and access your files in your Amazon EFS file system over the selected protocol.

   Choose **Next**.

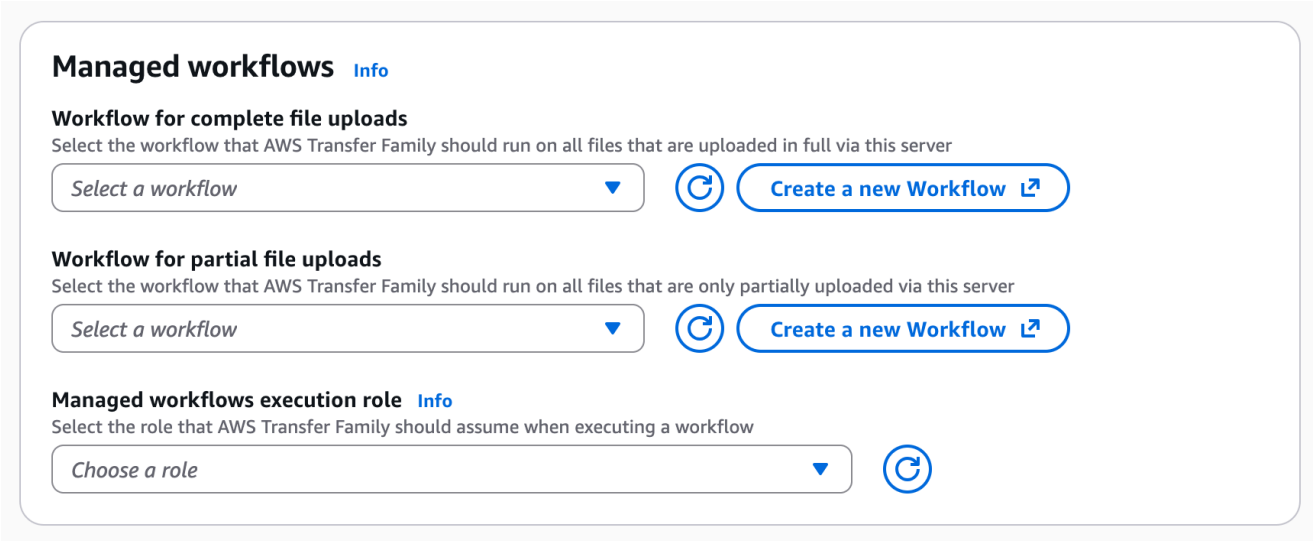
7. In **Configure additional details**, do the following:

   a. For logging, specify an existing log group or create a new one (the default option).



If you choose **Create log group**, the CloudWatch console (https://console.aws.amazon.com/cloudwatch/) opens to the **Create log group** page. For details, see Create a log group in CloudWatch Logs.

b. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial upload. To learn more about processing your files by using managed workflows, see AWS Transfer Family managed workflows.



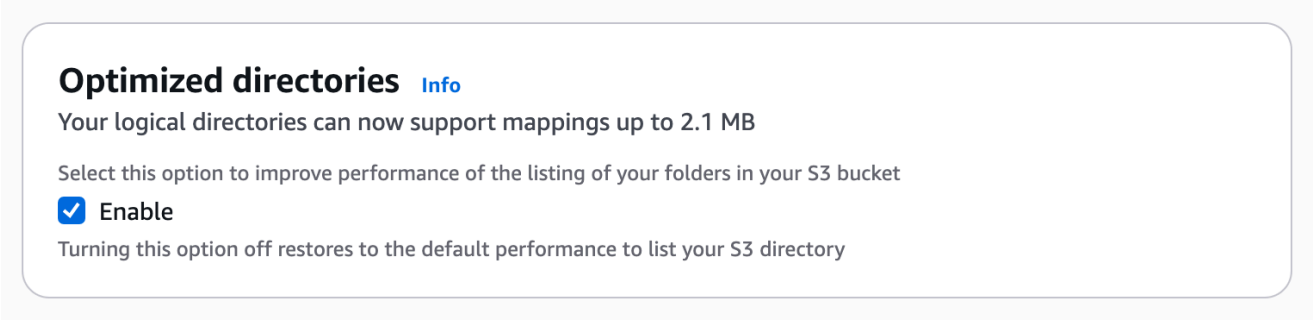
c. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server. Our latest security policy is the default: for details, see Security policies for AWS Transfer Family servers.

d. For **Server Host Key**, keep it blank.

e. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.

f. You can optimize performance for your Amazon S3 directories. For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds.

When you create your server using the console, optimized directories is enabled by default. If you create your server using the API, this behavior is not enabled by default.



g. Choose **Next**.

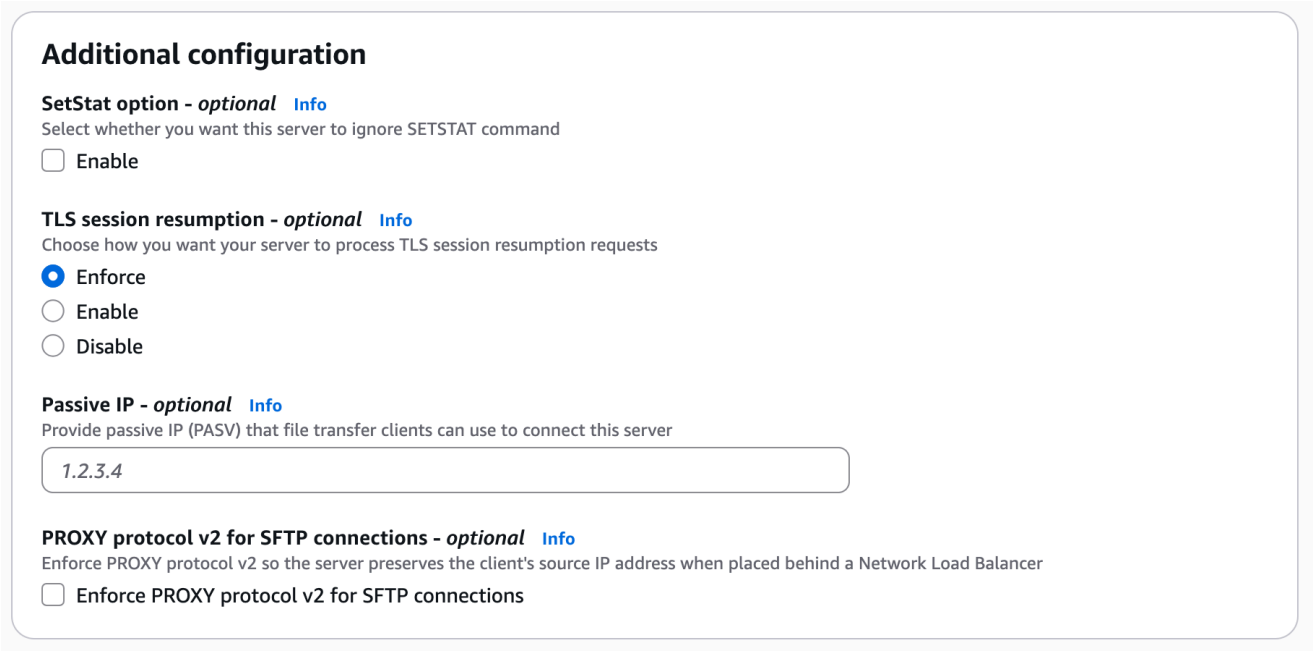
h. (Optional) You can configure AWS Transfer Family servers to display customized messages such as organizational policies or terms and conditions to your end users. You can also display customized Message of The Day (MOTD) to users who have successfully authenticated.

For **Display banner**, in the **Pre-authentication display banner** text box, enter the text message that you want to display to your users before they authenticate, and in the **Post-authentication display banner** text box, enter the text that you want to display to your users after they successfully authenticate.

i. (Optional) You can configure the following additional options.

- **SetStat option**: enable this option to ignore the error that is generated when a client attempts to use `SETSTAT` on a file you are uploading to an Amazon S3 bucket. For additional details, see the `SetStatOption` documentation in the ProtocolDetails topic.

- **TLS session resumption**: provides a mechanism to resume or share a negotiated secret key between the control and data connection for an FTPS session. For additional details, see the `TlsSessionResumptionMode` documentation in the ProtocolDetails topic.
- **Passive IP**: indicates passive mode, for FTP and FTPS protocols. Enter a single IPv4 address, such as the public IP address of a firewall, router, or load balancer. For additional details, see the `PassiveIp` documentation in the ProtocolDetails topic.
- **PROXY protocol configuration**: this option appears but is unavailable when SFTP is not one of the protocols for this server.



8. In **Review and create**, review your choices.

   - If you want to edit any of them, choose **Edit** next to the step.

     > **Note**
     >
     > You must review each step after the step that you chose to edit.

   - If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations for your users.

Next steps: For the next step, continue on to Working with custom identity providers to set up users.

Create an FTP-enabled server

File Transfer Protocol (FTP) is a network protocol used for the transfer of data. FTP uses a separate channel for control and data transfers. The control channel is open until terminated or inactivity timeout. The data channel is active for the duration of the transfer. FTP uses clear text and does not support encryption of traffic.

> **Note**
>
> When you enable FTP, you must choose the internal access option for the VPC-hosted endpoint. If you need your server to have data traverse the public network, you must use secure protocols, such as SFTP or FTPS.

> **Note**
>
> For important considerations about Network Load Balancers, see Avoid placing NLBs and NATs in front of AWS Transfer Family servers.

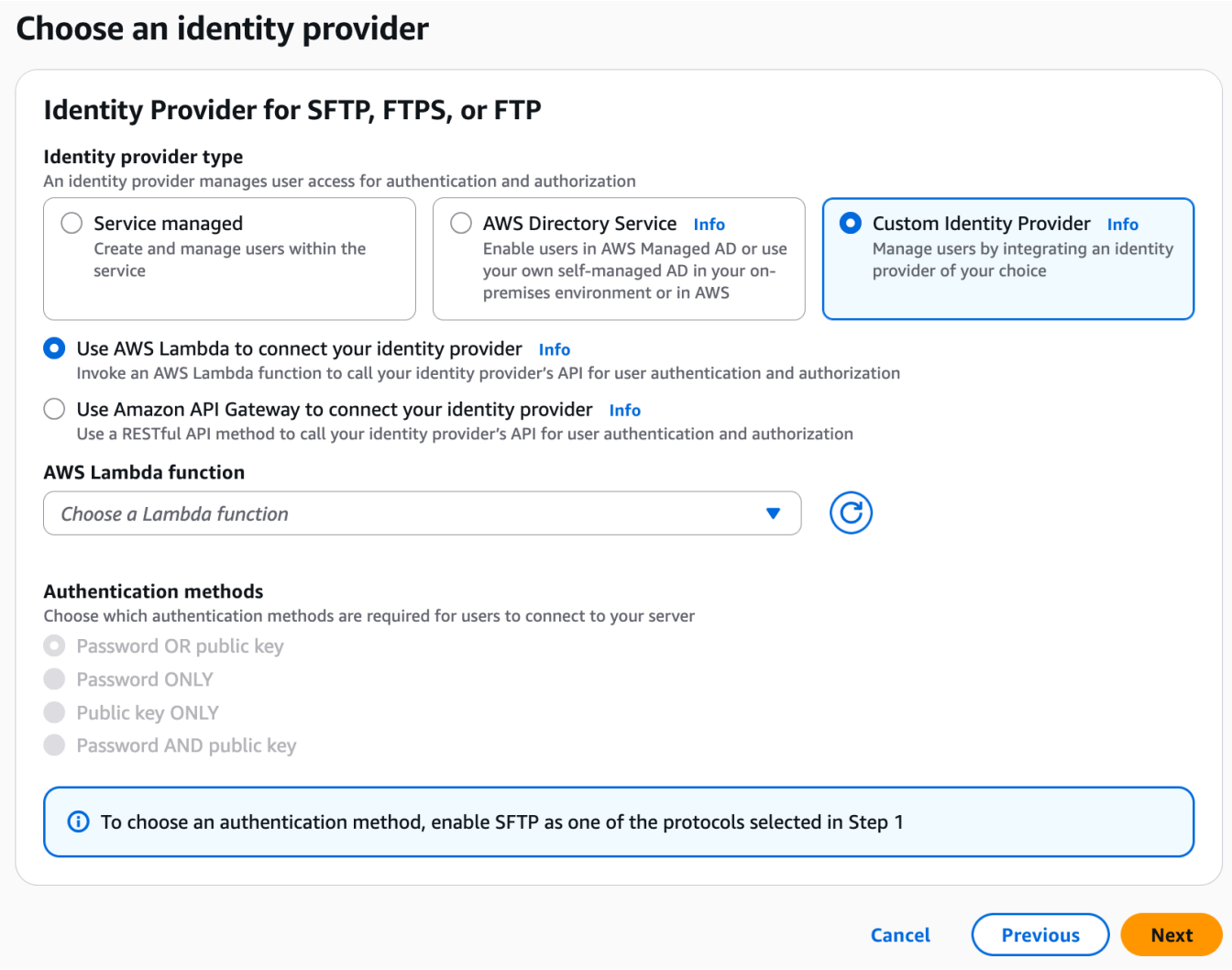
To create an FTP-enabled server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/ and select **Servers** from the navigation pane, then choose **Create server**.
2. In **Choose protocols**, select **FTP**, and then choose **Next**.
3. In **Choose an identity provider**, choose the identity provider that you want to use to manage user access. You have the following options:

   - **AWS Directory Service for Microsoft Active Directory** – You provide an Directory Service directory to access the endpoint. By doing so, you can use credentials stored in your Active Directory to authenticate your users. To learn more about working with AWS Managed Microsoft AD identity providers, see Using AWS Directory Service for Microsoft Active Directory.

> **Note**
>
> - Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
> - To set up a server with Directory Service as your identity provider, you need to add some Directory Service permissions. For details, see Before you start using AWS Directory Service for Microsoft Active Directory.

- **Custom identity provider** – Choose either of the following options:
  - **Use AWS Lambda to connect your identity provider** – You can use an existing identity provider, backed by a Lambda function. You provide the name of the Lambda function. For more information, see Using AWS Lambda to integrate your identity provider.
  - **Use Amazon API Gateway to connect your identity provider** – You can create an API Gateway method backed by a Lambda function for use as an identity provider. You provide an Amazon API Gateway URL and an invocation role. For more information, see Using Amazon API Gateway to integrate your identity provider.



4. Choose **Next**.
5. In **Choose an endpoint**, do the following:

> **Note**
>
> FTP servers for Transfer Family operate over Port 21 (Control Channel) and Port Range 8192–8200 (Data Channel).

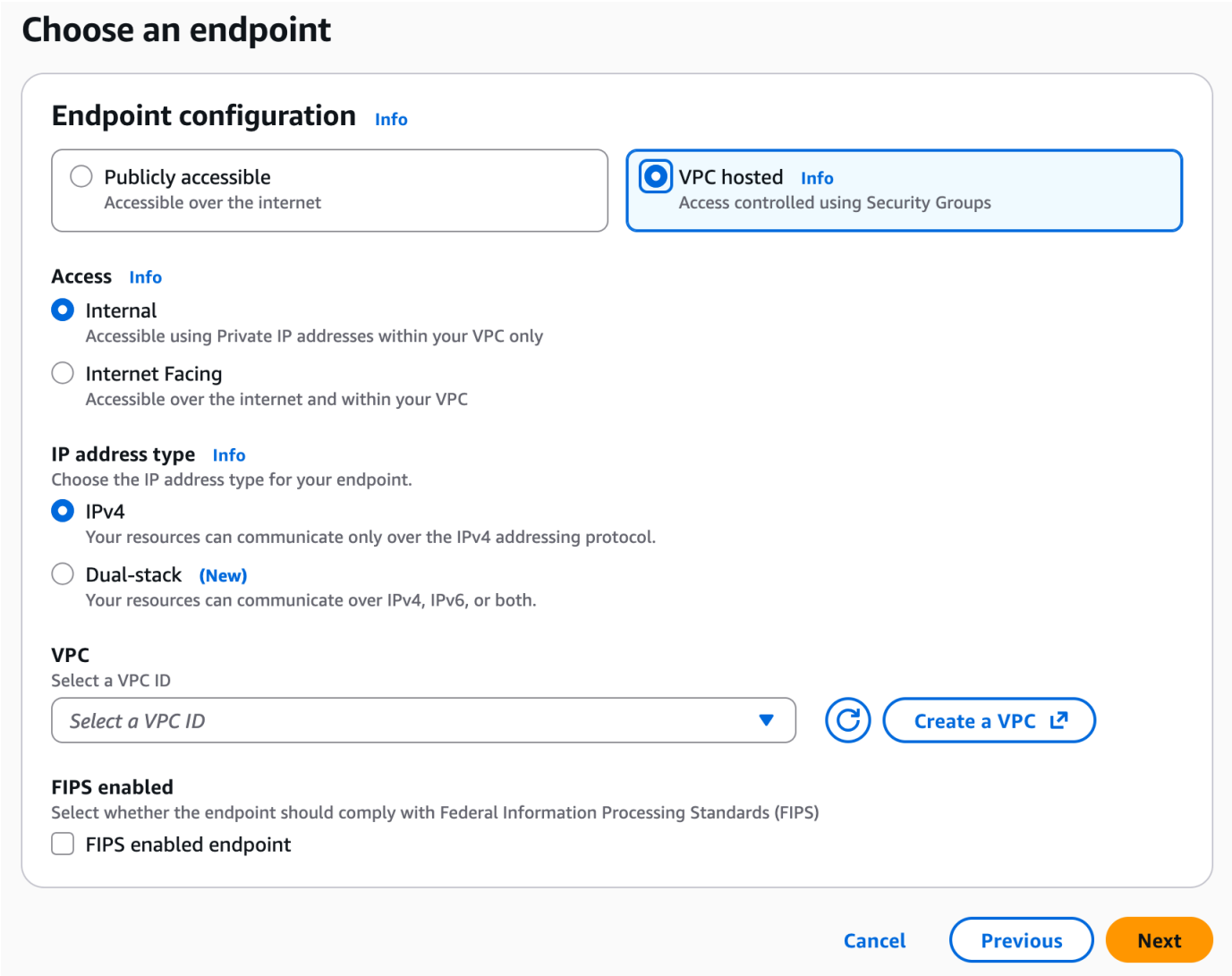
a. For **Endpoint type**, choose **VPC hosted** to host your server's endpoint. For information about setting up your VPC hosted endpoint, see Create a server in a virtual private cloud.

> **Note**
>
> Publicly accessible endpoints are not supported.

b. For **FIPS Enabled**, keep the **FIPS Enabled endpoint** check box cleared.

> **Note**
>
> FIPS-enabled endpoints are not supported for FTP servers.

c. Choose **Next**.



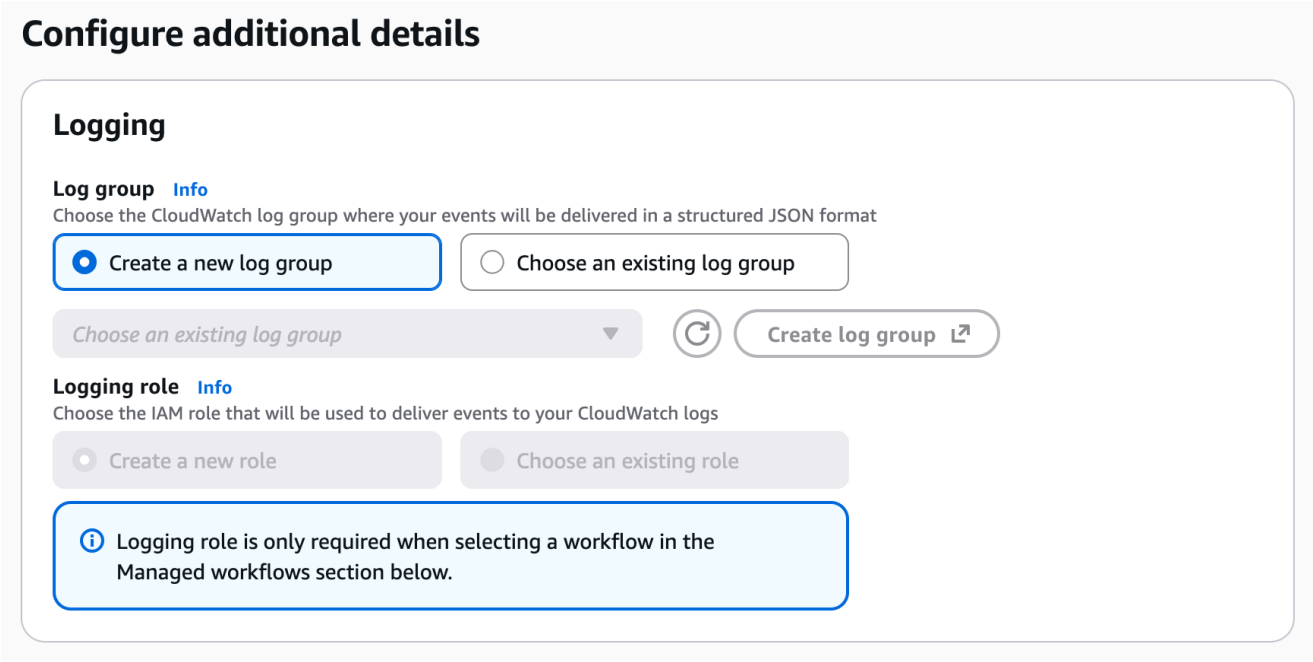
6. On the **Choose domain** page, choose the AWS storage service that you want to use to store and access your data over the selected protocol.

   - Choose **Amazon S3** to store and access your files as objects over the selected protocol.
   - Choose **Amazon EFS** to store and access your files in your Amazon EFS file system over the selected protocol.

   Choose **Next**.

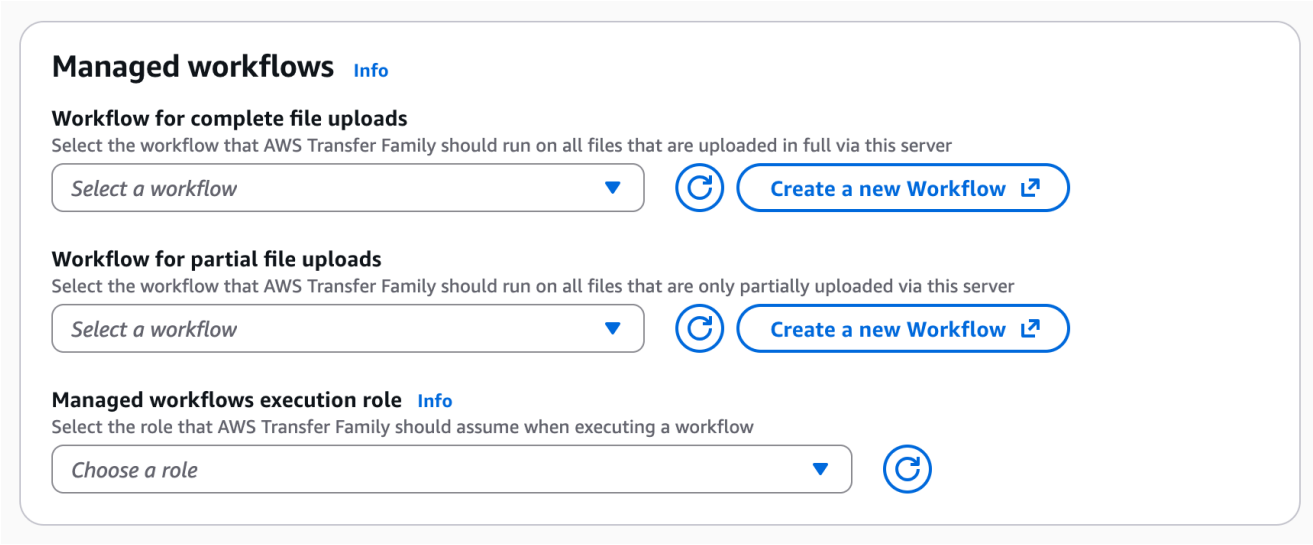
7. In **Configure additional details**, do the following:

   a. For logging, specify an existing log group or create a new one (the default option).



   If you choose **Create log group**, the CloudWatch console (https://console.aws.amazon.com/cloudwatch/) opens to the **Create log group** page. For details, see Create a log group in CloudWatch Logs.

   b. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial upload. To learn more about processing your files by using managed workflows, see AWS Transfer Family managed workflows.



c. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

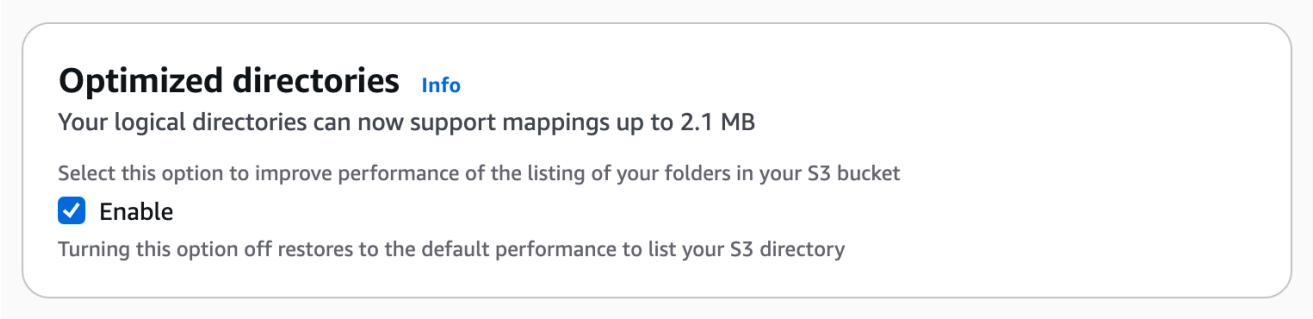
> **Note**
>
> Transfer Family assigns the latest security policy to your FTP server. However, since the FTP protocol doesn't use any encryption, FTP servers do not use any of the security policy algorithms. Unless your server also uses the FTPS or SFTP protocol, the security policy remains unused.

d. For **Server Host Key**, keep it blank.

e. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.

f. You can optimize performance for your Amazon S3 directories. For example, suppose that you go into your home directory, and you have 10,000 subdirectories. In other words, your Amazon S3 bucket has 10,000 folders. In this scenario, if you run the `ls` (list) command, the list operation takes between six and eight minutes. However, if you optimize your directories, this operation takes only a few seconds.

   When you create your server using the console, optimized directories is enabled by default. If you create your server using the API, this behavior is not enabled by default.



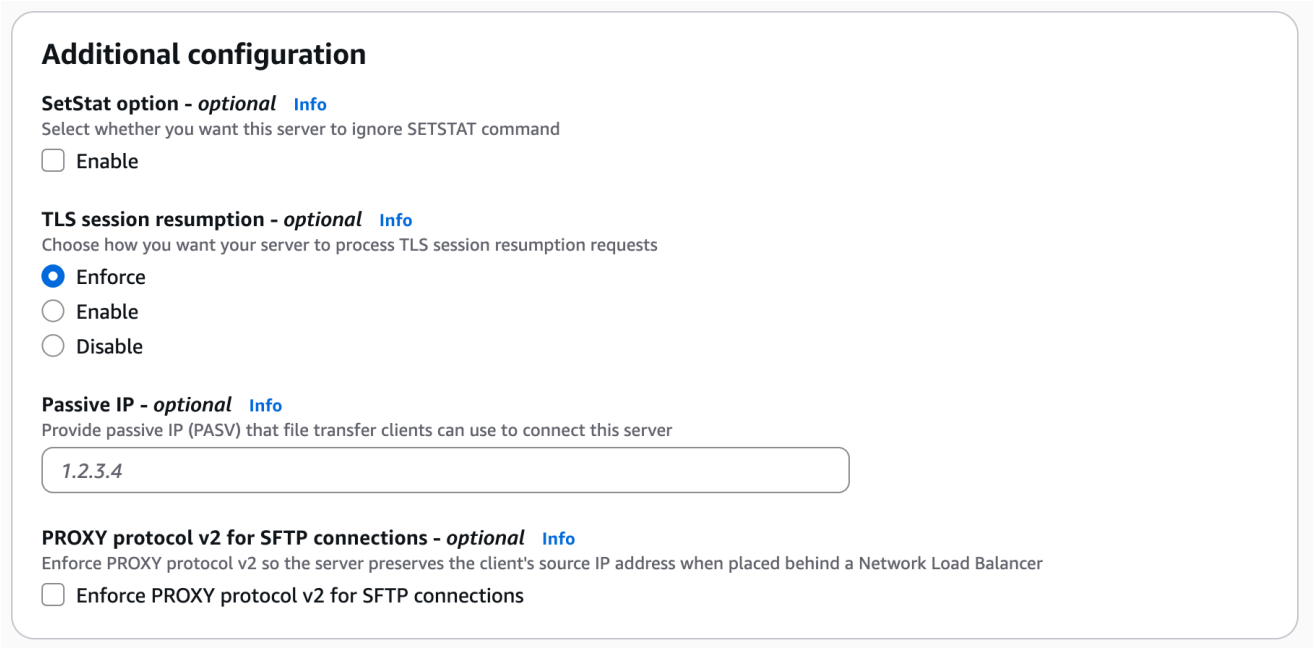
g. Choose **Next**.

h. (Optional) You can configure AWS Transfer Family servers to display customized messages such as organizational policies or terms and conditions to your end users. You can also display customized Message of The Day (MOTD) to users who have successfully authenticated.

   For **Display banner**, in the **Pre-authentication display banner** text box, enter the text message that you want to display to your users before they authenticate, and in the **Post-authentication display banner** text box, enter the text that you want to display to your users after they successfully authenticate.

i. (Optional) You can configure the following additional options.

   - **SetStat option**: enable this option to ignore the error that is generated when a client attempts to use `SETSTAT` on a file you are uploading to an Amazon S3 bucket. For additional details, see the `SetStatOption` documentation in the ProtocolDetails topic.
   - **TLS session resumption**: provides a mechanism to resume or share a negotiated secret key between the control and data connection for an FTPS session. For additional details, see the `TlsSessionResumptionMode` documentation in the ProtocolDetails topic.
   - **Passive IP**: indicates passive mode, for FTP and FTPS protocols. Enter a single IPv4 address, such as the public IP address of a firewall, router, or load balancer. For additional details, see the `PassiveIp` documentation in the ProtocolDetails topic.
   - **PROXY protocol configuration**: this option appears but is unavailable when SFTP is not one of the protocols for this server.



8. In **Review and create**, review your choices.

   - If you want to edit any of them, choose **Edit** next to the step.

     > **Note**
     >
     > You must review each step after the step that you chose to edit.

   - If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations for your users.

Next steps – For the next step, continue on to Working with custom identity providers to set up users.

Create a server in a virtual private cloud

You can host your server's endpoint inside a virtual private cloud (VPC) to use for transferring data to and from an Amazon S3 bucket or Amazon EFS file system without going over the public internet.

Note

After May 19, 2021, you won't be able to create a server using `EndpointType=VPC_ENDPOINT` in your AWS account if your account hasn't already done so before May 19, 2021. If you have already created servers with `EndpointType=VPC_ENDPOINT` in your AWS account on or before February 21, 2021, you will not be affected. After this date, use `EndpointType=`**`VPC`**. For more information, see the section called “Discontinuing the use of VPC_ENDPOINT”.

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and a server. You can then use this server to transfer data over your client to and from your Amazon S3 bucket without using public IP addressing or requiring an internet gateway.

Using Amazon VPC, you can launch AWS resources in a custom virtual network. You can use a VPC to control your network settings, such as the IP address range, subnets, route tables, and network gateways. For more information about VPCs, see What Is Amazon VPC? in the *Amazon VPC User Guide*.

In the next sections, find instructions on how to create and connect your VPC to a server. As an overview, you do this as follows:

1. Set up a server using a VPC endpoint.
2. Connect to your server using a client that is inside your VPC through the VPC endpoint. Doing this enables you to transfer data that is stored in your Amazon S3 bucket over your client using AWS Transfer Family. You can perform this transfer even though the network is disconnected from the public internet.
3. In addition, if you choose to make your server's endpoint internet-facing, you can associate Elastic IP addresses with your endpoint. Doing this lets clients outside of your VPC connect to your server. You can use VPC security groups to control access to authenticated users whose requests originate only from allowed addresses.

Note

AWS Transfer Family supports dual-stack endpoints, allowing your server to communicate over both IPv4 and IPv6. To enable dual-stack support, select the **Enable DNS dual-stack**

endpoint option when creating your VPC endpoint. Note that both your VPC and subnets must be configured to support IPv6 before you can use this feature. Dual-stack support is particularly useful when you have clients that need to connect using either protocol. For information about dual-stack (IPv4 and IPv6) server endpoints, see IPv6 support for Transfer Family servers.

Topics

- Create a server endpoint that can be accessed only within your VPC
- Create an internet-facing endpoint for your server
- Change the endpoint type for your server
- Discontinuing the use of VPC_ENDPOINT
- Limiting VPC endpoint access for Transfer Family servers
- Additional networking features
- Updating the AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC

Create a server endpoint that can be accessed only within your VPC

In the following procedure, you create a server endpoint that is accessible only to resources within your VPC.

To create a server endpoint inside a VPC

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. From the navigation pane, select **Servers**, then choose **Create server**.
3. In **Choose protocols**, select one or more protocols, and then choose **Next**. For more information about protocols, see Step 2: Create an SFTP-enabled server.
4. In **Choose an identity provider**, choose **Service managed** to store user identities and keys in AWS Transfer Family, and then choose **Next**.

   This procedure uses the service-managed option. If you choose **Custom**, you provide an Amazon API Gateway endpoint and an AWS Identity and Access Management (IAM) role to access the endpoint. By doing so, you can integrate your directory service to authenticate and authorize your users. To learn more about working with custom identity providers, see Working with custom identity providers.

5. In **Choose an endpoint**, do the following:

   a. For **Endpoint type**, choose the **VPC hosted** endpoint type to host your server's endpoint.

   b. For **Access**, choose **Internal** to make your endpoint only accessible to clients using the endpoint's private IP addresses.

      For details on the **Internet Facing** option, see Create an internet-facing endpoint for your server. A server that is created in a VPC for internal access only doesn't support custom hostnames.

   c. For **VPC**, choose an existing VPC ID or choose **Create a VPC** to create a new VPC.

   d. In the **Availability Zones** section, choose up to three Availability Zones and associated subnets.

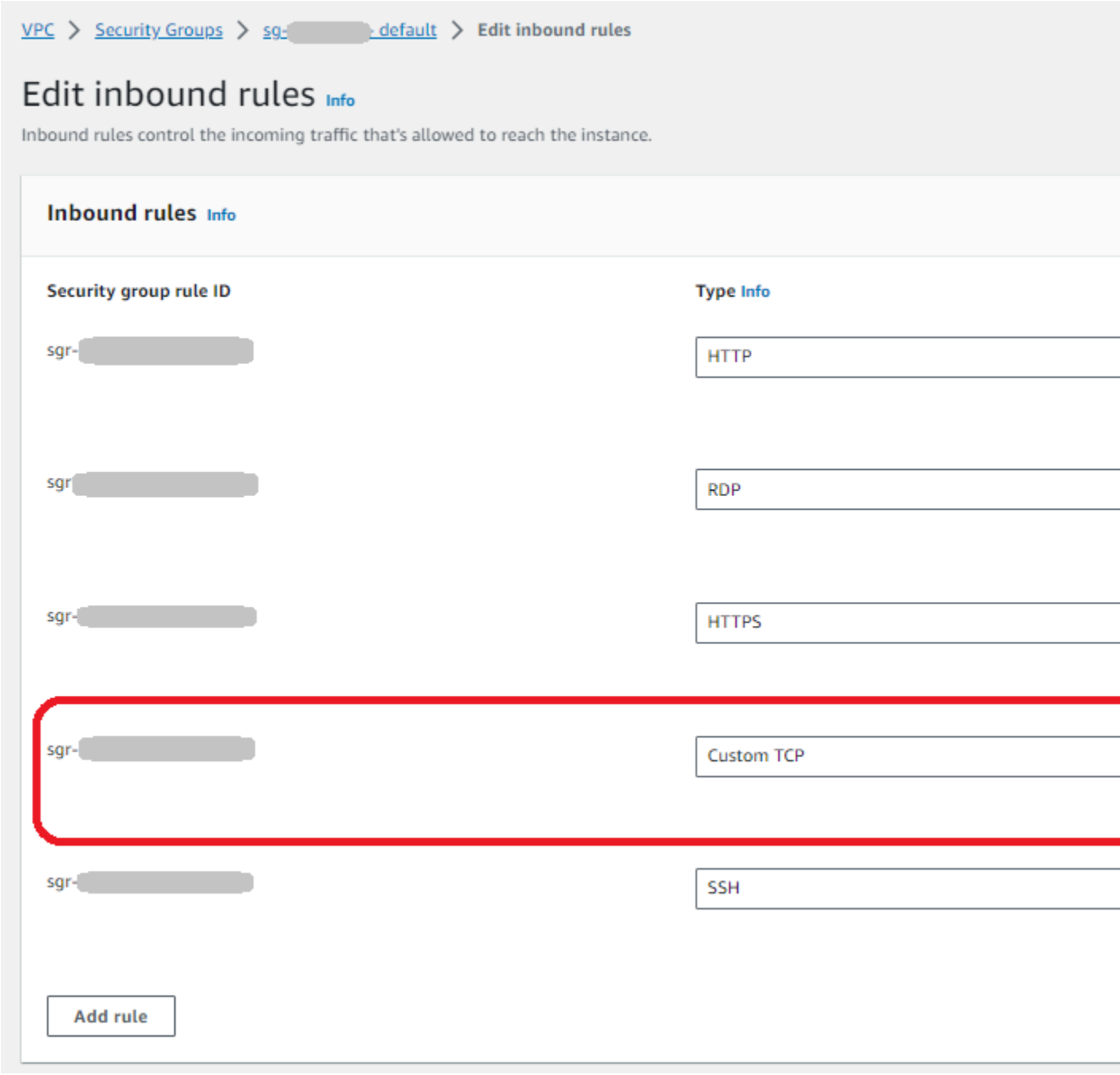
   e. In the **Security Groups** section, choose an existing security group ID or IDs or choose **Create a security group** to create a new security group. For more information about security groups, see Security groups for your VPC in the *Amazon Virtual Private Cloud User Guide*. To create a security group, see Creating a security group in the *Amazon Virtual Private Cloud User Guide*.

      > **Note**
      >
      > Your VPC automatically comes with a default security group. If you don't specify a different security group or groups when you launch the server, we associate the default security group with your server.

      - For the inbound rules for the security group, you can configure SSH traffic to use port 22, 2222, 22000, or any combination. Port 22 is configured by default. To use port 2222 or port 22000, you add an inbound rule to your security group. For the type, choose **Custom TCP**, then enter either **2222** or **22000** for **Port range**, and for the source, enter the same CIDR range that you have for your SSH port 22 rule.
      - For the inbound rules for the security group, configure FTP and FTPS traffic to use **Port range 21** for the control channel and **8192-8200** for the data channel.

Note

You can also use port 2223 for clients that require TCP "piggy-back" ACKs, or the ability for the final ack of the TCP 3-way handshake to also contain data.
Some client software may be incompatible with port 2223: for example, a client that requires the server to send the SFTP Identification String before the client does.



f. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure the endpoint complies with Federal Information Processing Standards (FIPS).

> **Note**
>
> FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see AWS Transfer Family endpoints and quotas in the *AWS*

General Reference. For more information about FIPS, see Federal Information Processing Standard (FIPS) 140-2 .

g. Choose **Next**.

6. In **Configure additional details**, do the following:

a. For **CloudWatch logging**, choose one of the following to enable Amazon CloudWatch logging of your user activity:

- **Create a new role** to allow Transfer Family to automatically create the IAM role, as long as you have the right permissions to create a new role. The IAM role that is created is called `AWSTransferLoggingAccess`.
- **Choose an existing role** to choose an existing IAM role from your account. Under **Logging role**, choose the role. This IAM role should include a trust policy with **Service** set to `transfer.amazonaws.com`.

  For more information about CloudWatch logging, see Configure CloudWatch logging role.

> **Note**
>
> - You can't view end-user activity in CloudWatch if you don't specify a logging role.
> - If you don't want to set up a CloudWatch logging role, select **Choose an existing role**, but don't select a logging role.

b. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

> **Note**
>
> By default, the `TransferSecurityPolicy-2024-01` security policy is attached to your server unless you choose a different one.

For more information about security policies, see Security policies for AWS Transfer Family servers.

c. (Optional: this section is only for migrating users from an existing SFTP-enabled server.) For **Server Host Key**, enter an RSA, ED25519, or ECDSA private key that will be used to identify your server when clients connect to it over SFTP.

d. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.

e. Choose **Next**.

7. In **Review and create**, review your choices. If you:

- Want to edit any of them, choose **Edit** next to the step.

> **Note**
>
> You will need to review each step after the step that you chose to edit.

- Have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

It can take a couple of minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations, but you'll need to create a user first. For details on creating users, see Managing users for server endpoints.

Create an internet-facing endpoint for your server

In the following procedure, you create a server endpoint. This endpoint is accessible over the internet only to clients whose source IP addresses are allowed in your VPC's default security group. Additionally, by using Elastic IP addresses to make your endpoint internet-facing, your clients can use the Elastic IP address to allow access to your endpoint in their firewalls.

> **Note**
>
> Only SFTP and FTPS can be used on an internet-facing VPC hosted endpoint.

To create an internet-facing endpoint

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. From the navigation pane, select **Servers**, then choose **Create server**.

3. In **Choose protocols**, select one or more protocols, and then choose **Next**. For more information about protocols, see Step 2: Create an SFTP-enabled server.
4. In **Choose an identity provider**, choose **Service managed** to store user identities and keys in AWS Transfer Family, and then choose **Next**.

   This procedure uses the service-managed option. If you choose **Custom**, you provide an Amazon API Gateway endpoint and an AWS Identity and Access Management (IAM) role to access the endpoint. By doing so, you can integrate your directory service to authenticate and authorize your users. To learn more about working with custom identity providers, see Working with custom identity providers.
5. In **Choose an endpoint**, do the following:

   a. For **Endpoint type**, choose the **VPC hosted** endpoint type to host your server's endpoint.

   b. For **Access**, choose **Internet Facing** to make your endpoint accessible to clients over the internet.

   > **Note**
   >
   > When you choose **Internet Facing**, you can choose an existing Elastic IP address in each subnet or subnets. Or you can go to the VPC console (https://console.aws.amazon.com/vpc/) to allocate one or more new Elastic IP addresses. These addresses can be owned either by AWS or by you. You can't associate Elastic IP addresses that are already in use with your endpoint.

   c. (Optional) For **Custom hostname**, choose one of the following:

   > **Note**
   >
   > Customers in AWS GovCloud (US) need to connect via the Elastic IP address directly, or create a hostname record within Commercial Route 53 that points to their EIP. For more information about using Route 53 for GovCloud endpoints, see Setting up Amazon Route 53 with your AWS GovCloud (US) resources in the *AWS GovCloud (US) User Guide*.

   - **Amazon Route 53 DNS alias** – if the hostname that you want to use is registered with Route 53. You can then enter the hostname.

- **Other DNS** – if the hostname that you want to use is registered with another DNS provider. You can then enter the hostname.
- **None** – to use the server's endpoint and not use a custom hostname. The server hostname takes the form `server-id.server.transfer.region.amazonaws.com`.

> **Note**
>
> For customers in AWS GovCloud (US), selecting **None** does not create a hostname in this format.

To learn more about working with custom hostnames, see Working with custom hostnames.

d. For **VPC**, choose an existing VPC ID or choose **Create a VPC** to create a new VPC.

e. In the **Availability Zones** section, choose up to three Availability Zones and associated subnets. For **IPv4 Addresses**, choose an **Elastic IP address** for each subnet. This is the IP address that your clients can use to allow access to your endpoint in their firewalls.

Tip: You must use a public subnet for your Availability Zones, or first setup an internet gateway if you want to use a private subnet.

f. In the **Security Groups** section, choose an existing security group ID or IDs or choose **Create a security group** to create a new security group. For more information about security groups, see Security groups for your VPC in the *Amazon Virtual Private Cloud User Guide*. To create a security group, see Creating a security group in the *Amazon Virtual Private Cloud User Guide*.

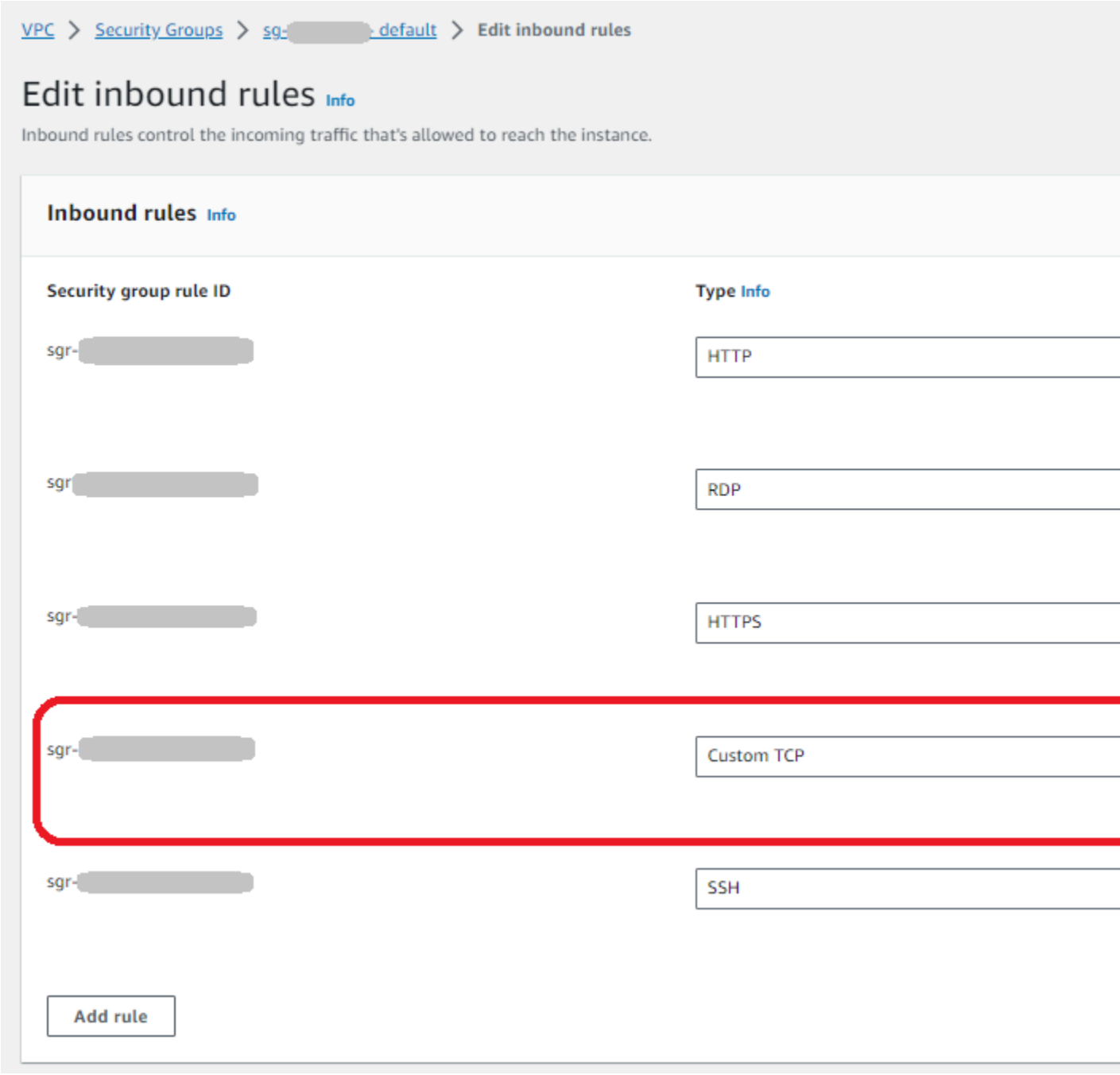
> **Note**
>
> Your VPC automatically comes with a default security group. If you don't specify a different security group or groups when you launch the server, we associate the default security group with your server.

- For the inbound rules for the security group, you can configure SSH traffic to use port 22, 2222, 22000, or any combination. Port 22 is configured by default. To use port 2222 or port 22000, you add an inbound rule to your security group. For the type, choose

Custom TCP, then enter either **2222** or **22000** for **Port range**, and for the source, enter the same CIDR range that you have for your SSH port 22 rule.

- For the inbound rules for the security group, configure FTPS traffic to use **Port range 21** for the control channel and **8192-8200** for the data channel.

> **Note**
>
> You can also use port 2223 for clients that require TCP "piggy-back" ACKs, or the ability for the final ack of the TCP 3-way handshake to also contain data.
> Some client software may be incompatible with port 2223: for example, a client that requires the server to send the SFTP Identification String before the client does.



g. (Optional) For **FIPS Enabled**, select the **FIPS Enabled endpoint** check box to ensure the endpoint complies with Federal Information Processing Standards (FIPS).

> **Note**
>
> FIPS-enabled endpoints are only available in North American AWS Regions. For available Regions, see AWS Transfer Family endpoints and quotas in the *AWS*

General Reference. For more information about FIPS, see Federal Information Processing Standard (FIPS) 140-2 .

h. Choose **Next**.

6. In **Configure additional details**, do the following:

a. For **CloudWatch logging**, choose one of the following to enable Amazon CloudWatch logging of your user activity:

- **Create a new role** to allow Transfer Family to automatically create the IAM role, as long as you have the right permissions to create a new role. The IAM role that is created is called `AWSTransferLoggingAccess`.
- **Choose an existing role** to choose an existing IAM role from your account. Under **Logging role**, choose the role. This IAM role should include a trust policy with **Service** set to `transfer.amazonaws.com`.

  For more information about CloudWatch logging, see Configure CloudWatch logging role.

> **Note**
>
> - You can't view end-user activity in CloudWatch if you don't specify a logging role.
> - If you don't want to set up a CloudWatch logging role, select **Choose an existing role**, but don't select a logging role.

b. For **Cryptographic algorithm options**, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

> **Note**
>
> By default, the `TransferSecurityPolicy-2024-01` security policy is attached to your server unless you choose a different one.

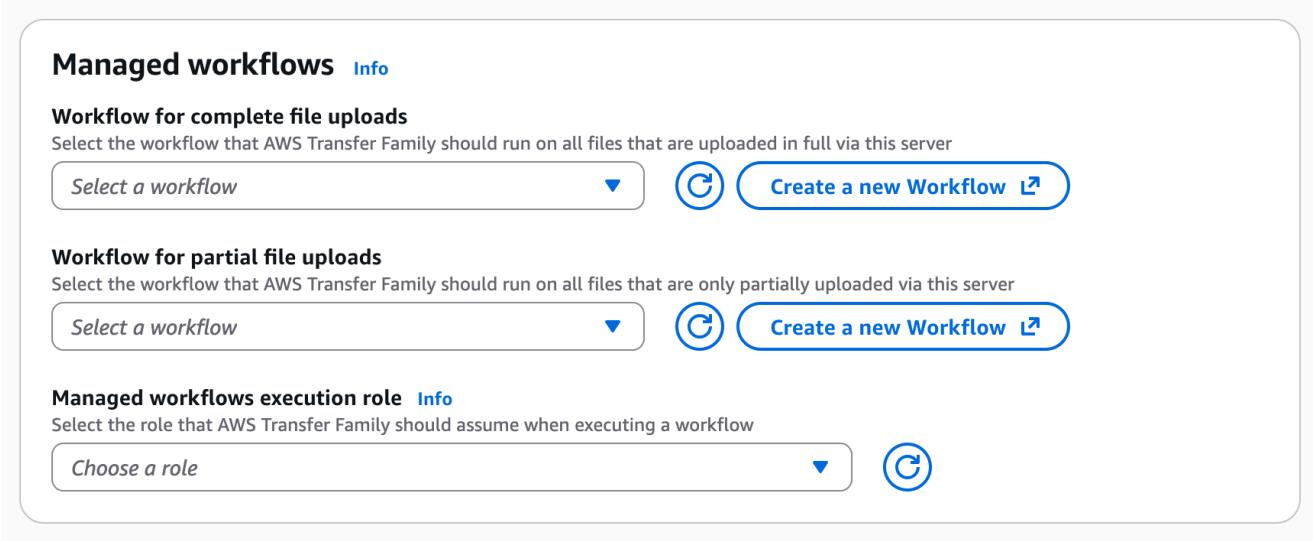
For more information about security policies, see Security policies for AWS Transfer Family servers.

c. (Optional: this section is only for migrating users from an existing SFTP-enabled server.) For **Server Host Key**, enter an RSA, ED25519, or ECDSA private key that will be used to identify your server when clients connect to it over SFTP.

d. (Optional) For **Tags**, for **Key** and **Value**, enter one or more tags as key-value pairs, and then choose **Add tag**.

e. Choose **Next**.

f. (Optional) For **Managed workflows**, choose workflow IDs (and a corresponding role) that Transfer Family should assume when executing the workflow. You can choose one workflow to execute upon a complete upload, and another to execute upon a partial upload. To learn more about processing your files by using managed workflows, see AWS Transfer Family managed workflows.



7. In **Review and create**, review your choices. If you:

- Want to edit any of them, choose **Edit** next to the step.

> **Note**
>
> You will need to review each step after the step that you chose to edit.

- Have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

You can choose the server ID to see the detailed settings of the server that you just created. After the column **Public IPv4 address** has been populated, the Elastic IP addresses that you provided are successfully associated with your server's endpoint.

> **Note**
>
> When your server in a VPC is online, only the subnets can be modified and only through the UpdateServer API. You must stop the server to add or change the server endpoint's Elastic IP addresses.

Change the endpoint type for your server

If you have an existing server that is accessible over the internet (that is, has a public endpoint type), you can change its endpoint to a VPC endpoint.

> **Note**
>
> If you have an existing server in a VPC displayed as `VPC_ENDPOINT`, we recommend that you modify it to the new VPC endpoint type. With this new endpoint type, you no longer need to use a Network Load Balancer (NLB) to associate Elastic IP addresses with your server's endpoint. Also, you can use VPC security groups to restrict access to your server's endpoint. However, you can continue to use the `VPC_ENDPOINT` endpoint type as needed.

The following procedure assumes that you have a server that uses either the current public endpoint type or the older `VPC_ENDPOINT` type.

To change the endpoint type for your server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that you want to change the endpoint type for.

> **Important**
>
> You must stop the server before you can change its endpoint.

4. For **Actions**, choose **Stop**.
5. In the confirmation dialog box that appears, choose **Stop** to confirm that you want to stop the server.

> **Note**
>
> Before proceeding to the next step, in **Endpoint details**, wait for the **Status** of the server to change to **Offline**; this can take a couple of minutes. You might have to choose **Refresh** on the **Servers** page to see the status change.
> You won't be able to make any edits until the server is **Offline**.

6. In **Endpoint details**, choose **Edit**.
7. In **Edit endpoint configuration**, do the following:

   a. For **Edit endpoint type**, choose **VPC hosted**.

   b. For **Access**, choose one of the following:

   - **Internal** to make your endpoint only accessible to clients using the endpoint's private IP addresses.
   - **Internet Facing** to make your endpoint accessible to clients over the public internet.

   > **Note**
   >
   > When you choose **Internet Facing**, you can choose an existing Elastic IP address in each subnet or subnets. Or, you can go to the VPC console (https://console.aws.amazon.com/vpc/) to allocate one or more new Elastic IP addresses. These addresses can be owned either by AWS or by you. You can't associate Elastic IP addresses that are already in use with your endpoint.

   c. (Optional for internet facing access only) For **Custom hostname**, choose one of the following:

   - **Amazon Route 53 DNS alias** – if the hostname that you want to use is registered with Route 53. You can then enter the hostname.
   - **Other DNS** – if the hostname that you want to use is registered with another DNS provider. You can then enter the hostname.

- **None** – to use the server's endpoint and not use a custom hostname. The server hostname takes the form *serverId*.server.transfer.*regionId*.amazonaws.com.

  To learn more about working with custom hostnames, see Working with custom hostnames.

d. For **VPC**, choose an existing VPC ID, or choose **Create a VPC** to create a new VPC.

e. In the **Availability Zones** section, select up to three Availability Zones and associated subnets. If **Internet Facing** is chosen, also choose an Elastic IP address for each subnet.

> **Note**
>
> If you want the maximum of three Availability Zones, but there are not enough available, create them in the VPC console (https://console.aws.amazon.com/vpc/). If you modify the subnets or Elastic IP addresses, the server takes a few minutes to update. You can't save your changes until the server update is complete.

f. Choose **Save**.

8. For **Actions**, choose **Start** and wait for the status of the server to change to **Online**; this can take a couple of minutes.

> **Note**
>
> If you changed a public endpoint type to a VPC endpoint type, notice that **Endpoint type** for your server has changed to **VPC**.

The default security group is attached to the endpoint. To change or add additional security groups, see Creating Security Groups.

Discontinuing the use of VPC_ENDPOINT

AWS Transfer Family has discontinued the ability to create servers with `EndpointType=VPC_ENDPOINT` for new AWS accounts. As of May 19, 2021, AWS accounts that don't own AWS Transfer Family servers with an endpoint type of `VPC_ENDPOINT` will not be able to create new servers with `EndpointType=VPC_ENDPOINT`. If you already own servers that use the `VPC_ENDPOINT` endpoint type, we recommend that you start using `EndpointType=VPC` as

soon as possible. For details, see Update your AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC.

We launched the new VPC endpoint type earlier in 2020. For more information, see AWS Transfer Family for SFTP supports VPC Security Groups and Elastic IP addresses. This new endpoint is more feature rich and cost effective and there are no PrivateLink charges. For more information, see AWS PrivateLink pricing.

This endpoint type is functionally equivalent to the previous endpoint type (`VPC_ENDPOINT`). You can attach Elastic IP addresses directly to the endpoint to make it internet facing and use security groups for source IP filtering. For more information, see the Use IP allow listing to secure your AWS Transfer Family for SFTP servers blog post.

You can also host this endpoint in a shared VPC environment. For more information, see AWS Transfer Family now supports shared services VPC environments.

In addition to SFTP, you can use the VPC `EndpointType` to enable FTPS and FTP. We don't plan to add these features and FTPS/FTP support to `EndpointType=VPC_ENDPOINT`. We have also removed this endpoint type as an option from the AWS Transfer Family console.

You can change the endpoint type for your server using the Transfer Family console, AWS CLI, API, SDKs, or CloudFormation. To change your server's endpoint type, see Updating the AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC.

If you have any questions, contact AWS Support or your AWS account team.

> **Note**
>
> We do not plan to add these features and FTPS or FTP support to EndpointType=VPC_ENDPOINT. We are no longer offering it as an option on the AWS Transfer Family Console.

If you have additional questions, you can contact us through AWS Support or your account team.

Limiting VPC endpoint access for Transfer Family servers

When creating an AWS Transfer Family server with VPC endpoint type, your IAM users and principals need permissions to create and delete VPC endpoints. However, your organization's security policies may restrict these permissions. You can use IAM policies to allow VPC endpoint

creation and deletion specifically for Transfer Family while maintaining restrictions for other services.

> **Important**
>
> The following IAM policy allows users to create and delete VPC endpoints only for Transfer Family servers while denying these operations for other services:

```
{
    "Effect": "Deny",
    "Action": [
        "ec2:CreateVpcEndpoint",
        "ec2:DeleteVpcEndpoints"
    ],
    "Resource": ["*"],
    "Condition": {
        "ForAnyValue:StringNotLike": {
            "ec2:VpceServiceName": [
                "com.amazonaws.INPUT-YOUR-REGION.transfer.server.*"
            ]
        },
        "StringNotLike": {
            "aws:PrincipalArn": [
                "arn:aws:iam::*:role/INPUT-YOUR-ROLE"
            ]
        }
    }
}
```

Replace *INPUT-YOUR-REGION* with your AWS Region (for example, **us-east-1**) and *INPUT-YOUR-ROLE* with the IAM role you want to grant these permissions to.

Additional networking features

AWS Transfer Family provides several advanced networking features that enhance security and flexibility when using VPC configurations:

- **Shared VPC environment support** - You can host your Transfer Family server endpoint in a shared VPC environment. For more information, see Using VPC hosted endpoints in shared VPCs with AWS Transfer Family.

- **Authentication and security** - You can use an AWS Web Application Firewall to protect your Amazon API Gateway endpoint. For more information, see Securing AWS Transfer Family with AWS Web Application Firewall and Amazon API Gateway.

Updating the AWS Transfer Family server endpoint type from VPC_ENDPOINT to VPC

You can use the AWS Management Console, CloudFormation, or the Transfer Family API to update a server's `EndpointType` from `VPC_ENDPOINT` to `VPC`. Detailed procedures and examples for using each of these methods to update a server endpoint type are provided in the following sections. If you have servers in multiple AWS regions and in multiple AWS accounts, you can use the example script provided in the following section, with modifications, to identify servers using the `VPC_ENDPOINT` type that you will need to update.

Topics

- Identifying servers using the VPC_ENDPOINT endpoint type
- Updating the server endpoint type using the AWS Management Console
- Updating the server endpoint type using CloudFormation
- Updating the server EndpointType using the API

Identifying servers using the `VPC_ENDPOINT` endpoint type

You can identify which servers are using the `VPC_ENDPOINT` using the AWS Management Console.

To identify servers using the `VPC_ENDPOINT` endpoint type using the console

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. Choose **Servers** in the navigation pane to display the list of servers in your account in that region.
3. Sort the list of servers by the **Endpoint type** to see all servers using `VPC_ENDPOINT`.

To identify servers using `VPC_ENDPOINT` across multiple AWS Regions and accounts

If you have servers in multiple AWS regions and in multiple AWS accounts, you can use the following example script, with modifications, to identify servers using the `VPC_ENDPOINT` endpoint type. The example script uses the Amazon EC2 DescribeRegions and the Transfer Family

ListServers API operations. If you have many AWS accounts, you could loop through your accounts using an IAM Role with read only auditor access if you authenticate using session profiles to your identity provider.

1. Following is a simple example.

```
import boto3

profile = input("Enter the name of the AWS account you'll be working in: ")
session = boto3.Session(profile_name=profile)

ec2 = session.client("ec2")

regions = ec2.describe_regions()

for region in regions['Regions']:
    region_name = region['RegionName']
    if region_name=='ap-northeast-3': #https://github.com/boto/boto3/issues/1943
        continue
    transfer = session.client("transfer", region_name=region_name)
    servers = transfer.list_servers()
    for server in servers['Servers']:
       if server['EndpointType']=='VPC_ENDPOINT':
           print(server['ServerId'], region_name)
```

2. After you have the list of the servers to update, you can use one of the methods described in the following sections to update the `EndpointType` to `VPC`.

Updating the server endpoint type using the AWS Management Console

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that you want to change the endpoint type for.

> **Important**
>
> You must stop the server before you can change its endpoint.

4. For **Actions**, choose **Stop**.

5. In the confirmation dialog box that appears, choose **Stop** to confirm that you want to stop the server.

   > **Note**
   >
   > Before proceeding to the next step, wait for the **Status** of the server to change to **Offline**; this can take a couple of minutes. You might have to choose **Refresh** on the **Servers** page to see the status change.

6. After the status changes to **Offline**, choose the server to display the server details page.
7. In the **Endpoint details** section, choose **Edit**.
8. Choose **VPC hosted** for the **Endpoint type**.
9. Choose **Save**
10. For **Actions**, choose **Start** and wait for the status of the server to change to **Online**; this can take a couple of minutes.

Updating the server endpoint type using CloudFormation

This section describes how to use CloudFormation to update a server's `EndpointType` to VPC. Use this procedure for Transfer Family servers that you have deployed using CloudFormation. In this example, the original CloudFormation template used to deploy the Transfer Family server is shown as follows:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Create AWS Transfer Server with VPC_ENDPOINT endpoint type'
Parameters:
  SecurityGroupId:
    Type: AWS::EC2::SecurityGroup::Id
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Type: AWS::EC2::VPC::Id
Resources:
  TransferServer:
    Type: AWS::Transfer::Server
    Properties:
      Domain: S3
      EndpointDetails:
        VpcEndpointId: !Ref VPCEndpoint
```

```
      EndpointType: VPC_ENDPOINT
      IdentityProviderType: SERVICE_MANAGED
      Protocols:
        - SFTP
  VPCEndpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: com.amazonaws.us-east-1.transfer.server
      SecurityGroupIds:
        - !Ref SecurityGroupId
      SubnetIds:
        - !Select [0, !Ref SubnetIds]
        - !Select [1, !Ref SubnetIds]
        - !Select [2, !Ref SubnetIds]
      VpcEndpointType: Interface
      VpcId: !Ref VpcId
```

The template is updated with the following changes:

- The `EndpointType` was changed to `VPC`.
- The `AWS::EC2::VPCEndpoint` resource is removed.
- The `SecurityGroupId`, `SubnetIds`, and `VpcId` were moved to the `EndpointDetails` section of the `AWS::Transfer::Server` resource,
- The `VpcEndpointId` property of `EndpointDetails` was removed.

The updated template looks as follows:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Create AWS Transfer Server with VPC endpoint type'
Parameters:
  SecurityGroupId:
    Type: AWS::EC2::SecurityGroup::Id
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Type: AWS::EC2::VPC::Id
Resources:
  TransferServer:
    Type: AWS::Transfer::Server
    Properties:
      Domain: S3
```

```
    EndpointDetails:
      SecurityGroupIds:
        - !Ref SecurityGroupId
      SubnetIds:
        - !Select [0, !Ref SubnetIds]
        - !Select [1, !Ref SubnetIds]
        - !Select [2, !Ref SubnetIds]
      VpcId: !Ref VpcId
    EndpointType: VPC
    IdentityProviderType: SERVICE_MANAGED
    Protocols:
      - SFTP
```

To update the endpoint type of Transfer Family servers deployed using CloudFormation

1. Stop the server that you want to update using the following steps.

   a. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.

   b. In the navigation pane, choose **Servers**.

   c. Select the check box of the server that you want to change the endpoint type for.

   > **Important**
   >
   > You must stop the server before you can change its endpoint.

   d. For **Actions**, choose **Stop**.

   e. In the confirmation dialog box that appears, choose **Stop** to confirm that you want to stop the server.

   > **Note**
   >
   > Before proceeding to the next step, wait for the **Status** of the server to change to **Offline**; this can take a couple of minutes. You might have to choose **Refresh** on the **Servers** page to see the status change.

2. Update the CloudFormation stack

   a. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.

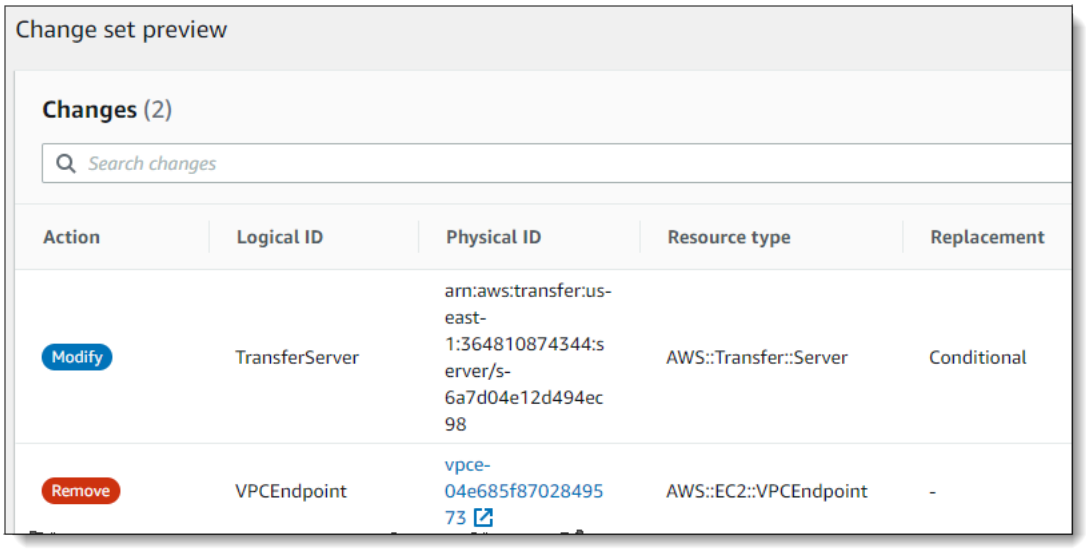
   b. Choose the stack used to create the Transfer Family server.

c. Choose **Update**.

d. Choose **Replace current template**

e. Upload the new template. CloudFormation Change Sets help you understand how template changes will affect running resources before you implement them. In this example, the Transfer server resource will be modified, and the VPCEndpoint resource will be removed. The VPC endpoint type server creates a VPC Endpoint on your behalf, replacing the original `VPCEndpoint` resource.

After uploading the new template, the change set will look similar to the following:



f. Update the stack.

3. Once the stack update is complete, navigate to the Transfer Family management console at https://console.aws.amazon.com/transfer/.

4. Restart the server. Choose the server you updated in CloudFormation, and then choose **Start** from the **Actions** menu.

Updating the server EndpointType using the API

You can use the describe-server AWS CLI command, or the UpdateServer API command. The following example script stops the Transfer Family server, updates the EndpointType, removes the VPC_ENDPOINT, and starts the server.

```
import boto3
import time
```

```
profile = input("Enter the name of the AWS account you'll be working in: ")
region_name = input("Enter the AWS Region you're working in: ")
server_id = input("Enter the AWS Transfer Server Id: ")

session = boto3.Session(profile_name=profile)

ec2 = session.client("ec2", region_name=region_name)
transfer = session.client("transfer", region_name=region_name)

group_ids=[]

transfer_description = transfer.describe_server(ServerId=server_id)
if transfer_description['Server']['EndpointType']=='VPC_ENDPOINT':
    transfer_vpc_endpoint = transfer_description['Server']['EndpointDetails']
['VpcEndpointId']
    transfer_vpc_endpoint_descriptions =
 ec2.describe_vpc_endpoints(VpcEndpointIds=[transfer_vpc_endpoint])
    for transfer_vpc_endpoint_description in
 transfer_vpc_endpoint_descriptions['VpcEndpoints']:
        subnet_ids=transfer_vpc_endpoint_description['SubnetIds']
        group_id_list=transfer_vpc_endpoint_description['Groups']
        vpc_id=transfer_vpc_endpoint_description['VpcId']
        for group_id in group_id_list:
             group_ids.append(group_id['GroupId'])
   if transfer_description['Server']['State']=='ONLINE':
        transfer_stop = transfer.stop_server(ServerId=server_id)
        print(transfer_stop)
        time.sleep(300) #safe
        transfer_update =
 transfer.update_server(ServerId=server_id,EndpointType='VPC',EndpointDetails={'SecurityGroupId
        print(transfer_update)
        time.sleep(10)
        transfer_start = transfer.start_server(ServerId=server_id)
        print(transfer_start)
        delete_vpc_endpoint =
 ec2.delete_vpc_endpoints(VpcEndpointIds=[transfer_vpc_endpoint])
```

Working with custom hostnames

Your *server host name* is the hostname that your users enter in their clients when they connect to your server. You can use a custom domain that you have registered for your server hostname when you work with AWS Transfer Family. For example, you might use a custom hostname like `mysftpserver.mysubdomain.domain.com`.

To redirect traffic from your registered custom domain to your server endpoint, you can use Amazon Route 53 or any Domain Name System (DNS) provider. Route 53 is the DNS service that AWS Transfer Family natively supports.

Topics

- Use Amazon Route 53 as your DNS provider
- Use other DNS providers
- Custom hostnames for non-console created servers

On the console, you can choose one of these options for setting up a custom hostname:

- **Amazon Route 53 DNS alias** – if the hostname that you want to use is registered with Route 53. You can then enter the hostname.
- **Other DNS** – if the hostname that you want to use is registered with another DNS provider. You can then enter the hostname.
- **None** – to use the server's endpoint and not use a custom hostname.

You set this option when you create a new server or edit the configuration of an existing server. For more information about creating a new server, see Step 2: Create an SFTP-enabled server. For more information about editing the configuration of an existing server, see Edit server details.

For more details about using your own domain for the server hostname and how AWS Transfer Family uses Route 53, see the following sections.

Use Amazon Route 53 as your DNS provider

When you create a server, you can use Amazon Route 53 as your DNS provider. Before you use a domain with Route 53, you register the domain. For more information, see How Domain registration works in the *Amazon Route 53 Developer Guide*.

When you use Route 53 to provide DNS routing to your server, AWS Transfer Family uses the custom hostname that you entered to extract its hosted zone. When AWS Transfer Family extracts a hosted zone, three things can happen:

1. If you're new to Route 53 and don't have a hosted zone, AWS Transfer Family adds a new hosted zone and a `CNAME` record. The value of this `CNAME` record is the endpoint hostname for your server. A *CNAME* is an alternate domain name.

2. If you have a hosted zone in Route 53 without any CNAME records, AWS Transfer Family adds a CNAME record to the hosted zone.
3. If the service detects that a CNAME record already exists in the hosted zone, you see an error indicating that a CNAME record already exists. In this case, change the value of the CNAME record to the hostname of your server.

For more information about hosted zones in Route 53, see Hosted zone in the *Amazon Route 53 Developer Guide*.

Use other DNS providers

When you create a server, you can also use DNS providers other than Amazon Route 53. If you use an alternate DNS provider, make sure that traffic from your domain is directed to your server endpoint.

To do so, set your domain to the endpoint hostname for the server.

- For IPv4 endpoints, the hostname looks like this in the console:

  serverid.server.transfer.*region*.amazonaws.com

- For dual-stack endpoints, the hostname looks like this in the console:

  serverid.transfer-server.*region*.on.aws

> **Note**
>
> If your server has a VPC endpoint, then the format for the hostname is different from those described above. To find your VPC endpoint, select the VPC on the server's details page, then select the **VPC endpoint ID** on the VPC dashboard. The endpoint is the first DNS name of those listed.

Custom hostnames for non-console created servers

When you create a server using AWS Cloud Development Kit (AWS CDK), CloudFormation, or through the CLI, you must add a tag if you want that server to have a custom hostname. When you create a Transfer Family server by using the console, the tagging is done automatically.

> **Note**
>
> You also need to create a DNS record to redirect traffic from your domain to your server endpoint. For details, see Working with records in the *Amazon Route 53 Developer Guide*.

Use the following keys for your custom hostname:

- Add `transfer:customHostname` to display the custom hostname in the console.
- If you are using Route 53 as your DNS provider, add `transfer:route53HostedZoneId`. This tag links the custom hostname to your Route 53 Hosted Zone ID.

To add the custom hostname, issue the following CLI command.

```
aws transfer tag-resource --arn arn:aws:transfer:region:AWS account:server/server-ID --
tags Key=transfer:customHostname,Value="custom-host-name"
```

For example:

```
aws transfer tag-resource --arn arn:aws:transfer:us-east-1:111122223333:server/
s-1234567890abcdef0 --tags Key=transfer:customHostname,Value="abc.example.com"
```

If you are using Route 53, issue the following command to link your custom hostname to your Route 53 Hosted Zone ID.

```
aws transfer tag-resource --arn server-ARN:server/server-ID --tags
 Key=transfer:route53HostedZoneId,Value=HOSTED-ZONE-ID
```

For example:

```
aws transfer tag-resource --arn arn:aws:transfer:us-east-1:111122223333:server/
s-1234567890abcdef0 --tags Key=transfer:route53HostedZoneId,Value=ABCDE1111222233334444
```

Assuming the sample values from the previous command, run the following command to view your tags:

```
aws transfer list-tags-for-resource --arn arn:aws:transfer:us-
east-1:111122223333:server/s-1234567890abcdef0
```

```
"Tags": [
   {
      "Key": "transfer:route53HostedZoneId",
      "Value": "/hostedzone/ABCDE1111222233334444"
   },
   {
      "Key": "transfer:customHostname",
      "Value": "abc.example.com"
   }
 ]
```

Note

Your public, hosted zones and their IDs are available on Amazon Route 53.
Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.

Working with Network Load Balancers

Whether you should place a Network Load Balancer (NLB) in front of an AWS Transfer Family server depends on the protocol that your server uses. Review the following guidance before you add an NLB to your architecture.

- **FTP and FTPS** – We recommend against placing an NLB in front of the server. An NLB in this path increases costs and reduces the number of simultaneous connections that the server accepts. If your FTP or FTPS implementation requires an NLB, see Avoid placing NLBs and NATs in front of AWS Transfer Family servers for configuration recommendations.
- **SFTP** – An NLB is supported for servers that use a VPC-hosted endpoint, though we recommend avoiding an NLB where possible. Where an NLB is required, you can preserve the client's source IP address with PROXY protocol v2 (PPv2).

A common reason to place an NLB in front of an AWS Transfer Family server is to offer a custom listener port. AWS Transfer Family already offers several ports, so review AWS Transfer Family endpoint type matrix and Create an SFTP-enabled server for the supported ports before you add an NLB to expose a custom listener port.

Preserving the client's source IP for SFTP with PROXY protocol v2

This feature applies to SFTP servers that use a VPC-hosted endpoint. When an NLB is in the path between your client and your SFTP server, AWS Transfer Family sees the private IP address of the NLB instead of the client's source IP address. PROXY protocol v2 (PPv2) solves this problem. The NLB adds a PPv2 header that carries the client's source IP address, and AWS Transfer Family reads that header.

When you enable PPv2, the client's source IP address becomes available in the following places:

- Authentication requests that AWS Transfer Family sends to your custom identity provider. You can then write IP-based access policies.
- Log entries in Amazon CloudWatch Logs.

SSH client compatibility with PROXY protocol v2

After the TCP connection is established, both the client and the server send an SSH identification string as their first data. This string identifies their SSH protocol and software versions, for example `SSH-2.0-OpenSSH_10.3`. For more information, see SSH transport layer protocol (RFC 4253) on the IETF Datatracker website.

Some older SSH clients wait to receive the server's SSH identification string before they send their own. Clients that still support SSH protocol version 1 do this so that they can choose a protocol version based on the server's identification string. When PROXY protocol v2 is enabled on an NLB target group, the NLB doesn't deliver the server's identification string to the client until the client sends data. These clients connect and wait for the server's identification string, so they never send their own, and the connection stalls until it times out.

The following table lists common SSH clients and the minimum version of each that is known to be compatible with PROXY protocol v2 in its default configuration. This list isn't comprehensive. An older version might also work if you configure it to use only SSH protocol version 2. If you use a client that isn't in this table, test its compatibility before you enable PROXY protocol v2.

Client	Minimum version
WinSCP	5.8.1
OpenSSH	6.2

Client	Minimum version
JSch	0.1.34
Paramiko	All versions
SSH.NET	2020.0.0
FileZilla	3.14.0
ssh2 (Node.js)	All versions
Apache MINA SSHD	All versions
libssh2	0.6
SSHJ	All versions
Cyberduck	2.2

To find the SSH clients that connect to your server, enable structured logging for your server. The `client` field contains the client's identification string, for example `SSH-2.0-OpenSSH_10.3`. For more information, see JSON structured logs for Transfer Family. If a client waits for the server's SSH identification string before sending its own, or doesn't appear in this table, ask those users to upgrade before you enable PROXY protocol v2 for your NLB.

Enable PROXY protocol v2 enforcement on an existing server

The `SftpMode` member of `ProxyConfig` controls how your server handles the PPv2 header. It has exactly two values: `NONE` (the default) and `PROXY_PROTOCOL_V2_ENFORCED`. For more information, see ProtocolDetails in the *AWS Transfer Family API Reference*.

> **Important**
>
> With `PROXY_PROTOCOL_V2_ENFORCED`, you must restrict the server's VPC endpoint security group to allow inbound traffic only through the trusted NLB. Because the server trusts the source IP address in the PPv2 header, this prevents a client from bypassing the NLB and presenting a spoofed source IP address.

Complete the following three steps in order to enable enforcement without refusing any connections. This order keeps the migration free of downtime.

Step 1: Enable PROXY protocol v2 on the NLB target group

Enable the `proxy_protocol_v2.enabled` attribute on the NLB target group that points at your server's VPC endpoint. This attribute tells the NLB to prepend a PPv2 header to each connection. For instructions, see Proxy protocol for your Network Load Balancer in the *Elastic Load Balancing User Guide*.

While the server is set to `SftpMode=NONE`, it accepts connections with or without a PPv2 header.

> **Note**
>
> Some SSH clients wait to receive the server's SSH identification string before sending their own. When you enable PROXY protocol v2, these clients fail to complete the connection. Before you enable PROXY protocol v2, confirm that your users' clients send their identification string first. For details, see SSH client compatibility with PROXY protocol v2.

Step 2: Verify that the header arrives on every connection

Before you enforce PPv2, confirm that the server receives a PPv2 header on every connection. Inspect your server logs for the `CONNECTED` activity type. In structured (JSON) logs in Amazon CloudWatch Logs, check the `proxy-protocol-v2-header` field for the value `ignored`. In legacy logs, check the `ProxyProtocolV2Header` field for the same value.

The field has the following semantics:

- The field is absent when the connection included no PPv2 header.
- The value is `ignored` when the connection included a PPv2 header while `SftpMode` is `NONE`.
- The value is `applied` when the connection included a PPv2 header and the server honored it under `PROXY_PROTOCOL_V2_ENFORCED`.

Keep checking the logs until every `CONNECTED` entry carries the field with the value `ignored`. An absent field means that some path still reaches the server without a PPv2 header. If you enforce while any path is missing the header, the server refuses those connections. For more information

about these log fields, see JSON structured logs for Transfer Family and Legacy logs for Transfer Family.

Step 3: Enforce PROXY protocol v2

Update the server so that the `SftpMode` member of `ProxyConfig` is `PROXY_PROTOCOL_V2_ENFORCED`. The server then requires a valid PPv2 header on every incoming SFTP connection. The server refuses connections that arrive without a valid PPv2 header and logs an error to Amazon CloudWatch Logs. To roll back, set `SftpMode` to `NONE`.

Console

On the **Server details** page, choose **Edit** next to **Additional details**, and then turn on the **PROXY protocol configuration** option. For the full edit-server procedure, see Edit server details.

AWS CLI

Run the following command to enforce PROXY protocol v2 on your server.

```
aws transfer update-server --server-id your-server-id --protocol-details
 ProxyConfig={SftpMode=PROXY_PROTOCOL_V2_ENFORCED}
```

If successful, the command returns the following code and updates your server.

```
{
    "ServerId": "your-server-id"
}
```

FTP and FTPS Network Load Balancer considerations

Although we recommend avoiding Network Load Balancers in front of AWS Transfer Family servers, if your FTP or FTPS implementation requires an NLB or NAT in the communication route from the client, follow these recommendations:

- For an NLB, use port 21 for health checks, instead of ports 8192-8200.
- For the AWS Transfer Family server, enable TLS session resumption by setting `TlsSessionResumptionMode = ENFORCED`.

> **Note**
>
> This is the recommended mode, as it provides enhanced security:
>
> - Requires clients to use TLS session resumption for subsequent connections.
> - Provides stronger security guarantees by ensuring consistent encryption parameters.
> - Helps prevent potential downgrade attacks.
> - Maintains compliance with security standards while optimizing performance.

- If possible, migrate away from using an NLB to take full advantage of AWS Transfer Family performance and connection limits.

For additional guidance on NLB alternatives, contact the AWS Transfer Family Product Management team through AWS Support. For more information about improving your security posture, see the blog post Six tips to improve the security of your AWS Transfer Family server.

Security guidance for NLBs is provided in Avoid placing NLBs and NATs in front of AWS Transfer Family servers.

Transferring files over a server endpoint using a client

You transfer files over the AWS Transfer Family service by specifying the transfer operation in a client. AWS Transfer Family supports the following clients:

- We support version 3 of the SFTP protocol.
- OpenSSH (macOS and Linux)

> **Note**
>
> This client works only with servers that are enabled for Secure Shell (SSH) File Transfer Protocol (SFTP).

- WinSCP (Microsoft Windows only)
- Cyberduck (Windows, macOS, and Linux)
- FileZilla (Windows, macOS, and Linux)

The following limitations apply to every client:

- The SCP protocol is not supported, as it is considered insecure. You can use the OpenSSH `scp` command as described in Using the `scp` command.
- The maximum number of concurrent, multiplexed, SFTP sessions per connection is 10.
- For idle connections, the timeout value is 1800 seconds (30 minutes) for all protocols (SFTP/FTP/FTPS). If there is no activity after this period, the client may be disconnected. For unresponsive connections:
  - SFTP has a 300 seconds (5 minutes) timeout when a client is completely unresponsive.
  - FTPS and FTP have an approximately 10 minute unresponsive timeout that is handled by the underlying library.
- Amazon S3 and Amazon EFS (due to the NFSv4 protocol) require filenames to be in UTF-8 encoding. Using different encoding can lead to unexpected results. For Amazon S3, see Object key naming guidelines.
- For File Transfer Protocol over SSL (FTPS), only Explicit mode is supported. Implicit mode is not supported.
- For File Transfer Protocol (FTP) and FTPS, only Passive mode is supported.
- For FTP and FTPS, only STREAM mode is supported.
- For FTP and FTPS, only Image/Binary mode is supported.
- For FTP and FTPS, TLS - PROT C (unprotected) TLS for the data connection is the default but PROT C is not supported in the AWS Transfer Family FTPS protocol. So for FTPS, you need to issue PROT P for your data operation to be accepted.
- If you are using Amazon S3 for your server's storage, and if your client contains an option to use multiple connections for a single transfer, make sure to disable the option. Otherwise, large file uploads can fail in unpredictable ways. Note that if you are using Amazon EFS as your storage backend, EFS *does* support multiple connections for a single transfer.

The following is a list of available commands for FTP and FTPS:

Available commands					
ABOR	FEAT	MLST	PASS	RETR	STOR
AUTH	LANG	MKD	PASV	RMD	STOU

Available commands					
CDUP	LIST	MODE	PBSZ	RNFR	STRU
CWD	MDTM	NLST	PROT	RNTO	SYST
DELE	MFMT	NOOP	PWD	SIZE	TYPE
EPSV	MLSD	OPTS	QUIT	STAT	USER

> **Note**
>
> APPE is not supported.

For SFTP, the following operations are currently not supported for users that are using the logical home directory on servers that are using Amazon Elastic File System (Amazon EFS).

Unsupported SFTP commands			
SSH_FXP_READLINK	SSH_FXP_SYMLINK	SSH_FXP_STAT when the requested file is a symlink	SSH_FXP_REALPATH when the requested path contains any symlink components

Generate public-private key pair

Before you can transfer a file, you must have a public-private key pair available. If you have not previously generated a key pair, see Generate SSH keys for service-managed users.

Topics

- Available SFTP/FTPS/FTP Commands
- Find your Amazon VPC endpoint
- Avoid setstat errors
- Use OpenSSH
- Use WinSCP

- Use Cyberduck
- Use FileZilla
- Use a Perl client
- Use LFTP
- Post upload processing
- SFTP messages

Available SFTP/FTPS/FTP Commands

The following table describes the available commands for AWS Transfer Family, for the SFTP, FTPS, and FTP protocols.

> **Note**
>
> The table mentions *files* and *directories* for Amazon S3, which only supports buckets and objects: there is no hierarchy. However, you can use prefixes in object key names to imply a hierarchy and organize your data in a way similar to folders. This behavior is described in Working with object metadata in the *Amazon Simple Storage Service User Guide*.

SFTP/FTPS/FTP Commands

Command	Amazon S3	Amazon EFS
`cd`	Supported	Supported
`chgrp`	Not supported	Supported (`root` or `owner` only)
`chmod`	Not supported	Supported (`root` only)
`chmtime`	Not supported	Supported
`chown`	Not supported	Supported (`root` only)
`get`	Supported	Supported (including resolving symbolic links)

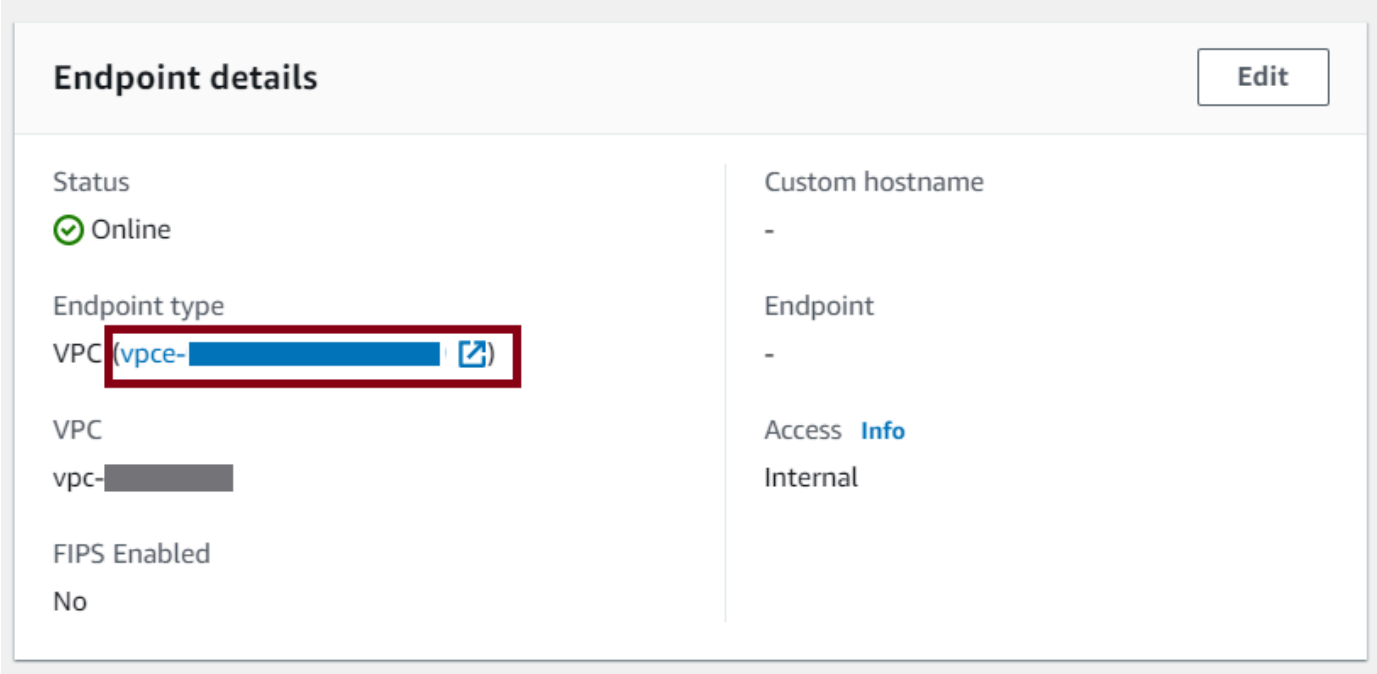
Command	Amazon S3	Amazon EFS
`ln -s`	Not supported	Supported
`ls/dir`	Supported	Supported
`mkdir`	Supported	Supported
`put`	Supported	Supported
`pwd`	Supported	Supported
`rename`	Supported for files only **Note** Renaming that would overwrite an existing file is not supported.	Supported **Note** Renaming that would overwrite an existing file or directory is not supported.
`rm`	Supported	Supported
`rmdir`	Supported (empty directories only)	Supported
`version`	Supported	Supported

Find your Amazon VPC endpoint

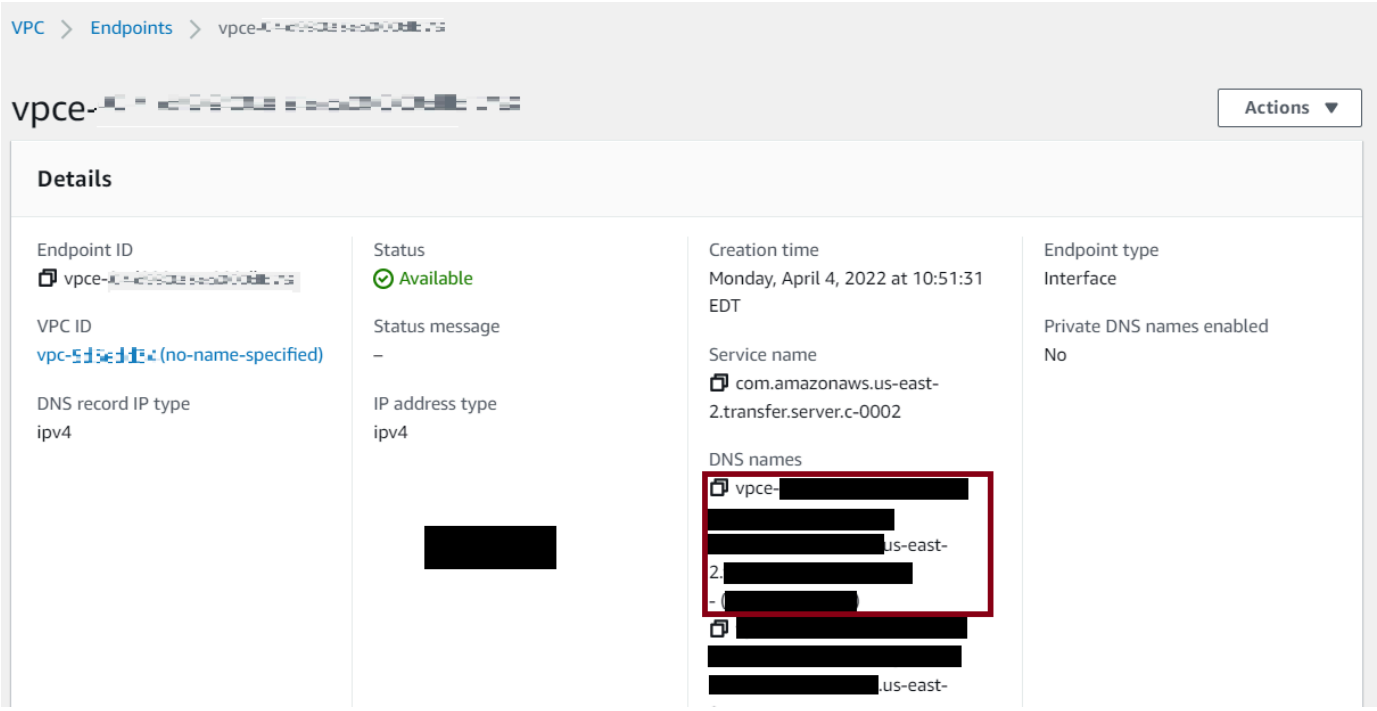
If the endpoint type for your Transfer Family server is VPC, identifying the endpoint to use for transferring files is not straightforward. In this case, use the following procedure to find your Amazon VPC endpoint.

To find your Amazon VPC endpoint

1. Navigate to your server's details page.
2. In the **Endpoint details** pane, select the **VPC**.



3. In the Amazon VPC dashboard, select the **VPC endpoint ID**.
4. In the list of **DNS names**, your server endpoint is the first one listed.



Avoid `setstat` errors

Some SFTP file transfer clients can attempt to change the attributes of remote files, including timestamp and permissions, using commands, such as SETSTAT when uploading the file. However, these commands are not compatible with object storage systems, such as Amazon S3. Due to this incompatibility, file uploads from these clients can result in errors even when the file is otherwise successfully uploaded.

- When you call the `CreateServer` or `UpdateServer` API, use the `ProtocolDetails` option `SetStatOption` to ignore the error that is generated when the client attempts to use SETSTAT on a file you are uploading to an S3 bucket.
- Set the value to `ENABLE_NO_OP` to have the Transfer Family server ignore the SETSTAT command, and upload files without needing to make any changes to your SFTP client.
- Note that while the `SetStatOption` `ENABLE_NO_OP` setting ignores the error, it *does* generate a log entry in CloudWatch Logs, so you can determine when the client is making a SETSTAT call.

For the API details for this option, see ProtocolDetails.

Use OpenSSH

This section contains instructions to transfer files from the command line using OpenSSH.

> **Note**
>
> This client works only with an SFTP-enabled server.

Topics

- Using OpenSSH
- Using the scp command

Using OpenSSH

To transfer files over AWS Transfer Family using the OpenSSH command line utility

1. On Linux, macOS, or Windows, open a command terminal.

2. At the prompt, enter the following command:

   ```
   sftp -i transfer-key sftp_user@service_endpoint
   ```

   In the preceding command, *sftp_user* is the username and *transfer-key* is the SSH private key. Here, *service_endpoint* is the server's endpoint as shown in the AWS Transfer Family console for the selected server.

   > **Note**
   >
   > This command uses settings that are in the default `ssh_config` file. Unless you have previously edited this file, SFTP uses port 22. You can specify a different port (for example 2222) by adding a **-P** flag to the command, as follows.
   >
   > ```
   > sftp -P 2222 -i transfer-key sftp_user@service_endpoint
   > ```
   >
   > Alternatively, if you always want to use port 2222 or port 22000, you can update your default port in your `ssh_config` file.

   An `sftp` prompt should appear.

3. (Optional) To view the user's home directory, enter the following command at the `sftp` prompt:

   ```
   pwd
   ```

4. To upload a file from your file system to the Transfer Family server, use the `put` command. For example, to upload `hello.txt` (assuming that file is in your current directory on your file system), run the following command at the `sftp` prompt:

   ```
   put hello.txt
   ```

   A message similar to the following appears, indicating that the file transfer is in progress, or complete.

   ```
   Uploading hello.txt to /amzn-s3-demo-bucket/home/sftp_user/hello.txt

   hello.txt 100% 127 0.1KB/s 00:00
   ```

> **Note**
>
> After your server is created, it can take a few minutes for the server endpoint hostname to be resolvable by the DNS service in your environment.

Using the scp command

Transfer Family doesn't support the SCP protocol. However, you can use the OpenSSH `scp` command if you need this functionality.

The recommendation for using SCP over SFTP is to use OpenSSH version 9.0 or later. In OpenSSH version 9 and later, the `scp` command defaults to using the SFTP protocol for file transfers instead of the legacy SCP protocol.

> **Important**
>
> Ensure that your Transfer Family server has been configured to use S3 optimized directory access.

Use WinSCP

Use the instructions that follow to transfer files from the command line using WinSCP.

> **Note**
>
> If you are using WinSCP 5.19, you can directly connect to Amazon S3 using your AWS credentials and upload/download files. For more details, see Connecting to Amazon S3 service.

To transfer files over AWS Transfer Family using WinSCP

1. Open the WinSCP client.
2. In the **Login** dialog box, for **File protocol**, choose a protocol: **SFTP** or **FTP**.

   If you chose FTP, for **Encryption**, choose one of the following:

- **No encryption** for FTP
- **TLS/SSL Explicit encryption** for FTPS

3. For **Host name**, enter your server endpoint. The server endpoint is located on the **Server details** page. For more information, see View SFTP, FTPS, and FTP server details.

   If your server uses a VPC endpoint, see Find your Amazon VPC endpoint.

4. For **Port number**, enter the following:

   - **22** for SFTP
   - **21** for FTP/FTPS

5. For **User name**, enter the name for the user that you created for your specific identity provider.

   Tip: The username should be one of the users you created or configured for your identity provider. AWS Transfer Family provides the following identity providers:

   - Working with service-managed users
   - Using AWS Directory Service for Microsoft Active Directory
   - Working with custom identity providers

6. Choose **Advanced** to open the **Advanced Site Settings** dialog box. In the **SSH** section, choose **Authentication**.
7. For **Private key file**, browse for and choose the SSH private key file from your file system.

   If WinSCP offers to convert your SSH private key to the PPK format, choose **OK**.

8. Choose **OK** to return to the **Login** dialog box, and then choose **Save**.
9. In the **Save session as site** dialog box, choose **OK** to complete your connection setup.
10. In the **Login** dialog box, choose **Tools**, and then choose **Preferences**.
11. In the **Preferences** dialog box, for **Transfer**, choose **Endurance**.

    For the **Enable transfer resume/transfer to temporary filename for** option, choose **Disable**.

> **Important**
>
> If you leave this option enabled, it increases upload costs, substantially decreasing upload performance. It also can lead to failures of large file uploads.

12. For **Transfer**, choose **Background**, and clear the **Use multiple connections for single transfer** check box.

    Tip: If you leave this option selected, large file uploads can fail in unpredictable ways. For example, orphaned multipart uploads that incur Amazon S3 charges can be created. Silent data corruption can also occur.

13. Perform your file transfer.

    You can use drag-and-drop methods to copy files between the target and source windows. You can use toolbar icons to upload, download, delete, edit, or modify the properties of files in WinSCP.

> **Note**
>
> This note does not apply if you are using Amazon EFS for storage.
> Commands that attempt to change attributes of remote files, including timestamps, are not compatible with object storage systems such as Amazon S3. Therefore, if you are using Amazon S3 for storage, be sure to disable WinSCP timestamp settings (or use the `SetStatOption` as described in Avoid `setstat` errors) before you perform file transfers. To do so, in the **WinSCP Transfer settings** dialog box, disable the **Set permissions** upload option and the **Preserve timestamp** common option.

Use Cyberduck

Use the instructions that follow to transfer files from the command line using Cyberduck.

To transfer files over AWS Transfer Family using Cyberduck

1. Open the Cyberduck client.
2. Choose **Open Connection**.
3. In the **Open Connection** dialog box, choose a protocol: **SFTP (SSH File Transfer Protocol)**, **FTP-SSL (Explicit AUTH TLS)**, or **FTP (File Transfer Protocol)**.
4. For **Server**, enter your server endpoint. The server endpoint is located on the **Server details** page. For more information, see View SFTP, FTPS, and FTP server details.

   If your server uses a VPC endpoint, see Find your Amazon VPC endpoint.

5. For **Port number**, enter the following:

   - **22** for SFTP
   - **21** for FTP/FTPS
6. For **Username**, enter the name for the user that you created in Managing users for server endpoints.
7. If SFTP is selected, for **SSH Private Key**, choose or enter the SSH private key.
8. Choose **Connect**.
9. Perform your file transfer.

   Depending on where your files are, do one of the following:

   - In your local directory (the source), choose the files that you want to transfer, and drag and drop them into the Amazon S3 directory (the target).
   - In the Amazon S3 directory (the source), choose the files that you want to transfer, and drag and drop them into your local directory (the target).

Use FileZilla

Use the instructions that follow to transfer files using FileZilla.

To set up FileZilla for a file transfer

1. Open the FileZilla client.
2. Choose **File**, and then choose **Site Manager**.
3. In the **Site Manager** dialog box, choose **New site**.
4. On the **General** tab, for **Protocol**, choose a protocol: **SFTP** or **FTP**.

   If you chose FTP, for **Encryption**, choose one of the following:

   - **Only use plain FTP (insecure)** – for FTP
   - **Use explicit FTP over TLS if available** – for FTPS
5. For **Host name**, enter the protocol that you are using, followed by your server endpoint. The server endpoint is located on the **Server details** page. For more information, see View SFTP, FTPS, and FTP server details.

   - If you are using SFTP, enter: `sftp://hostname`

- If you are using FTPS, enter: `ftps://hostname`

Make sure to replace *hostname* with your actual server endpoint.

If your server uses a VPC endpoint, see Find your Amazon VPC endpoint.

6. For **Port number**, enter the following:
   - **22** for SFTP
   - **21** for FTP/FTPS
7. If SFTP is selected, for **Logon Type**, choose **Key file**.

   For **Key file**, choose or enter the SSH private key.
8. For **User**, enter the name for the user that you created in Managing users for server endpoints.
9. Choose **Connect**.
10. Perform your file transfer.

> **Note**
>
> If you interrupt a file transfer in progress, AWS Transfer Family might write a partial object in your Amazon S3 bucket. If you interrupt an upload, check that the file size in the Amazon S3 bucket matches the file size of the source object before continuing.

Use a Perl client

If you use the NET::SFTP::Foreign perl client, you must set the `queue_size` to 1. For example:

```
my $sftp = Net::SFTP::Foreign->new('user@s-12345.server.transfer.us-east-2.amazonaws.com', queue_size => 1);
```

> **Note**
>
> This workaround is needed for revisions of `Net::SFTP::Foreign` prior to 1.92.02.

Use LFTP

LFTP is a free FTP client that allows users to perform file transfers via the command-line interface from most Linux machines.

For large file downloads, LFTP has a known issue with out of order packets, causing the file transfer to fail.

Post upload processing

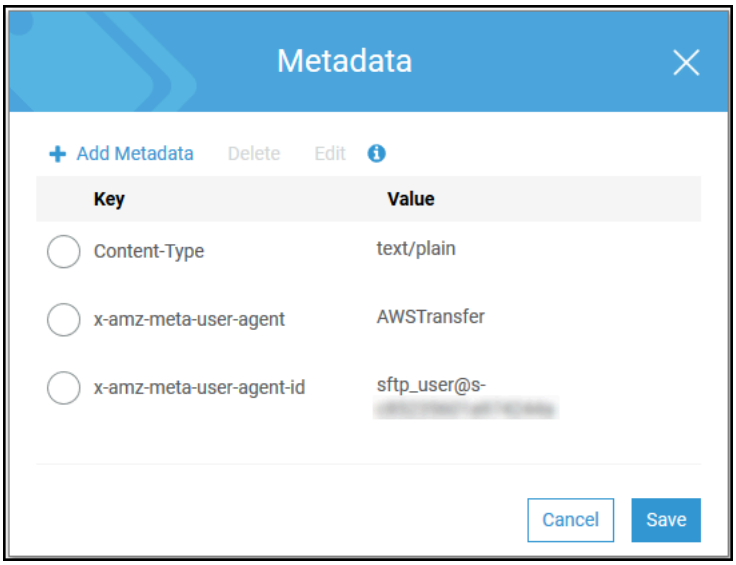
You can view post upload processing information including Amazon S3 object metadata and event notifications.

Topics

- Amazon S3 object metadata
- Amazon S3 event notifications

Amazon S3 object metadata

As a part of your object's metadata you see a key called `x-amz-meta-user-agent` whose value is `AWSTransfer` and `x-amz-meta-user-agent-id` whose value is `username@server-id`. The `username` is the Transfer Family user who uploaded the file and `server-id` is the server used for the upload. This information can be accessed using the HeadObject operation on the S3 object inside your Lambda function.



Amazon S3 event notifications

When an object is uploaded to your S3 bucket using Transfer Family, `RoleSessionName` is contained in the Requester field in the S3 event notification structure as `[AWS:Role Unique Identifier]/username.sessionid@server-id`. For example, the following are the contents for a sample Requester field from an S3 access log for a file that was copied to the S3 bucket.

`arn:aws:sts::AWS-Account-ID:assumed-role/`*`IamRoleName`*`/username.sessionid@server-id`

In the Requester field above, it shows the IAM Role called `IamRoleName`. For more information about configuring S3 event notifications, see Configuring Amazon S3 event notifications in the *Amazon Simple Storage Service Developer Guide*. For more information about AWS Identity and Access Management (IAM) role unique identifiers, see Unique identifiers in the *AWS Identity and Access Management User Guide*.

SFTP messages

This section describes client side messages that you may receive during or after your SFTP file transfers when using a Transfer Family server. For more information on any SFTP event, check your SFTP client logs. You can use that information to troubleshoot any errors, or forward that information to your network team for their help in identifying the issue.

SFTP client-side messages

Activity	Description
AUTH_FAILURE	The user failed authentication. This can be any kind of failure from a custom identity provider or service managed user. The details in the event help clarify the root cause of the failure.
CLOSE	Indicates that an opened file or directory is closed successfully.
CONNECTED/DISCONNECTED	Indicates normal connection success and disconnections.
CREATE_SYMLINK	A symbolic link was created (successfully or unsuccessfully).

Activity	Description
DELETE	A file was deleted (successfully or unsuccess fully).
ERROR	A general, unexpected error. The associated description contains information that can help you or your network administrators to identify the specific issue.
EXIT_REASON	Emitted when an unexpected error caused termination of your SFTP session. The message associated with the event describes the cause.
MKDIR	A directory was created (successfully or unsuccessfully).
OPEN	A file was opened for read or write (successf ully or unsuccessfully)
PARTIAL_CLOSE	The client disconnected from the server while a file was still open with no CLOSE message received. Transfer Family stores the received portion of the file (which could in fact be the complete file) and emits the PARTIAL_C LOSE event to alert the customer about the issue. Workflows integration also receives an on`PartialClose` event to handle the file appropriately.
RENAME	A file was renamed (successfully or unsuccess fully)
RMDIR	A directory was deleted (successfully or unsuccessfully)

Activity	Description
SETSTAT	The attributes of a file are changed (successf ully or unsuccessfully). **Note** Transfer Family doesn't support SETSTAT if you are using Amazon S3 for storage. The Avoid `setstat` errors section provides details on how to avoid `SetStat` errors, by turning off the setting. This avoids you receiving a `fail unsupported error :` instead, you receive `success but do nothing` message.
TLS_RESUME_FAILURE	The server is configured to enforce TLS Session Resumption and the client does not support it.

Managing users for server endpoints

In the following sections, you can find information about how to add users using AWS Transfer Family, AWS Directory Service for Microsoft Active Directory or a custom identity provider.

As part of each user's properties, you also store that user's Secure Shell (SSH) public key. Doing so is required for key-based authentication. The private key is stored locally on your user's computer. When your user sends an authentication request to your server by using a client, your server first confirms that the user has access to the associated SSH private key. The server then successfully authenticates the user.

Note

For automated deployment and management of users with multiple SSH keys, see Transfer Family Terraform modules.

In addition, you specify a user's home directory, or landing directory, and assign an AWS Identity and Access Management (IAM) role to the user. Optionally, you can provide a session policy to limit user access only to the home directory of your Amazon S3 bucket.

> **Important**
>
> AWS Transfer Family blocks usernames that are 1 or 2 characters long from authenticating to SFTP servers. Additionally, we also block the `root`user name.
> The reason behind this is due to the large volume of malicious login attempts by password scanners.

Amazon EFS vs. Amazon S3

Characteristics of each storage option:

- To limit access: Amazon S3 supports session policies; Amazon EFS supports POSIX user, group, and secondary group IDs
- Both support public/private keys
- Both support home directories
- Both support logical directories

> **Note**
>
> For Amazon S3, most of the support for logical directories is via API/CLI. You can use the **Restricted** check box in the console to lock down a user to their home directory, but you cannot specify a virtual directory structure.

Logical directories

If you are specifying logical directory values for your user, the parameter you use depends on the type of user.

- For service-managed users, provide logical directory values in `HomeDirectoryMappings`.
- For custom identity provider users, provide logical directory values in `HomeDirectoryDetails`.

AWS Transfer Family supports specifying a HomeDirectory value when using the LOGICAL HomeDirectoryType. This applies to Service Managed users, Active Directory access, and Custom Identity Provider implementations where the HomeDirectoryDetails are provided in the response.

> **Important**
>
> When specifying a HomeDirectory with LOGICAL HomeDirectoryType, the value must map to one of your logical directory mappings. The service validates this during both user creation and updates to prevent configurations that would not work.

Default behavior

By default, if left unspecified, the HomeDirectory is set to "/" for LOGICAL mode. This behavior is unchanged and remains compatible with existing user definitions.

- Make sure to map your HomeDirectory to an *Entry* and not a *Target*. For more details, see Rules for using logical directories.
- For details on how a virtual directory is structured see Virtual directory structure.

Custom Identity Provider considerations

When using a Custom Identity Provider, you can now specify a HomeDirectory in the response while using LOGICAL HomeDirectoryType. The TestIdentityProvider API call will produce correct results when the Custom IDP specifies a HomeDirectory in LOGICAL mode.

Example Custom IDP response with HomeDirectory and LOGICAL HomeDirectoryType:

```
{
  "Role": "arn:aws:iam::123456789012:role/transfer-user-role",
  "HomeDirectoryType": "LOGICAL",
  "HomeDirectory": "/marketing",
  "HomeDirectoryDetails": "[{\"Entry\":\"/\",\"Target\":\"/bucket/home\"},{\"Entry\":
\"/marketing\",\"Target\":\"/marketing-bucket/campaigns\"}]"
}
```

Active Directory group quotas

AWS Transfer Family has a default limit of 100 Active Directory groups per server. If your use case requires more than 100 groups, consider using a custom identity provider solution as described in Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

This limit applies to servers using the following identity providers:

- AWS Directory Service for Microsoft Active Directory
- AWS Directory Service for Entra ID Domain Services

If you need to request a service limit increase, see AWS service quotas in the *AWS General Reference*. If your use case requires more than 100 groups, consider using a custom identity provider solution as described in Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

For troubleshooting information related to Active Directory group limits, see Active Directory group limits exceeded.

Topics

- Working with service-managed users
- Working with custom identity providers
- Using AWS Directory Service for Microsoft Active Directory
- Using AWS Directory Service for Entra ID Domain Services

Working with service-managed users

You can add either Amazon S3 or Amazon EFS service-managed users to your server, depending on the server's **Domain** setting. For more information, see Configuring an SFTP, FTPS, or FTP server endpoint.

If you use a service-managed identity type, you add users to your file transfer protocol enabled server. When you do so, each username must be unique on your server.

To add a service-managed user programmatically, see the example for the CreateUser API.

> **Note**
>
> For service-managed users there is a limit of 2,000 logical directory entries. For information about using logical directories see Using logical directories to simplify your Transfer Family directory structures.

Topics

- Adding Amazon S3 service-managed users
- Adding Amazon EFS service-managed users
- Managing service-managed users

Adding Amazon S3 service-managed users

> **Note**
>
> If you want to configure a cross account Amazon S3 bucket, follow the steps mentioned in this Knowledge Center article: How do I configure my AWS Transfer Family server to use an Amazon Simple Storage Service bucket that's in another AWS account?.

To add an Amazon S3 service-managed user to your server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/, then select **Servers** from the navigation pane.
2. On the **Servers** page, select the check box of the server that you want to add a user to.
3. Choose **Add user**.
4. In the **User configuration** section, for **Username**, enter the username. This username must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A-Z, 0–9, underscore '_', hyphen '-', period '.' and at sign '@'. The username can't start with a hyphen '-', period '.' or at sign '@'.
5. For **Access**, choose the IAM role that you previously created that provides access to your Amazon S3 bucket.

   You created this IAM role using the procedure in Create an IAM role and policy. That IAM role includes an IAM policy that provides access to your Amazon S3 bucket. It also includes a trust

relationship with the AWS Transfer Family service, defined in another IAM policy. If you need fine-grained access control for your users, refer to the Enhance data access control with AWS Transfer Family and Amazon S3 blog post.

6. (Optional) For **Policy**, select one of the following:

   - **None**
   - **Existing policy**
   - **Select a policy from IAM**: allows you to choose an existing session policy. Choose **View** to see a JSON object containing the details of the policy.
   - **Auto-generate policy based on home folder**: generates a session policy for you. Choose **View** to see a JSON object containing the details of the policy.

   > **Note**
   >
   > If you choose **Auto-generate policy based on home folder**, do not select **Restricted** for this user.

   To learn more about session policies, see Create an IAM role and policy, Creating a session policy for an Amazon S3 bucket, or Dynamic permission management approaches.

7. For **Home directory**, choose the Amazon S3 bucket to store the data to transfer using AWS Transfer Family. Enter the path to the home directory where your user lands when they log in using their client.

   If you keep this parameter blank, the `root` directory of your Amazon S3 bucket is used. In this case, make sure that your IAM role provides access to this `root` directory.

   > **Note**
   >
   > We recommend that you choose a directory path that contains the user name of the user, which enables you to effectively use a session policy. The session policy limits user access in the Amazon S3 bucket to that user's home directory.

8. (Optional) For **Restricted**, select the check box so that your users can't access anything outside of that folder and can't see the Amazon S3 bucket or folder name.

> **Note**
>
> Assigning the user a home directory and restricting the user to that home directory should be sufficient to lock down the user's access to the designated folder. If you need to apply further controls, use a session policy.
> If you select **Restricted** for this user, you cannot select **Auto-generate policy based on home folder**, because home folder is not a defined value for Restricted users.

9. For **SSH public key**, enter the public SSH key portion of the SSH key pair.

   Your key is validated by the service before you can add your new user.

   > **Note**
   >
   > For instructions on how to generate an SSH key pair, see Generate SSH keys for service-managed users.

10. (Optional) For **Key** and **Value**, enter one or more tags as key-value pairs, and choose **Add tag**.
11. Choose **Add** to add your new user to the server that you chose.

    The new user appears in the **Users** section of the **Server details** page.

Next steps – For the next step, continue on to Transferring files over a server endpoint using a client.

Adding Amazon EFS service-managed users

Amazon EFS uses the Portable Operating System Interface (POSIX) file permission model to represent file ownership.

- For more details on Amazon EFS file ownership, see Amazon EFS file ownership.
- For more details on setting up directories for your EFS users, see Set up Amazon EFS users for Transfer Family.

To add an Amazon EFS service-managed user to your server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/, then select **Servers** from the navigation pane.
2. On the **Servers** page, select the Amazon EFS server that you want to add a user to.
3. Choose **Add user** to display the **Add user** page.
4. In the **User configuration** section, use the following settings.

    a. The **Username**, must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A-Z, 0–9, underscore '_', hyphen '-', period '.', and at sign "@". The username can't start with a hyphen '-', period '.', or at sign "@".

    b. For **User ID** and **Group ID**, note the following:

    - For the first user that you create, we recommend that you enter a value of **0** for both **Group ID** and **User ID**. This grants the user administrator privileges for Amazon EFS.
    - For additional users, enter the user's POSIX user ID and group ID. These IDs are used for all Amazon Elastic File System operations performed by the user.
    - For **User ID** and **Group ID**, do not use any leading zeroes. For example, **12345** is acceptable, **012345** is not.

    c. (Optional) For **Secondary Group IDs**, enter one or more additional POSIX group IDs for each user, separated by commas.

    d. For **Access**, choose the IAM role that:

    - Gives the user access to only the Amazon EFS resources (file systems) that you want them to access.
    - Defines which file system operations that the user can and cannot perform.

    We recommend that you use the IAM role for Amazon EFS file system selection with mount access and read/write permissions. For example, the combination of the following two AWS managed policies, while quite permissive, grants the necessary permissions for your user:

    - AmazonElasticFileSystemClientFullAccess
    - AWSTransferConsoleFullAccess

For more information, see the blog post AWS Transfer Family support for Amazon Elastic File System.

e. For **Home directory**, do the following:

- Choose the Amazon EFS file system that you want to use for storing the data to transfer using AWS Transfer Family.
- Decide whether to set the home directory to **Restricted**. Setting the home directory to **Restricted** has the following effects:
  - Amazon EFS users can't access any files or directories outside of that folder.
  - Amazon EFS users can't see the Amazon EFS file system name (**fs-xxxxxxx**).

> **Note**
>
> When you select the **Restricted** option, symlinks don't resolve for Amazon EFS users.

- (Optional) Enter the path to the home directory that you want users to be in when they log in using their client.

  If you don't specify a home directory, the root directory of your Amazon EFS file system is used. In this case, make sure that your IAM role provides access to this root directory.

5. For **SSH public key**, enter the public SSH key portion of the SSH key pair.

   Your key is validated by the service before you can add your new user.

> **Note**
>
> For instructions on how to generate an SSH key pair, see Generate SSH keys for service-managed users.

6. (Optional) Enter any tags for the user. For **Key** and **Value**, enter one or more tags as key-value pairs, and choose **Add tag**.
7. Choose **Add** to add your new user to the server that you chose.

   The new user appears in the **Users** section of the **Server details** page.

Issues that you might encounter when you first SFTP to your Transfer Family server:

- If you run the `sftp` command and the prompt doesn't appear, you might encounter the following message:

  ```
  Couldn't canonicalize: Permission denied

  Need cwd
  ```

  In this case, you must increase the policy permissions for your user's role. You can add an AWS managed policy, such as `AmazonElasticFileSystemClientFullAccess`.
- If you enter `pwd` at the `sftp` prompt to view the user's home directory, you might see the following message, where *`USER-HOME-DIRECTORY`* is the home directory for the SFTP user:

  ```
  remote readdir("/USER-HOME-DIRECTORY"): No such file or directory
  ```

  In this case, you should be able to navigate to the parent directory (`cd ..`), and create the user's home directory (`mkdir username`) .

Next steps – For the next step, continue on to Transferring files over a server endpoint using a client.

Managing service-managed users

In this section, you can find information about how to view a list of users, how to edit user details, and how to add an SSH public key.

- View a list of users
- View or edit user details
- Delete a user
- Add SSH public key
- Delete SSH public key

To find a list of your users

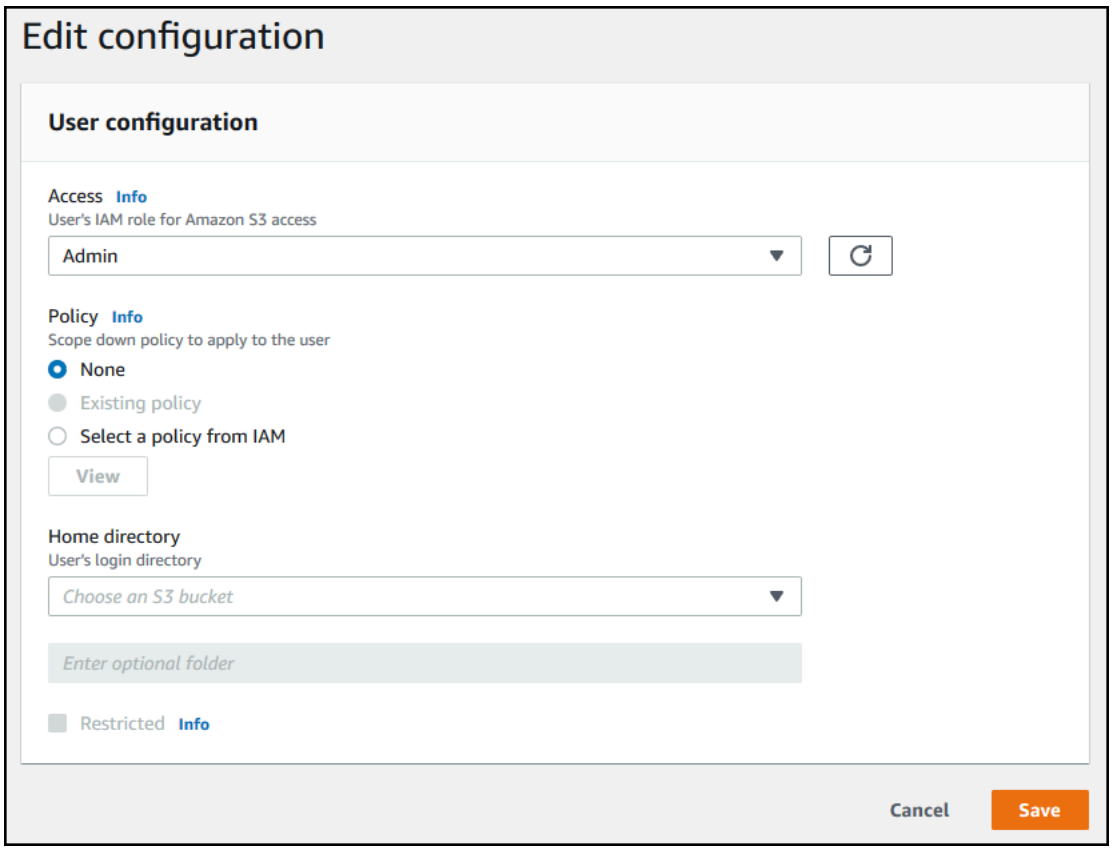
1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. Select **Servers** from the navigation pane to display the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.

4. Under **Users**, view a list of users.

To view or edit user details

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. Select **Servers** from the navigation pane to display the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, choose a username to see the **User details** page.

   You can change the user's properties on this page by choosing **Edit**.
5. On the **Users details** page, choose **Edit** next to **User configuration**.



6. On the **Edit configuration** page, for **Access**, choose the IAM role that you previously created that provides access to your Amazon S3 bucket.

   You created this IAM role using the procedure in Create an IAM role and policy. That IAM role includes an IAM policy that provides access to your Amazon S3 bucket. It also includes a trust relationship with the AWS Transfer Family service, defined in another IAM policy.
7. (Optional) For **Policy**, choose one of the following:

- **None**
- **Existing policy**
- **Select a policy from IAM** to choose an existing policy. Choose **View** to see a JSON object containing the details of the policy.

To learn more about session policies, see Create an IAM role and policy. To learn more about creating a session policy, see Creating a session policy for an Amazon S3 bucket.

8. For **Home directory**, choose the Amazon S3 bucket to store the data to transfer using AWS Transfer Family. Enter the path to the home directory where your user lands when they log in using their client.

   If you leave this parameter blank, the `root` directory of your Amazon S3 bucket is used. In this case, make sure that your IAM role provides access to this `root` directory.

   > **Note**
   >
   > We recommend that you choose a directory path that contains the user name of the user, which enables you to effectively use a session policy. The session policy limits user access in the Amazon S3 bucket to that user's home directory.

9. (Optional) For **Restricted**, select the check box so that your users can't access anything outside of that folder and can't see the Amazon S3 bucket or folder name.

   > **Note**
   >
   > When assigning the user a home directory and restricting the user to that home directory, this should be sufficient enough to lock down the user's access to the designated folder. Use a session policy when you need to apply further controls.

10. Choose **Save** to save your changes.

To delete a user

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. Select **Servers** from the navigation pane to display the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.

4. Under **Users**, choose a username to see the **User details** page.
5. On the **Users details** page, choose **Delete** to the right of the username.
6. In the confirmation dialog box that appears, enter the word **`delete`**, and then choose **Delete** to confirm that you want to delete the user.

The user is deleted from the **users** list.

To add an SSH public key for a user

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, choose a username to see the **User details** page.
5. Choose **Add SSH public key** to add a new SSH public key to a user.

> **Note**
>
> SSH keys are used only by servers that are enabled for Secure Shell (SSH) File Transfer Protocol (SFTP). For information about how to generate an SSH key pair, see Generate SSH keys for service-managed users.

6. For **SSH public key**, enter the SSH public key portion of the SSH key pair.

   Your key is validated by the service before you can add your new user. The format of the SSH key is `ssh-rsa` *`string`*. To generate an SSH key pair, see Generate SSH keys for service-managed users.
7. Choose **Add key**.

To delete an SSH public key for a user

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, choose a username to see the **User details** page.
5. To delete a public key, select its SSH key check box and choose **Delete**.

Working with custom identity providers

AWS Transfer Family offers several options for custom identity providers to authenticate and authorize users for secure file transfers. Here are the main approaches:

- Custom identity provider solution—This topic describes the Transfer Family custom identity provider solution, using a toolkit hosted in GitHub.

  > **Note**
  >
  > For most use cases, this is the recommended option. Specifically, if you need to support more than 100 Active Directory groups, the custom identity provider solution offers a scalable alternative without group limitations. This solution is described in the blog post, Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

- Using Amazon API Gateway to integrate your identity provider—This topic describes how to use an AWS Lambda function to back an Amazon API Gateway method.

  You can provide a RESTful interface with a single Amazon API Gateway method. Transfer Family calls this method to connect to your identity provider, which authenticates and authorizes your users for access to Amazon S3 or Amazon EFS. Use this option if you need a RESTful API to integrate your identity provider or if you want to use AWS WAF to leverage its capabilities for geo-blocking or rate-limiting requests. For details, see Using Amazon API Gateway to integrate your identity provider.

- Dynamic permission management approaches—This topic describes approaches for managing user permissions dynamically using session policies.

  To authenticate your users, you can use your existing identity provider with AWS Transfer Family. You integrate your identity provider using an AWS Lambda function, which authenticates and authorizes your users for access to Amazon S3 or Amazon Elastic File System (Amazon EFS). For details, see Using AWS Lambda to integrate your identity provider. You can also access CloudWatch graphs for metrics such as number of files and bytes transferred in the AWS Transfer Family Management Console, giving you a single pane of glass to monitor file transfers using a centralized dashboard.

- Transfer Family provides a blog post and a workshop that walk you through building a file transfer solution. This solution leverages AWS Transfer Family for managed SFTP/FTPS endpoints and Amazon Cognito and DynamoDB for user management.

The blog post is available at Using Amazon Cognito as an identity provider with AWS Transfer Family and Amazon S3. You can view the details for the workshop here.

> **Note**
>
> For custom identity providers, the username must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A-Z, 0–9, underscore '_', hyphen '-', period '.' and at sign '@'. The username can't start with a hyphen '-', period '.' or at sign '@'.

When implementing a custom identity provider, consider the following best practices:

- Deploy the solution in the same AWS account and region as your Transfer Family servers.
- Implement the principle of least privilege when configuring IAM roles and policies.
- Use features like IP allow-listing and standardized logging for enhanced security.
- Test your custom identity provider thoroughly in a non-production environment before deployment.

Topics

- Custom identity provider solution
- Using AWS Lambda to integrate your identity provider
- Using Amazon API Gateway to integrate your identity provider
- Using multiple authentication methods
- IPv6 support for custom identity providers

Custom identity provider solution

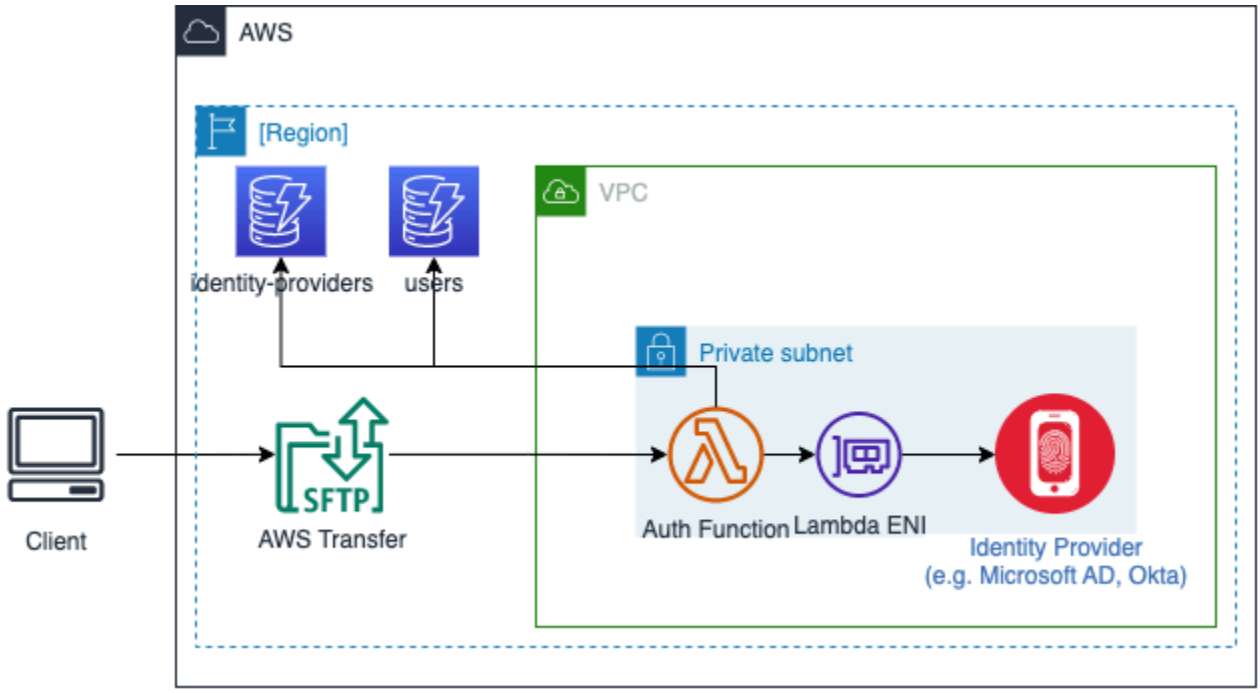
The AWS Transfer Family custom identity provider solution is a modular custom identity provider solution that solves for many common authentication and authorization use cases that enterprises have when implementing the service. This solution provides a reusable foundation for implementing custom identity providers with granular per-user session configuration and separates authentication and authorization logic, offering a flexible and easy-to-maintain foundation for various use cases.

With the AWS Transfer Family custom identity provider solution, you can address common enterprise authentication and authorization use cases. This modular solution offers:

- A reusable foundation for implementing custom identity providers
- Granular per-user session configuration
- Separated authentication and authorization logic

Implementation details for the custom identity toolkit

The solution provides a flexible and maintainable base for various use cases. To get started, review the toolkit at https://github.com/aws-samples/toolkit-for-aws-transfer-family, then follow the deployment instructions in the Getting started section.



Note

If you have previously used custom identity provider templates and examples, consider adopting this solution instead. Moving forward, provider-specific modules will standardize on this solution. Ongoing maintenance and feature enhancements will be applied to this solution.

This solution contains standard patterns for implementing a custom provider that accounts for details including logging and where to store the additional session metadata needed for AWS Transfer Family, such as the `HomeDirectoryDetails` parameter. This solution provides a reusable foundation for implementing custom identity providers with granular per-user session configuration, and decouples the identity provider authentication logic from the reusable logic that builds a configuration that is returned to Transfer Family to complete authentication and establish settings for the session.

The code and supporting resources for this solution are available at https://github.com/aws-samples/toolkit-for-aws-transfer-family.

The toolkit contains the following features:

- An AWS Serverless Application Model template that provisions the required resources. Optionally, deploy and configure Amazon API Gateway to incorporate AWS WAF, as described in the blog post Securing AWS Transfer Family with AWS Web Application Firewall and Amazon API Gateway.
- An Amazon DynamoDB schema to store configuration metadata about identity providers, including user session settings such as `HomeDirectoryDetails`, `Role`, and `Policy`.
- A modular approach that enables you to add new identity providers to the solution in the future, as modules.
- Attribute retrieval: Optionally retrieve IAM role and POSIX Profile (UID and GID) attributes from supported identity providers, including AD, LDAP, and Okta.
- Support for multiple identity providers connected to a single Transfer Family server and multiple Transfer Family servers using the same deployment of the solution.
- Built-in IP allow-list checking such as IP allow lists that can optionally be configured on a per-user or per-identity provider basis.
- Detailed logging with configurable log-level and tracing support to aid in troubleshooting.

Before you begin to deploy the custom identity provider solution, you need to have the following AWS resources.

- An Amazon Virtual Private Cloud (VPC) with private subnets, with internet connectivity through either a NAT gateway or a DynamoDB gateway endpoint.
- Appropriate IAM permissions to perform the following tasks:
  - Deploy the `custom-idp.yaml` CloudFormation template,

- Create AWS CodePipeline projects
- Create AWS CodeBuild projects
- Create IAM roles and policies

> **Important**
>
> You must deploy the solution to the same AWS account and AWS Region that contains your target Transfer Family servers.

Supported identity providers

The following list contains details for identity providers that are supported for the custom identity provider solution.

Provider	Password flows	Public key flows	Multi-factor	Attribute retrieval	Details
Active Directory and LDAP	Yes	Yes	No	Yes	User verification can be performed as part of public key authentication flow.
Argon2 (local hash)	Yes	No	No	No	Argon2 hashes are stored in the user record for 'local' password based authentication use cases.

Provider	Password flows	Public key flows	Multi-factor	Attribute retrieval	Details
Amazon Cognito	Yes	No	Yes*	No	Time-based One-Time Password (TOTP)-based multi-factor authentication only. *SMS-based MFA is not supported.
Entra ID (formerly Azure AD)	Yes	No	No	No	
Okta	Yes	Yes	Yes*	Yes	TOTP-based MFA only.
Public key	No	Yes	No	No	Public keys are stored in the user record in DynamoDB.
Secrets Manager	Yes	Yes	No	No	

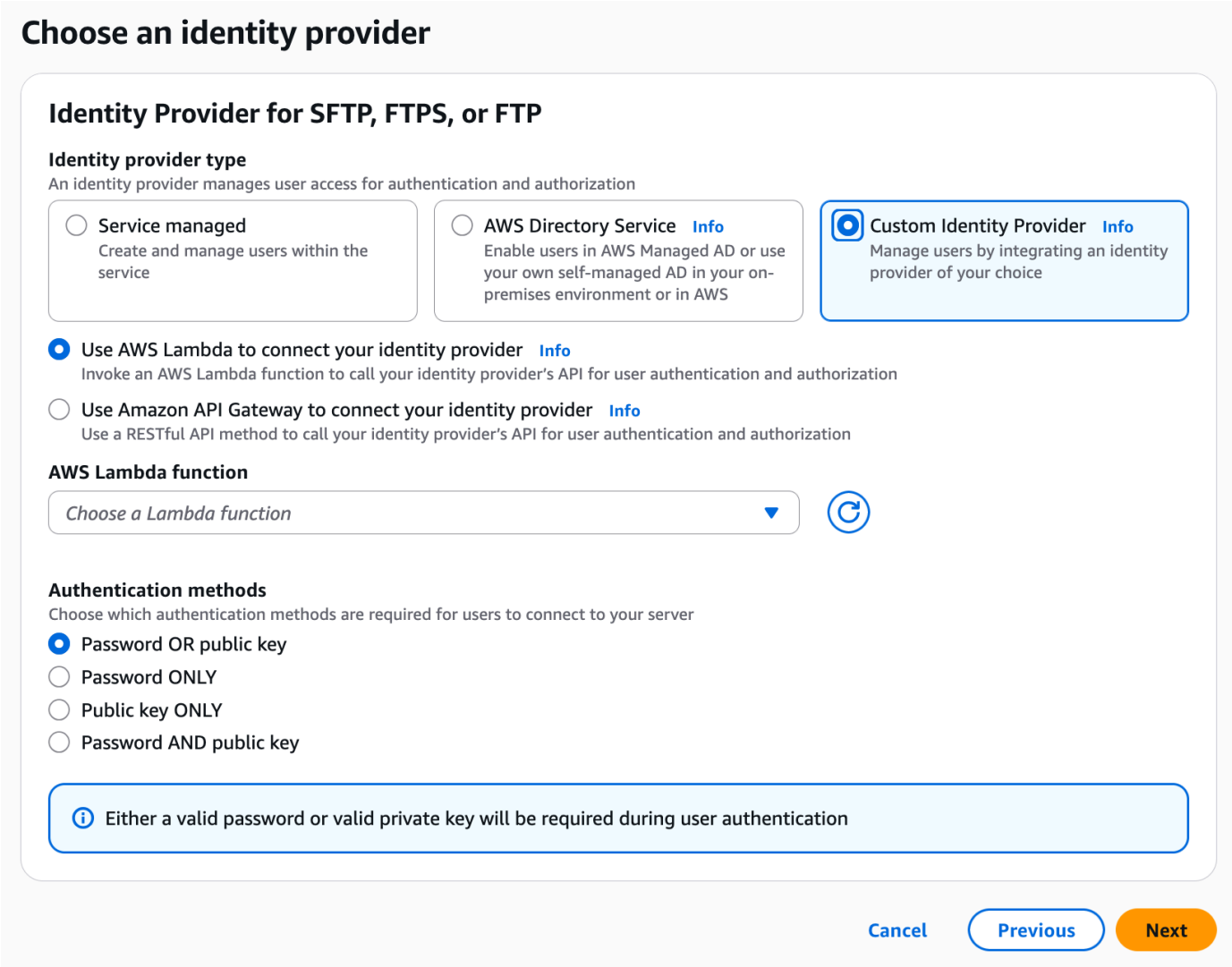
Using AWS Lambda to integrate your identity provider

This topic describes how to create an AWS Lambda function that connects to your custom identity provider. You can use any custom identity provider, such as Okta, Secrets Manager, OneLogin, or a custom data store that includes authorization and authentication logic.

For most use cases, the recommended way to configure a custom identity provider is to use the Custom identity provider solution.

> **Note**
>
> Before you create a Transfer Family server that uses Lambda as the identity provider, you must create the function. For an example Lambda function, see Example Lambda functions. Or, you can deploy a CloudFormation stack that uses one of the Lambda function templates. Also, make sure your Lambda function uses a resource-based policy that trusts Transfer Family. For an example policy, see Lambda resource-based policy.

1. Open the AWS Transfer Family console.
2. Choose **Create server** to open the **Create server** page. For **Choose an identity provider**, choose **Custom Identity Provider**, as shown in the following screenshot.



> **Note**
>
> The choice of authentication methods is only available if you enable SFTP as one of the protocols for your Transfer Family server.

3. Make sure the default value, **Use AWS Lambda to connect your identity provider**, is selected.
4. For **AWS Lambda function**, choose the name of your Lambda function.
5. Fill in the remaining boxes, and then choose **Create server**. For details on the remaining steps for creating a server, see Configuring an SFTP, FTPS, or FTP server endpoint.

Lambda resource-based policy

You must have a policy that references the Transfer Family server and Lambda ARNs. For example, you could use the following policy with your Lambda function that connects to your identity provider. The policy is escaped JSON as a string.

```
"Policy":
"{\"Version\":\"2012-10-17\",
\"Id\":\"default\",
\"Statement\":[
  {\"Sid\":\"AllowTransferInvocation\",
  \"Effect\":\"Allow\",
  \"Principal\":{\"Service\":\"transfer.amazonaws.com\"},
  \"Action\":\"lambda:InvokeFunction\",
  \"Resource\":\"arn:aws:lambda:region:123456789012:function:my-lambda-auth-
function\",
  \"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:transfer:region:123456789012:server/server-id\"}}}
]}"
```

> **Note**
>
> In the example policy above, replace each *user input placeholder* with your own information.

Event message structure

The event message structure from SFTP server sent to the authorizer Lambda function for a custom IDP is as follows.

```
{
    "username": "value",
    "password": "value",
    "protocol": "SFTP",
    "serverId": "s-abcd123456",
    "sourceIp": "192.168.0.100"
}
```

Where `username` and `password` are the values for the sign-in credentials that are sent to the server.

For example, you enter the following command to connect:

```
sftp bobusa@server_hostname
```

You are then prompted to enter your password:

```
Enter password:
    mysecretpassword
```

You can check this from your Lambda function by printing the passed event from within the Lambda function. It should look similar to the following text block.

```
{
    "username": "bobusa",
    "password": "mysecretpassword",
    "protocol": "SFTP",
    "serverId": "s-abcd123456",
    "sourceIp": "192.168.0.100"
}
```

The event structure is similar for FTP and FTPS: the only difference is those values are used for the `protocol` parameter, rather than SFTP.

Lambda functions for authentication

To implement different authentication strategies, edit the Lambda function. To help you meet your application's needs, you can deploy a CloudFormation stack. For more information about Lambda, see the AWS Lambda Developer Guide or Building Lambda functions with Node.js.

Topics

- Valid Lambda values
- Example Lambda functions
- Testing your configuration
- Lambda function templates

Valid Lambda values

The following table describes details for the values that Transfer Family accepts for Lambda functions that are used for custom identity providers.

Value	Description	Required
`Role`	Specifies the Amazon Resource Name (ARN) of the IAM role that controls your users' access to your Amazon S3 bucket or Amazon EFS file system. The policies attached to this role determine the level of access that you want to provide your users when transferring files into and out of your Amazon S3 or Amazon EFS file system. The IAM role should also contain a trust relationship that allows the server to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see To establish a trust relationship.	Required
`PosixProfile`	The full POSIX identity, including user ID (`Uid`), group ID (`Gid`), and any secondary group IDs (`SecondaryGids`), that controls your users' access to your Amazon EFS file systems. The POSIX permissions that are set on files and directories in your	Required for Amazon EFS backing storage

Value	Description	Required
	file system determine the level of access your users get when transferring files into and out of your Amazon EFS file systems.	
`PublicKeys`	A list of SSH public key values that are valid for this user. An empty list implies that this is not a valid login. Must not be returned during password authentication.	Optional
`Policy`	A session policy for your user so that you can use the same IAM role across multiple users. This policy scopes down user access to portions of their Amazon S3 bucket. For more information about using session policies with custom identity providers, see the session policy examples in this topic.	Optional

Value	Description	Required
`HomeDirectoryType`	The type of landing directory (folder) that you want your users' home directory to be when they log in to the server. • If you set it to `PATH`, the user sees the absolute Amazon S3 bucket or Amazon EFS paths as is in their file transfer protocol clients. • If you set it to `LOGICAL`, you must provide mappings in the `HomeDirectoryDetails` parameter to make Amazon S3 or Amazon EFS paths visible to your users.	Optional
`HomeDirectoryDetails`	Logical directory mappings that specify which Amazon S3 or Amazon EFS paths and keys should be visible to your user and how you want to make them visible. You must specify the `Entry` and `Target` pair, where `Entry` shows how the path is made visible and `Target` is the actual Amazon S3 or Amazon EFS path.	Required if `HomeDirectoryType` has a value of `LOGICAL`

Value	Description	Required
`HomeDirectory`	The landing directory for a user when they log in to the server using the client. The format depends on your storage backend: • **For Amazon S3:** `/bucket-name/user-home-directory` Example: `/amzn-s3-demo-bucket/users/john` • **For Amazon EFS:** `/fs-12345/user-home-directory` Example: `/fs-faa1a123/users/john` **Important** The bucket name or Amazon EFS file system ID must be included in the path. Omitting this information will result in "File not found" errors during file transfers.	Optional

Note

`HomeDirectoryDetails` is a string representation of a JSON map. This is in contrast to `PosixProfile`, which is an actual JSON map object, and `PublicKeys` which is a JSON array of strings. See the code examples for the language-specific details.

HomeDirectory Format Requirements

When using the `HomeDirectory` parameter, ensure you include the complete path format:

- **For Amazon S3 storage:** Always include the bucket name in the format `/bucket-name/path`
- **For Amazon EFS storage:** Always include the file system ID in the format `/fs-12345/path`

A common cause of "File not found" errors is omitting the bucket name or EFS file system ID from the `HomeDirectory` path. Setting `HomeDirectory` to just `/` without the storage identifier will cause authentication to succeed but file operations to fail.

Example Lambda functions

This section presents some example Lambda functions, in both NodeJS and Python.

Note

In these examples, the user, role, POSIX profile, password, and home directory details are all examples, and must be replaced with your actual values.

Logical home directory, NodeJS

The following NodeJS example function provides the details for a user that has a logical home directory.

```
// GetUserConfig Lambda

exports.handler = (event, context, callback) => {
```

```
  console.log("Username:", event.username, "ServerId: ", event.serverId);

  var response;
  // Check if the username presented for authentication is correct. This doesn't
 check the value of the server ID, only that it is provided.
  if (event.serverId !== "" && event.username == 'example-user') {
    var homeDirectoryDetails = [
      {
        Entry: "/",
        Target: "/fs-faa1a123"
      }
    ];
    response = {
      Role: 'arn:aws:iam::123456789012:role/transfer-access-role', // The user is
 authenticated if and only if the Role field is not blank
      PosixProfile: {"Gid": 65534, "Uid": 65534}, // Required for EFS access, but
 not needed for S3
      HomeDirectoryDetails: JSON.stringify(homeDirectoryDetails),
      HomeDirectoryType: "LOGICAL",
    };

    // Check if password is provided
    if (!event.password) {
      // If no password provided, return the user's SSH public key
      response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789" ];
    // Check if password is correct
    } else if (event.password !== 'Password1234') {
      // Return HTTP status 200 but with no role in the response to indicate
 authentication failure
      response = {};
    }
  } else {
    // Return HTTP status 200 but with no role in the response to indicate
 authentication failure
    response = {};
  }
  callback(null, response);
};
```

Path-based home directory, NodeJS

The following NodeJS example function provides the details for a user that has a path-based home directory.

```
// GetUserConfig Lambda

exports.handler = (event, context, callback) => {
  console.log("Username:", event.username, "ServerId: ", event.serverId);

  var response;
  // Check if the username presented for authentication is correct. This doesn't
 check the value of the server ID, only that it is provided.
  // There is also event.protocol (one of "FTP", "FTPS", "SFTP") and event.sourceIp
 (e.g., "127.0.0.1") to further restrict logins.
  if (event.serverId !== "" && event.username == 'example-user') {
    response = {
      Role: 'arn:aws:iam::123456789012:role/transfer-access-role', // The user is
 authenticated if and only if the Role field is not blank
      Policy: '', // Optional, JSON stringified blob to further restrict this user's
 permissions
      // HomeDirectory format depends on your storage backend:
      // For S3: '/bucket-name/user-home-directory' (e.g., '/my-transfer-bucket/
users/john')
      // For EFS: '/fs-12345/user-home-directory' (e.g., '/fs-faa1a123/users/john')
      HomeDirectory: '/my-transfer-bucket/users/example-user' // S3 example -
 replace with your bucket name
      // HomeDirectory: '/fs-faa1a123/users/example-user' // EFS example - uncomment
 for EFS
    };

    // Check if password is provided
    if (!event.password) {
      // If no password provided, return the user's SSH public key
     response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789" ];
    // Check if password is correct
    } else if (event.password !== 'Password1234') {
      // Return HTTP status 200 but with no role in the response to indicate
 authentication failure
      response = {};
    }
  } else {
    // Return HTTP status 200 but with no role in the response to indicate
 authentication failure
    response = {};
  }
  callback(null, response);
```

```
};
```

Logical home directory, Python

The following Python example function provides the details for a user that has a logical home directory.

```
# GetUserConfig Python Lambda with LOGICAL HomeDirectoryDetails
import json

def lambda_handler(event, context):
  print("Username: {}, ServerId: {}".format(event['username'], event['serverId']))

  response = {}

  # Check if the username presented for authentication is correct. This doesn't
 check the value of the server ID, only that it is provided.
  if event['serverId'] != '' and event['username'] == 'example-user':
    homeDirectoryDetails = [
      {
        'Entry': '/',
        'Target': '/fs-faa1a123'
      }
    ]
    response = {
      'Role': 'arn:aws:iam::123456789012:role/transfer-access-role', # The user will
 be authenticated if and only if the Role field is not blank
      'PosixProfile': {"Gid": 65534, "Uid": 65534}, # Required for EFS access, but
 not needed for S3
      'HomeDirectoryDetails': json.dumps(homeDirectoryDetails),
      'HomeDirectoryType': "LOGICAL"
    }

    # Check if password is provided
    if event.get('password', '') == '':
      # If no password provided, return the user's SSH public key
     response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789" ]
    # Check if password is correct
    elif event['password'] != 'Password1234':
      # Return HTTP status 200 but with no role in the response to indicate
 authentication failure
      response = {}
```

```
  else:
    # Return HTTP status 200 but with no role in the response to indicate
 authentication failure
    response = {}

  return response
```

Path-based home directory, Python

The following Python example function provides the details for a user that has a path-based home directory.

```
# GetUserConfig Python Lambda with PATH HomeDirectory

def lambda_handler(event, context):
  print("Username: {}, ServerId: {}".format(event['username'], event['serverId']))

  response = {}

  # Check if the username presented for authentication is correct. This doesn't
 check the value of the server ID, only that it is provided.
  # There is also event.protocol (one of "FTP", "FTPS", "SFTP") and event.sourceIp
 (e.g., "127.0.0.1") to further restrict logins.
  if event['serverId'] != '' and event['username'] == 'example-user':
    response = {
      'Role': 'arn:aws:iam::123456789012:role/transfer-access-role', # The user will
 be authenticated if and only if the Role field is not blank
      'Policy': '', #  Optional, JSON stringified blob to further restrict this
 user's permissions
      # HomeDirectory format depends on your storage backend:
      # For S3: '/bucket-name/user-home-directory' (e.g., '/my-transfer-bucket/
users/john')
      # For EFS: '/fs-12345/user-home-directory' (e.g., '/fs-faa1a123/users/john')
      'HomeDirectory': '/my-transfer-bucket/users/example-user', # S3 example -
 replace with your bucket name
      # 'HomeDirectory': '/fs-faa1a123/users/example-user', # EFS example -
 uncomment for EFS
      'HomeDirectoryType': "PATH" # Not strictly required, defaults to PATH
    }

    # Check if password is provided
    if event.get('password', '') == '':
      # If no password provided, return the user's SSH public key
```

```
     response['PublicKeys'] = [ "ssh-
rsa abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789" ]
    # Check if password is correct
    elif event['password'] != 'Password1234':
      # Return HTTP status 200 but with no role in the response to indicate
 authentication failure
      response = {}
  else:
    # Return HTTP status 200 but with no role in the response to indicate
 authentication failure
    response = {}

  return response
```

Testing your configuration

After you create your custom identity provider, you should test your configuration.

Console

To test your configuration by using the AWS Transfer Family console

1. Open the AWS Transfer Family console.
2. On the **Servers** page, choose your new server, choose **Actions**, and then choose **Test**.
3. Enter the text for **Username** and **Password** that you set when you deployed the CloudFormation stack. If you kept the default options, the username is `myuser` and the password is `MySuperSecretPassword`.
4. Choose the **Server protocol** and enter the IP address for **Source IP**, if you set them when you deployed the CloudFormation stack.

CLI

To test your configuration by using the AWS CLI

1. Run the test-identity-provider command. Replace each *`user input placeholder`* with your own information, as described in the subsequent steps.

```
aws transfer test-identity-provider --server-id s-1234abcd5678efgh --user-
name myuser --user-password MySuperSecretPassword --server-protocol FTP --
source-ip 127.0.0.1
```

2. Enter the server ID.
3. Enter the username and password that you set when you deployed the CloudFormation stack. If you kept the default options, the username is `myuser` and the password is `MySuperSecretPassword`.
4. Enter the server protocol and source IP address, if you set them when you deployed the CloudFormation stack.

If user authentication succeeds, the test returns a `StatusCode: 200` HTTP response, an empty string `Message: ""` (which would contain a reason for failure otherwise), and a `Response` field.

> **Note**
>
> In the response example below, the `Response` field is a JSON object that has been "stringified" (converted into a flat JSON string that can be used inside a program), and contains the details of the user's roles and permissions.


```
{
    "Response":"{\"Policy\":\"{\\\"Version\\\":\\\"2012-10-17\\\",\\\"Statement\\\":
[{\\\"Sid\\\":\\\"ReadAndListAllBuckets\\\",\\\"Effect\\\":\\\"Allow\\\",\\\"Action\\
\":[\\\"s3:ListAllMybuckets\\\",\\\"s3:GetBucketLocation\\\",\\\"s3:ListBucket\\\",\\
\"s3:GetObjectVersion\\\",\\\"s3:GetObjectVersion\\\"],\\\"Resource\\\":\\\"*\\\"}]}\",
\"Role\":\"arn:aws:iam::000000000000:role/MyUserS3AccessRole\",\"HomeDirectory\":\"/
\"}",
    "StatusCode": 200,
    "Message": ""
}
```


Lambda function templates

You can deploy an CloudFormation stack that uses a Lambda function for authentication. We provide several templates that authenticate and authorize your users using sign-in credentials. You can modify these templates or AWS Lambda code to further customize user access.

> **Note**
>
> You can create a FIPS-enabled AWS Transfer Family server through CloudFormation by specifying a FIPS-enabled security policy in your template. Available security policies are described in Security policies for AWS Transfer Family servers

To create an CloudFormation stack to use for authentication

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Follow the instructions for deploying an CloudFormation stack from an existing template in Selecting a stack template in the *AWS CloudFormation User Guide*.
3. Use one of the following templates to create a Lambda function to use for authentication in Transfer Family.

   - Classic (Amazon Cognito) stack template

     A basic template for creating a AWS Lambda for use as a custom identity provider in AWS Transfer Family. It authenticates against Amazon Cognito for password-based authentication and public keys are returned from an Amazon S3 bucket if public key based authentication is used. After deployment, you can modify the Lambda function code to do something different.

   - AWS Secrets Manager stack template

     A basic template that uses AWS Lambda with an AWS Transfer Family server to integrate Secrets Manager as an identity provider. It authenticates against an entry in AWS Secrets Manager of the format `aws/transfer/`*`server-id`*`/`*`username`*. Additionally, the secret must hold the key-value pairs for all user properties returned to Transfer Family. After deployment, you can modify the Lambda function code to do something different.

   - Okta stack template: A basic template that uses AWS Lambda with an AWS Transfer Family server to integrate Okta as a custom identity provider.
   - Okta-mfa stack template: A basic template that uses AWS Lambda with an AWS Transfer Family server to integrate Okta, with Multi Factor Authentication, as a custom identity provider.
   - Azure Active Directory template: details for this stack are described in the blog post Authenticating to AWS Transfer Family with Azure Active Directory and AWS Lambda.

After the stack has been deployed, you can view details about it on the **Outputs** tab in the CloudFormation console.

Deploying one of these stacks is the easiest way to integrate a custom identity provider into the Transfer Family workflow.

Using Amazon API Gateway to integrate your identity provider

This topic describes how to use an AWS Lambda function to back an API Gateway method. Use this option if you need a RESTful API to integrate your identity provider or if you want to use AWS WAF to leverage its capabilities for geo-blocking or rate-limiting requests.

For most use cases, the recommended way to configure a custom identity provider is to use the Custom identity provider solution.

Limitations if using an API Gateway to integrate your identity provider

- This configuration does not support custom domains.
- This configuration does not support a private API Gateway URL.

If you need either of these, you can use Lambda as an identity provider, without API Gateway. For details, see Using AWS Lambda to integrate your identity provider.

Authenticating using an API Gateway method

You can create an API Gateway method for use as an identity provider for Transfer Family. This approach provides a highly secure way for you to create and provide APIs. With API Gateway, you can create an HTTPS endpoint so that all incoming API operations are transmitted with greater security. For more details about the API Gateway service, see the API Gateway Developer Guide.

API Gateway offers an authorization method named `AWS_IAM`, which gives you the same authentication based on AWS Identity and Access Management (IAM) that AWS uses internally. If you enable authentication with `AWS_IAM`, only callers with explicit permissions to call an API can reach that API's API Gateway method.

To use your API Gateway method as a custom identity provider for Transfer Family, enable IAM for your API Gateway method. As part of this process, you provide an IAM role with permissions for Transfer Family to use your gateway.

Note

To improve security, you can configure a web application firewall. AWS WAF is a web application firewall that lets you monitor the HTTP and HTTPS requests that are forwarded to an Amazon API Gateway. For details, see Add a web application firewall.

Do not enable API Gateway caching

Do not enable caching for your API Gateway method when using it as a custom identity provider for Transfer Family. Caching is inappropriate and invalid for authentication requests because:

- Each authentication request is unique and requires a live response, not a cached response
- Caching provides no benefits since Transfer Family never sends duplicate or repeated requests to the API Gateway
- Enabling caching will cause the API Gateway to respond with mismatched data, resulting in invalid responses to authentication requests

To use your API Gateway method for custom authentication with Transfer Family

1. Create an CloudFormation stack. To do this:

   Note

   The stack templates have been updated to use BASE64-encoded passwords: for details, see Improvements to the CloudFormation templates.

   a. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.

   b. Follow the instructions for deploying an CloudFormation stack from an existing template in Selecting a stack template in the *AWS CloudFormation User Guide*.

   c. Use one of the following basic templates to create an AWS Lambda-backed API Gateway method for use as a custom identity provider in Transfer Family.

      - Basic stack template

By default, your API Gateway method is used as a custom identity provider to authenticate a single user in a single server using a hard-coded SSH (Secure Shell) key or password. After deployment, you can modify the Lambda function code to do something different.

- AWS Secrets Manager stack template

  By default, your API Gateway method authenticates against an entry in Secrets Manager of the format `aws/transfer/`*`server-id`*`/`*`username`*. Additionally, the secret must hold the key-value pairs for all user properties returned to Transfer Family. After deployment, you can modify the Lambda function code to do something different. For more information, see the blog postEnable password authentication for AWS Transfer Family using AWS Secrets Manager.

- Okta stack template

  Your API Gateway method integrates with Okta as a custom identity provider in Transfer Family. For more information, see the blog post Using Okta as an identity provider with AWS Transfer Family.

Deploying one of these stacks is the easiest way to integrate a custom identity provider into the Transfer Family workflow. Each stack uses the Lambda function to support your API method based on API Gateway. You can then use your API method as a custom identity provider in Transfer Family. By default, the Lambda function authenticates a single user called `myuser` with a password of `MySuperSecretPassword`. After deployment, you can edit these credentials or update the Lambda function code to do something different.

> **Important**
>
> We recommend that you edit the default user and password credentials.

After the stack has been deployed, you can view details about it on the **Outputs** tab in the CloudFormation console. These details include the stack's Amazon Resource Name (ARN), the ARN of the IAM role that the stack created, and the URL for your new gateway.

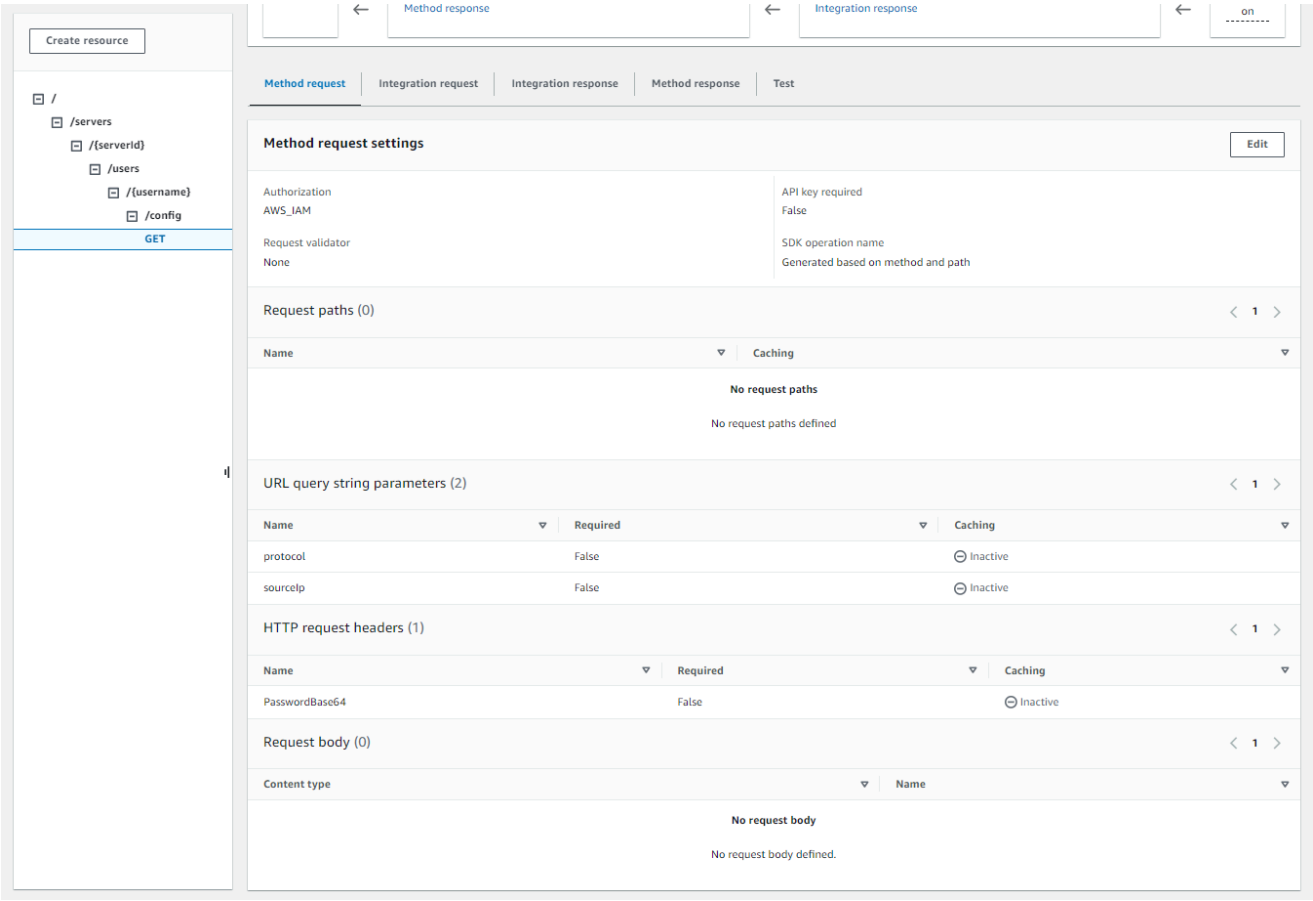
> **Note**
>
> If you are using the custom identity provider option to enable password–based authentication for your users, and you enable the request and response logging provided by API Gateway, API Gateway logs your users' passwords to your Amazon CloudWatch Logs. We don't recommend using this log in your production environment. For more information, see Set up CloudWatch API logging in API Gateway in the *API Gateway Developer Guide*.

2. Check the API Gateway method configuration for your server. To do this:

   a. Open the API Gateway console at https://console.aws.amazon.com/apigateway/.

   b. Choose the **Transfer Custom Identity Provider basic template API** that the CloudFormation template generated. You might need to select your region to see your gateways.

   c. In the **Resources** pane, choose **GET**. The following screenshot shows the correct method configuration.

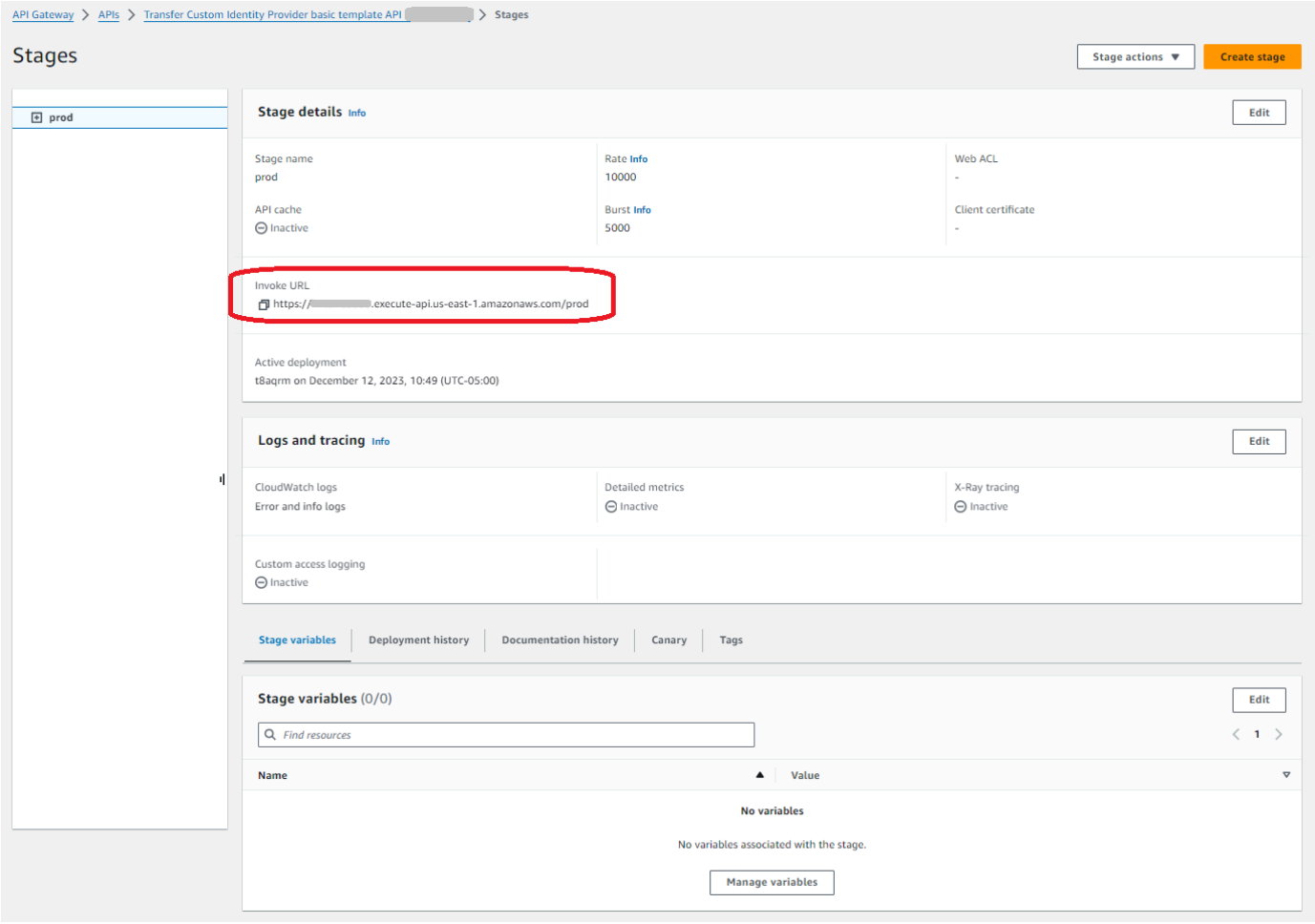


At this point, your API gateway is ready to be deployed.

3. For **Actions**, choose **Deploy API**. For **Deployment stage**, choose **prod**, and then choose **Deploy**.

   After the API Gateway method is successfully deployed, view its performance in **Stages** > **Stage details**, as shown in the following screenshot.

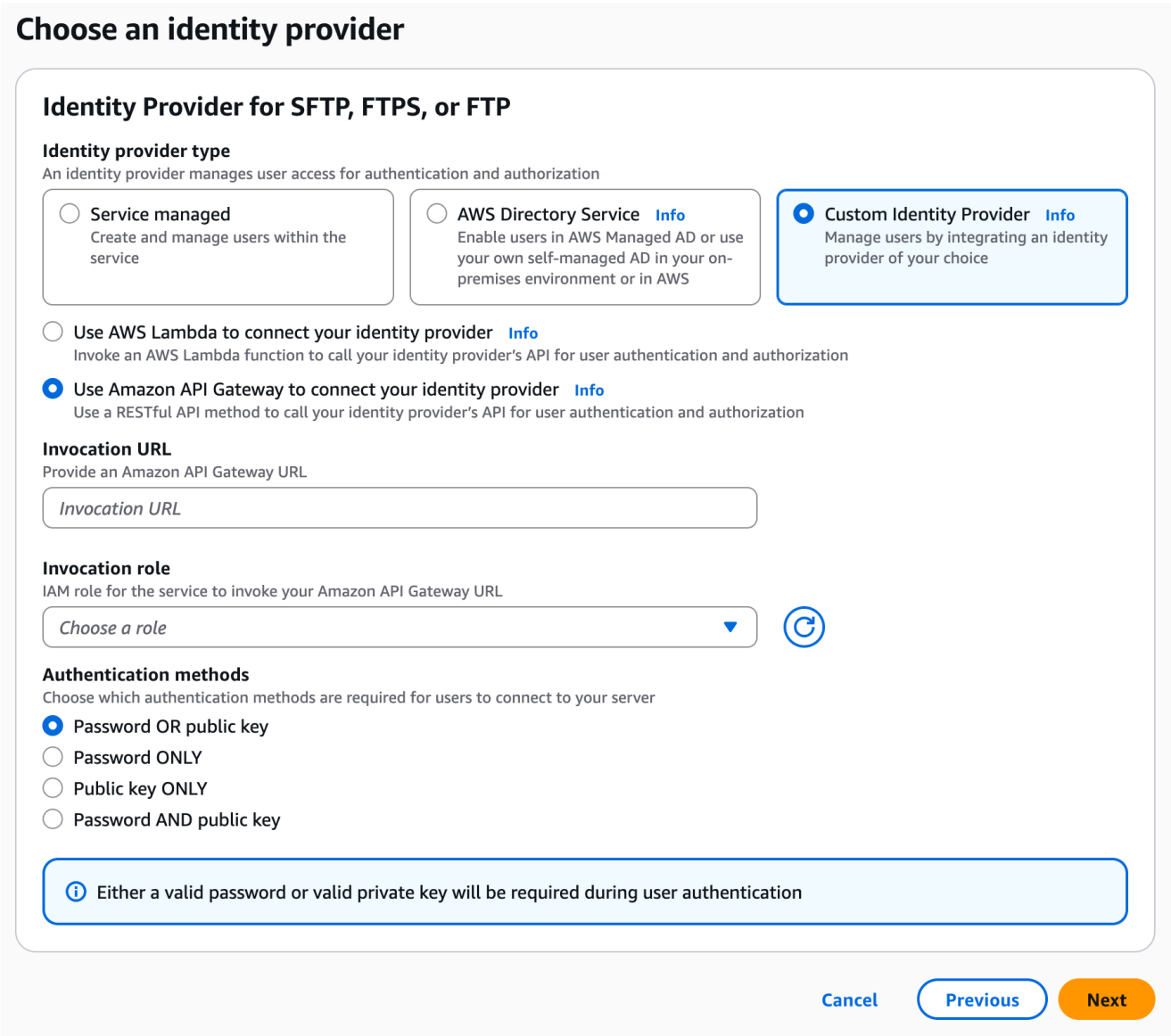
> **Note**
>
> Copy the **Invoke URL** address that appears at the top of the screen. You might need it for the next step.



4. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.

5. A Transfer Family should have been created for you, when you created the stack. If not, configure your server using these steps.

   a. Choose **Create server** to open the **Create server** page. For **Choose an identity provider**, choose **Custom**, then select **Use Amazon API Gateway to connect to your identity provider**, as shown in the following screenshot.



   b. In the **Provide an Amazon API Gateway URL** text box, paste the **Invoke URL** address of the API Gateway endpoint that you created in step 3 of this procedure.

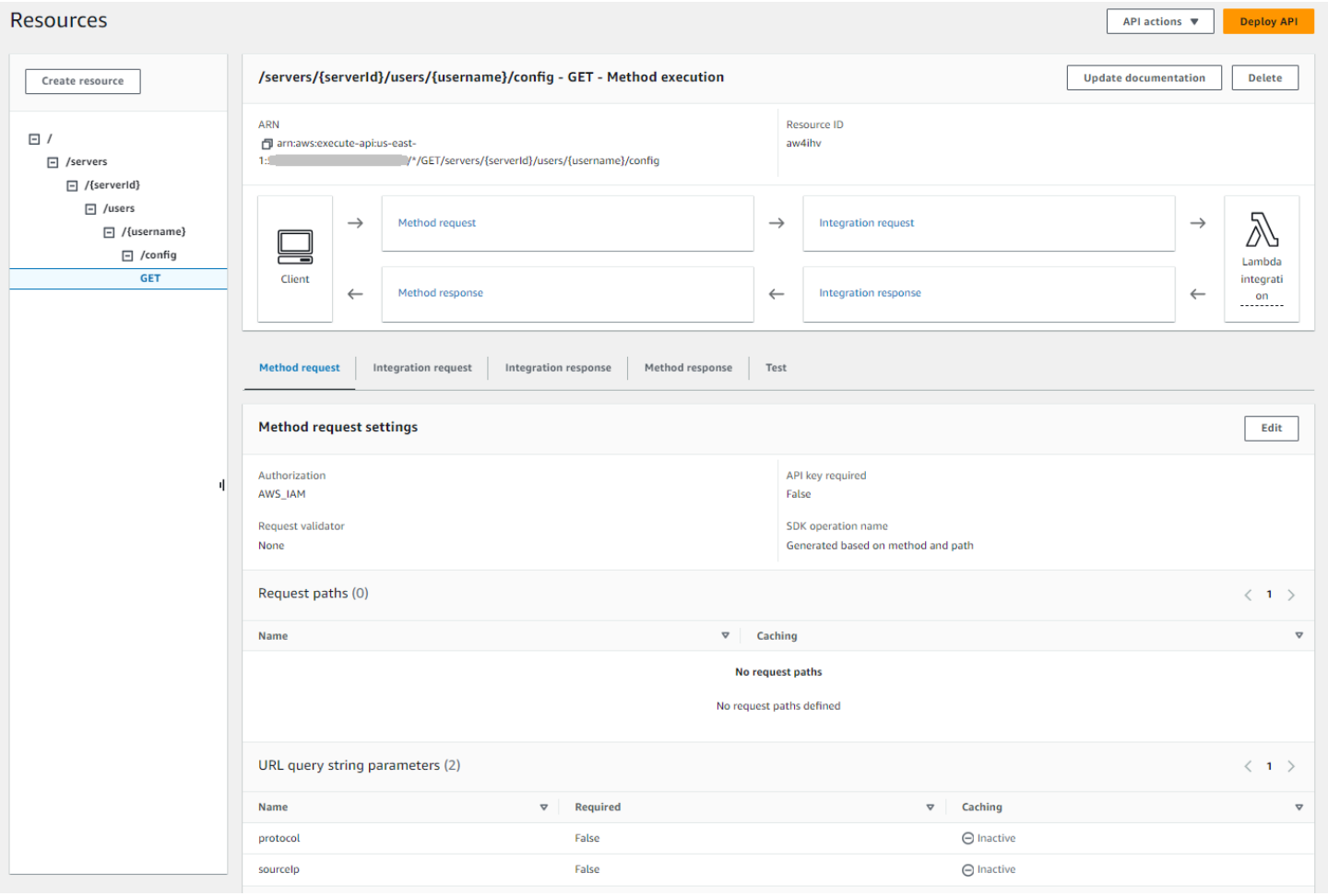
   c. For **Role**, choose the IAM role that was created by the CloudFormation template. This role allows Transfer Family to invoke your API gateway method.

The invocation role contains the CloudFormation stack name that you selected for the stack that you created in step 1. It has the following format: *`CloudFormation-stack-name`*`-TransferIdentityProviderRole-`*`ABC123DEF456GHI`*.

d. Fill in the remaining boxes, and then choose **Create server**. For details on the remaining steps for creating a server, see Configuring an SFTP, FTPS, or FTP server endpoint.

Implementing your API Gateway method

To create a custom identity provider for Transfer Family, your API Gateway method must implement a single method that has a resource path of `/servers/`*`serverId`*`/users/`*`username`*`/config`. The *`serverId`* and *`username`* values come from the RESTful resource path. Also, add `sourceIp` and `protocol` as **URL Query String Parameters** in the **Method Request**, as shown in the following image.



> **Note**
>
> The username must be a minimum of 3 and a maximum of 100 characters. You can use the following characters in the username: a–z, A-Z, 0–9, underscore '_', hyphen '-', period '.' and at sign '@'. The username can't start with a hyphen '-', period '.' or at sign '@'.

If Transfer Family attempts password authentication for your user, the service supplies a `Password:` header field. In the absence of a `Password:` header, Transfer Family attempts public key authentication to authenticate your user.

When you are using an identity provider to authenticate and authorize end users, in addition to validating their credentials, you can allow or deny access requests based on the IP addresses of the clients used by your end users. You can use this feature to ensure that data stored in your S3 buckets or your Amazon EFS file system can be accessed over the supported protocols only from IP addresses that you have specified as trusted. To enable this feature, you must include `sourceIp` in the Query string.

If you have multiple protocols enabled for your server and want to provide access using the same username over multiple protocols, you can do so as long as the credentials specific to each protocol have been set up in your identity provider. To enable this feature, you must include the *`protocol`* value in the RESTful resource path.

Your API Gateway method should always return HTTP status code `200`. Any other HTTP status code means that there was an error accessing the API.

Amazon S3 example response

The example response body is a JSON document of the following form for Amazon S3.

```
{
 "Role": "IAM role with configured S3 permissions",
 "PublicKeys": [
     "ssh-rsa public-key1",
     "ssh-rsa public-key2"
  ],
 "Policy": "STS Assume role session policy",
 "HomeDirectory": "/amzn-s3-demo-bucket/path/to/home/directory"
}
```

Note

The policy is escaped JSON as a string. For example:

```
"Policy":
"{
  \"Version\": \"2012-10-17\",
  \"Statement\":
     [
     {\"Condition\":
        {\"StringLike\":
            {\"s3:prefix\":
               [\"user/*\", \"user/\"]}},
     \"Resource\": \"arn:aws:s3:::amzn-s3-demo-bucket\",
     \"Action\": \"s3:ListBucket\",
     \"Effect\": \"Allow\",
     \"Sid\": \"ListHomeDir\"},
     {\"Resource\": \"arn:aws:s3:::*\",
        \"Action\": [\"s3:PutObject\",
        \"s3:GetObject\",
        \"s3:DeleteObjectVersion\",
        \"s3:DeleteObject\",
        \"s3:GetObjectVersion\",
        \"s3:GetObjectACL\",
        \"s3:PutObjectACL\"],
     \"Effect\": \"Allow\",
     \"Sid\": \"HomeDirObjectAccess\"}]
}"
```

The following example response shows that a user has a logical home directory type.

```
{
   "Role": "arn:aws:iam::123456789012:role/transfer-access-role-s3",
   "HomeDirectoryType":"LOGICAL",
   "HomeDirectoryDetails":"[{\"Entry\":\"/\",\"Target\":\"/amzn-s3-demo-bucket1\"}]",
   "PublicKeys":[""]
}
```

Amazon EFS example response

The example response body is a JSON document of the following form for Amazon EFS.

```
{
 "Role": "IAM role with configured EFS permissions",
 "PublicKeys": [
     "ssh-rsa public-key1",
     "ssh-rsa public-key2"
  ],
 "PosixProfile": {
   "Uid": "POSIX user ID",
   "Gid": "POSIX group ID",
   "SecondaryGids": [Optional list of secondary Group IDs],
 },
 "HomeDirectory": "/fs-id/path/to/home/directory"
}
```

The `Role` field shows that successful authentication occurred. When doing password authentication (when you supply a `Password:` header), you don't need to provide SSH public keys. If a user can't be authenticated, for example, if the password is incorrect, your method should return a response without `Role` set. An example of such a response is an empty JSON object.

The following example response shows a user that has a logical home directory type.

```
{
    "Role": "arn:aws:iam::123456789012:role/transfer-access-role-efs",
    "HomeDirectoryType": "LOGICAL",
    "HomeDirectoryDetails":"[{\"Entry\":\"/\",\"Target\":\"/faa1a123\"}]",
    "PublicKeys":[""],
    "PosixProfile":{"Uid":65534,"Gid":65534}
}
```

You can include user policies in the Lambda function in JSON format. For more information about configuring user policies in Transfer Family, see Managing access controls.

Default Lambda function

To implement different authentication strategies, edit the Lambda function that your gateway uses. To help you meet your application's needs, you can use the following example Lambda functions in Node.js. For more information about Lambda, see the AWS Lambda Developer Guide or Building Lambda functions with Node.js.

The following example Lambda function takes your username, password (if you're performing password authentication), server ID, protocol, and client IP address. You can use a combination of these inputs to look up your identity provider and determine if the login should be accepted.

> **Note**
>
> If you have multiple protocols enabled for your server and want to provide access using the same username over multiple protocols, you can do so as long as the credentials specific to the protocol have been set up in your identity provider.
> For File Transfer Protocol (FTP), we recommend maintaining separate credentials from Secure Shell (SSH) File Transfer Protocol (SFTP) and File Transfer Protocol over SSL (FTPS). We recommend maintaining separate credentials for FTP because, unlike SFTP and FTPS, FTP transmits credentials in clear text. By isolating FTP credentials from SFTP or FTPS, if FTP credentials are shared or exposed, your workloads using SFTP or FTPS remain secure.

This example function returns the role and logical home directory details, along with the public keys (if it performs public key authentication).

When you create service-managed users, you set their home directory, either logical or physical. Similarly, we need the Lambda function results to convey the desired user physical or logical directory structure. The parameters you set depend on the value for the HomeDirectoryType field.

- `HomeDirectoryType` set to `PATH` – the `HomeDirectory` field must then be an absolute Amazon S3 bucket prefix or Amazon EFS absolute path that is visible to your users.
- `HomeDirectoryType` set to `LOGICAL` – Do *not* set a `HomeDirectory` field. Instead, we set a `HomeDirectoryDetails` field that provides the desired Entry/Target mappings, similar to the described values in the HomeDirectoryDetails parameter for service-managed users.

The example functions are listed in Example Lambda functions.

Lambda function for use with AWS Secrets Manager

To use AWS Secrets Manager as your identity provider, you can work with the Lambda function in the sample CloudFormation template. The Lambda function queries the Secrets Manager service with your credentials and, if successful, returns a designated secret. For more information about Secrets Manager, see the AWS Secrets Manager User Guide.

To download a sample CloudFormation template that uses this Lambda function, go to the Amazon S3 bucket provided by AWS Transfer Family.

Improvements to the CloudFormation templates

Improvements to the API Gateway interface have been made to the published CloudFormation templates. The templates now use BASE64-encoded passwords with the API Gateway. Your existing deployments continue to work without this enhancement, but don't allow for passwords with characters outside the basic US-ASCII character set.

The changes in the template that enable this capability are as follows:

- The `GetUserConfigRequest AWS::ApiGateway::Method` resource has to have this `RequestTemplates` code (the line in italics is the updated line)

```
RequestTemplates:
   application/json: |
   {
      "username": "$util.urlDecode($input.params('username'))",
      "password":
 "$util.escapeJavaScript($util.base64Decode($input.params('PasswordBase64'))).replaceAll("\
\'","'")",
      "protocol": "$input.params('protocol')",
      "serverId": "$input.params('serverId')",
      "sourceIp": "$input.params('sourceIp')"
}
```

- The `RequestParameters` for the `GetUserConfig` resource must change to use the `PasswordBase64` header (the line in italics is the updated line):

```
RequestParameters:
   method.request.header.PasswordBase64: false
   method.request.querystring.protocol: false
   method.request.querystring.sourceIp: false
```

To check if the template for your stack is the latest

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. From the list of stacks, choose your stack.
3. From the details panel, choose the **Template** tab.

4. Look for the following:

   - Search for `RequestTemplates`, and make sure you have this line:

     ```
     "password":
      "$util.escapeJavaScript($util.base64Decode($input.params('PasswordBase64'))).replaceAll(
     \'","'")",
     ```

   - Search for `RequestParameters`, and make sure you have this line:

     ```
     method.request.header.PasswordBase64: false
     ```

If you don't see the updated lines, edit your stack. For details on how to update your CloudFormation stack, see Modifying a stack template in the *AWS CloudFormation; User Guide*.

Using multiple authentication methods

The Transfer Family server controls the AND logic when you use multiple authentication methods. Transfer Family treats this as two separate requests to your custom identity provider: however, their effect is combined.

Both requests must return successfully with the correct response to allow the authentication to complete. Transfer Family requires the two responses to be complete, meaning they contain all of the required elements (role, home directory, policy and the POSIX profile if you're using Amazon EFS for storage). Transfer Family also requires that the password response must not include public keys.

The public key request must have a separate response from the identity provider. That behavior is unchanged when using **Password OR Key** or **Password AND Key**.

The SSH/SFTP protocol challenges the software client first with a public key authentication, then requests a password authentication. This operation mandates both are successful before the user is allowed to complete the authentication.

For custom identity provider options, you can specify any of the following options for how to authenticate.

- **Password OR Key** – users can authenticate with either their password or their key. This is the default value.

- **Password ONLY** – users must provide their password to connect.
- **Key ONLY** – users must provide their private key to connect.
- **Password AND Key** – users must provide both their private key and their password to connect. The server checks the key first, and then if the key is valid, the system prompts for a password. If the private key provided does not match the public key that is stored, authentication fails.

IPv6 support for custom identity providers

AWS Transfer Family custom identity providers fully support IPv6 connections. When implementing a custom identity provider, your Lambda function can receive and process authentication requests from both IPv4 and IPv6 clients without any additional configuration. The Lambda function receives the client's IP address in the `sourceIp` field of the request, which can be either an IPv4 address (for example, `203.0.113.42`) or an IPv6 address (for example, `2001:db8:85a3:8d3:1319:8a2e:370:7348`). Your custom identity provider implementation should handle both address formats appropriately.

> **Important**
>
> If your custom identity provider performs IP-based validation or logging, ensure your implementation properly handles IPv6 address formats. IPv6 addresses are longer than IPv4 addresses and use a different notation format.

> **Note**
>
> When handling IPv6 addresses in your custom identity provider, ensure you're using proper IPv6 address parsing functions rather than simple string comparisons. IPv6 addresses can be represented in various canonical formats (for example `fd00:b600::ec2` or `fd00:b600:0:0:0:0:0:ec2`). Use appropriate IPv6 address libraries or functions in your implementation language to correctly validate and compare IPv6 addresses.

Example Handling both IPv4 and IPv6 addresses in a custom identity provider

```
def lambda_handler(event, context):
    # Extract the source IP address from the request
```

```
    source_ip = event.get('sourceIp', '')

    # Log the client IP address (works for both IPv4 and IPv6)
    print(f"Authentication request from: {source_ip}")

    # Example of IP-based validation that works with both IPv4 and IPv6
    if is_ip_allowed(source_ip):
        # Continue with authentication
        # ...
    else:
        # Reject the authentication request
        return {
            "Role": "",
            "HomeDirectory": "",
            "Status": "DENIED"
        }
```

For more information about implementing custom identity providers, see Using AWS Lambda to integrate your identity provider.

Using AWS Directory Service for Microsoft Active Directory

You can use AWS Transfer Family to authenticate your file transfer end users using AWS Directory Service for Microsoft Active Directory. It enables seamless migration of file transfer workflows that rely on Active Directory authentication without changing end users' credentials or needing a custom authorizer.

With AWS Managed Microsoft AD, you can securely provide Directory Service users and groups access over SFTP, FTPS, and FTP for data stored in Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS). If you use Active Directory to store your users' credentials, you now have an easier way to enable file transfers for these users.

You can provide access to Active Directory groups in AWS Managed Microsoft AD in your on-premises environment or in the AWS Cloud using Active Directory connectors. You can give users that are already configured in your Microsoft Windows environment, either in the AWS Cloud or in their on-premises network, access to an AWS Transfer Family server that uses AWS Managed Microsoft AD for identity. The AWS storage blog contains a post that details a solution for using Active Directory with Transfer Family: Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

Note

- AWS Transfer Family does not support Simple AD.
- Transfer Family does not support cross-region Active Directory configurations: we only support Active Directory integrations that are in the same region as that of the Transfer Family server.
- Transfer Family does not support using either AWS Managed Microsoft AD or AD Connector to enable multi-factor authentication (MFA) for your existing RADIUS-based MFA infrastructure.
- AWS Transfer Family does not support replicated regions of Managed Active Directory.

To use AWS Managed Microsoft AD, you must perform the following steps:

1. Create one or more AWS Managed Microsoft AD directories using the Directory Service console.
2. Use the Transfer Family console to create a server that uses AWS Managed Microsoft AD as its identity provider.
3. Set up AWS Directory using an Active Directory Connector.
4. Add access from one or more of your Directory Service groups.
5. Although not required, we recommend that you test and verify user access.

Topics

- Before you start using AWS Directory Service for Microsoft Active Directory
- Working with Active Directory realms
- Choosing AWS Managed Microsoft AD as your identity provider
- Connecting to on-prem Microsoft Active Directory
- Granting access to groups
- Testing users
- Deleting server access for a group
- Connecting to the server using SSH (Secure Shell)
- Connecting AWS Transfer Family to a self-managed Active Directory using forests and trusts

Before you start using AWS Directory Service for Microsoft Active Directory

> **Note**
>
> AWS Transfer Family has a default limit of 100 Active Directory groups per server. If your use case requires more than 100 groups, consider using a custom identity provider solution as described in Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

Provide a unique identifier for your AD groups

Before you can use AWS Managed Microsoft AD, you must provide a unique identifier for each group in your Microsoft AD directory. You can use the security identifier (SID) for each group to do this. The users of the group that you associate have access to your Amazon S3 or Amazon EFS resources over the enabled protocols using AWS Transfer Family.

Use the following Windows PowerShell command to retrieve the SID for a group, replacing *YourGroupName* with the name of the group.

```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties * | Select
 SamAccountName,ObjectSid
```

> **Note**
>
> If you are using AWS Directory Service as your identity provider, and if `userPrincipalName` and `SamAccountName` have different values, AWS Transfer Family accepts the value in `SamAccountName`. Transfer Family does not accept the value specified in `userPrincipalName`.

Add Directory Service permissions to your role

You also need Directory Service API permissions to use AWS Directory Service as your identity provider. The following permissions are required or suggested:

- `ds:DescribeDirectories` is required for Transfer Family to look up the directory
- `ds:AuthorizeApplication` is required to add authorization for Transfer Family

- `ds:UnauthorizeApplication` is suggested to remove any resources that are provisionally created, in case something goes wrong during the server creation process

Add these permissions to the role you are using for creating your Transfer Family servers. For more details on these permissions, see Directory Service API permissions: Actions, resources, and conditions reference.

Working with Active Directory realms

When you are considering how to have your Active Directory users access AWS Transfer Family servers, keep in mind the user's realm, and their group's realm. Ideally, the user's realm and their group's realm should match. That is, both the user and the group are in the default realm, or both are in the trusted realm. If this is not the case, the user cannot be authenticated by Transfer Family.

You can test the user to ensure the configuration is correct. For details, see Testing users. If there is a problem with the user/group realm, you receive the error, No associated access found for user's groups.

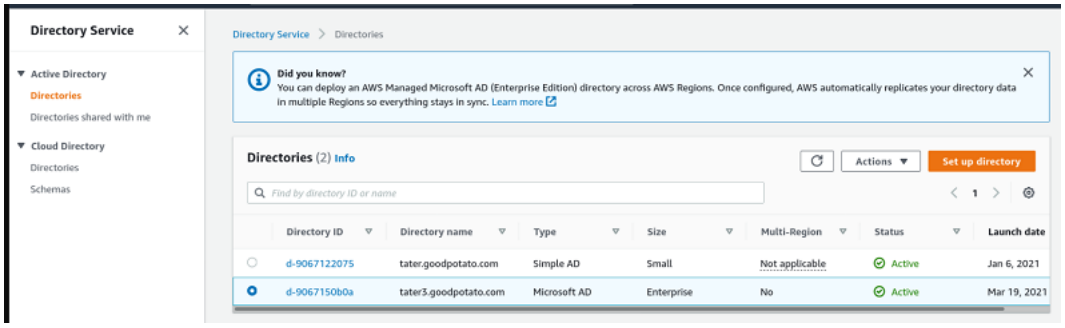
Choosing AWS Managed Microsoft AD as your identity provider

This section describes how to use AWS Directory Service for Microsoft Active Directory with a server.

To use AWS Managed Microsoft AD with Transfer Family

1. Sign in to the AWS Management Console and open the Directory Service console at https://console.aws.amazon.com/directoryservicev2/.

   Use the Directory Service console to configure one or more managed directories. For more information, see AWS Managed Microsoft AD in the *Directory Service Admin Guide*.

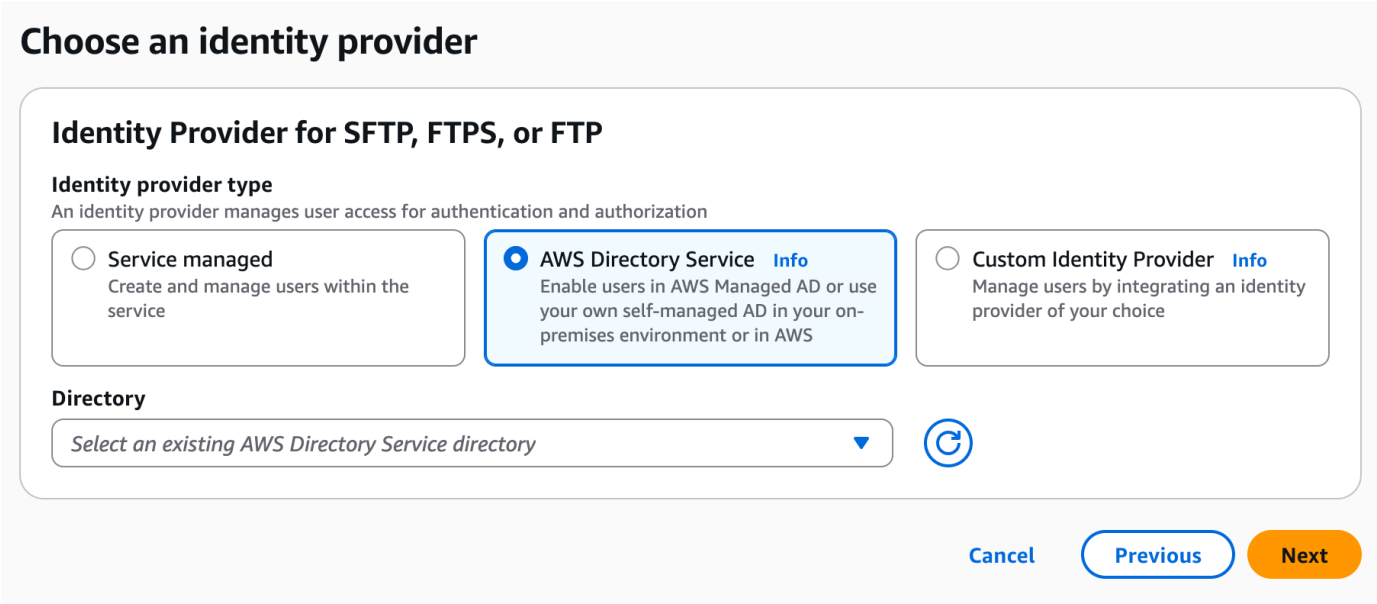


2. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/, and choose **Create server**.

3. On the **Choose protocols** page, choose one or more protocols from the list.

> **Note**
>
> If you select **FTPS**, you must provide the AWS Certificate Manager certificate.

4. For **Choose an identity provider**, choose **AWS Directory Service**.



5. The **Directory** list contains all the managed directories that you have configured. Choose a directory from the list, and choose **Next**.

> **Note**
>
> - Cross-Account and Shared directories are not supported for AWS Managed Microsoft AD.
> - To set up a server with Directory Service as your identity provider, you need to add some Directory Service permissions. For details, see Before you start using AWS Directory Service for Microsoft Active Directory.

6. To finish creating the server, use one of the following procedures:

- Create an SFTP-enabled server
- Create an FTPS-enabled server
- Create an FTP-enabled server

In those procedures, continue with the step that follows choosing an identity provider.

> **Important**
>
> You can't delete a Microsoft AD directory in Directory Service if you used it in a Transfer Family server. You must delete the server first, and then you can delete the directory.

Connecting to on-prem Microsoft Active Directory

This section describes how to set up an AWS Directory using an AD Connector

To set up your AWS Directory using AD Connector

1. Open the Directory Service console and select **Directories**.
2. Select **Set up directory**.
3. For directory type, choose **AD Connector**.
4. Select a directory size, select **Next**, then select your VPC and Subnets.
5. Select **Next**, then fill in the fields as follows:
   - **Directory DNS name**: enter the domain name you are using for your Microsoft Active Directory.
   - **DNS IP addresses**: enter you Microsoft Active Directory IP addresses.
   - **Server account username** and **password**: enter the details for the service account to use.
6. Complete the screens to create the directory service.

The next step is to create a Transfer Family server with the SFTP protocol, and the identity provider type of **AWS Directory Service**. From **Directory** drop down list, select the directory you added in the previous procedure.

Granting access to groups

After you create the server, you must choose which groups in the directory should have access to upload and download files over the enabled protocols using AWS Transfer Family. You do this by creating an *access*.

> **Note**
>
> AWS Transfer Family has a default limit of 100 Active Directory groups per server. If your use case requires more than 100 groups, consider using a custom identity provider solution as described in Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

> **Note**
>
> Users must belong *directly* to the group to which you are granting access. For example, assume that Bob is a user and belongs to groupA, and groupA itself is included in groupB.
>
> - If you grant access to groupA, Bob is granted access.
> - If you grant access to groupB (and not to groupA), Bob does not have access.

To grant access to a group

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. Navigate to your server details page.
3. In the **Accesses** section, choose **Add access**.
4. Enter the SID for the AWS Managed Microsoft AD directory that you want to have access to this server.

   > **Note**
   >
   > For information about how to find the SID for your group, see the section called "Before you start using AWS Directory Service for Microsoft Active Directory".

5. For **Access**, choose an AWS Identity and Access Management (IAM) role for the group.
6. In the **Policy** section, choose a policy. The default setting is **None**.
7. For **Home directory**, choose an Amazon S3 bucket that corresponds to the group's home directory.

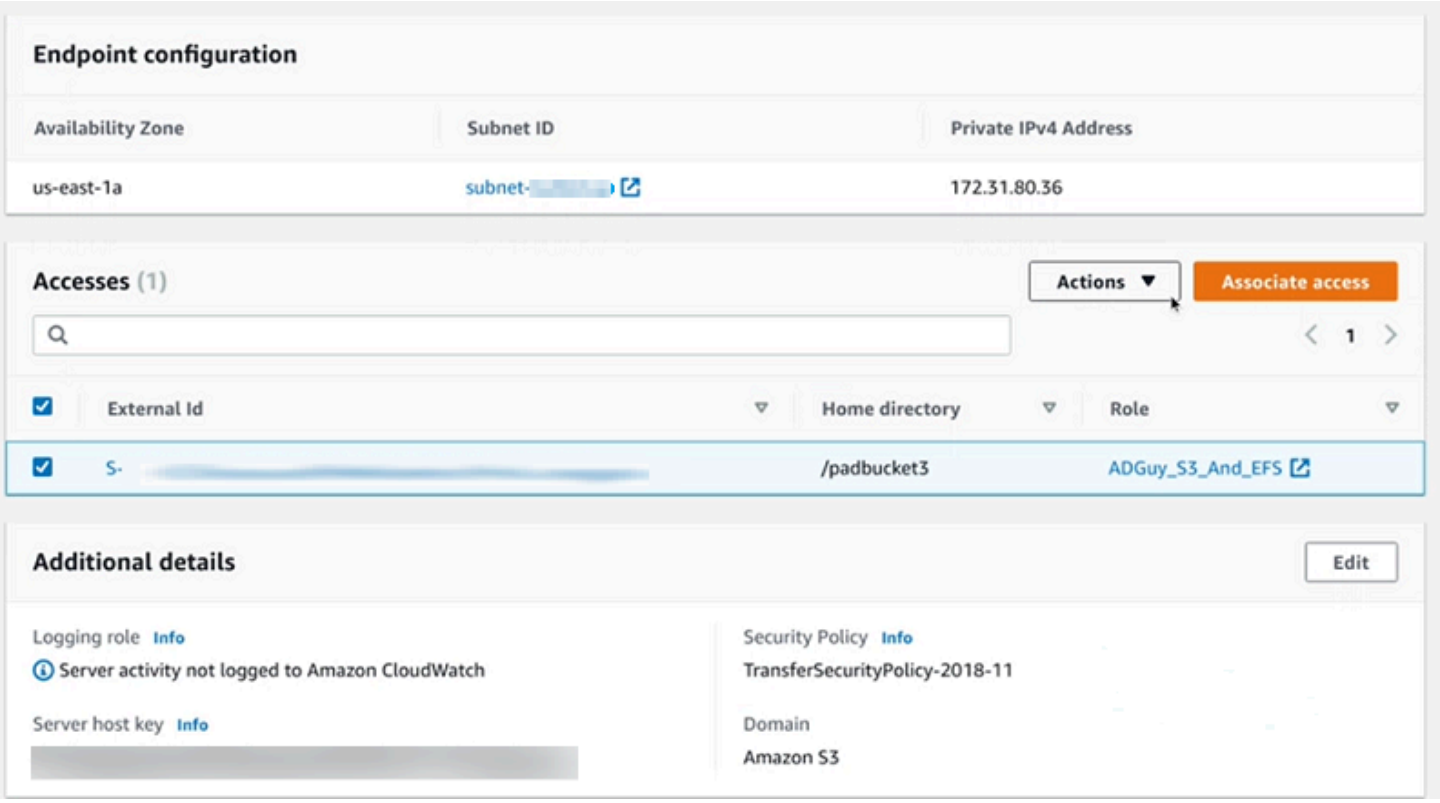
> **Note**
>
> You can limit the portions of the bucket that users see by creating a session policy. For example, to limit users to their own folder under the `/filetest` directory, enter the following text in the box.
>
> ```
> /filetest/${transfer:UserName}
> ```
>
> To learn more about creating a session policy, see Creating a session policy for an Amazon S3 bucket.

8. Choose **Add** to create the association.
9. Choose your server.
10. Choose **Add access**.
    - Enter the SID for the group.

    > **Note**
    >
    > For information about how to find the SID, see the section called "Before you start using AWS Directory Service for Microsoft Active Directory".

11. Choose **Add access**.

In the **Accesses** section, the accesses for the server are listed.



Testing users

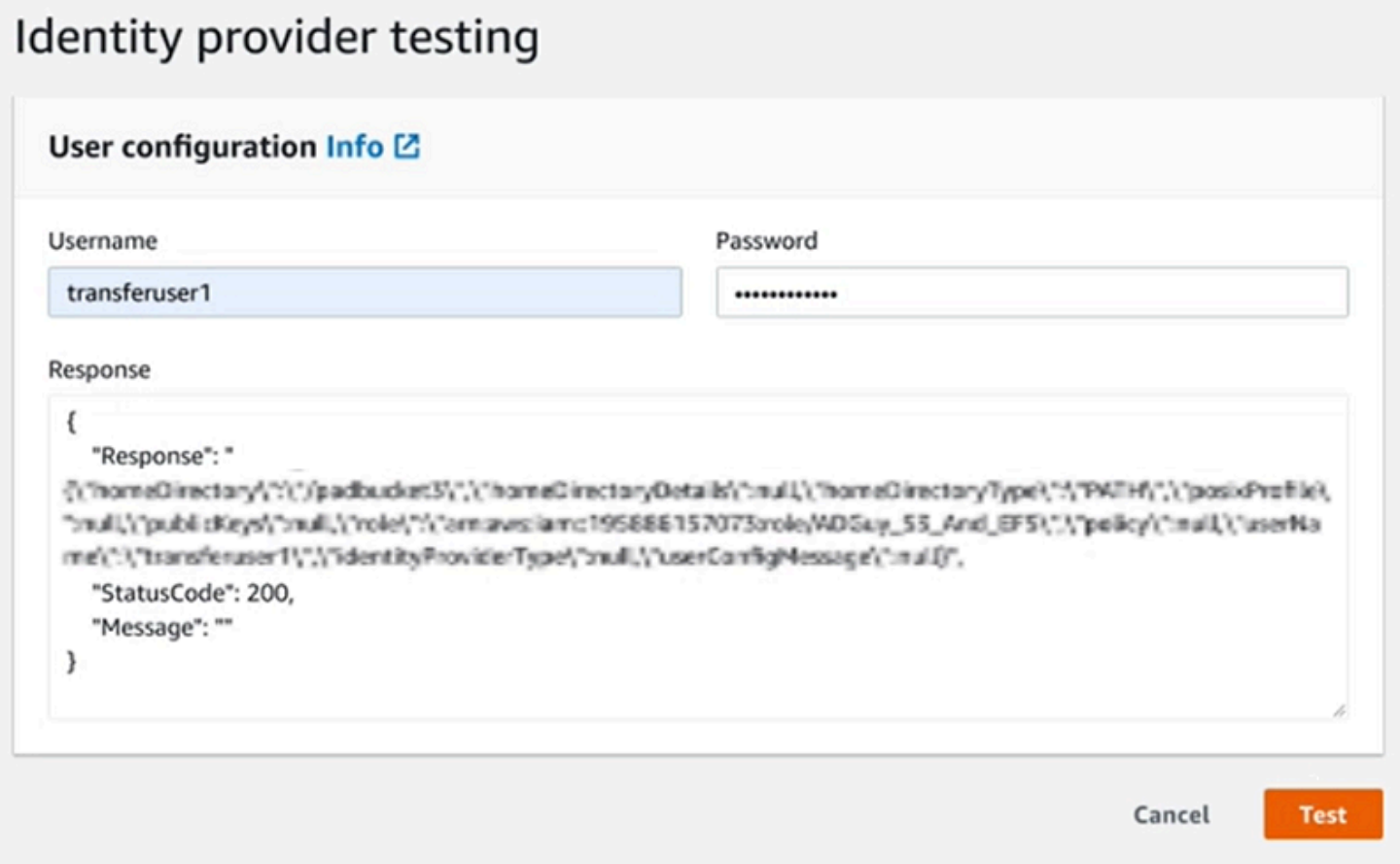
You can test whether a user has access to the AWS Managed Microsoft AD directory for your server.

> **Note**
>
> A user must be in exactly one group (an external ID) that is listed in the **Access** section of the **Endpoint configuration** page. If the user is in no groups, or is in more than a single group, that user is not granted access.

To test whether a specific user has access

1. On the server details page, choose **Actions**, and then choose **Test**.
2. For **Identity provider testing**, enter the sign-in credentials for a user that is in one of the groups that has access.
3. Choose **Test**.

You see a successful identity provider test, showing that the selected user has been granted access to the server.



If the user belongs to more than one group that has access, you receive the following response.

```
"Response":"",
"StatusCode":200,
"Message":"More than one associated access found for user's groups."
```

Deleting server access for a group

To delete server access for a group

1. On the server details page, choose **Actions**, and then choose **Delete Access**.
2. In the dialog box, confirm that you want to remove access for this group.

When you return to the server details page, you see that the access for this group is no longer listed.

Connecting to the server using SSH (Secure Shell)

After you configure your server and users, you can connect to the server using SSH and use the fully qualified username for a user that has access.

```
sftp user@active-directory-domain@vpc-endpoint
```

For example: `transferuserexample@mycompany.com@vpce-0123456abcdef-789xyz.vpc-svc-987654zyxabc.us-east-1.vpce.amazonaws.com`.

This format targets the search of the federation, limiting the search of a potentially large Active Directory.

> **Note**
>
> You can specify the simple username. However, in this case, the Active Directory code has to search all the directories in the federation. This might limit the search, and authentication might fail even if the user should have access.

After authenticating, the user is located in the home directory that you specified when you configured the user.

Connecting AWS Transfer Family to a self-managed Active Directory using forests and trusts

Directory Service has the following options available to connect to a self-managed Active Directory:

- One-way forest trust (outgoing from AWS Managed Microsoft AD and incoming for on-premises Active Directory) works only for the root domain.
- For child domains, you can use either of the following:
  - Use two-way trust between AWS Managed Microsoft AD and on-premises Active Directory
  - Use one-way external trust to each child domain.

When connecting to the server using a trusted domain, the user needs to specify the trusted domain, for example `transferuserexample@mycompany.com`.

Using AWS Directory Service for Entra ID Domain Services

For customers that need SFTP Transfer only, and do not want to manage a domain, there is Simple Active Directory. Alternatively, customers who want the benefits of Active Directory and high availability in a fully managed service can use AWS Managed Microsoft AD. Finally, for customers who want to take advantage of their existing Active Directory forest for their SFTP Transfer, there is Active Directory Connector.

Note the following:

- To take advantage of your existing Active Directory forest for your SFTP Transfer needs, you can use Active Directory Connector.
- If you want the benefits of Active Directory and high availability in a fully managed service, you can use AWS Directory Service for Microsoft Active Directory. For details, see Using AWS Directory Service for Microsoft Active Directory.

This topic describes how to use an Active Directory Connector and Entra ID (formerly Azure AD) Domain Services to authenticate SFTP Transfer users with Entra ID.

Topics

- Before you start using AWS Directory Service for Entra ID Domain Services
- Step 1: Adding Entra ID Domain Services
- Step 2: Creating a service account
- Step 3: Setting up AWS Directory using AD Connector
- Step 4: Setting up AWS Transfer Family server
- Step 5: Granting access to groups
- Step 6: Testing users

Before you start using AWS Directory Service for Entra ID Domain Services

Note

AWS Transfer Family has a default limit of 100 Active Directory groups per server. If your use case requires more than 100 groups, consider using a custom identity provider solution

as described in Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

For AWS, you need the following:

- A virtual private cloud (VPC) in an AWS region where you are using your Transfer Family servers
- At least two private subnets in your VPC
- The VPC must have internet connectivity
- A customer gateway and Virtual private gateway for site-to-site VPN connection with Microsoft Entra

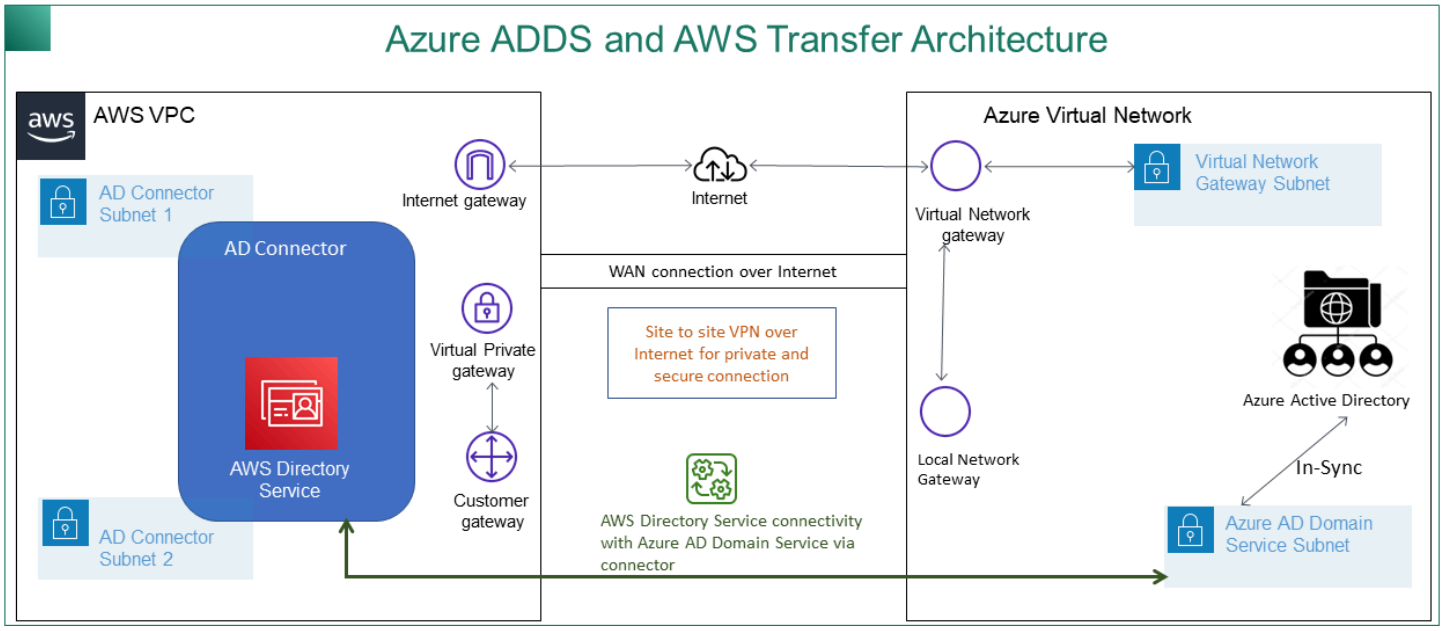
For Microsoft Entra, you need the following:

- An Entra ID and Active directory domain service
- An Entra resource group
- An Entra virtual network
- VPN connectivity between your Amazon VPC and your Entra resource group

> **Note**
>
> This can be through native IPSEC tunnels or using VPN appliances. In this topic, we use IPSEC tunnels between an Entra Virtual network gateway and local network gateway. The tunnels must be configured to allow traffic between your Entra Domain Service endpoints and the subnets that house your AWS VPC.

- A customer gateway and Virtual private gateway for site-to-site VPN connection with Microsoft Entra

The following diagram shows the configuration needed before you begin.

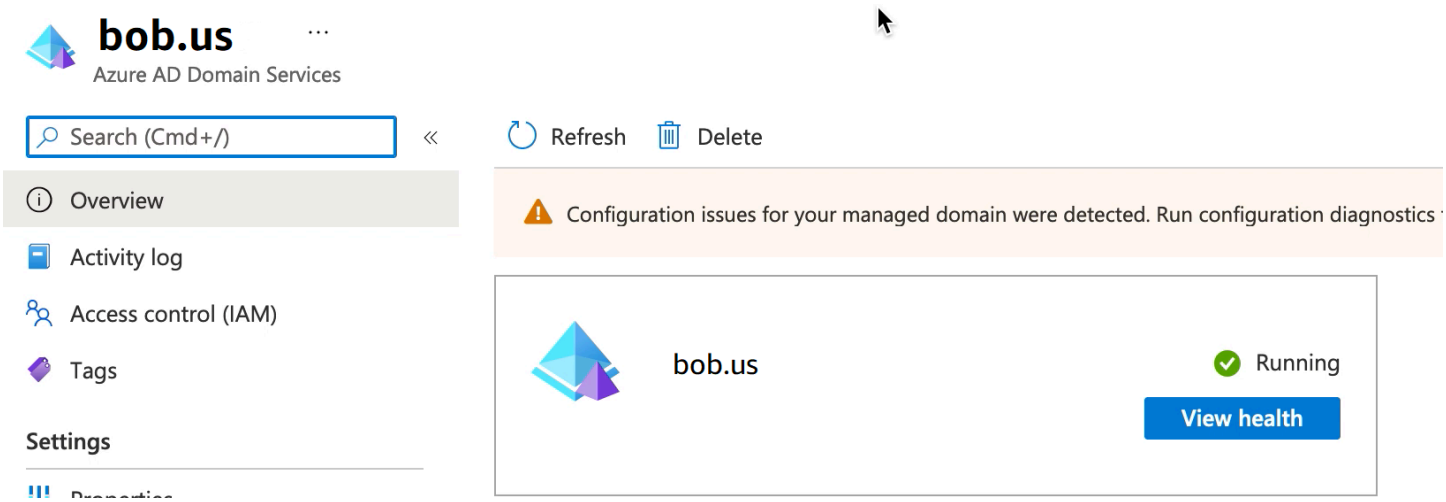


Step 1: Adding Entra ID Domain Services

Entra ID does not support Domain joining instances by default. To perform actions like Domain Join, and to use tools such as Group Policy, administrators must enable Entra ID Domain Services. If you have not already added Entra DS, or your existing implementation is not associated with the domain that you want your SFTP Transfer server to use, you must add a new instance.

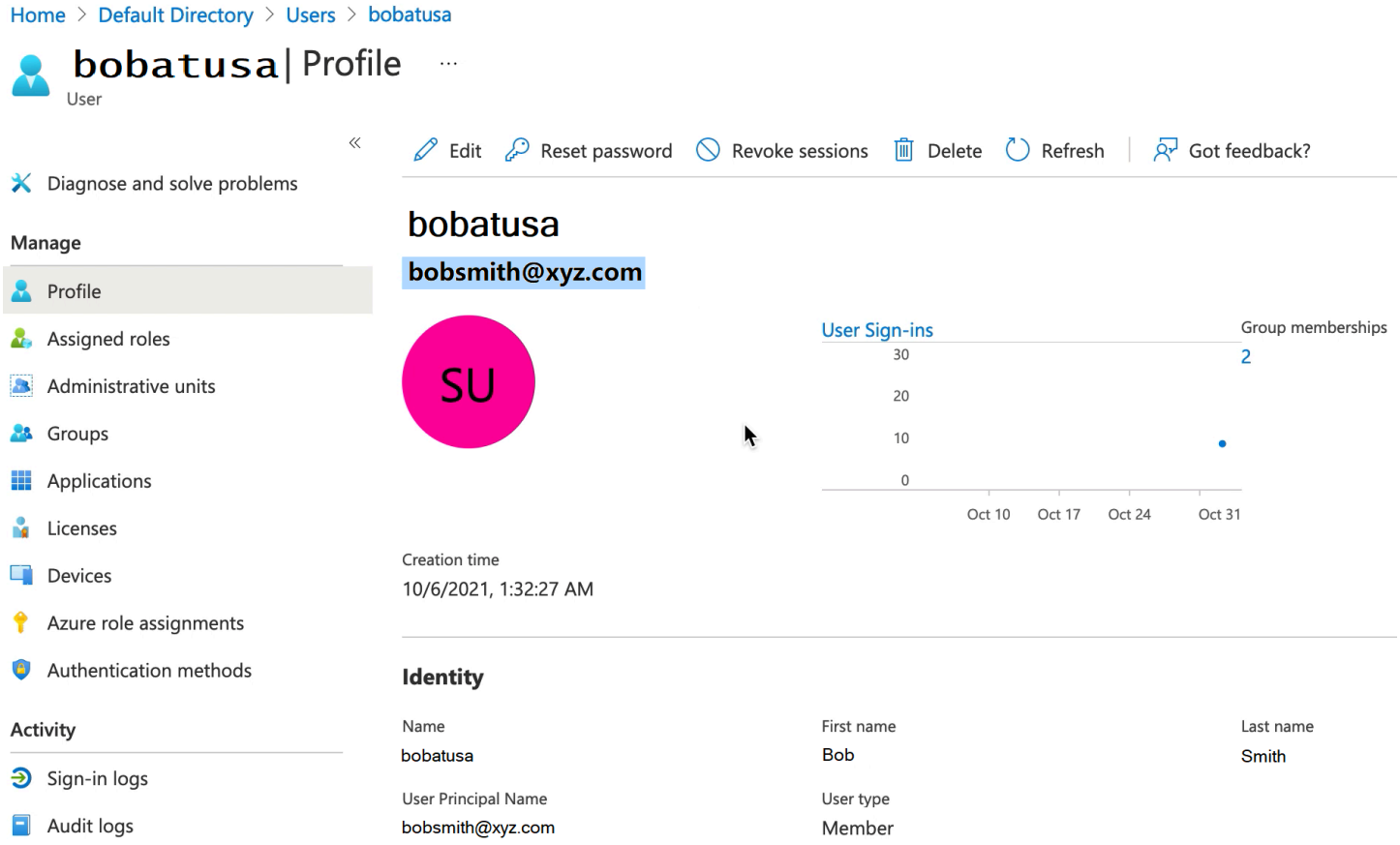
For information about enabling Entra ID Domain Services, see Tutorial: Create and configure a Microsoft Entra Domain Services managed domain.

> **Note**
>
> When you enable Entra DS, make sure it is configured for the resource group and the Entra domain to which you are connecting your SFTP Transfer server.



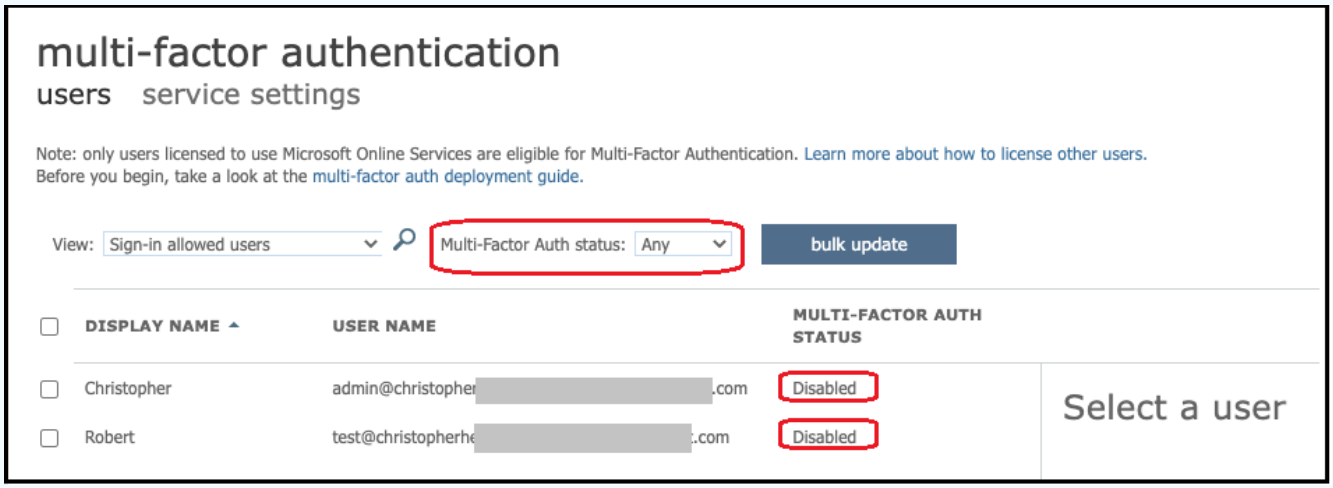
Step 2: Creating a service account

Entra must have one service account that is part of an Admin group in Entra DS. This account is used with the AWS Active Directory connector. Make sure this account is in sync with Entra DS.



Tip

Multi-factor authentication for Entra ID is not supported for Transfer Family servers that use the SFTP protocol. The Transfer Family server cannot provide the MFA token after a user authenticates to SFTP. Make sure to disable MFA before you attempt to connect.



Step 3: Setting up AWS Directory using AD Connector

After you have configured Entra DS, and created a service account with IPSEC VPN tunnels between your AWS VPC and Entra Virtual network, you can test the connectivity by pinging the Entra DS DNS IP address from any AWS EC2 instance.

After you verify the connection is active, you can continue below.

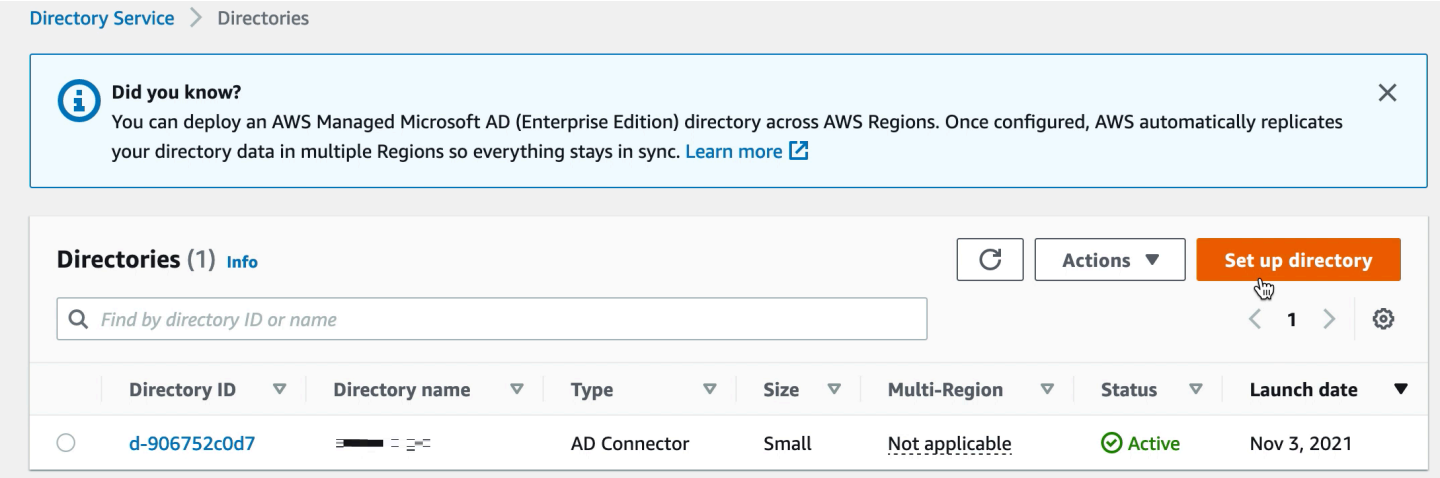
To set up your AWS Directory using AD Connector

1. Open the Directory Service console and select **Directories**.
2. Select **Set up directory**.
3. For directory type, choose **AD Connector**.
4. Select a directory size, select **Next**, then select your VPC and Subnets.
5. Select **Next**, then fill in the fields as follows:

   - **Directory DNS name**: enter the domain name you are using for your Entra DS.
   - **DNS IP addresses**: enter your Entra DS IP addresses.
   - **Server account username** and **password**: enter the details for the service account you created in *Step 2: Create a service account*.

6. Complete the screens to create the directory service.

Now the directory status should be **Active**, and it is ready to be used with an SFTP Transfer server.



Step 4: Setting up AWS Transfer Family server

Create a Transfer Family server with the SFTP protocol, and the identity provider type of **AWS Directory Service**. From **Directory** drop down list, select the directory you added in *Step 3: Setup AWS Directory using AD Connector*.

> **Note**
>
> You can't delete a Microsoft AD directory in AWS Directory Service if you used it in a Transfer Family server. You must delete the server first, and then you can delete the directory.

Step 5: Granting access to groups

After you create the server, you must choose which groups in the directory should have access to upload and download files over the enabled protocols using AWS Transfer Family. You do this by creating an *access*.

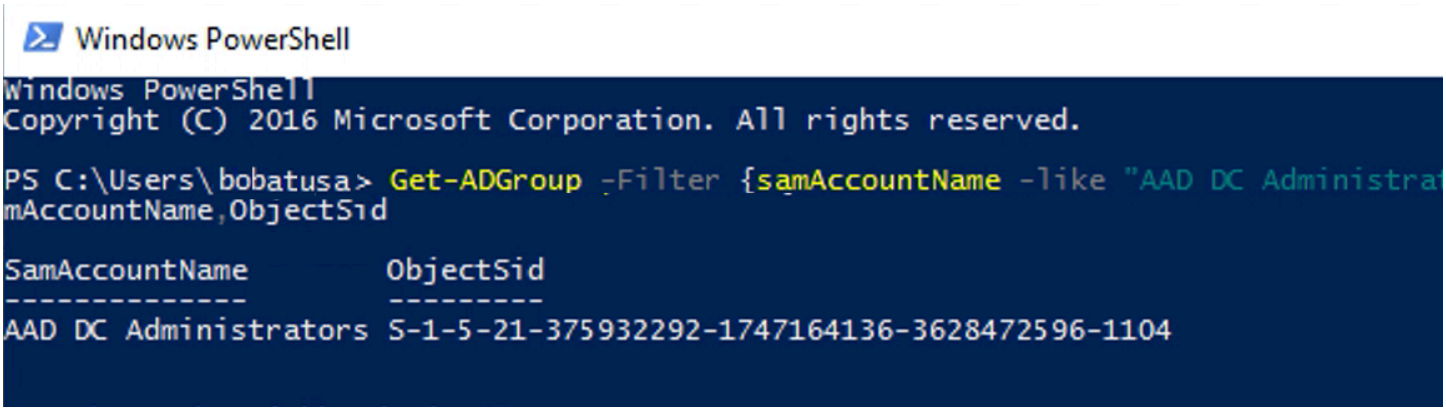
> **Note**
>
> Users must belong *directly* to the group to which you are granting access. For example, assume that Bob is a user and belongs to groupA, and groupA itself is included in groupB.

- If you grant access to groupA, Bob is granted access.
- If you grant access to groupB (and not to groupA), Bob does not have access.

In order to grant access you need to retrieve the SID for the group.

Use the following Windows PowerShell command to retrieve the SID for a group, replacing *YourGroupName* with the name of the group.

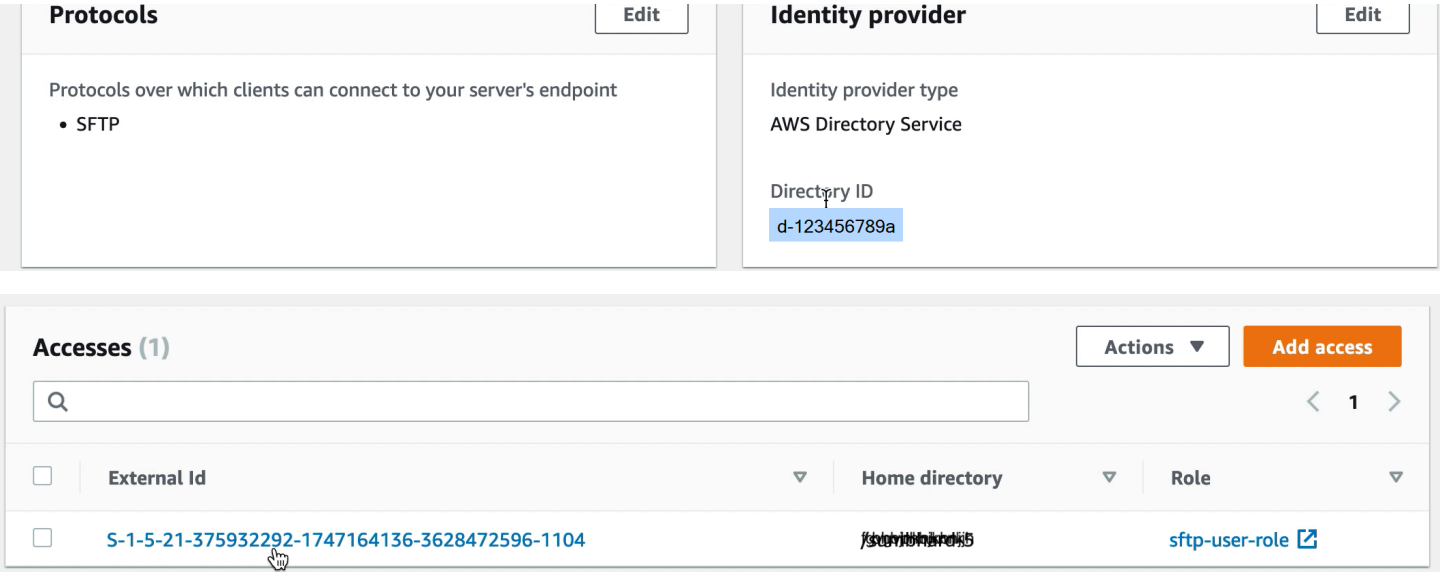
```
Get-ADGroup -Filter {samAccountName -like "YourGroupName*"} -Properties * | Select
 SamAccountName,ObjectSid
```



Grant access to groups

1. Open https://console.aws.amazon.com/transfer/.
2. Navigate to your server details page and in the **Accesses** section, choose **Add access**.
3. Enter the SID you received from the output of the previous procedure.
4. For **Access**, choose an AWS Identity and Access Management role for the group.
5. In the **Policy** section, choose a policy. The default value is **None**.
6. For **Home directory**, choose an Amazon S3 bucket that corresponds to the group's home directory.
7. Choose **Add** to create the association.

The details from your Transfer server should look similar to the following:



Step 6: Testing users

You can test (Testing users) whether a user has access to the AWS Managed Microsoft AD directory for your server. A user must be in exactly one group (an external ID) that is listed in the **Access** section of the **Endpoint configuration** page. If the user is in no groups, or is in more than a single group, that user is not granted access.

Using logical directories to simplify your Transfer Family directory structures

Logical directories simplify your AWS Transfer Family server directory structure. With logical directories, you can create a virtual directory structure with user-friendly names that users navigate when connecting to your Amazon S3 bucket or Amazon EFS file system. This prevents users from seeing the actual directory paths, bucket names, and file system names.

> **Note**
>
> You should use session policies so that your end users can only perform operations that you allow them to perform.
>
> You should use logical directories to create a user-friendly, virtual directory for your end users and abstract away bucket names. Logical directory mappings only allow users to access their designated logical paths and subdirectories, and forbid relative paths that traverse the logical roots.

Transfer Family validates every path that might include relative elements and actively blocks these paths from resolving before we pass these paths to Amazon S3; this prevents your users from moving beyond their logical mappings.
Even though Transfer Family prevents your end users from accessing directories outside of their logical directory, we recommend you also use unique roles or session policies to enforce least privilege at the storage level.

Understanding chroot and directory structure

A **chroot** operation lets you set a user's root directory to any location in your storage hierarchy. This restricts users to their configured home or root directory, preventing access to higher-level directories.

Consider a case where an Amazon S3 user is limited to `amzn-s3-demo-bucket/home/${transfer:UserName}`. Without **chroot**, some clients might allow users to move up to /amzn-s3-demo-bucket/home, requiring a logout and login to return to their proper directory. Performing a **chroot** operation prevents this issue.

You can create custom directory structures across multiple buckets and prefixes. This is useful if your workflow requires a specific directory layout that bucket prefixes alone can't provide. You can also link to multiple non-contiguous locations within Amazon S3, similar to creating a symbolic link in a Linux file system where your directory path references a different location in the file system.

Rules for using logical directories

This section describes some rules and other considerations for using logical directories.

Mapping limits

- Only one mapping is allowed when `Entry` is `"/"` (no overlapping paths are allowed).
- Logical directories support mappings of up to 2.1 MB for custom IDP and AD users, and 2,000 entries for service-managed users. You can calculate your mappings size as follows:
  1. Write out a typical mapping in the format `{"Entry":"/`*`entry-path`*`","Target":"/`*`target-path`*`"}`, where *`entry-path`* and *`target-path`* are the actual values that you will use.
  2. Count the characters in that string, then add one (1).
  3. Multiply that number by the approximate number of mappings that you have for your server.

If the number that you estimated in step 3 is less than 2.1 MB, then your mappings are within the acceptable limit.

Target path requirements

- Use `${transfer:UserName}` variable if the bucket or file system path has been parameterized based on the username.
- Targets can be configured to point to different Amazon S3 buckets or file systems, as long as the associated IAM role has the necessary permissions to access those storage locations.
- All targets must begin with a forward slash (/) but can't end with one. For example, `/amzn-s3-demo-bucket/images` is correct, while `amzn-s3-demo-bucket/images` and `/amzn-s3-demo-bucket/images/` are not.

Storage considerations

- Amazon S3 is an object store where folders exist only as a virtual concept. When using Amazon S3 storage, Transfer Family reports prefixes as directories in STAT operations, even if there is no zero-byte object with a trailing slash. A proper zero-byte object with a trailing slash is also reported as a directory in STAT operations. This behavior is described in Organizing objects in the Amazon S3 console using folders in the *Amazon Simple Storage Service User Guide*.
- For applications that need to distinguish between files and folders, use Amazon Elastic File System (Amazon EFS) as your Transfer Family storage option.
- If you're specifying logical directory values for your user, the parameter that you use depends on the type of user:
  - For service-managed users, provide logical directory values in `HomeDirectoryMappings`.
  - For custom identity provider users, provide logical directory values in `HomeDirectoryDetails`.

User directory values

- The parameter for specifying logical directory values depends on your user type:
  - For service-managed users, provide logical directory values in `HomeDirectoryMappings`.
  - For custom identity provider users, provide logical directory values in `HomeDirectoryDetails`.

- When using LOGICAL HomeDirectoryType, you can specify a HomeDirectory value for Service Managed users, Active Directory access, and Custom Identity Provider implementations where the HomeDirectoryDetails are provided in the response. If not specified, HomeDirectory defaults to /.

For details on how to implement logical directories, see Implementing logical directories.

Implementing logical directories

Important

Root directory requirements

- If you are not using Amazon S3 performance optimization settings your root directory must exist on startup.
- For Amazon S3, this means creating a zero-byte object that ends with a forward slash (/).
- To avoid this requirement, consider enabling Amazon S3 performance optimization when you create or update your server.
- When specifying a HomeDirectory with LOGICAL HomeDirectoryType, the value must map to one of your logical directory mappings. The service validates this during both user creation and updates to prevent configurations that would not work.

Logical home directory configuration

When using LOGICAL as your HomeDirectoryType, note the following:

- The HomeDirectory value must correspond to one of your existing logical directory mappings.
- The system automatically validates this during user creation and updates.
- This validation prevents configurations that would cause access issues.

Enable logical directories

To use logical directories for a user, set the `HomeDirectoryType` parameter to `LOGICAL`. Do this when you create a new user or update an existing user.

```
"HomeDirectoryType": "LOGICAL"
```

Enable chroot for users

For **chroot**, create a directory structure that consists of a single `Entry` and `Target` pairing for each user. The Entry **/** represents the root folder, while **Target** specifies the actual location in your bucket or file system.

Example for Amazon S3

```
[{"Entry": "/", "Target": "/amzn-s3-demo-bucket/jane"}]
```

Example for Amazon EFS

```
[{"Entry": "/", "Target": "/fs-faa1a123/jane"}]
```

You can use an absolute path as in the previous example, or you can use a dynamic substitution for the username with `${transfer:UserName}`, as in the following example.

```
[{"Entry": "/", "Target":
"/amzn-s3-demo-bucket/${transfer:UserName}"}]
```

In the preceding example, the user is locked to their root directory and cannot traverse up higher in the hierarchy.

Virtual directory structure

For a virtual directory structure, you can create multiple `Entry Target` pairings, with targets anywhere in your S3 buckets or EFS file systems, including across multiple buckets or file systems, as long as the user's IAM role mapping has permissions to access them.

In the following virtual structure example, when the user logs into AWS SFTP, they are in the root directory with sub-directories of `/pics`, `/doc`, `/reporting`, and `/anotherpath/subpath/financials`.

> **Note**
>
> Unless you choose to optimize performance for your Amazon S3 directories (when you create or update a server), either the user or an administrator needs to create the

directories if they don't already exist. Avoiding this issue is a reason to consider optimizing Amazon S3 performance.
For Amazon EFS, you still need the administrator to create the logical mappings or the `/` directory.

```
[
{"Entry": "/pics", "Target": "/amzn-s3-demo-bucket1/pics"},
{"Entry": "/doc", "Target": "/amzn-s3-demo-bucket1/anotherpath/docs"},
{"Entry": "/reporting", "Target": "/amzn-s3-demo-bucket2/Q1"},
{"Entry": "/anotherpath/subpath/financials", "Target": "/amzn-s3-demo-bucket2/
financials"}]
```

Note

You can only upload files to the specific folders that you map. This means that in the previous example, you cannot upload to `/anotherpath` or `anotherpath/subpath` directories; only `anotherpath/subpath/financials`. You also cannot map to those paths directly, as overlapping paths are not allowed.
For example, assume that you create the following mappings:

```
{
   "Entry": "/pics",
   "Target": "/amzn-s3-demo-bucket/pics"
},
{
   "Entry": "/doc",
   "Target": "/amzn-s3-demo-bucket/mydocs"
},
{
   "Entry": "/temp",
   "Target": "/amzn-s3-demo-bucket2/temporary"
}
```

You can only upload files to those buckets. When you first connect through `sftp`, you are dropped into the root directory, `/`. If you attempt to upload a file to that directory, the upload fails. The following commands show an example sequence:

```
sftp> pwd
```

```
Remote working directory: /
sftp> put file
Uploading file to /file
remote open("/file"): No such file or directory
```

To upload to any `directory/sub-directory`, you must explicitly map the path to the `sub-directory`.

For more information about configuring logical directories and **chroot** for your users, including an AWS CloudFormation template that you can download and use, see Simplify your AWS SFTP Structure with chroot and logical directories in the AWS Storage Blog.

Configure logical directories examples

In this example, we create a user and assign two logical directories. The following command creates a new user (for an existing Transfer Family server) with logical directories `pics` and doc.

```
aws transfer create-user \
    --user-name marymajor \
    --server-id s-11112222333344445 \
    --role arn:aws:iam::1234abcd5678:role/marymajor-role \
    --home-directory-type LOGICAL \
    --home-directory-mappings "[{\"Entry\":\"/pics\", \"Target\":\"/amzn-s3-demo-
bucket1/pics\"}, {\"Entry\":\"/doc\", \"Target\":\"/amzn-s3-demo-bucket2/test/mydocs
\"}]" \
    --ssh-public-key-body file://~/.ssh/id_rsa.pub
```

If **`marymajor`** is an existing user and her home directory type is PATH, you can change it to LOGICAL with a similar command as the previous one.

```
aws transfer update-user \
    --user-name marymajor \
    --server-id s-11112222333344445 \
    --role arn:aws:iam::1234abcd5678:role/marymajor-role \
    --home-directory-type LOGICAL \
    --home-directory-mappings "[{\"Entry\":\"/pics\", \"Target\":\"/amzn-s3-demo-
bucket1/pics\"}, {\"Entry\":\"/doc\", \"Target\":\"/amzn-s3-demo-bucket2/test/mydocs
\"}]"
```

Note the following:

- If the directories `/amzn-s3-demo-bucket1/pics` and `/amzn-s3-demo-bucket2/test/mydocs` don't already exist, the user (or an administrator) needs to create them.

  > **Note**
  >
  > These directories are created automatically by the Transfer Family server if you have configured optimized directories.

- When **`marymajor`** connects to the server, and runs the `ls -l` command, Mary sees the following:

```
drwxr--r--   1        -        -        0 Mar 17 15:42 doc
drwxr--r--   1        -        -        0 Mar 17 16:04 pics
```

- **`marymajor`** cannot create any files or directories at this level. However, within `pics` and `doc`, she can add sub-directories.
- Files that Mary adds to `pics` and `doc` are added to Amazon S3 paths `/amzn-s3-demo-bucket1/pics` and `/amzn-s3-demo-bucket2/test/mydocs` respectively.
- In this example, we specify two different buckets to illustrate that possibility. However, you can use the same bucket for several or all of the logical directories that you specify for the user.

This example provides an alternate configuration for a logical home path.

```
aws transfer create-user \
    --user-name marymajor \
    --server-id s-11112222333344445 \
    --role arn:aws:iam::1234abcd5678:role/marymajor-role \
    --home-directory-type LOGICAL \
    --home-directory /home/marymajor \
    --home-directory-mappings "[{\"Entry\":\"/home/marymajor/pics\", \"Target\":\"/
amzn-s3-demo-bucket1/pics\"}, {\"Entry\":\"/home/marymajor/doc\", \"Target\":\"/amzn-
s3-demo-bucket2/test/mydocs\"}]" \
    --ssh-public-key-body file://~/.ssh/id_rsa.pub
```

Note the following:

- The mappings provide for a common path, `/home/marymajor`, which is the first part of the two logical paths. Files then can be added to the `pics` and `doc` folders.
- As in the previous example, the home directory, `/home/marymajor`, is read-only.

Configure logical directories for Amazon EFS

If your Transfer Family server uses Amazon EFS, the home directory for the user must be created with read and write access before the user can work in their logical home directory. The user cannot create this directory themselves, as they would lack permissions for `mkdir` on their logical home directory.

If the user's home directory does not exist, and they run an `ls` command, the system responds as follows:

```
sftp> ls
remote readdir ("/"): No such file or directory
```

A user with administrative access to the parent directory needs to create the user's logical home directory.

Custom AWS Lambda response

You can use logical directories with a Lambda function that connects to your custom identity provider. To do so, in your Lambda function, you specify the `HomeDirectoryType` as **`LOGICAL`**, and add `Entry` and `Target` values for the `HomeDirectoryDetails` parameter. For example:

```
HomeDirectoryType: "LOGICAL"
HomeDirectoryDetails: "[{\"Entry\": \"/\", \"Target\": \"/amzn-s3-demo-bucket/
theRealFolder"}]"
```

The following code is an example of a successful response from a custom Lambda authentication call.

```
aws transfer test-identity-provider \
    --server-id s-1234567890abcdef0 \
    --user-name myuser
{
    "Url": "https://a1b2c3d4e5.execute-api.us-east-2.amazonaws.com/prod/servers/
s-1234567890abcdef0/users/myuser/config",
    "Message": "",
    "Response": "{\"Role\": \"arn:aws:iam::123456789012:role/bob-usa-role\",
                 \"HomeDirectoryType\": \"LOGICAL\",
                 \"HomeDirectoryDetails\": \"[{\\\"Entry\\\":\\\"/myhome\\\",\\
\"Target\\\":\\\"/amzn-s3-demo-bucket/theRealFolder\\\"}]\",
                 \"PublicKeys\": \"[ssh-rsa myrsapubkey]\"}",
```

```
    "StatusCode": 200
}
```

> **Note**
>
> The `"Url":` line is returned only if you are using an API Gateway method as your custom identity provider.

Access your FSx for NetApp ONTAP file systems with Transfer Family

Contents

- Overview
- Prerequisites
  - FSx for NetApp ONTAP requirements
  - Required IAM permissions
- How FSx storage works with Transfer Family
  - File system user identity
- Creating an S3 access point for FSx
  - Access point naming
  - Creating an access point for FSx for NetApp ONTAP
  - Configuring file system permissions
- Using S3 access point aliases with FSx
  - About access point aliases
  - Finding your access point alias
- Configuring Transfer Family for FSx storage
  - Creating an IAM role
- Managing users for FSx storage
  - Creating a user
  - Configuring multiple directory mappings
- Configuring file transfer clients

- WinSCP configuration
- Other SFTP clients
- Troubleshooting FSx storage
  - File operation issues

Overview

Transfer Family supports Amazon FSx for NetApp ONTAP through S3 access points. Amazon FSx for NetApp ONTAP is a fully managed service that provides highly reliable, scalable, high-performing, and feature-rich file storage built on NetApp's popular ONTAP file system. When you configure Transfer Family with an FSx file system, your users connect to Transfer Family endpoints using standard file transfer clients. Transfer Family routes file operations through an S3 access point attached to your FSx volume, while your data remains on the FSx file system. To learn more about FSx for NetApp ONTAP, see What is Amazon FSx for NetApp ONTAP?

This integration enables you to:

- Transfer files using SFTP, FTPS, or FTP protocols to enterprise-grade file storage
- Access the same data through multiple protocols (SFTP, NFS, SMB)
- Use FSx features such as snapshots, backups, and data tiering

> **Important**
>
> Some file operations are not supported when using FSx file systems with Transfer Family, including rename and append operations. For a complete list of limitations, see Access point compatibility.

Prerequisites

Before you configure Transfer Family with Amazon FSx, you must meet the following requirements.

FSx for NetApp ONTAP requirements

To use FSx for NetApp ONTAP with Transfer Family, you need:

- An FSx for NetApp ONTAP file system running ONTAP version 9.17.1 or later

- The file system and S3 access point in the same AWS Region
- The same AWS account owning both the file system and access point

To learn more, see Getting started with Amazon FSx for NetApp ONTAP.

Required IAM permissions

You can configure each S3 access point with distinct permissions and network controls that S3 applies for any request that is made using that access point. S3 access points support IAM resource policies that you can use to control the use of the access point by resource, user, or other conditions. For an application or user to access files through an access point, both the access point and the underlying volume must permit the request. For more information, see IAM access point policies.

Amazon S3 access points for FSx use a dual-layer authorization model that combines IAM permissions with file system-level permissions. This approach ensures that data access requests are properly authorized at both the AWS service level and the underlying file system level.

For an application or user to successfully access data through an access point, both the S3 access point policy and the underlying FSx volume must permit the request.

To create and configure this integration, you need the following permissions:

- `fsx:CreateAndAttachS3AccessPoint`
- `s3:CreateAccessPoint`
- `s3:GetAccessPoint`
- `s3:PutAccessPointPolicy` (if creating an optional access point policy)

How FSx storage works with Transfer Family

When you configure Transfer Family with an FSx file system, the following components work together to enable file transfers:

1. A user connects to the Transfer Family server using an SFTP, FTPS, or FTP client.
2. Transfer Family authenticates the user using service-managed identities, a custom identity provider, or AWS Directory Service for Microsoft Active Directory. Once authenticated, Transfer Family assumes the IAM role associated with the user.

3. For each file operation, Transfer Family acts as a standard S3 API client, making requests to the S3 Access Point using the assumed IAM role of the user and verifies permissions against the S3 access point policy.
4. The FSx file system verifies that the file system user associated with the access point has permission to perform the requested operation. The file operation is then performed on the FSx volume.

For a file operation to succeed, both authorization layers must permit the request.

> **Note**
>
> Attaching an S3 access point to an FSx volume does not change how the volume behaves when accessed directly through NFS or SMB. Existing file protocol access continues to work unchanged.

File system user identity

Each access point uses a file system user identity that you specify when creating the access point. This identity authorizes all file access requests made through that access point. The file system user is a user account on the underlying Amazon FSx file system. If the file system user has read-only access, then only read requests made using the access point are authorized, and write requests are blocked. If the file system user has read-write access, then both read and write requests to the attached volume made using the access point are authorized.

Creating an S3 access point for FSx

Before you configure Transfer Family, you must create an S3 access point attached to your FSx volume. S3 access points are named network endpoints that are attached to a data source such as a bucket or Amazon FSx for ONTAP volume. You can create and attach an access point to an FSx for NetApp ONTAP using the Amazon FSx console, AWS CLI, or API. Once attached, you can use the S3 object APIs to access your file data. Your data continues to reside on the Amazon FSx file system and continues to be directly accessible for your existing workloads. You continue to manage your storage using all the FSx for NetApp ONTAP storage management capabilities, including backups, snapshots, user and group quotas, and compression.

For more information, see Creating access points.

Access point naming

When you name your access point, follow these guidelines:

- Access point names must be unique within your AWS account and Region.
- Names cannot end with `-ext-s3alias` (reserved for aliases).
- Avoid including sensitive information in names because they are published in DNS.

For a full list of naming rules, see Access points naming rules, restrictions, and limitations.

Creating an access point for FSx for NetApp ONTAP

Use the following procedure to create an S3 access point for an FSx for NetApp ONTAP volume.

To create an access point (console)

1. Open the Amazon FSx console at https://console.aws.amazon.com/fsx/.
2. In the navigation pane, choose **File systems**.
3. Choose your FSx for NetApp ONTAP file system.
4. Choose the **Volumes** tab.
5. Select the volume that you want to attach.
6. For **Actions**, choose **Create S3 access point**.
7. For **Access point name**, enter a descriptive name (for example, `transfer-family-ap`).
8. For **File system user identity type**, choose one of the following:
   - **UNIX identity** - For volumes with UNIX security style
   - **Windows identity** - For volumes with NTFS security style
9. (Optional) For **Access point policy**, enter an IAM policy that defines which IAM principals can perform which operations on objects accessed through this access point. For more information, see Managing access point access.
10. Choose **Create**.
11. After creation, note the access point alias for use in Transfer Family configuration.

Note

When AWS Transfer Family accesses S3 resources on behalf of your connected SFTP/FTPS users, requests originate from AWS Transfer Family infrastructure, not from your VPC. Because of this, S3 Access Points configured with a VPC network origin will deny these requests. However, even if you use an Access Point configured with an Internet network origin, all traffic between Transfer Family and the Access Point remains private and travels over the AWS backbone network - it does not traverse the public internet.

Configuring file system permissions

The file system user that you specify determines what operations Transfer Family users can perform. You must configure appropriate permissions on your FSx volume.

UNIX example:

```
# Create a directory for Transfer Family users
mkdir -p /vol1/transfer-users

# Set ownership to match the access point user
chown 1001:1001 /vol1/transfer-users

# Set permissions
chmod 755 /vol1/transfer-users
```

Windows example:

```
# Create a directory for Transfer Family users
New-Item -Path "D:\vol1\transfer-users" -ItemType Directory

# Set permissions for the file system user associated with the access point
# Replace DOMAIN\TransferUser with your Windows user identity
icacls "D:\vol1\transfer-users" /grant "DOMAIN\TransferUser:(OI)(CI)M" /T

# Verify permissions
icacls "D:\vol1\transfer-users"
```

Using S3 access point aliases with FSx

When you use FSx file systems with Transfer Family, you must use S3 access point aliases. Transfer Family does not support using access point ARNs or other reference methods for FSx storage.

Important

AWS Transfer Family only supports S3 access point aliases when using FSx file systems. You cannot use access point ARNs or virtual-hosted-style URIs.

Important

The access point must be in the same Region as the volume.

About access point aliases

When you create an S3 access point attached to an FSx volume, Amazon S3 automatically generates an access point alias. This alias is a unique identifier that you can use anywhere you use an S3 bucket name.

For access points attached to FSx volumes, the alias uses the following format:

```
access-point-name-metadata-ext-s3alias
```

Example alias:

```
my-fsx-ap-aqfqprnstn7aefdfbarligizwgyfouse1a-ext-s3alias
```

Note

The `-ext-s3alias` suffix is reserved for FSx access point aliases. You cannot use this suffix in access point names.

Finding your access point alias

You can find the access point alias after creating the access point.

To find the access point alias (console)

1. Open the Amazon FSx console at https://console.aws.amazon.com/fsx/.
2. In the navigation pane, choose **File systems**.
3. Choose your file system.
4. Choose the **Volumes** tab and select the volume you created the access point for.
5. Go to **S3 access point details** column.
6. The alias is displayed in the **Alias** column.

To find the access point alias (CLI)

Use the `describe-s3-access-point-attachments` command.

```
aws fsx describe-s3-access-point-attachments \
    --filters Name=file-system-id,Values=fs-0123456789abcdef0
```

The response includes the alias:

```
{
    "S3AccessPointAttachments": [
        {
            "S3AccessPoint": {
                "ResourceARN": "arn:aws:s3:us-east-1:111122223333:accesspoint/my-fsx-
ap",
                "Alias": "my-fsx-ap-aqfqprnstn7aefdfbarligizwgyfouse1a-ext-s3alias"
            }
        }
    ]
}
```

When you configure Transfer Family users, use the access point alias in home directory mappings.

Home directory format:

```
/access-point-alias/path/to/directory
```

Example:

```
/my-fsx-ap-aqfqprnstn7aefdfbarligizwgyfouse1a-ext-s3alias/users/jsmith
```

Configuring Transfer Family for FSx storage

After you create the S3 access point, configure a Transfer Family server to use it.

Creating an IAM role

You must create an IAM role that grants Transfer Family access to the S3 access point.

> **Important**
>
> IAM policies require the Access Point ARN format, not the alias. Use the format `arn:aws:s3:region:account-id:accesspoint/access-point-name` in your IAM policy Resource statements. The access point alias (ending in `-ext-s3alias`) is only used for home directory mappings.

To create the IAM role

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, then choose **Create role**.
3. For **Trusted entity type**, choose **AWS service**.
4. For **Use case**, choose **Transfer**.
5. Choose **Next**.
6. Choose **Create policy** and enter your policy (see sample policy below).
7. Attach the policy to the role and choose **Create role**.

Example IAM policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFileOperations",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObjectTagging",
                "s3:PutObjectTagging"
            ],
            "Resource": "arn:aws:s3:us-east-2:111122223333:accesspoint/my-fsx-ap/
object/*"
        },
        {
            "Sid": "AllowDirectoryOperations",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:us-east-2:111122223333:accesspoint/my-fsx-ap"
        }
    ]
}
```

Managing users for FSx storage

Create Transfer Family users with home directory mappings that use the S3 access point alias.

Creating a user

When you create a user for FSx storage, use the access point alias in home directory mappings.

To create a Service Managed user (console)

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Choose your server.
4. In the Users section, choose **Add user**.
5. For **Username**, enter a username.
6. For **Role**, choose the IAM role that you created.
7. For **Home directory**, choose **Restricted**.

8. For **Home directory mappings**, add a mapping using the access point alias:

```
[{"Entry": "/", "Target": "/my-fsx-ap-aqfqprnstn7aefdfbarligizwgyfouse1a-ext-
s3alias/users/jsmith"}]
```

To create a user (CLI)

Use the `create-user` command. Replace the access point alias with your alias.

```
aws transfer create-user \
    --server-id s-0123456789abcdef0 \
    --user-name jsmith \
    --role arn:aws:iam::111122223333:role/TransferFamilyFSxRole \
    --home-directory-type LOGICAL \
    --home-directory-mappings '[
        {
            "Entry": "/",
            "Target": "/my-fsx-ap-aqfqprnstn7aefdfbarligizwgyfouse1a-ext-s3alias/users/
jsmith"
        }
    ]'
```

Configuring multiple directory mappings

You can map multiple virtual directories to different paths on the FSx volume.

Example: Separate upload and download directories

```
aws transfer create-user \
    --server-id s-0123456789abcdef0 \
    --user-name jsmith \
    --role arn:aws:iam::111122223333:role/TransferFamilyFSxRole \
    --home-directory-type LOGICAL \
    --home-directory-mappings '[
        {
            "Entry": "/inbox",
            "Target": "/my-fsx-ap-aqfqprnstn7aefdfbarligizwgyfouse1a-ext-s3alias/users/
jsmith/inbox"
        },
        {
            "Entry": "/outbox",
```

```
            "Target": "/my-fsx-ap-aqfqprnstn7aefdfbarligizwgyfouse1a-ext-s3alias/users/
jsmith/outbox"
        }
    ]'
```

Configuring file transfer clients

When using FSx file systems with Transfer Family, you must configure your file transfer clients to disable features that are not supported.

WinSCP configuration

WinSCP uses a temporary rename feature by default that is not supported with S3 access points for FSx.

> **Warning**
>
> If you do not disable the temporary rename feature in WinSCP, file uploads will fail.

To disable temporary rename in WinSCP

1. Open WinSCP.
2. On the Login dialog, choose **Edit** to modify your session settings.
3. Choose **Advanced**.
4. In the left navigation, under **Transfer**, choose **Endurance**.
5. For **Enable transfer resume/transfer to temporary filename**, choose **Disable**.
6. Choose **OK** to save the settings.

Alternatively, you can disable this setting for an existing session:

1. Connect to your Transfer Family server.
2. Choose **Options**, then **Preferences**.
3. Choose **Transfer**, then **Endurance**.
4. For **Enable transfer resume/transfer to temporary filename**, choose **Disable**.
5. Choose **OK**.

Other SFTP clients

For other SFTP clients, disable the following features if available:

- Temporary file uploads (upload to temp file, then rename)
- Resume transfers using temporary files
- Atomic uploads using rename operations
- Append mode for uploads

Consult your client documentation for specific configuration steps.

Troubleshooting FSx storage

This section describes how to identify and resolve common issues when using Transfer Family with FSx file systems.

File operation issues

Permission denied

If you receive permission denied errors:

1. Verify the IAM role has the correct permissions for the access point alias. You can do this by testing directly with S3 APIs.
2. Check that the access point policy allows the IAM role.
3. Verify the file system user has permissions on the target path.
4. Confirm the home directory mapping uses the correct access point alias.

Upload fails with WinSCP

If file uploads fail with WinSCP, disable temporary rename:

1. In WinSCP, choose **Options**, then **Preferences**.
2. Choose **Transfer**, then **Endurance**.
3. For **Enable transfer resume/transfer to temporary filename**, choose **Disable**.

For more information, see the section called “Configuring file transfer clients”.

File upload fails

If file uploads fail:

1. Verify the file size is within the FSx upload limit. For more information, see Access point compatibility.
2. Check that the FSx volume has sufficient available storage.
3. Monitor CloudWatch metrics for throttling.

Transfer Family web apps

You can create web apps to enable a simple interface for transferring data to and from Amazon Simple Storage Service (S3) over a web browser. This does not require you to create or provision AWS Transfer Family servers.

Before the introduction of Transfer Family web apps, end users needed to use a client, custom-built, or a third-party solution to access their data in Amazon S3. This was due to stringent security requirements for customers and partners, and because clients apps are challenging for non-technical users to operate.

With the launch of web apps, you can now extend a branded, secure, and highly available portal for your end users to browse, upload, and download data in Amazon S3. Web apps are natively integrated with AWS IAM Identity Center and Amazon S3 Access Grants. This means that only your authenticated users can view the data that they're authorized to access. Web apps are built using Storage Browser for Amazon S3 and offer the same end user functionalities in a fully managed offering without having to write code or host your own application.

For more information about the other AWS services that you use with Transfer Family web apps, see the following documentation:

- Managing access with S3 Access Grants in the Amazon Simple Storage Service User Guide
- AWS IAM Identity Center User Guide
- Amazon S3 Access Grants workshop
- Announcing AWS Transfer Family web apps for fully managed Amazon S3 file transfers

The following resources are available to help you to get started with Transfer Family web apps.

- The user guide offers a detailed, step-by-step walkthrough of setting up a Transfer Family web app here:Tutorial: Setting up a basic Transfer Family web app.
- The **AWS Getting Started Resource Center** offers a tutorial here: Getting started with AWS Transfer Family web app.
- View Getting started with AWS Transfer Family web app.

AWS Regions for Transfer Family web apps

AWS Transfer Family web apps are available in all the Transfer Family supported regions, as listed in the AWS Region Table.

VPC endpoints for web apps are supported in all AWS Regions where web apps are available.

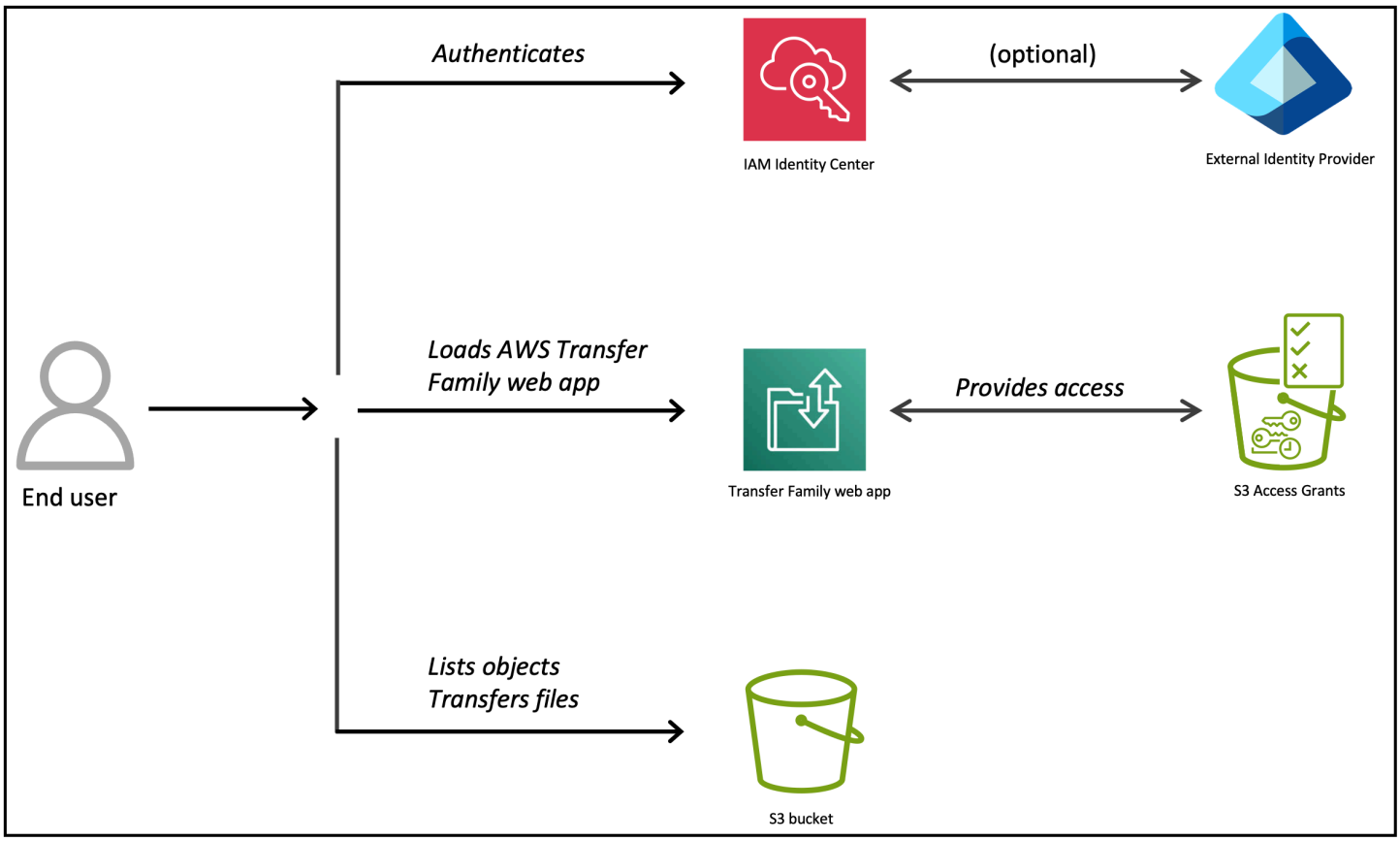
Browser compatibility for AWS Transfer Family web apps

Transfer Family web apps support the following browsers.

Browser	Version	Compatibility
Microsoft Edge	Latest 3 versions	Compatible
Mozilla Firefox	Latest 3 versions	Compatible
Google Chrome	Latest 3 versions	Compatible
Apple Safari	Latest 3 versions	Compatible

How to create a Transfer Family web app

The following diagram illustrates the Transfer Family web app architecture.



Based on the diagram, you can see that Transfer Family web apps interact with the following AWS services:

- Amazon S3 for storage and Amazon S3 Access Grants to acquire session credentials.
- AWS IAM Identity Center as the federated identity provider.
- Amazon CloudFront if you configure a custom URL for your web app.

Note the following limitations when using web apps.

- Maximum number of search results per query: 10,000
- The Amazon S3 buckets that are used by the Transfer Family web app must be in the same account as the web app itself. Cross-account buckets are not currently supported.
- Maximum search breadth per query: 10,000 searched files
- Maximum upload size per file: 160 GB (149 GiB)
- Maximum size file for copying: 5.36 GB (5 GiB)
- Folder names starting or ending with dots (.) are not supported

Prerequisites

In AWS Identity and Access Management, configure the necessary roles. Paste in the code blocks that we provide in the instructions. For information about configuring the necessary roles, see Configure IAM roles for Transfer Family web apps.

- Create an identity bearer role.
- Create an IAM role to be used by S3 Access Grants. S3 Access Grants assumes this IAM role to vend temporary credentials to the grantee for the registered Amazon S3 location.

Process to create a Transfer Family web app

To create your web app and get your end users up and running, you perform the following tasks:

1. *Configure IAM Identity Center to act as your federated identity provider*. Perform the following tasks in IAM Identity Center. For more details about configuring IAM Identity Center, see Configure your identity provider for Transfer Family web apps.
    a. Create an IAM Identity Center instance, if you don't already have one.
    b. Determine your identity source. It can be the default IAM Identity Center directory or a third-party provider (for example Okta).
    c. Create or identify the users or groups that will be using your web app.
    d. If you are using the IAM Identity Center directory for your identity source, note the user or group IDs that you create. You need them later when you create an access grant by using S3 Access Grants.
2. *In Amazon S3, configure Amazon S3 Access Grants.* For more information about S3 Access Grants, see Configure Amazon S3 Access Grants for Transfer Family web apps.
    - Create an S3 Access Grants instance if you don't already have one in that AWS Region.
    - Register your location using the IAM role.
    - Create the access grant.
3. *In Transfer Family, perform the following tasks.*
    a. Create the Transfer Family web app. For more information about how to create the Transfer Family web app, see Configure a Transfer Family web app.

> **Important**
>
> Set up Cross-origin resource sharing (CORS) for all Amazon S3 buckets that are used by your web app. For information about setting up CORS, see Set up Cross-origin resource sharing (CORS) for your bucket.

b. Assign users or groups to the web app. For more information about how to assign users and groups, see Assign or add users or groups to a Transfer Family web app.

c. (Optional) Update the access endpoint for your web app with a custom URL. For information about creating a custom URL, see Update your access endpoint with a custom URL.

d. Provide your end users with the access endpoint URL so that they can log in and interact with your web app.

Configure your identity provider for Transfer Family web apps

The following section describes how to configure your identity provider.

To begin, you must have an identity source. You can use an IAM Identity Center directory, AWS Directory Service for Microsoft Active Directory, or an external identity provider. Transfer Family uses IAM Identity Center as a federated identity provider, which is a system that stores user credentials and authenticates users across multiple organizations.

If you're not using an IAM Identity Center directory as your identity source, see the following topics:

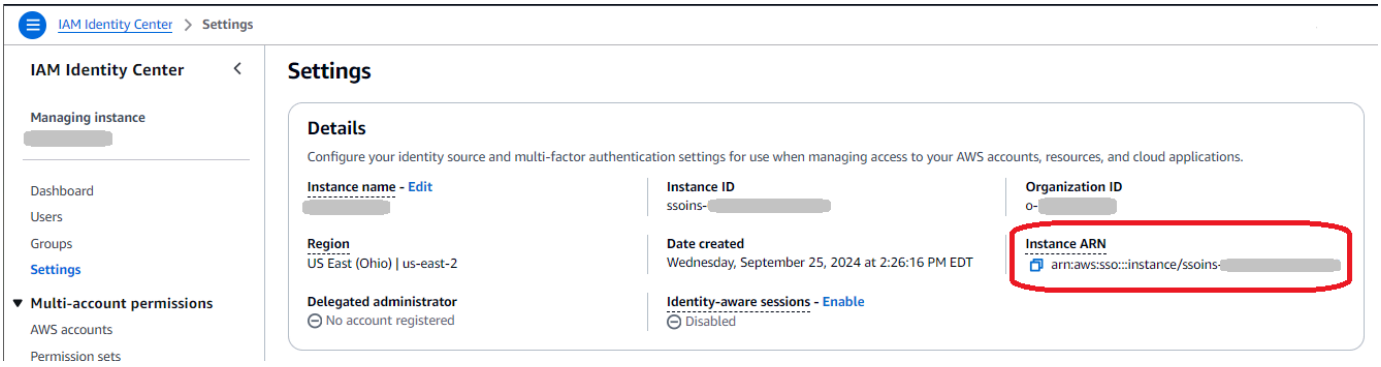
- Manage an external identity provider
- Connect to a Microsoft AD directory
- Organization and account instances of IAM Identity Center
- IAM Identity Center identity source tutorials

> **Note**
>
> You can only have one identity source in IAM Identity Center, per instance, per AWS Region. For details, see IAM Identity Center prerequisites and considerations.

If you plan to use the IAM Identity Center directory as your identity source, and want a quick setup, you can skip this topic and go to Create a Transfer Family web app to create an IAM Identity Center instance from the wizard.

To configure AWS IAM Identity Center for use with Transfer Family web apps

1. Sign in to the AWS Management Console and open the AWS IAM Identity Center console at https://console.aws.amazon.com/singlesignon/.
2. You can create and use either an account instance or an organization instance of AWS IAM Identity Center.
   - For details about account instances, see Create an account instance of AWS IAM Identity Center. With an account instance of IAM Identity Center, you can deploy supported AWS managed applications and OpenID Connect (OIDC)-based customer managed applications. Account instances support isolated deployments of applications in a single AWS account, leveraging IAM Identity Center workforce identity and access portal features.
   - For details about organization instances, see Organization instances of IAM Identity Center. You can centrally manage the access of users and groups with a single organization instance.
3. On the IAM Identity Center **Settings** page, note down your Instance ARN. You will need this value when you create an **Amazon S3 Access Grant** instance.



4. Create one or more users and, optionally, groups, to use with your Transfer Family web app. If you're using an IAM Identity Center directory as your identity provider, you can also add users directly from the web app itself. For more information, see Assign or add users or groups to a Transfer Family web app.

Configure IAM roles for Transfer Family web apps

You will need two roles: one to use as an identity bearer role for your web app, and a second to use for configuring an access grant. An identity bearer role is a role that includes an authenticated user's identity in its sessions. It's used to make requests to S3 Access Grants for data access on behalf of the user.

> **Note**
>
> You can skip the procedure for creating an identity bearer role. For information about having the Transfer Family service create the identity bearer role, see Create a Transfer Family web app.
> You can skip the procedure for creating an access grants role. In the procedure for creating an access grant, in the step where you register an S3 location, choose **Create new role**.

Create an identity bearer role

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Roles**, and then **Create role**.
3. Choose **Custom trust policy** and then paste in the following code.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service":"transfer.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetContext"
            ]
        }
    ]
}
```

4. Choose **Next** and then skip **Add permissions** and select **Next** again.

5. Enter a name, for example `web-app-identity-bearer`.
6. Choose **Create role** to create the identity bearer role.
7. Choose the role that you just created from the list, then in the **Permissions policies** panel, choose **Add permissions** > **Create inline policy**.
8. In the **Policy editor**, select **JSON** and then paste in the following code block.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetDataAccess",
                "s3:ListCallerAccessGrants",
                "s3:ListAccessGrantsInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

9. For the policy name, enter `AllowS3AccessGrants`, and then select **Create policy**.

Next, you create the role that S3 Access Grants assumes to vend temporary credentials to the grantee.

> **Note**
>
> If you allow the service to create the identity bearer role for you, that role sets confused deputy protection. Therefore, its code is different from what is displayed here.

Create an access grants role

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Roles**, and then **Create role**. This role should have permission to access your S3 data in the AWS Region.
3. Choose **Custom trust policy**, and then paste in the following code.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "access-grants.s3.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetContext"
            ]
        }
    ]
}
```

4. Choose **Next** add a minimal policy as described in Register a location. While not recommended, you can add the **AmazonS3FullAccess** managed policy, which may be too permissive for your needs.
5. Choose **Next**, and enter a name (for example `access-grants-location`).
6. Choose **Create role** to create the role.

> **Note**
>
> If you allow the service to create the access grants role for you, that role sets confused deputy protection. Therefore, its code is different from what is displayed here.

Configure a Transfer Family web app

This section describes the procedures for creating a Transfer Family web app. To assign users and groups that can use it, see Assign or add users or groups to a Transfer Family web app.

Note

Repeat these procedures to add additional web apps. You can reuse the IAM roles that you created earlier. Make sure to add the access endpoints for the new web apps to each bucket's Cross-origin resource sharing (CORS) policy.

Create a Transfer Family web app

Note

If you are not using the IAM Identity Center directory for your identity provider, don't attempt to create a web app until you have already set up IAM Identity Center and configured a third party identity provider, as described in Configure your identity provider for Transfer Family web apps.

Complete the following steps to create a Transfer Family web app.

To create a Transfer Family web app

1. Sign in to the AWS Management Console and open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Web apps**.
3. Choose **Create web app**.

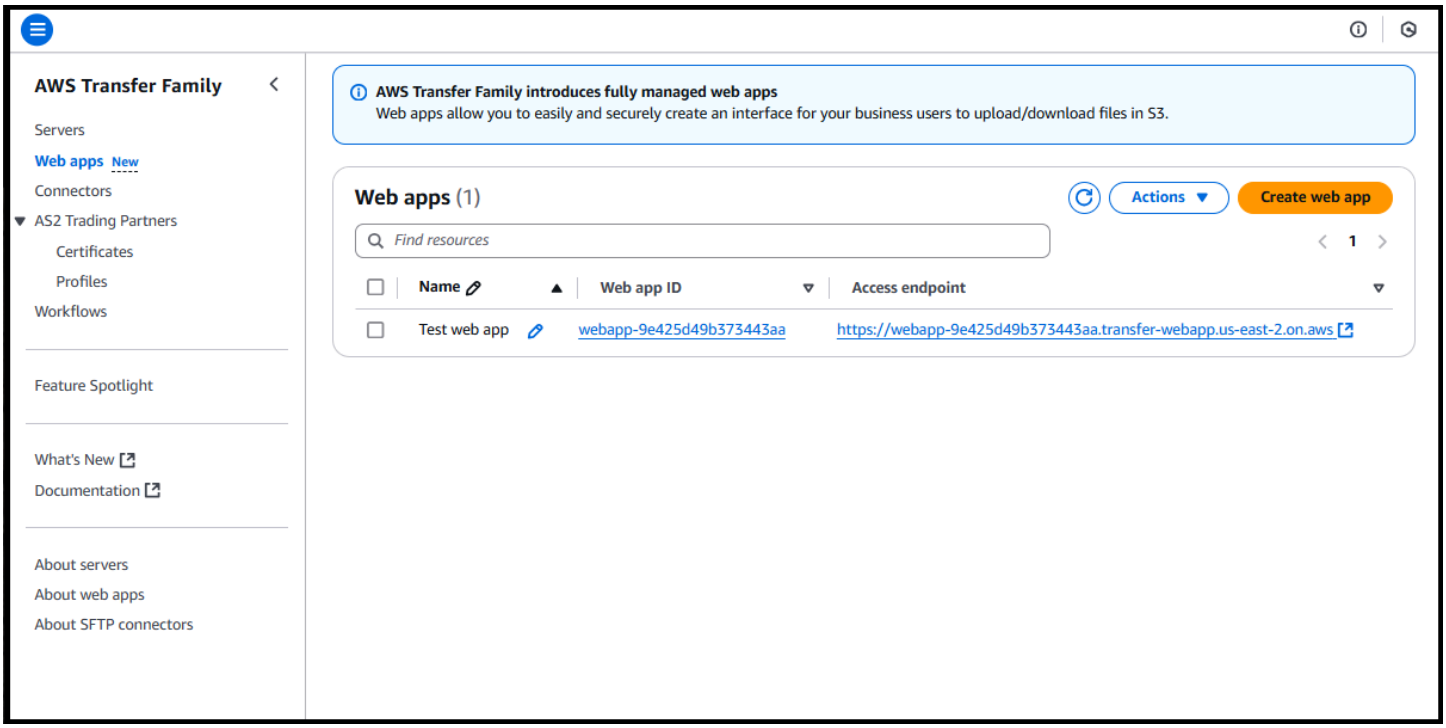
   For authentication access, the pane is populated as follows.

   - If you have already created either an organization or account instance in AWS IAM Identity Center, then you see this message: **Your AWS Transfer Family application connected to an account instance of IAM Identity Center**.
   - If you already have an account instance and are a member of an organization instance, you have the option to choose which instance to connect.
   - If you don't already have an account instance, or are a member in an organization instance, you're presented with the options to create an account instance.
4. For **Endpoint type**, choose the **Publicly accessible** endpoint type. For a **VPC hosted** endpoint, see Create a Transfer Family web app in a VPC.

5. In the **Permission type** pane, you can use a previously created role, or have the service create one for you.

   - If you have already created an identity bearer role, choose **Use an existing role** and choose your role from the **Select an existing role** menu.
   - To have the service create a role for you, choose **Create and use a new service role**.

6. In the **Web app units** pane, choose a value. One web app unit allows web app activity from up to 250 unique sessions. When creating a web app, you provision how many units you need based on your expected peak workload volumes. Changing your web app units has an impact on your billing. For information about pricing, see AWS Transfer Family Pricing.
7. If you are using Transfer Family in an AWS GovCloud (US) Region, you can select the **FIPS Enabled endpoint** checkbox in the **FIPS Enabled** pane. For all other AWS Regions, this option is unavailable.
8. (Optional) Add a tag to help you organize your web apps. We suggest that you add a tag with **Name** as the key and a descriptive name as the value.
9. Choose **Next**. On this screen, you can optionally provide a title for your web app. If you don't provide a title, the default title of **Transfer Web App** is supplied. You can also upload image files for your logo and favicon.
10. Choose **Next**, then choose **Create web app**.



> **Note**
>
> Make sure to set up a Cross-origin resource sharing (CORS) policy for all of the buckets that are accessed from the web app endpoint.

IP addressing

Transfer Family web apps use dual-stack endpoints, supporting both IPv4 and IPv6 connectivity. For authentication, web apps use AWS IAM Identity Center, which also supports IPv6 through dual-stack endpoints.

If your account had web apps in a Region when dual-stack IAM Identity Center endpoint support launched, all web apps in that account and Region continue to use the IPv4-only IAM Identity Center endpoint for backwards compatibility. This applies to both existing and newly created web apps. Accounts that did not have web apps at the time of launch use the IAM Identity Center dual-stack endpoint. If you need to change which IAM Identity Center endpoint your web app uses, contact AWS Support.

> **Note**
>
> To determine which endpoint your web app uses, inspect the URL when you are redirected to the IAM Identity Center sign-in page:
>
> - IPv4-only: `[Region].signin.aws.amazon.com`
> - Dual-stack: `[Region].sso.signin.aws`

If your organization uses firewalls or network gateways, allowlist both the existing IPv4 and the new dual-stack IAM Identity Center endpoints to ensure uninterrupted access regardless of which endpoint your web apps use. For the full list of domains and URL endpoints to allowlist, see Update firewalls and gateways to allow access to the AWS access portal in the *IAM Identity Center User Guide*.

For web apps hosted in a VPC, you can choose the IP address type for your VPC endpoint: IPv4 only or dual-stack (IPv4 and IPv6). The default is dual-stack for backward compatibility. Choose IPv4 if your VPC is IPv4-only or if you don't need IPv6 connectivity. You can change the IP address type after creation by updating the web app.

Create a Transfer Family web app in a VPC

This section describes the procedures for creating a Transfer Family web app in a VPC. You can host your web app's endpoint inside a virtual private cloud (VPC) to use for transferring data to and from an Amazon S3 bucket without going over the public internet. To assign users and groups that can use your web app, see Assign or add users or groups to a Transfer Family web app.

Note

To ensure a private end-to-end data flow when using a Transfer Family web app VPC endpoint, you must implement three additional components. First, set up a PrivateLink endpoint for Amazon S3 Control API operations, which is necessary for Amazon S3 Access Grants API calls. Second, configure an endpoint for Amazon S3 data access using either a PrivateLink Amazon S3 Gateway endpoint (for traffic from within your VPC) or an Amazon S3 Interface endpoint (for traffic from on-premises networks via VPN or Direct Connect). Third, lock down your Amazon S3 bucket access by updating the bucket policies to only permit traffic from these VPC endpoints. This combination ensures all data transfers remain within your private network infrastructure and never traverse the public internet.

Create a Transfer Family web app

Prerequisites

- AWS IAM Identity Center set up with configured identity provider. See Configure your identity provider for Transfer Family web apps.
- VPC and networking components set up. See Create a VPC.
- API endpoint set up for Amazon S3 Control operations. See Accessing Amazon S3 interface endpoints.
- VPC endpoints for Amazon S3 (Gateway or Interface) set up. See Types of VPC endpoints for Amazon S3. If you're using an interface endpoint, you must enable private DNS. For an example, see Introducing private DNS support for Amazon S3 with AWS PrivateLink.

Note

AWS IAM Identity Center does not support VPC endpoints; all authentication requests transit the public internet. Additionally, Transfer Family web applications require internet

access to load static content (such as JavaScript, CSS, and HTML files). The requirements for public internet access are separate from data access. Your VPC endpoint ensures that connections are routed through your VPC infrastructure.

To create a Transfer Family web app

1. Sign in to the AWS Management Console and open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Web apps**.
3. Choose **Create web app**. For authentication access, the pane is populated as follows.
   - If you have already created either an organization or account instance in AWS IAM Identity Center, then you see this message: **Your AWS Transfer Family application connected to an account instance of IAM Identity Center**.
   - If you already have an account instance and are a member of an organization instance, you have the option to choose which instance to connect.
   - If you don't already have an account instance, or are a member in an organization instance, you're presented with the options to create an account instance.
4. In the **Endpoint configuration** section, choose how your users will access your web app:
   - **Publicly accessible**: Your web app endpoint is accessible over the public via HTTPS. This option does not require any VPC configuration, making it straightforward to set up and suitable for applications intended for wide public use.
   - **VPC hosted**: Your web app endpoint is hosted within your Virtual Private Cloud (VPC), providing private network access through your VPC network, AWS Direct Connect, or VPN connections. This option offers enhanced security through network isolation and is recommended for internal applications.

     When configuring a VPC hosted endpoint, you'll need to specify:
     - **VPC**: Select an existing VPC or create a new one. A **Create a VPC** button is available.
     - **IP address type**: Choose the IP address type for your VPC endpoint. Select **IPv4** for IPv4-only connectivity, or **Dual-stack** for both IPv4 and IPv6 connectivity. The default is Dual-stack.
     - **Availability zones**: Choose the availability zones where your endpoint will be deployed.
     - **Subnets**: Select subnets within each chosen availability zone.

- **Security groups**: Select or create security groups to control access based on source IP addresses. If not specified, the VPC's default security group is used. Manage security groups through the VPC Console. Configure your VPC security groups to allow inbound traffic from your network over HTTPS on TCP port 443. This is required for IAM Identity Center authentication and web app static content loading.

Post-creation steps

- Make sure to set up a Cross-origin resource sharing (CORS) policy for all of the buckets that are accessed from the web app endpoint. See Cross-origin resource sharing (CORS) policy.
- Update your bucket policy to allow traffic only originating from your VPC through your VPC endpoint. See Restricting access to a specific VPC endpoint.
- Assign or add users or groups to Transfer Family web app. See Assign or add users or groups to a Transfer Family web app.

Cross-origin resource sharing (CORS) policy

You must set up cross-origin resource sharing (CORS) for all buckets that are used by your web app. For more information about CORS, see Set up Cross-origin resource sharing (CORS) for your bucket.

> **Important**
>
> Before using the following example policy, replace the Allowed Origin with your access endpoint. Otherwise, your end users will receive an error when they attempt to access a location on your web app.

Example policy:

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "PUT",
```

```
      "POST",
      "DELETE",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://vpce-1234567-example.vpce-mq.transfer-webapp.us-east-1.on.aws"
    ],
    "ExposeHeaders": [
      "last-modified",
      "content-length",
      "etag",
      "x-amz-version-id",
      "content-type",
      "x-amz-request-id",
      "x-amz-id-2",
      "date",
      "x-amz-cf-id",
      "x-amz-storage-class",
      "access-control-expose-headers"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

Restricting access to a specific VPC endpoint

The following is an example of an Amazon S3 bucket policy that restricts access to a specific bucket, `amzn-s3-demo-bucket`, only from the VPC endpoint with the ID `vpce-1a2b3c4d`. If the specified endpoint is not used, the policy denies all access to the bucket. The `aws:SourceVpce` condition specifies the endpoint. The `aws:SourceVpce` condition doesn't require an ARN for the VPC endpoint resource, only the VPC endpoint ID. For more information about updating your bucket policy to only allow traffic originating from your VPC, see Controlling access from VPC endpoints with bucket policies. For more information about using conditions in a policy, see Bucket policy examples using condition keys. As a pre-requisite to applying this policy, you should create an Amazon S3 VPC endpoint.

> **Important**
>
> Before using the following example policy, replace the VPC endpoint ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.

```
{
  "Version":"2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    {
      "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket",
                   "arn:aws:s3:::amzn-s3-demo-bucket/*"],
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpce": "vpce-1a2b3c4d"
        }
      }
    }
  ]
}
```

Assign or add users or groups to a Transfer Family web app

After you create a Transfer Family web app, you can assign users and groups who can then access the web app. You can either retrieve users that are already created and stored in IAM Identity Center, or you can add new users directly (if you're using an IAM Identity Center directory as your identity provider). If you add new users, they are also added to your IAM Identity Center instance.

Note the following:

- You can only add new users if you are using the IAM Identity Center directory as your identity source and have the proper permissions. If you are a member of an organization instance, you might not have the necessary permissions to add users.

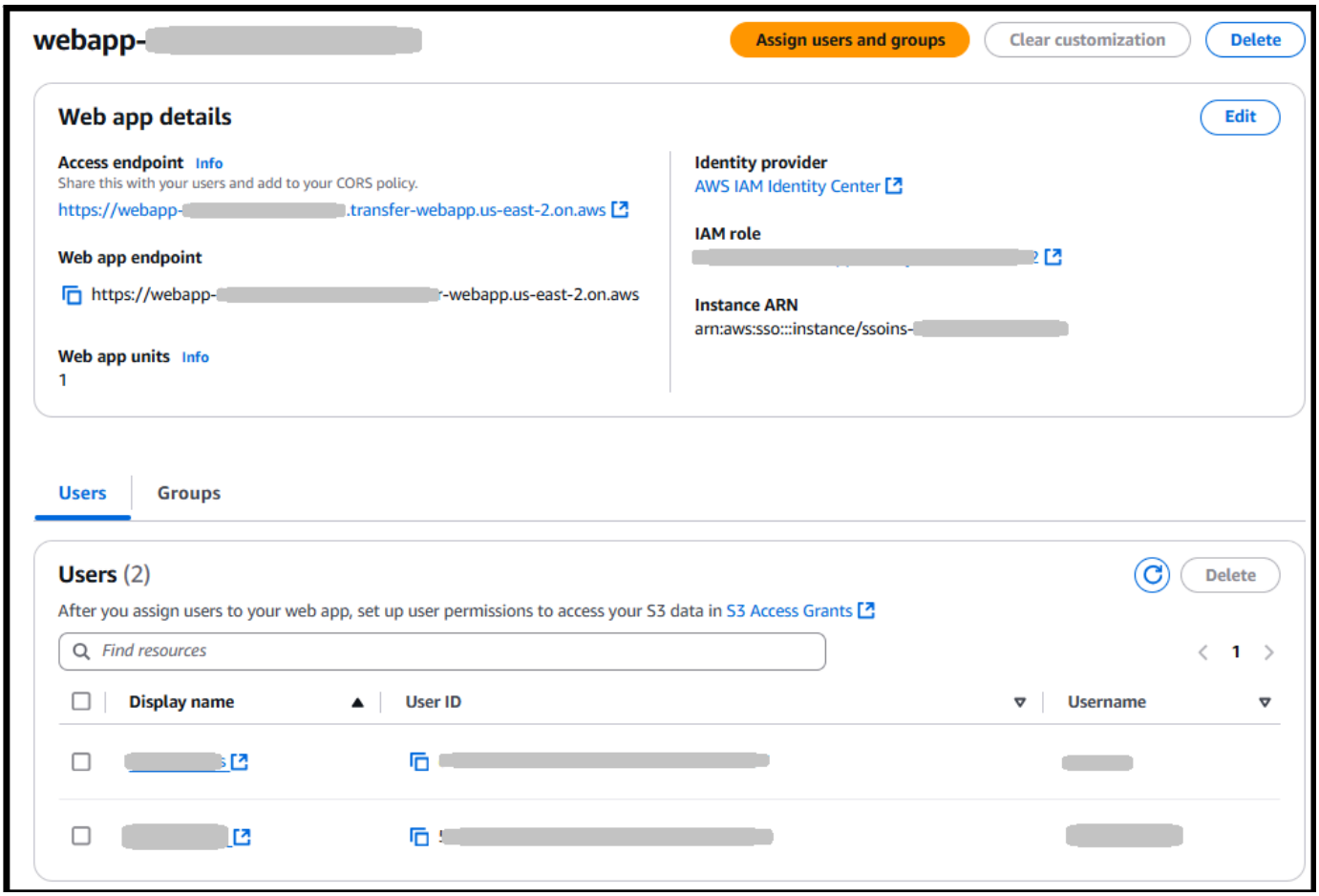
  > **Note**
  >
  > If you don't assign users or groups to your application, your users will get an error when they attempt to log into your web app.

- If you create a new user, you must also create an S3 access grant for this user so that they can access data on your web app.

- After you create a new user, that user receives an onboarding email from IAM Identity Center with directions for how to proceed.

To assign users to a Transfer Family web app

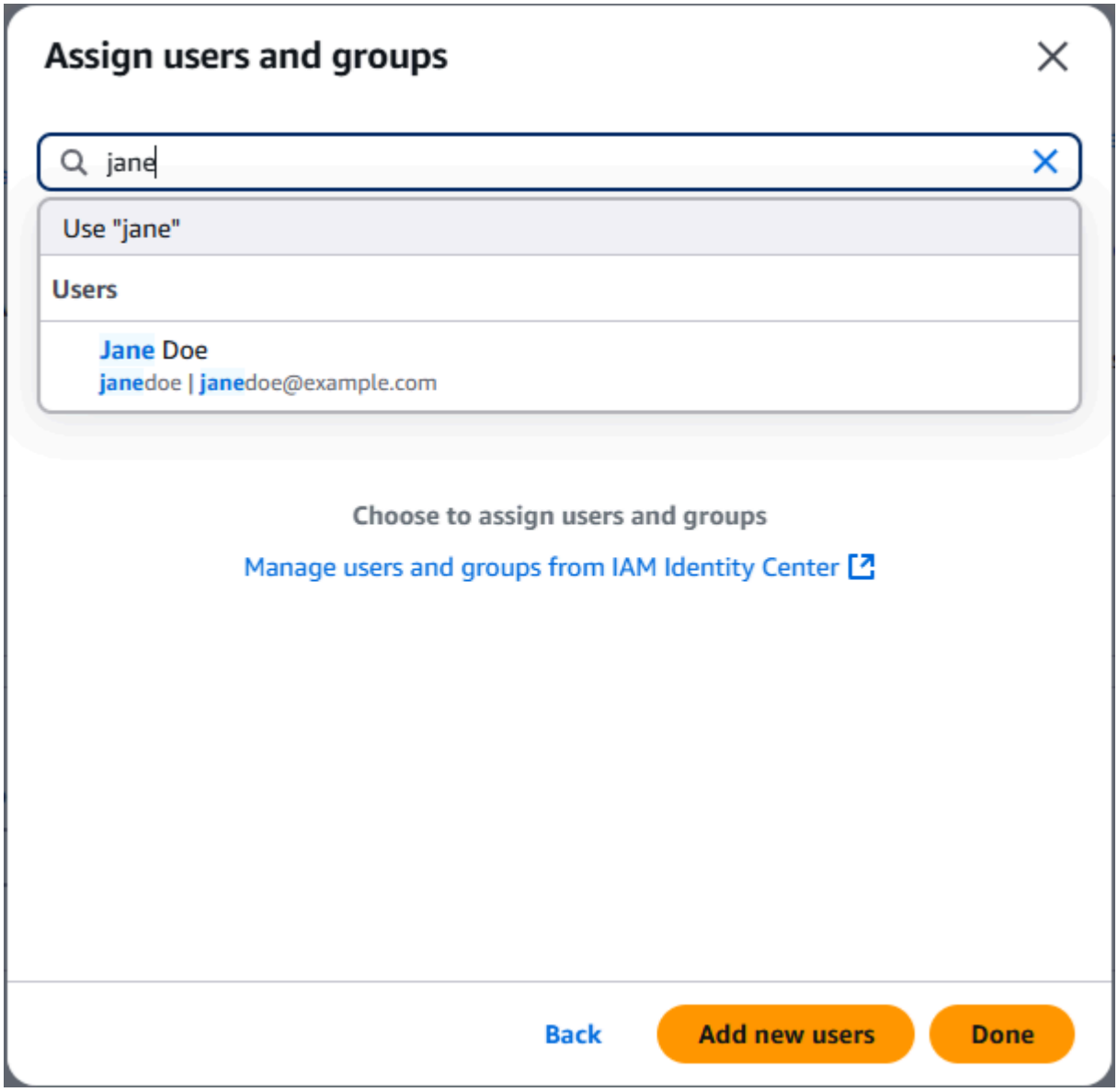
1. Navigate to your web app list, and choose the one that you want to edit.
2. Choose **Assign users and groups**.



3. To assign users that you previously created in IAM Identity Center, select **Assign existing users and groups**. To create new users, skip ahead to step 4.

   a. An information screen appears. Choose **Get started** to continue.

   b. Search for the user. Note that no users appear until you begin entering your search criteria. You must search by the *display name*, not the *username*, if different. Only exact matches are returned. If you can't find your user, navigate to the IAM Identity Center management console, find the user, then copy and paste their display name here.



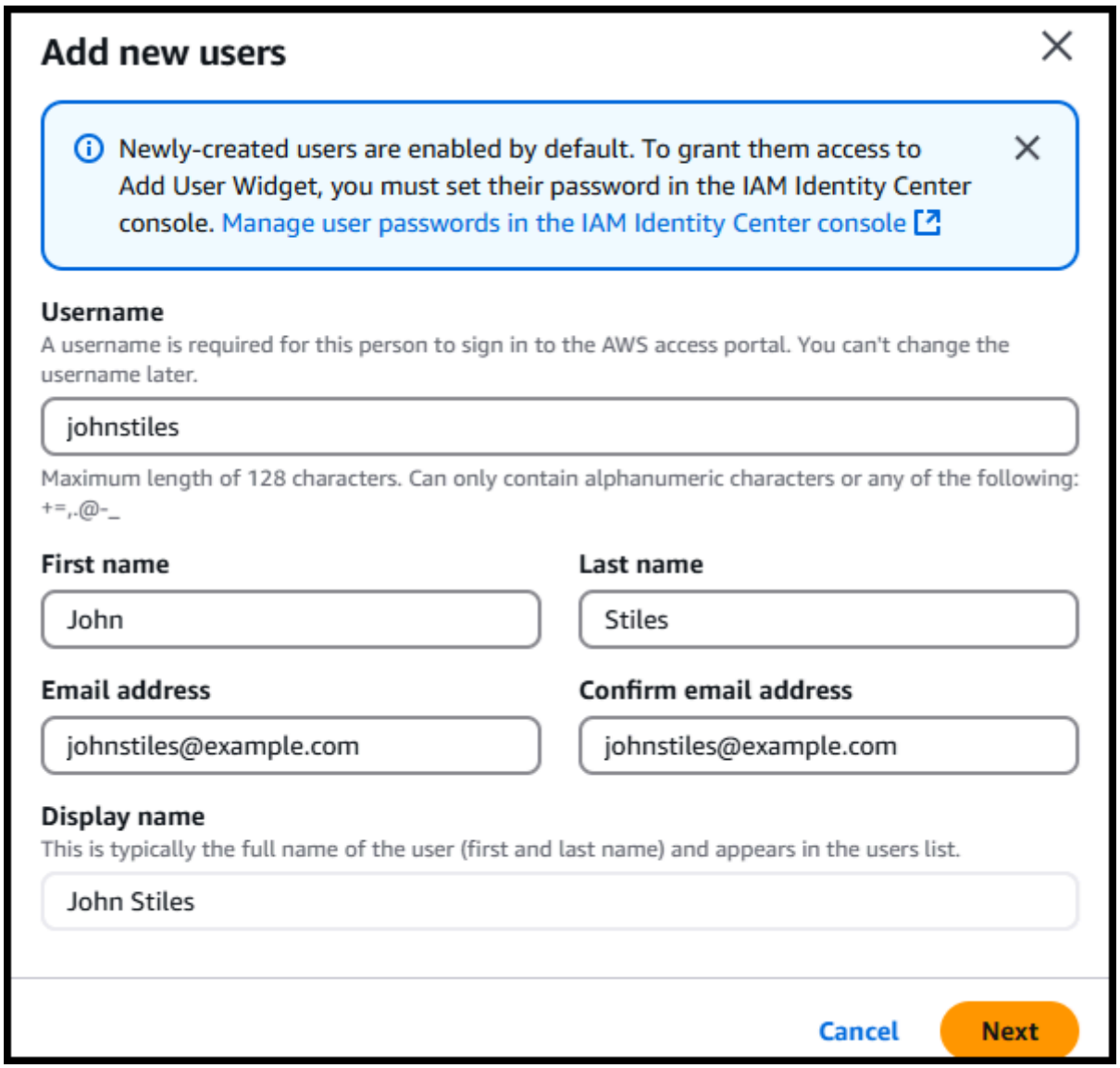
c. Choose the users and groups to add, then choose **Assign**.

4. To create a new user, select **Add and assign new users**.

   a. An information screen appears. Choose **Get started** to continue.

   b. Choose **Add new users**.

   c. Enter the following user details into the dialog box: username, first and last name, and an email address.



d. Choose **Next**, then choose **Add** to add the user and close the dialog box, or **Add new user** to create another user.

Set up Cross-origin resource sharing (CORS) for your bucket

You must set up cross-origin resource sharing (CORS) for all buckets that are used by your web app. A *CORS configuration* is a document that defines rules that identify the origins that you will allow to access your bucket. For more information about CORS, see Configuring cross-origin resource sharing (CORS).

Important

If you don't set up CORS, your end users receive an error when they attempt to access a location on your web app.

To set up Cross-origin resource sharing (CORS) for your Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Buckets** from the left navigation panel and search for your bucket in the search dialog, then choose the **Permissions** tab.
3. In **Cross-origin resource sharing (CORS)**, choose **Edit** and paste in the following code. Replace *`WebAppEndpoint`* with the actual access endpoint for your web app. This can be either the VPC hosted or public access endpoint that's created when the web app is created, or a custom access endpoint, if you create one. Make sure not to enter trailing slashes, because doing so causes errors when users attempt to log on to your web app.

   - Incorrect example: `https://webapp-c7bf3423.transfer-webapp.us-east-2.on.aws/`
   - Correct examples:
     - `https://webapp-c7bf3423.transfer-webapp.us-east-2.on.aws`
     - `https://vpce-05668789767a-fh45z079.vpce-mq.transfer-webapp.us-east-1.on.aws`

   If you are reusing a bucket for multiple web apps, append their endpoints to the `AllowedOrigins` list.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "PUT",
      "POST",
      "DELETE",
```

```
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://WebAppEndpoint"
    ],
    "ExposeHeaders": [
      "last-modified",
       "content-length",
      "etag",
      "x-amz-version-id",
      "content-type",
      "x-amz-request-id",
      "x-amz-id-2",
      "date",
      "x-amz-cf-id",
      "x-amz-storage-class",
      "access-control-expose-headers"
     ],
    "MaxAgeSeconds": 3000
  }
]
```

4. Choose **Save changes** to update the CORS.

To test your CORS configuration, see Testing CORS.

Configure Amazon S3 Access Grants for Transfer Family web apps

This topic describes how to add an access grant using Amazon S3 Access Grants. This access grant defines access to your data directly to your users and groups in your corporate directory and vends just-in-time, least privilege, temporary credentials based on grants. An individual grant in an S3 Access Grants instance allows a specific user or group in a corporate directory—to get access within a location that is registered in your S3 Access Grants instance. For more details, see S3 Access Grants concepts in the Amazon S3 User Guide.

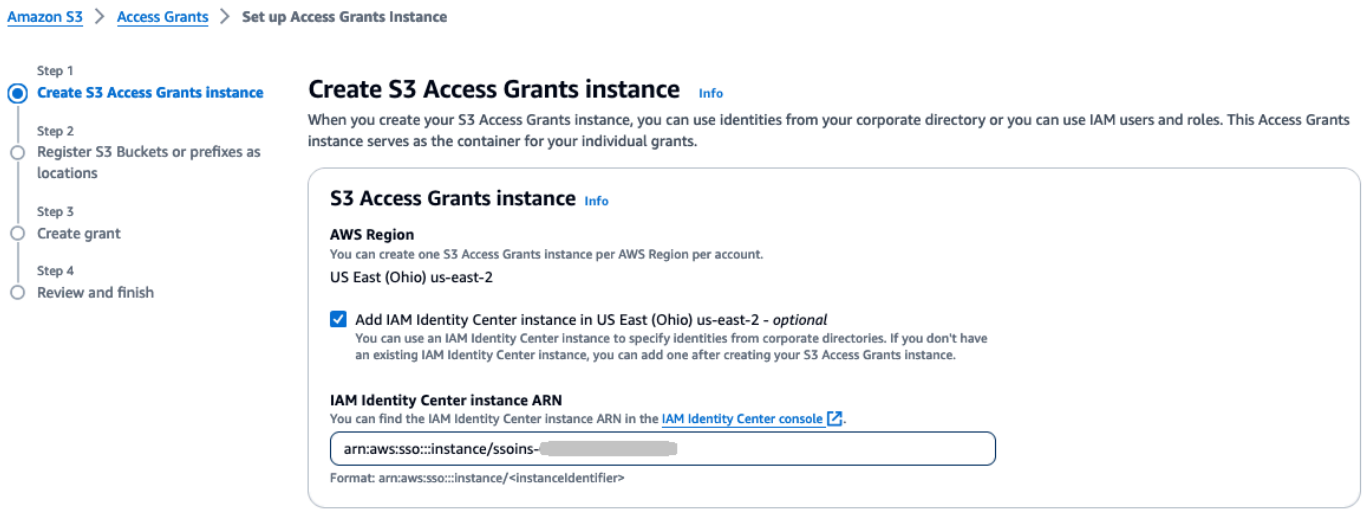
> **Note**
>
> You can't use the IAM Identity Center directory with S3 Access Grants other than with Transfer Family web apps.

You must specify an Amazon S3 access grant for identity propagation. An Amazon S3 access grant stores the data that your end users must access. When your end users sign in to your Transfer Family web app, S3 Access Grants passes a user's identity to the trusted application. This section describes how to add and configure an Amazon S3 access grant instance and then an access grant for an Amazon S3 bucket.

> **Note**
>
> Have your IAM Identity Center instance ARN and user or group ID ready, as you need them to complete setting up your access grant.

To create a grant using Amazon S3 Access Grants

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket, or note an existing bucket to use with your web app. For information on creating buckets, see the Amazon S3 User Guide.
3. From the left navigation pane, choose **Access Grants**.
4. Choose **Create S3 Access Grants instance** and provide the following information.
   - Select **Add IAM Identity Center instance in *your-Region*** where *your-Region* is your AWS Region. Keep this box cleared if you are not using IAM Identity Center as your identity provider.
   - Paste in your IAM Identity Center instance ARN.



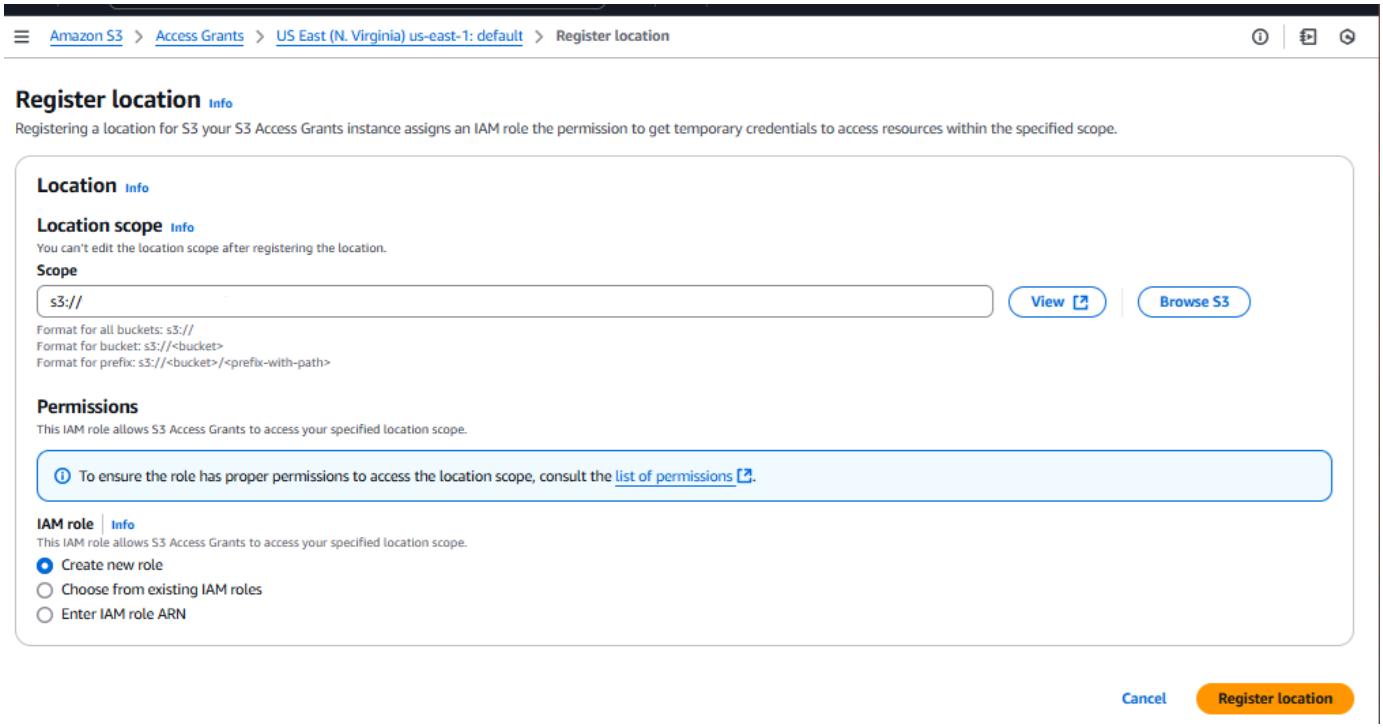
Choose **Next** to continue.

5. **Register S3 Buckets or prefixes as locations**. We recommend that you register the default location, `s3://`, and map it to an IAM role. The location at this default path covers access to all of your Amazon S3 buckets in the AWS Region of your account. When you create an access grant, you can narrow the scope to a bucket, a prefix, or an object within the default location.

   Provide the following information.

   - For the **Scope**, browse for a bucket or enter the name of your bucket, and optionally a prefix.
   - For the IAM role, choose **Create new role** to have the service create a role.

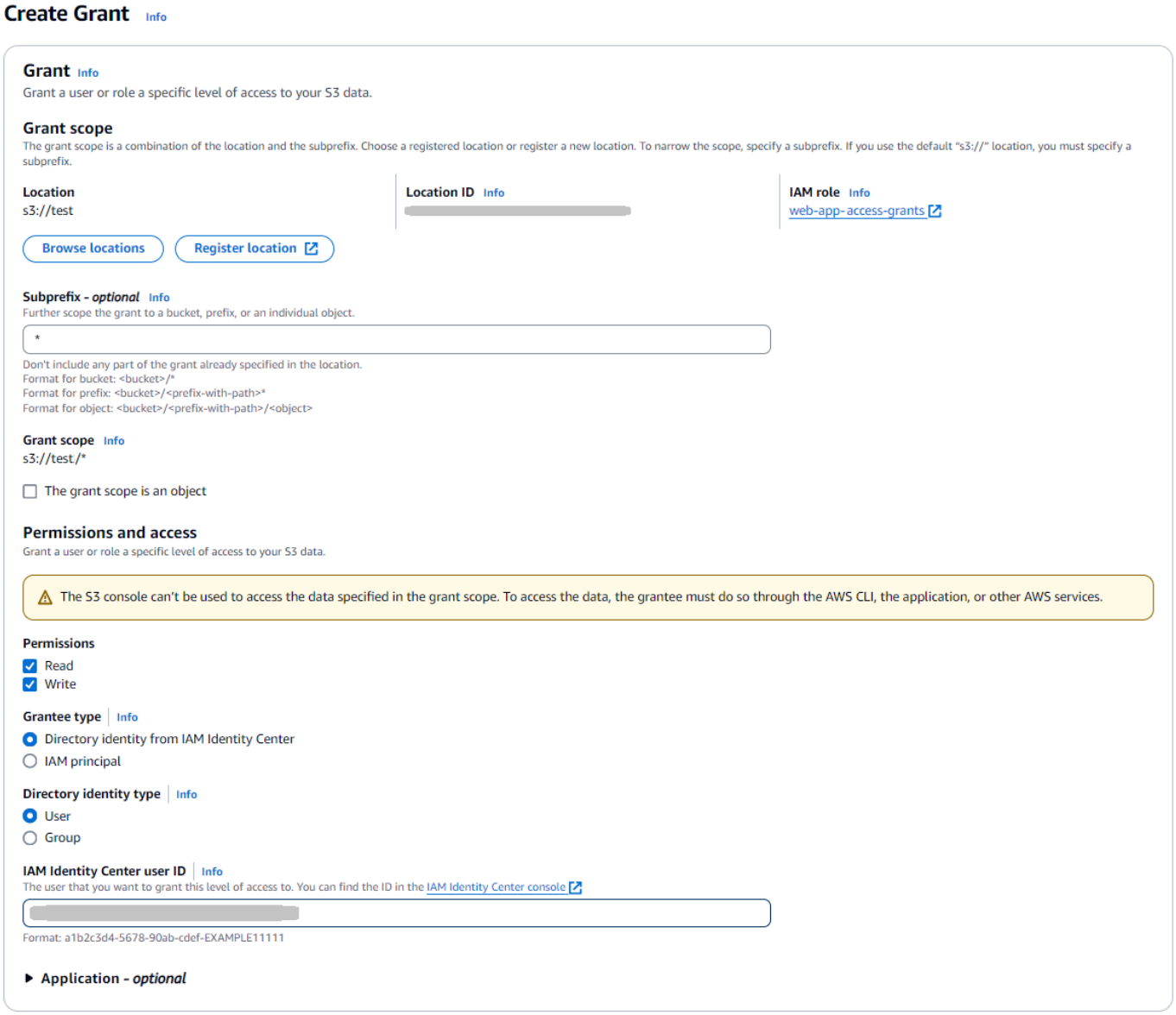
     Alternatively, you can create the role yourself, as described in Configure IAM roles for Transfer Family web apps, and then enter its ARN here.



Choose **Next** to continue.

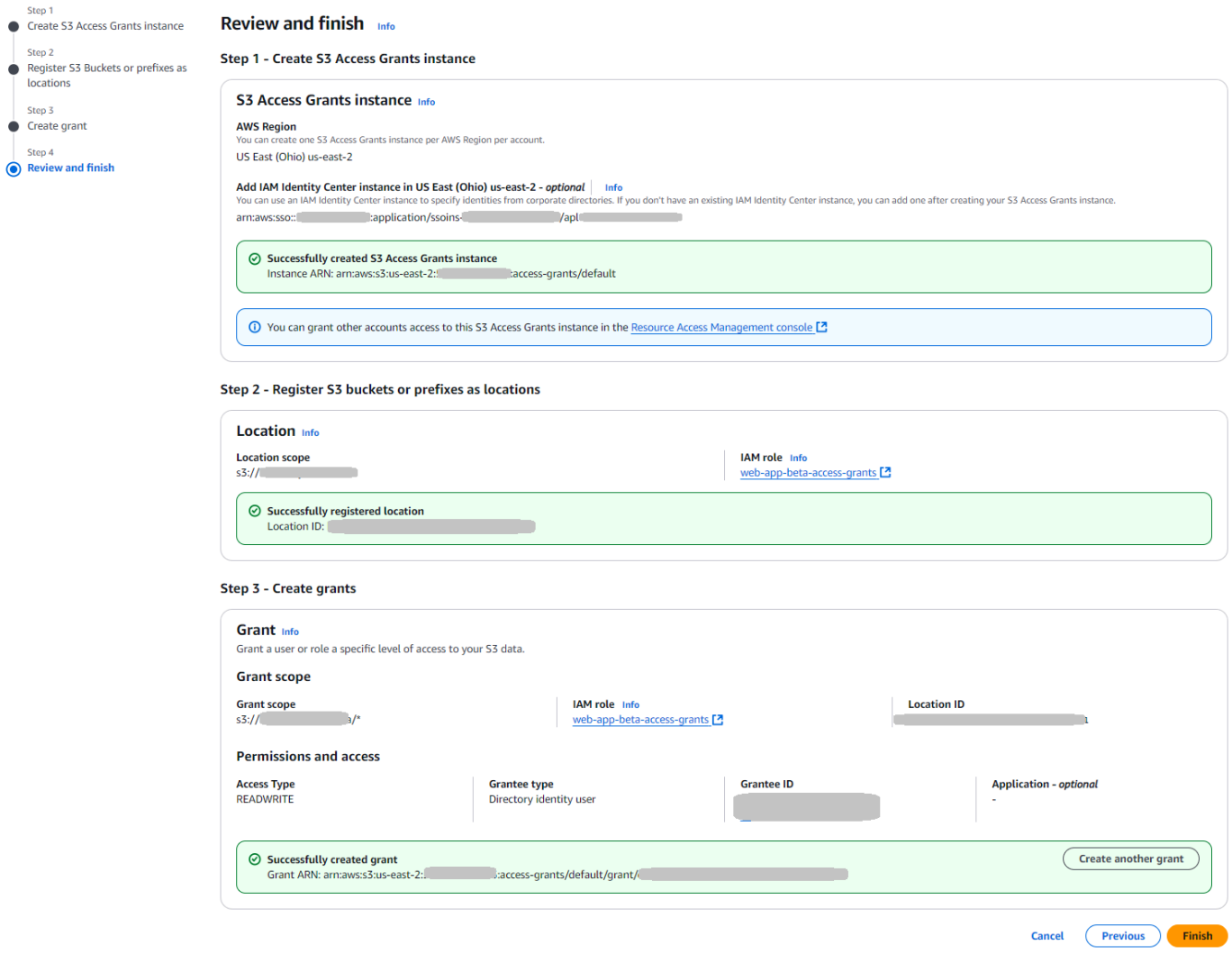
6. In the **Create Grant** screen, provide the following details.

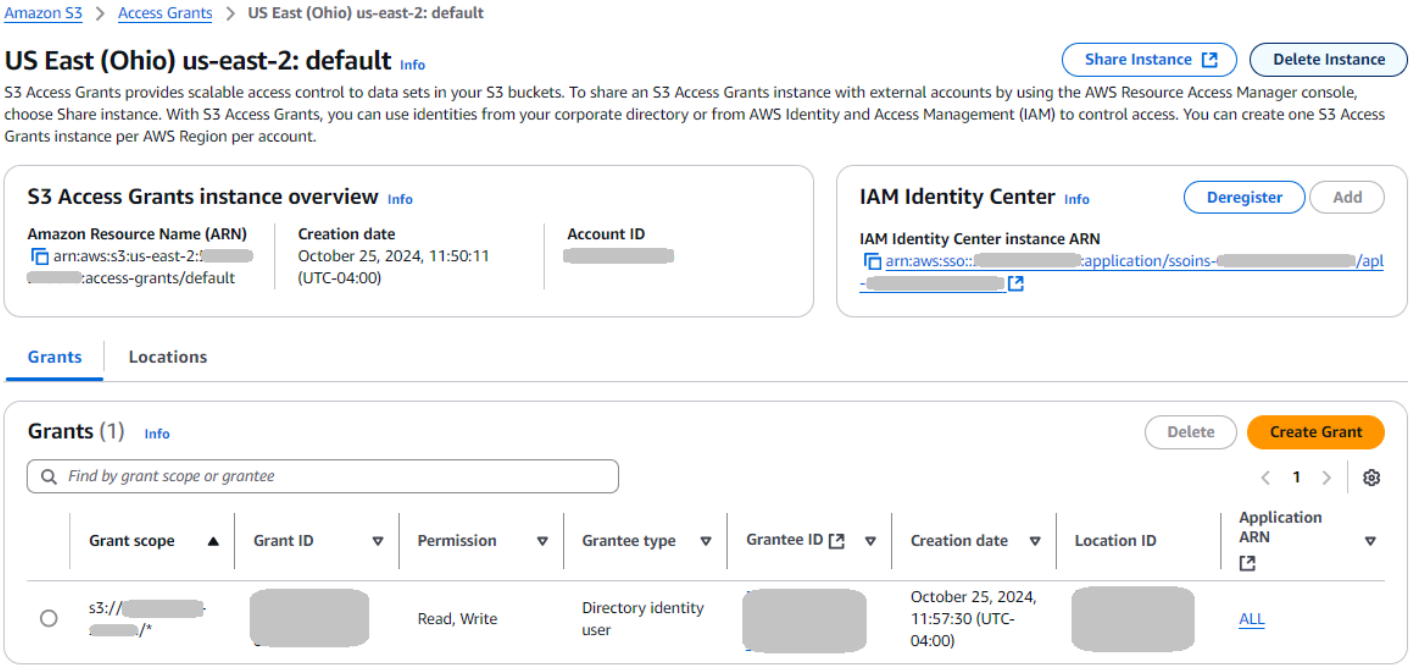
   - For **Permissions**, select **Read** and **Write**. The access grant permissions can be either read-only or read & write, but write-only is not supported.
   - For **Grantee type**, choose **Directory identity from IAM Identity Center**.
   - For **Directory identity type**, select **User** or **Group**, depending on which type you want to register now.
   - In **IAM Identity Center user/group ID**, paste in the ID for your user or group. This ID is available in the **IAM Identity Center** console and in your Transfer Family web app in your users and groups table.



Choose **Next**.

7. Review the settings on the screen. If everything is correct, choose **Finish** to create the access grant. Alternatively, you can choose **Cancel** or **Previous** to make changes.


Step 1
Create S3 Access Grants instance
Step 2
Register S3 Buckets or prefixes as locations
Step 3
Create grant
Step 4
Review and finish
Review and finish Info
Step 1 - Create S3 Access Grants instance
S3 Access Grants instance Info
AWS Region
You can create one S3 Access Grants instance per AWS Region per account.
US East (Ohio) us-east-2
Add IAM Identity Center instance in US East (Ohio) us-east-2 - optional Info
You can use an IAM Identity Center instance to specify identities from corporate directories. If you don't have an existing IAM Identity Center instance, you can add one after creating your S3 Access Grants instance.
arn:aws:sso::[redacted]:application/ssoins-[redacted]/apl[redacted]
Successfully created S3 Access Grants instance
Instance ARN: arn:aws:s3:us-east-2:[redacted]:access-grants/default
You can grant other accounts access to this S3 Access Grants instance in the Resource Access Management console
Step 2 - Register S3 buckets or prefixes as locations
Location Info
Location scope
s3://[redacted]
IAM role Info
web-app-beta-access-grants
Successfully registered location
Location ID: [redacted]
Step 3 - Create grants
Grant Info
Grant a user or role a specific level of access to your S3 data.
Grant scope
Grant scope
s3://[redacted]a/*
IAM role Info
web-app-beta-access-grants
Location ID
[redacted]
Permissions and access
Access Type
READWRITE
Grantee type
Directory identity user
Grantee ID
[redacted]
Application - optional
-
Successfully created grant
Grant ARN: arn:aws:s3:us-east-2:[redacted]:access-grants/default/grant/[redacted]
Create another grant
Cancel
Previous
Finish



This completes the setup for your web app. The users and groups that you've configured can visit the web app at the access point, log in, and upload and download files.

Update your access endpoint with a custom URL

The default access endpoint that is created with your web app contains service-generated identifiers. To provide a branded experience, you may want to provide a custom URL for your users to access your Transfer Family web app. This topic describes how to update your access endpoint with a custom URL.

To customize your web app URL

1. Set up TLS termination and forwarding for your custom domain name. The steps differ depending on whether you use a public or VPC web app.

    a. For a public web app, create a CloudFront distribution by using the Transfer Family supplied AWS CloudFormation template, CloudFormation stack template.

        i. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.

        ii. Choose **Create stack** and specify the following.

- In the **Prerequisite - Prepare template** section, choose **Choose an existing template**.
- In the **Specify template** section, choose **Upload a template file**.
- Save the CloudFormation stack template file, and then upload it here.

iii. Choose **Next** and provide the following information.

- **WebAppEndpoint**: copy the value from your web app
- **AccessEndpoint**: provide the custom domain name that you want to use
- **AcmCertificateArn**: provide the ARN for a public or private SSL/TLS certificate that is stored in AWS Certificate Manager

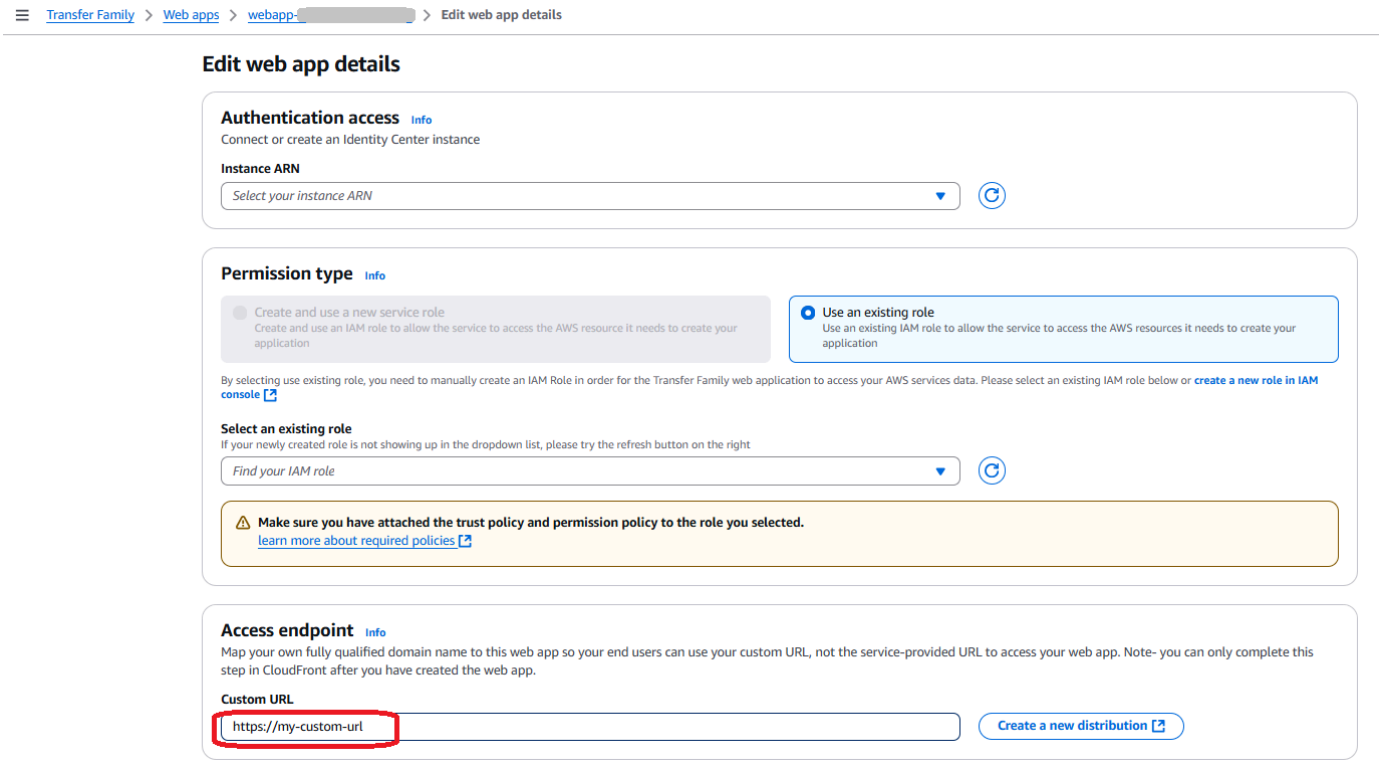
iv. Complete the CloudFormation wizard until your new stack is created.

v. After the stack is created, you need to create DNS records to route traffic for your custom domain name to the CloudFront distribution. If you're using Route 53 for DNS, you can create an Alias or CNAME record to the CloudFront distribution name. For information about using Amazon Route 53 with CloudFront, see Configuring Amazon Route 53 to route traffic to a CloudFront distribution.

b. For a VPC web app, you must provide your own TLS termination solution for your custom domain. Your solution must use a valid SSL/TLS certificate for your custom domain. Configure your solution to forward traffic to your web app's PrivateLink DNS name.

> **Note**
>
> For your own forwarding solution, configure the following:
>
> - Ensure that you are using HTTPS.
> - Forward all HTTP methods: GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE.
> - Forward all HTTP headers in the request and response, except the `Host` header. The `Host` header sent to the web app should be the WebAppEndpoint value.
> - Disable all response caching.

2. In your web app, edit the **Access endpoint**, updating the **Custom URL** to the URL that you want to use.



3. Update your cross-origin resource sharing policy by replacing the default access endpoint with the following line in the `AllowedOrigins` code block:

```
"https://custom-url"
```

You need to make this change for each bucket used by your web app.

After you make your update, the `AllowedOrigins` section of your CORS policy should look like the following:

```
"AllowedOrigins": [
    "https://custom-url"],
```

You need only a single AllowedOrigins entry for each Transfer Family web app.

See the Set up Cross-origin resource sharing (CORS) for your Amazon S3 bucket procedure for more details.

You can now visit your custom access endpoint, and share this link with your end users.

CloudTrail logging for Transfer Family web apps

CloudTrail is an AWS service that creates a record of actions taken within your AWS account. It continuously monitors and records API operations for activities like console sign-ins, AWS Command Line Interface commands, and SDK/API operations. This allows you to keep a log of who took what action, when, and from where. CloudTrail helps with auditing, access management, and regulatory compliance by providing a history of all activity in your AWS environment.

For Transfer Family web apps, you can track both authentication events and data access operations performed by your users. To enable comprehensive logging, you need to:

1. Configure CloudTrail to log management events for tracking authentication activities.
2. Enable Amazon S3 data events to track file operations performed through your web app.

See also

- CloudTrail use cases for IAM Identity Center
- Understanding IAM Identity Center sign-in events
- CloudTrail userIdentity element
- Enabling CloudTrail event logging for S3 buckets and objects
- Amazon S3 CloudTrail events

Enabling Amazon S3 data events

To track file operations performed through Transfer Family web apps on your Amazon S3 buckets, you need to enable data events for those buckets. Data events provide object-level API activity and are particularly useful for tracking file uploads, downloads, and other operations performed by web app users.

To enable Amazon S3 data events for your Transfer Family web app:

1. Open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. In the navigation pane, choose **Trails**, and then select an existing trail or create a new one.
3. Under **Advanced event selectors**, choose **Edit**.
4. Choose **Add advanced event selector**.
5. For the first field selector:

- Set **Field** to `eventCategory`
- Set **Operator** to **Equals**
- Set **Value** to `Data`

6. Choose **Add field** and for the second field selector:
   - Set **Field** to `resources.type`
   - Set **Operator** to **Equals**
   - Set **Value** to `AWS::S3::Object`
7. (Optional) To log events for specific buckets only, choose **Add field** and add:
   - Set **Field** to `resources.ARN`
   - Set **Operator** to **Starts with**
   - Set **Value** to `arn:aws:s3:::your-bucket-name/`
8. Choose **Save changes**.

Alternatively, you can use the legacy data events configuration:

1. Under **Data events**, choose **Edit**.
2. For **Data event type**, select **S3 bucket and object events**.
3. Choose the Amazon S3 buckets to log data events for. You can select **All current and future S3 buckets** or specify individual buckets.
4. Choose whether to log **Read** events, **Write** events, or both.
5. Choose **Save changes**.

After enabling data events, you can access these logs in the Amazon S3 bucket configured for CloudTrail. The logs include details such as the user who performed the action, the action timestamp, the specific object affected, and the `onBehalfOf` field that helps trace the `userId` for actions performed through Transfer Family web apps.

Finding and viewing your logs

There are several ways to find and view CloudTrail logs for your Transfer Family web app:

Using the CloudTrail console

The fastest way to view recent events:

1. Open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. Choose **Event history**.
3. Filter events by:
   - **Event source**: `signin.amazonaws.com` for web app events
   - **Event source**: `s3.amazonaws.com` for file operations
4. Click any event to view detailed information.

Accessing logs in Amazon S3

To access the complete log files stored in Amazon S3:

1. Identify your CloudTrail trail's Amazon S3 bucket:

```
aws cloudtrail describe-trails --query 'trailList[*].[Name,S3BucketName]' --output
 table
```

2. Navigate to the log files in Amazon S3:

```
aws s3 ls s3://your-cloudtrail-bucket/AWSLogs/account-id/CloudTrail/region/YYYY/MM/
DD/
```

3. Download and search log files for your web app ID:

```
aws s3 cp s3://your-cloudtrail-bucket/AWSLogs/account-id/CloudTrail/region/YYYY/MM/
DD/ . --recursive
gunzip *.json.gz
grep -l "webapp-1a2b3c4d5e6f7g8h9" *.json
```

Using AWS CLI to search events

Search for specific web app events using the AWS CLI:

```
aws logs filter-log-events \
  --log-group-name /aws/cloudtrail/your-trail-name \
  --filter-pattern "webapp-1a2b3c4d5e6f7g8h9" \
  --start-time $(date -d "1 day ago" +%s)000
```

Or search for authentication events:

```
aws logs filter-log-events \
  --log-group-name /aws/cloudtrail/your-trail-name \
  --filter-pattern "UserAuthentication" \
  --start-time $(date -d "1 day ago" +%s)000
```

Authentication log examples

CloudTrail logs authentication events for Transfer Family web apps, which can help you track successful and failed sign-in attempts. These logs are particularly useful for security monitoring and compliance purposes.

Topics

- Example log entry for credential verification
- Example log entry for sign-in authentication
- Example log entry for ListCallerAccessGrants
- Example log entry for GetDataAccess event

Example log entry for credential verification

The following example shows a CloudTrail log entry for a credential verification event that occurs during the authentication process.


```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "Unknown",
        "principalId": "123456789012",
        "arn": "",
        "accountId": "123456789012",
        "accessKeyId": "",
        "userName": "demo-user-2",
        "onBehalfOf": {
            "userId": "f12bb510-a011-702f-10dd-5607e2776dbc",
            "identityStoreArn": "arn:aws:identitystore::123456789012:identitystore/
d-9a670c546e"
        },
        "credentialId": "58138a11-87e5-401d-8f0b-7161c9389112"
    },
    "eventTime": "2025-08-08T15:29:30Z",
```

```
    "eventSource": "signin.amazonaws.com",
    "eventName": "CredentialVerification",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.224",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "AuthWorkflowID": "f304a48b-7b6d-41c8-b136-4f49c91c1f31",
        "CredentialType": "PASSWORD"
    },
    "requestID": "ff936828-4a81-453c-802d-81368b6bca1a",
    "eventID": "70cb7008-493d-42c2-a9eb-38bf168af6a8",
    "readOnly": false,
    "eventType": "AWSServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "CredentialVerification": "Success"
    },
    "eventCategory": "Management"
}
```

This event provides additional detail about the credential verification step in the authentication process, showing the specific credential ID and authentication workflow ID used.

Example log entry for sign-in authentication

The following example shows a CloudTrail log entry for a successful user authentication event during web app sign-in using IAM Identity Center.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "Unknown",
        "principalId": "123456789012",
        "arn": "",
        "accountId": "123456789012",
        "accessKeyId": "",
        "userName": "demo-user-2",
        "onBehalfOf": {
            "userId": "f12bb510-a011-702f-10dd-5607e2776dbc",
```

```
            "identityStoreArn": "arn:aws:identitystore::123456789012:identitystore/
d-9a670c546e"
        },
        "credentialId": "b41f0a02-1635-4d07-a414-aecf9e14b906"
    },
    "eventTime": "2025-08-07T14:09:07Z",
    "eventSource": "signin.amazonaws.com",
    "eventName": "UserAuthentication",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "192.0.2.14",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "AuthWorkflowID": "7a4ef12c-7c4b-4bc3-b5bd-c2469afcc795",
        "LoginTo": "https://example.awsapps.com/start/",
        "CredentialType": "PASSWORD"
    },
    "requestID": "fc91bcf0-ac53-4454-a1a0-fb911eacc095",
    "eventID": "18522007-1e60-4a71-b2b5-150baf504ab3",
    "readOnly": false,
    "eventType": "AWSServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "UserAuthentication": "Success"
    },
    "eventCategory": "Management"
}
```


In this example, note the following important fields:

- `eventSource`: Shows "signin.amazonaws.com", indicating this is an IAM Identity Center authentication event.
- `userIdentity.onBehalfOf`: Contains the user ID and identity store ARN for the web app user.
- `additionalEventData.LoginTo`: Shows the IAM Identity Center application URL being accessed.
- `additionalEventData.CredentialType`: Indicates the authentication method used (PASSWORD).
- `serviceEventDetails`: Shows the authentication result (Success).

Example log entry for ListCallerAccessGrants

The following example shows a CloudTrail log entry for a ListCallerAccessGrants event, which occurs when Transfer Family web app queries available access grants for a user.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAEXAMPLEID:aws-transfer",
        "arn": "arn:aws:sts::123456789012:assumed-role/AWSTransferWebAppIdentityBearer-
us-east-2/aws-transfer",
        "accountId": "123456789012",
        "accessKeyId": "ASIAEXAMPLEKEY",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAEXAMPLEID",
                "arn": "arn:aws:iam::123456789012:role/service-role/
AWSTransferWebAppIdentityBearer-us-east-2",
                "accountId": "123456789012",
                "userName": "AWSTransferWebAppIdentityBearer-us-east-2"
            },
            "attributes": {
                "creationDate": "2025-08-08T15:29:34Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "transfer.amazonaws.com",
        "onBehalfOf": {
            "userId": "f12bb510-a011-702f-10dd-5607e2776dbc",
            "identityStoreArn": "arn:aws:identitystore::123456789012:identitystore/
d-9a670c546e"
        }
    },
    "eventTime": "2025-08-08T15:29:35Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "ListCallerAccessGrants",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "transfer.amazonaws.com",
    "userAgent": "transfer.amazonaws.com",
    "requestParameters": {
        "Host": "123456789012.s3-control.dualstack.us-east-2.amazonaws.com",
```

```
        "allowedByApplication": "true",
        "maxResults": "100"
    },
    "responseElements": null,
    "additionalEventData": {
        "SignatureVersion": "SigV4",
        "CipherSuite": "TLS_AES_128_GCM_SHA256",
        "bytesTransferredIn": 0,
        "AuthenticationMethod": "AuthHeader",
        "x-amz-id-2": "1g34AaAELn/
fntxwrifVsr41VDl8dp5ygWFasHJFNVq5FDCWYfX0ye7s4tWHEJC8ppI5lLePYLIcw3iTXAgn5Q==",
        "bytesTransferredOut": 462
    },
    "requestID": "48485MTZEDWT0ANT",
    "eventID": "3de5dd60-b7cf-474c-a1ab-631467c1a5c3",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS:S3::AccessGrantsInstance",
            "ARN": "arn:aws:s3:us-east-2:123456789012:access-grants/default"
        }
    ],
    "eventType": "AWSApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


In this example, note the following important fields:

- `eventName`: Shows this is a ListCallerAccessGrants event, which queries available S3 access grants.
- `requestParameters.allowedByApplication`: Indicates the query is filtered to grants allowed by the application.
- `requestParameters.maxResults`: Shows the maximum number of grants to return in the response.
- `userIdentity.onBehalfOf`: Links the request to the specific web app user.

This event helps track when Transfer Family web app queries what S3 resources a user has access to, providing visibility into access grant discovery operations.

Example log entry for GetDataAccess event

The following example shows a CloudTrail log entry for a GetDataAccess event, which occurs when Transfer Family web app requests access permissions for S3 resources on behalf of a user.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROASEQRAEABP7ADWEZA5:aws-transfer",
        "arn": "arn:aws:sts::123456789012:assumed-role/AWSTransferWebAppIdentityBearer-
ap-southeast-1/aws-transfer",
        "accountId": "123456789012",
        "accessKeyId": "ASIAEXAMPLEKEY",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROASEQRAEABP7ADWEZA5",
                "arn": "arn:aws:iam::123456789012:role/service-role/
AWSTransferWebAppIdentityBearer-ap-southeast-1",
                "accountId": "123456789012",
                "userName": "AWSTransferWebAppIdentityBearer-ap-southeast-1"
            },
            "attributes": {
                "creationDate": "2025-05-08T16:09:05Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "transfer.amazonaws.com",
        "onBehalfOf": {
            "identityStoreArn": "arn:aws:identitystore::123456789012:identitystore/
d-9667b0da7a",
            "userId": "191a35ec-10a1-70c1-e4ab-e2802411e13e"
        }
    },
    "eventTime": "2025-05-08T16:10:25Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "GetDataAccess",
    "awsRegion": "ap-southeast-1",
    "sourceIPAddress": "transfer.amazonaws.com",
```

```
    "userAgent": "transfer.amazonaws.com",
    "requestParameters": {
        "Host": "123456789012.s3-control.dualstack.ap-southeast-1.amazonaws.com",
        "durationSeconds": 900,
        "permission": "READWRITE",
        "target": "s3://amzn-s3-demo-bucket/users/john.doe/documents/*"
    },
    "responseElements": null,
    "additionalEventData": {
        "AuthenticationMethod": "AuthHeader",
        "CipherSuite": "TLS_AES_128_GCM_SHA256",
        "SignatureVersion": "SigV4",
        "bytesTransferredIn": 0,
        "bytesTransferredOut": 2244,
        "x-amz-id-2": "8ce8sZOgNwsaj9w1mzagyA
+csONjYl8FgEw4FGpE8DARi90aNC0RFWlTYNEn7ChqE9RCJrTzMvS+ru7Vz2xXHrkQt/1uQ9exZTZdlhX+/fM="
    },
    "requestID": "BXGSKKQXCWS5RAHB",
    "eventID": "c11db1d1-dfb8-431e-8625-48eba2ebadfe",
    "readOnly": true,
    "resources": [
        {
            "type": "AWS:S3::AccessGrantsInstance",
            "ARN": "arn:aws:s3:ap-southeast-1:123456789012:access-grants/default",
            "accountId": "123456789012"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


In this example, note the following important fields:

- `eventName`: Shows this is a GetDataAccess event, which occurs when Transfer Family requests access permissions for S3 resources.
- `userIdentity.onBehalfOf`: Contains the identity store ARN and user ID, linking the access request to the specific web app user.
- `requestParameters.target`: Shows the S3 path pattern for which access was requested.

- `requestParameters.permission`: Indicates the type of access requested (READWRITE, READ, or WRITE).
- `requestParameters.durationSeconds`: Shows how long the access grant is valid (typically 900 seconds/15 minutes).
- `sourceIPAddress` and `userAgent`: Both show "transfer.amazonaws.com", indicating this is an internal service request.

GetDataAccess events are particularly useful for tracking when Transfer Family web app users are granted access to specific S3 resources, helping you monitor access patterns and ensure proper authorization.

Viewing CloudTrail log entries

There are several ways to view and analyze CloudTrail log entries for your Transfer Family web app:

Using the CloudTrail console

The CloudTrail console provides a user-friendly interface for viewing and filtering log entries:

1. Open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. In the navigation pane, choose **Event history**.
3. Use the filter options to narrow down the events:
   - Set **Event source** to `transfer.amazonaws.com` to view only Transfer Family events.
   - Filter by **Event name** to see specific operations like `UserAuthentication`.
   - Use **Time range** to focus on events within a specific period.
4. Click on any event to view its detailed information.

Accessing logs in Amazon S3

If you've configured a CloudTrail trail to deliver logs to an Amazon S3 bucket, you can access the raw log files directly:

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Navigate to the bucket and prefix where your CloudTrail logs are stored.
3. The logs are organized by year, month, day, and region. Navigate to the appropriate directory.

4. Download and open the log files, which are in JSON format.

Troubleshooting your web apps

> **Note**
>
> These troubleshooting tips are meant for the web app administrator rather than the end user. For end users, if you encounter any problems, contact your web app administrator. All instances of *you* in the following paragraphs refer to the web app admin.

Troubleshoot network errors

Description

Your end user sees a network banner **Network Error** upon loading the web app endpoint.

Cause

The most common issues are as follows:

- The admin did not assign the user that is attempting to log on to the new application.
- The admin did not add the necessary actions to your IAM roles.
- You see a list of S3 Access Grants assigned to your user, but CORS is not configured correctly for your Amazon S3 bucket or buckets.

Solution

- In IAM Identity Center, make sure to assign the user to the correct application. Or, if you have a group assigned, make sure that the user attempting to log in belongs to the correct group. This is described in Assign or add users or groups to a Transfer Family web app.
- Check whether your roles contain the necessary actions in the **Custom trust policy** for both `sts:AssumeRole` and `sts:SetContext` actions. This is described in Configure IAM roles for Transfer Family web apps.
- Check the CORS policy for all of the buckets used by your web app. This is described in Set up Cross-origin resource sharing (CORS) for your Amazon S3 bucket.

Troubleshoot configured bucket not appearing

Description

Everything appears to be configured correctly, but the Amazon S3 bucket doesn't appear in the web app.

Cause

One possible cause is that the Amazon S3 bucket is not in the same account as the web app.

Solution

Ensure that the Amazon S3 bucket is in the same account as the web app. Cross-account buckets are not currently supported.

Troubleshoot custom URL errors

Description

When your end user signs into the web app, they receive the error message **Authorization failed: missing authorization code.**

Cause

If you used CloudFront directly, rather than the supplied CloudFormation template, you have likely misconfigured the origin request policy to not forward query strings.

Solution

Update your origin request policy to forward query strings and cookies to the origin.

Description

When your end user attempts to access a Transfer Family web app, they receive a 404 response.

Cause

If you used CloudFront directly, rather than the supplied CloudFormation template, you have likely misconfigured the cache policy to include the `Host` header in the cache key or misconfigured the origin request policy to forward the `Host` header.

Solution

- Make sure that your cache policy does not include the `Host` header in the cache key
- Make sure that your origin request policy does not forward the `Host` header.

Troubleshoot miscellaneous errors

Description

Your end user cannot log in, or cannot view any buckets or files, or you receive another error.

Cause

One possible cause is that the IAM Identity Center instance ARN doesn't match the value for your grants ARN or your web app IAM Identity Center instance ARN.

Solution

Check the following items to see if they match.

- In IAM Identity Center, navigate to **Settings** and view the **Instance ARN**.

```
arn:aws:sso:::instance/ssoins-instance-identifier
```

- In Amazon S3, navigate to **Access Grants** and view your **IAM Identity Center instance ARN**.

```
arn:aws:sso::account-id:application/ssoins-instance-identifier/apl-1234567890abcdef0
```

- In Transfer Family, navigate to your web app details page and view its Instance ARN.

```
arn:aws:sso:::instance/ssoins-instance-identifier
```

The *instance-identifier* value must be the same in all three of these places.

Duplicate S3 buckets appearing in web app

Description

Users see the same S3 bucket listed multiple times in the Transfer Family web app interface.

Cause

This occurs when a user is part of multiple Active Directory groups that have duplicate grants to the same S3 bucket. The web app lists all top-level grants associated with the user (UID or GID) regardless of whether the user has multiple grants assigned to the same bucket location.

Solution

To resolve this issue, administrators should de-duplicate the grants so that each user has only one grant to each S3 location. Review your S3 Access Grants configuration and consolidate duplicate grants for the same bucket across different Active Directory groups.

End user instructions for Transfer Family web apps

> **Note**
>
> In this topic, the information is meant for the end users that are interacting with the web app. All instances of *you* in this topic refer to end users.

This topic describes how to access an AWS Transfer Family web app that you are authorized to use, and describes how you can interact with it.

Web app quotas

Note the following limitations when using web apps.

- Maximum number of search results per query: 10,000
- The Amazon S3 buckets that are used by the Transfer Family web app must be in the same account as the web app itself. Cross-account buckets are not currently supported.
- Maximum search breadth per query: 10,000 searched files
- Maximum upload size per file: 160 GB (149 GiB)
- Maximum size file for copying: 5.36 GB (5 GiB)
- Folder names starting or ending with dots (.) are not supported

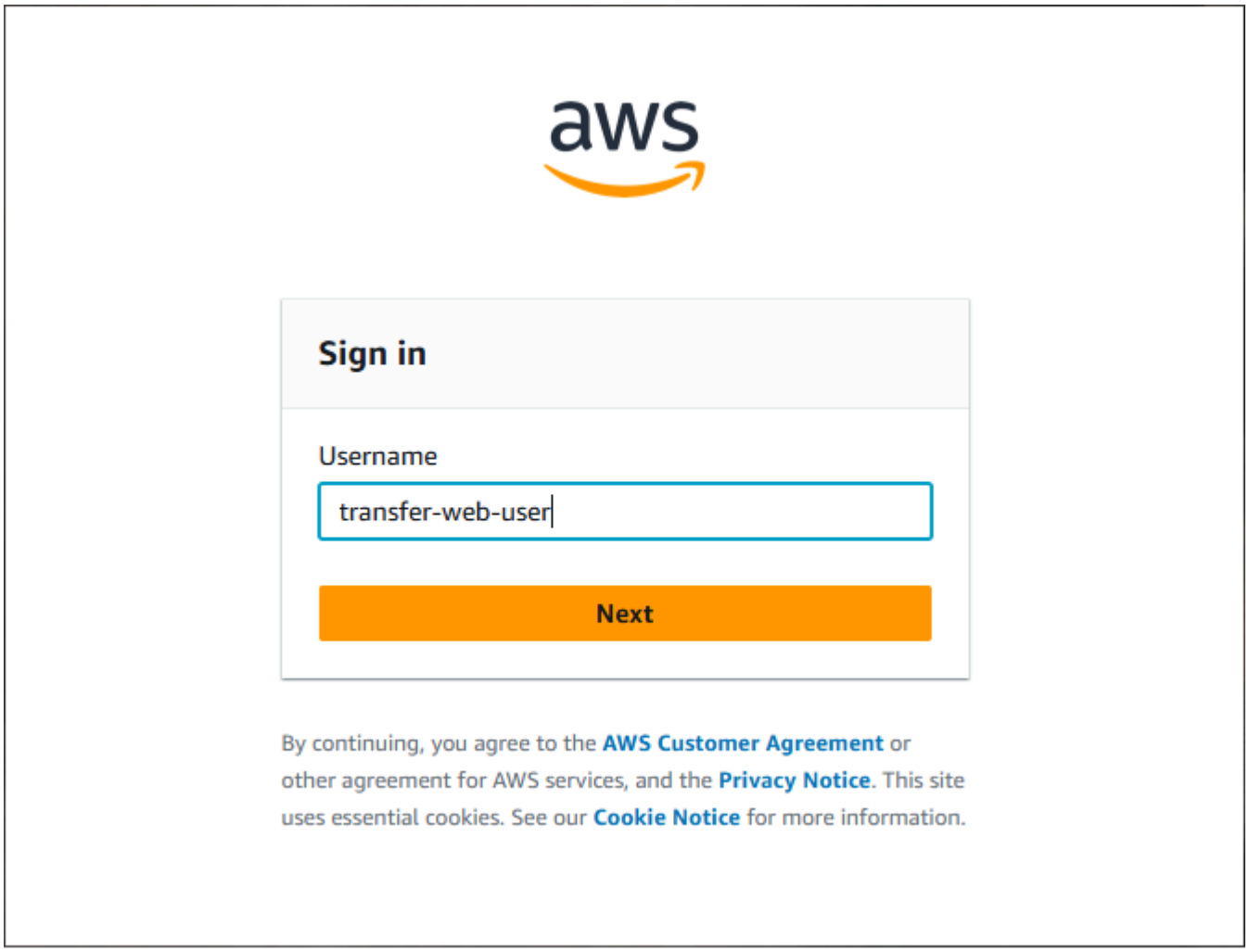
User experience for IAM Identity Center users

This section describes the user experience if your organization used IAM Identity Center to configure its users.

To access a Transfer Family web app

1. You should receive an email from **no-reply@login.awsapps.com** titled “Invitation to join AWS IAM Identity Center.” Accept the invitation to activate your user account.
2. In the message, choose the URL below **Your AWS access portal URL**.

   This takes you to the AWS sign in screen.



3. Enter your credentials and choose **Sign in**.

   This takes you to the AWS access portal, which shows a list of your available applications.
4. Choose the application for your Transfer Family web app.

User experience for third-party identity provider users

If your organization did not use AWS IAM Identity Center to configure its users, your onboarding experience will depend upon the identity provider application that they used to configure their end users. After you authenticate and sign in, your web app interface is the same as that described in the next section.

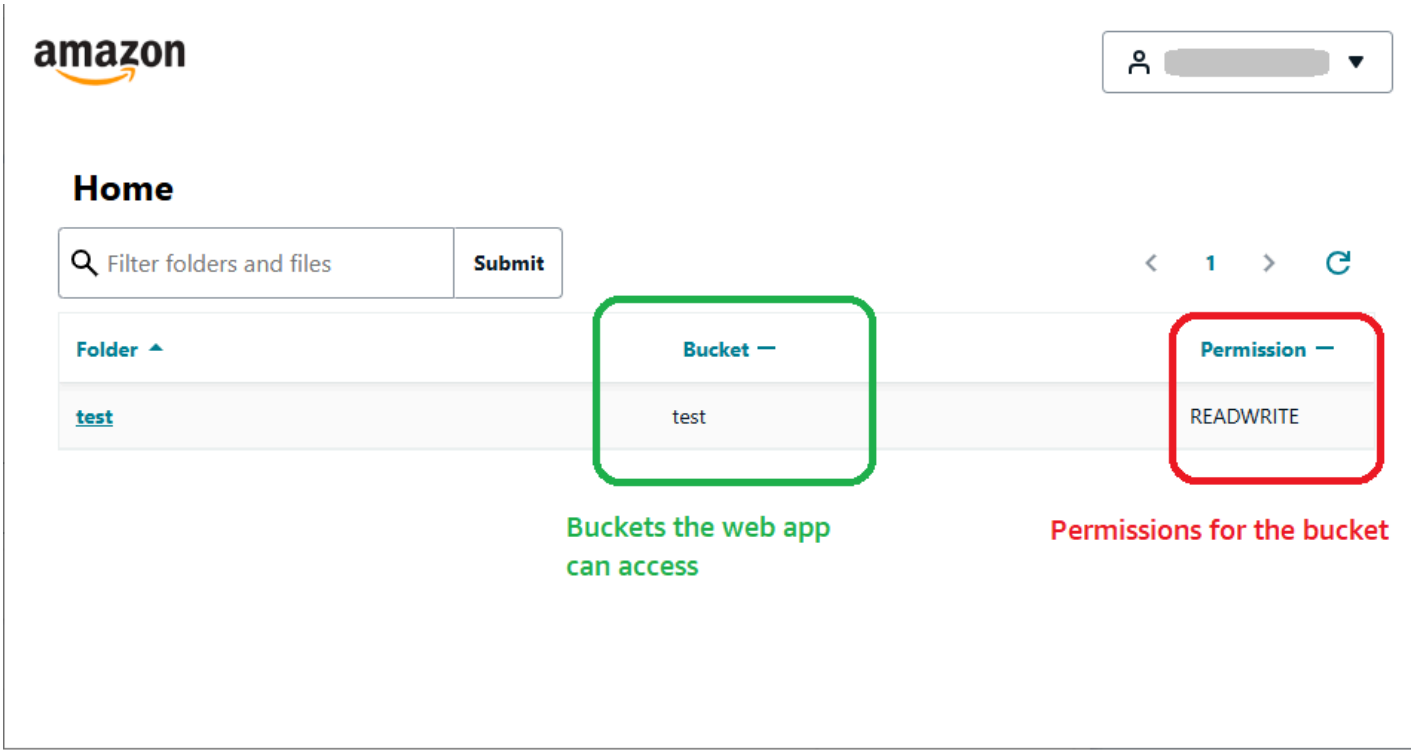
> **Note**
>
> If a user belongs to multiple Active Directory groups that have grants to the same Amazon S3 bucket, the bucket appears multiple times in the web app interface. This occurs because the web app lists all top-level grants associated with the user's UID or GID, including duplicate grants to the same bucket. To prevent duplicate listings, administrators can consolidate multiple grants so that each user has only one grant per Amazon S3 location.

Transfer Family end user interface

After you have authenticated and signed in, you can interact with the web app.

There are four main views.

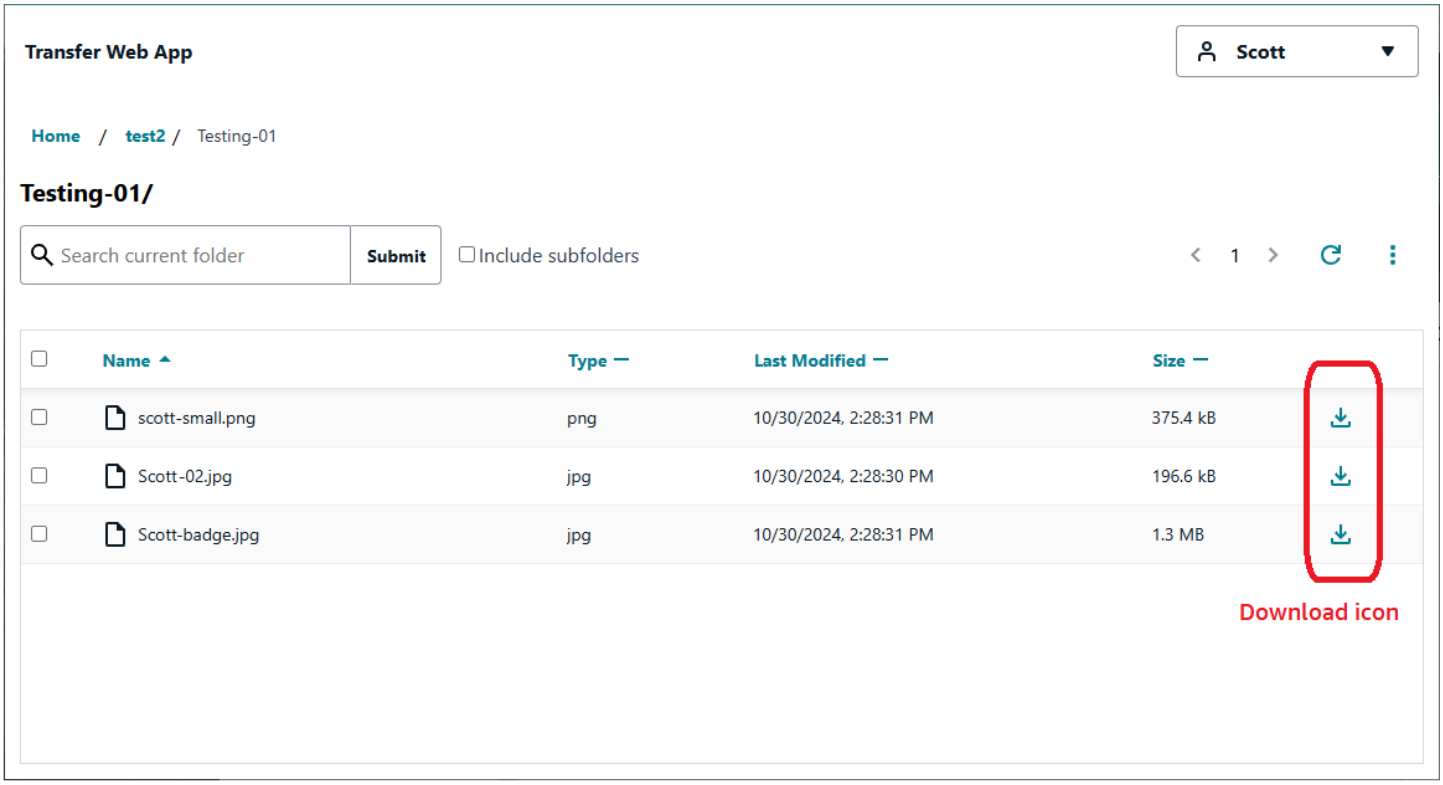
- **Home page:** Your home page lists the S3 locations, which you can access, as well as the permissions for each. An S3 *location* is an S3 bucket or prefix, which you can define when using S3 Access Grants. This is the initial view for users that shows the root level S3 resources that your end users have access to and the permissions for each S3 location.



- **Location details:** This view allows users to browse files and folders in S3, and upload or download files.
- **Location action:** After you choose an action (such as **Upload**), it opens up another view of the file location.
- **Vertical ellipsis:** The vertical ellipsis icon opens the **Actions** menu.

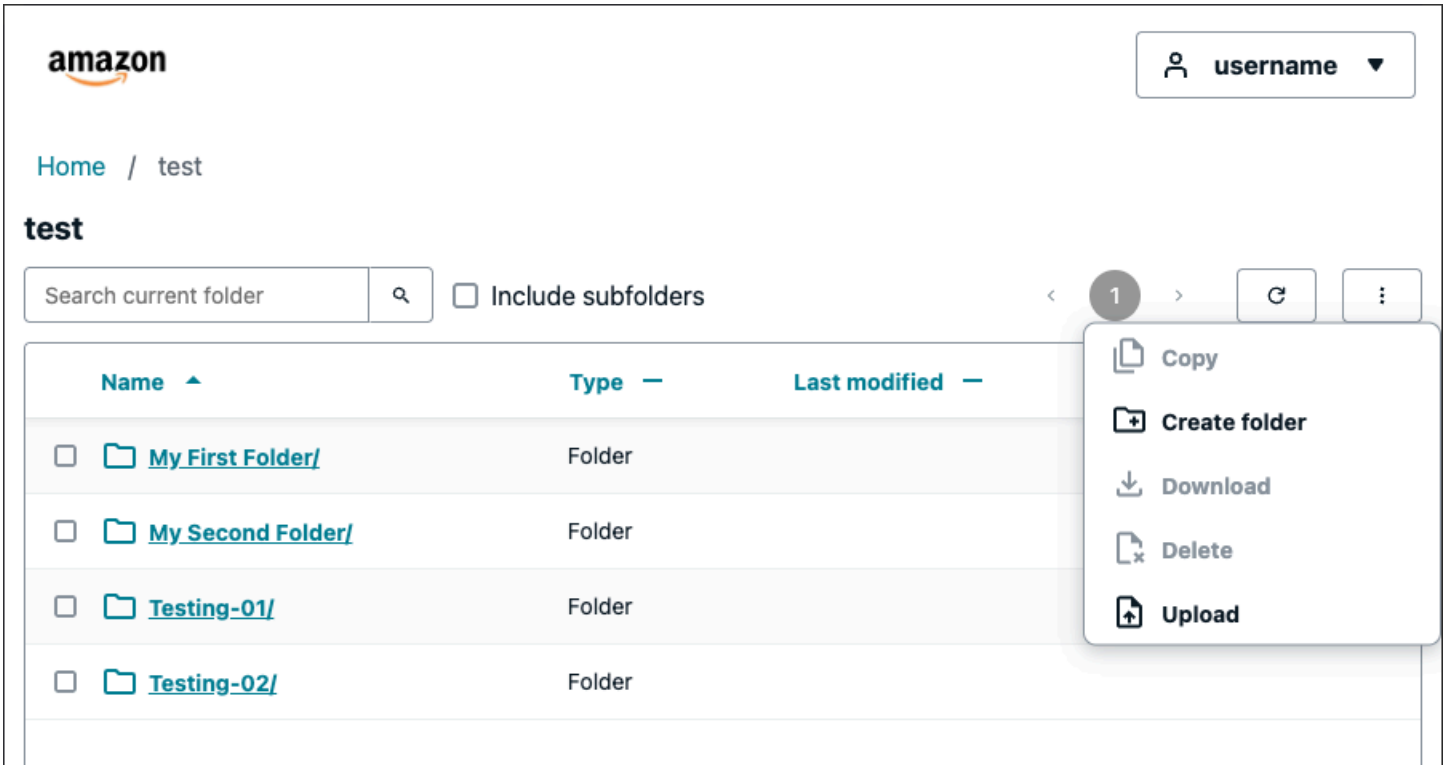
Available actions

Most of the actions are available from the **Actions** menu. To download a single file, you can also use the download icon that appears next to the filename.



From a folder, use the **Actions** menu to perform any of the following tasks:

- Copy one or more files to another location.
- Create a folder.
- Download one or more files and folders. A single file downloads as-is; selecting multiple items, or any folder, packages your selection into a single `.zip` file that preserves the original folder structure.
- Delete one or more files.
- Upload one or more files or folders, including any subfolders.
- Select a folder and navigate to it. You can then perform any of the previously listed actions.
- Sort by page.
- Filter by file or folder name per folder and subfolders.


amazon
username
Home / test
test
Search current folder
Include subfolders
1
Copy
Create folder
Download
Delete
Upload
Name
Type
Last modified
My First Folder/
Folder
My Second Folder/
Folder
Testing-01/
Folder
Testing-02/
Folder

AWS Transfer Family SFTP connectors

An AWS Transfer Family SFTP connector establishes a connection with a remote SFTP server to transfer files between Amazon storage and a remote server, using the SFTP protocol. You can send files from Amazon S3 to an external, partner-owned SFTP server, retrieve files from a partner's SFTP server to Amazon S3 or list, delete, rename or move files on the remote server. SFTP connectors support two egress types: service managed (using AWS managed infrastructure) and VPC (routing through your VPC using Amazon VPC Lattice). Using SFTP connectors, you can build automated, event-driven file transfer workflows in AWS .

View AWS Transfer Family SFTP connectors for a brief introduction to Transfer Family SFTP connectors.

Topics

- Creating SFTP connectors
- VPC connectivity for SFTP connectors
- Using SFTP connectors
- Monitoring SFTP connectors
- Managing SFTP connectors
- Scaling and quotas for SFTP connectors
- Reference architectures using SFTP connectors

Creating SFTP connectors

This topic describes how to create SFTP connectors. Each connector provides the ability to connect with one remote SFTP server. You perform the following high-level tasks to configure an SFTP connector.

> **Note**
>
> For VPC-based connectors that route traffic through your Virtual Private Cloud, see Create an SFTP connector with VPC-based egress.

1. Store the authentication credentials for the connector in AWS Secrets Manager.

2. Create the connector, by specifying the secret ARN, the remote server's URL or Resource Configuration ARN, the security policy containing the algorithms that will be supported by the connector, and other configuration settings.
3. After you create the connector, you can test it to ensure that it can establish connections with the remote SFTP server.

Choosing SFTP connector egress type

When you create a SFTP connector, you choose the Egress Type between "Service managed" and "VPC Lattice".

- **Service managed** (default): The connector uses NAT gateways and IP addresses owned by AWS Transfer Family to route connections over the public internet. The service provides 3 static IP addresses for your connectors that need to be allowlisted on the remote servers to establish connections.
- **VPC Lattice**: The connector routes traffic through your VPC environment using Amazon VPC Lattice. Use VPC connectivity for SFTP connectors in these scenarios:
  - **Private SFTP servers**: Connect to SFTP servers that are only accessible from your VPC
  - **On-premises connectivity**: Connect to on-premises SFTP servers through AWS Direct Connect or AWS Virtual Private Network connections
  - **Custom IP addresses**: Present your own NAT gateways and Elastic IP addresses to the remote server
  - **Centralized security controls**: Route file transfers through your organization's central ingress/egress controls

The following matrix helps you choose the right connector type for your use-cases.

SFTP Connector Egress Type matrix

Capability	Egress Type = Service managed	Egress Type = VPC Lattice
Connectivity to Publicly hosted (internet-accessible) SFTP servers	Supported	Supported[1]

Capability	Egress Type = Service managed	Egress Type = VPC Lattice
Connectivity to Privately hosted (on-premises) SFTP servers	Not supported	Supported[2]
Connectivity to Privately hosted (in-VPC) SFTP servers	Not supported	Supported
Static IP addresses presented to remote SFTP server	Supported via service supplied static IP addresses	Supported via customer owned static IP addresses
Bandwidth available	50 MBPS per account	Higher bandwidth, as available from customer owned Resource Gateway and NAT Gateway
Traffic routing to internet over customer-owned NAT Gateways and Network Firewalls	Not supported. NAT Gateways are owned and managed by Transfer Family service.	Supported

[1] *With Egress Type = VPC Lattice, connectivity to publicly hosted servers is supported using the egress infrastructure (NAT Gateways) setup in your egress VPCs.*

[2] *With Egress Type = VPC Lattice, connectivity to privately hosted servers is supported using existing networks in your VPC, such as AWS Direct Connect or VPN.*

Choosing IP addressing mode

When you create an SFTP connector with service-managed egress, you can choose between two IP addressing modes:

- **IPv4 only** (default): The connector uses IPv4 addresses exclusively to connect to the remote SFTP server. This is the default mode when creating connectors through the console, AWS CLI, or API.

- **Dual-stack**: The connector supports both IPv6 and IPv4 addresses. In dual-stack mode, the connector prefers IPv6 when DNS resolution returns IPv6 results, and uses IPv4 when only IPv4 DNS results are returned.

> **Note**
>
> IP addressing mode applies only to connectors with service-managed egress type. Connectors that use VPC Lattice egress do not support this setting.

Topics

- Store authentication credentials for SFTP connectors in Secrets Manager
- Create an SFTP connector with service-managed egress
- Create an SFTP connector with VPC-based egress
- Test an SFTP connector

Store authentication credentials for SFTP connectors in Secrets Manager

You can use Secrets Manager to store user credentials for your SFTP connectors. When you create your secret, you must provide a username. Additionally, you can provide either a password, a private key, or both. For details, see Quotas for SFTP connectors.

> **Note**
>
> When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see AWS Secrets Manager Pricing.

To store user credentials in Secrets Manager for an SFTP connector

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at https://console.aws.amazon.com/secretsmanager/.
2. In the left navigation pane, choose **Secrets**.
3. On the **Secrets** page, choose **Store a new secret**.

4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
5. Provide the key/value information for your secret: you need to provide the username, and either a private key or a password.

   a. In the **Key/value pairs** section, choose the **Key/value** tab.

   - **Key** – Enter **`Username`**.
   - **value** – Enter the name of the user that is authorized to connect to the partner's server.

   b. If you want to provide a key pair, choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

   - **Key** – Enter **`PrivateKey`**.
   - **value** – paste in your private key.

   Tip: The private key data that you enter must correspond to the public key that is stored for this user on the remote SFTP server.

   > **Note**
   >
   > It is not possible to use a passphrase-protected private key for authentication with an AWS Transfer Family SFTP connector.

   For details on how to generate a public/private key pair, see Creating SSH keys on macOS, Linux, or Unix.

   c. If you want to provide a password, choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

   - **Key** – Enter **`Password`**.
   - **value** – Enter the password for the user.

6. Choose **Next**.
7. On the **Configure secret** page, enter a name and description for your secret. We recommend that you use a prefix of **`aws/transfer/`** for the name. For example, you could name your secret **`aws/transfer/connector-1`**.
8. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
9. On the **Review** page, choose **Store** to create and store the secret.

Secret version fallback

When you rotate credentials for your SFTP connector, external partners might take time to update to the new credentials. During this transition, the connector might fail to authenticate because it only attempts the current version of the secret by default.

To support graceful rotation without downtime, you can configure an ordered list of version stages on your connector. The connector tries each version stage in sequence during authentication—it uses the first stage that authenticates successfully. For example, if you configure `["AWSCURRENT", "AWSPREVIOUS"]`, the connector tries the new credentials first. If authentication fails, the connector falls back to the previous credentials.

You configure the version stage list once on your connector—subsequent rotations require no manual intervention.

> **Note**
>
> When rotating secrets, you must keep the `Username` value the same across all version stages. The connector does not support different usernames for different version stages. If you need to change the username, update `UserSecretId` to point to a new secret rather than rotating the username within the same secret.

For information about configuring version stage fallback on your connector, see Managing SFTP connectors.

For more information about Secrets Manager version stages, see Version stages in the *AWS Secrets Manager User Guide*.

Create an SFTP connector with service-managed egress

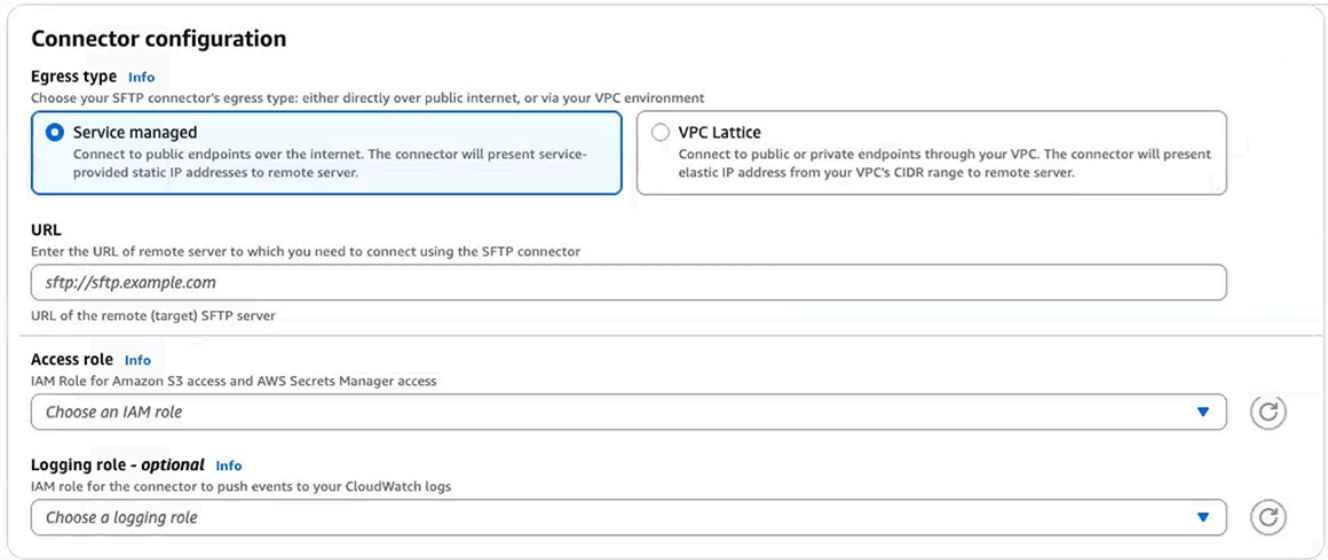
This procedure explains how to create SFTP connectors by using the AWS Transfer Family console or AWS CLI.

Console

To create an SFTP connector

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.

2. In the left navigation pane, choose **SFTP Connectors**, then choose **Create SFTP connector**.
3. In the **Connector configuration** section, for **Egress type**, choose **Service managed**. This option uses AWS Transfer Family managed egress infrastructure. The Transfer Family service provides and manages static IP addresses for each SFTP connector.
4. In the **Connector configuration** section, provide the following information:



- For the **URL**, enter the URL for a remote SFTP server. This URL must be formatted as `sftp://`*`partner-SFTP-server-url`*, for example `sftp://AnyCompany.com`.

  > **Note**
  >
  > Optionally, you can provide a port number in your URL. The format is `sftp://`*`partner-SFTP-server-url`*`:`*`port-number`*. The default port number (when no port is specified) is port 22.

- For the **Access role**, choose the Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role to use.
  - **Make sure that this role provides read and write access** to the parent directory of the file location that's used in the `StartFileTransfer` request.
  - **Make sure that this role provides permission** for `secretsmanager:GetSecretValue` to access the secret.

> **Note**
>
> In the policy, you must specify the ARN for the secret. The ARN contains the secret name, but appends the name with six, random, alphanumeric characters. An ARN for a secret has the following format.
>
> ```
> arn:aws:secretsmanager:region:account-id:secret:aws/
> transfer/SecretName-6RandomCharacters
> ```

- **Make sure this role contains a trust relationship** that allows the connector to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see To establish a trust relationship.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Sid": "AllowListingOfUserFolder",
        "Action": [
            "s3:ListBucket",
            "s3:GetBucketLocation"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ]
    },
    {
        "Sid": "HomeDirObjectAccess",
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:GetObjectVersion",
            "s3:GetObjectACL",
            "s3:PutObjectACL"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
    },
    {
        "Sid": "GetConnectorSecretValue",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue"
        ],
        "Resource": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:aws/transfer/SecretName-6RandomCharacters"
    }
  ]
}
```

> **Note**
>
> For the access role, the example grants access to a single secret. However, you can use a wildcard character, which can save work if you want to reuse the same IAM role for multiple users and secrets. For example, the following resource statement grants permissions for all secrets that have names beginning with `aws/transfer`.
>
> ```
> "Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/
> transfer/*"
> ```
>
> You can also store secrets containing your SFTP credentials in another AWS account. For details on enabling cross-account secret access, see Permissions to AWS Secrets Manager secrets for users in a different account.

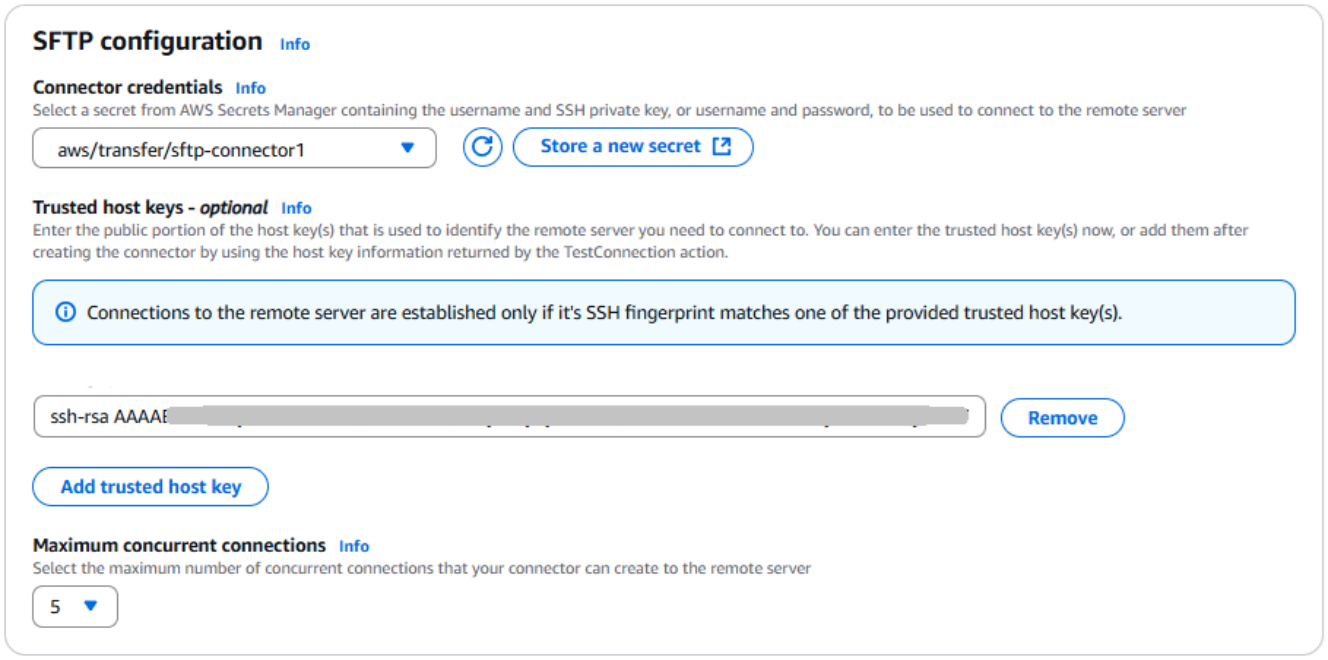
5. Complete the connector configuration:

   - (Optional) For the **Logging role**, choose the IAM role for the connector to use to push events to your CloudWatch logs. The following example policy lists the necessary permissions to log events for SFTP connectors.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/transfer/*"
        }
    ]
}
```

6. In the **SFTP Configuration** section, provide the following information:



- For **Connector credentials**, from the dropdown list, choose the name of a secret in AWS Secrets Manager that contains the SFTP user's private key or password. You must create a secret and store it in a specific manner. For details, see Store authentication credentials for SFTP connectors in Secrets Manager.
- (Optional) You have an option to create your connector while leaving the `TrustedHostKeys` parameter empty. However, your connector will not be able to transfer files with the remote server until you provide this parameter in your connector's configuration. You can enter the Trusted host key(s) at the time of creating your connector, or update your connector later by using the host key information returned by the `TestConnection` console action or API command. That is, for the **Trusted host keys** text box, you can do either of the following:

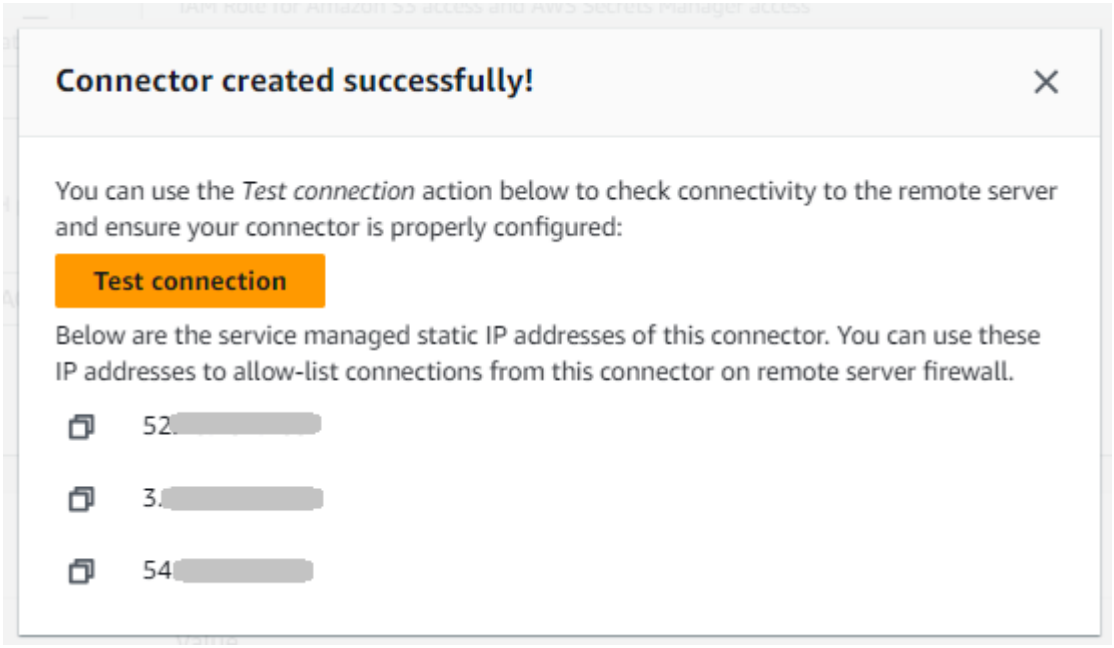
- **Provide the Trusted Host Key(s) at the time of creating your connector.** Paste in the public portion of the host key that is used to identify the external server. You can add more than one key, by choosing **Add trusted host key** to add an additional key. You can use the `ssh-keyscan` command against the SFTP server to retrieve the necessary key. For details about the format and type of trusted host keys that Transfer Family supports, see SFTPConnectorConfig.
- *Leave the Trusted Host Key(s) text box empty when creating your connector and update your connector at a later time with this information.* If you do not have the host key information at the time of creating your connector, you can leave this parameter empty for now and proceed with creating your connector. After the connector is created, use the new connector's ID to run the `TestConnection` command, either in the AWS CLI or from the connector's detail page. If successful, `TestConnection` will return the necessary host key information. You can then edit your connector using the console (or by running the `UpdateConnector` AWS CLI command) and add the host key information that was returned when you ran `TestConnection`.

  > **Important**
  >
  > If you retrieve the remote server's host key by running `TestConnection`, make sure that you perform out-of-band validation on the key that is returned.
  > You must accept the new key as trusted, or verify the presented fingerprint with a previously known fingerprint that you have received from the owner of the remote SFTP server you are connecting to.

- (Optional) For **Maximum concurrent connections**, from the dropdown list, choose the number of concurrent connections that your connector creates to the remote server. The default selection on the console is **5**.

  This setting specifies the number of active connections that your connector can establish with the remote server at the same time. Creating concurrent connections can enhance connector performance by enabling parallel operations.
- (Optional) For **Ordered secret version stages**, enter an ordered list of Secrets Manager version stage labels. The connector attempts authentication using each stage in order during file transfers. If you don't specify version stages, the connector uses only `AWSCURRENT`. For example, specify `AWSCURRENT,AWSPREVIOUS` to enable automatic fallback during secret rotation.

7. In the **Cryptographic algorithm options** section, choose a **Security policy** from the dropdown list in the **Security Policy** field. The security policy enables you to select the cryptographic algorithms that your connector supports. For details on the available security policies and algorithms, see Security policies for AWS Transfer Family SFTP connectors.
8. (Optional) In the **Tags** section, for **Key** and **Value**, enter one or more tags as key-value pairs.
9. After you have confirmed all of your settings, choose **Create SFTP connector** to create the SFTP connector. If the connector is created successfully, a screen appears with a list of the assigned static IP addresses and a **Test connection** button. Use the button to test the configuration for your new connector.



The **Connectors** page appears, with the ID of your new SFTP connector added to the list. To view the details for your connectors, see View SFTP connector details.

CLI

You use the create-connector command to create a connector. To use this command to create an SFTP connector, you must provide the following information.

- The URL for a remote SFTP server. This URL must be formatted as `sftp://`*`partner-SFTP-server-url`*, for example `sftp://AnyCompany.com`.
- The access role. Choose the Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role to use.

- **Make sure that this role provides read and write access** to the parent directory of the file location that's used in the `StartFileTransfer` request.
- **Make sure that this role provides permission** for `secretsmanager:GetSecretValue` to access the secret.

> **Note**
>
> In the policy, you must specify the ARN for the secret. The ARN contains the secret name, but appends the name with six, random, alphanumeric characters. An ARN for a secret has the following format.
>
> ```
> arn:aws:secretsmanager:region:account-id:secret:aws/
> transfer/SecretName-6RandomCharacters
> ```

- **Make sure this role contains a trust relationship** that allows the connector to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see To establish a trust relationship.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Sid": "AllowListingOfUserFolder",
        "Action": [
            "s3:ListBucket",
            "s3:GetBucketLocation"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ]
    },
    {
        "Sid": "HomeDirObjectAccess",
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
```

```
            "s3:GetObjectVersion",
            "s3:GetObjectACL",
            "s3:PutObjectACL"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
        "Sid": "GetConnectorSecretValue",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue"
        ],
        "Resource": "arn:aws:secretsmanager:us-west-2:111122223333:secret:aws/
transfer/SecretName-6RandomCharacters"
    }
  ]
}
```

Note

For the access role, the example grants access to a single secret. However, you can use a wildcard character, which can save work if you want to reuse the same IAM role for multiple users and secrets. For example, the following resource statement grants permissions for all secrets that have names beginning with `aws/transfer`.

```
"Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/transfer/
*"
```

You can also store secrets containing your SFTP credentials in another AWS account. For details on enabling cross-account secret access, see Permissions to AWS Secrets Manager secrets for users in a different account.

- (Optional) Choose the IAM role for the connector to use to push events to your CloudWatch logs. The following example policy lists the necessary permissions to log events for SFTP connectors.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/transfer/*"
        }
    ]
}
```

- Provide the following SFTP configuration information.
  - The ARN of a secret in AWS Secrets Manager that contains the SFTP user's private key or password.
  - The public portion of the host key that is used to identify the external server. You can provide multiple trusted host keys if you like.

  The easiest way to provide the SFTP information is to save it to a file. For example, copy the following example text to a file named `testSFTPConfig.json`.

```
// Listing for testSFTPConfig.json
{
   "UserSecretId": "arn:aws::secretsmanager:us-east-2:123456789012:secret:aws/
transfer/example-username-key",
   "TrustedHostKeys": [
      "sftp.example.com ssh-rsa AAAAbbbb...EEEE="
   ],
   "OrderedUserSecretVersionStages": ["AWSCURRENT", "AWSPREVIOUS"]
}
```

  > **Note**
  >
  > The `OrderedUserSecretVersionStages` parameter is optional. If you omit this parameter, the connector defaults to `["AWSCURRENT"]`.

- Specify a security policy for your connector, entering the security policy name.

> **Note**
>
> The `SecretId` can be either the entire ARN or the name of the secret (*example-username-key* in the previous listing).

Then run the following command to create the connector:

```
aws transfer create-connector --url "sftp://partner-SFTP-server-url" \
--access-role your-IAM-role-for-bucket-access \
--logging-role arn:aws:iam::your-account-id:role/service-role/
AWSTransferLoggingAccess \
--sftp-config file:///path/to/testSFTPConfig.json \
--security-policy-name security-policy-name \
--maximum-concurrent-connections integer-from-1-to-5
```

When you describe a VPC egress type connector, the response includes the new fields:

```
{
   "Connector": {
      "AccessRole": "arn:aws:iam::123456789012:role/connector-role",
      "Arn": "arn:aws:transfer:us-east-1:123456789012:connector/
c-1234567890abcdef0",
      "ConnectorId": "c-1234567890abcdef0",
      "Status": "ACTIVE",
      "EgressConfig": {
        "VpcLattice": {
          "ResourceConfigurationArn": "arn:aws:vpc-lattice:us-
east-1:123456789012:resourceconfiguration/rcfg-12345678",
          "PortNumber": 22
        }
      },
      "EgressType": "VPC",
      "ServiceManagedEgressIpAddresses": null,
      "SftpConfig": {
         "TrustedHostKeys": [ "ssh-rsa AAAAB3NzaC..." ],
         "UserSecretId": "aws/transfer/connector-secret"
      },
      "Url": "sftp://my.sftp.server.com:22"
   }
}
```

Note that `ServiceManagedEgressIpAddresses` is null for VPC egress type connectors since traffic routes through your VPC instead of AWS managed infrastructure.

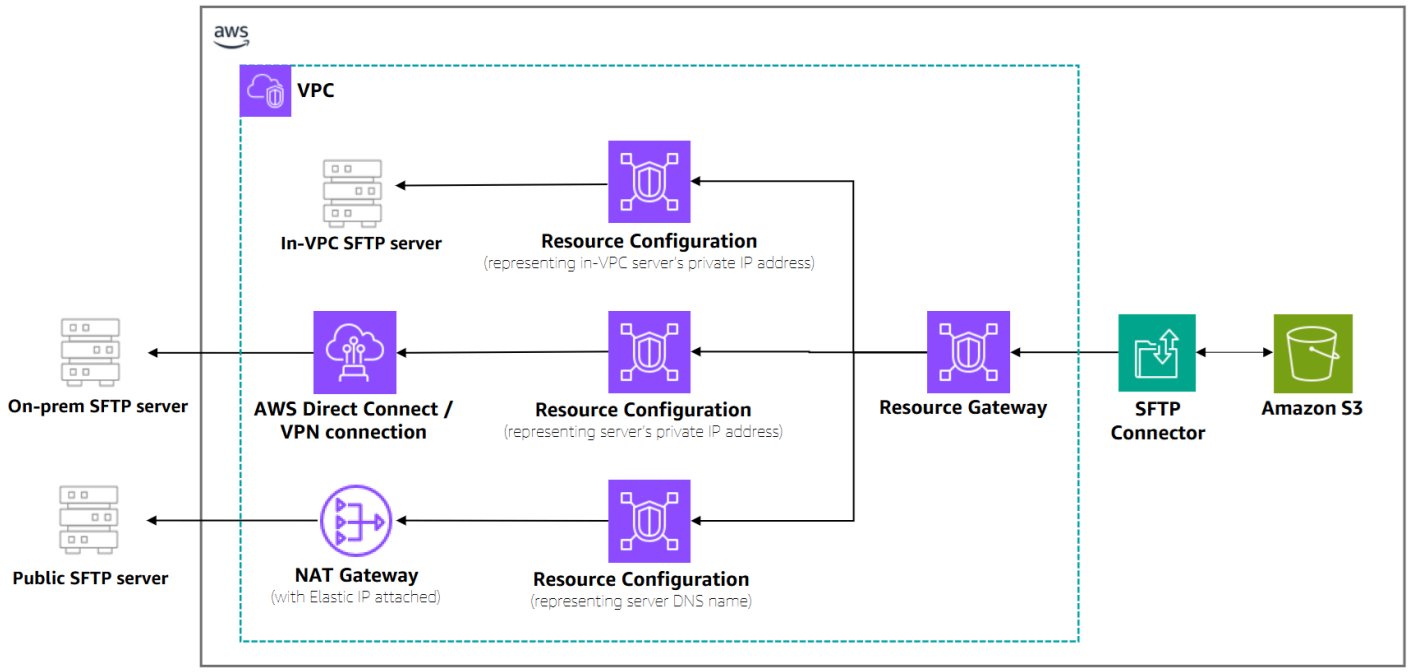
Create an SFTP connector with VPC-based egress

This topic provides step-by-step instructions for creating SFTP connectors with VPC connectivity. VPC_LATTICE-enabled connectors use Amazon VPC Lattice to route traffic through your Virtual Private Cloud, enabling secure connections to private endpoints or using your own NAT gateways for internet access.

When to use VPC connectivity

Use VPC connectivity for SFTP connectors in these scenarios:

- **Private SFTP servers**: Connect to SFTP servers that are only accessible from your VPC.
- **On-premises connectivity**: Connect to on-premises SFTP servers through AWS Direct Connect or AWS Site-to-Site VPN connections.
- **Custom IP addresses**: Use your own NAT gateways and Elastic IP addresses, including BYOIP scenarios.
- **Centralized security controls**: Route file transfers through your organization's central ingress/egress controls.



Prerequisites for VPC_LATTICE-enabled SFTP connectors

Before creating a VPC_LATTICE-enabled SFTP connector, you must complete the following prerequisites:

How VPC-based connectivity works

VPC Lattice enables you to securely share VPC resources with other AWS services. AWS Transfer Family uses a service network to simplify the resource sharing process. The key components are:

- **Resource Gateway**: Serves as the point of access into your VPC. You create this in your VPC with a minimum of two Availability Zones.
- **Resource Configuration**: Contains the private IP address or public DNS name of the SFTP server you want to connect to.

When you create a VPC_LATTICE-enabled connector, AWS Transfer Family uses Forward Access Session (FAS) to temporarily obtain your credentials and associate your Resource Configuration with our service network.

Required setup steps

1. **VPC infrastructure**: Ensure you have a properly configured VPC with the necessary subnets, route tables, and security groups for your SFTP server connectivity requirements.
2. **Resource Gateway**: Create a Resource Gateway in your VPC using the VPC Lattice `create-resource-gateway` command. The Resource Gateway must be associated with subnets in at least two Availability Zones. For more information, see Resource gateways in the *Amazon VPC Lattice User Guide*.
3. **Resource Configuration**: Create a Resource Configuration that represents the target SFTP server using the VPC Lattice `create-resource-configuration` command. You can specify either:
   - A private IP address for private endpoints
   - A public DNS name for public endpoints (IP addresses are not supported for public endpoints)
4. **Authentication credentials**: Store the SFTP user credentials in AWS Secrets Manager as described in Store authentication credentials for SFTP connectors in Secrets Manager.

Important

The Resource Gateway and Resource Configuration must be created in the same AWS account. When creating a Resource Configuration, you must first have a Resource Gateway in place.

For more information on VPC resource configurations, see Resource configurations in the *Amazon VPC Lattice User Guide*.

Note

VPC connectivity for SFTP connectors is available in AWS Regions where Amazon VPC Lattice resources are available. For more information, see VPC Lattice FAQs. Availability Zone support varies by region, and Resource Gateways require a minimum of two Availability Zones.

Create a VPC_LATTICE-enabled SFTP connector

After completing the prerequisites, you can create an SFTP connector with VPC connectivity using the AWS CLI, AWS Management Console, or AWS SDKs.

Console

To create a VPC_LATTICE-enabled SFTP connector

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **SFTP Connectors**, then choose **Create SFTP connector**.
3. In the **Connector configuration** section, for **Egress type**, choose **VPC Lattice**.

   This option routes traffic through your VPC using Amazon VPC Lattice for cross-VPC resource access. You can use this option to connect to privately hosted server endpoints, route traffic through your VPC's security controls, or use your own NAT gateways and Elastic IP addresses. The address of the remote SFTP server is represented as a Resource Configuration in your VPC. For more information about Resource Configurations, see Resource configurations for VPC resources in the Amazon VPC Lattice User Guide.
4. Complete the connector configuration:

- For the **Access role**, choose the Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role to use.
  - **Make sure that this role provides read and write access** to the parent directory of the file location that's used in the `StartFileTransfer` request.
  - **Make sure that this role provides permission** for `secretsmanager:GetSecretValue` to access the secret.

    > **Note**
    >
    > In the policy, you must specify the ARN for the secret. The ARN contains the secret name, but appends the name with six, random, alphanumeric characters. An ARN for a secret has the following format.
    >
    > ```
    > arn:aws:secretsmanager:region:account-id:secret:aws/
    > transfer/SecretName-6RandomCharacters
    > ```

  - **Make sure this role contains a trust relationship** that allows the connector to access your resources when servicing your users' transfer requests. For details on establishing a trust relationship, see To establish a trust relationship.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Sid": "AllowListingOfUserFolder",
        "Action": [
            "s3:ListBucket",
            "s3:GetBucketLocation"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ]
    },
    {
        "Sid": "HomeDirObjectAccess",
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
```

```
            "s3:GetObject",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:GetObjectVersion",
            "s3:GetObjectACL",
            "s3:PutObjectACL"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
        "Sid": "GetConnectorSecretValue",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue"
        ],
        "Resource": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:aws/transfer/SecretName-6RandomCharacters"
    }
  ]
}
```

Note

For the access role, the example grants access to a single secret. However, you can use a wildcard character, which can save work if you want to reuse the same IAM role for multiple users and secrets. For example, the following resource statement grants permissions for all secrets that have names beginning with `aws/transfer`.

```
"Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/
transfer/*"
```

You can also store secrets containing your SFTP credentials in another AWS account. For details on enabling cross-account secret access, see Permissions to AWS Secrets Manager secrets for users in a different account.

- For **Resource Configuration ARN**, enter the ARN of the VPC Lattice Resource Configuration that points to your SFTP server:

```
arn:aws:vpc-lattice:region:account-id:resourceconfiguration/rcfg-12345678
```

- (Optional) For the **Logging role**, choose the IAM role for the connector to use to push events to your CloudWatch logs.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/transfer/*"
        }
    ]
}
```

5. In the **SFTP Configuration** section, provide the following information:

   - For **Connector credentials**, choose the name of a secret in AWS Secrets Manager that contains the SFTP user's private key or password.
   - For **Trusted host keys**, paste in the public portion of the host key that is used to identify the external server, or leave empty to configure later using the `TestConnection` command.

     Since this host key is for a VPC_LATTICE connector, remove the host name in the key
   - (Optional) For **Maximum concurrent connections**, choose the number of concurrent connections that your connector creates to the remote server (default is 5).
   - (Optional) For **Ordered secret version stages**, enter an ordered list of Secrets Manager version stage labels. The connector attempts authentication using each stage in order during file transfers. If you don't specify version stages, the connector uses only `AWSCURRENT`. For example, specify `AWSCURRENT`, `AWSPREVIOUS` to enable automatic fallback during secret rotation.
6. In the **Cryptographic algorithm options** section, choose a **Security policy** from the dropdown list.
7. (Optional) In the **Tags** section, add tags as key-value pairs.

8. Choose **Create SFTP connector** to create the VPC_LATTICE-enabled SFTP connector.

   The connector will be created with a status of `PENDING` while the resource association is being provisioned, which typically takes several minutes. Once the status changes to `ACTIVE`, the connector is ready for use.

CLI

Use the following command to create a VPC_LATTICE-enabled SFTP connector:

```
aws transfer create-connector \
    --url "sftp://my.sftp.server.com:22" \
    --access-role arn:aws:iam::123456789012:role/TransferConnectorRole \
    --sftp-config UserSecretId=my-secret-id,TrustedHostKeys="ssh-rsa AAAAB3NzaC..."
 \
    --egress-config VpcLattice={ResourceConfigurationArn=arn:aws:vpc-lattice:us-
east-1:123456789012:resourceconfiguration/rcfg-1234567890abcdef0} \
    --security-policy-name TransferSecurityPolicy-2024-01
```

The key parameter for VPC connectivity is `--egress-config`, which specifies the Resource Configuration ARN that defines your SFTP server target.

Monitoring VPC connector status

VPC_LATTICE-enabled connectors have an asynchronous setup process. After creation, monitor the connector status:

- **PENDING**: The connector is being provisioned. Service network provisioning is in progress, which typically takes several minutes.
- **ACTIVE**: The connector is ready for use and can transfer files.
- **ERRORED**: The connector failed to provision. Check the error details for troubleshooting information.

Check the connector status using the `describe-connector` command:

```
aws transfer describe-connector --connector-id c-1234567890abcdef0
```

During the PENDING state, the `test-connection` API will return "Connector not available" until provisioning is complete.

Limitations and considerations

- **Public endpoints**: When connecting to public endpoints through VPC, you must provide a DNS name in the Resource Configuration. Public IP addresses are not supported.
- **Regional availability**: VPC connectivity is available in select AWS Regions. Cross-region resource sharing is not supported.
- **Availability Zone requirements**: Resource Gateways must be associated with subnets in at least two Availability Zones. Not all Availability Zones support VPC Lattice in every region.
- **Connection limits**: Maximum of 350 connections per resource with a 350-second idle timeout for TCP connections.

Cost considerations

There are no additional charges from AWS Transfer Family beyond regular service charges. However, customers may be subject to additional charges from Amazon VPC Lattice associated with sharing their Amazon Virtual Private Cloud resources, and NAT gateway charges if they use their own NAT gateways for egress to internet.

For complete AWS Transfer Family pricing information, see the AWS Transfer Family pricing page.

VPC connectivity examples for SFTP connectors

This section provides examples of creating SFTP connectors with VPC connectivity for various scenarios. Before using these examples, ensure you have completed the VPC infrastructure setup as described in the VPC connectivity documentation.

Example: Private endpoint connection

This example shows how to create an SFTP connector that connects to a private SFTP server accessible only from your VPC.

Prerequisites

1. Create a Resource Gateway in your VPC:

```
aws vpc-lattice create-resource-gateway \
    --name my-private-server-gateway \
    --vpc-identifier vpc-1234567890abcdef0 \
    --subnet-ids subnet-1234567890abcdef0 subnet-0987654321fedcba0
```

2. Create a Resource Configuration for your private SFTP server:

```
aws vpc-lattice create-resource-configuration \
    --name my-private-server-config \
    --resource-gateway-identifier rgw-1234567890abcdef0 \
    --resource-configuration-definition ipResource={ipAddress="10.0.1.100"} \
    --port-ranges 22
```

Create the VPC_LATTICE-enabled connector

1. Create the SFTP connector with VPC connectivity:

```
aws transfer create-connector \
    --access-role arn:aws:iam::123456789012:role/TransferConnectorRole \
    --sftp-config UserSecretId=my-private-server-credentials,TrustedHostKeys="ssh-
rsa AAAAB3NzaC..." \
    --egress-config VpcLattice={ResourceConfigurationArn=arn:aws:vpc-lattice:us-
east-1:123456789012:resourceconfiguration/rcfg-1234567890abcdef0,PortNumber=22}
```

2. Monitor the connector status until it becomes `ACTIVE`:

```
aws transfer describe-connector --connector-id c-1234567890abcdef0
```

The remote SFTP server will see connections coming from the Resource Gateway's IP address within your VPC CIDR range.

Example: Public endpoint via VPC

This example shows how to route connections to a public SFTP server through your VPC to leverage centralized security controls and use your own NAT Gateway IP addresses.

Prerequisites

1. Create a Resource Gateway in your VPC (same as private endpoint example).
2. Create a Resource Configuration for the public SFTP server using its DNS name:

```
aws vpc-lattice create-resource-configuration \
    --name my-public-server-config \
    --resource-gateway-identifier rgw-1234567890abcdef0 \
```

```
    --resource-configuration-definition dnsResource={domainName="sftp.example.com"}
 \
    --port-ranges 22
```

> **Note**
>
> For public endpoints, you must use a DNS name, not an IP address.

Create the connector

- Create the SFTP connector:

```
aws transfer create-connector \
    --access-role arn:aws:iam::123456789012:role/TransferConnectorRole \
    --sftp-config UserSecretId=my-public-server-credentials,TrustedHostKeys="ssh-
rsa AAAAB3NzaC..." \
    --egress-config VpcLattice={ResourceConfigurationArn=arn:aws:vpc-lattice:us-
east-1:123456789012:resourceconfiguration/rcfg-0987654321fedcba0,PortNumber=22}
```

Traffic will flow from the connector to your Resource Gateway, then through your NAT Gateway to reach the public SFTP server. The remote server will see your NAT Gateway's Elastic IP address as the source.

Example: Cross-account private endpoint

This example shows how to connect to a private SFTP server in a different AWS account by using resource sharing.

> **Note**
>
> If you already have cross-VPC resource sharing enabled through other mechanisms, such as AWS Transit Gateway, you don't need to configure the resource sharing described here. The existing routing mechanisms, such as Transit Gateway route tables, are automatically used by SFTP connectors. You only need to create a Resource Configuration in the same account where you're creating the SFTP connector.

Account A (Resource Provider) - Share the Resource Configuration

1. Create Resource Gateway and Resource Configuration in Account A (same as previous examples).
2. Share the Resource Configuration with Account B using AWS Resource Access Manager:

```
aws ram create-resource-share \
    --name cross-account-sftp-share \
    --resource-arns arn:aws:vpc-lattice:us-
east-1:111111111111:resourceconfiguration/rcfg-1234567890abcdef0 \
    --principals 222222222222
```

Account B (Resource Consumer) - Accept and Use the Share

1. Accept the resource share invitation:

```
aws ram accept-resource-share-invitation \
    --resource-share-invitation-arn arn:aws:ram:us-east-1:111111111111:resource-
share-invitation/invitation-id
```

2. Create the SFTP connector in Account B:

```
aws transfer create-connector \
    --access-role arn:aws:iam::222222222222:role/TransferConnectorRole \
    --sftp-config UserSecretId=cross-account-server-
credentials,TrustedHostKeys="ssh-rsa AAAAB3NzaC..." \
    --egress-config VpcLattice={ResourceConfigurationArn=arn:aws:vpc-lattice:us-
east-1:111111111111:resourceconfiguration/rcfg-1234567890abcdef0,PortNumber=22}
```

The connector in Account B can now access the private SFTP server in Account A through the shared Resource Configuration.

Common troubleshooting scenarios

Here are solutions for common issues when creating VPC_LATTICE-enabled connectors:

- **Connector stuck in PENDING status**: Check that your Resource Gateway is ACTIVE and has subnets in supported Availability Zones. If the connector is still stuck with a status of PENDING,

call `UpdateConnector` using the same configuration parameters that you used initially. This triggers a new status event that might resolve the problem.

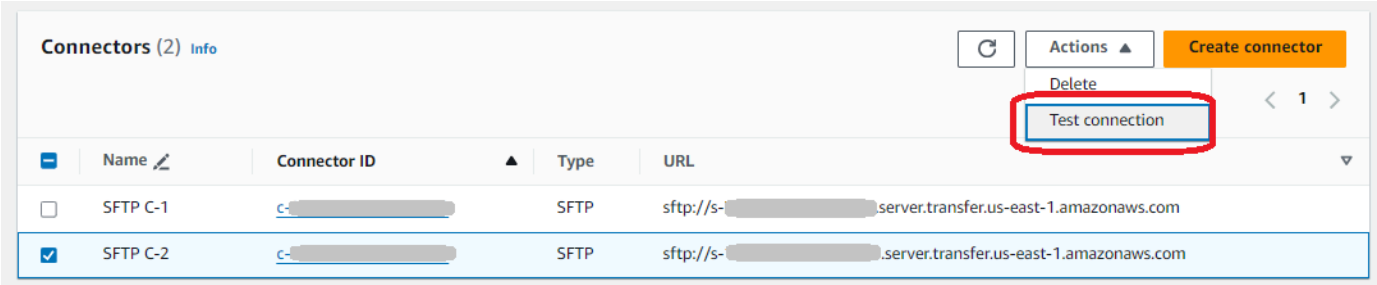
- **Connection timeouts**: Verify security group rules allow traffic on port 22 and that your VPC routing is correct.
- **DNS resolution issues**: For public endpoints, ensure your VPC has internet connectivity through a NAT Gateway or Internet Gateway.
- **Cross-account access denied**: Verify the resource share is accepted and the Resource Configuration ARN is correct. If the proper permission policy is attached to the resource configuration when the origin account creates the resource share, these permissions are required:`vpc-lattice:AssociateViaAWSService`, `vpc-lattice:AssociateViaAWSService-EventsAndStates`, `vpc-lattice:CreateServiceNetworkResourceAssociation`, `vpc-lattice:GetResourceConfiguration`.

Test an SFTP connector

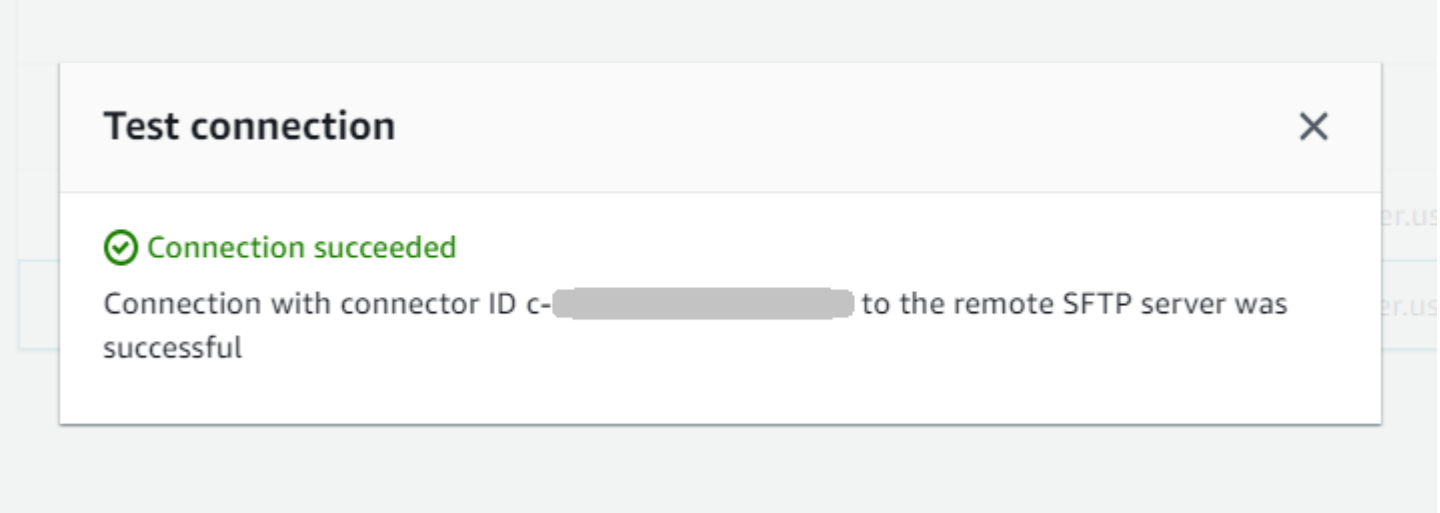
After you create an SFTP connector, we recommend that you test it before you attempt to transfer any files using your new connector.

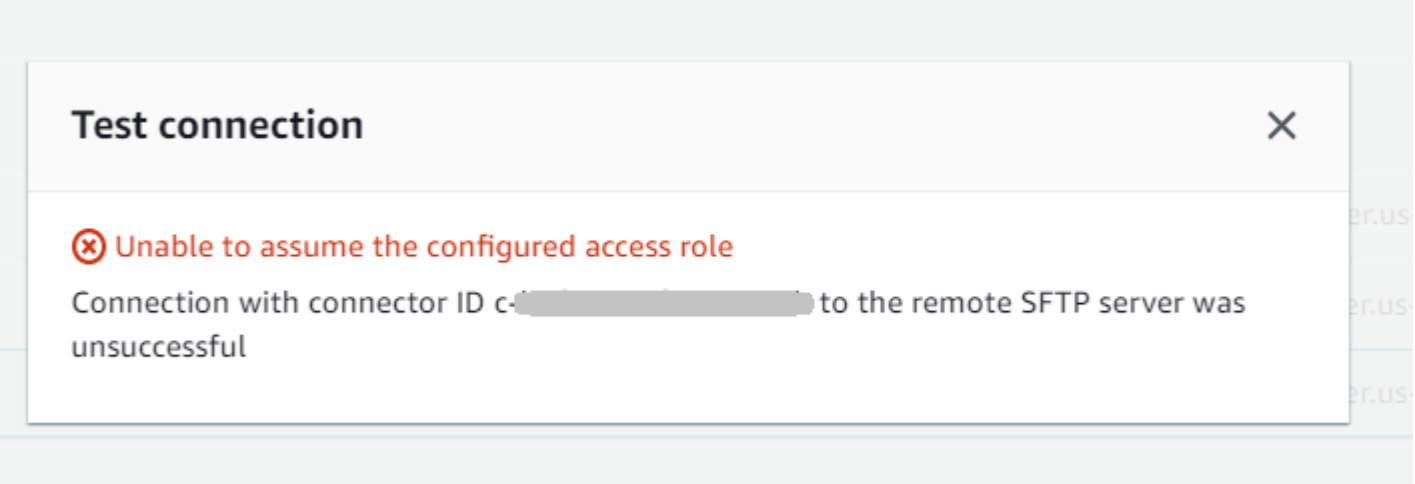
To test an SFTP connector

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **SFTP Connectors**, and select a connector.
3. From the **Actions** menu, choose **Test connection**.



The system returns a message, indicating whether the test passes or fails. If the test fails, the system provides an error message based on the reason the test failed.





Note

To use the API to test your connector, see the TestConnection API documentation.

VPC connectivity for SFTP connectors

AWS Transfer Family SFTP connectors support connectivity to remote SFTP servers through your VPC environments using Amazon VPC Lattice. This enables you to connect with privately hosted SFTP servers or route internet traffic through your VPC's security controls, and use your own NAT gateways and Elastic IP addresses.

Egress types

SFTP connectors can use one of two egress types:

- **Service Managed** (default): The connector uses NAT gateways and IP addresses owned by AWS Transfer Family to route connections over the public internet.
- **VPC_LATTICE**: The connector routes traffic through your VPC environment using Cross-VPC Resource Access.

When to use VPC connectivity

Use VPC connectivity for SFTP connectors in these scenarios:

- **Private SFTP servers**: Connect to SFTP servers that are only accessible from your VPC.
- **On-premises connectivity**: Connect to on-premises SFTP servers through AWS Direct Connect or AWS Site-to-Site VPN connections.
- **Custom IP addresses**: Use your own NAT gateways and Elastic IP addresses, including BYOIP scenarios.
- **Centralized security controls**: Route file transfers through your organization's central ingress/egress controls.

Requirements

Before creating a VPC_LATTICE-enabled SFTP connector, you need:

- VPC and related infrastructure (subnets, route tables, security groups)
- Resource Gateway in your VPC (minimum two Availability Zones)
- Resource Configuration specifying the target SFTP server

For detailed setup instructions, see Create a VPC_LATTICE-enabled SFTP connector. And, for examples, see VPC connectivity examples for SFTP connectors.

Using SFTP connectors

This topic describes how to perform the supported file operations using your SFTP connector. You can also find example commands to perform these operations by selecting your connector's details on the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.

After you have created an SFTP connector, you can use it to perform the following file operations on the remote SFTP server that it's associated with.

- Send files from Amazon S3 to the remote SFTP server.
- Retrieve files from the remote SFTP server to Amazon S3.
- List files and sub-folders from a directory on the remote SFTP server.
- Delete, rename or move files and directories on the remote SFTP server.

For details on creating connectors, see Creating SFTP connectors.

Topics

- Transfer files
- List contents of a remote directory
- Move, rename, or delete files or directories on the remote server

Transfer files

Topics

- Send and retrieve files by using an SFTP connector

Send and retrieve files by using an SFTP connector

To send and retrieve files by using an SFTP connector, you use the StartFileTransfer API operation and specify the following parameters, depending on whether you're *sending files* (outbound transfers) or *receiving files* (inbound transfers). Note that each `StartFileTransfer` request can contain 10 distinct paths.

> **Note**
>
> By default, SFTP connectors process one file at a time, transferring files sequentially. You have an option to accelerate transfer performance by having your connectors create concurrent sessions with remote servers that support concurrent sessions from the same user, and process up to 5 files in parallel.
>
> To enable concurrent connections for any connector, you can edit the **Maximum conncurent connections** setting when creating or updating a connector. For details, see Create an SFTP connector with service-managed egress.

- **Outbound transfers**
  - `send-file-paths` contains from one to ten source file paths, for files to transfer to the partner's SFTP server.
  - `remote-directory-path` is the remote path to send a file to on the customer's SFTP server.
- **Inbound transfers**
  - `retrieve-file-paths` contains from one to ten remote paths. Each path specifies a location for transferring files from the partner's SFTP server to your Transfer Family server.
  - `local-directory-path` is the Amazon S3 location (bucket and optional prefix) where your files are stored.

To send files, you specify the `send-file-paths` and `remote-directory-path` parameters. You can specify up to 10 files for the `send-file-paths` parameter. The following example command sends the files named `/amzn-s3-demo-source-bucket/file1.txt` and `/amzn-s3-demo-source-bucket/file2.txt`, located in Amazon S3 storage, to the `/tmp` directory on your partner's SFTP server. To use this example command, replace the *amzn-s3-demo-source-bucket* with your own bucket.

```
aws transfer start-file-transfer --send-file-paths /amzn-s3-demo-source-bucket/
file1.txt /amzn-s3-demo-source-bucket/file2.txt \
    --remote-directory-path /tmp --connector-id c-1111AAAA2222BBBB3 --region us-east-2
```

To retrieve files, you specify the `retrieve-file-paths` and `local-directory-path` parameters. The following example retrieves the files `/my/remote/file1.txt` and `/my/remote/file2.txt` on the partner's SFTP server, and places it in the Amazon S3 location /amzn-s3-demo-bucket/*prefix*. To use this example command, replace the *user input placeholders* with your own information.

```
aws transfer start-file-transfer --retrieve-file-paths /my/remote/file1.txt  /my/
remote/file2.txt \
   --local-directory-path /amzn-s3-demo-bucket/prefix --connector-id
 c-2222BBBB3333CCCC4 --region us-east-2
```

The previous examples specify absolute paths on the SFTP server. You can also use relative paths: that is, paths that are relative to the SFTP user's home directory. For example, if the SFTP user is `marymajor` and their home directory on the SFTP server is `/users/marymajor/`, the following

command sends `/amzn-s3-demo-source-bucket/file1.txt` to `/users/marymajor/test-connectors/file1.txt`

```
aws transfer start-file-transfer --send-file-paths /amzn-s3-demo-source-bucket/
file1.txt \
   --remote-directory-path test-connectors --connector-id c-2222BBBB3333CCCC4 --
region us-east-2
```

List contents of a remote directory

Before you retrieve files from a remote SFTP server, you can retrieve the contents of a directory on the remote SFTP server. To do this, you use the StartDirectoryListing API operation.

The following example lists the contents of the home folder on the remote SFTP server, which is specified in the connector's configuration. The results are placed into the Amazon S3 location `/amzn-s3-demo-bucket/connector-files`, and into a file named `c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json`.

```
aws transfer start-directory-listing  \
   --connector-id c-AAAA1111BBBB2222C  \
   --output-directory-path /amzn-s3-demo-bucket/example/connector-files  \
   --remote-directory-path /home
```

This AWS CLI command returns a listing ID and the name of the file that contains the results.

```
{
    "ListingId": "6666abcd-11aa-22bb-cc33-0000aaaa3333",
    "OutputFileName": "c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json"
}
```

Note

The naming convention for the output file is *connector-ID*-*listing-ID*.json.

The JSON file contains the following information:

- `filePath`: the complete path of a remote file, relative to the directory of the listing request for your SFTP connector on the remote server.

- `modifiedTimestamp`: the last time the file was modified, in seconds, Coordinated Universal Time (UTC) format. This field is optional. If the remote file attributes don't contain a timestamp, it is omitted from the file listing.
- `size`: the size of the file, in bytes. This field is optional. If the remote file attributes don't contain a file size, it is omitted from the file listing.
- `path`: the complete path of a remote directory, relative to the directory of the listing request for your SFTP connector on the remote server.
- `truncated`: a flag indicating whether the list output contains all of the items contained in the remote directory or not. If your `truncated` output value is true, you can increase the value provided in the optional `max-items` input attribute to be able to list more items (up to the maximum allowed list size of 200,000 items).

The following is an example of the contents of the output file (`c-AAAA1111BBBB2222C-6666abcd-11aa-22bb-cc33-0000aaaa3333.json`), where the remote directory contains two files and two sub-directories (paths).

```
{
    "files": [
        {
            "filePath": "/home/what.txt",
            "modifiedTimestamp": "2024-01-30T20:34:54Z",
            "size" : 2323
        },
        {
            "filePath": "/home/how.pgp",
            "modifiedTimestamp": "2024-01-30T20:34:54Z",
            "size" : 4691
        }
    ],
    "paths": [
        {
            "path": "/home/magic"
        },
        {
            "path": "/home/aws"
        },
    ],
    "truncated": "false"
}
```

Move, rename, or delete files or directories on the remote server

Topics

- Move or rename files or directories on the remote SFTP server
- Delete files or directories on the remote SFTP server

Move or rename files or directories on the remote SFTP server

You can use an SFTP connector to move or rename files and directories on a remote SFTP server. Note that the remote server needs to support these operations for successful processing using connectors.

Some common use cases are as follows.

- A remote server generates or receives a new file every hour, with the same filename but a different timestamp. To keep the main folder up to date (so that it contains only the latest file), you can use a connector to move older files to an archived folder.
- You use a connector to list all of the files in a remote directory, then transfer all of the files to your local storage. You can then use a connector to move the files to an archived folder on the remote server.

You must use a `StartRemoteMove` call for each file or directory you want to process, as the command takes a single source and destination file or directory as arguments. However, you can accelerate performance by having your connectors create concurrent sessions with remote servers that support concurrent sessions from the same user, and move/rename up to 5 files in parallel.

The following example moves a file on the remote SFTP server from `/source/folder/sourceFile` to `/destination/targetFile`, and returns a unique identifier for the operation.

```
aws transfer --connector-id c-AAAA1111BBBB2222C start-remote-move \
   --source-path /source/folder/sourceFile --target-path /destination/targetFile
```

> **Note**
>
> For the move/rename operations, Transfer Family uses the standard SFTP SSH_FXP_RENAME command to do the move/rename operation.

Delete files or directories on the remote SFTP server

You can use an SFTP connector to delete files or directories on a remote SFTP server. Note that the remote server needs to support these operations for successful processing using connectors.

> **Note**
>
> Delete operations for remote directories are only supported for empty directories.

Some common use cases are as follows.

- You use a connector to retrieve a file from a remote SFTP server, store it in your Amazon S3 bucket, then encrypt it. Finally, you can use a connector to delete the unencrypted file on the remote server.
- You use a connector to list all of the files in a remote directory, then transfer all of the files to your local storage. You can then use a connector to delete all of the files that you transferred. You could also delete the remote directory if you prefer.

You must use a `StartRemoteDelete` call for each file or directory you want to delete, as the command takes a single file or directory as an argument. However, you can accelerate performance by having your connectors create concurrent sessions with remote servers that support concurrent sessions from the same user, and delete up to 5 files/directories in parallel.

The following example deletes a file on the remote SFTP server in the path `/delete/folder/deleteFile`, and returns a unique identifier for the operation.

```
aws transfer start-remote-delete --connector-id c-AAAA1111BBBB2222C \
   --delete-path /delete/folder/deleteFile
```

> **Note**
>
> For the delete operation, Transfer Family uses the standard `SSH_FXP_REMOVE` command to delete a file, and `SSH_FXP_RMDIR` to delete a directory.

Monitoring SFTP connectors

You can monitor the status of your connector operations using any of the following ways. Choose the approach that meet your needs.

Use the connector API to query the status of file transfer requests

To track the progress of a file transfer operation, you use the ListFileTransferResults API operation, which returns real-time updates and detailed information on the status of each individual file being transferred in a specific file transfer operation. You specify the file transfer by providing its Connector ID and its Transfer ID. The following example returns a list of files for connector ID `a-11112222333344444` and transfer-ID `aa1b2c3d4-5678-90ab-cdef-EXAMPLE11111`.

```
aws transfer list-file-transfer-results --connector-id a-11112222333344444 --transfer-
id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

> **Note**
>
> File transfer results are available up to 7 days after you call the `ListFileTransferResults` API operation.

You can also view logs and events for your file transfer requests that use SFTP connectors. Amazon EventBridge events for Transfer Family are described in SFTP connector events. For how to view Transfer Family CloudWatch log entries, see Viewing Transfer Family log streams.

View SFTP connector events in Amazon EventBridge

For each operation performed by SFTP connectors, Transfer Family automatically generates and sends events to the default event bus in your Amazon EventBridge account. The events contain detailed metadata about the operation, including the operation status. You can subscribe to these events in EventBridge, apply filters on specific event criteria such as operation status, and automatically trigger downstream actions based on the status. For details on the events generated by SFTP connector operations, see SFTP connector events.

View SFTP connector logs in Amazon CloudWatch

All SFTP connector operations generate detailed logs in CloudWatch. For example log entries generated by SFTP connectors, see Example log entries for SFTP connectors.

Monitoring VPC egress type connectors

VPC egress type connectors provide additional monitoring capabilities and considerations beyond standard service managed connectors:

Connector status monitoring

VPC_LATTICE connectors include additional information to help you monitor the provisioning and operational state:

- **EgressType field**: Shows VPC for VPC_LATTICE egress type connectors
- **EgressConfig field**: Contains the Resource Configuration ARN and port information

Monitor connector status using the `describe-connector` API:

```
aws transfer describe-connector --connector-id c-1234567890abcdef0
```

VPC Lattice cost monitoring

VPC egress type connectors incur additional VPC Lattice charges that you should monitor:

- **Resource provider charges**: You are billed $0.006/GB for data processing as the resource provider (billed directly by VPC Lattice)
- **Resource consumer charges**: AWS Transfer Family absorbs the $0.01/GB resource consumer costs (first 1 PB)
- **NAT Gateway charges**: For public endpoints accessed via VPC, additional NAT Gateway and data transfer charges may apply
- **Transfer Family charges**: Standard $0.40/GB data processing fees still apply

Monitor VPC Lattice usage and costs through the AWS Cost and Billing console, filtering by the VPC Lattice service.

Network monitoring for VPC connectors

Monitor network activity and performance for VPC egress type connectors:

- **VPC Flow Logs**: Enable VPC Flow Logs to monitor network traffic patterns between Resource Gateways and SFTP servers

- **VPC Lattice access logs**: VPC Lattice provides access logs showing source/destination IP addresses, connection timing, and data transfer volumes
- **Security group monitoring**: Monitor security group rules and traffic patterns to ensure proper network access controls
- **DNS resolution monitoring**: Monitor DNS resolution times and failures for service network endpoints

Example VPC Lattice access log entry:

```
{
  "eventTimestamp": "2025-01-16T20:59:08.531Z",
  "serviceNetworkArn": "arn:aws:vpc-lattice:us-east-1:123456789012:servicenetwork/
sn-1234567890abcdef0",
  "sourceVpcArn": "arn:aws:ec2:us-east-1:123456789012:vpc/vpc-12345678",
  "resourceConfigurationArn": "arn:aws:vpc-lattice:us-
east-1:123456789012:resourceconfiguration/rcfg-12345678",
  "protocol": "tcp",
  "sourceIpPort": "10.0.1.100:33760",
  "destinationIpPort": "10.0.2.200:22",
  "gatewayIpPort": "10.0.1.150:1769",
  "resourceIpPort": "10.0.2.200:22"
}
```

Troubleshooting through monitoring

Use monitoring data to troubleshoot common VPC connector issues:

- **PENDING status**: Monitor DNS resolution progress and wait for ACTIVE status before attempting transfers
- **Connection timeouts**: Check VPC Flow Logs and security group rules for blocked traffic on port 22
- **Transfer failures**: Review CloudWatch logs for detailed error messages and VPC Lattice access logs for network-level issues
- **Performance issues**: Monitor VPC Lattice access logs for connection timing and throughput metrics

Managing SFTP connectors

This topic describes how to view and update SFTP connectors.

> **Note**
>
> Each connector is automatically assigned static IP addresses that remain unchanged over the lifetime of the connector. This allows you to connect with remote SFTP servers that only accept inbound connections from known IP addresses. Your connectors are assigned a set of static IP addresses that are shared by all connectors using the same protocol (SFTP or AS2) in your AWS account. These addresses are not dedicated exclusively to your account. If you need dedicated, predictable IP addresses, configure your connectors with VPC-based egress. For more information, see the section called “Create an SFTP connector with VPC-based egress”.

Update SFTP connectors

To change the existing parameter values for your connectors, you can run the `update-connector` command. The following command updates the secret for the connector *`connector-id`*, in the Region *`region-id`* to *`secret-ARN`*. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws transfer update-connector --sftp-config '{"UserSecretId":"secret-ARN"}' \
   --connector-id connector-id --region region-id
```

The following command configures ordered secret version stages for fallback authentication on an existing connector. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws transfer update-connector --sftp-config '{"OrderedUserSecretVersionStages":
 ["AWSCURRENT", "AWSPREVIOUS"]}' \
   --connector-id connector-id --region region-id
```

Updating VPC connectivity settings

You can update VPC connectivity settings for existing connectors, including switching between service-managed and VPC egress types or changing the Resource Configuration ARN.

To switch a connector from service-managed to VPC egress:

```
aws transfer update-connector \
   --connector-id connector-id \
   --egress-type VPC \
   --egress-config ResourceConfigurationArn=resource-configuration-arn
```

To update the Resource Configuration ARN for a VPC_LATTICE-enabled connector:

```
aws transfer update-connector \
   --connector-id connector-id \
   --egress-config ResourceConfigurationArn=new-resource-configuration-arn
```

Note

When updating VPC connectivity settings, the connector status will change to `PENDING` during the reconfiguration process. Monitor the connector status using the `describe-connector` command.

View SFTP connector details

You can find a list of details and properties for an SFTP connector in the AWS Transfer Family console.

To view connector details

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Connectors**.
3. Choose the identifier in the **Connector ID** column to see the details page for the selected connector.

You can change the properties for the SFTP connector by choosing **Edit** on the connector details page.

Monitoring VPC connector status

VPC_LATTICE-enabled connectors include additional status information to help you monitor the provisioning process:

- **Status**: Shows `PENDING`, `ACTIVE`, or `ERRORED`
- **EgressType**: Shows `VPC` or `SERVICE_MANAGED`
- **EgressConfig**: Contains the Resource Configuration ARN for VPC connectors
- **Error**: Provides detailed error information if the connector is in `ERRORED` state

For VPC connectors, the `ServiceManagedEgressIpAddresses` field will be null since traffic uses your VPC IP addresses instead.

> **Note**
>
> You can get much of this information, albeit in a different format, by running the following AWS Command Line Interface (AWS CLI) command. To use this example command, replace the *`user input placeholders`* with your own information.
>
> ```
> aws transfer describe-connector --connector-id your-connector-id
> ```
>
> For more information, see DescribeConnector in the API reference.

Scaling and quotas for SFTP connectors

Topics

- Quotas for SFTP connectors
- Scaling your SFTP connectors

Quotas for SFTP connectors

The following quotas are in place for SFTP connectors.

> **Note**
>
> More service quotas for SFTP connectors are listed in AWS Transfer Family endpoints and quotas in the *Amazon Web Services General Reference*.

SFTP connector quotas

Name	Default	Adjustable
Maximum test connection transactions per second (TPS)	1 request per second, per account	No
Maximum queue size for pending file transfers	1000	No
Maximum file size	150 gibibytes (GiB)	No
Maximum transfer time per file	12 hours	No
Maximum request wait time per file	12 hours	No
Maximum bandwidth for connectors per account (both SFTP and AS2 connectors contribute to this value)	50 MBps	No
Maximum number of items for directory listing operations	200,000	No
Maximum number of ordered secret version stages per connector	2	No

Note

By default, SFTP connectors process one file at a time, transferring files sequentially. You have an option to accelerate transfer performance by having your connectors create concurrent sessions with remote servers that support concurrent sessions from the same user, and process up to 5 files in parallel.

To enable concurrent connections for any connector, you can edit the **Maximum conncurent connections** setting when creating or updating a connector. For details, see Create an SFTP connector with service-managed egress.

For storing the credentials for SFTP connectors, there are quotas associated with each Secrets Manager secret. If you use the same secret to store multiple types of keys, for multiple purposes, you may encounter these quotas.

- Total length for a single secret: 12,000 characters
- Maximum length of the **`Password`** string: 1024 characters
- Maximum length of the **`PrivateKey`** string: 8192 characters
- Maximum length of the **`Username`** string: 100 characters

Scaling your SFTP connectors

This section describes considerations for how to scale your AWS Transfer Family SFTP connector workloads. You need to take into account the following four quotas that apply when you want to scale your workloads with SFTP connectors.

- **The maximum queue size.** This refers to the maximum number of pending operations in a connector's queue that have been requested. A pending operation refers to any previously submitted transfer request that has not yet completed, either successfully or unsuccessfully.

  The maximum queue depth for pending requests is currently set at 1,000 per connector (as defined in AWS Transfer Family service quotas). Your workloads may exceed this service limit when you request thousands of transfer operations over a short duration, and you will receive a `ThrottlingException` with the message Exceeded maximum pending requests. If your workloads are subject to this quota, contact the Transfer Family service team via AWS Support or your account team to discuss your scalability requirements.

  You can also take either or both of the following actions.

  - Distribute your file volumes across multiple connectors.
  - Have your connectors create concurrent sessions with the remote server to process multiple requests from the queue in parallel.

- **The number of concurrent sessions.** By default, an SFTP connector transfers one file at a time, transferring files sequentially from its queue.

  You have an option to accelerate transfer performance by having your connectors transfer multiple files in parallel. You can create concurrent sessions with remote servers that support concurrent sessions from the same user, and process up to 5 files in parallel. When you create an SFTP connector, choose a value up to 5 for the **Maximum concurrent connections** setting when you create or update the connector. For details, see Create an SFTP connector with service-managed egress.
- **The rate of `StartFileTransfer` requests.** You can request up to 100 file paths per second for transfer with each SFTP connector. The requested file paths are added to your connectors' queue for processing. You can use the `StartFileTransfer` command recursively to request up to 100 file paths per second per connector, irrespective of the number of files provided in an individual `StartFileTransfer` command.
- **The number of connectors per account.** By default, you can create up to 100 SFTP connectors per account in each AWS Region. If you distribute your file volumes across multiple connectors, you might reach this quota. You can request an increase by using Service Quotas. Requests for up to 1,000 SFTP connectors per account are approved automatically.

Reference architectures using SFTP connectors

This section lists the reference materials that are available for configuring automated file transfer workflows using SFTP connectors. You can design your own event-driven architectures by using the SFTP connector events in Amazon EventBridge, to orchestrate between your file transfer action and pre- and post-processing actions in AWS.

Blog posts

The following blog post provides a reference architecture to build an MFT workflow using SFTP connectors, including encryption of files using PGP before sending them to a remote SFTP server using SFTP connectors: Architecting secure and compliant managed file transfers with AWS Transfer Family SFTP connectors and PGP encryption.

Workshops

- The following workshop provides hands on labs for configuring SFTP connectors and using your connectors to send or retrieve files from remote SFTP servers: Transfer Family - SFTP workshop.

- The following workshop provides hands on labs to build fully automated and event-driven workflows involving file transfer to or from external SFTP servers to Amazon S3, and common pre- and post-processing of those files: Event-driven MFT workshop.

  This video provides a walk through of this workshop.

Solutions

AWS Transfer Family provides the following solutions:

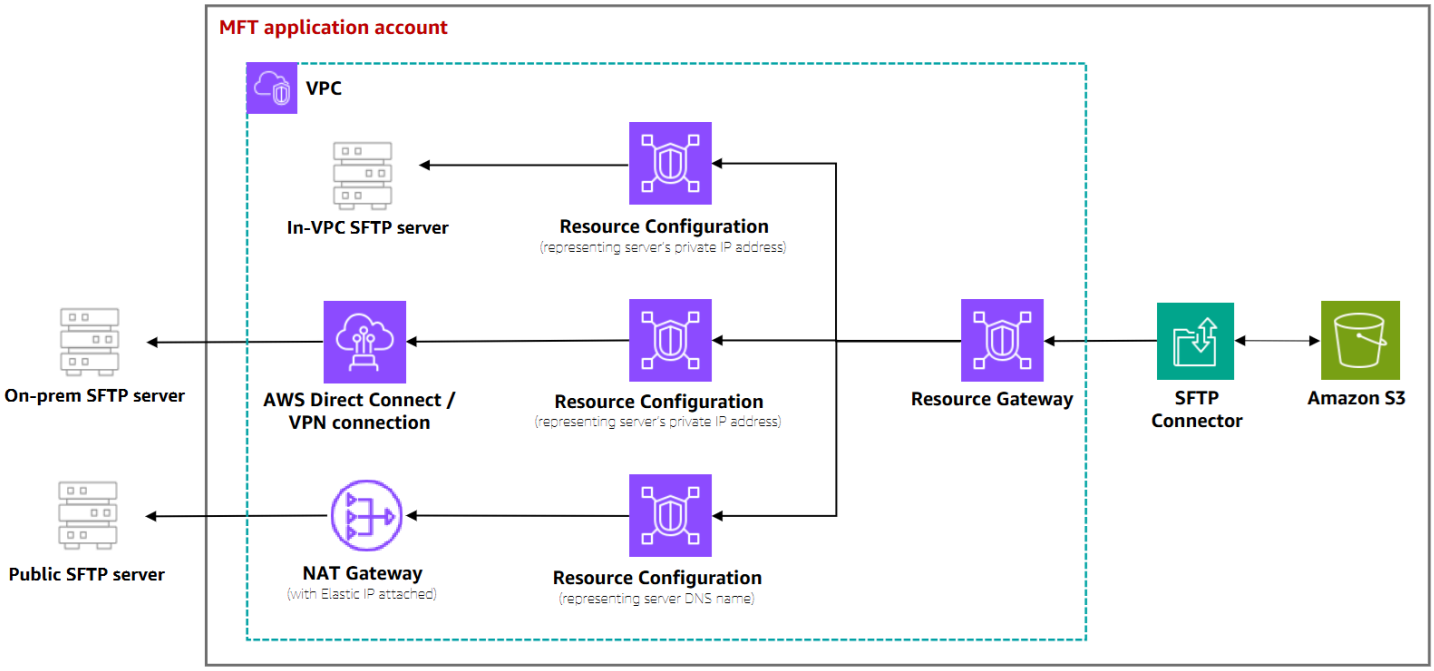
- The File transfer synchronization solution provides a reference architecture to automate the process of syncing remote SFTP directories—including entire folder structures—with your local Amazon S3 buckets using an SFTP connector. It orchestrates the process of listing remote directories, detecting changes, and transferring new or modified files.
- Serverlessland - Selective file transfer between remote SFTP server & S3; using AWS Transfer Family provides a sample pattern for listing files stored on remote SFTP locations, and transferring selective files to Amazon S3.

VPC reference architectures

The following reference architectures show common patterns for deploying VPC_LATTICE-enabled SFTP connectors. These examples help you understand where VPC Lattice resources need to be created in your overall AWS architecture.

Single account with shared egress infrastructure

In this architecture, the egress infrastructure (NAT Gateway, VPN tunnel, or Direct Connect) is configured in a VPC within the same account as your SFTP connectors. All connectors can share the same Resource Gateway and NAT Gateway.

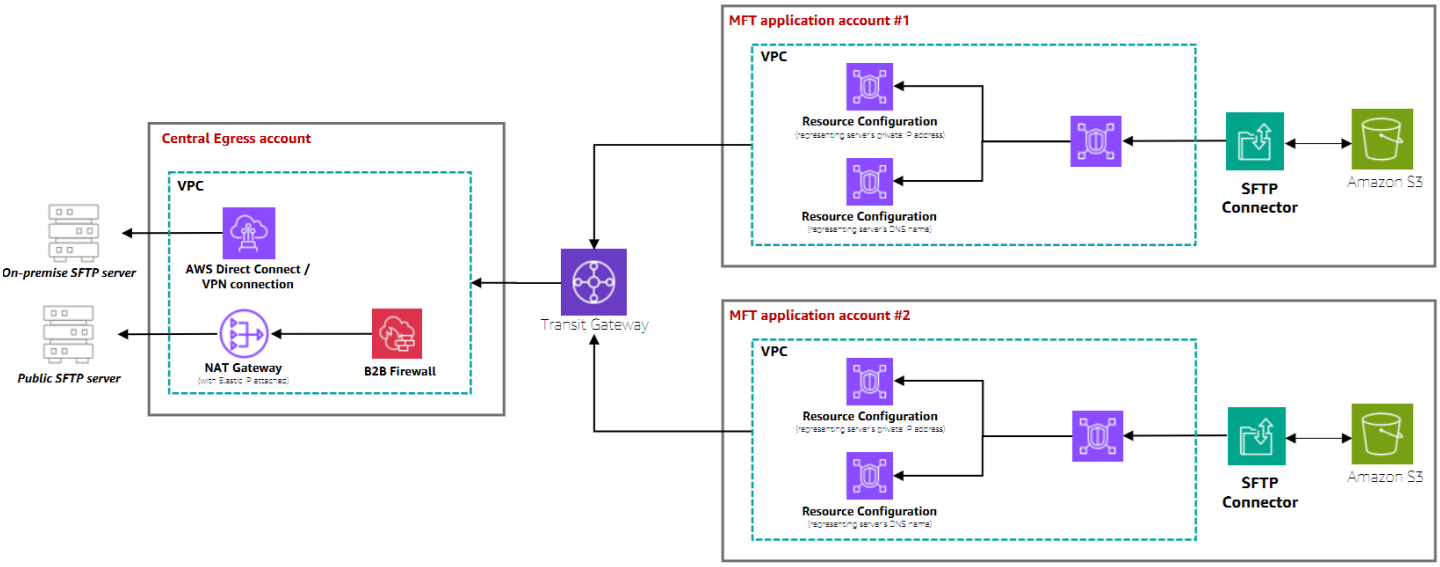


This pattern is ideal when:

- All SFTP connectors are managed within a single AWS account
- Egress infrastructure is setup in a VPC within the same account as your SFTP connectors

Cross-account with centralized egress infrastructure

In this architecture, the egress infrastructure (NAT Gateway, VPN tunnel, Direct Connect, or B2B Firewalls) is configured in a central Egress account managed by the networking team. SFTP connectors are created in the MFT Application account managed by the MFT admin team. Cross-account networking is established using Transit Gateway to honor existing networking rules.



This pattern is ideal when:

- Network infrastructure is managed by a separate team in a dedicated account
- You have existing routes (such as AWS Transit Gateway) between the account where SFTP connectors are created and the account where Egress infrastructure is setup. SFTP connectors will be able to leverage your existing routes connecting the two accounts.
- Centralized security controls and B2B firewalls are required
- You need to maintain separation of duties between networking and application teams

AWS Transfer Family for AS2

Applicability Statement 2 (AS2) is an RFC-defined file-transmission specification that includes strong message protection and verification mechanisms. Protecting an AS2 payload in transit uses Cryptographic Message Syntax (CMS) with encryption and digital signatures to provide data protection and peer authentication. A signed Message Disposition Notice (MDN) response payload provides verification (non-repudiation) that a message was received and successfully decrypted.

The AS2 protocol is critical for workflows with compliance requirements that rely on having data protection and security features built into the protocol. AWS Transfer Family AS2 endpoints are Drummond certified, enabling customers in industries such as retail, life sciences, manufacturing, financial services, and utilities to securely transact with their business partners.

When you use AS2 with Transfer Family, the transacted data is natively accessible in AWS for:

- Processing, analysis, and machine learning
- Integration with enterprise resource planning (ERP) systems
- Integration with customer relationship management (CRM) systems

To exchange files with a partner who has an AS2-enabled server, you must:

- Generate a public-private key pair for encryption
- Generate a public-private key pair for signing
- Exchange the public keys with your partner

> **Important**
>
> HTTPS AS2 server endpoints are not currently supported. You are responsible for TLS termination.

Transfer Family provides a workshop that you can attend, in which you can configure a Transfer Family endpoint with AS2 enabled, and a Transfer Family AS2 connector. You can view the details for this workshop here.

For step-by-step instructions to configure AS2 in Transfer Family, see the following:

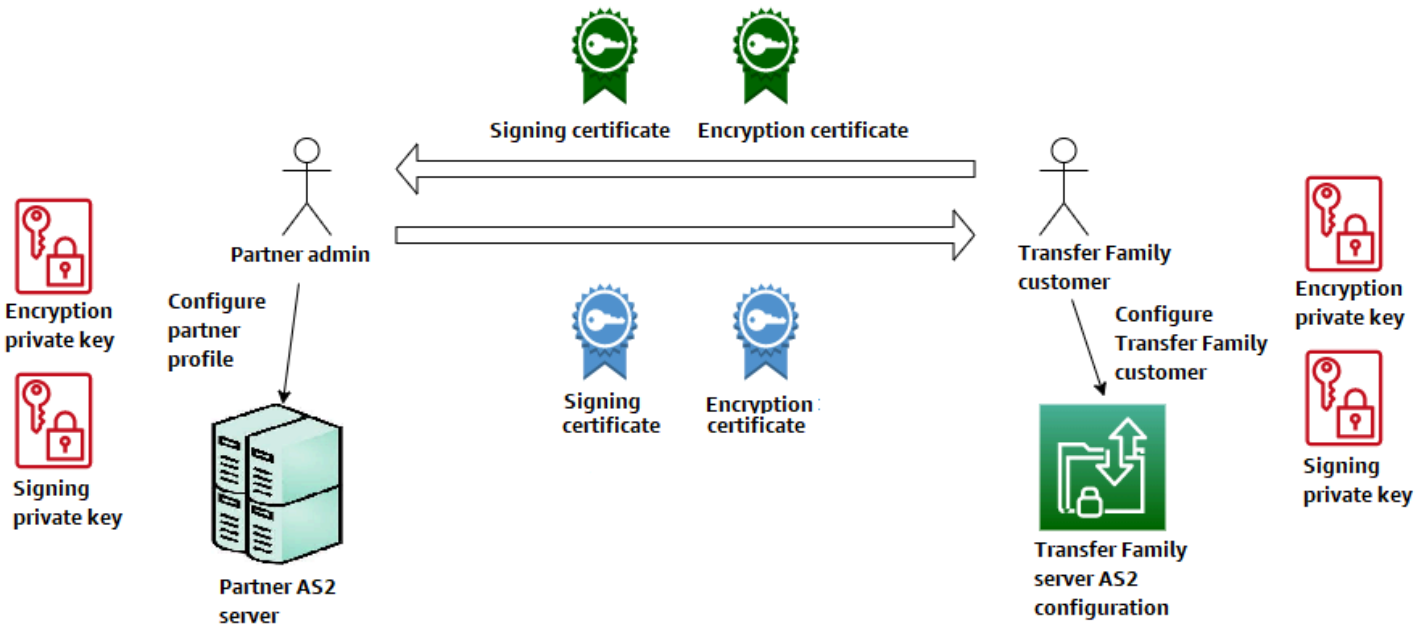
1. Import AS2 certificates
2. Create AS2 profiles
3. Create an AS2 server
4. Create an AS2 agreement
5. Configure AS2 connectors

For a complete example, see Setting up an AS2 configuration.

> **Note**
>
> To show support for AS2 Terraform templates, add a thumbs up reaction (#) to the Transfer Family Terraform templates feature request. You can also add a comment describing your use case.

AS2 use cases

If you are an AWS Transfer Family customer who wants to exchange files with a partner who has an AS2-enabled server, the most complex part of the setup involves generating one public-private key pair for encryption and another for signing and exchanging the public keys with the partner.



Consider the following variations for using AWS Transfer Family with AS2.

> **Note**
>
> *Trading partner* is the partner associated with that partner profile.
> All mentions of *MDN* in the following table assume *signed* MDNs.

AS2 use cases

Inbound-only use cases

- **Transfer encrypted AS2 messages from a trading partner to a Transfer Family server.**

  In this case, you do the following:

  1. Create profiles for your trading partner and yourself.
  2. Create a Transfer Family server that uses the AS2 protocol.
  3. Create an agreement and add it to your server.
  4. Import a certificate with a private key and add it to your profile, and then import the public key to your partner profile for encryption.
  5. After you have these items, send the public key for your certificate to your trading partner.

  Now your partner can send you encrypted messages and you can decrypt them and store them in your Amazon S3 bucket.
- **Transfer encrypted AS2 messages from a trading partner to a Transfer Family server and add signing.**

  In this scenario, you are still doing only inbound transfers, but now you want to have your partner sign the messages that they send. In this case, import the trading partner's signing public key (as a signing certificate added to your partner's profile).
- **Transfer encrypted AS2 messages from a trading partner to a Transfer Family server and add signing and sending an MDN response.**

  In this scenario, you are still doing only inbound transfers, but now, in addition to receiving signed payloads, your trading partner wants to receive a signed MDN response.

  1. Import your public and private signing keys (as a signing certificate to your profile).
  2. Send the public signing key to your trading partner.

Outbound-only use cases

- **Transfer encrypted AS2 messages from a Transfer Family server to a trading partner.**

  This case is similar to the inbound-only transfer use case, except that instead of adding an agreement to your AS2 server, you create a connector. In this case, you import your trading partner's public key to their profile.
- **Transfer encrypted AS2 messages from a Transfer Family server to a trading partner and add signing.**

  You are still doing only outbound transfers, but now your trading partner wants you to sign the message that you send to them.

  1. Import your signing private key (as a signing certificate added to your profile).
  2. Send your trading partner your public key.
- **Transfer encrypted AS2 messages from a Transfer Family server to a trading partner and add signing and send an MDN response.**

  You are still doing only outbound transfers, but now, in addition to sending signed payloads, you want to receive a signed MDN response from your trading partner.

  1. Your trading partner sends you their public signing key.
  2. Import your trading partner's public key (as a signing certificate added to your partner profile).

Inbound and outbound use cases

- **Transfer encrypted AS2 messages in both directions between a Transfer Family server and a trading partner.**

  In this case, you do the following:

  1. Create profiles for your trading partner and yourself.
  2. Create a Transfer Family server that uses the AS2 protocol.
  3. Create an agreement and add it to your server.
  4. Create a connector.
  5. Import a certificate with a private key and add it to your profile, and then import the public key to your partner profile for encryption.
  6. Receive a public key from your trading partner and add it to their profile for encryption.
  7. After you have these items, send the public key for your certificate to your trading partner.

  Now you and your trading partner can exchange encrypted messages, and you can both decrypt them. You can store the messages that you receive in your Amazon S3 bucket, and your partner can decrypt and store the messages that you send to them.
- **Transfer encrypted AS2 messages in both directions between a Transfer Family server and a trading partner and add signing.**

  Now you and your partner want signed messages.

  1. Import your signing private key (as a signing certificate added to your profile).
  2. Send your trading partner your public key.
  3. Import your trading partner's signing public key and add it to their profile.
- **Transfer encrypted AS2 messages in both directions between a Transfer Family server and a trading partner and add signing and send an MDN response.**

  Now, you want to exchange signed payloads, and both you and your trading partner want MDN responses.

  1. Your trading partner sends you their public signing key.
  2. Import your trading partner's public key (as a signing certificate to your partner profile).
  3. Send your public key to your trading partner.

AS2 CloudFormation templates

This topic provides information about AWS CloudFormation templates that you can use to quickly deploy AS2 servers and configurations for AWS Transfer Family. These templates automate the setup process and help you implement best practices for AS2 file transfers.

- The basic AS2 template is described in Use a template to create a demo Transfer Family AS2 stack
- The AS2 template for customizing HTTP headers is described in Customize HTTP headers for AS2 messages.

Customizing AS2 templates

You can customize the provided templates to meet your specific requirements:

1. Download the template from the S3 URL.
2. Modify the YAML code to adjust configurations such as:
   - Security settings and certificate configurations
   - Network architecture and VPC settings
   - Storage options and file handling
   - Monitoring and notification preferences
3. Upload your modified template to your own S3 bucket.
4. Deploy the customized template using the CloudFormation console or AWS CLI.

> **Important**
>
> When customizing templates, ensure that you maintain the dependencies between resources and follow security best practices.

Testing your AS2 deployment

After deploying an AS2 server using a template, you can test the configuration:

1. Check the CloudFormation stack outputs for sample commands and endpoint information.

2. Use the AWS CLI to send a test file:

```
aws s3api put-object --bucket your-bucket-name --key test.txt --body test.txt
aws transfer start-file-transfer --connector-id your-connector-id --send-file-
paths /your-bucket-name/test.txt
```

3. Verify file delivery in the destination S3 bucket.
4. Check CloudWatch logs for successful processing and MDN responses.

For more comprehensive testing, consider using third-party AS2 clients to send files to your Transfer Family AS2 server.

Best practices for AS2 template deployment

Follow these best practices when using AS2 CloudFormation templates:

Security

- Use strong certificates and rotate them regularly.
- Implement least-privilege IAM policies.
- Restrict network access using security groups.

Reliability

- Deploy across multiple Availability Zones.
- Implement monitoring and alerting for failed transfers.
- Set up automated retries for failed transfers.

Performance

- Choose appropriate instance types for your transfer volume.
- Implement S3 lifecycle policies for efficient file management.
- Monitor and optimize network configurations.

Cost Optimization

- Use auto-scaling for variable workloads.
- Implement S3 storage classes for older files.

Monitor and adjust resources based on actual usage.

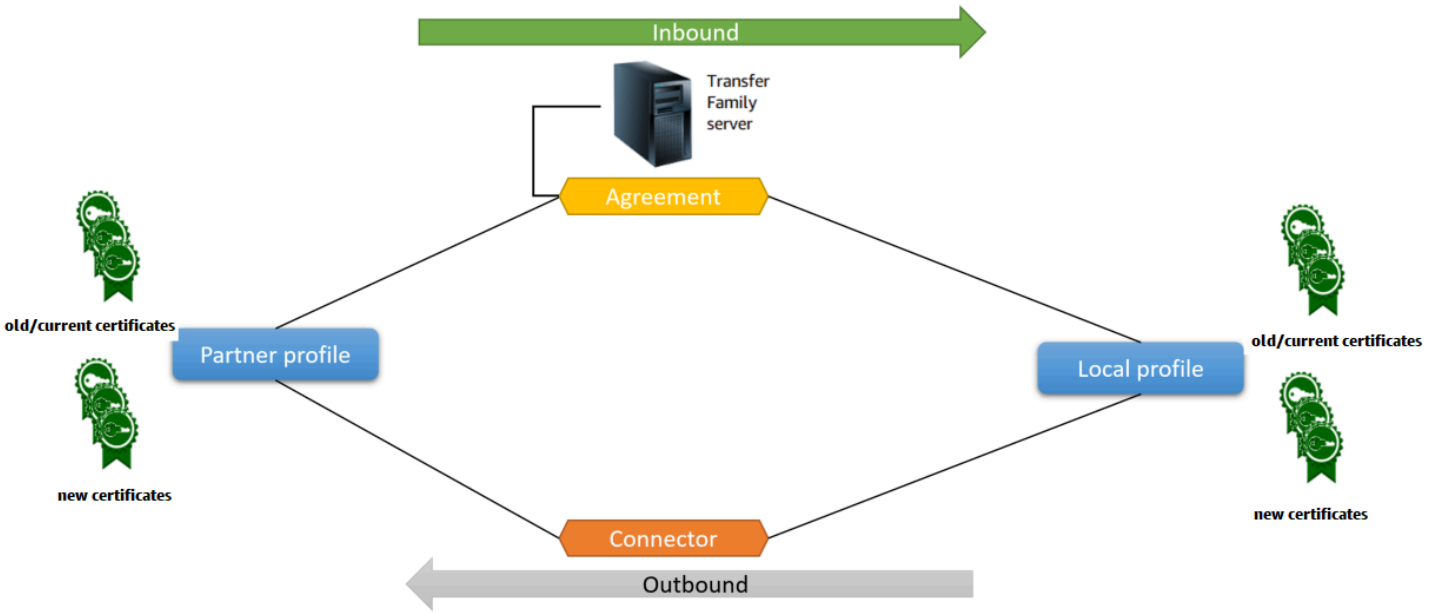
Configuring AS2

To create an AS2-enabled server, you must also specify the following components:

- **Agreements** – Bilateral trading partner *agreements*, or partnerships, define the relationship between the two parties that are exchanging messages (files). To define an agreement, Transfer Family combines server, local profile, partner profile, and certificate information. Transfer Family AS2-inbound processes use agreements.
- **Certificates** – *Public key (X.509) certificates* are used in AS2 communication for message encryption and verification. Certificates are also used for connector endpoints.
- **Local profiles and partner profiles** – A *local profile* defines the local (AS2-enabled Transfer Family server) organization or "party." Similarly, a *partner profile* defines the remote partner organization, external to Transfer Family.

While not required for all AS2-enabled servers, for outbound transfers, you need a **connector**. A connector captures the parameters for an outbound connection. The connector is required for sending files to a customer's external, non AWS server.

The following diagram shows the relationship between the AS2 objects involved in the inbound and outbound processes.



For an end-to-end example AS2 configuration, see Setting up an AS2 configuration.

Topics

- AS2 configurations
- AS2 quotas and limitations
- AS2 features and capabilities

AS2 configurations

This topic describes the supported configurations, features, and capabilities for transfers that use the Applicability Statement 2 (AS2) protocol, including the accepted ciphers and digests.

Signing, encryption, compression, MDN

For both inbound and outbound transfers, the following items are either required or optional:

- **Encryption** – Required (for HTTP transport, which is the only transport method currently supported). Unencrypted messages are only accepted if forwarded by a TLS-terminating proxy such as an Application Load Balancer (ALB) and the `X-Forwarded-Proto: https` header is present.
- **Signing** – Optional
- **Compression** – Optional (the only currently supported compression algorithm is ZLIB)
- **Message Disposition Notice (MDN)** – Optional

Ciphers

The following ciphers are supported for both inbound and outbound transfers:

- AES128_CBC
- AES192_CBC
- AES256_CBC
- 3DES (for backward compatibility only)

Digests

The following digests are supported:

- **Inbound signing and MDN** – SHA1, SHA256, SHA384, SHA512
- **Outbound signing and MDN** – SHA1, SHA256, SHA384, SHA512

MDN

For MDN responses, certain types are supported, as follows:

- **Inbound transfers** – Synchronous and asynchronous
- **Outbound transfers** – Synchronous and asynchronous
- **Simple Mail Transfer Protocol (SMTP) (email MDN)** – Not supported

Transports

- **Inbound transfers** – HTTP is the only currently supported transport, and you must specify it explicitly.

  > **Note**
  >
  > If you need to use HTTPS for inbound transfers, you can terminate TLS on an Application Load Balancer or a Network Load Balancer. This is described in Receive AS2 messages over HTTPS.

- **Outbound transfers** – If you provide an HTTP URL, you must also specify an encryption algorithm. If you provide an HTTPS URL, you have the option of specifying **NONE** for your encryption algorithm.

AS2 quotas and limitations

This section discusses quotas and limitations for AS2

Topics

- AS2 quotas
- Quotas for handling secrets
- Known limitations

AS2 quotas

The following quotas are in place for AS2 file transfers. To request an increase for a quota that's adjustable, see AWS service quotas in the *AWS General Reference*.

AS2 quotas

Name	Default	Adjustable
Maximum number of files per outbound request	10	No
Maximum number of outbound requests per second	100	No
Maximum number of inbound requests per second	100	No
Maximum outbound bandwidth per account (outbound SFTP and AS2 requests both contribute to this value)	50 MB per second	No

Quotas for handling secrets

AWS Transfer Family makes calls to AWS Secrets Manager on behalf of AS2 customers that are using Basic authentication. Additionally Secrets Manager makes calls to AWS KMS.

> **Note**
>
> These quotas aren't specific to your use of secrets for Transfer Family: they're shared among all the services in your AWS account.

For Secrets Manager `GetSecretValue`, the quota that applies is **Combined rate of DescribeSecret and GetSecretValue API requests**, as described in AWS Secrets Manager quotas.

Secrets Manager `GetSecretValue`

Name	Value	Description
Combined rate of DescribeSecret and GetSecretValue API requests	Each supported Region: 10,000 per second	The maximum transactions per second for `DescribeSecret` and `GetSecretValue` API operations combined.

For AWS KMS, the following quotas apply for `Decrypt`. For details, see Request quotas for each AWS KMS API operation

AWS KMS `Decrypt`

Quota name	Default value (requests per second)

Cryptographic operations (symmetric) request rate	These shared quotas vary with the AWS Region and the type of AWS KMS key used in the request. Each quota is calculated separately.
	• 5,500 (shared)
	• 10,000 (shared) in the following Regions:
	  • US East (Ohio), us-east-2
	  • Asia Pacific (Singapore), ap-southeast-1
	  • Asia Pacific (Sydney), ap-southeast-2
	  • Asia Pacific (Tokyo), ap-northeast-1
	  • Europe (Frankfurt), eu-central-1
	  • Europe (London), eu-west-2
	• 50,000 (shared) in the following Regions:
	  • US East (N. Virginia), us-east-1
	  • US West (Oregon), us-west-2
	  • Europe (Ireland), eu-west-1

Quota name	Default value (requests per second)
Custom key store request quotas **Note** This quota only applies if you are using an external key store.	Custom key store request quotas are calculated separately for each custom key store. • 1,800 (shared) for each AWS CloudHSM key store • 1,800 (shared) for each external key store

Known limitations

- Server-side TCP keep-alive is not supported. The connection times out after 350 seconds of inactivity unless the client sends keep-alive packets.
- For an active agreement to be accepted by the service and appear in Amazon CloudWatch logs, messages must contain valid AS2 headers.
- The server that's receiving messages from AWS Transfer Family for AS2 must support the Cryptographic Message Syntax (CMS) algorithm protection attribute for validating message signatures, as defined in RFC 6211. This attribute is not supported in some older IBM Sterling products.
- Duplicate message IDs result in a processed/Warning: duplicate-document message.
- The key length for AS2 certificates must be at least 2048 bits, and at most 4096.
- When sending AS2 messages or asynchronous MDNs to a trading partner's HTTPS endpoint, the messages or MDNs must use a valid SSL certificate that's signed by a publicly trusted certificate authority (CA). Self-signed certificates are currently supported for outbound transfers only.
- The endpoint must support the TLS version 1.2 protocol and a cryptographic algorithm that's permitted by the security policy (as described in Security policies for AWS Transfer Family servers).
- Multiple attachments and certificate exchange messaging (CEM) from AS2 version 1.2 is not currently supported.
- Basic authentication is currently supported for outbound messages only.
- You can attach a file-processing workflow to a Transfer Family server that uses the AS2 protocol: however, AS2 messages don't execute workflows attached to the server.

AS2 features and capabilities

The following tables list the features and capabilities available for Transfer Family resources that use AS2.

AS2 features

Transfer Family offers the following features for AS2.

Feature	Supported by AWS Transfer Family
Drummond certification	Yes
AWS CloudFormation support	Yes
Amazon CloudWatch metrics	Yes
SHA-2 cryptographic algorithms	Yes
Support for Amazon S3	Yes
Support for Amazon EFS	No
Scheduled Messages	Yes [1]
AWS Transfer Family Managed Workflows	No
Certificate Exchange Messaging (CEM)	No
Mutual TLS (mTLS)	No
Support for self-signed certificates	Yes

1. Outbound Scheduled Messages available by scheduling AWS Lambda functions using Amazon EventBridge

AS2 send and receive capabilities

The following table provides a list of AWS Transfer Family AS2 send and receive capabilities.

Capability	Inbound: Receiving with server	Outbound: Sending with connector
TLS Encrypted Transport (HTTPS)	Yes [1]	Yes
Non-TLS Transport (HTTP)	Yes	Yes [2]
Synchronous MDN	Yes	Yes
Message Compression	Yes	Yes
Asynchronous MDN	Yes	Yes
Static IP Address	Yes	Yes
Bring Your Own IP Address	Yes	No
Multiple File Attachments	No	No
Basic Authentication	No	Yes
AS2 Restart	Not applicable	No
AS2 Reliability	No	No
Custom Subject per Message	Not applicable	No

1. Inbound TLS Encrypted Transport available with Network Load Balancer (NLB) or Application Load Balancer (ALB)

2. Outbound non-TLS Transport available only when encryption is enabled

Manage AS2 certificates

This topic discusses how to import and manage AS2 certificates. Importing certificates is the first step in the AS2 process for Transfer Family.

1. Import certificates
2. Create AS2 profiles

3. Create an AS2 server
4. Create an AS2 agreement
5. Configure AS2 connectors

Import AS2 certificates

The Transfer Family AS2 process uses certificate keys for both encryption and signing of transferred information. Partners can use the same key for both purposes, or a separate key for each. If you have common encryption keys kept in escrow by a trusted third-party so that data can be decrypted in the event of a disaster or security breach, we recommend having separate signing keys. By using separate signing keys (which you do not escrow), you don't compromise the non-repudiation features of your digital signatures.

> **Note**
>
> The key length for AS2 certificates must be at least 2048 bits, and at most 4096.

The following points detail how AS2 certificates are used during the process.

- Inbound AS2
  - The trading partner sends their public key for the signing certificate, and this key is imported to the partner profile.
  - The local party sends the public key for their encryption and signing certificates. The partner then imports the private key or keys. The local party can send separate certificate keys for signing and encryption, or can choose to use the same key for both purposes.
- Outbound AS2
  - The partner sends the public key for their encryption certificate, and this key is imported to the partner profile.
  - The local party sends the public key for the certificate for signing, and imports the private key of the certificate for signing.
  - If you are using HTTPS, you can import a self-signed Transport Layer Security (TLS) certificate.

For details on how to create certificates, see the section called “Step 1: Create certificates for AS2”.

This procedure explains how to import certificates by using the Transfer Family console. If you want to use the AWS CLI instead, see the section called "Step 2: Import certificates as Transfer Family certificate resources".

To specify an AS2-enabled certificate

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, under **AS2 Trading Partners**, choose **Certificates**.
3. Choose **Import certificate**.
4. In the **Certificate configuration** section, for **Certificate description**, enter an easily identifiable name for the certificate. Make sure that you can identify the certificate's purpose by its description. Additionally, choose the role for the certificate.
5. In the **Certificate usage** section, choose the purpose for this certificate. It can be used for encryption, signing, or both.

   Tip: If you choose **Encryption and signing** for the usage, Transfer Family creates two identical certificates (each having their own ID): one with a usage value of `ENCRYPTION` and one with a usage value of `SIGNING`.
6. In the **Certificate contents** section, provide a public certificate from a trading partner, or the public and private keys for a local certificate.

   Fill in the **Certificate contents** section with the appropriate details.

   - If you choose **Self-signed certificate**, you do not provide the certificate chain.
   - Paste the certificate text and its chain into the **Certificate and Certificate chain** field.
   - If this certificate is a local certificate, paste in its private key.
7. Choose **Import certificate** to complete the process and save the details for the imported certificate.

> **Note**
>
> TLS certificates can only be imported as a partner's public certificate. If you select **Public certificate from a partner**, and then select **Transport Layer Security (TLS)** for the usage, you receive a warning. Also, TLS certificates must be self-signed (that is, you must select **Self Signed Certificate** to import a TLS certificate).

AS2 certificate rotation

Often, certificates are valid for a period of six months to a year. You might have set up profiles that you want to persist for a longer duration. To facilitate this, Transfer Family provides certificate rotation. You can specify multiple certificates for a profile, allowing you to keep using the profile for multiple years. Transfer Family uses certificates for signing (optional) and encryption (mandatory). You can specify a single certificate for both purposes, if you like.

Certificate rotation is the process of replacing an old expiring certificate with a newer certificate. The transition is a gradual one to avoid disrupting transfers where a partner in the agreement has yet to configure a new certificate for outbound transfers or might be sending payloads that are signed or encrypted with an old certificate during a period when a newer certificate might also be in use. The intermediate period where both old and new certificates are valid is referred to as a *grace period*.

X.509 certificates have `Not Before` and `Not After` dates. However, these parameters might not provide enough control for administrators. Transfer Family provides `Active Date` and `Inactive Date` settings to control which certificate is used for outbound payloads and which is accepted for inbound payloads.

Certificate expiration monitoring

Transfer Family publishes a Amazon CloudWatch metric `DaysUntilExpiry` after importing a certificate. The metric emits the number of days between the current date and the date specified as the `InactiveDate` on the Certificate. The metric is found under the `Transfer` AWS namespace in the CloudWatch metrics dashboard.

This metric will always have a metric dimension for **CertificateId** and will optionally include a **Description** dimension if provided by the customer on the certificate. For more information about CloudWatch metric dimensions, see Dimension in the *CloudWatch API Reference*.

> **Note**
>
> It can take up to a full day after importing a Certificate for Transfer Family to emit this metric to the customer account.

You can use this metric to create CloudWatch alarms that notify you when certificates are approaching expiration.

Outbound certificate selection uses the maximum value that is prior to the date of the transfer as an `Inactive Date`. Inbound processes accept certificates within the range of `Not Before` and `Not After` and within the range of `Active Date` and `Inactive Date`.

Certificate rotation example

The following table describes one possible way to configure two certificates for a single profile.

Two certificates in rotation

Name	NOT BEFORE (controlled by certificate authority)	ACTIVE DATE (set by Transfer Family)	INACTIVE DATE (set by Transfer Family)	NOT AFTER (set by certificate authority)
Cert1 (older certificate)	2019-11-01	2020-01-01	2020-12-31	2024-01-01
Cert2 (newer certificate)	2020-11-01	2020-06-01	2021-06-01	2025-01-01

Note the following:

- When you specify an `Active Date` and `Inactive Date` for a certificate, the range must be inside the range between `Not Before` and `Not After`.
- We recommend that you configure several certificates for each profile, making sure that the active date range for all the certificates combined covers the amount of time for which you want to use the profile.
- We recommend that you specify some grace time between when your older certificate becomes inactive and when your newer certificate becomes active. In the preceding example, the first certificate does not become inactive until 2020-12-31, while the second certificate becomes active on 2020-06-01, providing a 6-month grace period. During the period from 2020-06-01 until 2020-12-31, both certificates are active.

Create AS2 profiles

This topic discusses how to create profiles for use in the AS2 process. A *local profile* defines the local (AS2-enabled Transfer Family server) organization or "party." Similarly, a *partner profile* defines the remote partner organization, external to Transfer Family.

1. Import AS2 certificates
2. Create AS2 profiles
3. Create an AS2 server
4. Create an AS2 agreement
5. Configure AS2 connectors

Use this procedure to create both local and partner profiles. This procedure explains how to create AS2 profiles by using the Transfer Family console. If you want to use the AWS CLI instead, see the section called “Step 3: Create profiles for you and your trading partner”.

To create an AS2 profile

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, under **AS2 Trading Partners**, choose **Profiles**, then choose **Create profile**.
3. In the **Profile configuration** section, enter the AS2 ID for the profile. This value is used for the AS2 protocol-specific HTTP headers `as2-from` and `as2-to` to identify the trading partnership, which determines the certificates to use, and so on.
4. In the **Profile type** section, choose **Local profile** or **Partner profile**.
5. In the **Certificates** section, choose one or more certificates from the dropdown menu.

   Tip: If you want to import a certificate that is not listed in the dropdown menu, select **Import a new Certificate**. This opens a new browser window at the **Import certificate** screen. For the procedure about importing certificates see Import AS2 certificates.
6. (Optional) In the **Tags** section, specify one or more key-value pairs to help identify this profile.
7. Choose **Create profile** to complete the process and save the new profile.

Create an AS2 server

This topic provides instructions for creating an AS2-enabled Transfer Family server, using either the console or a CloudFormation template. For an end-to-end example AS2 configuration, see Setting up an AS2 configuration. After you create an AS2 server, you can add an agreement to the server.

1. Import AS2 certificates
2. Create AS2 profiles
3. Create an AS2 server
4. Create an AS2 agreement
5. Configure AS2 connectors

Topics

- Create an AS2 server using the Transfer Family console
- Use a template to create a demo Transfer Family AS2 stack
- Create an AS2 agreement

Create an AS2 server using the Transfer Family console

This procedure explains how to create an AS2-enabled server by using the Transfer Family console. If you want to use the AWS CLI instead, see the section called “Step 4: Create a Transfer Family server that uses the AS2 protocol”.

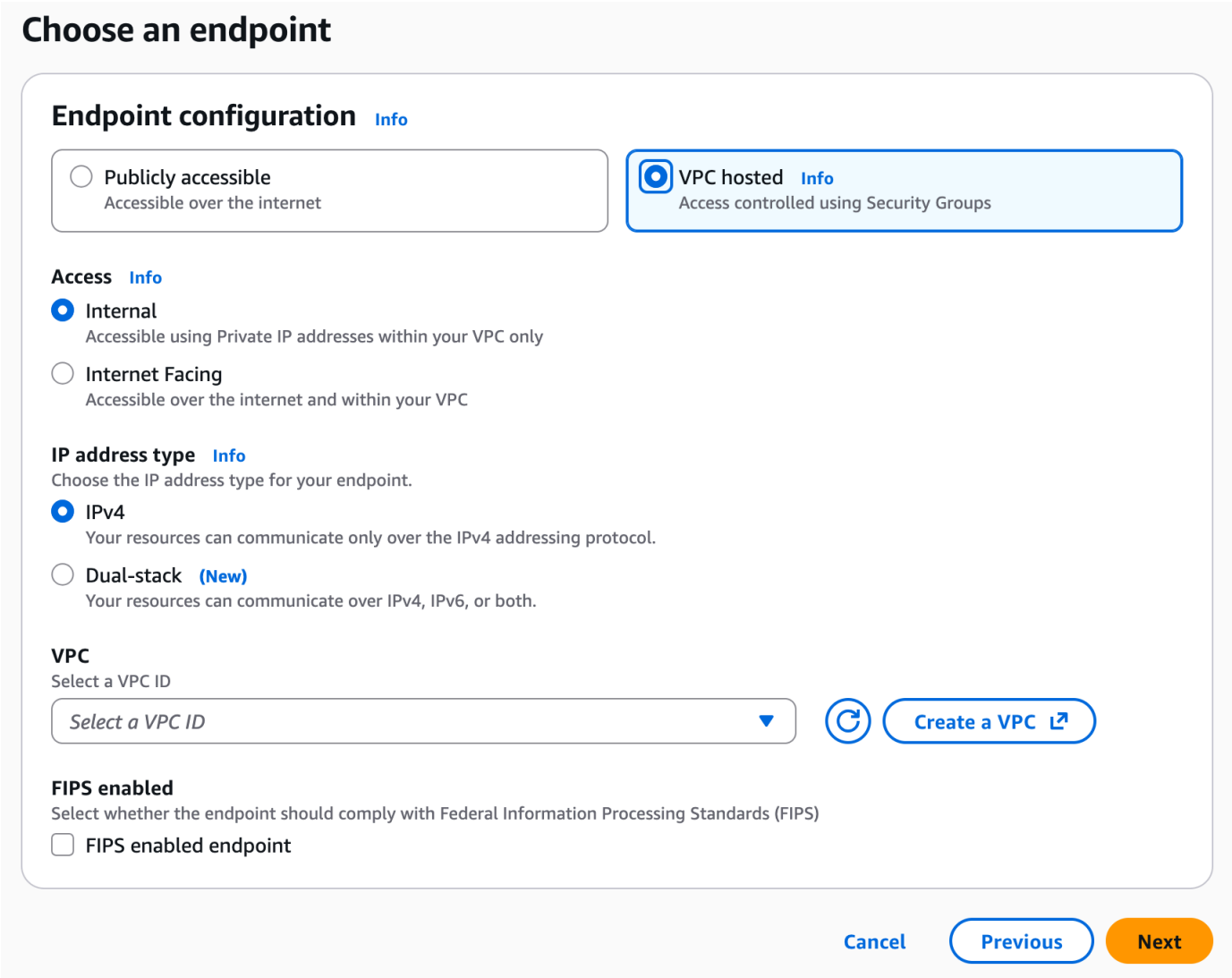
> **Note**
>
> You can attach a file-processing workflow to a Transfer Family server that uses the AS2 protocol: however, AS2 messages don't execute workflows attached to the server.

To create an AS2-enabled server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**, and then choose **Create server**.
3. On the **Choose protocols** page, select **AS2 (Applicability Statement 2)**, and then choose **Next**.
4. On the **Choose an identity provider** page, choose **Next**.

> **Note**
>
> For AS2, you cannot choose an identity provider because basic authentication is not supported for the AS2 protocol. Instead, you control access through virtual private cloud (VPC) security groups.

5. On the **Choose an endpoint** page, do the following:



   a. For **Endpoint type**, choose **VPC hosted** to host your server's endpoint. For information about setting up your VPC-hosted endpoint, see Create a server in a virtual private cloud.

> **Note**
>
> Publicly accessible endpoints are not supported for the AS2 protocol. To make your VPC endpoint accessible over the internet, choose **Internet Facing** under **Access**, and then supply your Elastic IP addresses.

b. For **Access**, choose one of the following options:

- **Internal** – Choose this option to provide access from within your VPC and VPC-connected environments, such as an on-premises data center over Direct Connect or VPN.
- **Internet Facing** – Choose this option to provide access over the internet and from within your VPC and VPC-connected environments, such as an on-premises data center over Direct Connect or VPN.

  If you choose **Internet Facing**, supply your Elastic IP addresses when prompted.

c. For **VPC**, either choose an existing VPC or choose **Create VPC** to create a new VPC.

d. For **FIPS Enabled**, keep the **FIPS Enabled endpoint** check box cleared.

> **Note**
>
> FIPS-enabled endpoints are not supported for the AS2 protocol.

e. Choose **Next**.

6. On the **Choose a domain** page, choose **Amazon S3** to store and access your files as objects by using the selected protocol.

   Choose **Next**.

7. On the **Configure additional details** page, choose the settings that you need.

> **Note**
>
> If you are configuring any other protocols along with AS2, all of the additional detail settings apply. However, for the AS2 protocol, the only settings that apply are those in the **CloudWatch logging** and **Tags** sections.

Even though setting up a CloudWatch logging role is optional, we highly recommend setting it up so that you can see the status of your messages and troubleshoot configuration issues.

8. On the **Review and create** page, review your choices to make sure they are correct.

   - If you want to edit any of your settings, choose **Edit** next to the step that you want to change.

     Note

     If you edit a step, we recommend that you review each step after the step that you chose to edit.

   - If you have no changes, choose **Create server** to create your server. You are taken to the **Servers** page, shown following, where your new server is listed.

     It can take several minutes before the status for your new server changes to **Online**. At that point, your server can perform file operations for your users.

Use a template to create a demo Transfer Family AS2 stack

We supply a self-contained, CloudFormation template to quickly create an AS2-enabled Transfer Family server. The template configures the server with a public Amazon VPC endpoint, certificates, local and partner profiles, an agreement, and a connector.

The basic AS2 server template creates the following resources:

- An AS2-enabled Transfer Family server with a VPC endpoint
- Local and partner AS2 profiles with certificates
- An agreement between the profiles
- An Amazon S3 bucket for file storage
- Required IAM roles and policies
- CloudWatch logging configuration

Before using this template, note the following:

- If you create a stack from this template, you will be billed for the AWS resources that are used.
- The template creates multiple certificates and places them in AWS Secrets Manager to store them securely. You can delete these certificates from Secrets Manager if you want, because you're charged for using this service. Deleting these certificates in Secrets Manager doesn't delete them from the Transfer Family server. Therefore, the functionality of the demo stack isn't affected. However, for certificates that you're going to use with a production AS2 server, you might want to use Secrets Manager to manage and periodically rotate your stored certificates.
- We recommend that you use the template as a base only, and mainly for demonstration purposes. If you want to use this demo stack in production, we recommend that you modify the template's YAML code to create a more robust stack. For example, create production-level certificates, and create an AWS Lambda function that you can use in production.

To create an AS2-enabled Transfer Family server from a CloudFormation template

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the left navigation pane, choose **Stacks**.
3. Choose **Create stack**, and then choose **With new resources (standard)**.
4. In the **Prerequisite - Prepare template** section, choose **Choose an existing template**.
5. Copy this link, AS2 demo template, and paste it into the **Amazon S3 URL** field.
6. Choose **Next**.
7. On the **Specify stack details** page, name your stack, and then specify the following parameters:

   - Under **AS2**, enter values for **Local AS2 ID** and **Partner AS2 ID**, or accept the defaults, `local` and `partner`, respectively.
   - Under **Network**, enter a value for **Security group ingress CIDR IP**, or accept the default, `0.0.0.0/0`.

   > **Note**
   >
   > This value, in CIDR format, specifies which IP addresses are allowed for incoming traffic to the AS2 server. The default value, `0.0.0.0/0`, allows all IP addresses.

- Under **General**, enter a value for **Prefix**, or accept the default, `transfer-as2`. This prefix is placed before any resource names that are created by the stack. For example, if you use the default prefix, your Amazon S3 bucket is named `transfer-as2-`*`amzn-s3-demo-bucket`*.

8. Choose **Next**. On the **Configure stack options** page, choose **Next** again.
9. Review the details for the stack that you're creating, and then choose **Create stack**.

> **Note**
>
> At the bottom of the page, under **Capabilities**, you must acknowledge that CloudFormation might create AWS Identity and Access Management (IAM) resources.

After the stack is created, you can send a test AS2 message from the partner server to your local Transfer Family server by using the AWS Command Line Interface (AWS CLI). A sample AWS CLI command for sending a test message is created along with all of the other resources in the stack.

To use this sample command, go to the **Outputs** tab of your stack, and copy the **TransferExampleAs2Command**. You can then run the command by using the AWS CLI. If you haven't already installed the AWS CLI, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

The sample command has the following format:

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key test.txt && aws transfer start-file-transfer --region aws-region --connector-id TransferConnectorId --send-file-paths /amzn-s3-demo-bucket/test.txt
```

> **Note**
>
> Your version of this command contains the actual values for the *amzn-s3-demo-bucket* and *TransferConnectorId* resources in your stack.

This sample command consists of two separate commands that are chained together by using the && string.

The first command creates a new, empty text file in your bucket:

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key test.txt
```

Then, the second command uses the connector to send the file from the partner profile to the local profile. The Transfer Family server has an agreement set up that allows the local profile to accept messages from the partner profile.

```
aws transfer start-file-transfer --region aws-region --connector-id TransferConnectorId
 --send-file-paths /amzn-s3-demo-bucket/test.txt
```

After you run the command, you can go to your Amazon S3 bucket (*amzn-s3-demo-bucket*) and view the contents. If the command is successful, you should see the following objects in your bucket:

- `processed/` – This folder contains a JSON file that describes the transferred file and the MDN response.
- `processing/` – This folder temporarily contains files as they are being processed, but after a transfer is completed, this folder should be empty.
- *server-id*`/` – This folder is named based on your Transfer Family server ID. It contains `from-`*partner* (this folder is dynamically named, based on the partner's AS2 ID), which itself contains `failed/`, `processed/`, and `processing/` folders. The `/`*server-id*`/from-`*partner*`/processed/` folder contains a copy of the transferred text file, and the corresponding JSON and MDN files.
- `test.txt` – This object is the (empty) file that was transferred.

Create an AS2 agreement

Agreements are associated with Transfer Family servers. They specify the details for trading partners that use the AS2 protocol to exchange messages or files by using Transfer Family, for *inbound* transfers—sending AS2 files from an external, partner-owned source to a Transfer Family server.

This procedure explains how to create AS2 agreements by using the Transfer Family console. If you want to use the AWS CLI instead, see the section called “Step 5: Create an agreement between you and your partner”.

To create an agreement for a Transfer Family server

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**, and then choose a server that uses the AS2 protocol.

   As an alternative, as long as you have at least one Transfer Family server that uses the AS2 protocol, select **Agreements to receive messages** from the **AS2 Trading Partners** menu. Then, in the **Create agreement** screen, select the AS2 server to which you want to associate this agreement.
3. On the server details page, scroll down to the **Agreements** section.
4. Choose **Add agreement**.
5. Fill in the agreement parameters, as follows:

   a. In the **Agreement configuration** section, enter a descriptive name. Make sure that you can identify the agreement's purpose by its name. Also, set the **Status** for the agreement: either **Active** (selected by default) or **Inactive**.

   b. In the **Communication configuration** section, choose a local profile and a partner profile. Also, choose whether or not to enforce message signing.

      - By default, **Enforce message signing** is enabled, which means that Transfer Family rejects unsigned messages from your trading partner for this agreement.
      - Clear this setting to allow Transfer Family to accept unsigned messages from your trading partner for this agreement.

   c. In the **Inbox directory configuration** section, provide the following information.

      - Determine whether or not to select **Specify separate directories to store your AS2 messages, MDN files, and JSON status files**.
        - If you select this option, you specify separate locations for payload files, failed files, MDN files, status files, and temporary files.
        - If you clear this option, all AS2 files go into the location that you specify for your base directory.
      - For **S3 Bucket**, choose an Amazon S3 bucket.
      - For **Prefix**, you can enter a prefix (folder) to use for storing files in the bucket.

For example, if you enter **amzn-s3-demo-bucket** for your bucket and **incoming** for your prefix, your AS2 files are saved to the /**amzn-s3-demo-bucket**/incoming folder.

- For **AWS IAM Role**, choose a role that can access the bucket you specified.
- For **Preserve filename**, choose whether to preserve original filenames for incoming AS2 message payloads.
  - If you select this setting, the filename provided by your trading parter is preserved when the file is saved in Amazon S3.
  - If you clear this setting, when Transfer Family saves the file, the filename is adjusted, as described in File names and locations.

d. (Optional) Add tags in the **Tags** section.

e. After you have entered all the information for the agreement, choose **Create agreement**.

The new agreement appears in the **Agreements** section of the server details page.

Configure AS2 connectors

The purpose of a connector is to establish a relationship between trading partners for *outbound* transfers—sending AS2 files from a Transfer Family server to an external, partner-owned destination. For the connector, you specify the local party, the remote partner, and their certificates (by creating local and partner profiles).

After you have a connector in place, you can transfer information to your trading partners. Each AS2 server is assigned three static IP addresses. AS2 connectors use these IP addresses for sending asynchronous MDNs to your trading partners over AS2.

> **Note**
>
> The message size received by a trading partner will not match the object size in Amazon S3. This discrepancy occurs because the AS2 message wraps the file in an envelope prior to sending. So, the file size might increase, even if the file is sent with compression. Therefore, make sure that the trading partner's maximum file size is greater than the size of the file that you are sending.

1. Import AS2 certificates

2. Create AS2 profiles
3. Create an AS2 server
4. Create an AS2 agreement
5. Create an AS2 connector

Create an AS2 connector

This procedure explains how to create AS2 connectors by using the AWS Transfer Family console. If you want to use the AWS CLI instead, see the section called "Step 6: Create a connector between you and your partner".

To create an AS2 connector

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Connectors to send messages**, from the **AS2 Trading Partners** menu, and then choose **Create AS2 connector**.
3. In the **Connector configuration** section, specify the following information:

   - **URL** – Enter the URL for outbound connections.
   - **IP addressing mode** – Choose **IPv4 only** (default) to use IPv4 addresses exclusively, or **Dual-stack** to support both IPv6 and IPv4. In dual-stack mode, the connector prefers IPv6 when DNS resolution returns IPv6 results, and uses IPv4 when only IPv4 DNS results are returned.
   - **Access role** – Choose the Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role to use. Make sure that this role provides read and write access to the parent directory of the file location that's used in the `StartFileTransfer` request. Additionally, make sure that the role provides read and write access to the parent directory of the files that you intend to send with `StartFileTransfer`.

     > **Note**
     >
     > If you're using Basic authentication for your connector, the access role requires the `secretsmanager:GetSecretValue` permission for the secret. If the secret is encrypted by using a customer managed key instead of the AWS managed key in AWS Secrets Manager, then the role also needs the `kms:Decrypt` permission for that key. If you name your secret with the prefix `aws/transfer/`, you can add the

necessary permission with a wildcard character (*), as shown in Example permission to create secrets.

- **Logging role** (optional) – Choose the IAM role for the connector to use to push events to your CloudWatch logs.

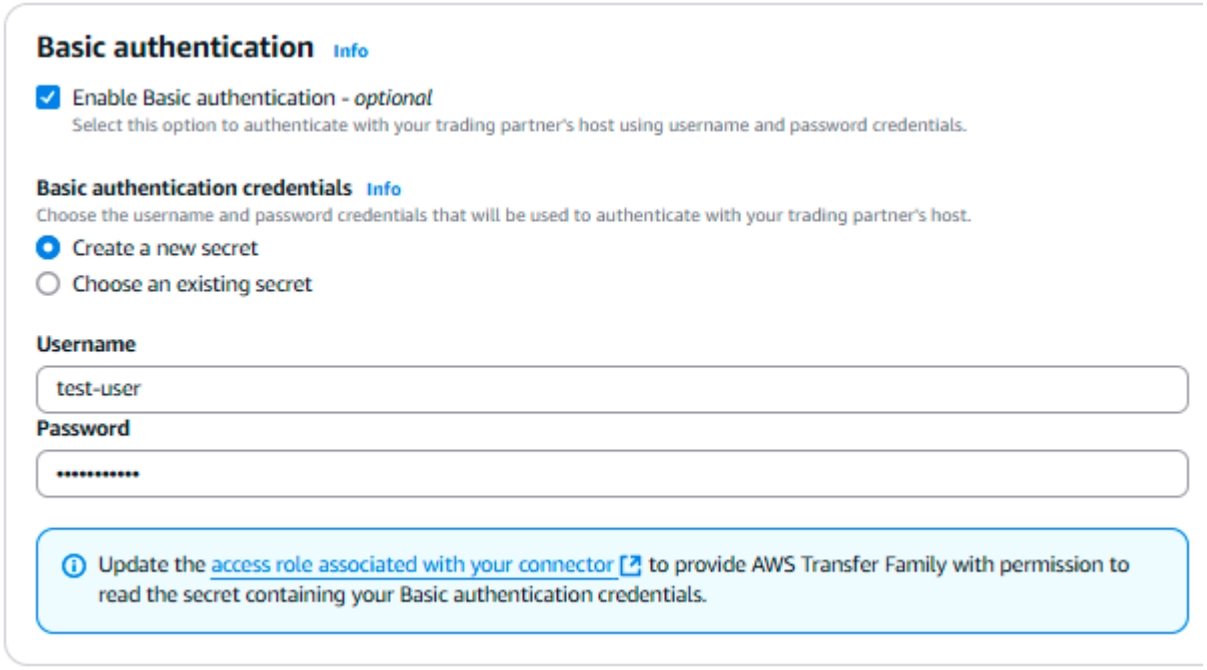
4. In the **AS2 configuration** section, choose the local and partner profiles, the encryption and signing algorithms, and whether to compress the transferred information. Note the following:

   - The **Preserve S3 Content-Type** parameter is enabled by default.

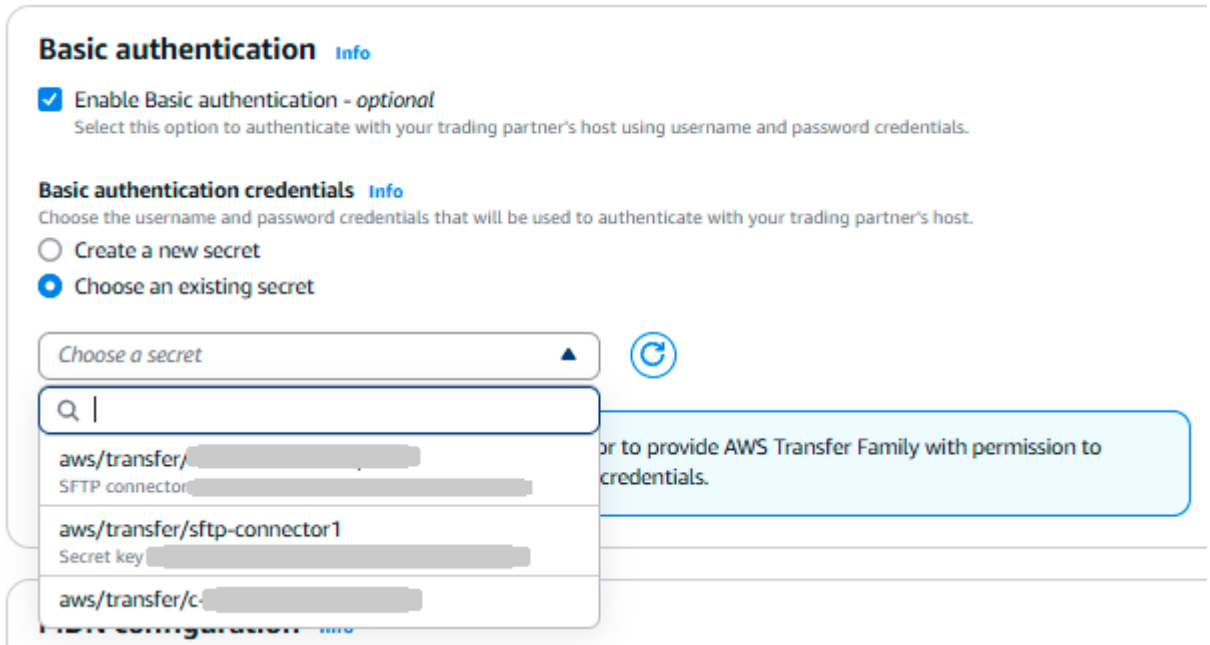
     When set, Transfer Family uses the Amazon S3 `Content-Type` that is associated with objects in S3 instead of having the content type mapped based on the file extension. Clear this setting if you want the service to map content type for your AS2 messages based on file extension, rather than using the content type from the S3 object.
   - For the encryption algorithm, do not choose `DES_EDE3_CBC` unless you must support a legacy client that requires it, as it is a weak encryption algorithm.
   - The **Subject** is used as the `subject` HTTP header attribute in AS2 messages that are being sent with the connector.
   - If you choose to create a connector without an encryption algorithm, you must specify `HTTPS` as your protocol.

5. In the **Basic authentication** section, specify the following information.

   - To send sign-on credentials along with outbound messages, select **Enable Basic authentication**. If you don't want to send any credentials with outbound messages, keep **Enable Basic authentication** cleared.
   - If you're using authentication, choose or create a secret.
     - To create a new secret, choose **Create a new secret** and then enter a username and password. These credentials must match the user that connects to the partner's endpoint.



- To use an existing secret, choose **Choose an existing secret**, and then choose a secret from the dropdown menu. For the details of creating a correctly formatted secret in Secrets Manager, see Enable Basic authentication for AS2 connectors.



6. In the **MDN configuration** section, specify the following information:

- **Request MDN** – You have the option to require your trading partner to send you an MDN after they have successfully received your message over AS2.

- **Signed MDN** – You have the option to require that MDNs be signed. This option is available only if you have selected **Request MDN**.

7. After you've confirmed all of your settings, choose **Create AS2 connector** to create the connector.

The **Connectors** page appears, with the ID of your new connector added to the list. To view the details for your connectors, see View AS2 connector details.

AS2 connector algorithms

When you create an AS2 connector, the following security algorithms are attached to the connector.

Type	Algorithm
TLS Cipher	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

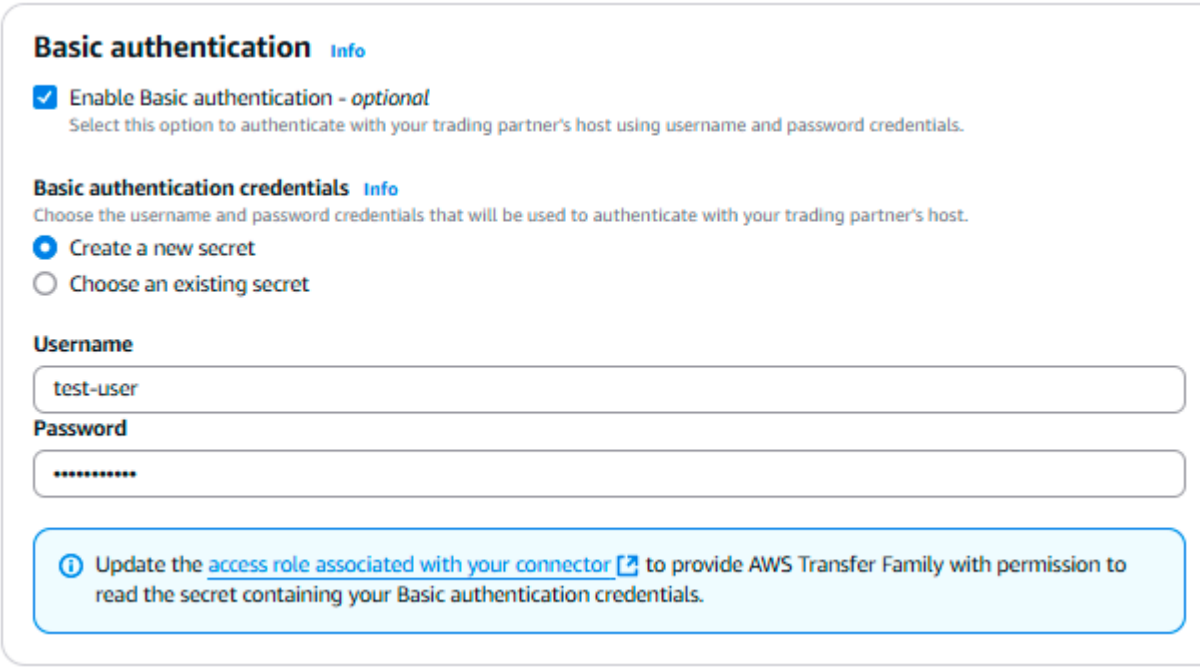
Basic authentication for AS2 connectors

When you create or update a Transfer Family server that uses the AS2 protocol, you can add Basic authentication for outbound messages. You do this by adding authentication information to a connector.

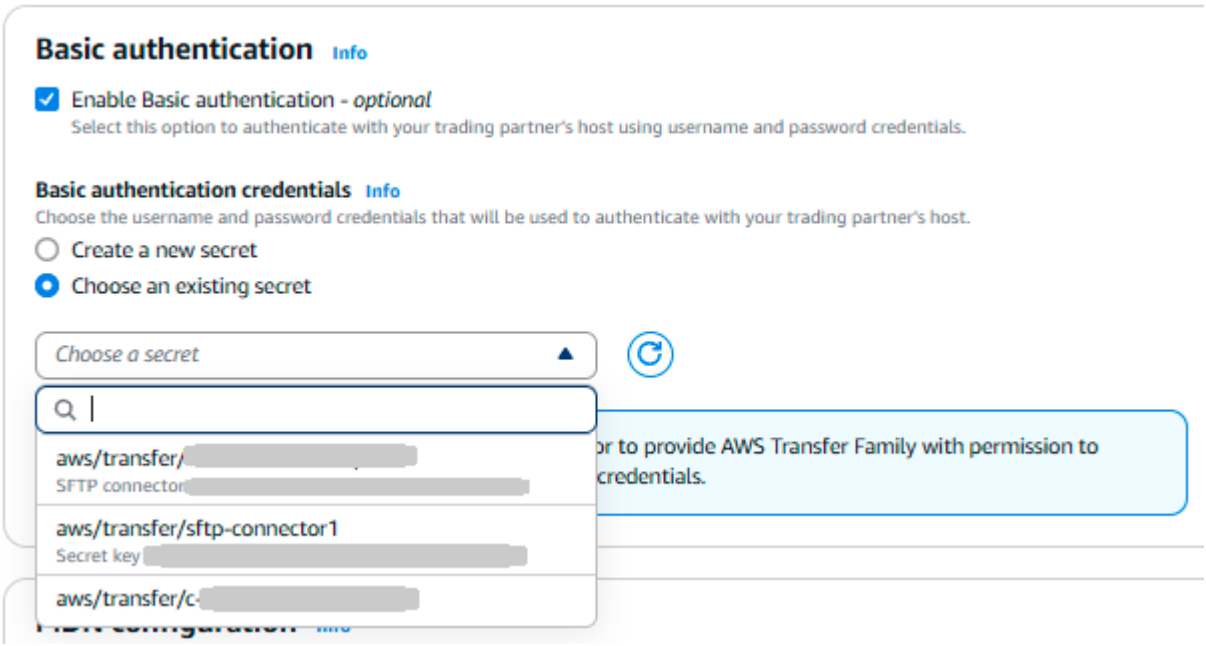
> **Note**
>
> Basic authentication is available only if you're using HTTPS.

To use authentication for your connector, select **Enable Basic authentication** in the **Basic authentication** section. After you enable Basic authentication, you can choose to create a new secret, or use an existing one. In either case, the credentials in the secret are sent with outbound messages that use this connector. The credentials must match the user that is attempting to connect to the trading partner's remote endpoint.

The following screenshot shows **Enable Basic authentication** selected, and **Create a new secret** chosen. After making these choices, you can enter a username and password for the secret.



The following screenshot shows **Enable Basic authentication** selected, and **Choose an existing secret** chosen. Your secret must be in the correct format, as described in Enable Basic authentication for AS2 connectors.



Enable Basic authentication for AS2 connectors

When you enable Basic authentication for AS2 connectors, you can either create a new secret in the Transfer Family console, or you can use a secret that you create in AWS Secrets Manager. In either case, your secret is stored in Secrets Manager.

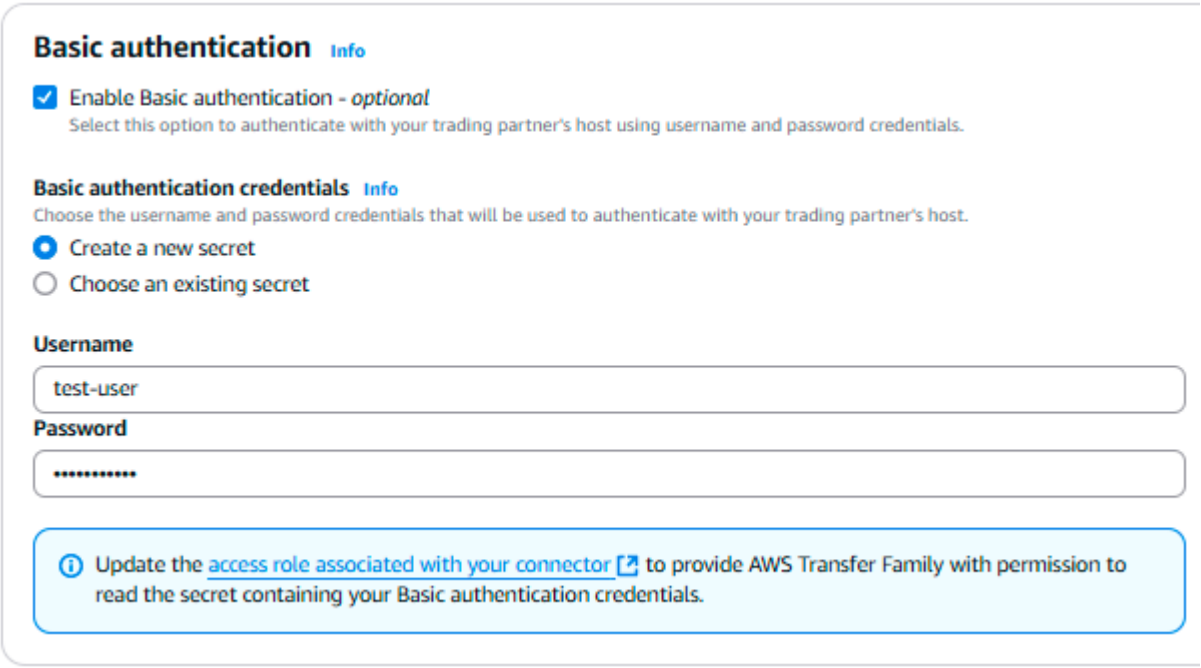
Topics

- Create a new secret in the console
- Use an existing secret
- Create a secret in AWS Secrets Manager

Create a new secret in the console

When you're creating a connector in the console, you can create a new secret.

To create a new secret, choose **Create a new secret** and then enter a username and password. These credentials must match the user that connects to the partner's endpoint.



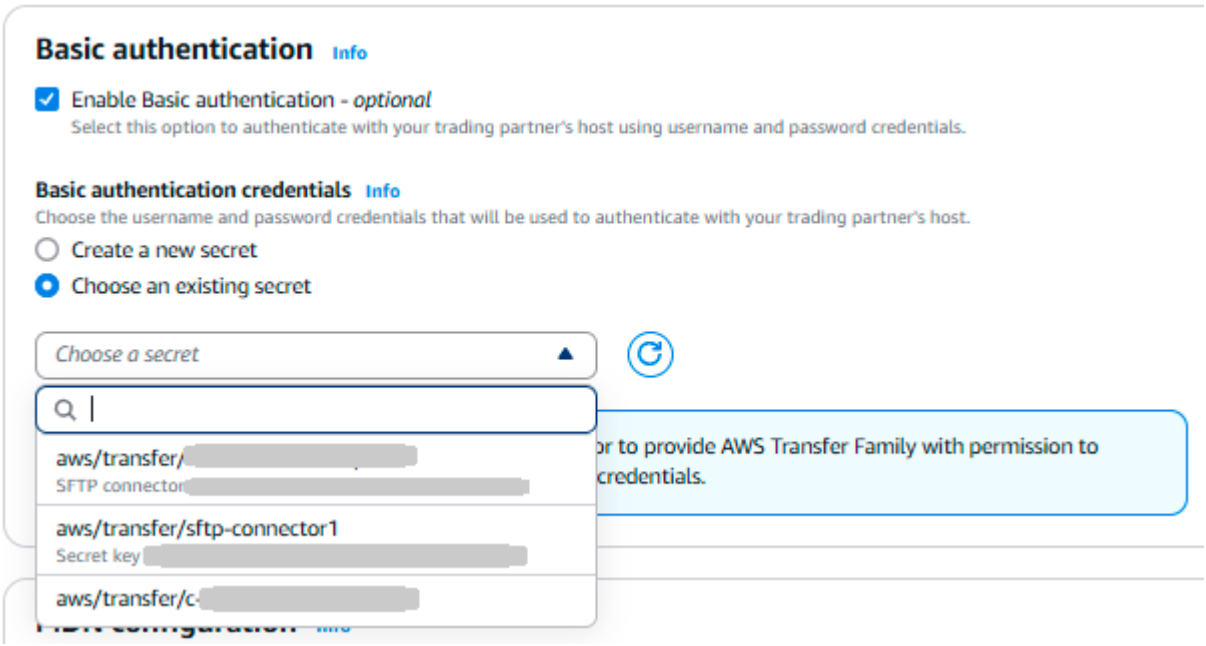
Note

When you create a new secret in the console, the name of the secret follows this naming convention: **`/aws/transfer/`*`connector-id`***, where *`connector-id`* is the ID of the connector that you're creating. Consider this when you are trying to locate the secret in AWS Secrets Manager.

Use an existing secret

When you're creating a connector in the console, you can specify an existing secret.

To use an existing secret, choose **Choose an existing secret**, and then choose a secret from the dropdown menu. For the details of creating a correctly formatted secret in Secrets Manager, see Create a secret in AWS Secrets Manager.



Create a secret in AWS Secrets Manager

The following procedure describes how to create an appropriate secret for use with your AS2 connector.

> **Note**
>
> Basic authentication is available only if you're using HTTPS.

To store user credentials in Secrets Manager for AS2 Basic authentication

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at https://console.aws.amazon.com/secretsmanager/.
2. In the left navigation pane, choose **Secrets**.
3. On the **Secrets** page, choose **Store a new secret**.
4. On the **Choose secret type** page, for **Secret type**, choose **Other type of secret**.
5. In the **Key/value pairs** section, choose the **Key/value** tab.
   - **Key** – Enter **`Username`**.
   - **value** – Enter the name of the user that is authorized to connect to the partner' server.

6. If you want to provide a password, choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

   Choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

   - **Key** – Enter **`Password`**.
   - **value** – Enter the password for the user.
7. If you want to provide a private key, choose **Add row**, and in the **Key/value pairs** section, choose the **Key/value** tab.

   - **Key** – Enter **`PrivateKey`**.
   - **value** – Enter a private key for the user. This value must be stored in OpenSSH format, and must correspond to the public key that is stored for this user in the remote server.
8. Choose **Next**.
9. On the **Configure secret** page, enter a name and description for your secret. We recommend that you use a prefix of **`aws/transfer/`** for the name. For example, you could name your secret **`aws/transfer/connector-1`**.
10. Choose **Next**, and then accept the defaults on the **Configure rotation** page. Then choose **Next**.
11. On the **Review** page, choose **Store** to create and store the secret.

After you create the secret, you can choose it when you are creating a connector (see Configure AS2 connectors). In the step where you enable Basic authentication, choose the secret from the dropdown list of available secrets.

View AS2 connector details

You can find a list of details and properties for an AS2 AWS Transfer Family connector in the AWS Transfer Family console. An AS2 connector's properties include its URL, roles, profiles, MDNs, tags, and monitoring metrics.

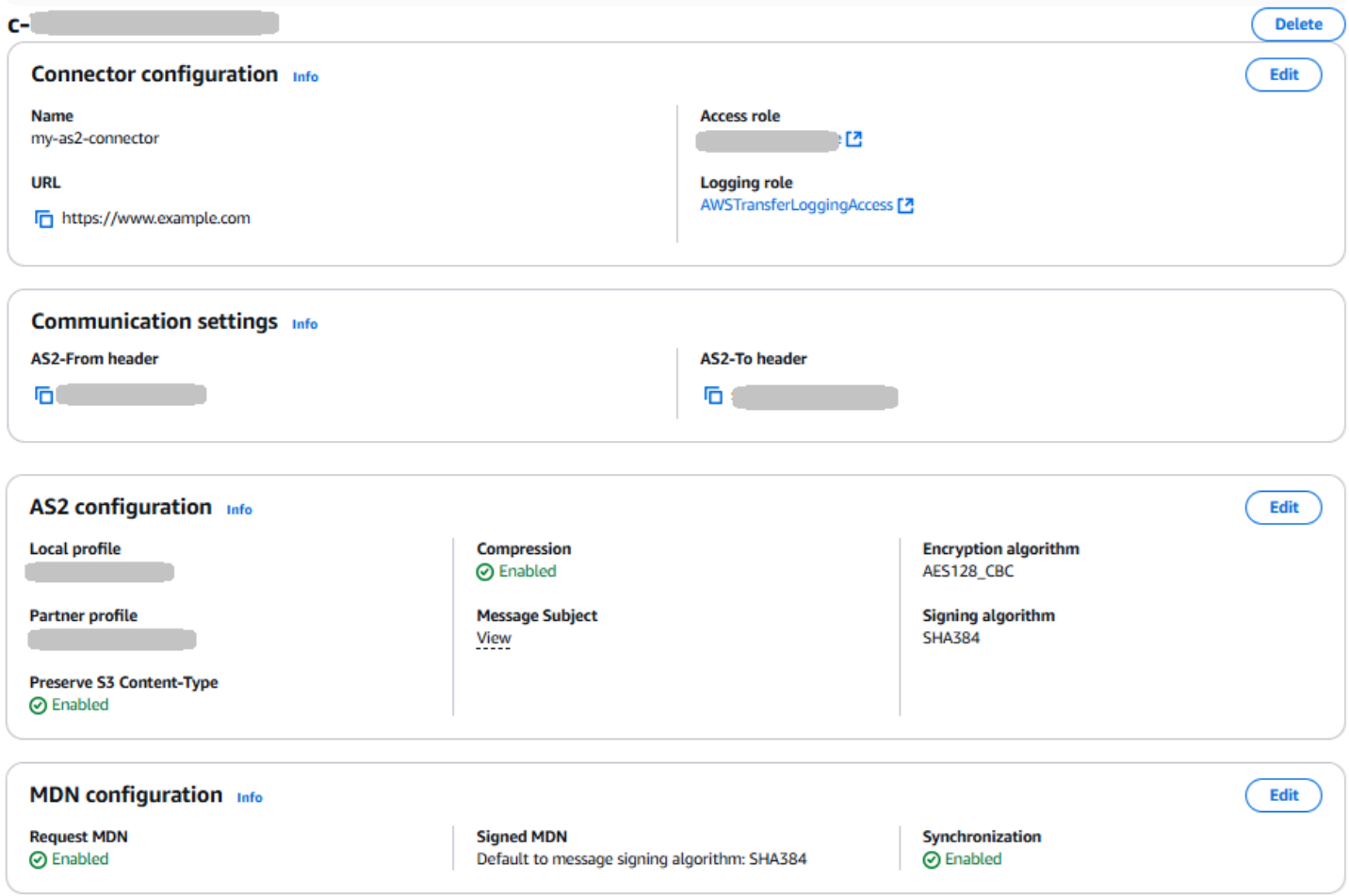
This is the procedure for viewing connector details.

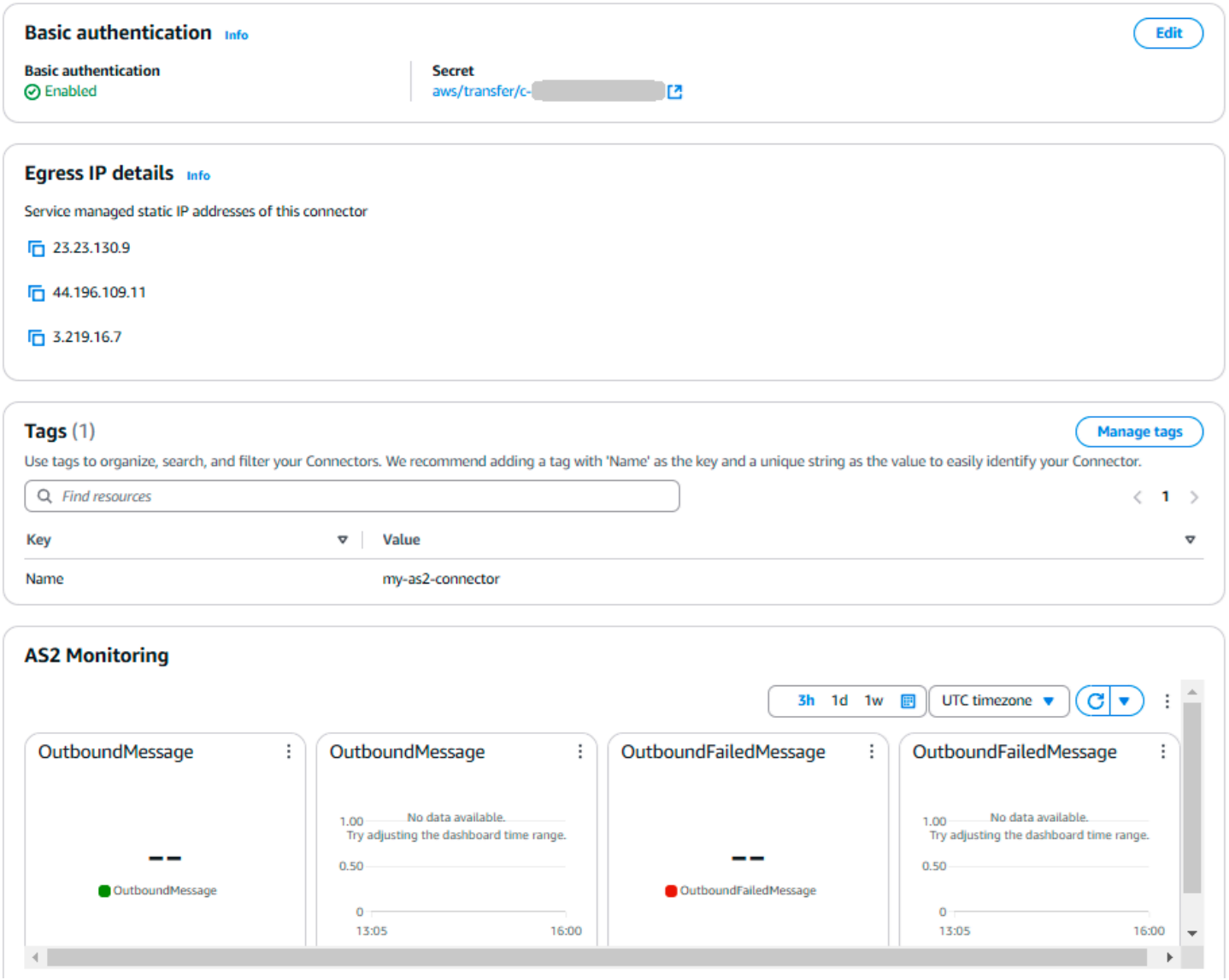
To view connector details

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Connectors**.

3. Choose the identifier in the **Connector ID** column to see the details page for the selected connector.

You can change the properties for the AS2 connector on the connector's details page by choosing **Edit**.





Note

You can get much of this information, albeit in a different format, by running the following AWS Command Line Interface (AWS CLI command:

```
aws transfer describe-connector --connector-id your-connector-id
```

For more information, see DescribeConnector in the API reference.

Sending and receiving AS2 messages

This section describes the processes for sending and receiving AS2 messages. It also provide details on filenames and locations associated with AS2 messages.

The following table lists the available encryption algorithms for AS2 messages, and when you can use them.

Encryption algorithm	HTTP	HTTPS	Notes
AES128_CBC	Yes	Yes	
AES192_CBC	Yes	Yes	
AES256_CBC	Yes	Yes	
DES_EDE3_CBC	Yes	Yes	Only use this algorithm if you must support a legacy client that requires it, as it is a weak encryption algorithm.
NONE	No	Yes	If you are sending messages to a Transfer Family server, you can only select `NONE` if you are using an Application Load Balancer (ALB).

Topics

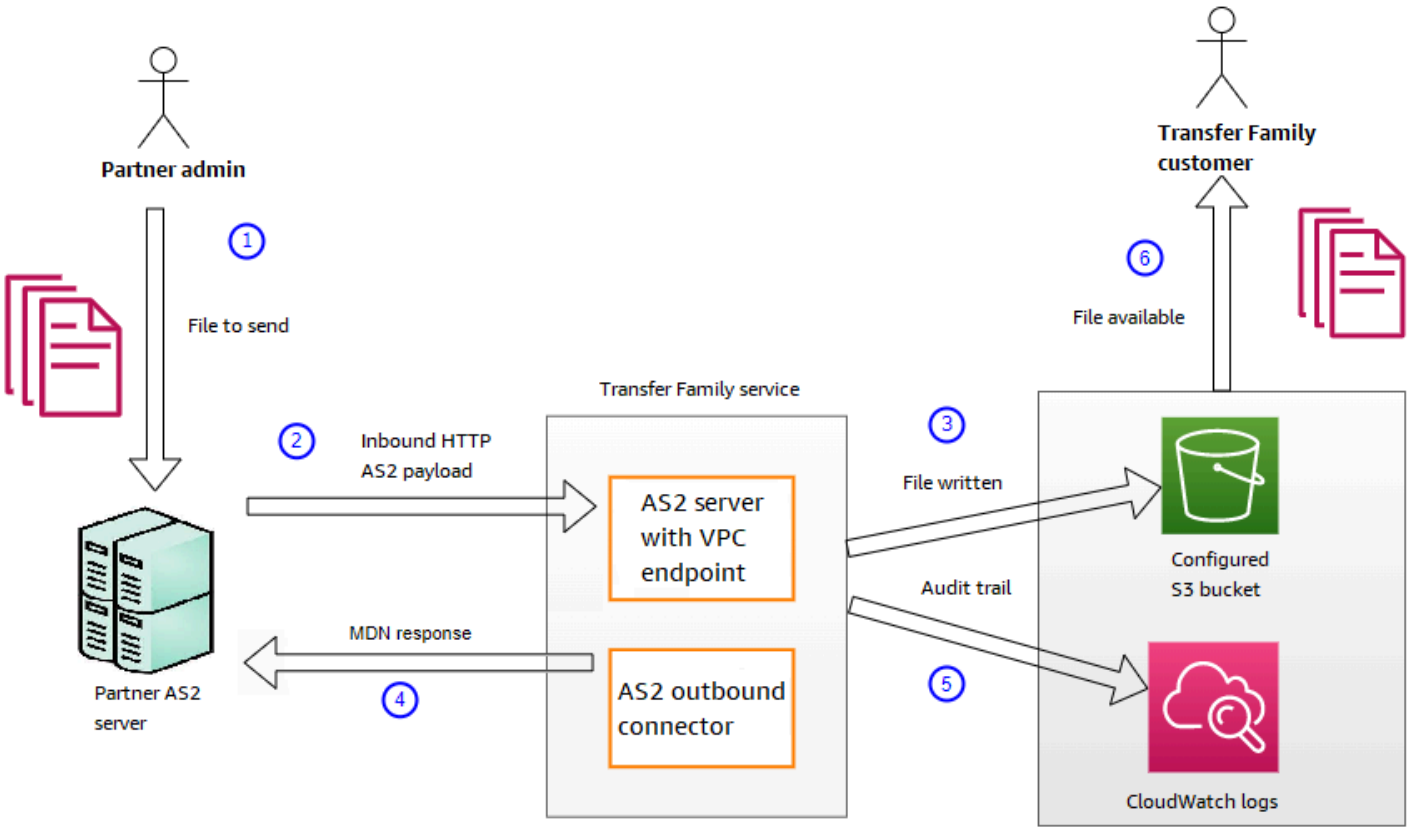
- Receive AS2 message process
- Sending and receiving AS2 messages over HTTPS
- Transferring files by using an AS2 connector
- File names and locations

- Status codes
- Sample JSON files

Receive AS2 message process

The inbound process is defined as a message or file that's being transferred to your AWS Transfer Family server. The sequence for inbound messages is as follows:

1. An admin or automated process starts an AS2 file transfer on the partner's remote AS2 server.
2. The partner's remote AS2 server signs and encrypts the file contents, then sends an HTTP POST request to an AS2 inbound endpoint hosted on Transfer Family.
3. Using the configured values for the server, partners, certificates, and agreement, Transfer Family decrypts and verifies the AS2 payload. The file contents are stored in the configured Amazon S3 file store.
4. The signed MDN response is returned either inline with the HTTP response, or asynchronously through a separate HTTP POST request back to the originating server.
5. An audit trail is written to Amazon CloudWatch with details about the exchange.
6. The decrypted file is available in a folder named `inbox/processed`.



Sending and receiving AS2 messages over HTTPS

This section describes how to configure a Transfer Family server that uses the AS2 protocol to send and receive messages over HTTPS.

Topics

- Send AS2 messages over HTTPS
- Receive AS2 messages over HTTPS

Send AS2 messages over HTTPS

To send AS2 messages using HTTPS, create a connector with the following information:

- For the URL, specify an HTTPS URL
- For the encryption algorithm, select any of the available algorithms.

> **Note**
>
> To send messages to a Transfer Family server while not using encryption (that is, you select `NONE` for the encryption algorithm), you must use an Application Load Balancer (ALB).

- Provide the remaining values for the connector as described in Configure AS2 connectors.

Receive AS2 messages over HTTPS

AWS Transfer Family AS2 servers currently only provide HTTP transport over port 5080. However, you can terminate TLS on a network or application load balancer in front of your Transfer Family server VPC endpoint by using a port and certificate of your choosing. With this approach, you can have incoming AS2 messages use HTTPS.

Prerequisites

- The VPC must be in the same AWS Region as your Transfer Family server.
- The subnets of your VPC must be within the Availability Zones that you want to use your server in.

> **Note**
>
> Each Transfer Family server can support up to three Availability Zones.

- Allocate up to three Elastic IP addresses in the same Region as your server. Or, you can choose to bring your own IP address range (BYOIP).

> **Note**
>
> The number of Elastic IP addresses must match the number of Availability Zones that you use with your server endpoints.

You can configure either a Network Load Balance (NLB) or an Application Load Balancer (ALB). The following table lists the pros and cons for each approach.

The table below provides the differences in capabilities when you use an NLB versus an ALB to terminate TLS.

Feature	Network Load Balancer (NLB)	Application Load Balancer (ALB)
Latency	Lower latency as it operates at the network layer.	Higher latency as it operates at the application layer.
Static IP support	Can attach Elastic IP addresses that can be static.	Cannot attach Elastic IP addresses: provides a domain whose underlying IP addresses can change.
Advanced routing	Doesn't support advanced routing.	Supports advanced routing. Can inject `X-Forwarded-Proto` header required for AS2 without encryption. This header is described in X-Forwarded-Proto on the developer.mozilla.org website.
TLS/SSL termination	Supports TLS/SSL terminati on	Supports TLS/SSL terminati on
Mutual TLS (mTLS)	Transfer Family doesn't currently support using an NLB for mTLS	Support for mTLS

Configure NLB

This procedure describes how to set up an internet-facing Network Load Balancer (NLB) in your VPC.

To create a Network Load Balancer and define the VPC endpoint of the server as the load balancer's target

1. Open the Amazon Elastic Compute Cloud console at https://console.aws.amazon.com/ec2/.
2. From the navigation pane, choose **Load Balancers**, and then choose **Create load balancer**.
3. Under **Network Load Balancer**, choose **Create**.
4. In the **Basic configuration** section, enter the following information:
   - For **Name**, enter a descriptive name for the load balancer.
   - For **Scheme**, choose **Internet-facing**.
   - For **IP address type**, choose **IPv4**.
5. In the **Network mapping** section, enter the following information:
   - For **VPC**, choose the virtual private cloud (VPC) that you created.
   - Under **Mappings**, choose the Availability Zones associated with the public subnets that are available in the same VPC that you use with your server endpoints.
   - For the **IPv4 address** of each subnet, choose one of the Elastic IP addresses that you allocated.
6. In the **Listeners and routing** section, enter the following information:
   - For **Protocol**, choose **TLS**.
   - For **Port**, enter **5080**.
   - For **Default action**, choose **Create target group**. For the details of creating a new target group, see To create a target group.

   After you create a target group, enter its name in the **Default action** field.
7. In the **Secure listener settings** section, choose your certificate in the **Default SSL/TLS certificate** area.
8. Choose **Create load balancer** to create your NLB.
9. (Optional, but recommended) Turn on access logs for the Network Load Balancer to maintain a full audit trail, as described in Access logs for your Network Load Balancer.

We recommend this step because the TLS connection is terminated at the NLB. Therefore, the source IP address that's reflected in your Transfer Family AS2 CloudWatch log groups is the NLB's private IP address, instead of your trading partner's external IP address.

Configure ALB

This procedure describes how to set up an Application Load Balancer (ALB) in your VPC.

To create an Application Load Balancer and define the VPC endpoint of the server as the load balancer's target

1. Open the Amazon Elastic Compute Cloud console at https://console.aws.amazon.com/ec2/.
2. From the navigation pane, choose **Load Balancers**, and then choose **Create load balancer**.
3. Under **Application Load Balancer**, choose **Create**.
4. In the ALB console, create a new HTTP listener on port 443 (HTTPS).
5. (Optional). If you want to set up mutual authentication (mTLS), configure security settings and a trust store.
    a. Attach your SSL/TLS certificate to the listener.
    b. Under **Client certificate handling**, select **Mutual authentication (mTLS)**.
    c. Choose **Verify with trust store**.
    d. Under **Advanced mTLS settings**, choose or create a trust store by uploading your CA certificates.
6. Create a new target group and add the private IP addresses of your Transfer Family AS2 server endpoints as targets on port 5080. For the details of creating a new target group, see To create a target group.
7. Configure health checks for the target group to use the HTTP protocol on port 5080.
8. Create a new rule to forward HTTPS traffic from the listener to the target group.
9. Configure the listener to use your SSL/TLS certificate.

After you set up the load balancer, clients communicate with the load balancer over the custom port listener. Then, the load balancer communicates with the server over port 5080.

To create a target group

1. After you choose **Create target group** in the previous procedure, you are taken to the **Specify group details** page for a new target group.
2. In the **Basic configuration** section, enter the following information.
   - For **Choose a target type**, choose **IP addresses**.
   - For **Target group name**, enter a name for the target group.
   - For **Protocol**, your selection is dependent upon whether you are using an ALB or an NLB.
     - For a Network Load Balancer (NLB), choose **TCP**
     - For an Application Load Balancer (ALB), choose **HTTP**
   - For **Port**, enter **5080**.
   - For **IP address** type, choose **IPv4**.
   - For **VPC**, choose the VPC that you created for your Transfer Family AS2 server.
3. In the **Health checks** section, choose the **Health check protocol**.
   - For an ALB, choose **HTTP**
   - For an NLB, choose **TCP**
4. Choose **Next**.
5. On the **Register targets** page, enter the following information:
   - For **Network**, confirm that the VPC that you created for your Transfer Family AS2 server is specified.
   - For **IPv4 address**, enter the private IPv4 address of your Transfer Family AS2 server's endpoints.

     If you have more than one endpoint for your server, choose **Add IPv4 address** to add another row for entering another IPv4 address. Repeat this process until you've entered the private IP addresses for all of your server's endpoints.
   - Make sure that **Ports** is set to **5080**.
   - Choose **Include as pending below** to add your entries to the **Review targets** section.
6. In the **Review targets** section, review your IP targets.
7. Choose **Create target group**, then go back to the previous procedure for creating your NLB and enter the new target group where indicated.

Test access to the server from an Elastic IP address

Connect to the server over the custom port by using an Elastic IP address or the DNS name of the Network Load Balancer.

> **Important**
>
> Manage access to your server from client IP addresses by using the network access control lists (network ACLs) for the subnets configured on the load balancer. Network ACL permissions are set at the subnet level, so the rules apply to all resources that are using the subnet. You can't control access from client IP addresses by using security groups, because the load balancer's target type is set to **IP addresses** instead of **Instances**. Therefore, the load balancer doesn't preserve source IP addresses. If the Network Load Balancer's health checks fail, this means that the load balancer can't connect to the server endpoint. To troubleshoot this issue, check the following:
>
> - Confirm that the server endpoint's associated security group allows inbound connections from the subnets that are configured on the load balancer. The load balancer must be able to connect to the server endpoint over port 5080.
> - Confirm that the server's **State** is **Online**.

Transferring files by using an AS2 connector

AS2 connectors establish a relationship between trading partners for transfers of AS2 messages from a Transfer Family server to an external, partner-owned destination.

You can use Transfer Family to send AS2 messages by referencing the connector ID and the paths to the files, as illustrated in the following `start-file-transfer` AWS Command Line Interface (AWS CLI) command:

```
aws transfer start-file-transfer --connector-id c-1234567890abcdef0 \
--send-file-paths "/amzn-s3-demo-source-bucket/myfile1.txt" "/amzn-s3-demo-source-bucket/myfile2.txt"
```

To get the details for your connectors, run the following command:

```
aws transfer list-connectors
```

The `list-connectors` command returns the connector IDs, URLs, and Amazon Resource Names (ARNs) for your connectors.

To return the properties of a particular connector, run the following command with the ID that you want to use:

```
aws transfer describe-connector --connector-id your-connector-id
```

The `describe-connector` command returns all of the properties for the connector, including its URL, roles, profiles, Message Disposition Notices (MDNs), tags, and monitoring metrics.

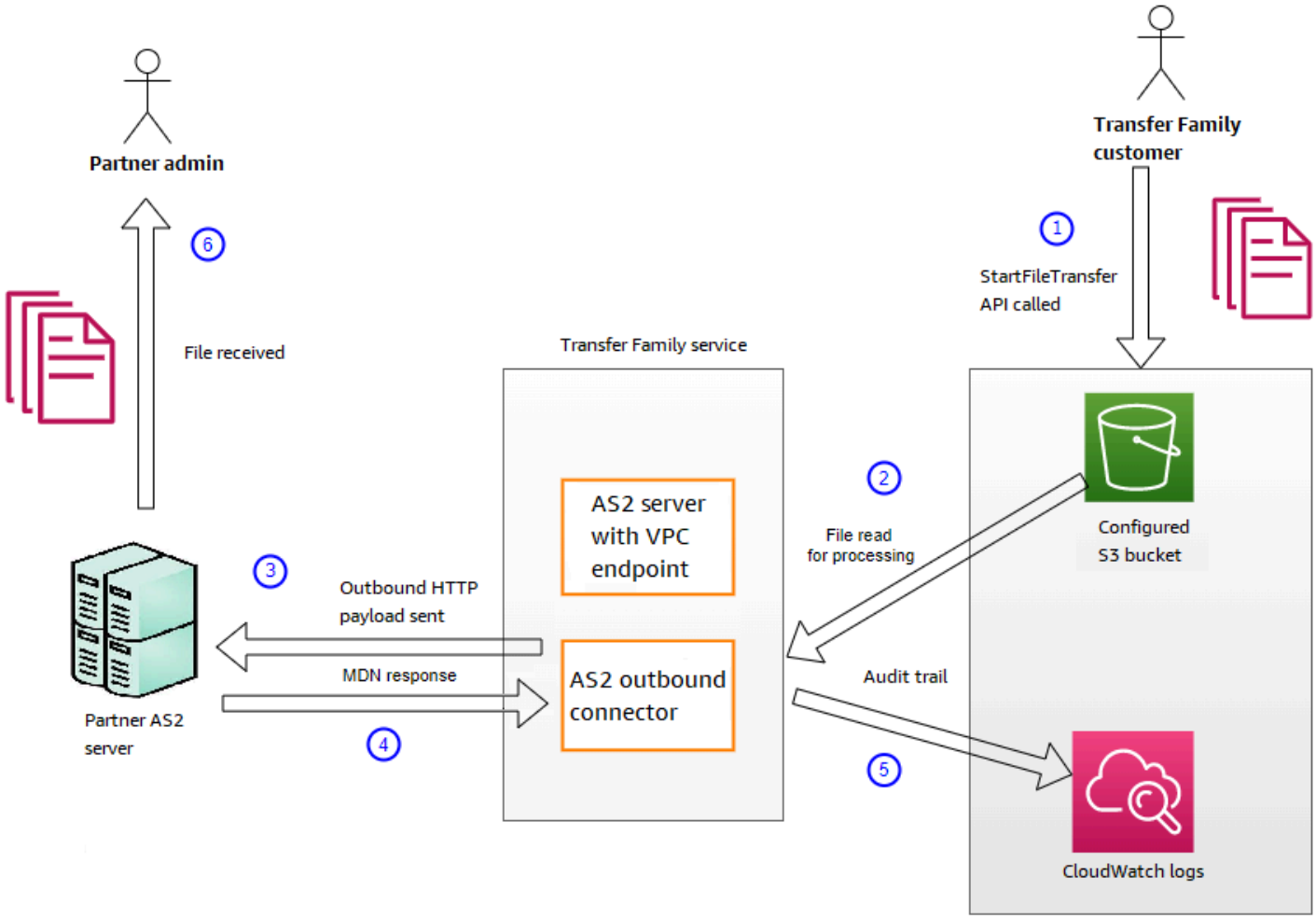
You can confirm that the partner successfully received the files by viewing the JSON and MDN files. These files are named according to the conventions described in File names and locations. If you configured a logging role when you created the connector, you can also check your CloudWatch logs for the status of AS2 messages.

To view AS2 connector details, see View AS2 connector details. For more information about creating AS2 connectors, see Configure AS2 connectors.

To send an AS2 outbound message

The outbound process is defined as a message or file being sent from AWS to an external client or service. The sequence for outbound messages is as follows:

1. An admin calls the `start-file-transfer` AWS Command Line Interface (AWS CLI) command or the `StartFileTransfer` API operation. This operation references a `connector` configuration.
2. Transfer Family detects a new file request and locates the file. The file is compressed, signed, and encrypted.
3. A transfer HTTP client performs an HTTP POST request to transmit the payload to the partner's AS2 server.
4. The process returns the signed MDN response, either inline with the HTTP response (synchronous or asynchronous MDN).
5. As the file moves between different stages of transmission, the process delivers the MDN response receipt and processing details to the customer.
6. The remote AS2 server makes the decrypted and verified file available to the partner admin.



AS2 processing supports many of the RFC 4130 protocols, with a focus on common use cases and integration with existing AS2-enabled server implementations. For details of the supported configurations, see AS2 configurations.

File names and locations

This section discusses the file-naming conventions for AS2 transfers.

For inbound file transfers, note the following:

- You specify the base directory in an agreement. The base directory is the Amazon S3 bucket name combined with a prefix, if any. For example, `/amzn-s3-demo-bucket/AS2-folder`.
- If an incoming file is processed successfully, the file (and the corresponding JSON file) is saved to the `/processed` folder. For example, `/amzn-s3-demo-bucket/AS2-folder/processed`.

  The JSON file contains the following fields:

- agreement-id
- as2-from
- as2-to
- as2-message-id
- transfer-id
- client-ip
- connector-id
- failure-message
- file-path
- message-subject
- mdn-message-id
- mdn-subject
- requester-file-name
- requester-content-type
- server-id
- status-code
- failure-code
- transfer-size

- If an incoming file cannot be processed successfully, the file (and the corresponding JSON file) is saved to the /failed folder. For example, /amzn-s3-demo-bucket/AS2-folder/failed.
- The transferred file is stored in the processed folder as *original_filename*.*messageId*.*original_extension*. That is, the message ID for the transfer is appended to the name of the file, before its original extension.
- A JSON file is created and saved as *original_filename*.*messageId*.*original_extension*.json. In addition to the message ID being added, the string .json is appended to the transferred file's name.
- A Message Disposition Notice (MDN) file is created and saved as *original_filename*.*messageId*.*original_extension*.mdn. In addition to the message ID being added, the string .mdn is appended to the transferred file's name.
- If there is an inbound file named ExampleFileInS3Payload.dat, the following files are created:

- **File** – ExampleFileInS3Payload.c4d6b6c7-23ea-4b8c-9ada-0cb811dc8b35@44313c54b0a46a36.
- **JSON** – ExampleFileInS3Payload.c4d6b6c7-23ea-4b8c-9ada-0cb811dc8b35@44313c54b0a46a36.
- **MDN** – ExampleFileInS3Payload.c4d6b6c7-23ea-4b8c-9ada-0cb811dc8b35@44313c54b0a46a36.

For outbound transfers, the naming is similar, with the difference that there is no incoming message file, and also, the transfer ID for the transferred message is added to the file name. The transfer ID is returned by the `StartFileTransfer` API operation (or when another process or script calls this operation).

- The `transfer-id` is an identifier that is associated with a file transfer. All requests that are part of a `StartFileTransfer` call share a `transfer-id`.
- The base directory is the same as the path that you use for the source file. That is, the base directory is the path that you specify in the `StartFileTransfer` API operation or `start-file-transfer` AWS CLI command. For example:

  ```
  aws transfer start-file-transfer --send-file-paths /amzn-s3-demo-bucket/AS2-folder/file-to-send.txt
  ```

  If you run this command, MDN and JSON files are saved in `/amzn-s3-demo-bucket/AS2-folder/processed` (for successful transfers), or `/amzn-s3-demo-bucket/AS2-folder/failed` (for unsuccessful transfers).
- A JSON file is created and saved as *original_filename*.*transferId*.*messageId*.*original_extension*.json.
- An MDN file is created and saved as *original_filename*.*transferId*.*messageId*.*original_extension*.mdn.
- If there is an outbound file named `ExampleFileOutTestOutboundSyncMdn.dat`, the following files are created:
  - **JSON** – ExampleFileOutTestOutboundSyncMdn.dedf4601-4e90-4043-b16b-579af35e0d83.fbe18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa.dat.j
  - **MDN** – ExampleFileOutTestOutboundSyncMdn.dedf4601-4e90-4043-b16b-579af35e0d83.fbe18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa.dat.m

You can also check the CloudWatch logs to view the details of your transfers, including any that failed.

Status codes

The following table lists all of the status codes that can be logged to CloudWatch logs when you or your partner send an AS2 message. Different message processing steps apply to different message types and are intended for monitoring only. The COMPLETED and FAILED states represent the final step in processing, and are visible in JSON files.

Code	Description	Processing completed?
PROCESSING	The message is in the process of being converted to its final format. For example, decompression and decryptio n steps both have this status.	No
MDN_TRANSMIT	Message processing is sending an MDN response.	No
MDN_RECEIVE	Message processing is receiving an MDN response.	No
COMPLETED	Message processing has completed successfully. This state includes when an MDN is sent for an inbound message or for MDN verificat ion of outbound messages.	Yes
FAILED	The message processing has failed. For a list of error codes, see AS2 error codes.	Yes

Sample JSON files

This section lists sample JSON files for both inbound and outbound transfers, including sample files for successful transfers and transfers that fail.

Sample outbound file that is successfully transferred:


```
{
  "requester-content-type": "application/octet-stream",
  "message-subject": "File xyzTest from MyCompany_OID to partner YourCompany",
  "requester-file-name": "TestOutboundSyncMdn-9lmCr79hV.dat",
  "as2-from": "MyCompany_OID",
  "connector-id": "c-c21c63ceaaf34d99b",
  "status-code": "COMPLETED",
  "disposition": "automatic-action/MDN-sent-automatically; processed",
  "transfer-size": 3198,
  "mdn-message-id": "OPENAS2-11072022063009+0000-df865189-1450-435b-9b8d-
d8bc0cee97fd@PartnerA_OID_MyCompany_OID",
  "mdn-subject": "Message be18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa has been
 accepted",
  "as2-to": "PartnerA_OID",
  "transfer-id": "dedf4601-4e90-4043-b16b-579af35e0d83",
  "file-path": "/amzn-s3-demo-bucket/as2testcell0000/openAs2/
TestOutboundSyncMdn-9lmCr79hV.dat",
  "as2-message-id": "fbe18db8-7361-42ff-8ab6-49ec1e435f34@c9c705f0baaaabaa",
  "timestamp": "2022-07-11T06:30:10.791274Z"
}
```


Sample outbound file that is unsuccessfully transferred:


```
{
  "failure-code": "HTTP_ERROR_RESPONSE_FROM_PARTNER",
  "status-code": "FAILED",
  "requester-content-type": "application/octet-stream",
  "subject": "Test run from Id da86e74d6e57464aae1a55b8596bad0a to partner
 9f8474d7714e476e8a46ce8c93a48c6c",
  "transfer-size": 3198,
  "requester-file-name": "openAs2TestOutboundWrongAs2Ids-necco-3VYn5n8wE.dat",
  "as2-message-id": "9a9cc9ab-7893-4cb6-992a-5ed8b90775ff@718de4cec1374598",
  "failure-message": "http://Test123456789.us-east-1.elb.amazonaws.com:10080 returned
 status 500 for message with ID 9a9cc9ab-7893-4cb6-992a-5ed8b90775ff@718de4cec1374598",
  "transfer-id": "07bd3e07-a652-4cc6-9412-73ffdb97ab92",
```

```
  "connector-id": "c-056e15cc851f4b2e9",
  "file-path": "/amzn-s3-demo-bucket-4c1tq6ohjt9y/as2IntegCell0002/openAs2/
openAs2TestOutboundWrongAs2Ids-necco-3VYn5n8wE.dat",
  "timestamp": "2022-07-11T21:17:24.802378Z"
}
```

Sample inbound file that is successfully transferred:

```
{
  "requester-content-type": "application/EDI-X12",
  "subject": "File openAs2TestInboundAsyncMdn-necco-5Ab6bTfCO.dat sent from MyCompany
 to PartnerA",
  "client-ip": "10.0.109.105",
  "requester-file-name": "openAs2TestInboundAsyncMdn-necco-5Ab6bTfCO.dat",
  "as2-from": "MyCompany_OID",
  "status-code": "COMPLETED",
  "disposition": "automatic-action/MDN-sent-automatically; processed",
  "transfer-size": 1050,
  "mdn-subject": "Message Disposition Notification",
  "as2-message-id": "OPENAS2-11072022233606+0000-5dab0452-0ca1-4f9b-b622-
fba84effff3c@MyCompany_OID_PartnerA_OID",
  "as2-to": "PartnerA_OID",
  "agreement-id": "a-f5c5cbea5f7741988",
  "file-path": "processed/openAs2TestInboundAsyncMdn-
necco-5Ab6bTfCO.OPENAS2-11072022233606+0000-5dab0452-0ca1-4f9b-b622-
fba84effff3c@MyCompany_OID_PartnerA_OID.dat",
  "server-id": "s-5f7422b04c2447ef9",
  "timestamp": "2022-07-11T23:36:36.105030Z"
}
```

Sample inbound file that is unsuccessfully transferred:

```
{
  "failure-code": "INVALID_REQUEST",
  "status-code": "FAILED",
  "subject": "Sending a request from InboundHttpClientTests",
  "client-ip": "10.0.117.27",
  "as2-message-id": "testFailedLogs-TestRunConfig-Default-inbound-direct-
integ-0c97ee55-af56-4988-b7b4-a3e0576f8f9c@necco",
  "as2-to": "0beff6af56c548f28b0e78841dce44f9",
  "failure-message": "Unsupported date format: 2022/123/456T",
  "agreement-id": "a-0ceec8ca0a3348d6a",
  "as2-from": "ab91a398aed0422d9dd1362710213880",
```

```
  "file-path": "failed/01187f15-523c-43ac-9fd6-51b5ad2b08f3.testFailedLogs-
TestRunConfig-Default-inbound-direct-integ-0c97ee55-af56-4988-b7b4-a3e0576f8f9c@necco",
  "server-id": "s-0582af12e44540b9b",
  "timestamp": "2022-07-11T06:30:03.662939Z"
}
```

Customize HTTP headers for AS2 messages

When sending AS2 messages to trading partners, you may need to customize the HTTP headers to meet specific requirements or enhance compatibility with your partner's AS2 server configuration. This CloudFormation template creates an infrastructure to enable customized HTTP headers for AS2 messages sent through AWS Transfer Family. It sets up an Amazon API Gateway and Lambda function to act as a proxy, allowing dynamic modification of headers required by trading partners' AS2 servers.

Use this template to do the following:

- Add custom HTTP headers to outbound AS2 messages
- Override default header values with custom values

> **Important**
>
> Be careful when overriding default header values, as it can cause send failures: some AS2 headers are required.

- Ensure compatibility with trading partners that have specific header requirements

Template Overview

The template creates the following main components:

- A Lambda function that processes and forwards AS2 messages
- An Amazon API Gateway to expose the Lambda function
- IAM roles and permissions for the Lambda function
- Conditional resources for HTTPS support

The template file is available here: Dynamic HTTP headers template.

How It Works

1. The Amazon API Gateway receives incoming AS2 messages from AWS Transfer Family.
2. The request is forwarded to the Lambda function.
3. The Lambda function processes the request, adding or modifying headers as needed.
4. The modified request is then forwarded to the partner's AS2 server.
5. The response from the partner's server is returned through the Lambda and Amazon API Gateway back to AWS Transfer Family.

Key Features

- *Dynamic Header Modification:* Allows customization of the Subject header and addition of other required headers.
- *Protocol Support:* Works with both HTTP and HTTPS protocols.
- *Flexible Configuration:* Allows specification of partner host, port, and path.

Implementation Details

The template implements the following key components:

Lambda Function

The core of the solution is a Node.js Lambda function that:

- Receives requests from the Amazon API Gateway
- Modifies headers based on configuration and incoming request data
- Forwards the modified request to the partner's AS2 server
- Handles both HTTP and HTTPS protocols
- Includes error handling and logging

Amazon API Gateway

An HTTP API is set up to:

- Receive incoming AS2 messages

- Route requests to the Lambda function
- Return responses back to AWS Transfer Family

Template parameters

Enter information for the template parameters as follows. Note that all of these parameters are strings.

- `Environment`: this parameter is used to name the resources that the template creates: whether they are intended for a development or production environment. Valid values are **dev** and **prod**.
- `PartnerHost`: the IP address or hostname of the AS2 partner server.
- `PartnerPort`: the port number for the AS2 partner server. If not specified, defaults to 80 for HTTP and 443 for HTTPS.
- `PartnerPath`: the path to the AS2 endpoint on the partner server
- `ProtocolType`: the protocol to use for the AS2 communication: valid values are **HTTP** and **HTTPS**.

Conditional Resources

For HTTPS support, the template conditionally creates:

- A Lambda Layer for CA certificates
- HTTPS-specific configuration in the Lambda function

Deployment and Usage

To customize AS2 HTTP headers using a CloudFormation template

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the left navigation pane, choose **Stacks**.
3. Choose **Create stack**, and then choose **With new resources (standard)**.
4. In the **Prerequisite - Prepare template** section, choose **Choose an existing template**.
5. Copy this link, Dynamic HTTP headers template, and paste it into the **Amazon S3 URL** field.
6. Choose **Next**.

7. Fill in the parameter details with your information. These are detailed in Template parameters.
8. Choose **Next**. On the **Configure stack options** page, choose **Next** again.
9. Review the details for the stack that you're creating, and then choose **Create stack**.

> **Note**
>
> At the bottom of the page, under **Capabilities**, you must acknowledge that CloudFormation might create AWS Identity and Access Management (IAM) resources.

After deploying this CloudFormation stack:

1. Note the Amazon API Gateway endpoint URL provided in the stack outputs.
2. Update your existing AWS Transfer Family Connector to use this new Amazon API Gateway endpoint.
3. The solution will now handle AS2 messages, adding or modifying headers as configured.

> **Warning**
>
> Only modify the Subject header or add headers that your partner explicitly expects. Changing other headers may cause transfer failures.

Monitoring AS2 usage

You can monitor AS2 activity using Amazon CloudWatch and AWS CloudTrail. To view other Transfer Family server metrics, see Amazon CloudWatch logging for AWS Transfer Family servers

AS2 metrics

Metric	Description
`InboundMessage`	The total number of AS2 messages successfully received from a trading partner. Units: Count Period: 5 minutes

Metric	Description
`InboundFailedMessage`	The total number of AS2 messages that were unsuccessfully received from a trading partner. That is, a trading partner sent a message, but the Transfer Family server was not able to successfully process it. Units: Count Period: 5 minutes
`OutboundMessage`	The total number of AS2 messages successfu lly sent from the Transfer Family server to a trading partner. Units: Count Period: 5 minute
`OutboundFailedMessage`	The total number of AS2 messages that were unsuccessfully sent to a trading partner. That is, they were sent from the Transfer Family server, but were not successfully received by the trading partner. Units: Count Period: 5 minutes

Metric	Description
`DaysUntilExpiry`	The number of days until a Certificate expires determined by the `InactiveDate` set on the Certificate when importing. Units: Count Dimensions: `CertificateId` , `Descripti on` (if provided) Period: 1 day For more information, see AS2 certificate rotation.

AS2 Status codes

The following table lists all of the status codes that can be logged to CloudWatch logs when you or your partner send an AS2 message. Different message processing steps apply to different message types and are intended for monitoring only. The COMPLETED and FAILED states represent the final step in processing, and are visible in JSON files.

Code	Description	Processing completed?
PROCESSING	The message is in the process of being converted to its final format. For example, decompression and decryptio n steps both have this status.	No
MDN_TRANSMIT	Message processing is sending an MDN response.	No
MDN_RECEIVE	Message processing is receiving an MDN response.	No

Code	Description	Processing completed?
COMPLETED	Message processing has completed successfully. This state includes when an MDN is sent for an inbound message or for MDN verificat ion of outbound messages.	Yes
FAILED	The message processing has failed. For a list of error codes, see AS2 error codes.	Yes

AS2 error codes

The following table lists and describes error codes that you might receive from AS2 file transfers.

AS2 error codes

Code	Error	Description and resolution
`ACCESS_DENIED`	• Access denied. Check if your access role has necessary permissions. • Invalid file path *`send-file-path`* • Failed to get credentials with ErrorCode: *`error-code`*	Occurs when handling a `StartFileTransfer` request where any of the `SendFilePaths` are not valid or malformed. That is, the path is missing the Amazon S3 bucket name, or the path includes characters that aren't valid. Also occurs if Transfer Family fails to assume the access role or logging role. Ensure that the path contains a valid Amazon S3 bucket name and key name.

Code	Error	Description and resolution
AGREEMENT_NOT_FOUND	Agreement was not found.	Either the agreement was not found, or the agreement is associated with an inactive profile. Update the agreement within the Transfer Family server to include active profiles.
CONNECTOR_NOT_FOUND	Connector or related configuration was not found.	Either the connector was not found, or the connector is associated with an inactive profile. Update the connector to include active profiles.

Code	Error	Description and resolution
`CREDENTIALS_RETRIEVAL_FAILED`	1. Secret not found in Secrets Manager. 2. Cannot access Secrets Manager. 3. Failed to decrypt secret in Secrets Manager. 4. Cannot get secret value due to throttling.	For AS2 Basic authentication, the secret must be formatted correctly. The following resolutions correspond to the errors listed in the previous column. 1. Ensure that the secret ID is correct. 2. Ensure that the access role has the appropriate permissions to read the secret. The access role must provide read and write access to the parent directory of the file location used in the `StartFileTransfer` request. Additionally, make sure that the role provides read and write access to the parent directory of the files that you intend to send with `StartFileTransfer` . 3. If a customer managed key is being used for the secret, ensure that the access role has permissions for the AWS Key Management Service (AWS KMS) key. 4. For the applicable quotas, see Quotas for handling secrets.

Code	Error	Description and resolution
DECOMPRESSION_FAILED	Failed to decompress message.	Either the file sent is corrupt, or the compression algorithm is not valid. Resend the message and verify that ZLIB compressi on is used, or resend the message without compression enabled.
DECRYPT_FAILED	Failed to decrypt message *message-ID* . Ensure that the partner has the correct public encryption key.	Decryption failed. Confirm that the partner sent a payload by using a valid certificate and that encryptio n was performed by using a valid encryption algorithm.
DECRYPT_FAILED_INV ALID_SMIME_FORMAT	Unable to parse enveloped mimePart.	MIME payload is either corrupt or in an unsupported SMIME format. The sender should make sure that the format they're using is supported, and then resend the payload.

Code	Error	Description and resolution
DECRYPT_FAILED_NO_DECRYPTION_KEY_FOUND	No matching decryption key found.	The partner profile did not have a certificate assigned that matched the message, or the certificates that matched the message are now expired or no longer valid. You must update the partner profile and ensure that it contains a valid certificate.
DECRYPT_FAILED_UNSUPPORTED_ENCRYPTION_ALG	SMIME Payload Decryption requested using unsupported algorithm with ID: *encryption-ID* .	The remote sender has sent an AS2 payload with an unsupported encryption algorithm. The sender must choose an encryption algorithm that's supported by AWS Transfer Family.
DUPLICATE_MESSAGE	Duplicate or double processed step.	The payload has a duplicate processing step. For example, there are two encryption steps. Resend the message with a single step for signing, compression, and encryption.

Code	Error	Description and resolution
ENCRYPT_FAILED_NO_ENCRYPTION_KEY_FOUND	No valid public encryption certificates found in profile: *local-profile-ID*	Transfer Family is attempting to encrypt an outbound message, but no encryption certificates are found for the local profile. Resolution options: • Ensure that the local profile has a certificate and private key for encryption attached. • Ensure that the encryption certificate is currently active.
ENCRYPTION_FAILED	Failed to encrypt file *file-name* .	The file to be sent is not available for encryption. Verify that the file is in its expected AS2 location and that AWS Transfer Family has permission to read the file.
FILE_SIZE_TOO_LARGE	File size is too large.	This occurs when sending or receiving a file that exceeds the file size limit.
HTTP_ERROR_RESPONSE_FROM_PARTNER	*partner-URL* returned status 400 for message with ID=*message-ID* .	Communicating with the partner's AS2 server returned an unexpected HTTP response code. The partner might be able to provide more diagnostics from their AS2 server logs.

Code	Error	Description and resolution
INSUFFICENT_MESSAGE_SECURITY_UNENCRYPTED	Encryption is required.	The partner sent an unencrypted message to Transfer Family, which is not supported. The sender must use an encrypted payload.
INVALID_ENDPOINT_PROTOCOL	Only HTTP and HTTPS are supported.	You must specify HTTP or HTTPS as the protocol in your AS2 connector configuration.

Code	Error	Description and resolution
`INVALID_REQUEST`	1. There is a problem with a message header. 2. Could not parse secret JSON. Secret JSON did not match expected format. 3. Secret must be a JSON string. 4. Username must not contain a colon. Username must not contain control characters. Username must contain only ASCII characters. Password must not contain control characters. Password must contain only ASCII characters.	This error has several causes. The following resolutions correspond to the errors listed in the previous column. 1. Check the `as2-from` and `as2-to` fields. Make sure that the original message ID is accurate for the MDN format. Also make sure that the message ID format is not missing any AS2 headers. 2. Ensure that the secret value matches the documented format, as described in Enable Basic authentication for AS2 connectors. 3. Ensure that the secret is provided as a string, and not as a binary. 4. Make the necessary correction to the username or password.

Code	Error	Description and resolution
INVALID_URL_FORMAT	Invalid URL format: *URL*	This occurs when you are sending an outbound message using a connector configured with a malformed URL. Ensure that the connector is configured with a valid HTTP or HTTPS URL.
MDN_RESPONSE_INDICATES_AUTHENTICATION_FAILED	Not applicable	The receiver cannot authenticate the sender. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: authentication-failed.
MDN_RESPONSE_INDICATES_DECOMPRESSION_FAILED	Not applicable	This occurs when the receiver cannot decompress the message contents. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: decompression-failed.
MDN_RESPONSE_INDICATES_DECRYPTION_FAILED	Not applicable	The receiver cannot decrypt the message contents. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: authentication-failed.

Code	Error	Description and resolution
`MDN_RESPONSE_INDICATES_INSUFFICIENT_MESSAGE_SECURITY`	Not applicable	The receiver expects the message to be signed or encrypted, but it isn't. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: insufficient-message-security. Enable signing and/or encryption on the connector to match the trading partner's expectations.
`MDN_RESPONSE_INDICATES_INTEGRITY_CHECK_FAILED`	Not applicable	The receiver cannot verify content integrity. The trading partner returns an MDN to Transfer Family with the disposition modifier Error: integrity-check-failed.
`PATH_NOT_FOUND`	Unable to create directory *`file-path`* . The parent path could not be found.	Transfer Family is attempting to create a directory in the customer's Amazon S3 bucket, but the bucket is not found. Ensure that each path mentioned in the `StartFileTransfer` command contains the name of an existing bucket.

Code	Error	Description and resolution
SEND_FILE_NOT_FOUND	File path *file-path* not found.	Transfer Family can't locate the file in the send file operation. Check that the configured home directory and path are valid and that Transfer Family has read permissions for the file.
SERVER_NOT_FOUND	Server associated with the message cannot be found.	Transfer Family could not find the server when receiving a message. This can happen if the server is deleted during the processing of an incoming message.
SERVER_NOT_ONLINE	Server *server-ID* is not online.	The Transfer Family server is offline. Start the server so that it can receive and process messages.
SIGNING_FAILED	Failed to sign file.	The file to be sent is not available for signing, or signing could not be performed. Verify that the file is in its expected AS2 location and that AWS Transfer Family has permission to read the file.

Code	Error	Description and resolution
SIGNING_FAILED_NO_SIGNING_KEY_FOUND	No certificate found for profile: *local-profile-ID*.	Attempting to sign an outbound message, but no signing certificates are found for the local profile. Resolution options: • Ensure that the local profile has a certificate and private key for signing attached. • Ensure that the signing certificate is currently active.
UNABLE_RESOLVE_HOST_TO_IP_ADDRESS	Unable to resolve hostname to IP addresses.	Transfer Family is unable to perform DNS to IP address resolution on the public DNS server that is configured in the AS2 connector. Update the connector to point to a valid partner URL.
UNABLE_TO_CONNECT_TO_REMOTE_HOST_OR_IP	Connection to endpoint timed out.	Transfer Family cannot establish a socket connection to the configured partner's AS2 server. Check that the partner's AS2 server is available at the configured IP address.

Code	Error	Description and resolution
UNABLE_TO_RESOLVE_HOSTNAME	Unable to resolve hostname *hostname*.	The Transfer Family server could not resolve the partner's hostname by using a public DNS server. Check that the configured host is registered and that the DNS record has had time to publish.
VERIFICATION_FAILED	Signature verification failed for AS2 message *message-ID* or a MIC code did not match.	Check that the sender's signing certificate matches the signing certificates for the remote profile. Also check that the MIC algorithms are compatible with AWS Transfer Family.
VERIFICATION_FAILED_NO_MATCHING_KEY_FOUND	• No public certificate matching message signature could be found in profile: *partner-profile-ID* . • Cannot get certificates for non-existent profile: *partner-profile-ID* . • No valid certificate was found in profile: *partner-profile-ID* .	AWS Transfer Family is attempting to verify the signature for a received message, but no matching signing certificate is found for the partner profile. Resolution options: • Ensure that the partner profile has a signing certificate attached. • Ensure that the certificate is currently active. • Ensure that the certificate is the correct signing certificate for the partner.

Certificate expiration monitoring

AWS Transfer Family automatically monitors the expiration dates of AS2 certificates and publishes a Amazon CloudWatch metric to help you track when certificates are approaching expiration. This allows you to proactively manage certificate renewals and avoid service disruptions.

DaysUntilExpiry metric

When you import a certificate for AS2 use, Transfer Family automatically creates a CloudWatch metric called `DaysUntilExpiry`. This metric tracks the number of days remaining until the certificate expires based on the `InactiveDate` you specified when importing the certificate.

Metric details:

- **Metric name:** `DaysUntilExpiry`
- **Namespace:** `AWS/Transfer`
- **Dimensions:** `CertificateId` (always present), `Description` (if provided during certificate import)
- **Units:** Count (days)
- **Frequency:** Published daily

> **Important**
>
> It can take up to a full day after importing a certificate for Transfer Family to emit this metric to your account.

The metric value decreases by one each day as the certificate approaches its inactive date. For example, if a certificate has 30 days until expiration, the metric will show 30, then 29 the next day, and so on.

Best practices for certificate monitoring

Follow these best practices when setting up certificate expiration monitoring:

- **Set multiple alert thresholds:** Create alarms for different time periods (for example, 30 days, 14 days, and 7 days before expiration) to provide adequate time for certificate renewal.

- **Use appropriate statistics:** Use the `Maximum` statistic when creating alarms to ensure you capture the most recent metric value.
- **Configure proper alarm actions:** Set up notifications to alert the appropriate team members who can renew certificates.
- **Test your alerts:** Regularly test your notification system to ensure alerts are delivered properly.
- **Document your process:** Maintain documentation about your certificate renewal process and who is responsible for different certificates.

Example alarm configurations

Here are some example alarm configurations for different notification scenarios:

30-day expiration warning

Create an alarm that triggers when a certificate has 30 days or fewer until expiration:

- **Metric:** DaysUntilExpiry
- **Statistic:** Maximum
- **Period:** 1 day
- **Threshold:** 30
- **Comparison:** Less than or equal to threshold
- **Missing data treatment:** Treat missing data as good (not breaching)

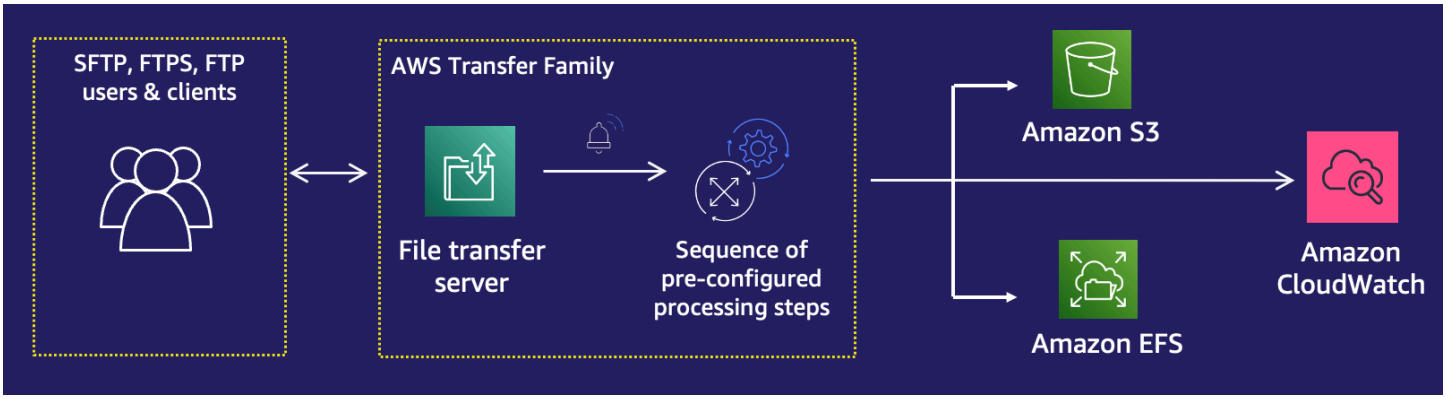
Critical 7-day expiration warning

Create a critical alarm that triggers when a certificate has 7 days or fewer until expiration:

- **Metric:** DaysUntilExpiry
- **Statistic:** Maximum
- **Period:** 1 day
- **Threshold:** 7
- **Comparison:** Less than or equal to threshold
- **Missing data treatment:** Treat missing data as good (not breaching)

AWS Transfer Family managed workflows

AWS Transfer Family supports managed workflows for file processing. With managed workflows, you can kick off a workflow after a file has been transferred over SFTP, FTPS, or FTP. Using this feature, you can securely and cost effectively meet your compliance requirements for business-to-business (B2B) file exchanges by coordinating all the necessary steps required for file processing. In addition, you benefit from end-to-end auditing and visibility.



By orchestrating file-processing tasks, managed workflows help you preprocess data before it is consumed by your downstream applications. Such file-processing tasks might include:

- Moving files to user-specific folders.
- Decrypting files as part of a workflow.
- Tagging files.
- Performing custom processing by creating and attaching an AWS Lambda function to a workflow.
- Sending notifications when a file has been successfully transferred. (For a blog post that details this use case, see Customize file delivery notifications using AWS Transfer Family managed workflows.)

To quickly replicate and standardize common post-upload file processing tasks spanning multiple business units in your organization, you can deploy workflows by using infrastructure as code (IaC). You can specify a managed workflow to be initiated on files that are uploaded in full. You can also specify a different managed workflow to be initiated on files that are only partially uploaded because of a premature session disconnect. Built-in exception handling helps you quickly react to file-processing outcomes, while offering you control over how to handle failures. In addition, each workflow step produces detailed logs, which you can audit to trace the data lineage.

To get started, perform the following tasks:

1. Set up your workflow to contain preprocessing actions, such as copying, tagging, and other steps based on your requirements. See Create a workflow for details.
2. Configure an execution role, which Transfer Family uses to run the workflow. See IAM policies for workflows for details.
3. Map the workflow to a server, so that on file arrival, the actions specified in this workflow are evaluated and initiated in real time. See Configure and run a workflow for details.

Related information

- To monitor your workflow executions, see Using CloudWatch metrics for Transfer Family servers.
- For detailed execution logs and troubleshooting information, see Troubleshoot workflow-related errors using Amazon CloudWatch.
- Transfer Family provides a blog post and a workshop that walk you through building a file transfer solution. This solution leverages AWS Transfer Family for managed SFTP/FTPS endpoints and Amazon Cognito and DynamoDB for user management.

  The blog post is available at Using Amazon Cognito as an identity provider with AWS Transfer Family and Amazon S3. You can view the details for the workshop here.
- View AWS Transfer Family Managed Workflows for a brief introduction to Transfer Family workflows.
- The following workshop provides hands on labs to build fully automated and event-driven workflows involving file transfer to or from external SFTP servers to Amazon S3, and common pre- and post-processing of those files: Event-driven MFT workshop.

  This video provides a walk through of this workshop.

Topics

- Create a workflow
- Use predefined steps
- Use custom file-processing steps
- IAM policies for workflows
- Exception handling for a workflow
- Monitor workflow execution

- Create a workflow from a template
- Remove a workflow from a Transfer Family server
- Managed workflows restrictions and limitations

For more help getting started with managed workflows, see the following resources:

- AWS Transfer Family managed workflows demo video
- Building a cloud-native file transfer platform using AWS Transfer Family workflows blog post

Create a workflow

You can create a managed workflow by using the AWS Management Console, as described in this topic. To make the workflow creation process as easy as possible, contextual help panels are available for most of the sections in the console.

A workflow has two kinds of steps:

- **Nominal steps** – Nominal steps are file-processing steps that you want to apply to incoming files. If you select more than one nominal step, each step is processed in a linear sequence.
- **Exception-handling steps** – Exception handlers are file-processing steps that AWS Transfer Family executes in case any nominal steps fail or result in validation errors.

Create a workflow

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Workflows**.
3. On the **Workflows** page, choose **Create workflow**.
4. On the **Create workflow** page, enter a description. This description appears on the **Workflows** page.
5. In the **Nominal steps** section, choose **Add step**. Add one or more steps.
    a. Choose a step type from the available options. For more information about the various step types, see the section called “Use predefined steps”.
    b. Choose **Next**, then configure parameters for the step.
    c. Choose **Next**, then review the details for the step.

d. Choose **Create step** to add the step and continue.

e. Continue adding steps as needed. The maximum number of steps in a workflow is 8.

f. After you have added all of the necessary nominal steps, scroll down to the **Exception handlers – *optional*** section, and choose **Add step**.

> **Note**
>
> So that you are informed of failures in real time, we recommend that you set up exception handlers and steps to execute when your workflow fails.

6. To configure exception handlers, add steps in the same manner as described previously. If a file causes any step to throw an exception, your exception handlers are invoked one by one.
7. (Optional) Scroll down to the **Tags** section, and add tags for your workflow.
8. Review the configuration, and choose **Create workflow**.

> **Important**
>
> After you've created a workflow, you can't edit it, so make sure to review the configuration carefully.

Configure and run a workflow

Before you can run a workflow, you need to associate it with a Transfer Family server.

To configure Transfer Family to run a workflow on uploaded files

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**.
   - To add the workflow to an existing server, choose the server that you want to use for your workflow.
   - Alternatively, create a new server and add the workflow to it. For more information, see Configuring an SFTP, FTPS, or FTP server endpoint.
3. On the details page for the server, scroll down to the **Additional details** section, and then choose **Edit**.

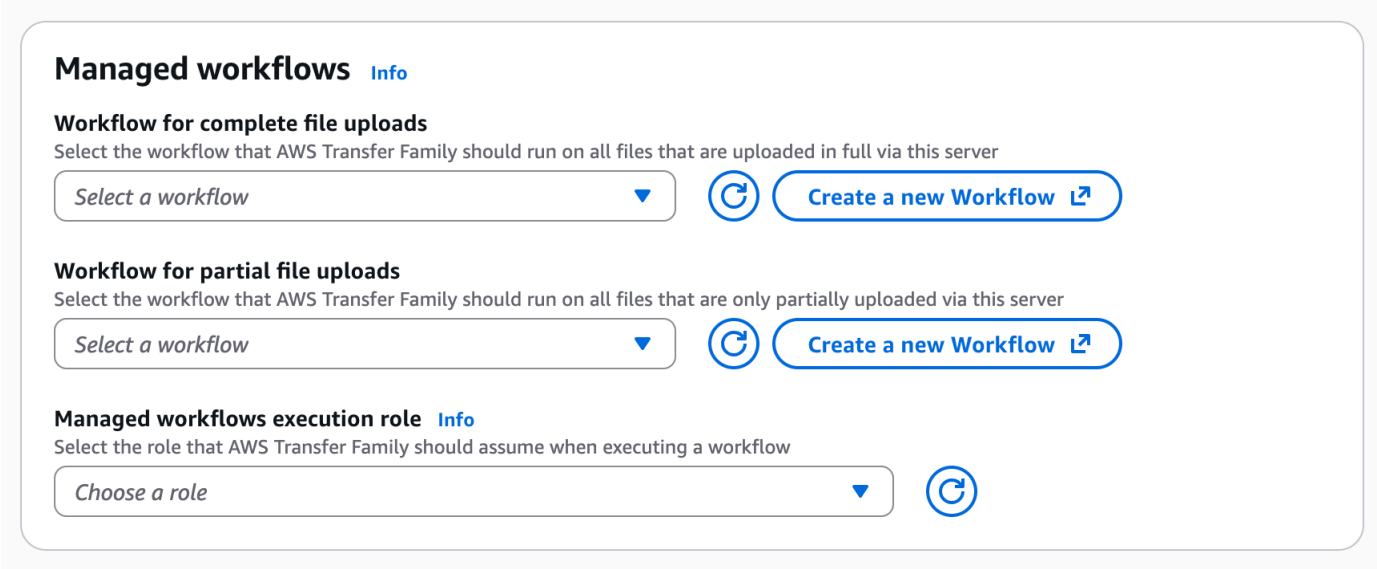
> **Note**
>
> By default, servers do not have any associated workflows. You use the **Additional details** section to associate a workflow with the selected server.

4. On the **Edit additional details** page, in the **Managed workflows** section, select a workflow to be run on all uploads.

> **Note**
>
> If you do not already have a workflow, choose **Create a new Workflow** to create one.

   a. Choose the workflow ID to use.

   b. Choose an execution role. This is the role that Transfer Family assumes when executing the workflow's steps. For more information, see IAM policies for workflows. Choose **Save**.



> **Note**
>
> If you no longer want a workflow to be associated with the server, you can remove the association. For details, see Remove a workflow from a Transfer Family server.

To execute a workflow

To execute a workflow, you upload a file to a Transfer Family server that you configured with an associated workflow.

> **Note**
>
> Anytime you remove a workflow from a server and replace it with a new one, or update server configuration (which impacts a workflow's execution role), you must wait approximately 10 minutes before executing the new workflow. The Transfer Family server caches the workflow details, and it takes 10 minutes for the server to refresh its cache. Additionally, you must log out of any active SFTP sessions, and then log back in after the 10-minute waiting period to see the changes.

Example

```
# Execute a workflow
> sftp bob@s-1234567890abcdef0.server.transfer.us-east-1.amazonaws.com

Connected to s-1234567890abcdef0.server.transfer.us-east-1.amazonaws.com.
sftp> put doc1.pdf
Uploading doc1.pdf to /amzn-s3-demo-bucket/home/users/bob/doc1.pdf
doc1.pdf                                                                    100% 5013KB
 601.0KB/s   00:08
sftp> exit
>
```

After your file has been uploaded, the action defined is performed on your file. For example, if your workflow contains a copy step, the file is copied to the location that you defined in that step. You can use Amazon CloudWatch Logs to track the steps that executed and their execution status.

View workflow details

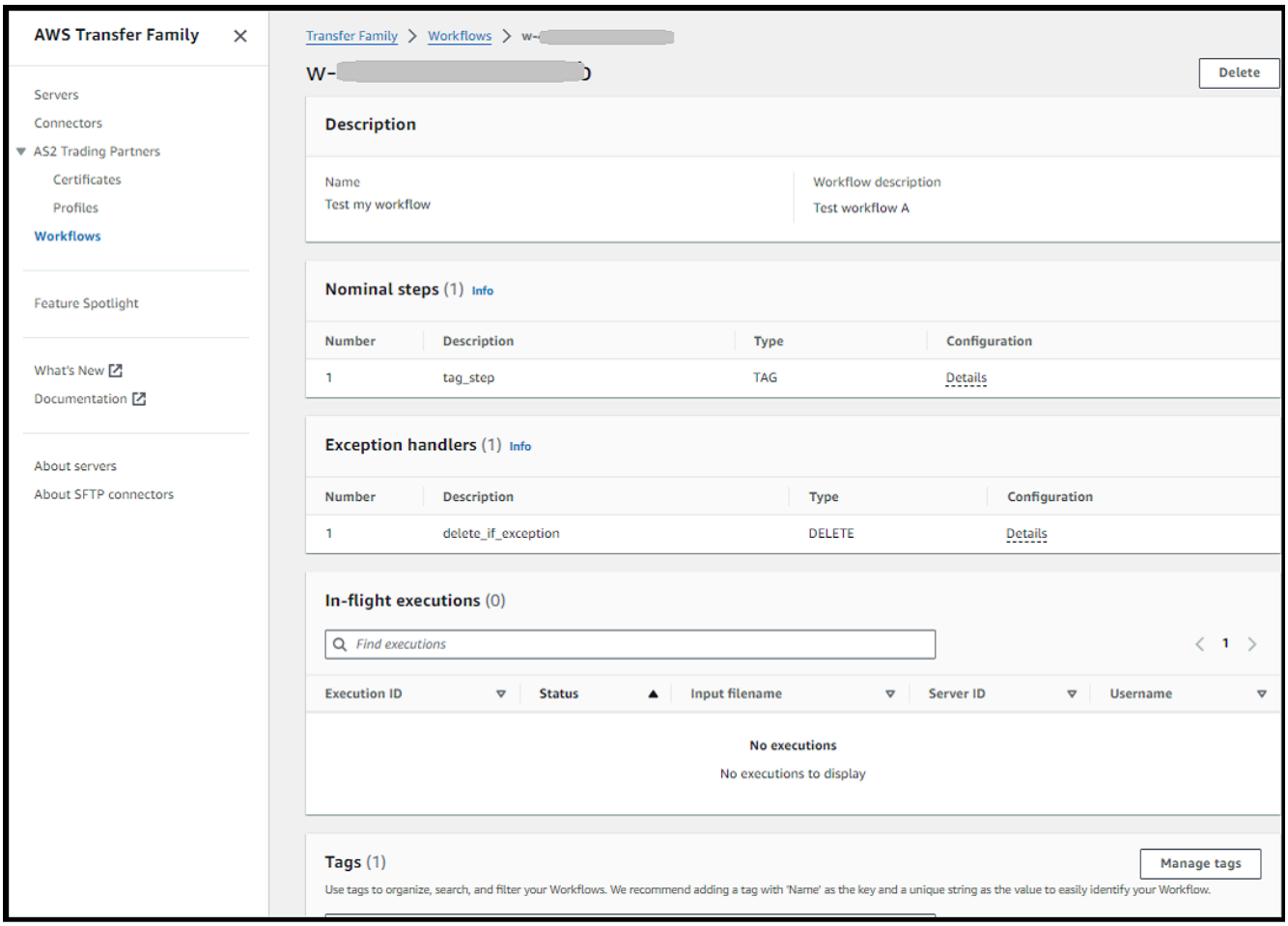
You can view details about previously created workflows or to workflow executions. To view these details, you can use the console or the AWS Command Line Interface (AWS CLI).

Console

View workflow details

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Workflows**.
3. On the **Workflows** page, choose a workflow.

   The workflow details page opens.



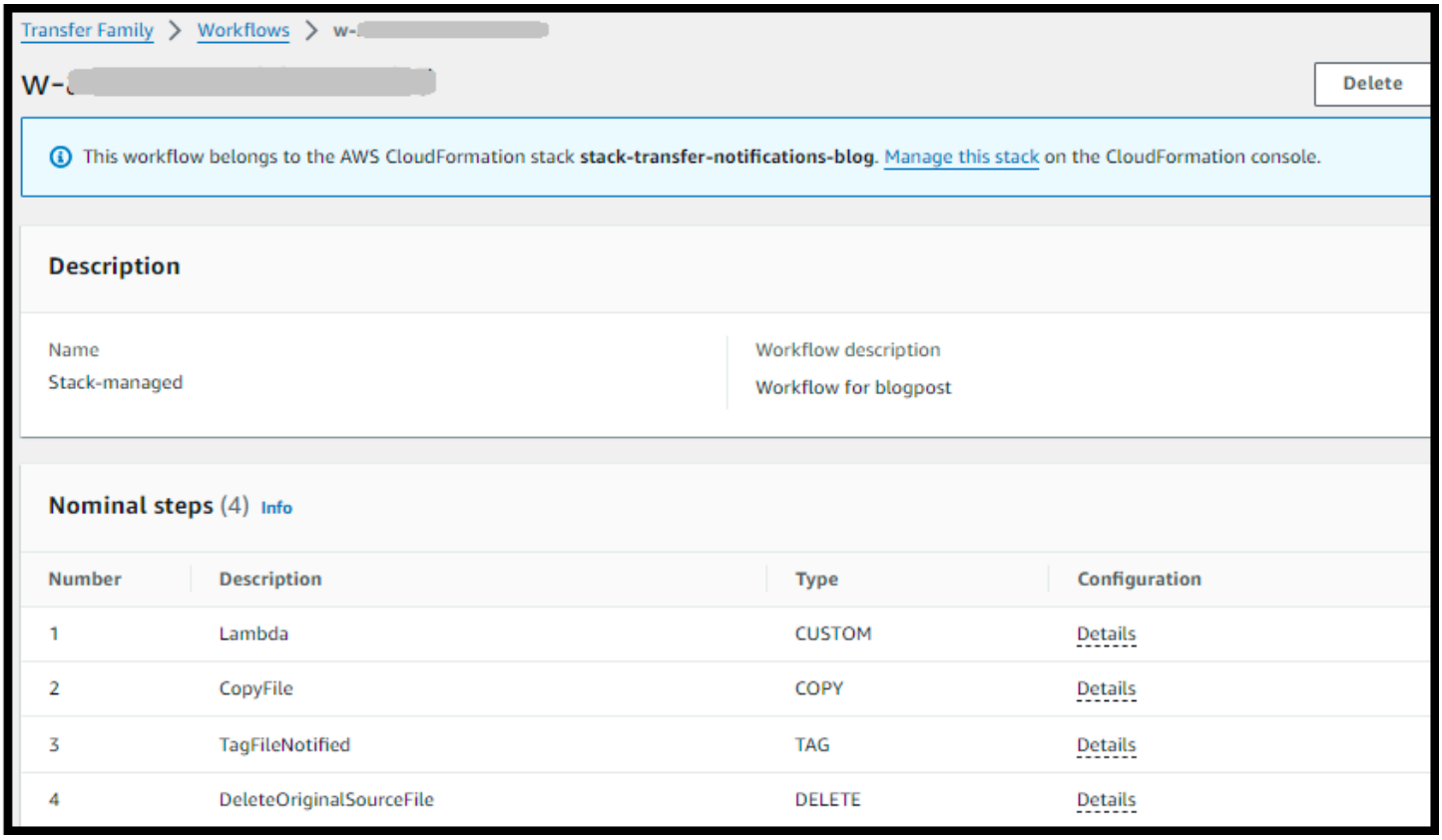
CLI

To view the workflow details, use the `describe-workflow` CLI command, as shown in the following example. Replace the workflow ID *`w-1234567890abcdef0`* with your own value. For more information, see describe-workflow in the *AWS CLI Command Reference*.

```
# View Workflow details
```

```
> aws transfer describe-workflow --workflow-id w-1234567890abcdef0
{
    "Workflow": {
        "Arn": "arn:aws:transfer:us-east-1:111122223333:workflow/
w-1234567890abcdef0",
        "WorkflowId": "w-1234567890abcdef0",
        "Name": "Copy file to shared_files",
        "Steps": [
            {
                "Type": "COPY",
                "CopyStepDetails": {
                "Name": "Copy to shared",
                "FileLocation": {
                    "S3FileLocation": {
                        "Bucket": "amzn-s3-demo-bucket",
                        "Key": "home/shared_files/"
                    }
                }
                }
            }
        ],
        "OnException": {}
    }
}
```

If your workflow was created as part of an AWS CloudFormation stack, you can manage the workflow using the CloudFormation console (https://console.aws.amazon.com/cloudformation).



Use predefined steps

When you're creating a workflow, you can choose to add one of the following predefined steps discussed in this topic. You can also choose to add your own custom file-processing steps. For more information, see the section called “Use custom file-processing steps”.

Topics

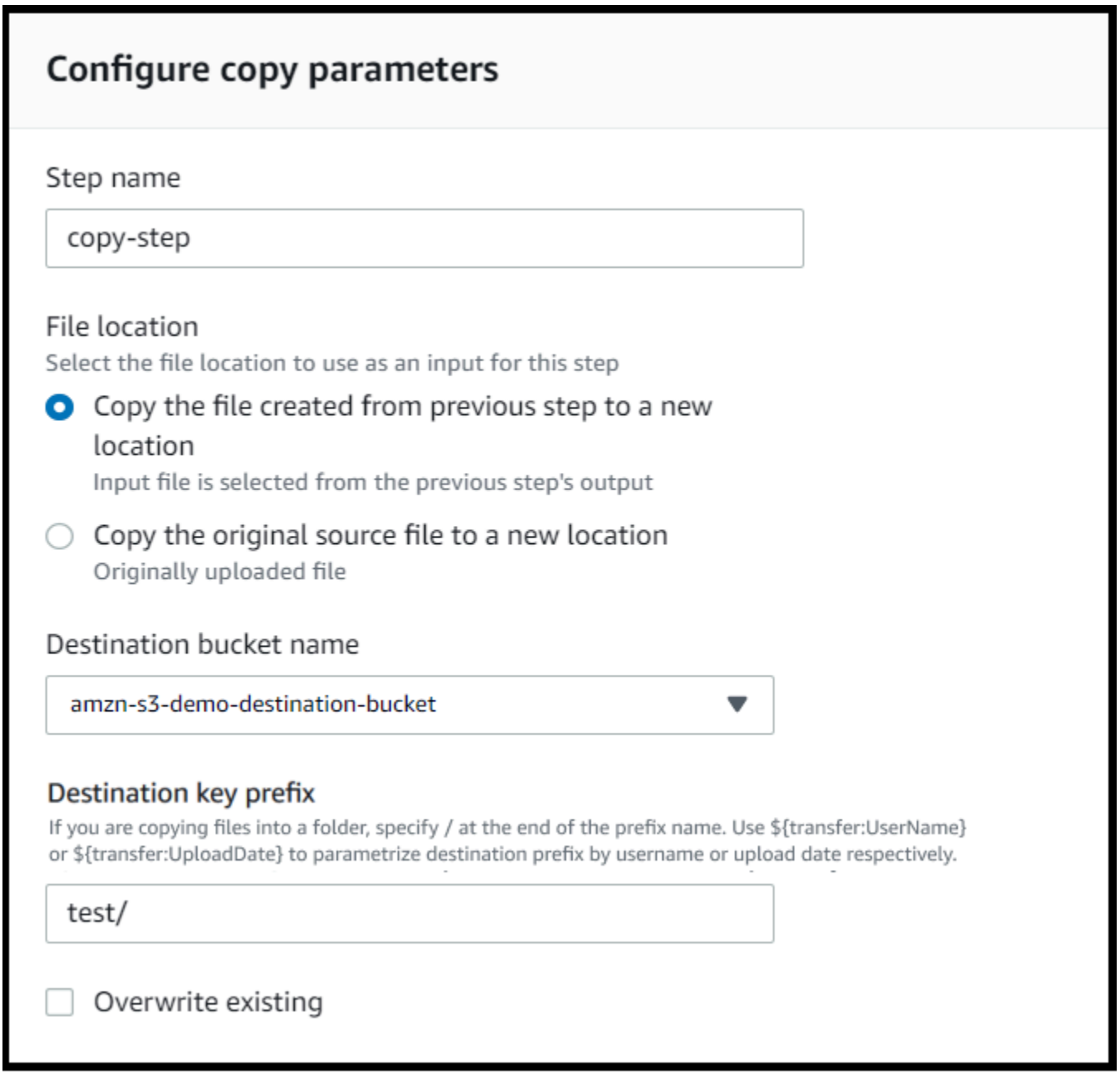
- Copy file
- Decrypt file
- Tag file
- Delete file
- Named variables for workflows
- Example tag and delete workflow

Copy file

A copy file step creates a copy of the uploaded file in a new Amazon S3 location. Currently, you can use a copy file step only with Amazon S3.

The following copy file step copies files into the `test` folder in *`amzn-s3-demo-destination-bucket`*.

If the copy file step is not the first step of your workflow, you can specify the **File location**. By specifying the file location, you can copy either the file that was used in the previous step or the original file that was uploaded. You can use this feature to make multiple copies of the original file while keeping the source file intact for file archival and records retention. For an example, see Example tag and delete workflow.



Provide the bucket and key details

You must provide the bucket name and a key for the destination of the copy file step. The key can be either a path name or a file name. Whether the key is treated as a path name or a file name is determined by whether you end the key with the forward slash (/) character.

If the final character is /, your file is copied to the folder, and its name does not change. If the final character is alphanumeric, your uploaded file is renamed to the key value. In this case, if a file with that name already exists, the behavior depends on the setting for the **Overwrite existing** field.

- If **Overwrite existing** is selected, the existing file is replaced with the file being processed.
- If **Overwrite existing** is not selected, nothing happens, and the workflow processing stops.

> **Tip**
>
> If concurrent writes are executed on the same file path, it may result in unexpected behavior when overwriting files.

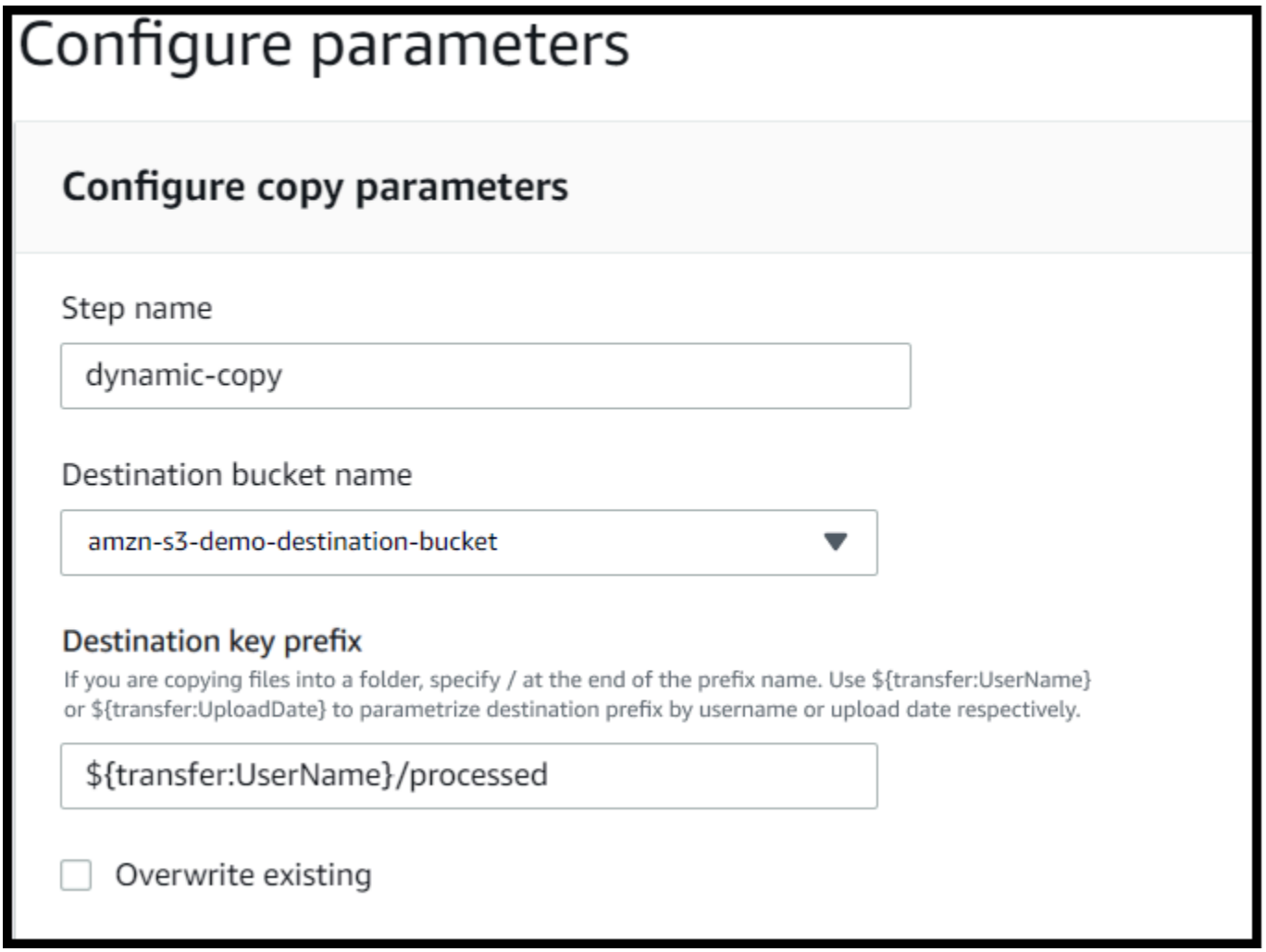
For example, if your key value is `test/`, your uploaded files are copied to the `test` folder. If your key value is `test/today`, (and **Overwrite existing** is selected) every file you upload is copied to a file named `today` in the `test` folder, and each succeeding file overwrites the previous one.

> **Note**
>
> Amazon S3 supports buckets and objects, and there is no hierarchy. However, you can use prefixes and delimiters in object key names to imply a hierarchy and organize your data in a way similar to folders.

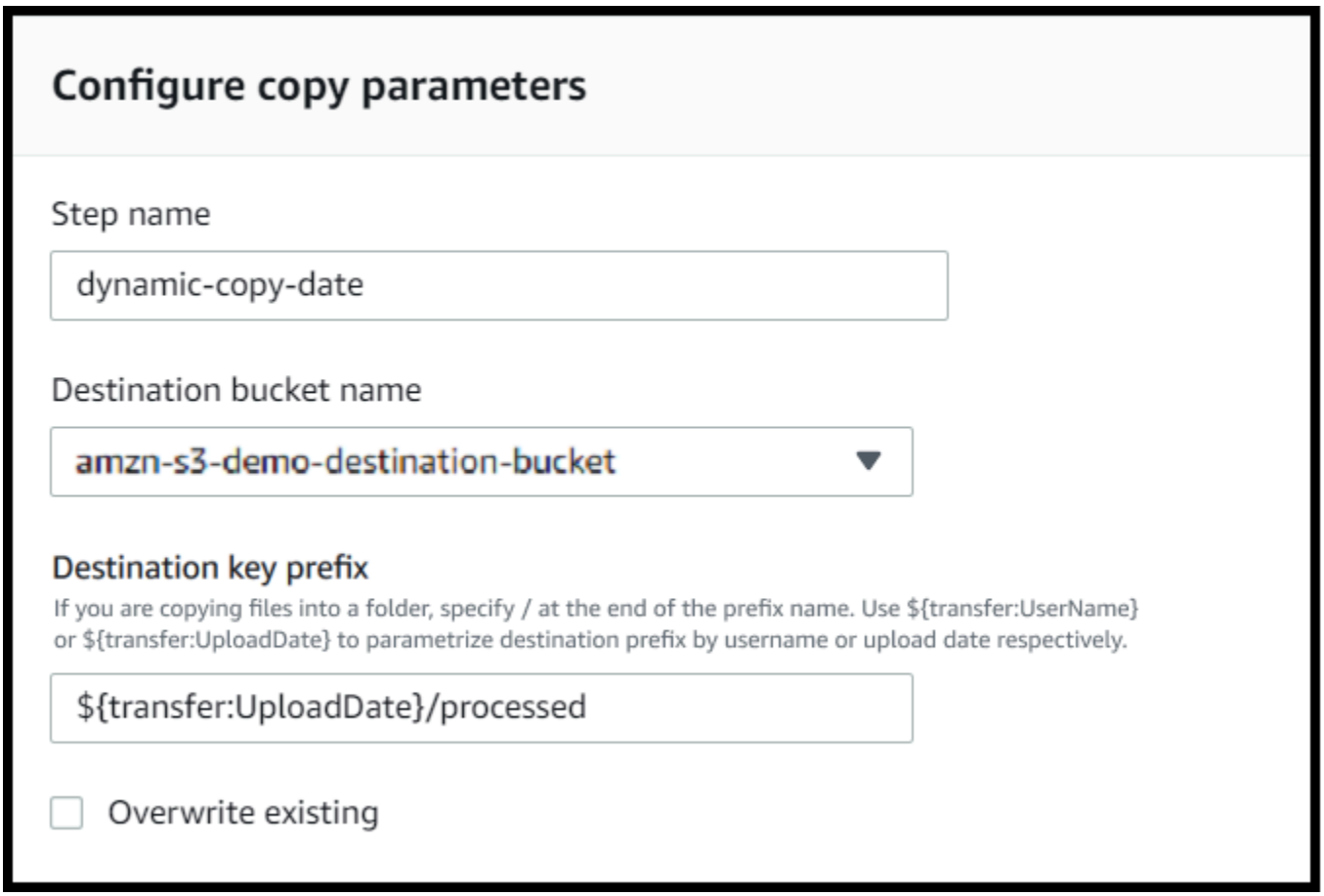
Use a named variable in a copy file step

In a copy file step, you can use a variable to dynamically copy your files into user-specific folders. Currently, you can use `${transfer:UserName}` or `${transfer:UploadDate}` as a variable to copy files to a destination location for the given user who's uploading files, or based on the current date.

In the following example, if the user `richard-roe` uploads a file, it gets copied into the `amzn-s3-demo-destination-bucket/richard-roe/processed/` folder. If the user `mary-major` uploads a file, it gets copied into the `amzn-s3-demo-destination-bucket/mary-major/processed/` folder.



Similarly, you can use `${transfer:UploadDate}` as a variable to copy files to a destination location named for the current date. In the following example, if you set the destination to `${transfer:UploadDate}/processed` on February 1, 2022, files uploaded are copied into the `amzn-s3-demo-destination-bucket/2022-02-01/processed/` folder.



You can also use both of these variables together, combining their functionality. For example, you could set the **Destination key prefix** to **`folder/${transfer:UserName}/${transfer:UploadDate}/`**, which would created nested folders, for example `folder/marymajor/2023-01-05/`.

IAM permissions for copy step

To allow a copy step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
    "Sid": "ListBucket",
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-destination-bucket"
    ]
}, {
    "Sid": "HomeDirObjectAccess",
```

```
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
}
```

Note

The `s3:ListBucket` permission is only necessary if you do not select **Overwrite existing**. This permission checks your bucket to see if a file with the same name already exists. If you have selected **Overwrite existing**, the workflow doesn't need to check for the file, and can just write it.
If your Amazon S3 files have tags, you need to add one or two permissions to your IAM policy.

- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.
- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

Decrypt file

The AWS storage blog has a post that describes how to simply decrypt files without writing any code using Transfer Family Managed workflows, Encrypt and decrypt files with PGP and AWS Transfer Family.

Supported symmetric encryption algorithms

For PGP decryption, Transfer Family supports symmetric encryption algorithms that are used to encrypt the actual file data within PGP files.

- For detailed information about supported symmetric encryption algorithms, see PGP symmetric encryption algorithms.
- For information about PGP key pair algorithms used with these symmetric algorithms, see PGP key pair algorithms.

Use PGP decryption in your workflow

Transfer Family has built-in support for Pretty Good Privacy (PGP) decryption. You can use PGP decryption on files that are uploaded over SFTP, FTPS, or FTP to Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS).

To use PGP decryption, you must create and store the PGP private keys that will be used for decryption of your files. Your users can then encrypt files by using corresponding PGP encryption keys before uploading the files to your Transfer Family server. After you receive the encrypted files, you can decrypt those files in your workflow. For a detailed tutorial, see Setting up a managed workflow for decrypting a file.

For information about supported PGP algorithms and recommendations, see PGP encryption and decryption algorithms.

To use PGP decryption in your workflow

1. Identify a Transfer Family server to host your workflow, or create a new one. You need to have the server ID before you can store your PGP keys in AWS Secrets Manager with the correct secret name.
2. Store your PGP key in AWS Secrets Manager under the required secret name. For details, see Manage PGP keys. Workflows can automatically locate the correct PGP key to be used for decryption based on the secret name in Secrets Manager.

   > **Note**
   >
   > When you store secrets in Secrets Manager, your AWS account incurs charges. For information about pricing, see AWS Secrets Manager Pricing.

3. Encrypt a file by using your PGP key pair. (For a list of supported clients, see Supported PGP clients.) If you are using the command line, run the following command. To use this command, replace *username@example.com* with the email address that you used to create the PGP key pair. Replace *testfile.txt* with the name of the file that you want to encrypt.

   ```
   gpg -e -r username@example.com testfile.txt
   ```

> **Important**
>
> When encrypting files for use with AWS Transfer Family workflows, always ensure you specify a non-anonymous recipient using the `-r` parameter. Anonymous encryption (without specifying a recipient) can cause decryption failures in the workflow because the system won't be able to identify which key to use for decryption. Debugging information for this issue is available at Troubleshoot anonymous recipient encryption issues.

4. Upload the encrypted file to your Transfer Family server.
5. Configure a decryption step in your workflow. For more information, see Add a decryption step.

Add a decryption step

A decryption step decrypts an encrypted file that was uploaded to Amazon S3 or Amazon EFS as part of your workflow. For details about configuring decryption, see Use PGP decryption in your workflow.

When you create your decryption step for a workflow, you must specify the destination for the decrypted files. You must also select whether to overwrite existing files if a file already exists at the destination location. You can monitor the decryption workflow results and get audit logs for each file in real time by using Amazon CloudWatch Logs.

After you choose the **Decrypt file** type for your step, the **Configure parameters** page appears. Fill in the values for the **Configure PGP decryption parameters** section.

The available options are as follows:

- **Step name** – Enter a descriptive name for the step.
- **File location** – By specifying the file location, you can decrypt either the file that was used in the previous step or the original file that was uploaded.

> **Note**
>
> This parameter is not available if this step is the first step of the workflow.

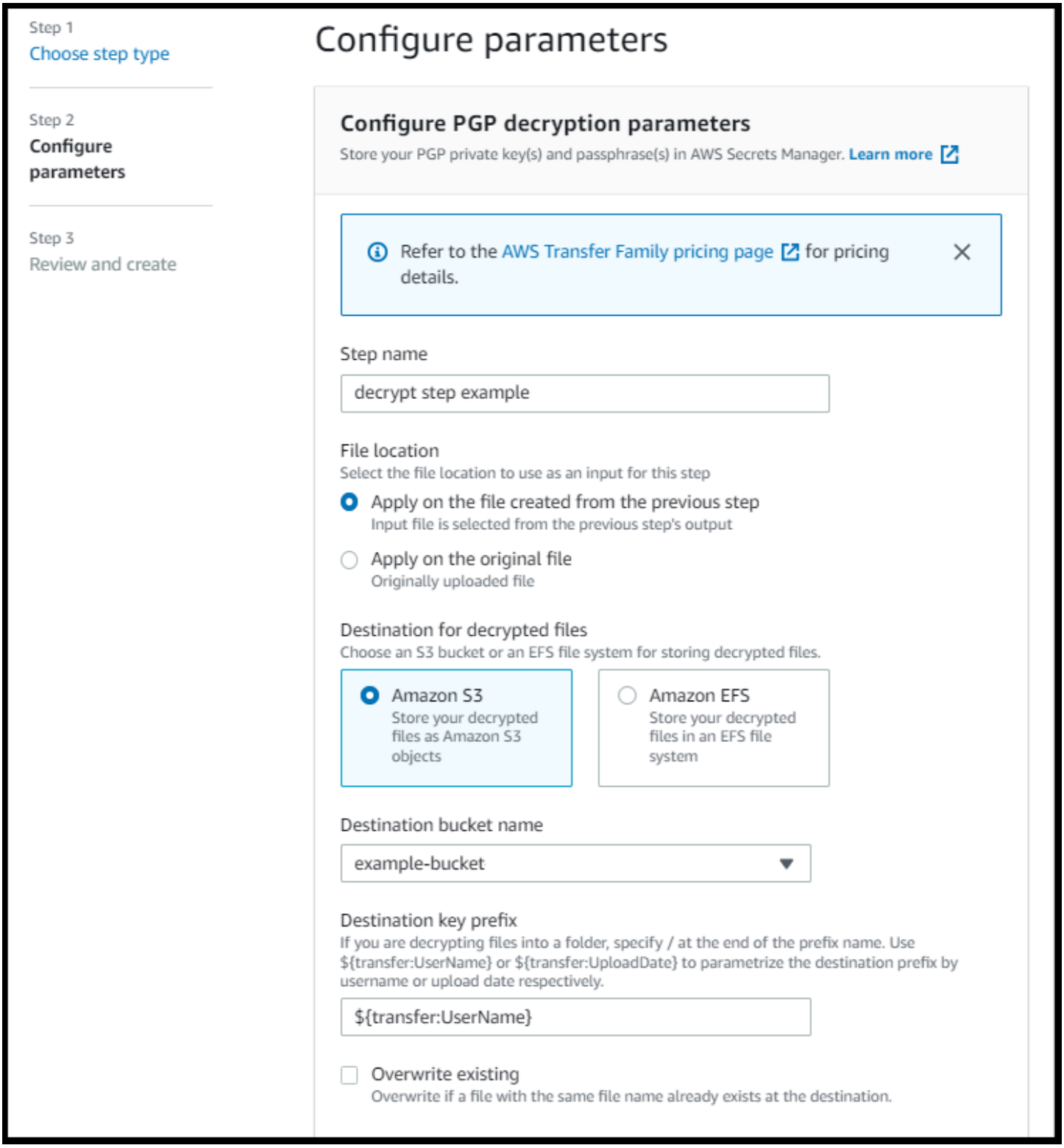
- **Destination for decrypted files** – Choose an Amazon S3 bucket or an Amazon EFS file system as the destination for the decrypted file.
  - If you choose Amazon S3, you must provide a destination bucket name and a destination key prefix. To parameterize the destination key prefix by username, enter **`${transfer:UserName}`** for **Destination key prefix**. Similarly, to parameterize the destination key prefix by upload date, enter **`${Transfer:UploadDate}`** for **Destination key prefix**.
  - If you choose Amazon EFS, you must provide a destination file system and path.

> **Note**
>
> The storage option that you choose here must match the storage system that's used by the Transfer Family server with which this workflow is associated. Otherwise, you will receive an error when you attempt to run this workflow.

- **Overwrite existing** – If you upload a file, and a file with the same filename already exists at the destination, the behavior depends on the setting for this parameter:
  - If **Overwrite existing** is selected, the existing file is replaced with the file being processed.
  - If **Overwrite existing** is not selected, nothing happens, and the workflow processing stops.

> **Tip**
>
> If concurrent writes are executed on the same file path, it may result in unexpected behavior when overwriting files.

The following screenshot shows an example of the options that you might choose for your decrypt file step.


Step 1
Choose step type
Step 2
Configure parameters
Step 3
Review and create
Configure parameters
Configure PGP decryption parameters
Store your PGP private key(s) and passphrase(s) in AWS Secrets Manager. Learn more
Refer to the AWS Transfer Family pricing page for pricing details.
Step name
decrypt step example
File location
Select the file location to use as an input for this step
Apply on the file created from the previous step
Input file is selected from the previous step's output
Apply on the original file
Originally uploaded file
Destination for decrypted files
Choose an S3 bucket or an EFS file system for storing decrypted files.
Amazon S3
Store your decrypted files as Amazon S3 objects
Amazon EFS
Store your decrypted files in an EFS file system
Destination bucket name
example-bucket
Destination key prefix
If you are decrypting files into a folder, specify / at the end of the prefix name. Use ${transfer:UserName} or ${transfer:UploadDate} to parametrize the destination prefix by username or upload date respectively.
${transfer:UserName}
Overwrite existing
Overwrite if a file with the same file name already exists at the destination.

IAM permissions for decrypt step

To allow a decrypt step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
    "Sid": "ListBucket",
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-destination-bucket"
    ]
}, {
    "Sid": "HomeDirObjectAccess",
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
}, {
    "Sid": "Decrypt",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue",
    ],
    "Resource": "arn:aws:secretsmanager:region:account-id:secret:aws/transfer/*"
}
```

Note

The `s3:ListBucket` permission is only necessary if you do not select **Overwrite existing**. This permission checks your bucket to see if a file with the same name already exists. If you have selected **Overwrite existing**, the workflow doesn't need to check for the file, and can just write it.

If your Amazon S3 files have tags, you need to add one or two permissions to your IAM policy.

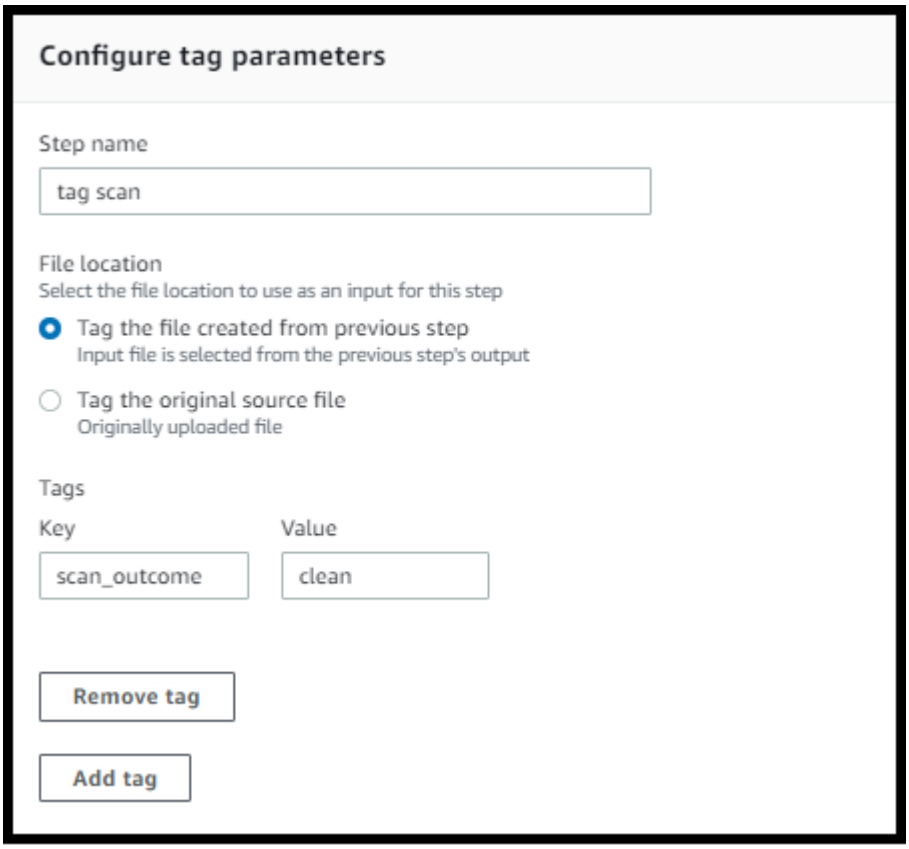
- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.

- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

Tag file

To tag incoming files for further downstream processing, use a tag step. Enter the value of the tag that you would like to assign to the incoming files. Currently, the tag operation is supported only if you are using Amazon S3 for your Transfer Family server storage.

The following example tag step assigns `scan_outcome` and `clean` as the tag key and value, respectively.



To allow a tag step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
            "Sid": "Tag",
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectTagging",
```

```
                "s3:PutObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
}
```

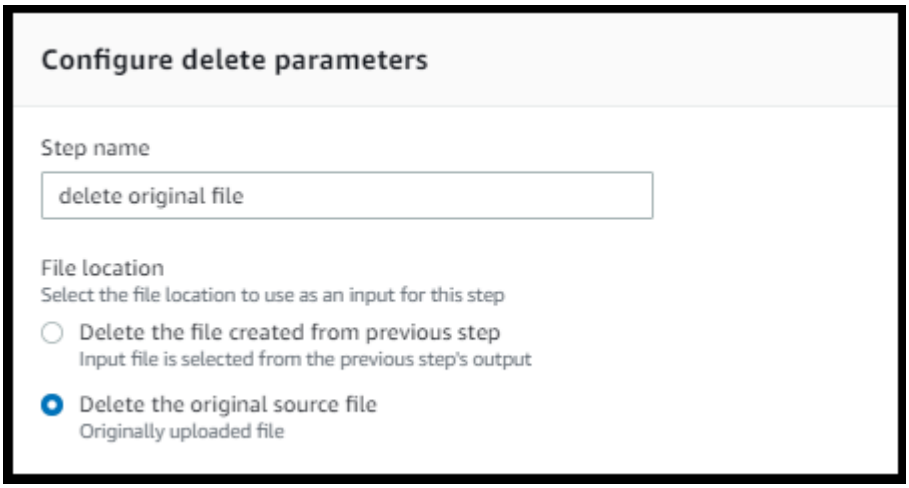
Note

If your workflow contains a tag step that runs before either a copy or decrypt step, you need to add one or two permissions to your IAM policy.

- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.
- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

Delete file

To delete a processed file from a previous workflow step or to delete the originally uploaded file, use a delete file step.



To allow a delete step to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
            "Sid": "Delete",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:DeleteObjectVersion",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:secretsmanager:region:account-ID:secret:aws/transfer/
*"
        }
```

Named variables for workflows

For copy and decrypt steps, you can use a variable to dynamically perform actions. Currently, AWS Transfer Family supports the following named variables.

- Use `${transfer:UserName}` to copy or decrypt files to a destination based on the user who's uploading the files.
- Use `${transfer:UploadDate}` to copy or decrypt files to a destination location based on the current date.

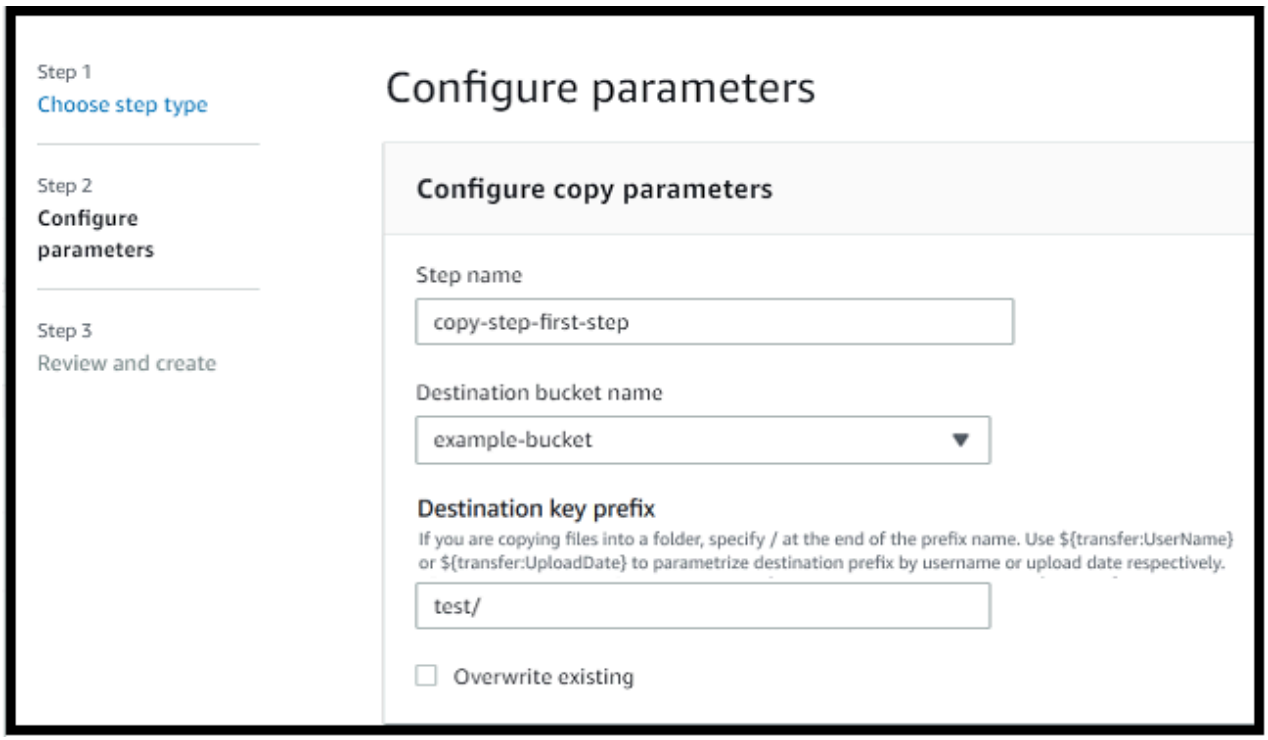
Example tag and delete workflow

The following example illustrates a workflow that tags incoming files that need to be processed by a downstream application, such as a data analytics platform. After tagging the incoming file, the workflow then deletes the originally uploaded file to save on storage costs.

Console

Example tag and move workflow

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Workflows**.
3. On the **Workflows** page, choose **Create workflow**.
4. On the **Create workflow** page, enter a description. This description appears on the **Workflows** page.
5. Add the first step (copy).
    a. In the **Nominal steps** section, choose **Add step**.
    b. Choose **Copy file**, then choose **Next**.
    c. Enter a step name, then select a destination bucket and a key prefix.



d. Choose **Next**, then review the details for the step.

e. Choose **Create step** to add the step and continue.

6. Add the second step (tag).

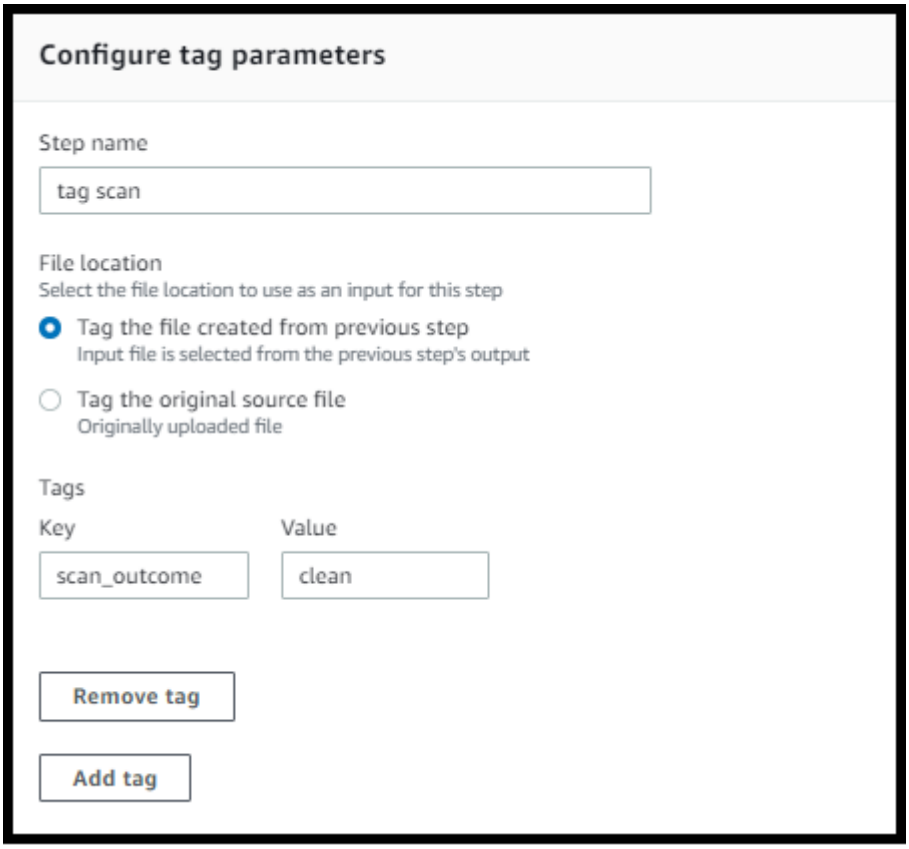
   a. In the **Nominal steps** section, choose **Add step**.

   b. Choose **Tag file**, then choose **Next**.

   c. Enter a step name.

   d. For **File location**, select **Tag the file created from previous step**.

   e. Enter a **Key** and **Value**.



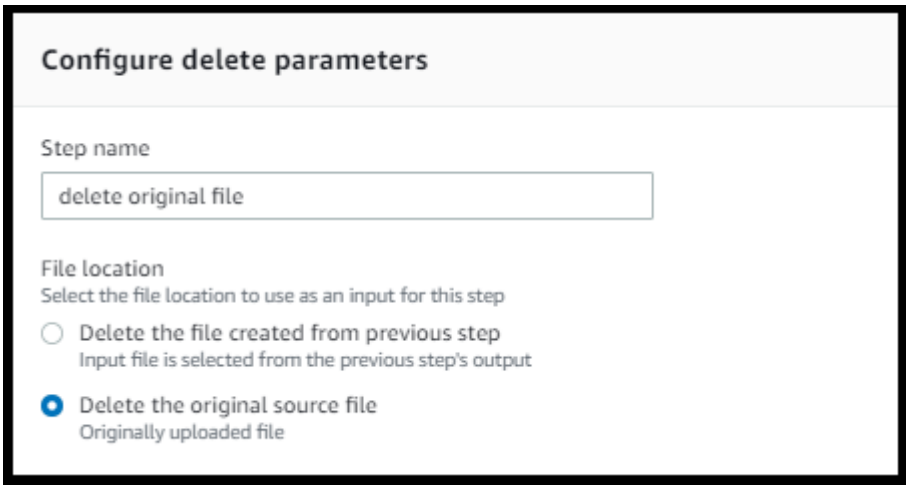
f. Choose **Next**, then review the details for the step.

g. Choose **Create step** to add the step and continue.

7. Add the third step (delete).

a. In the **Nominal steps** section, choose **Add step**.

b. Choose **Delete file**, then choose **Next**.



c. Enter a step name.

d. For **File location**, select **Delete the original source file**.

e. Choose **Next**, then review the details for the step.

f. Choose **Create step** to add the step and continue.

8. Review the workflow configuration, and then choose **Create workflow**.

CLI

Example tag and move workflow

1. Save the following code into a file; for example, `tagAndMoveWorkflow.json`. Replace each *`user input placeholder`* with your own information.

```
[
   {
       "Type": "COPY",
       "CopyStepDetails": {
          "Name": "CopyStep",
          "DestinationFileLocation": {
             "S3FileLocation": {
                "Bucket": "amzn-s3-demo-bucket",
                "Key": "test/"
             }
          }
       }
   },
   {
       "Type": "TAG",
       "TagStepDetails": {
          "Name": "TagStep",
          "Tags": [
             {
                "Key": "name",
                "Value": "demo"
             }
          ],
          "SourceFileLocation": "${previous.file}"
       }
   },
   {
      "Type": "DELETE",
      "DeleteStepDetails":{
```

```
        "Name":"DeleteStep",
        "SourceFileLocation": "${original.file}"
      }
  }
]
```

The first step copies the uploaded file to a new Amazon S3 location. The second step adds a tag (key-value pair) to the file (`previous.file`) that was copied to the new location. And, finally, the third step deletes the original file (`original.file`).

2. Create a workflow from the saved file. Replace each *`user input placeholder`* with your own information.

```
aws transfer create-workflow --description "short-description" --steps
 file://path-to-file --region region-ID
```

For example:

```
aws transfer create-workflow --description "copy-tag-delete workflow" --steps
 file://tagAndMoveWorkflow.json --region us-east-1
```

> **Note**
>
> For more details about using files to load parameters, see How to load parameters from a file.

3. Update an existing server.

> **Note**
>
> This step assumes you already have a Transfer Family server and you want to associate a workflow with it. If not, see Configuring an SFTP, FTPS, or FTP server endpoint. Replace each *`user input placeholder`* with your own information.

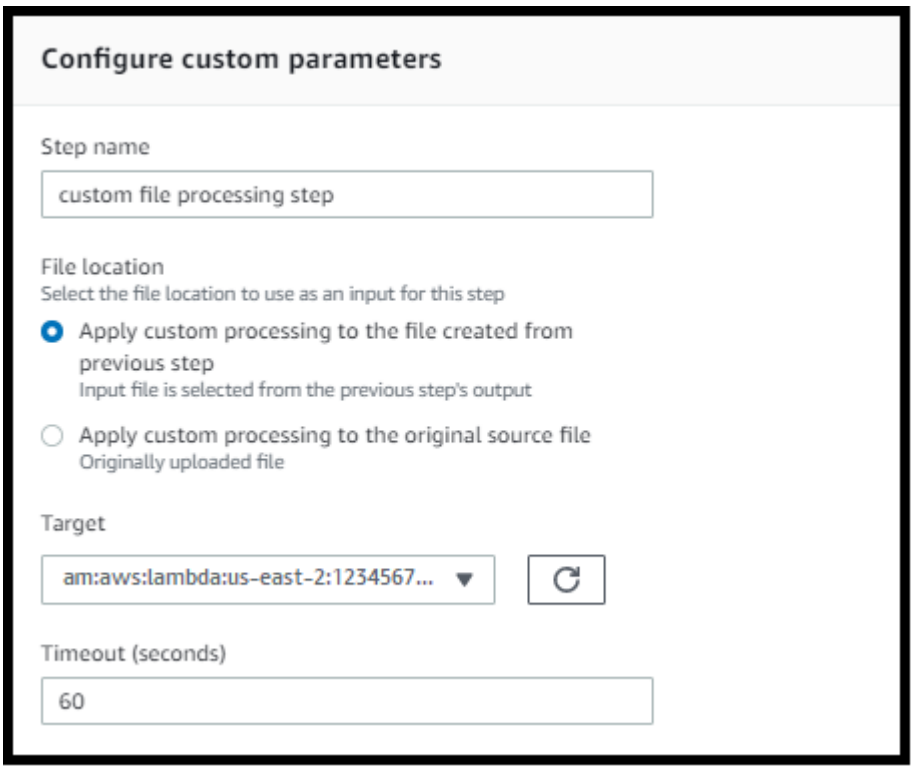
```
aws transfer update-server --server-id server-ID --region region-ID
  --workflow-details '{"OnUpload":[{ "WorkflowId": "workflow-
ID","ExecutionRole": "execution-role-ARN"}]}'
```

For example:

```
aws transfer update-server --server-id s-1234567890abcdef0 --region us-east-2
  --workflow-details '{"OnUpload":[{ "WorkflowId": "w-
abcdef01234567890","ExecutionRole": "arn:aws:iam::111111111111:role/nikki-wolf-
execution-role"}]}'
```

Use custom file-processing steps

By using a custom file-processing step, you can Bring Your Own file-processing logic using AWS Lambda. Upon file arrival, a Transfer Family server invokes a Lambda function that contains custom file-processing logic, such as encrypting files, scanning for malware, or checking for incorrect file types. In the following example, the target AWS Lambda function is used to process the output file from the previous step.



> **Note**
>
> For an example Lambda function, see Example Lambda function for a custom workflow step. For example events (including the location for files passed into the Lambda), see Example events sent to AWS Lambda upon file upload.

With a custom workflow step, you must configure the Lambda function to call the SendWorkflowStepState API operation. `SendWorkflowStepState` notifies the workflow execution that the step was completed with either a success or a failure status. The status of the `SendWorkflowStepState` API operation invokes an exception handler step or a nominal step in the linear sequence, based on the outcome of the Lambda function.

If the Lambda function fails or times out, the step fails, and you see `StepErrored` in your CloudWatch logs. If the Lambda function is part of the nominal step and the function responds to `SendWorkflowStepState` with `Status="FAILURE"` or times out, the flow continues with the exception handler steps. In this case, the workflow does not continue to execute the remaining (if any) nominal steps. For more details, see Exception handling for a workflow.

When you call the `SendWorkflowStepState` API operation, you must send the following parameters:

```
{
    "ExecutionId": "string",
    "Status": "string",
    "Token": "string",
    "WorkflowId": "string"
}
```

You can extract the `ExecutionId`, `Token`, and `WorkflowId` from the input event that is passed when the Lambda function executes (examples are shown in the following sections). The `Status` value can be either `SUCCESS` or `FAILURE`.

To be able to call the `SendWorkflowStepState` API operation from your Lambda function, you must use a version of the AWS SDK that was published after Managed Workflows were introduced.

Using multiple Lambda functions consecutively

When you use multiple custom steps one after the other, the **File location** option works differently than if you use only a single custom step. Transfer Family doesn't support passing the Lambda-

processed file back to use as the next step's input. So, if you have multiple custom steps all configured to use the `previous.file` option, they all use the same file location (the input file location for the first custom step).

> **Note**
>
> The `previous.file` setting also works differently if you have a predefined step (tag, copy, decrypt, or delete) after a custom step. If the predefined step is configured to use the `previous.file` setting, the predefined step uses the same input file that's used by the custom step. The processed file from the custom step is not passed to the predefined step.

Accessing a file after custom processing

If you're using Amazon S3 as your storage, and if your workflow includes a custom step that performs actions on the originally uploaded file, subsequent steps cannot access that processed file. That is, any step after the custom step cannot reference the updated file from the custom step output.

For example, suppose that you have the following three steps in your workflow.

- **Step 1** – Upload a file named `example-file.txt`.
- **Step 2** – Invoke a Lambda function that changes `example-file.txt` in some way.
- **Step 3** – Attempt to perform further processing on the updated version of `example-file.txt`.

If you configure the `sourceFileLocation` for Step 3 to be `${original.file}`, Step 3 uses the original file location from when the server uploaded the file to storage in Step 1. If you're using `${previous.file}` for Step 3, Step 3 reuses the file location that Step 2 used as input.

Therefore, Step 3 causes an error. For example, if step 3 attempts to copy the updated `example-file.txt`, you receive the following error:

```
{
    "type": "StepErrored",
    "details": {
        "errorType": "NOT_FOUND",
        "errorMessage": "ETag constraint not met (Service: null; Status Code: 412;
 Error Code: null; Request ID: null; S3 Extended Request ID: null; Proxy: null)",
        "stepType": "COPY",
```

```
        "stepName": "CopyFile"
    },
```

This error occurs because the custom step modifies the entity tag (ETag) for `example-file.txt` so that it doesn't match the original file.

> **Note**
>
> This behavior doesn't occur if you're using Amazon EFS because Amazon EFS doesn't use entity tags to identify files.

Example events sent to AWS Lambda upon file upload

The following examples show the events that are sent to AWS Lambda when a file upload is complete. One example uses a Transfer Family server where the domain is configured with Amazon S3. The other example uses a Transfer Family server where the domain uses Amazon EFS.

Custom step that uses an Amazon S3 domain

```
{
    "token": "MzI0Nzc4ZDktMGRmMi00MjFhLTgxMjUtYWZmZmRmODNkYjc0",
    "serviceMetadata": {
        "executionDetails": {
            "workflowId": "w-1234567890example",
            "executionId": "abcd1234-aa11-bb22-cc33-abcdef123456"
        },
        "transferDetails": {
            "sessionId": "36688ff5d2deda8c",
            "userName": "myuser",
            "serverId": "s-example1234567890"
        }
    },
    "fileLocation": {
        "domain": "S3",
        "bucket": "amzn-s3-demo-bucket",
        "key": "path/to/mykey",
        "eTag": "d8e8fca2dc0f896fd7cb4cb0031ba249",
        "versionId": null
    }
}
```

Custom step that uses an Amazon EFS domain

```
{
    "token": "MTg0N2Y3N2UtNWI5Ny00ZmZlLTk5YTgtZTU3YzViYjllNmZm",
    "serviceMetadata": {
        "executionDetails": {
            "workflowId": "w-1234567890example",
            "executionId": "abcd1234-aa11-bb22-cc33-abcdef123456"
        },
        "transferDetails": {
            "sessionId": "36688ff5d2deda8c",
            "userName": "myuser",
            "serverId": "s-example1234567890"
        }
    },
    "fileLocation": {
        "domain": "EFS",
        "fileSystemId": "fs-1234567",
        "path": "/path/to/myfile"
    }
}
```

Example Lambda function for a custom workflow step

The following Lambda function extracts the information regarding the execution status, and then calls the SendWorkflowStepState API operation to return the status to the workflow for the step—either SUCCESS or FAILURE. Before your function calls the `SendWorkflowStepState` API operation, you can configure Lambda to take an action based on your workflow logic.

```
import json
import boto3

transfer = boto3.client('transfer')

def lambda_handler(event, context):
    print(json.dumps(event))

    # call the SendWorkflowStepState API to notify the workflow about the step's
 SUCCESS or FAILURE status
    response = transfer.send_workflow_step_state(
        WorkflowId=event['serviceMetadata']['executionDetails']['workflowId'],
```

```
        ExecutionId=event['serviceMetadata']['executionDetails']['executionId'],
        Token=event['token'],
        Status='SUCCESS|FAILURE'
    )

    print(json.dumps(response))

    return {
      'statusCode': 200,
      'body': json.dumps(response)
    }
```

IAM permissions for a custom step

To allow a step that calls a Lambda to succeed, make sure the execution role for your workflow contains the following permissions.

```
{
    "Sid": "Custom",
    "Effect": "Allow",
    "Action": [
        "lambda:InvokeFunction"
    ],
    "Resource": [
        "arn:aws:lambda:region:account-id:function:function-name"
    ]
}
```

IAM policies for workflows

When you add a workflow to a server, you must select an execution role. The server uses this role when it executes the workflow. If the role does not have the proper permissions, AWS Transfer Family cannot run the workflow.

This section describes one possible set of AWS Identity and Access Management (IAM) permissions that you can use to execute a workflow. Other examples are described later in this topic.

> **Note**
>
> If your Amazon S3 files have tags, you need to add one or two permissions to your IAM policy.

- Add `s3:GetObjectTagging` for an Amazon S3 file that isn't versioned.
- Add `s3:GetObjectVersionTagging` for an Amazon S3 file that is versioned.

To create an execution role for your workflow

1. Create a new IAM role, and add the AWS managed policy `AWSTransferFullAccess` to the role. For more information about creating a new IAM role, see the section called “Create an IAM role and policy”.
2. Create another policy with the following permissions, and attach it to your role. Replace each *`user input placeholder`* with your own information.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConsoleAccess",
            "Effect": "Allow",
            "Action": "s3:GetBucketLocation",
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Sid": "GetObjectVersion",
            "Effect": "Allow",
```

```
            "Action": "s3:GetObjectVersion",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Sid": "Custom",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-name"
            ]
        },
        {
            "Sid": "Tag",
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectTagging",
                "s3:PutObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

3. Save this role and specify it as the execution role when you add a workflow to a server.

> **Note**
>
> When you're constructing IAM roles, AWS recommends that you restrict access to your resources as much as is possible for your workflow.

Workflow trust relationships

Workflow execution roles also require a trust relationship with `transfer.amazonaws.com`. To establish a trust relationship for AWS Transfer Family, see To establish a trust relationship.

While you're establishing your trust relationship, you can also take steps to avoid the *confused deputy* problem. For a description of this problem, as well as examples of how to avoid it, see the section called “Cross-service confused deputy prevention”.

Example execution role: Decrypt, copy, and tag

If you have workflows that include tagging, copying, and decrypt steps, you can use the following IAM policy. Replace each *`user input placeholder`* with your own information.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CopyRead",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
        },
        {
            "Sid": "CopyWrite",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectTagging"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
        },
        {
            "Sid": "CopyList",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            ]
        },
        {
            "Sid": "Tag",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectTagging",
                "s3:PutObjectVersionTagging"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:RequestObjectTag/Archive": "yes"
                }
            }
        },
        {
            "Sid": "ListBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            ]
        },
        {
            "Sid": "HomeDirObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObjectVersion",
                "s3:DeleteObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
        },
        {
            "Sid": "Decrypt",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:us-east-1:123456789012:secret:aws/
transfer/*"
        }
    ]
}
```

Example execution role: Run function and delete

In this example, you have a workflow that invokes an AWS Lambda function. If the workflow deletes the uploaded file and has an exception handler step to act upon a failed workflow execution in the previous step, use the following IAM policy. Replace each *`user input placeholder`* with your own information.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Delete",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
            ],
            "Resource": "arn:aws:s3:::bucket-name"
        },
        {
            "Sid": "Custom",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-name"
            ]
        }
    ]
}
```

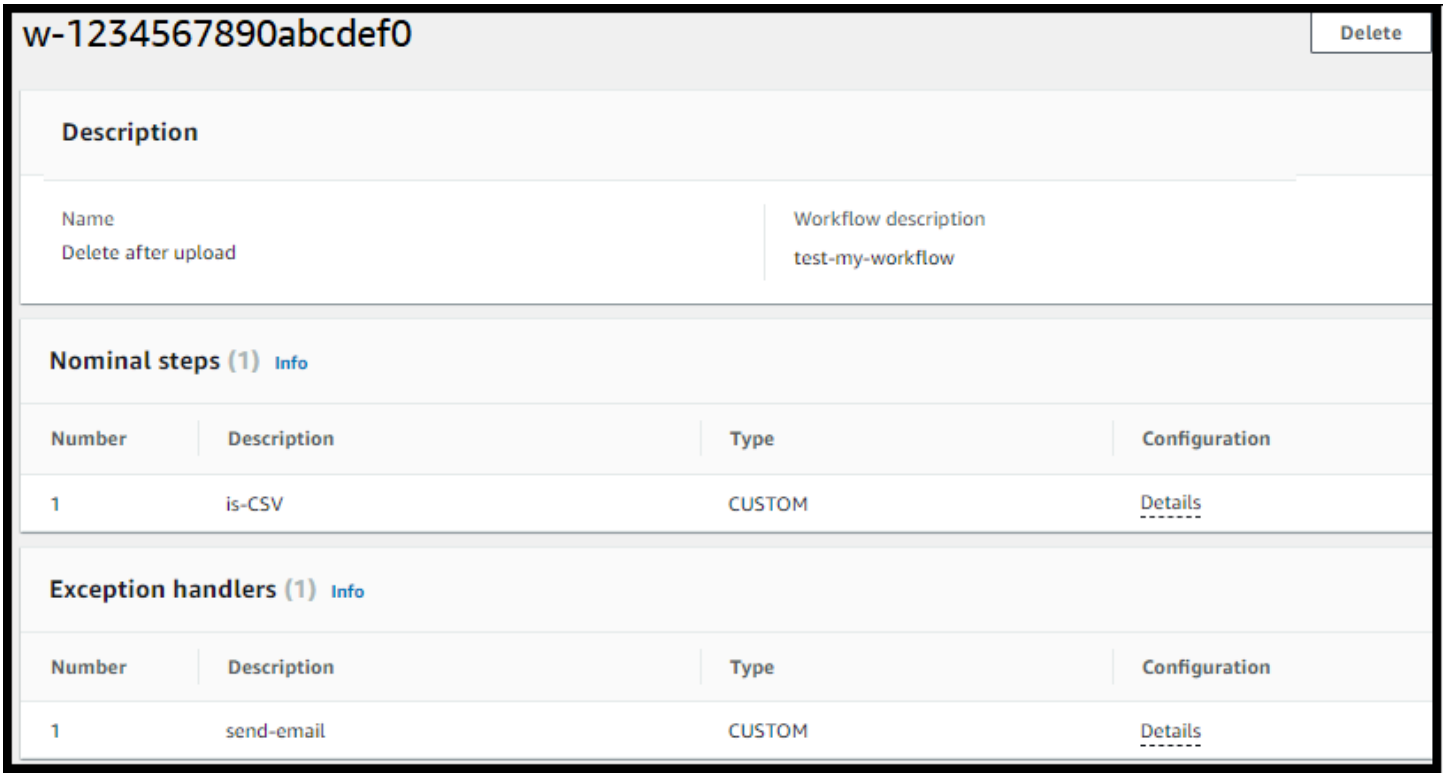
Exception handling for a workflow

If any errors occur during a workflow's execution, the exception-handling steps that you specified are executed. You specify the error-handling steps for a workflow in the same manner as you specify the nominal steps for the workflow. For example, suppose that you've configured custom processing in nominal steps to validate incoming files. If the file validation fails, an exception-handling step can send an email to the administrator.

The following example workflow contains two steps:

- One nominal step that checks whether the uploaded file is in CSV format
- An exception-handling step that sends an email in case the uploaded file is not in CSV format, and the nominal step fails

To initiate the exception-handling step, the AWS Lambda function in the nominal step must respond with `Status="FAILURE"`. For more information about error handling in workflows, see the section called “Use custom file-processing steps”.



Monitor workflow execution

Amazon CloudWatch monitors your AWS resources and the applications that you run in the AWS Cloud in real time. You can use Amazon CloudWatch to collect and track metrics, which are variables that you can measure for your workflows. You can view workflow metrics and consolidated logs by using Amazon CloudWatch.

CloudWatch logging for a workflow

CloudWatch provides consolidated auditing and logging for workflow progress and results.

AWS Transfer Family provides two ways to deliver managed workflow execution logs to Amazon CloudWatch:

- **A log group that you specify (structured log destination)** – You specify a CloudWatch Logs log group as a structured log destination for the workflow, and Transfer Family delivers the workflow execution logs to that log group for you. This option does not require a logging role. To use it, provide the Amazon Resource Name (ARN) of a log group in the `StructuredLogDestinations` parameter when you create the workflow (see StructuredLogDestinations in the *AWS Transfer Family API Reference*). The log group must be in the same AWS account and AWS Region as the workflow.

  To configure a structured log destination, the IAM policy for the principal that creates the workflow must contain the following permissions:

  - `logs:CreateLogDelivery`
  - `logs:DeleteLogDelivery`
  - `logs:DescribeLogGroups`
  - `logs:DescribeResourcePolicies`
  - `logs:GetLogDelivery`
  - `logs:ListLogDeliveries`
  - `logs:PutResourcePolicy`
  - `logs:UpdateLogDelivery`
- **A logging role** – You attach an IAM logging role to the server that the workflow is attached to. Transfer Family assumes the role to write the workflow execution logs to a log group named `/aws/transfer/`*`server-id`*. For details, see Configure CloudWatch logging role.

Both mechanisms write the same workflow execution log content in the same JSON format. The difference is only where the logs are delivered: to the log group that you specify, or to the `/aws/transfer/`*`server-id`* log group on the server by way of the logging role.

View Amazon CloudWatch logs for workflows

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Logs**, then choose **Log groups**.
3. On the **Log groups** page, on the navigation bar, choose the correct Region for your AWS Transfer Family server.

4. Choose the log group that contains your workflow logs.

   - If the workflow uses a structured log destination, choose the log group whose ARN you specified in the `StructuredLogDestinations` parameter of the workflow.
   - If the workflow uses a logging role on the server, choose the log group that corresponds to your server. For example, if your server ID is `s-1234567890abcdef0`, your log group is `/aws/transfer/s-1234567890abcdef0`.

5. On the log group details page, the most recent log streams are displayed. The log stream name for the workflow depends on which logging mechanism the workflow uses:

   - For a structured log destination, the workflow log stream is named *`workflowID`*`/`*`executionID`*. For example:

     ```
     w-abcdef01234567890/021345abcdef6789
     ```

   - For a logging role, the server log group contains two log streams for the user that you are exploring: one for each Secure Shell (SSH) File Transfer Protocol (SFTP) session, and one for the workflow that is being executed. The format for the workflow log stream is *`username`*`.`*`workflowID`*`.`*`uniqueStreamSuffix`*.

   For example, if your user is `mary-major`, you have the following log streams:

   ```
   mary-major-east.1234567890abcdef0
   mary.w-abcdef01234567890.021345abcdef6789
   ```

   > **Note**
   >
   > The 16-digit alphanumeric identifiers listed in this example are fictitious. The values that you see in Amazon CloudWatch are different.

The **Log events** page for a workflow log stream contains the details for the workflow execution. If the workflow uses a logging role, the `mary-major-usa-east.1234567890abcdef0` stream displays the details for each user session, and the `mary.w-abcdef01234567890.021345abcdef6789` stream contains the details for the workflow. If the workflow uses a structured log destination, the `w-abcdef01234567890/021345abcdef6789` stream contains the details for the workflow.

The following is a sample log stream for `mary.w-abcdef01234567890.021345abcdef6789`, based on a workflow (`w-abcdef01234567890`) that contains a copy step.

```
{
    "type": "ExecutionStarted",
    "details": {
        "input": {
            "initialFileLocation": {
                "bucket": "amzn-s3-demo-bucket",
                "key": "mary/workflowSteps2.json",
                "versionId": "version-id",
                "etag": "etag-id"
            }
        }
    },
    "workflowId":"w-abcdef01234567890",
    "executionId":"execution-id",
    "transferDetails": {
        "serverId":"s-server-id",
        "username":"mary",
        "sessionId":"session-id"
    }
},
{
    "type":"StepStarted",
    "details": {
        "input": {
            "fileLocation": {
                "backingStore":"S3",
                "bucket":"amzn-s3-demo-bucket",
                "key":"mary/workflowSteps2.json",
                "versionId":"version-id",
                "etag":"etag-id"
            }
        },
        "stepType":"COPY",
        "stepName":"copyToShared"
    },
    "workflowId":"w-abcdef01234567890",
    "executionId":"execution-id",
    "transferDetails": {
        "serverId":"s-server-id",
        "username":"mary",
```

```
        "sessionId":"session-id"
    }
},
{
    "type":"StepCompleted",
    "details":{
        "output":{},
        "stepType":"COPY",
        "stepName":"copyToShared"
    },
    "workflowId":"w-abcdef01234567890",
    "executionId":"execution-id",
    "transferDetails":{
        "serverId":"server-id",
        "username":"mary",
        "sessionId":"session-id"
    }
},
{
    "type":"ExecutionCompleted",
    "details": {},
    "workflowId":"w-abcdef01234567890",
    "executionId":"execution-id",
    "transferDetails":{
        "serverId":"s-server-id",
        "username":"mary",
        "sessionId":"session-id"
    }
}
```

CloudWatch metrics for workflows

AWS Transfer Family provides several metrics for workflows. You can view metrics for how many workflows executions started, completed successfully, and failed in the previous minute. All of the CloudWatch metrics for Transfer Family are described in Using CloudWatch metrics for Transfer Family servers.

Create a workflow from a template

You can deploy an CloudFormation stack that creates a workflow and a server from a template. This procedure contains an example that you can use to quickly deploy a workflow.

To create an CloudFormation stack that creates an AWS Transfer Family workflow and server

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Save the following code to a file.

   YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  SFTPServer:
    Type: 'AWS::Transfer::Server'
    Properties:
      WorkflowDetails:
        OnUpload:
          - ExecutionRole: workflow-execution-role-arn
            WorkflowId: !GetAtt
              - TransferWorkflow
              - WorkflowId
  TransferWorkflow:
    Type: AWS::Transfer::Workflow
    Properties:
      Description: Transfer Family Workflows Blog
      Steps:
        - Type: COPY
          CopyStepDetails:
            Name: copyToUserKey
            DestinationFileLocation:
              S3FileLocation:
                Bucket: archived-records
                Key: ${transfer:UserName}/
            OverwriteExisting: 'TRUE'
        - Type: TAG
          TagStepDetails:
            Name: tagFileForArchive
            Tags:
              - Key: Archive
                Value: yes
        - Type: CUSTOM
          CustomStepDetails:
            Name: transferExtract
            Target: arn:aws:lambda:region:account-id:function:function-name
            TimeoutSeconds: 60
        - Type: DELETE
```

```
          DeleteStepDetails:
            Name: DeleteInputFile
            SourceFileLocation: '${original.file}'
      Tags:
        - Key: Name
          Value: TransferFamilyWorkflows
```

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "SFTPServer": {
            "Type": "AWS::Transfer::Server",
            "Properties": {
                "WorkflowDetails": {
                    "OnUpload": [
                        {
                            "ExecutionRole": "workflow-execution-role-arn",
                            "WorkflowId": {
                                "Fn::GetAtt": [
                                    "TransferWorkflow",
                                    "WorkflowId"
                                ]
                            }
                        }
                    ]
                }
            }
        },
        "TransferWorkflow": {
            "Type": "AWS::Transfer::Workflow",
            "Properties": {
                "Description": "Transfer Family Workflows Blog",
                "Steps": [
                    {
                        "Type": "COPY",
                        "CopyStepDetails": {
                            "Name": "copyToUserKey",
                            "DestinationFileLocation": {
                                "S3FileLocation": {
                                    "Bucket": "archived-records",
                                    "Key": "${transfer:UserName}/"
```

```
                                }
                            },
                            "OverwriteExisting": "TRUE"
                        }
                    },
                    {
                        "Type": "TAG",
                        "TagStepDetails": {
                            "Name": "tagFileForArchive",
                            "Tags": [
                                {
                                    "Key": "Archive",
                                    "Value": "yes"
                                }
                            ]
                        }
                    },
                    {
                        "Type": "CUSTOM",
                        "CustomStepDetails": {
                            "Name": "transferExtract",
                            "Target": "arn:aws:lambda:region:account-
id:function:function-name",
                            "TimeoutSeconds": 60
                        }
                    },
                    {
                        "Type": "DELETE",
                        "DeleteStepDetails": {
                            "Name": "DeleteInputFile",
                            "SourceFileLocation": "${original.file}"
                        }
                    }
                ],
                "Tags": [
                    {
                        "Key": "Name",
                        "Value": "TransferFamilyWorkflows"
                    }
                ]
            }
        }
    }
```

```
}
```

3. Replace the following items with your actual values.

   - Replace *workflow-execution-role-arn* with the ARN for an actual workflow execution role. For example, `arn:aws:transfer:us-east-2:111122223333:workflow/w-1234567890abcdef0`
   - Replace `arn:aws:lambda:`*region*`:`*account-id*`:function:`*function-name* with the ARN for your Lambda function. For example, `arn:aws:lambda:us-east-2:123456789012:function:example-lambda-idp`.

4. Follow the instructions for deploying an CloudFormation stack from an existing template in Selecting a stack template in the *AWS CloudFormation User Guide*.

After the stack has been deployed, you can view details about it in the **Outputs** tab in the CloudFormation console. The template creates a new AWS Transfer Family SFTP server that uses service-managed users, and a new workflow, and associates the workflow with the new server.

Remove a workflow from a Transfer Family server

If you have associated a workflow with a Transfer Family server, and you now want to remove that association, you can do so by using the console or programmatically.

Console

To remove a workflow from a Transfer Family server

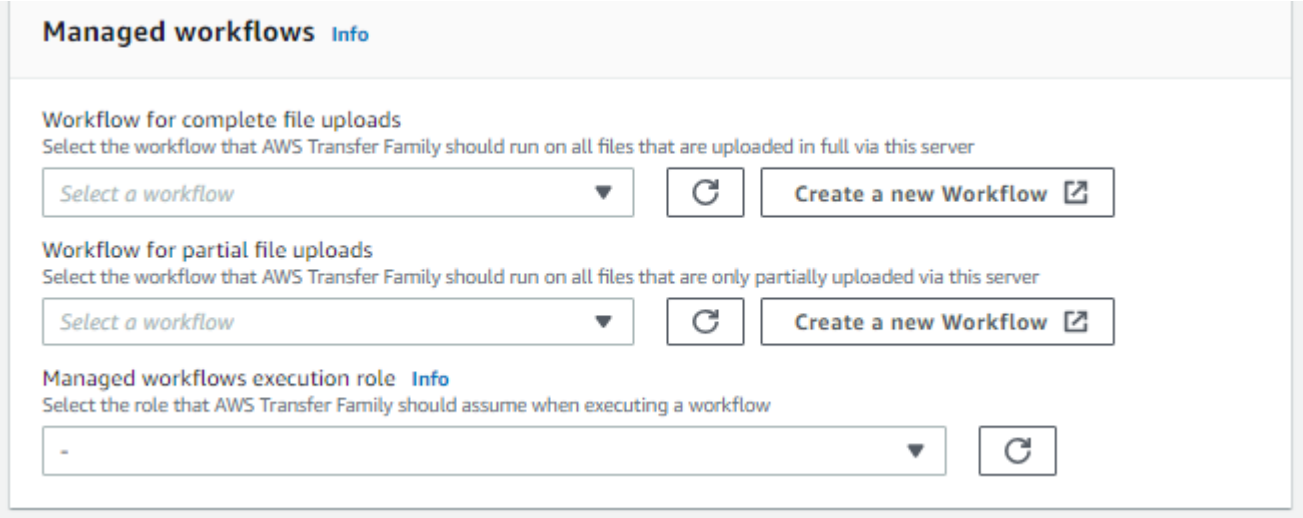
1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**.
3. Choose the identifier for the server in the **Server ID** column.
4. On the details page for the server, scroll down to the **Additional details** section, and then choose **Edit**.
5. On the **Edit additional details** page, in the **Managed workflows** section, clear the information for all settings:

   - Select the dash (-) from the list of workflows for the **Workflow for complete file uploads**.

- If not already cleared, select the dash (-) from the list of workflows for the **Workflow for partial file uploads**.
- Select the dash (-) from the list of roles for the **Managed workflows execution role**.

If you don't see the dash, scroll up until you see it, as it is the first value in each menu.

The screen should look like the following.



6. Scroll down and choose **Save** to save your changes.

CLI

You use the `update-server` (or `UpdateServer` for API) call, and provide empty arguments for the `OnUpload` and `OnPartialUpload` parameters.

From the AWS CLI, run the following command:

```
aws transfer update-server --server-id your-server-id --workflow-details
 '{"OnPartialUpload":[],"OnUpload":[]}'
```

Replace *`your-server-id`* with the ID for your server. For example, if your server ID is, `s-01234567890abcdef`, the command is as follows:

```
aws transfer update-server --server-id s-01234567890abcdef --workflow-details
 '{"OnPartialUpload":[],"OnUpload":[]}'
```

Managed workflows restrictions and limitations

Restrictions

The following restrictions currently apply to post-upload processing workflows for AWS Transfer Family.

- Cross-account and cross-region AWS Lambda functions are not supported. You can, however, copy across accounts, provided that your AWS Identity and Access Management (IAM) policies are correctly configured.
- For all workflow steps, any Amazon S3 buckets accessed by the workflow must be in the same region as the workflow itself.
- For a decryption step, the decryption destination must match the source for Region and backing store (for example, if the file to be decrypted is stored in Amazon S3, then the specified destination must also be in Amazon S3).
- Only asynchronous custom steps are supported.
- Custom step timeouts are approximate. That is, it might take slightly longer to time out than specified. Additionally, the workflow is dependent upon the Lambda function. Therefore, if the function is delayed during execution, the workflow is not aware of the delay.
- If you exceed your throttling limit, Transfer Family doesn't add workflow operations to the queue.
- Workflows are not initiated for files that have a size of 0. Files with a size greater than 0 do initiate the associated workflow.
- You can attach a file-processing workflow to a Transfer Family server that uses the AS2 protocol: however, AS2 messages don't execute workflows attached to the server.

Limitations

Additionally, the following functional limits apply to workflows for Transfer Family:

- The number of workflows per Region, per account, is limited to 10.
- The maximum timeout for custom steps is 30 minutes.
- The maximum number of steps in a workflow is 8.
- The maximum number of tags per workflow is 50.
- The maximum number of concurrent executions that contain a decrypt step is 250 per workflow.

- You can store a maximum of 3 PGP private keys, per Transfer Family server, per user.
- The maximum size for a decrypted file is 10 GB.
- We throttle the new execution rate using a token bucket system with a burst capacity of 100 and a refill rate of 1.
- Anytime you remove a workflow from a server and replace it with a new one, or update server configuration (which impacts a workflow's execution role), you must wait approximately 10 minutes before executing the new workflow. The Transfer Family server caches the workflow details, and it takes 10 minutes for the server to refresh its cache.

  Additionally, you must log out of any active SFTP sessions, and then log back in after the 10-minute waiting period to see the changes.

Managing servers

In this section, you can find information on how to view a list of your servers, how to view your server details, how to edit your server details, and how to change the host key for your SFTP-enabled server.

Topics

- View a list of servers
- Delete a server
- View SFTP, FTPS, and FTP server details
- View AS2 server details
- IPv6 support for Transfer Family servers
- Edit server details
- Edit identity provider configuration
- Managing host keys for your SFTP-enabled server
- Monitoring usage in the console

View a list of servers

On the AWS Transfer Family console, you can find a list of all your servers that are located in the AWS Region that you chose.

To find a list of your servers that exist in an AWS Region

- Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.

  If you have one or more servers in the current AWS Region, the console opens to show a list of your servers. If you don't see a list of servers, make sure that you are in the correct Region. You can also choose **Servers** from the navigation pane.

  For more information about viewing your server details, see View SFTP, FTPS, and FTP server details.

Delete a server

This procedure explains how to delete a Transfer Family server by using the AWS Transfer Family console or AWS CLI.

> **Important**
>
> You are billed, for each of the protocols enabled to access your endpoint, until you delete the server.

> **Warning**
>
> Deleting a server results in all its users being deleted. Data in the bucket that was accessed by using the server is not deleted, and remains accessible to AWS users that have privileges to those Amazon S3 buckets.

Console

To delete a server by using the console

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**.
3. Select the check box of the server that you want to delete.
4. For **Actions**, choose **Delete**.
5. In the confirmation dialog box that appears, enter the word **`delete`**, and then choose **Delete** to confirm that you want to delete the server.

The server is deleted from the **Servers** page and you are no longer billed for it.

AWS CLI

To delete a server by using the CLI

1. (Optional) Run the following command to view the details for the server that you want to delete permanently.

```
aws transfer describe-server --server-id your-server-id
```

This `describe-server` command returns all of the details for your server.

2. Run the following command to delete the server.

```
aws transfer delete-server --server-id your-server-id
```

If successful, the command deletes the server and does not return any information.

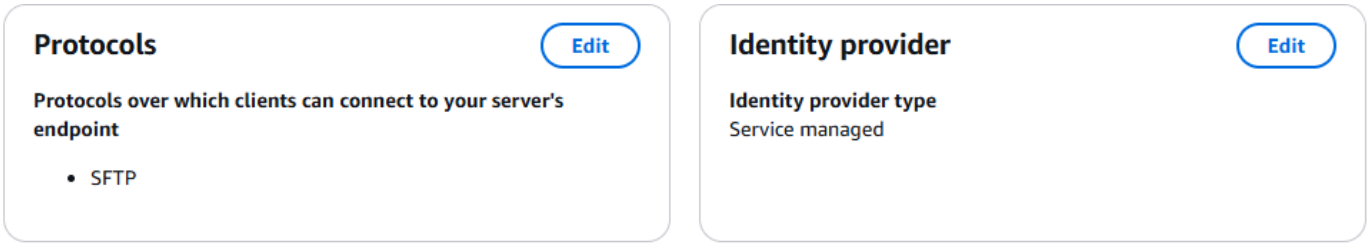
View SFTP, FTPS, and FTP server details

You can find a list of details and properties for an individual AWS Transfer Family server. Server properties include protocols, identity provider, status, endpoint type, custom hostname, endpoint, users, logging role, server host key, and tags.

To view server details

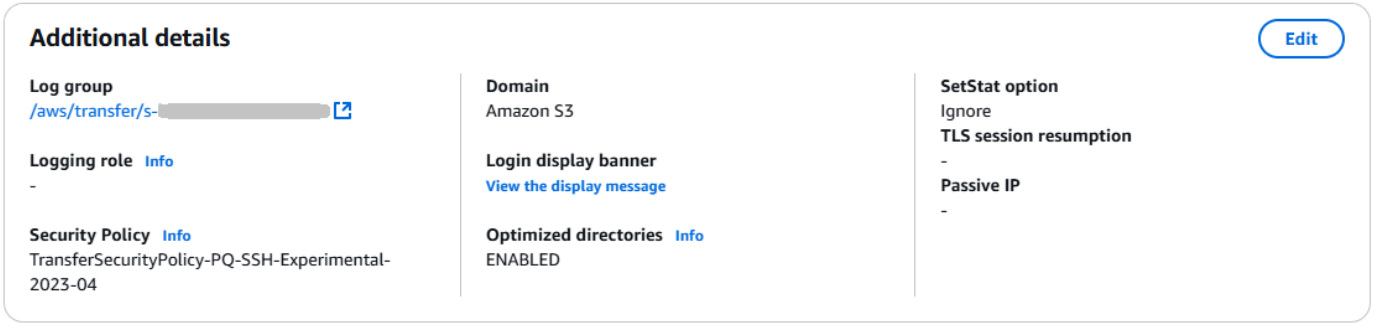
1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Choose the identifier in the **Server ID** column to see the **Server details** page, shown following.

   You can change the server's properties on this page by choosing **Edit**. For more information about editing server details, see Edit server details. The details page for AS2 servers differs slightly. For AS2 servers, see View AS2 server details.



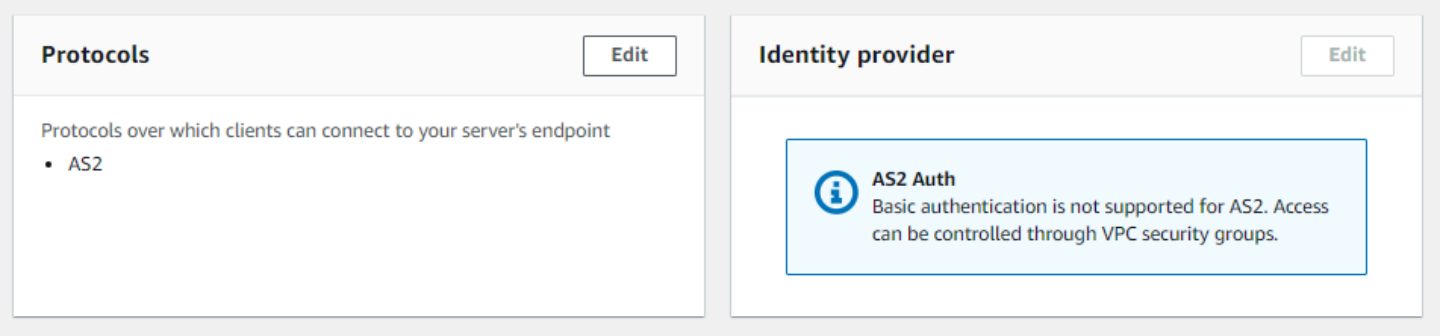
> **Note**
>
> The server host key **Description** and **Date imported** values are new as of September 2022. These values were introduced to support the multiple host keys feature.

This feature required migration of any single host keys that were in use before the introduction of multiple host keys.
The **Date imported** value for a migrated server host key is set to the last modified date for the server. That is, the date that you see for your migrated host key corresponds to the date that you last modified the server in any way, before the server host key migration.
The only key that was migrated is your oldest or only server host key. Any additional keys have their actual date from when you imported them. Additionally, the migrated key has a description that makes it easy to identify it as having been migrated.
The migration occurred between September 2 and September 13. The actual migration date within this range depends on the Region of your server.

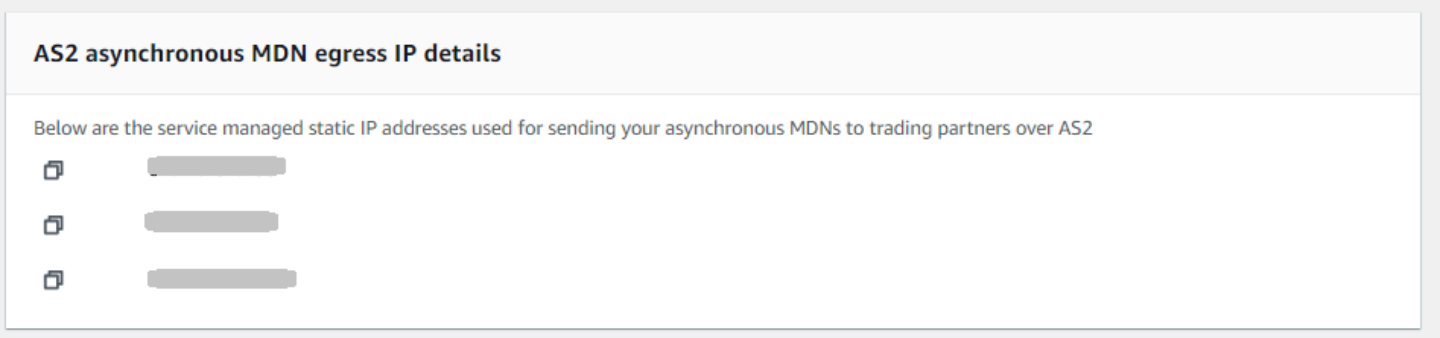


View AS2 server details

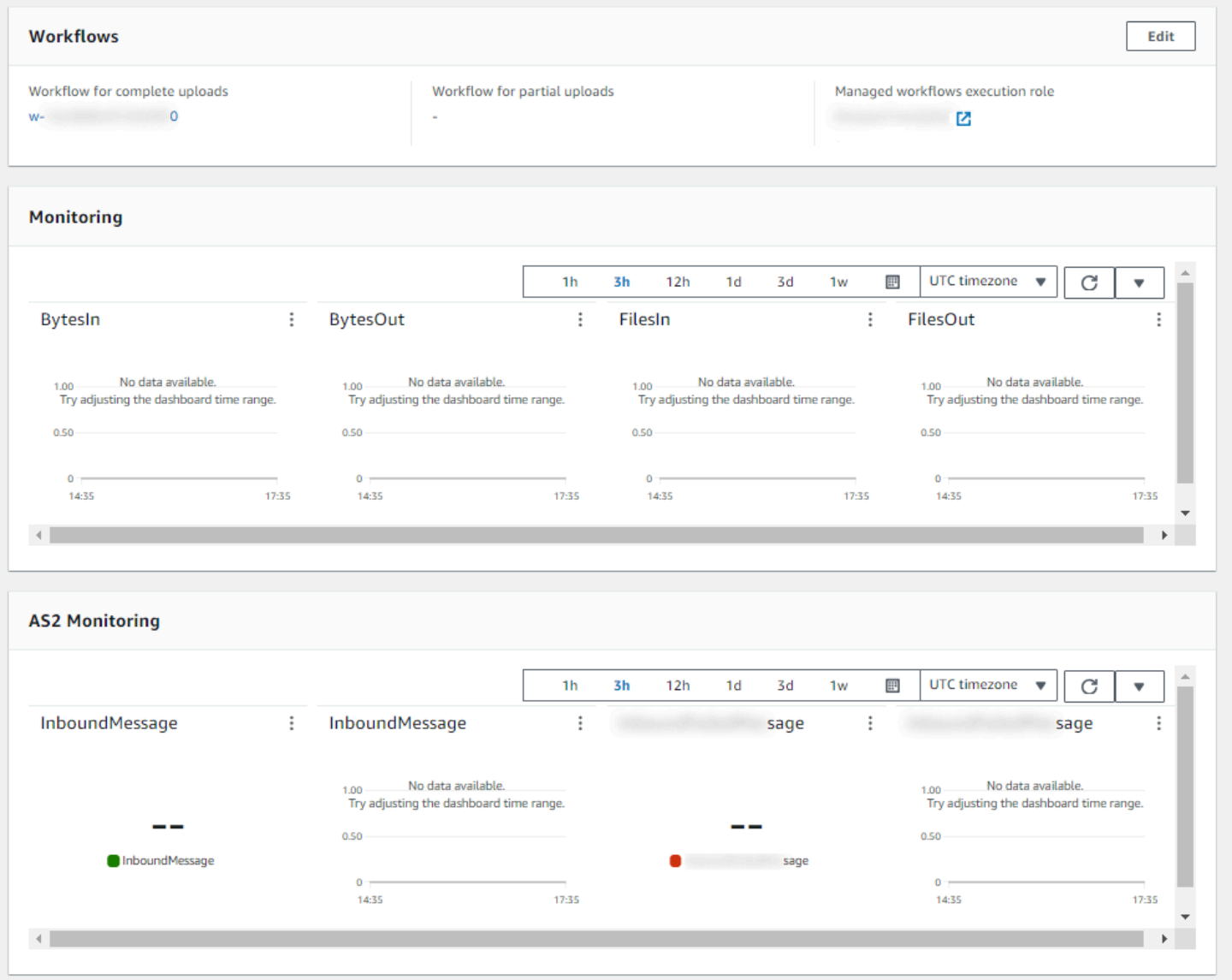
You can find a list of details and properties for an individual AWS Transfer Family server. Server properties include protocols, status, and more. For AS2 servers, you can also view the AS2 asynchronous MDN egress IP addresses.



Each AS2 server is assigned three static IP addresses. Use these IP addresses for sending asynchronous MDNs to your trading partners over AS2.



The bottom portion of the AS2 server details page contains details for any attached workflow and monitoring and tagging information.



IPv6 support for Transfer Family servers

AWS Transfer Family supports dual-stack (IPv4 and IPv6) endpoints for the following resources:

- SFTP public endpoints
- VPC internal endpoints for all protocols (SFTP/FTPS/FTP and AS2)
- Public endpoints for AS2-enabled Transfer Family servers, by using the steps provided in Using an Application Load Balancer for dual-stack AS2 server connectivity
- API endpoints

With dual-stack support, your Transfer Family endpoints can communicate with both IPv4 and IPv6 enabled clients. This enables you to gradually transition from IPv4 to IPv6 based systems without needing to switch all at once, meet IPv6 compliance requirements, and remove the need for expensive networking equipment to handle address translation between IPv4 and IPv6. For details, see DNS and Endpoints in the *AWS Transfer Family API Reference*. For a complete list of available endpoints, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*.

IPv6 limitations

The following Transfer Family resources do not currently support IPv6:

- VPC-Internet endpoints
- VPC_ENDPOINT endpoint type (deprecated)

The FTPS protocol supports the PASV and EPSV commands for requesting an open data port for file listing, getting, and putting operations. However, PASV doesn't work with IPv6 because it requires an IPv4-specific response. EPSV continues to work because it returns only port information.

To use FTPS, we recommend one of the following:

- Configure your FTPS client to use EPSV
- Use IPv4 instead of IPv6

SFTP supports both IPv4 and IPv6. We recommend using SFTP instead of FTPS when working with dual-stack endpoints.

Configuring IPv6 for servers

When creating a new server or updating an existing server, you can choose the IP address type:

- **IPv4** (default): For backwards compatibility, the server will only accept IPv4 connections.
- **Dual-stack**: The server will accept both IPv4 and IPv6 connections.

To update an existing server's IP address type:

1. Stop the server.
2. Edit the endpoint details.
3. Change the IP address type to **Dual-stack**.
4. Start the server.

> **Note**
>
> For VPC-Internet endpoints, dual-stack mode is not currently supported.

Using an Application Load Balancer for dual-stack AS2 server connectivity

You can enable dual-stack (IPv4 and IPv6) connectivity to your AS2 server by using an Application Load Balancer that has a public-facing endpoint. This allows trading partners to connect to your AS2 server using either IPv4 or IPv6.

To set up a dual-stack Application Load Balancer for your AS2 server

1. Create a VPC with the following settings:
   - VPC only
   - Manual IPv4 CIDR input
   - Amazon-provided IPv6 CIDR block
2. Create at least two subnets in different Availability Zones:
   - Add IPv6 CIDRs to the subnets

- When creating subnets, allocate only a subset of the VPC's IPv4/IPv6 addresses to leave addresses available for additional subnets

3. Create an internet gateway for the VPC.
4. Edit the route table and add two routes:
   - One route with *Destination* `0.0.0.0/0`
   - One route with *Destination* `::/0`
   - Set both route targets to the internet gateway you created
5. Create an AS2-enabled server in the VPC that you created in step 1. Make sure to specify the `IpAddressType` as `DUALSTACK`.

   For details on how to create a Transfer Family server that uses the AS2 protocol, see Create an AS2 server.
6. Create a target group:
   - For *Specify group details*, configure:
     - Target type: IP addresses
     - Name: Enter a name
     - Protocol: HTTP
     - Port: 5080
     - VPC: Select the VPC you created
     - Protocol version: HTTP1
     - Health checks: Use defaults
   - For *Register targets*:
     - Enter your AS2 server's private IPv4 address
     - Choose *Include as pending below*
7. Create an Application Load Balancer:
   - Enter a name
   - For *Scheme*, choose *Internet-facing*
   - For *IP address type*, choose *Dualstack*
   - For *Network mapping*:
     - Select the VPC you created

    - Select the Availability Zones where you created subnets
- For *Security groups*, select a security group that allows inbound IPv4 and IPv6 traffic from any IP address on port 80
- For *Listeners and routing*:
    - Protocol: HTTP
    - Port: 80
    - Default action: Forward to the target group you created
- Choose *Create load balancer*

After you create the Application Load Balancer, trading partners can use its DNS name to send traffic to your AS2 server. This configuration enables your AS2 server to accept connections from both IPv4 and IPv6 clients through the dual-stack Application Load Balancer.

Edit server details

After you create an AWS Transfer Family server, you can edit the server configuration.

Topics

- Edit the file transfer protocols
- Edit the server endpoint
- Edit your logging configuration
- Edit the security policy
- Change the managed workflow for your server
- Change the display banners for your server
- Put your server online or offline

To edit a server's configuration

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**.
3. Choose the identifier in the **Server ID** column to see the **Server details** page, shown following.

   You can change the server's properties on this page by choosing **Edit**:

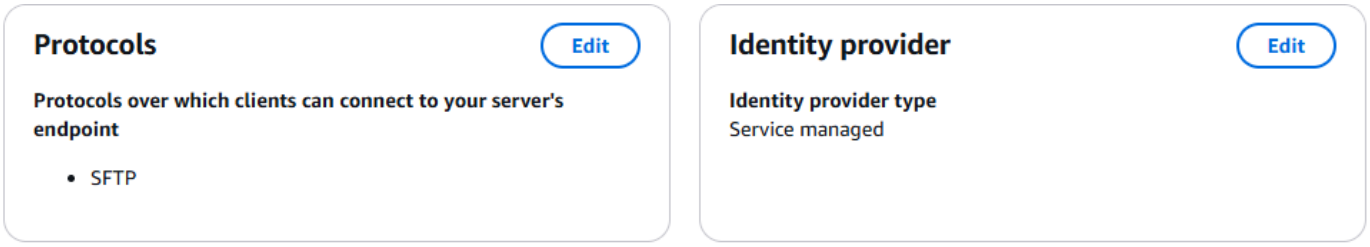
- To change the protocols, see Edit the file transfer protocols.
- For the identity provider, you can now change between any identity provider types (service-managed, AWS Directory Service, or custom identity provider). For details about changing identity provider types and the required information for each transition, see Edit identity provider configuration.
- To change the endpoint type or custom hostname, see Edit the server endpoint.
- To add an agreement, you need to first add AS2 as a protocol to your server. For details, see Edit the file transfer protocols.
- To manage host keys for your server, see Managing host keys for your SFTP-enabled server.
- Under **Additional details**, you can edit the following information:
  - To change the logging role, see Edit your logging configuration.
  - To change the security policy, see Edit the security policy.
  - To change the server host key, see Managing host keys for your SFTP-enabled server.
  - To change the managed workflow for your server, see Change the managed workflow for your server.
  - To edit the display banners for your server, see Change the display banners for your server.
- Under Additional configuration, you can edit the following information:
  - **SetStat option**: enable this option to ignore the error that is generated when a client attempts to use `SETSTAT` on a file you are uploading to an Amazon S3 bucket. For additional details, see the `SetStatOption` documentation in the ProtocolDetails topic.
  - **TLS session resumption**: provides a mechanism to resume or share a negotiated secret key between the control and data connection for an FTPS session. For additional details, see the `TlsSessionResumptionMode` documentation in the ProtocolDetails topic.
  - **Passive IP**: indicates passive mode, for FTP and FTPS protocols. Enter a single IPv4 address, such as the public IP address of a firewall, router, or load balancer. For additional details, see the `PassiveIp` documentation in the ProtocolDetails topic.

> **Note**
>
> Avoid placing Network Load Balancers (NLBs) or NAT gateways in front of AWS Transfer Family servers. This configuration increases costs and can cause performance issues. For more details, see Avoid placing NLBs and NATs in front of AWS Transfer Family servers.

- **PROXY protocol configuration**: this option is only available for SFTP servers that use a VPC-hosted endpoint. Turn on this option to preserve the client's source IP address if your server sits behind an NLB with PROXY protocol v2 (PPv2) enabled. For more information, see Working with Network Load Balancers and `ProxyConfig` in the ProtocolDetails API reference.
- **SFTP custom ports**: For an SFTP server that uses a `PUBLIC` endpoint type, you can choose the ports that the server listens on. Use the `SftpPorts` property to configure up to three ports. Each port can be any integer from 2000 to 65535, or 22. For each port, you can also set a `CommunicationMode` that controls which side sends data first. Use `CLIENT_TALK_FIRST` for clients that require TCP piggy-back ACKs. A piggy-back ACK carries data in the final ACK of the TCP three-way handshake. Use `SERVER_TALK_FIRST` if you are unsure of your clients' behavior or your clients do not depend on this. You set these options when you create or update a server. For more information, see the `SftpPorts` documentation in the ProtocolDetails topic.

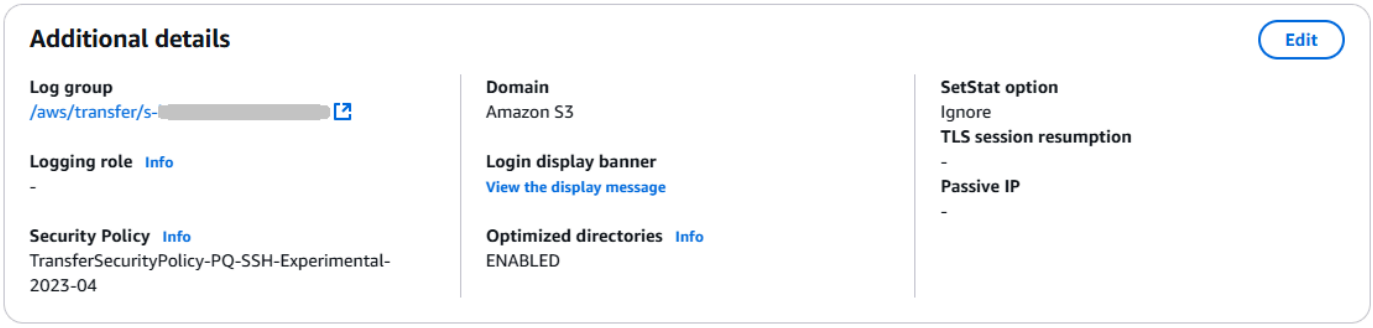
  > **Note**
  >
  > To change the SFTP custom ports, your server must be in the OFFLINE state. To stop your server, see Put your server online or offline.

- To start or stop your server, see Put your server online or offline.
- To delete a server, see Delete a server.
- To edit a user's properties, see Managing access controls.



> **Note**
>
> The server host key **Description** and **Date imported** values are new as of September 2022. These values were introduced to support the multiple host keys feature. This feature required migration of any single host keys that were in use before the introduction of multiple host keys.

The **Date imported** value for a migrated server host key is set to the last modified date for the server. That is, the date that you see for your migrated host key corresponds to the date that you last modified the server in any way, before the server host key migration.
The only key that was migrated is your oldest or only server host key. Any additional keys have their actual date from when you imported them. Additionally, the migrated key has a description that makes it easy to identify it as having been migrated.
The migration occurred between September 2 and September 13. The actual migration date within this range depends on the Region of your server.



Edit the file transfer protocols

On the AWS Transfer Family console, you can edit the file transfer protocol. The file transfer protocol connects the client to your server's endpoint.

To edit the protocols

1. On the **Server details** page, choose **Edit** next to **Protocols**.
2. On the **Edit protocols** page, select or clear the protocol check box or check boxes to add or remove the following file transfer protocols:

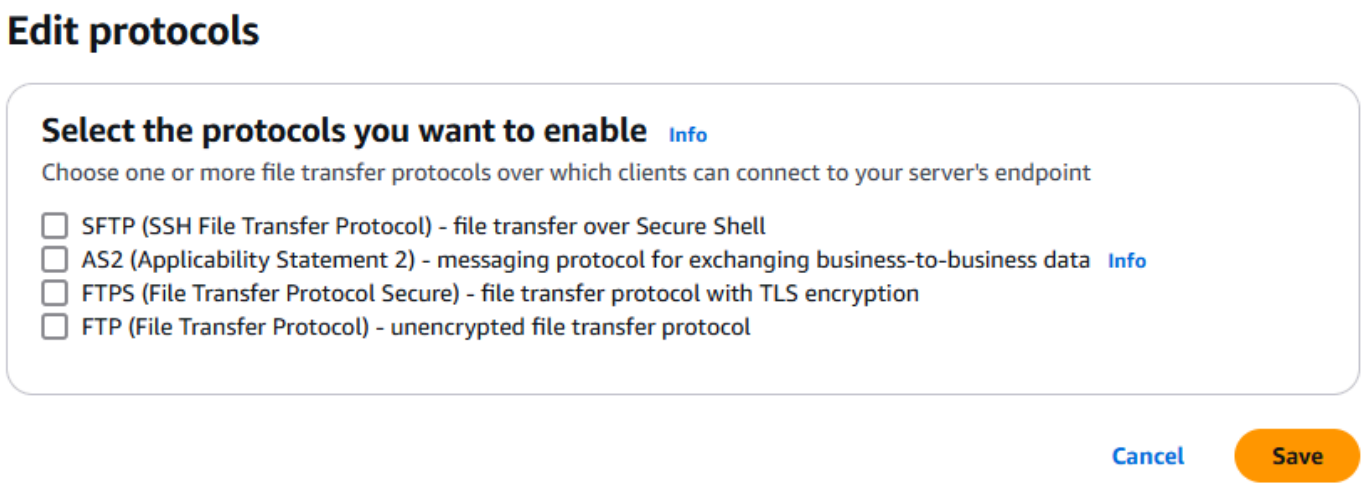
   - Secure Shell (SSH) File Transfer Protocol (SFTP) – file transfer over SSH

     For more information about SFTP, see Create an SFTP-enabled server.
   - File Transfer Protocol Secure (FTPS) – file transfer with TLS encryption

     For more information about FTP, see Create an FTPS-enabled server.
   - File Transfer Protocol (FTP) – unencrypted file transfer

For more information about FTPS, see Create an FTP-enabled server.

> **Note**
>
> If you have an existing server enabled only for SFTP, and you want to add FTPS and FTP, you must ensure that you have the right identity provider and endpoint type settings that are compatible with FTPS and FTP.



If you select **FTPS**, you must choose a certificate stored in AWS Certificate Manager (ACM) which will be used to identify your server when clients connect to it over FTPS.

To request a new public certificate, see Request a public certificate in the *AWS Certificate Manager User Guide*.

To import an existing certificate into ACM, see Importing certificates into ACM in the *AWS Certificate Manager User Guide*.

To request a private certificate to use FTPS through private IP addresses, see Requesting a private certificate in the *AWS Certificate Manager User Guide*.

Certificates with the following cryptographic algorithms and key sizes are supported:

- 2048-bit RSA (RSA_2048)
- 4096-bit RSA (RSA_4096)

- Elliptic Prime Curve 256 bit (EC_prime256v1)
- Elliptic Prime Curve 384 bit (EC_secp384r1)
- Elliptic Prime Curve 521 bit (EC_secp521r1)

> **Note**
>
> The certificate must be a valid SSL/TLS X.509 version 3 certificate with FQDN or IP address specified and contain information about the issuer.

3. Choose **Save**. You are returned to the **Server details** page.

Edit the server endpoint

On the AWS Transfer Family console, you can modify the server endpoint type and custom hostname. Additionally, for VPC endpoints, you can edit the availability zone information.

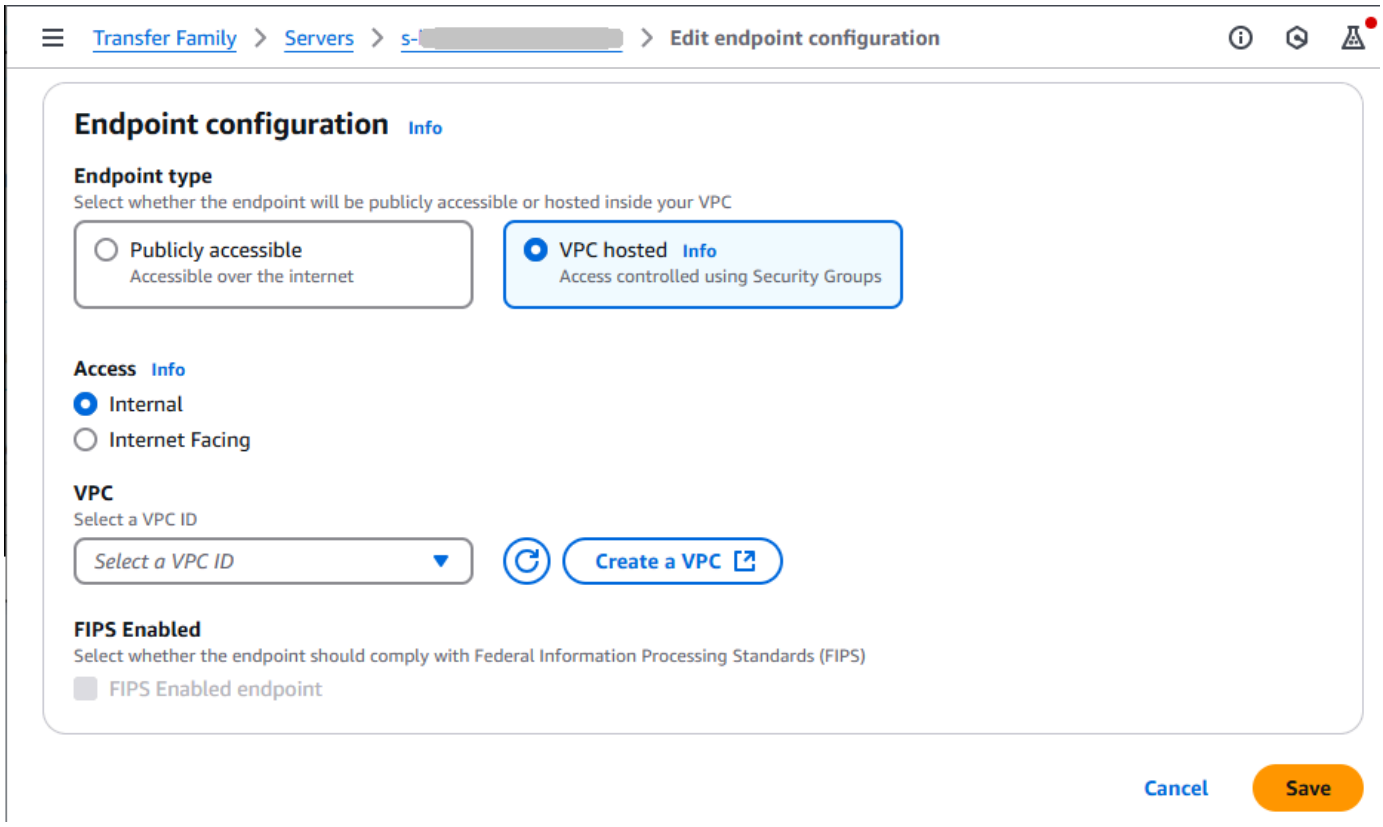
To edit the server endpoint details

1. On the **Server details** page, choose **Edit** next to **Endpoint details**.
2. Before you can edit the **Endpoint type**, you must first stop the server. Then, on the **Edit endpoint configuration** page, for **Endpoint type**, you can choose either of the following values:
   - **Public** – This option makes your server accessible over the internet.
   - **VPC** – This option makes your server accessible in your virtual private cloud (VPC). For information about VPC, see Create a server in a virtual private cloud.
3. For **Custom hostname**, choose one of the following:
   - **None** – If you don't want to use a custom domain, choose **None**.

     You get a server hostname provided by AWS Transfer Family. The server hostname takes the form `serverId.server.transfer.regionId.amazonaws.com`.
   - **Amazon Route 53 DNS alias** – To use a DNS alias automatically created for you in Route 53, choose this option.
   - **Other DNS** – To use a hostname that you already own in an external DNS service choose **Other DNS**.

Choosing **Amazon Route 53 DNS alias** or **Other DNS** specifies the name resolution method to associate with your server's endpoint.

For example, your custom domain might be `sftp.inbox.example.com`. A custom hostname uses a DNS name that you provide and that a DNS service can resolve. You can use Route 53 as your DNS resolver, or use your own DNS service provider. To learn how AWS Transfer Family uses Route 53 to route traffic from your custom domain to the server endpoint, see Working with custom hostnames.



4. For VPC endpoints, you can change the information in the **Availability Zones** pane.
5. Choose **Save**. You are returned to the **Server details** page.

Edit your logging configuration

On the AWS Transfer Family console, you can change your logging configuration.

> **Note**
>
> If Transfer Family created a CloudWatch logging IAM role for you when you created a server, the IAM role is called `AWSTransferLoggingAccess`. You can use it for all your Transfer Family servers.

To edit your logging configuration

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. Based on your configuration, choose between a logging role, structured JSON logging, or both. For more information, see Updating logging for a server.

Edit the security policy

This procedure explains how to change a Transfer Family server's security policy by using the AWS Transfer Family console or AWS CLI.

> **Note**
>
> If your endpoint is FIPS-enabled, you can't change the FIPS security policy to a non-FIPS security policy.

Console

To edit the security policy by using the console

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. In the **Cryptographic algorithm options** section, choose a security policy that contains the cryptographic algorithms enabled for use by your server.

   For more information about security policies, see Security policies for AWS Transfer Family servers.
3. Choose **Save**.

   You are returned to the **Server details** page where you can see the updated security policy.

AWS CLI

To edit the security policy by using the CLI

1. Run the following command to view the current security policy that is attached to your server.

   ```
   aws transfer describe-server --server-id your-server-id
   ```

   This `describe-server` command returns all of the details for your server, including the following line:

   ```
   "SecurityPolicyName": "TransferSecurityPolicy-2018-11"
   ```

   In this case, the security policy for the server is `TransferSecurityPolicy-2018-11`.

2. Make sure to provide the exact name of the security policy to the command.
   For example, run the following command to update the server to `TransferSecurityPolicy-2023-05`.

   ```
   aws transfer update-server --server-id your-server-id --security-policy-name
    "TransferSecurityPolicy-2023-05"
   ```

   > **Note**
   >
   > The names of the available security policies are listed in Security policies for AWS Transfer Family servers.

If successful, the command returns the following code, and updates your server's security policy.

```
{
    "ServerId": "your-server-id"
}
```

Change the managed workflow for your server

On the AWS Transfer Family console, you can change the managed workflow associated with the server.

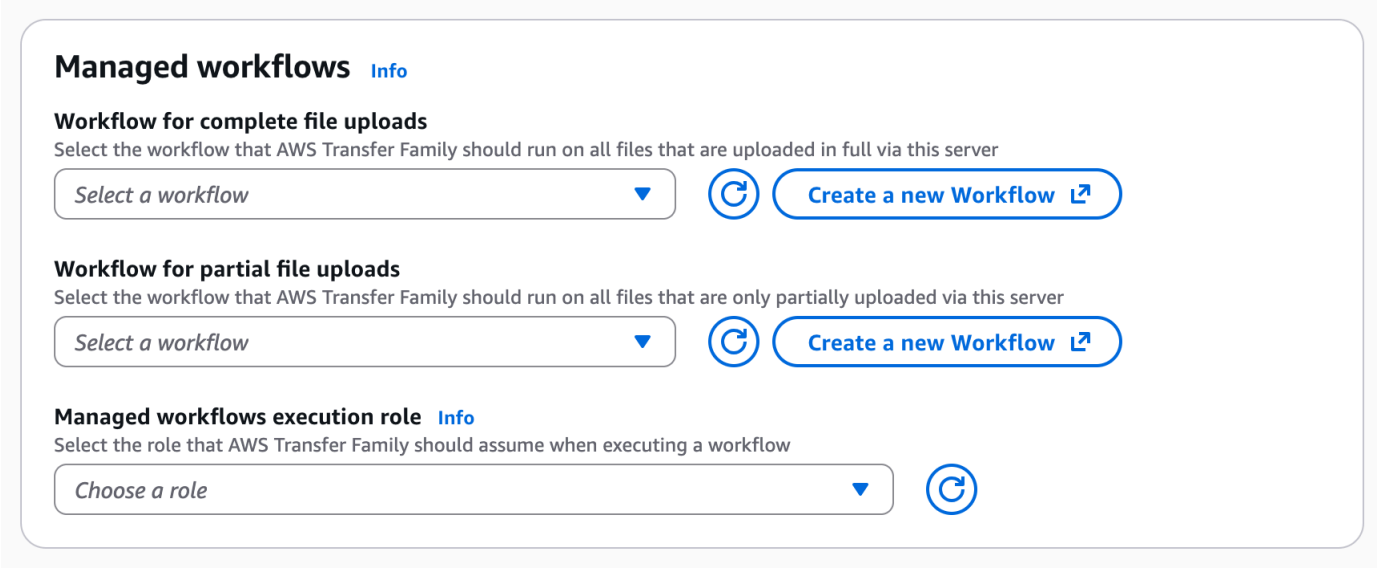
To change the managed workflow

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. On the **Edit additional details** page, in the **Managed workflows** section, select a workflow to be run on all uploads.

> **Note**
>
> If you do not already have a workflow, choose **Create a new workflow** to create one.

   a. Select the workflow ID to use.

   b. Choose an execution role. This is the role that Transfer Family assumes when executing the workflow's steps. For more information, see IAM policies for workflows. Choose **Save**.



3. Choose **Save**. You are returned to the **Server details** page.

Change the display banners for your server

On the AWS Transfer Family console, you can change the display banners associated with the server.

To change the display banners

1. On the **Server details** page, choose **Edit** next to **Additional details**.
2. On the **Edit additional details** page, in the **Display banners** section, enter text for the available display banners.
3. Choose **Save**. You are returned to the **Server details** page.

Put your server online or offline

On the AWS Transfer Family console, you can bring your server online or take it offline.

To bring your server online

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that is offline.
4. For **Actions**, choose **Start**.

It can take a couple of minutes for a server to switch from offline to online.

> **Note**
>
> When you stop a server to take it offline, currently you are still accruing service charges for that server. To eliminate additional server-based charges, delete that server.

To take your server offline

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the navigation pane, choose **Servers**.
3. Select the check box of the server that is online.
4. For **Actions**, choose **Stop**.

While a server is starting up or shutting down, servers aren't available for file operations. The console doesn't show the starting and stopping states.

If you find the error condition `START_FAILED` or `STOP_FAILED`, contact AWS Support to help resolve your issues.

Edit identity provider configuration

You can change your server's identity provider type from any type to any other type. The available identity provider types are:

- **Service managed** – Store user credentials within the service
- **AWS Directory Service** – Use AWS Managed Microsoft AD or AWS Directory Service for Entra ID Domain Services
- **Custom** – Use Lambda function or Amazon API Gateway to integrate with your existing identity provider

When changing identity provider types, you need to provide specific information depending on the transition you're making. The following sections describe the required information for each type of change.

Important

Considerations when changing identity providers:

- **User migration** – When changing identity provider types, existing user configurations are not automatically migrated. You'll need to set up users in the new identity provider system.
- **Testing** – Test the new identity provider configuration thoroughly before making the change in production environments.
- **Permissions** – Ensure that the new identity provider has the necessary IAM permissions and roles configured before making the change.

Changing to service-managed identity provider

When changing from any other identity provider type to service-managed, you need to:

- Select **Service managed** as the identity provider type
- Create new users directly in AWS Transfer Family after the change is complete, as existing user configurations from other identity providers won't be transferred

Example: If you're changing from a custom identity provider to service-managed, you'll need to recreate all user accounts and their associated permissions within the AWS Transfer Family service.

Changing to AWS Directory Service

When changing from any other identity provider type to AWS Directory Service, you need to provide:

- **Directory** – Select an existing AWS Managed Microsoft AD or AWS Directory Service for Entra ID Domain Services directory
- **Access** – Choose whether to restrict access to a specific group or allow access to all users in the directory
- **Access role** – An IAM role that allows AWS Transfer Family to access your directory

Example: If you're changing from service-managed to AWS Directory Service, you would select your existing `d-1234567890` directory, choose to restrict access to the `TransferUsers` group, and specify the `TransferDirectoryAccessRole` IAM role.

Changing to custom identity provider

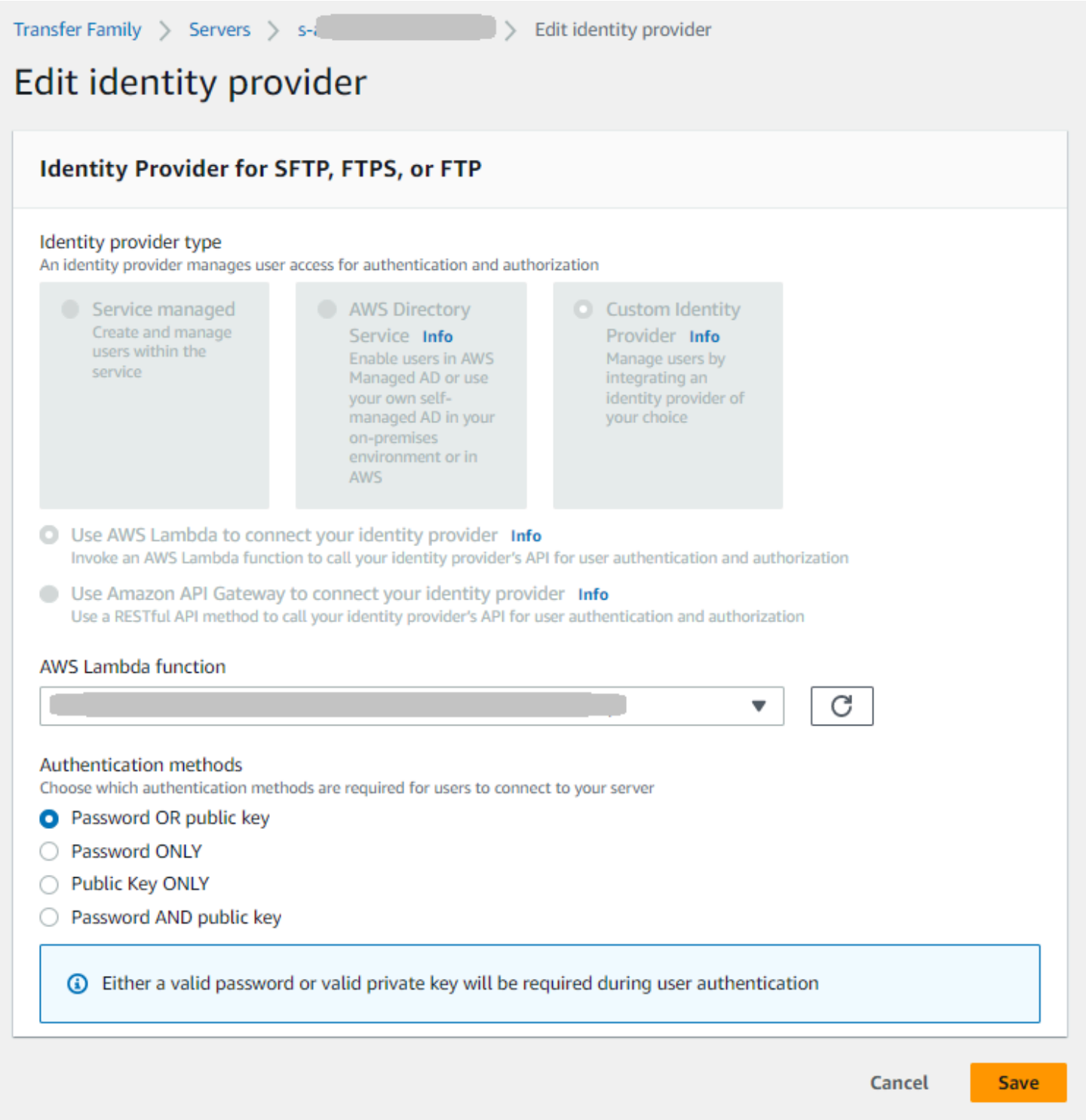
When changing from any other identity provider type to a custom identity provider, you need to choose between Lambda function or Amazon API Gateway and provide the required configuration:

Using Lambda function

For Lambda function integration, provide:

- **Function** – Select an existing Lambda function that handles authentication
- **Authentication method** (for SFTP protocol) – Choose password, public key, or both

Example: If you're changing from AWS Directory Service to a custom Lambda identity provider, you would select your `TransferCustomAuth` function and choose **Password** as the authentication method.



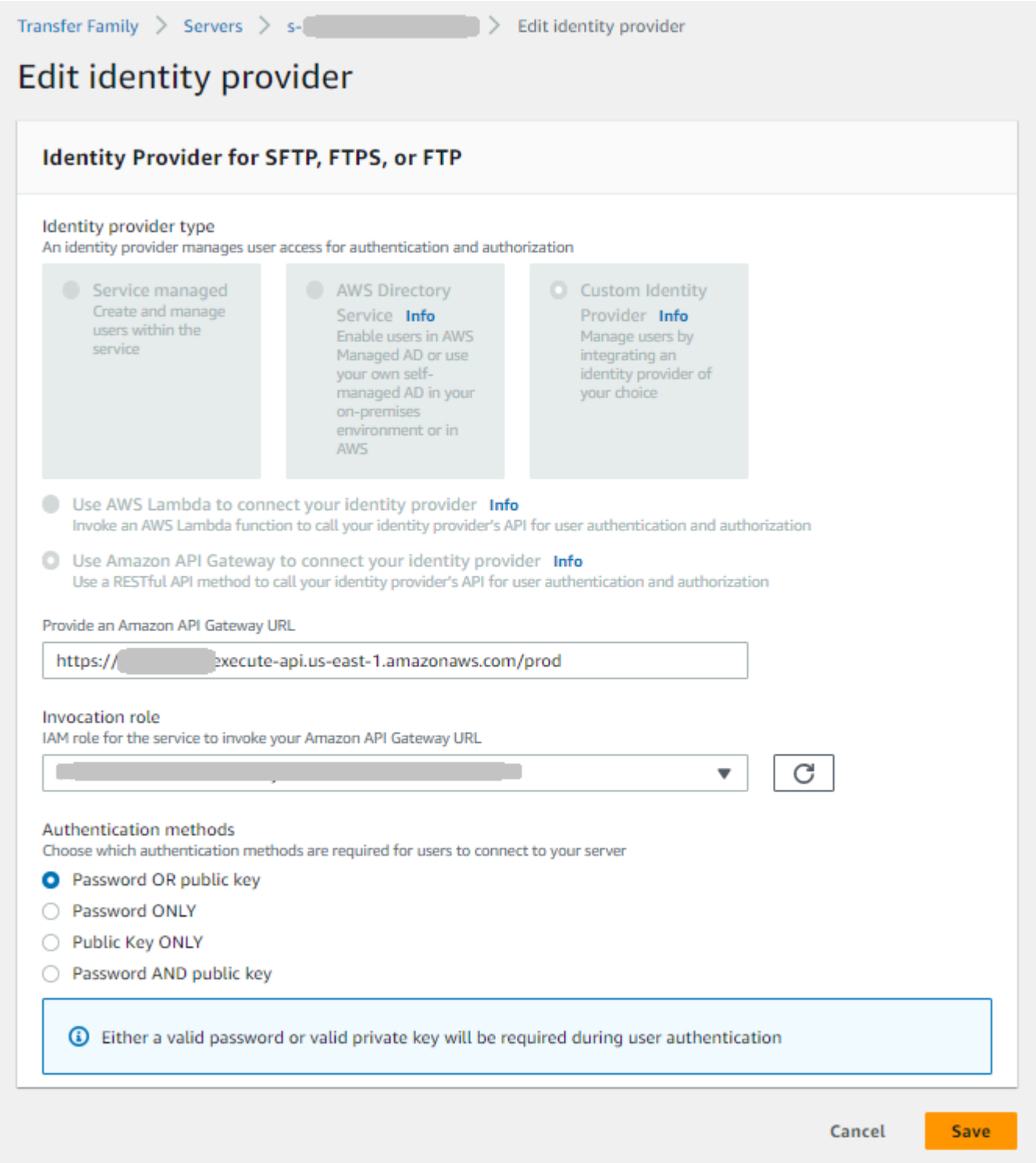
Using Amazon API Gateway

For Amazon API Gateway integration, provide:

- **API Gateway URL** – The invoke URL of your API Gateway endpoint

- **Invocation role** – An IAM role that allows AWS Transfer Family to invoke your API Gateway
- **Authentication method** (for SFTP protocol) – Choose password, public key, or both

Example: If you're changing from service-managed to API Gateway, you would provide the URL `https://abcdef123.execute-api.us-east-1.amazonaws.com/prod`, specify the `TransferApiGatewayInvocationRole` IAM role, and choose **Public key** as the authentication method.



Changing from Amazon API Gateway to Lambda function

A common transition is changing from Amazon API Gateway to Lambda function for custom identity provider integration. This change allows you to simplify your architecture while maintaining the same authentication logic.

Key considerations for this transition:

- **Same function, different permissions** – You can use the same Lambda function for both API Gateway and direct Lambda integration, but the resource policy must be updated.
- **Resource policy requirements** – When changing to direct Lambda integration, the function's resource policy must grant `transfer.amazonaws.com` permission to invoke the function, in addition to `apigateway.amazonaws.com`.

To make this change

1. Update your Lambda function's resource policy to allow `transfer.amazonaws.com` to invoke the function.
2. In the AWS Transfer Family console, change the identity provider from API Gateway to Lambda function.
3. Select your existing Lambda function.
4. Test the configuration to ensure authentication works correctly.

Example resource policy for direct Lambda integration:

```
{
   "Version":"2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Principal": {
         "Service": [
            "transfer.amazonaws.com",
            "apigateway.amazonaws.com"
         ]
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:function-name"
   }]
}
```

User preservation during identity provider transitions

When changing between identity provider types, existing user configurations are preserved in specific scenarios to enable efficient rollback in case of issues:

- **Service-managed to custom identity provider and back** – If you change from service-managed to a custom identity provider and then back to service-managed, all users are preserved in their last known configuration.
- **AWS Directory Service to custom identity provider and back** – If you change from AWS Directory Service to a custom identity provider and then back to AWS Directory Service, all definitions for Delegated Access groups are retained in their last known configuration.

This preservation behavior allows you to safely test custom identity provider configurations and roll back to your previous setup without losing user access configurations.

Important considerations when changing identity providers

- **User migration** – When changing identity provider types, existing user configurations are not automatically migrated. You'll need to set up users in the new identity provider system.
- **Testing** – Test the new identity provider configuration thoroughly before making the change in production environments.
- **Permissions** – Ensure that the new identity provider has the necessary IAM permissions and roles configured before making the change.

Managing host keys for your SFTP-enabled server

Server host keys are private keys used by the Transfer Family server to provide a unique identity to the caller, and to guarantee that it is the correct server. That guarantee is enforced by the presence of the correct public key in the caller's `known_hosts` file. (The `known_hosts` file is a standard feature used by most SSH clients to store the public keys for the servers that you've connected to.) You can retrieve the public key that corresponds to your server host key by running `ssh-keyscan` for your server.

> **Important**
>
> Accidentally changing a server's host key can be disruptive. Depending on how your SFTP client is configured, it can fail immediately, with the message that no trusted host key

exists, or present threatening prompts. If there are scripts for automating connections, they most likely would fail as well.

By default, AWS Transfer Family generates host keys for your SFTP-enabled server. You can import server host keys to preserve host identity and avoid updating client trust stores. When to import host keys lists a few reasons you might want to do this. If you do not provide host keys, new ones will be generated for you.

AWS Transfer Family supports multiple host keys of different types (RSA, ECDSA, and ED25519) to provide compatibility with a broader range of client host signature algorithms. Different key types enable specific algorithms: RSA keys enable **rsa-*** algorithms, ECDSA keys enable **ecdsa-*** algorithms, and ED25519 keys enable **ed25519** algorithms. Plan your key types at server creation time, as introducing additional key types after clients have started interacting with the server can be disruptive for some clients and may be as problematic as replacing existing host keys.

To prevent your users from being prompted to verify the authenticity of your SFTP-enabled server again, import the host key for your on-premises server to the SFTP-enabled server. Doing this also prevents your users from getting a warning about a potential man-in-the-middle attack.

You can also rotate host keys periodically, as an additional security measure. For details, see Rotating the server host keys.

Note

Server host keys are used by servers that support the SFTP protocol.

AWS Transfer Family supports two host key storage types, which control where Transfer Family keeps the private key material for a host key:

- Service managed (`SERVICE_MANAGED`) – Transfer Family stores and manages the host key material for you. This is the default storage type when you create a server by using the AWS CLI or the AWS API, and it requires no extra configuration.
- AWS Secrets Manager (`SECRETS_MANAGER`) – Transfer Family stores the host key in a secret in Secrets Manager that you can encrypt with your own AWS KMS key. This is the default storage type when you create a server in the AWS Transfer Family console.

For details about AWS Secrets Manager storage, see Storing SFTP server host keys in AWS Secrets Manager.

When to import host keys

While AWS Transfer Family can generate host keys automatically, there are several scenarios where importing your own host keys provides operational benefits:

- *Server migration* - You're migrating from an existing server to AWS Transfer Family and want to avoid updating client trust stores (`known_hosts` files) for existing clients.
- *Disaster recovery and failover* - You have multiple AWS Transfer Family servers (for example, one in US East (Ohio) and one in US West (Oregon)) that share the same public DNS name. Using the same host keys on both servers ensures seamless failover without client authentication failures.
- *Operational continuity* - You want the host key material available for use with other servers (AWS Transfer Family or otherwise) in the future to maintain consistent server identity across your infrastructure.
- *Algorithm control* - You want greater client compatibility by providing more Host Key Algorithms, or you want to control which algorithms clients can use by only offering keys compatible with specific algorithms.

The following topics provide detailed procedures for managing server host keys:

- Adding an additional server host key - Add additional host keys to your server
- Storing SFTP server host keys in AWS Secrets Manager - Store host keys in AWS Secrets Manager
- Deleting a server host key - Remove host keys from your server
- Rotating the server host keys - Rotate host keys for enhanced security
- Additional server host key information - View and manage host key details

Adding an additional server host key

On the AWS Transfer Family console, you can add additional server host keys. Adding additional host keys of differing formats can be useful for identifying a server when clients connect to it, as well as improving your security profile. For example, if your original key is an RSA key, you could add an additional ECDSA key.

> **Note**
>
> The SFTP client will connect using the oldest key in the configuration that matches the key's algorithm. The oldest key for each key type (RSA, ECDSA, or ED25519) is the active key for the server for that type.

Security note when a Transfer Family server has multiple types of host keys

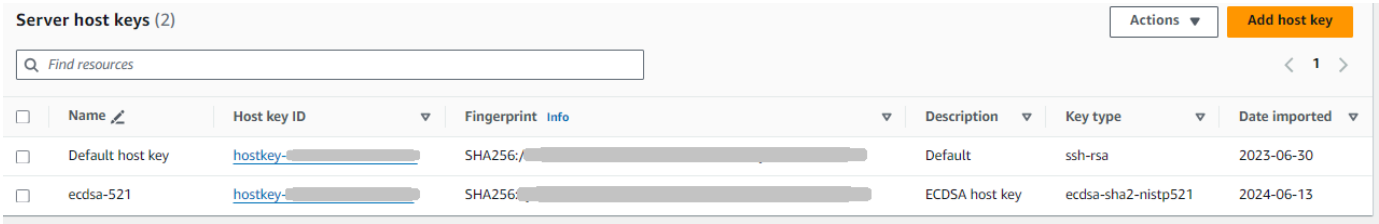
If a server has multiple types of host keys, the SFTP client can assign a preference by type. So, when there exist RSA, ECDSA, and ED25519 host keys for the server, the choice is driven by the preference by type.

Modern SFTP clients prefer ECDSA and ED25519 host keys when they exist. This becomes important if you want to add an ECDSA or ED25519 key when the server previously only had RSA keys. The addition of the new ECDSA or ED25519 key would potentially manifest as a security warning for a client.

To the client, the key will appear as having changed, when in fact it was not changed: the new key was added in addition to the existing RSA key. Keep this in mind if you decide to add new types of server host keys.

To add an additional server host key

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**, and then choose a server that uses the SFTP protocol.
3. On the server details page, scroll down to the **Server host keys** section.



4. Choose **Add host key**.

   The **Add server host key** page displays.
5. In the **Server Host Key** section, enter an RSA, ECDSA, or ED25519 private key that is used to identify your server when clients connect to it over the SFTP-enabled server.

> **Note**
>
> When you create a server host key, make sure to specify `-N ""` (no passphrase). See Creating SSH keys on macOS, Linux, or Unix for details on how to generate key pairs.

6. (Optional) Add a description to differentiate among multiple server host keys. You can also add tags for your key.
7. Choose **Add key**. You are returned to the **Server details** page.

To add a host key by using the AWS Command Line Interface (AWS CLI), use the ImportHostKey API operation and provide the new host key. If you create a new SFTP-enabled server, you provide your host key as a parameter in the CreateServer API operation. You can also use the AWS CLI to update the description for an existing host key.

The following example `import-host-key` AWS CLI command imports a host key for the specified SFTP-enabled server.

```
aws transfer import-host-key --description key-description --server-id your-server-id
 --host-key-body file://my-host-key
```

Storing SFTP server host keys in AWS Secrets Manager

Instead of letting Transfer Family manage your host key material, you can store the host keys for your SFTP-enabled server in a secret in AWS Secrets Manager (Secrets Manager). With Secrets Manager storage, Transfer Family stores the host key in a secret that you can encrypt with your own AWS Key Management Service (AWS KMS) key.

Choose Secrets Manager storage when you want to retrieve your host key material and reuse the same host key across servers and AWS Regions. You can also choose Secrets Manager storage when you want to encrypt your host key material with an AWS KMS key that you manage, or when you want to scope and audit access to the host key in Secrets Manager. To use Secrets Manager storage, set the storage type to `SECRETS_MANAGER`, as described in the following section.

Configuring the host key storage type

You set the storage type when you create a server or when you import or update a host key. You can use the AWS Management Console, the AWS CLI, or the AWS API. In the AWS Transfer Family console, new servers default to AWS Secrets Manager storage. In the AWS CLI and the AWS

API, new servers default to SERVICE_MANAGED storage if you don't set HostKeyStorageType. Transfer Family generates a host key for you if you don't provide one.

The parameter that you set depends on the API operation:

- When you create a server with the CreateServer API operation, set the HostKeyStorageType parameter to SERVICE_MANAGED or SECRETS_MANAGER.
- When you import or update a host key with the ImportHostKey or UpdateHostKey API operation, set the StorageType parameter to SERVICE_MANAGED or SECRETS_MANAGER.

To encrypt the Secrets Manager secret with a customer managed key, set the optional KmsKey parameter to the Amazon Resource Name (ARN) of an AWS KMS key when you import or update the host key. If you don't specify a key, Transfer Family uses the AWS managed key for Secrets Manager. To change the key for a host key that already uses Secrets Manager storage, call UpdateHostKey with the new KmsKey, and Transfer Family re-encrypts the secret under the new key. To use a customer managed key, you must also update its key policy. For more information, see Granting access to a customer managed key.

Note

When you call UpdateHostKey with StorageType set to SECRETS_MANAGER, your IAM identity must have the `transfer:ExportHostKey` permission, even if the host key already uses Secrets Manager storage. `transfer:ExportHostKey` is a permission-only action that doesn't map to an API operation.

You can change the storage type of an existing host key from SERVICE_MANAGED to SECRETS_MANAGER only if you imported the host key on or after October 14, 2026. To move an older host key to Secrets Manager storage, contact AWS Support to request an exception. Include your AWS account ID in your request.

The first time that you use Secrets Manager storage, Transfer Family creates the AWSServiceRoleForTransfer service-linked role in your account. Your IAM identity must have the `iam:CreateServiceLinkedRole` permission for this call. For more information, see Using service-linked roles for AWS Transfer Family.

The DescribeHostKey and ListHostKeys API operations report the StorageType of each host key. For a host key that uses Secrets Manager storage, the response also includes the SecretId of the Secrets Manager secret that holds the key material.

How Transfer Family manages the Secrets Manager secret

When you use Secrets Manager storage, Transfer Family creates a secret in your account with a name that begins with the `transfer!` prefix. Transfer Family uses the AWSServiceRoleForTransfer service-linked role to read the secret and serve the host key to incoming connections. For more information about this service-linked role, see Using service-linked roles for AWS Transfer Family.

> **Note**
>
> The secret is a managed secret. You can see it in the AWS Secrets Manager console, but Transfer Family controls its lifecycle, so we recommend that you don't modify or delete it manually. The AWSServiceRoleForTransfer service-linked role includes permissions that allow Transfer Family to eventually clean up orphaned secrets that are no longer associated with a host key. For more information, see AWS Secrets Manager secrets managed by other AWS services.

Working with the Secrets Manager secret

The following list describes what you can and can't do with the Secrets Manager secret that Transfer Family creates for your host key:

- **Finding the secret** – Transfer Family names the secret with a `transfer!` prefix. To help you scope access or filter secrets in the Secrets Manager console, Transfer Family applies the tags `aws:transfer:owningServerArn` and `aws:transfer:owningHostKeyArn`, in addition to the owning-service tag `aws:secretsmanager:owningService=transfer`.
- **Changing or rotating the host key** – You can't change the host key value through the Transfer Family API or in the Secrets Manager console, and Transfer Family doesn't use Secrets Manager automatic rotation for these secrets. To rotate a host key, import a new host key and delete the old one. For more information, see Rotating the server host keys.
- **Adding tags** – You can add your own tags to the secret.
- **Changing the encryption key** – Call `UpdateHostKey` with the `KmsKey` parameter. Transfer Family re-encrypts the secret under the new key. For more information, see Configuring the host key storage type.
- **Deleting the secret** – To remove a host key, use the `DeleteHostKey` API operation. When you delete a host key or its server, Transfer Family schedules the secret for deletion. For more information, see Deleting a server host key.

Important

We strongly recommend that you don't delete this secret manually. Deleting the secret prevents Transfer Family from reading the host key and disrupts your SFTP-enabled server. Likewise, if you remove the service-linked role's access to the secret or to its AWS KMS key, new connections to your server fail. Operations on host keys are eventually consistent.

Granting access to a customer managed key

If you encrypt the Secrets Manager secret with a customer managed key, the key policy must allow the Transfer Family service-linked role to call `kms:Decrypt`. The service-linked role's managed policy doesn't grant `kms:Decrypt`, so without this key policy statement Transfer Family can't read the host key material to serve it to incoming connections.

The following example key policy statement grants this access. The conditions limit the grant to calls that Secrets Manager makes on behalf of Transfer Family, for secrets whose names begin with `transfer!`. Replace *111122223333* with your AWS account ID and *us-east-1* with your AWS Region. If your Region is in a partition other than `aws`, also replace `arn:aws` with the ARN prefix for your partition.

Example – Key policy statement for the service-linked role

```
{
  "Sid": "Enable Transfer SLR to access",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/aws-service-role/transfer.amazonaws.com/
AWSServiceRoleForTransfer"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "secretsmanager.us-east-1.amazonaws.com"
    },
    "StringLike": {
      "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:transfer!*"
    }
```

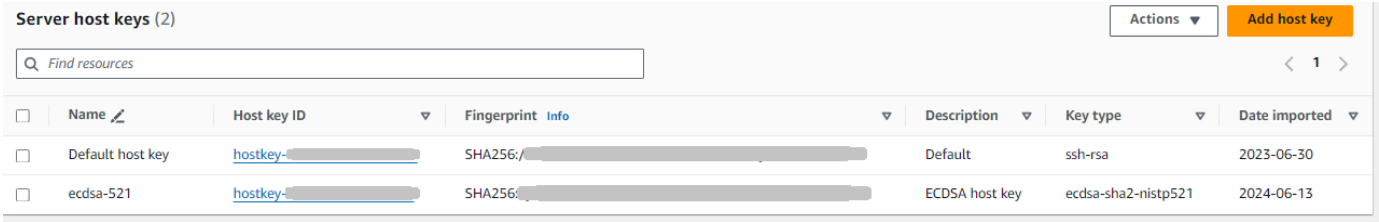
```
  }
}
```

Deleting a server host key

On the AWS Transfer Family console, you can delete a server host key.

To delete a server host key

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. In the left navigation pane, choose **Servers**, and then choose a server that uses the SFTP protocol.
3. On the server details page, scroll down to the **Server host keys** section.



4. In the **Server Host Keys** section, select a key, and then under **Actions**, choose **Delete**.
5. In the confirmation dialog box that appears, enter the word **`delete`**, and then choose **Delete** to confirm that you want to delete the host key.

The host key is deleted from the **Servers** page.

To delete the host key by using the AWS CLI, use the DeleteHostKey API operation and provide the server ID and host key ID.

The following example `delete-host-key` AWS CLI command deletes a host key for the specified SFTP-enabled server.

```
aws transfer delete-host-key --server-id your-server-id --host-key-id your-host-key-id
```

Rotating the server host keys

Periodically, you can rotate your server host key. This topic describes how the server chooses which key to apply, and the procedure for rotating these keys.

How the client chooses a server host key

The way that Transfer Family chooses which server key to apply depends on conditions for the SFTP client, as explained here. The assumption is that there is one older key and one newer key.

- An SFTP client has no prior public host key for the server. The first time the client connects to the server, either of the following occurs:
  - The client fails the connection, if it is configured to do so.
  - Or, the client chooses the first key that matches the possible available algorithms and asks the user if that key can be trusted. If so, the client auto-updates the `known_hosts` file (or whatever local configuration file or resource the client uses to record trust decisions) and enters that key.
- An SFTP client has an older key in its `known_hosts` file. The client prefers to use this key, even if a newer key exists, either for this key's algorithm or another algorithm. This is because the client has a higher level of trust for the key that is in its `known_hosts` file.
- An SFTP client has the new key (in any of the available algorithms) in its `known_hosts` keys file. The client ignores older keys because they are not trusted and uses the new key.
- An SFTP client has both keys in its `known_hosts` file. The client chooses the first key by index that matches the list of available keys offered by the server.

Transfer Family prefers that the SFTP client has all of the keys in its `known_hosts` file, since this allows the most flexibility when connecting to a Transfer Family server. Key rotation is based on the fact that multiple entries can exist in the `known_hosts` file for the same Transfer Family server.

Rotating the server host key procedure

As an example, assume that you have added the following set of server host keys to your Transfer Family server.

Server host keys

Host key type	Date added to the server
RSA	April 1, 2020
ECDSA	February 1, 2020
ED25519	December 1, 2019

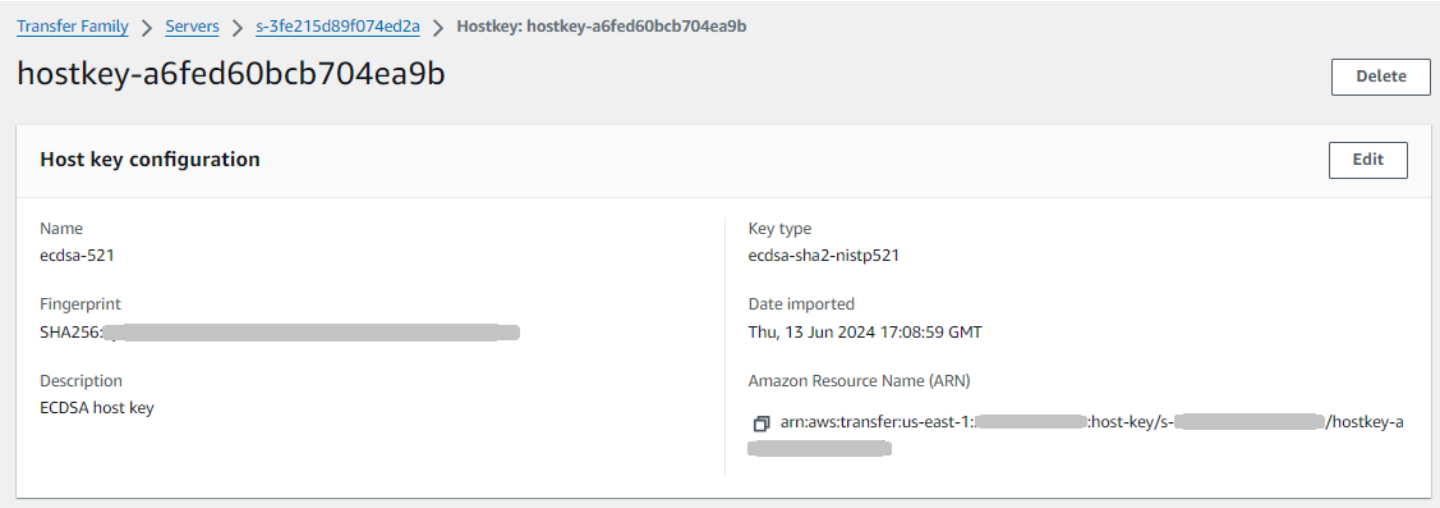
Host key type	Date added to the server
RSA	October 1, 2019
ECDSA	June 1, 2019
ED25519	March 1, 2019

To rotate the server host key

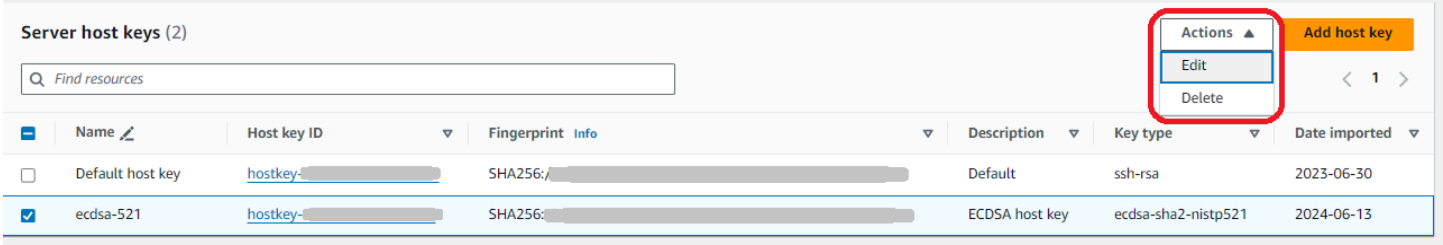
1. Add a new server host key. This procedure is described in Adding an additional server host key.
2. Delete one or more of the host keys of the same type that you had added previously. This procedure is described in Deleting a server host key.
3. All keys are visible, and can be active, subject to the behavior described previously in How the client chooses a server host key.

Additional server host key information

You can select a host key to display details for that key.



You can delete a host key, or edit its description from the **Actions** menu on the Server details screen. Select the host key, then choose the appropriate action from the menu.



Monitoring usage in the console

You can get information about your server's metrics on its **Server details** page. This provides you with a single place to monitor your file-transfers workloads. You can track how many files you have exchanged with your partners and closely track their usage using a centralized dashboard. For details, see View SFTP, FTPS, and FTP server details. The following table describes the metrics available for Transfer Family.

Namespace	Metric	Description
`AWS/Transfer`	`BytesIn`	The total number of bytes transferred into the server. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing **Units**: Count **Period**: 5 minutes
	`BytesOut`	The total number of bytes transferred out of the server. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family

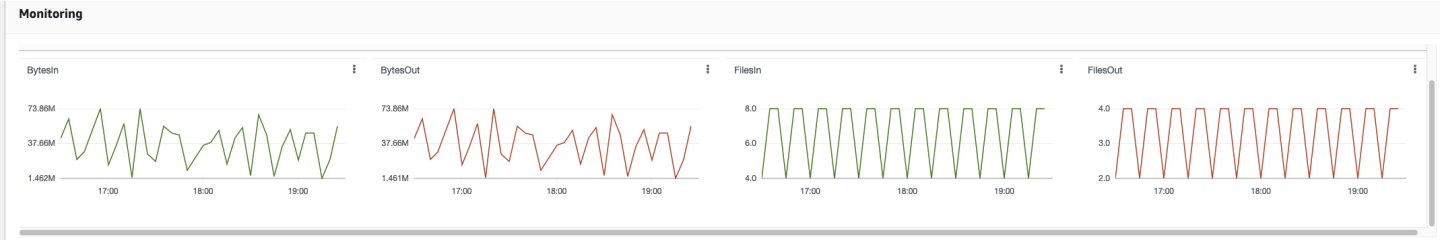
Namespace	Metric	Description
		server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer calls `StartFileTransfer` from their AS2 connector and emitted as soon as the outbound message has finished processing. **Units**: Count **Period**: 5 minutes
	`FilesIn`	The total number of files transferred into the server. For servers using the AS2 protocol, this metric represents the number of messages received. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing. **Units**: Count **Period**: 5 minutes

Namespace	Metric	Description
	`FilesOut`	The total number of files transferred out of the server. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer calls `StartFile Transfer` from their AS2 connector and emitted as soon as the outbound message has finished processin g. **Units**: Count **Period**: 5 minutes
	`InboundMe ssage`	The total number of AS2 messages successfully received from a trading partner. **Reporting criteria**: When the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing successfully **Units**: Count **Period**: 5 minutes

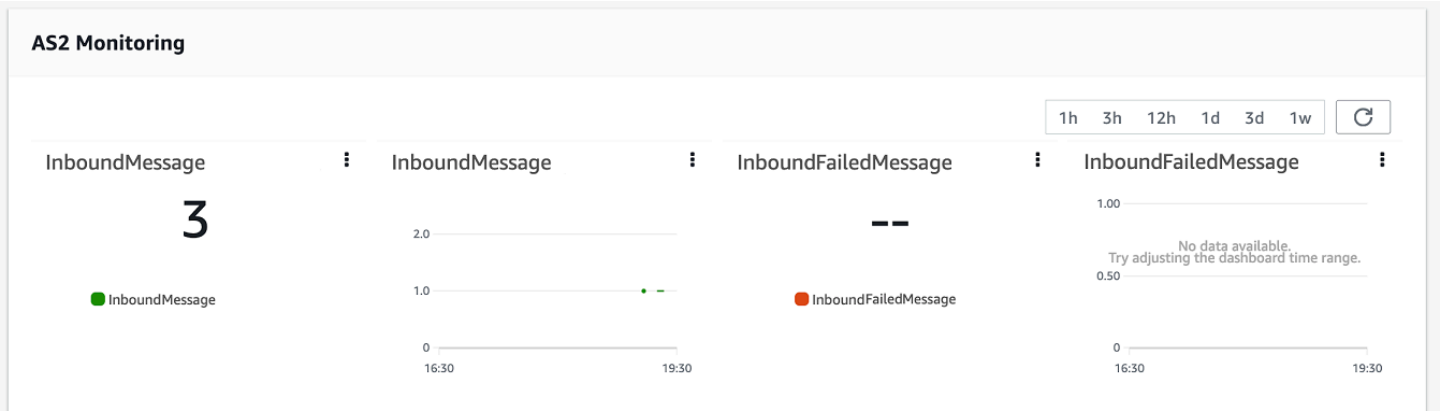
Namespace	Metric	Description
	`InboundFailedMessage`	The total number of AS2 messages that were unsuccessfully received from a trading partner. That is, a trading partner sent a message, but the Transfer Family server was not able to successfully process it. **Reporting criteria**: When the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing unsuccessfully **Units**: Count **Period**: 5 minutes
	`OnUploadExecutionsStarted`	The total number of workflow executions started on the server. **Reporting criteria**: Triggered every time an execution starts **Units**: Count **Period**: 1 minute
	`OnUploadExecutionsSuccess`	The total number of successful workflow executions on the server. **Reporting criteria**: Triggered every time an execution successfully finishes **Units**: Count **Period**: 1 minute

Namespace	Metric	Description
	`OnUploadExecutionsFailed`	The total number of unsuccessful workflow executions on the server. **Reporting criteria**: Triggered every time an execution does not successfully finish **Units**: Count **Period**: 1 minute
	`OutboundMessage`	The total number of AS2 messages successfully sent to a a trading partner. **Reporting criteria**: When the customer calls `StartFileTransfer` from their AS2 connector and emitted as soon as the outbound message has finished processing successfully **Units**: Count **Period**: 5 minutes
	`OutboundFailedMessage`	The total number of AS2 messages that were unsuccessfully sent to a trading partner. **Reporting criteria**: When the customer calls `StartFileTransfer` from their AS2 connector and emitted as soon as the outbound message has finished processing unsuccessfully **Units**: Count **Period**: 5 minutes

The **Monitoring** section contains four, individual graphs. These graphs show the bytes in, bytes out, files in, and files out.



For servers that have the AS2 protocol enabled, there is an **AS2 Monitoring** section below the **Monitoring** information. This section contains details for the number of inbound messages, both successful and failed.



To open the selected graph in its own window, choose the expand icon ().

You can also click a graph's vertical ellipsis icon () to open a dropdown menu with the following items:

- **Enlarge** – Opens the selected graph in its own window.
- **Refresh** – Reloads the graph with the most recent data.
- **View in metrics** – Opens the corresponding metrics details in Amazon CloudWatch.
- **View logs** – Opens the corresponding log group in CloudWatch.

Managing access controls

You can control a user's access to AWS Transfer Family resources by using an AWS Identity and Access Management (IAM) policy. An IAM policy is a statement, typically in JSON format, that allows a certain level of access to a resource. You use an IAM policy to define what file operations that you want to allow your users to perform and not perform. You can also use an IAM policy to define what Amazon S3 bucket or buckets that you want to give your users access to. To specify these policies for users, you create an IAM role for AWS Transfer Family that has the IAM policy and trust relationship associated with it.

Each user is assigned an IAM role. The type of IAM role that AWS Transfer Family uses is called a *service role*. When a user logs in to your server, AWS Transfer Family assumes the IAM role mapped to the user. To learn about creating an IAM role that provides a user access to an Amazon S3 bucket, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

You can grant write-only access to Amazon S3 objects by using certain permissions within an IAM policy. For details, see Grant ability to only write and list files.

The AWS Storage Blog contains a post detailing how to set up least privilege access. For details, see Implementing least privilege access in an AWS Transfer Family workflow.

> **Note**
>
> If your Amazon S3 bucket is encrypted using AWS Key Management Service (AWS KMS), you must specify additional permissions in your policy. For details, see Data protection and encryption. Additionally, you can see more information about session policies in the *IAM User Guide*.

Topics

- Allowing read and write access to an Amazon S3 bucket
- Creating a session policy for an Amazon S3 bucket
- Dynamic permission management approaches

Allowing read and write access to an Amazon S3 bucket

This section describes how to create an IAM policy that allows read and write access to a specific Amazon S3 bucket. Assigning an IAM role that has this IAM policy to your user gives that user read/write access to the specified Amazon S3 bucket.

The following policy provides programmatic read, write, and tagging access to an Amazon S3 bucket. The `GetObjectACL` and `PutObjectACL` statements are only required if you need to enable Cross Account Access. That is, your Transfer Family server needs to access a bucket in a different account.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"ReadWriteS3",
      "Action": [
            "s3:ListBucket"
                ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectTagging",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionTagging",
        "s3:GetObjectACL",
        "s3:PutObjectACL"
      ],
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/*"]
    }
  ]
}
```

The `ListBucket` action requires permission to the bucket itself. The `PUT`, `GET`, and `DELETE` actions require object permissions. Because these are different resources, they are specified using different Amazon Resource Names (ARNs).

To further restrict your users' access to only the `home` prefix of the specified Amazon S3 bucket, see Creating a session policy for an Amazon S3 bucket.

Creating a session policy for an Amazon S3 bucket

A *session policy* is an AWS Identity and Access Management (IAM) policy that restricts users to certain portions of an Amazon S3 bucket. It does so by evaluating access in real time.

> **Note**
>
> Session policies are only used with Amazon S3. For Amazon EFS, you use POSIX file permissions to limit access.

You can use a session policy when you need to give the same access to a group of users to a particular portion of your Amazon S3 bucket. For example, a group of users might need access to only the `home` directory. That group of users share the same IAM role.

> **Note**
>
> The maximum length of a session policy is 2048 characters. For more details, see the Policy request parameter for the `CreateUser` action in the *API reference*.

To create a session policy, use the following policy variables in your IAM policy:

- `${transfer:HomeBucket}`
- `${transfer:HomeDirectory}`
- `${transfer:HomeFolder}`
- `${transfer:UserName}`

Important

You can't use the preceding variables in Managed Policies. Nor can you use them as policy variables in an IAM role definition. You create these variables in an IAM policy and supply them directly when setting up your user. Also, you can't use the `${aws:Username}` variable in this session policy. This variable refers to an IAM user name and not the username required by AWS Transfer Family.

Example session policy

The following code shows an example session policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
          "Sid": "AllowListingOfUserFolder",
          "Action": [
              "s3:ListBucket"
          ],
          "Effect": "Allow",
          "Resource": [
              "arn:aws:s3:::${transfer:HomeBucket}"
          ],
          "Condition": {
              "StringLike": {
                  "s3:prefix": [
                      "${transfer:HomeFolder}/*",
                      "${transfer:HomeFolder}"
                  ]
              }
          }
      },
      {
          "Sid": "HomeDirObjectAccess",
          "Effect": "Allow",
          "Action": [
              "s3:PutObject",
              "s3:GetObject",
              "s3:DeleteObjectVersion",
```

```
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:GetObjectACL",
                "s3:PutObjectACL"
            ],
            "Resource": "arn:aws:s3:::${transfer:HomeDirectory}/*"
        }
    ]
}
```

Note

The preceding policy example assumes that users have their home directories set to include a trailing slash, to signify that it is a directory. If, on the other hand, you set a user's `HomeDirectory` without the trailing slash, then you should include it as part of your policy.

In the previous example policy, note the use of the `transfer:HomeFolder`, `transfer:HomeBucket`, and `transfer:HomeDirectory` policy parameters. These parameters are set for the `HomeDirectory` that is configured for the user, as described in HomeDirectory and Implementing your API Gateway method. These parameters have the following definitions:

- The `transfer:HomeBucket` parameter is replaced with the first component of `HomeDirectory`.
- The `transfer:HomeFolder` parameter is replaced with the remaining portions of the `HomeDirectory` parameter.
- The `transfer:HomeDirectory` parameter has the leading forward slash (/) removed so that it can be used as part of an S3 Amazon Resource Name (ARN) in a `Resource` statement.

Note

If you are using logical directories—that is, the user's `homeDirectoryType` is `LOGICAL`—these policy parameters (`HomeBucket`, `HomeDirectory`, and `HomeFolder`) are not supported.

For example, assume that the `HomeDirectory` parameter that is configured for the Transfer Family user is `/home/bob/amazon/stuff/`.

- `transfer:HomeBucket` is set to `/home`.
- `transfer:HomeFolder` is set to `/bob/amazon/stuff/`.
- `transfer:HomeDirectory` becomes `home/bob/amazon/stuff/`.

The first `"Sid"` allows the user to list all directories starting from `/home/bob/amazon/stuff/`.

The second `"Sid"` limits the user'`put` and `get` access to that same path, `/home/bob/amazon/stuff/`.

With the preceding policy in place, when a user logs in, they can access only objects in their home directory. At connection time, AWS Transfer Family replaces these variables with the appropriate values for the user. Doing this makes it easier to apply the same policy documents to multiple users. This approach reduces the overhead of IAM role and policy management for managing your users' access to your Amazon S3 bucket.

You can also use a session policy to customize access for each of your users based on your business requirements. For more information, see Permissions for AssumeRole, AssumeRoleWithSAML, and AssumeRoleWithWebIdentity in the *IAM User Guide*.

Note

AWS Transfer Family stores the policy JSON, instead of the Amazon Resource Name (ARN) of the policy. So, when you change the policy in the IAM console, you need to return to AWS Transfer Family console and update your users with the latest policy contents. You can update the user on the **Policy Info** tab in the **User configuration** section.
If you are using the AWS CLI, you can use the following command to update the policy.

```
aws transfer update-user --server-id server --user-name user --policy \
   "$(aws iam get-policy-version --policy-arn policy --version-id version --
output json)"
```

Nested substitutions for session policies

Nested substitutions are not performed in Transfer Family session policies. Session policies can use nested variables, such as `${transfer:HomeDirectory}`. When the policy is processed, the outer variable (e.g., `${transfer:HomeDirectory}`) might get replaced with a value that contains another variable (e.g., `{amzn-s3-demo-bucket:/$(transfer:UserName}`). However, the nested variable is not further substituted with the actual username (e.g., **johndoe**).

This means that when creating session policies for Transfer Family, you need to account for this behavior and ensure that the policy structure and variable usage are designed accordingly. The nested variables may not be resolved as expected, and the policy might not grant the intended permissions. It's important to thoroughly test and validate the session policies to ensure they work as expected. This behavior is a key consideration when implementing access control and permissions for your Transfer Family environment.

One solution to this issue is to use the actual Amazon S3 bucket name in your session policy. So, for example, rather than specifying `${transfer:HomeDirectory}` in your session policy, use the following, where amzn-s3-demo-bucket is your actual bucket: `${amzn-s3-demo-bucket/transfer:UserName}`.

Dynamic permission management approaches

Understanding Transfer Family permission architecture

AWS Transfer Family supports dynamic permission management through session policies, which allow you to restrict the effective permissions of IAM roles at runtime. This approach works for both service-managed users and custom identity provider users, but is only supported when transferring files to or from Amazon S3 (not Amazon EFS).

Every AWS Transfer Family user operates with a permission model that consists of:

1. *Base IAM role* - Defines the foundational permissions for the user
2. *Optional session policy* - Restricts (scopes down) the base permissions at runtime

The effective permissions are the intersection of the base role permissions and the session policy permissions. Session policies can only restrict permissions; they cannot grant additional permissions beyond what the base role allows.

This architecture applies to both user types:

- *Service-managed users* - Session policies can be configured directly in the user settings
- *Custom identity provider users* - Session policies can be returned as part of the authentication response or stored in AWS Secrets Manager

Two approaches to permission management

When designing permissions for Transfer Family users who need unique access patterns, you can choose between two main approaches:

One role per user

Create a separate IAM role for each Transfer Family user with specific permissions tailored to that user's needs. Use this approach when:

- Each user requires very different permissions
- Permission administration is handled by different people in your organization
- You need fine-grained control over individual user access

Shared role with session policies

Use a single IAM role with broad permissions (such as access to an entire Amazon S3 bucket containing multiple user home directories) and apply session policies to restrict each user to their specific area. This approach significantly reduces administrative overhead compared to managing separate roles for each user. Use this approach when:

- Users need similar types of access but to different resources (for example, all users need read/write access, but each only to their own folder)
- You want to simplify role management and avoid creating dozens or hundreds of individual roles
- Users should only access their designated home directories within a shared bucket
- Permission administration is centralized within your organization

For example, instead of creating separate roles for users "alice", "bob", and "charlie", you can create one role with access to the entire `s3://company-transfers/` bucket, then use session policies to restrict alice to `s3://company-transfers/alice/`, bob to `s3://company-transfers/bob/`, and so on.

Implementing session policies

Session policies work by restricting the effective permissions of the base IAM role assigned to a user. The final permissions are the intersection of the role's permissions and the session policy's permissions.

You can implement dynamic session policies in two ways:

Variable substitution

Use Transfer Family policy variables such as `${transfer:Username}`, `${transfer:HomeDirectory}`, and `${transfer:HomeBucket}` in your session policies. These variables are automatically replaced with the actual values at runtime. For more information about these variables, see Creating a session policy for an Amazon S3 bucket.

Dynamic generation

For custom identity providers, generate session policies on-the-fly as part of the authentication response from your Lambda function or API Gateway method. This approach allows you to create highly customized policies based on user attributes, group memberships, or external data sources at authentication time.

You can also store pre-generated session policies in AWS Secrets Manager by including a key named `Policy` with the session policy JSON as the value. This allows you to use the same broad IAM role across multiple users while maintaining user-specific access controls.

Note

Session policies are only supported for file transfers to and from Amazon S3. They do not apply to Amazon EFS file systems. For Amazon EFS, permissions are governed by UID/GID and permission bits applied within the filesystem itself.

Implementation by user type

Service-managed users

For service-managed users, you can specify session policies directly in the user configuration through the AWS Transfer Family console, API, or CLI. For more information, see Working with service-managed users.

Custom identity provider users

For custom identity provider users, you can provide session policies in two ways:

- Through AWS Secrets Manager by including a key named `Policy` with the session policy as the value
- Directly in the Lambda function response or API Gateway response as part of the authentication result

For more information, see Custom identity provider solution.

Example: Simplifying role management with session policies

This example demonstrates how dynamic permission management can significantly reduce administrative overhead while maintaining security.

Scenario

Your organization has 50 users who need SFTP access to transfer files. Each user should only access their own folder within a shared Amazon S3 bucket called `company-transfers`. Without session policies, you would need to create 50 separate IAM roles.

Traditional approach (without session policies)

- Create 50 IAM roles: `TransferRole-Alice`, `TransferRole-Bob`, `TransferRole-Charlie`, etc.
- Each role has specific permissions to only that user's folder
- Managing permissions requires updating individual roles
- Adding new users requires creating new roles

Dynamic approach (with session policies)

- Create 1 IAM role: `TransferRole-Shared` with broad permissions to the entire bucket
- Use session policies to restrict each user to their specific folder at runtime
- Managing permissions requires updating one role or session policy template
- Adding new users requires no new roles, just user configuration

Implementation

Here's how you would implement the dynamic approach (using the `company-transfers` bucket as an example to be replaced with your actual Amazon S3 bucket):

To implement dynamic permission management

1. Create one shared IAM role with broad Amazon S3 permissions:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::company-transfers/*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::company-transfers"
    }
  ]
}
```

2. Create a session policy template that restricts access to the user's folder:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
```

```
      "Resource": "arn:aws:s3:::company-transfers/${transfer:Username}/*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::company-transfers",
      "Condition": {
        "StringLike": {
          "s3:prefix": "${transfer:Username}*"
        }
      }
    }
  ]
}
```

3. Configure each user with:

   - The shared IAM role
   - The session policy applied as follows:
     - *Service-managed users*: Use the API or CLI to apply the JSON via the Policy parameter when creating or modifying users (the console only offers predefined policy options)
     - *Custom identity provider users*: Either return it as part of the Lambda function response during authentication, or store it in AWS Secrets Manager as a key named "Policy" alongside the user's credentials
   - Home directory: `/company-transfers/${transfer:Username}/`

AWS CloudTrail logging for AWS Transfer Family

AWS Transfer Family integrates with both AWS CloudTrail and Amazon CloudWatch. CloudTrail and CloudWatch serve different but complementary purposes.

- This topic covers integration with CloudTrail , an AWS service that creates a record of actions taken within your AWS account. It continuously monitors and records API operations for activities like console sign-ins, AWS Command Line Interface commands, and SDK/API operations. This allows you to keep a log of who took what action, when, and from where. CloudTrail helps with auditing, access management, and regulatory compliance by providing a history of all activity in your AWS environment. For details, see the AWS CloudTrail User Guide.
- Amazon CloudWatch logging for AWS Transfer Family servers covers integration with CloudWatch, a monitoring service for AWS resources and applications. It collects metrics and logs to provide visibility into resource utilization, application performance, and overall system health. CloudWatch helps with operational tasks like troubleshooting issues, setting alarms and autoscaling. For details, see the Amazon CloudWatch User Guide.

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 action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API operations, so they don't appear in any specific order.

For an ongoing record of events in your AWS account, including events for AWS Transfer Family, 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 AWS 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 the following:

- 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 AWS Transfer Family actions are logged by CloudTrail and are documented in the Actions API reference. For example, calls to the `CreateServer`, `ListUsers` and `StopServer` actions 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 AWS Identity and Access Management 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.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Transfer Family. 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 AWS Transfer Family, 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

- Enable AWS CloudTrail logging
- Example log entry for creating a server
- Data access log examples

Enable AWS CloudTrail logging

You can monitor AWS Transfer Family API operations using AWS CloudTrail. By monitoring API operations, you can get useful security and operational information. If you have Amazon S3 object level logging enabled, `RoleSessionName` is contained in the Requester field as `[AWS:Role Unique Identifier]/username.sessionid@server-id`. For more information about AWS Identity and Access Management (IAM) role unique identifiers, see Unique identifiers in the *AWS Identity and Access Management User Guide*.

> **Important**
>
> The maximum length of the `RoleSessionName` is 64 characters. If the `RoleSessionName` is longer, the `server-id` gets truncated.

Enabling Amazon S3 data events

To track file operations performed through AWS Transfer Family on your Amazon S3 buckets, you need to enable data events for those buckets. Data events provide object-level API activity and are particularly useful for tracking file uploads, downloads, and other operations performed by AWS Transfer Family users.

To enable Amazon S3 data events for your AWS Transfer Family server:

1. Open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. In the navigation pane, choose **Trails**, and then select an existing trail or create a new one.
3. Under **Data events**, choose **Edit**.
4. For **Data event type**, select **S3**.
5. Choose the Amazon S3 buckets to log data events for. You can log data events for all buckets or specify individual buckets.
6. Choose whether to log **Read** events, **Write** events, or both.
7. Choose **Save changes**.

After enabling data events, you can access these logs in the Amazon S3 bucket configured for your CloudTrail trail. The logs include details such as the user who performed the action, the action timestamp, the specific object affected, and the `onBehalfOf` field that helps trace the `userId` for actions performed through AWS Transfer Family.

Example log entry for creating a server

The following example shows a CloudTrail log entry (in JSON format) that demonstrates the `CreateServer` action.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
```

```
        "type": "AssumedRole",
        "principalId": "AAAA4FFF5HHHHH6NNWWWW:user1",
        "arn": "arn:aws:sts::123456789102:assumed-role/Admin/user1",
        "accountId": "123456789102",
        "accessKeyId": "AAAA52C2WWWWWW3BB4Z",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-12-18T20:03:57Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AAAA4FFF5HHHHH6NNWWWW",
                "arn": "arn:aws:iam::123456789102:role/Admin",
                "accountId": "123456789102",
                "userName": "Admin"
            }
        }
    },
    "eventTime": "2024-02-05T19:18:53Z",
    "eventSource": "transfer.amazonaws.com",
    "eventName": "CreateServer",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "11.22.1.2",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/121.0.0.0 Safari/537.36",
    "requestParameters": {
        "domain": "S3",
        "hostKey": "HIDDEN_DUE_TO_SECURITY_REASONS",
        "protocols": [
            "SFTP"
        ],
        "protocolDetails": {
            "passiveIp": "AUTO",
            "tlsSessionResumptionMode": "ENFORCED",
            "setStatOption": "DEFAULT"
        },
        "securityPolicyName": "TransferSecurityPolicy-2020-06",
        "s3StorageOptions": {
            "directoryListingOptimization": "ENABLED"
        }
    },
    "responseElements": {
        "serverId": "s-1234abcd5678efghi"
```

```
    },
    "requestID": "6fe7e9b1-72fc-45b0-a7f9-5840268aeadf",
    "eventID": "4781364f-7c1e-464e-9598-52d06aa9e63a",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789102",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "transfer.us-east-1.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```

Data access log examples

When you enable Amazon S3 data events for your CloudTrail trail, you can track file operations performed through AWS Transfer Family. These logs help you monitor who accessed what data, when, and how.

Example log entry for successful data access

The following example shows a CloudTrail log entry for a successful file download operation through AWS Transfer Family.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAEXAMPLEID:TransferSessionUser",
        "arn": "arn:aws:sts::123456789012:assumed-role/TransferS3AccessRole/
TransferSessionUser",
        "accountId": "123456789012",
        "accessKeyId": "ASIAEXAMPLEKEY",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAEXAMPLEID",
                "arn": "arn:aws:iam::123456789012:role/TransferS3AccessRole",
                "accountId": "123456789012",
```

```
                "userName": "TransferS3AccessRole"
            },
            "attributes": {
                "creationDate": "2025-07-15T16:12:05Z",
                "mfaAuthenticated": "true"
            }
        },
        "invokedBy": "transfer.amazonaws.com"
    },
    "eventTime": "2025-07-15T16:15:22Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "GetObject",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "transfer.amazonaws.com",
    "userAgent": "transfer.amazonaws.com",
    "requestParameters": {
        "bucketName": "my-transfer-bucket",
        "key": "users/john.doe/reports/quarterly-report-2025-Q2.pdf",
        "Host": "my-transfer-bucket.s3.amazonaws.com",
        "x-amz-request-payer": "requester"
    },
    "responseElements": null,
    "additionalEventData": {
        "SignatureVersion": "SigV4",
        "CipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "bytesTransferredIn": 0,
        "bytesTransferredOut": 2458732,
        "x-amz-id-2":
 "EXAMPLE123456789+abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ="
    },
    "requestID": "EXAMPLE123456789",
    "eventID": "example12-3456-7890-abcd-ef1234567890",
    "readOnly": true,
    "resources": [
        {
            "type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:::my-transfer-bucket/users/john.doe/reports/quarterly-
report-2025-Q2.pdf"
        },
        {
            "accountId": "123456789012",
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::my-transfer-bucket"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data",
    "requestParameters": {
        "x-amz-onBehalfOf": "john.doe.sessionid@s-abcd1234efgh5678"
    }
}
```

In this example, note the following important fields:

- `eventName`: Indicates the S3 API operation that was performed (GetObject for a file download).
- `requestParameters.bucketName` and `requestParameters.key`: Show which S3 object was accessed.
- `additionalEventData.bytesTransferredOut`: Shows the size of the downloaded file in bytes.
- `requestParameters.x-amz-onBehalfOf`: Contains the AWS Transfer Family username and session ID, allowing you to trace which AWS Transfer Family user performed the action.

The `x-amz-onBehalfOf` field is particularly important as it links the S3 API call back to the specific AWS Transfer Family user who initiated the action. This field follows the format `username.sessionid@server-id`, where:

- `username` is the AWS Transfer Family username.
- `sessionid` is a unique identifier for the user's session.
- `server-id` is the ID of the AWS Transfer Family server.

Common data access operations

When monitoring data access through AWS Transfer Family, you'll typically see the following S3 API operations in your CloudTrail logs:

Common S3 operations in AWS Transfer Family logs

S3 API Operation	AWS Transfer Family Action	Description
GetObject	File download	User downloaded a file from the server
PutObject	File upload	User uploaded a file to the server
DeleteObject	File deletion	User deleted a file from the server
ListObjects or ListObjectsV2	Directory listing	User listed files in a directory
CopyObject	File copy	User copied a file within the server

By monitoring these operations in your CloudTrail logs, you can track all file activities performed through your AWS Transfer Family server, helping you meet compliance requirements and detect unauthorized access.

Amazon CloudWatch logging for AWS Transfer Family servers

Amazon CloudWatch is a powerful monitoring and observability service that provides comprehensive visibility into your AWS resources, including AWS Transfer Family.

- Real-time monitoring: CloudWatch monitors Transfer Family resources and applications in real-time, allowing you to track and analyze their performance.
- Metrics collection: CloudWatch collects and tracks various metrics for your resources and applications, which are variables you can measure and use for analysis.
- CloudWatch home page: The CloudWatch home page automatically displays metrics about Transfer Family and other AWS services you use, providing a centralized view of your monitoring data.
- Custom dashboards: You can create custom dashboards in CloudWatch to display metrics specific to your custom applications and the resources you choose to monitor.
- Alarms and notifications: CloudWatch allows you to create alarms that monitor your metrics and trigger notifications or automated actions when certain thresholds are breached. This can be useful for monitoring file transfer activity in your Transfer Family servers and scaling resources accordingly.
- Cost optimization: You can use the data collected by CloudWatch to identify under-utilized resources and take actions, such as stopping or deleting instances, to optimize your costs.

Overall, the comprehensive monitoring capabilities in CloudWatch make it a valuable tool for managing and optimizing your Transfer Family infrastructure and the applications running on it.

Details for CloudWatch logging for Transfer Family web apps is available in CloudTrail logging for Transfer Family web apps.

Types of CloudWatch logging for Transfer Family

Transfer Family provides two ways to log events to CloudWatch:

- JSON structured logging
- Logging via a logging role

For Transfer Family servers and managed workflows, you can choose the logging mechanism that you prefer. For connectors, only logging roles are supported.

JSON structured logging

For logging server and managed workflow events, we recommend using JSON structured logging. This provides a more comprehensive logging format that enables CloudWatch log querying. For servers, this type of logging requires that the IAM policy for the user that creates the server (or edits the server's logging configuration) contains the following permissions:

- `logs:CreateLogDelivery`
- `logs:DeleteLogDelivery`
- `logs:DescribeLogGroups`
- `logs:DescribeResourcePolicies`
- `logs:GetLogDelivery`
- `logs:ListLogDeliveries`
- `logs:PutResourcePolicy`
- `logs:UpdateLogDelivery`

The following is an example policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

For details on setting up JSON structured logging, see Creating, updating, and viewing logging for servers.

Logging role

For connectors, and for managed workflows when you are not using JSON structured logging, you need to specify a logging role. To set access, you create a resource-based IAM policy and an IAM role that provides that access information. The following is an example policy for an AWS account that can log server events.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/transfer/*"
        }
    ]
}
```

For details on configuring a logging role to log workflow events see Managing logging for workflows.

Creating, updating, and viewing logging for servers

For all AWS Transfer Family servers, we provide structured logging. We recommend that you use structured logging for all new and existing Transfer Family servers. Benefits of using structured logging include the following:

- Receive logs in a structured JSON format.

- Query your logs with Amazon CloudWatch Logs Insights, which automatically discovers JSON formatted fields.
- Share log groups across AWS Transfer Family resources allows you to combine log streams from multiple servers into a single log group, making it easier to manage your monitoring configurations and log retention settings.
- Create aggregated metrics and visualizations that can be added to CloudWatch dashboards.
- Track usage and performance data by using log groups to create consolidated log metrics, visualizations, and dashboards.

To enable logging for a workflow that does not have a structured log destination configured, you must use a logging role on the server that the workflow is attached to. If the workflow has a structured log destination configured (a CloudWatch Logs log group specified in its `StructuredLogDestinations` parameter), Transfer Family delivers the workflow execution logs to that log group and a logging role is not required. For details, see Managing logging for workflows.

> **Note**
>
> When you add a logging role, the logging group is always `/aws/transfer/your-serverID`, and can't be changed. This means, that unless you are sending your structured server logs to the same group, you will be logging to two separate logging groups.
> If you know that you are going to associate a workflow with your server, and thus need to add a logging role, you can set up structured logging to log to the default log group of `/aws/transfer/your-serverID`.
> To modify your logging group, see StructuredLogDestinations in the *AWS Transfer Family API Reference*.

If you create a new server by using the Transfer Family console, logging is enabled by default. After you create the server, you can use the `UpdateServer` API operation to change your logging configuration. For details, see StructuredLogDestinations.

For a workflow that does not have a structured log destination configured, if you want logging enabled, you must specify a logging role:

- If you associate a workflow with a server, using either the `CreateServer` or `UpdateServer` API operation, the system does not automatically create a logging role. If you want to log the events

of a workflow that does not have a structured log destination, you need to explicitly attach a logging role to the server.

- If you create a server using the Transfer Family console and you attach a workflow, logs are sent to a log group that contains the server ID in the name. The format is `/aws/transfer/server-id`, for example, `/aws/transfer/s-1111aaaa2222bbbb3`. The server logs can be sent to this same log group or a different one.

Logging considerations for creating and editing servers in the console

- New servers created through the console only support structured JSON logging, unless a workflow is attached to the server.
- *No logging* is not an option for new servers that you create in the console.
- Existing servers can enable structured JSON logging through the console at any time.
- Enabling structured JSON logging through the console disables the existing logging method, so as to not double charge customers. The exception is if a workflow is attached to the server.
- If you enable structured JSON logging, you cannot later disable it through the console.
- If you enable structured JSON logging, you can change the log group destination through the console at any time.
- If you enable structured JSON logging, you cannot edit the logging role through the console if you have enabled both logging types through the API. The exception is if your server has a workflow attached. However, the logging role does continue to appear in **Additional details**.

Logging considerations for creating and editing servers using the API or SDK

- If you create a new server through the API, you can configure either or both types of logging, or choose no logging.
- For existing servers, enable and disable structured JSON logging at any time.
- You can change the log group through the API at any time.
- You can change the logging role through the API at any time.

To enable structured logging, you must be logged into an account with the following permissions

- `logs:CreateLogDelivery`

- `logs:DeleteLogDelivery`
- `logs:DescribeLogGroups`
- `logs:DescribeResourcePolicies`
- `logs:GetLogDelivery`
- `logs:ListLogDeliveries`
- `logs:PutResourcePolicy`
- `logs:UpdateLogDelivery`

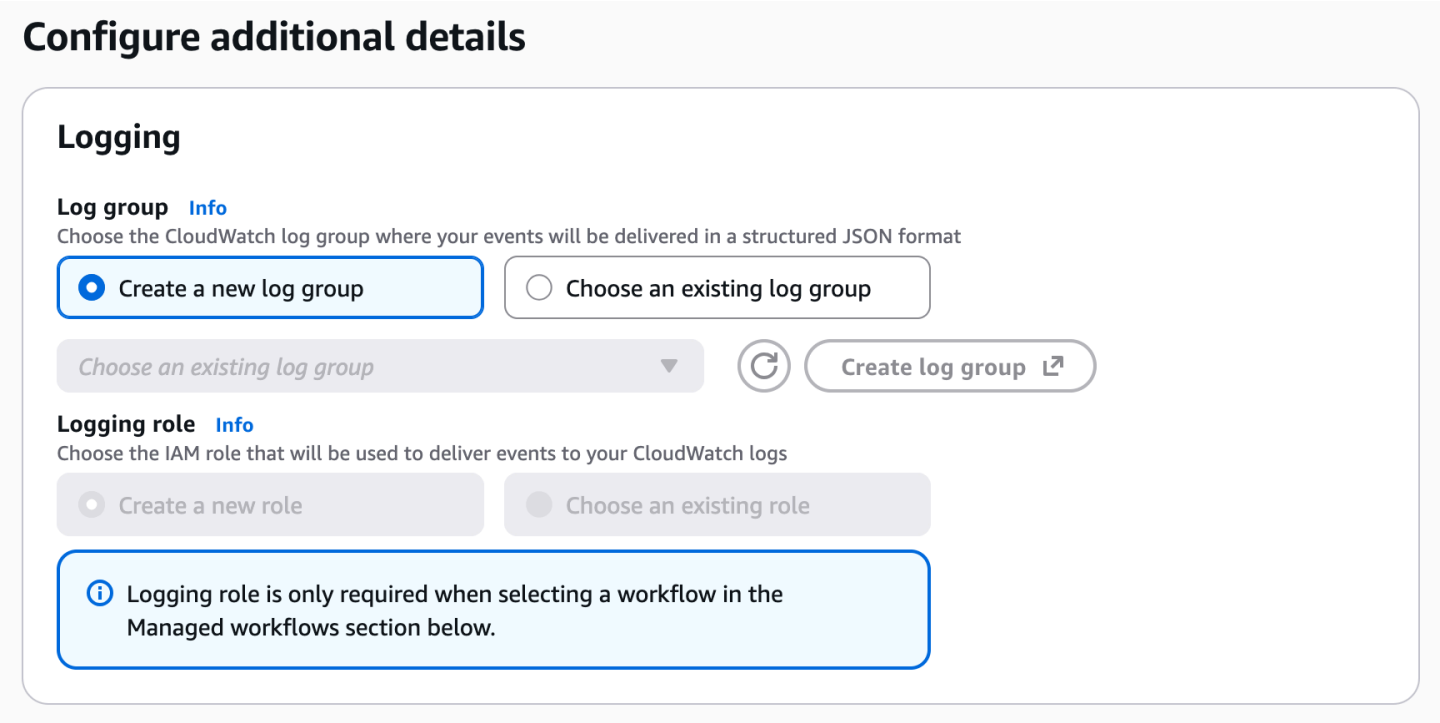
An example policy is available in the section Configure CloudWatch logging role.

Topics

- Creating logging for servers
- Updating logging for a server
- Viewing the server configuration

Creating logging for servers

When you create a new server, on the **Configure additional details** page, you can specify an existing log group, or create a new one.



If you choose **Create log group**, the CloudWatch console (https://console.aws.amazon.com/cloudwatch/) opens to the **Create log group** page. For details, see Create a log group in CloudWatch Logs.

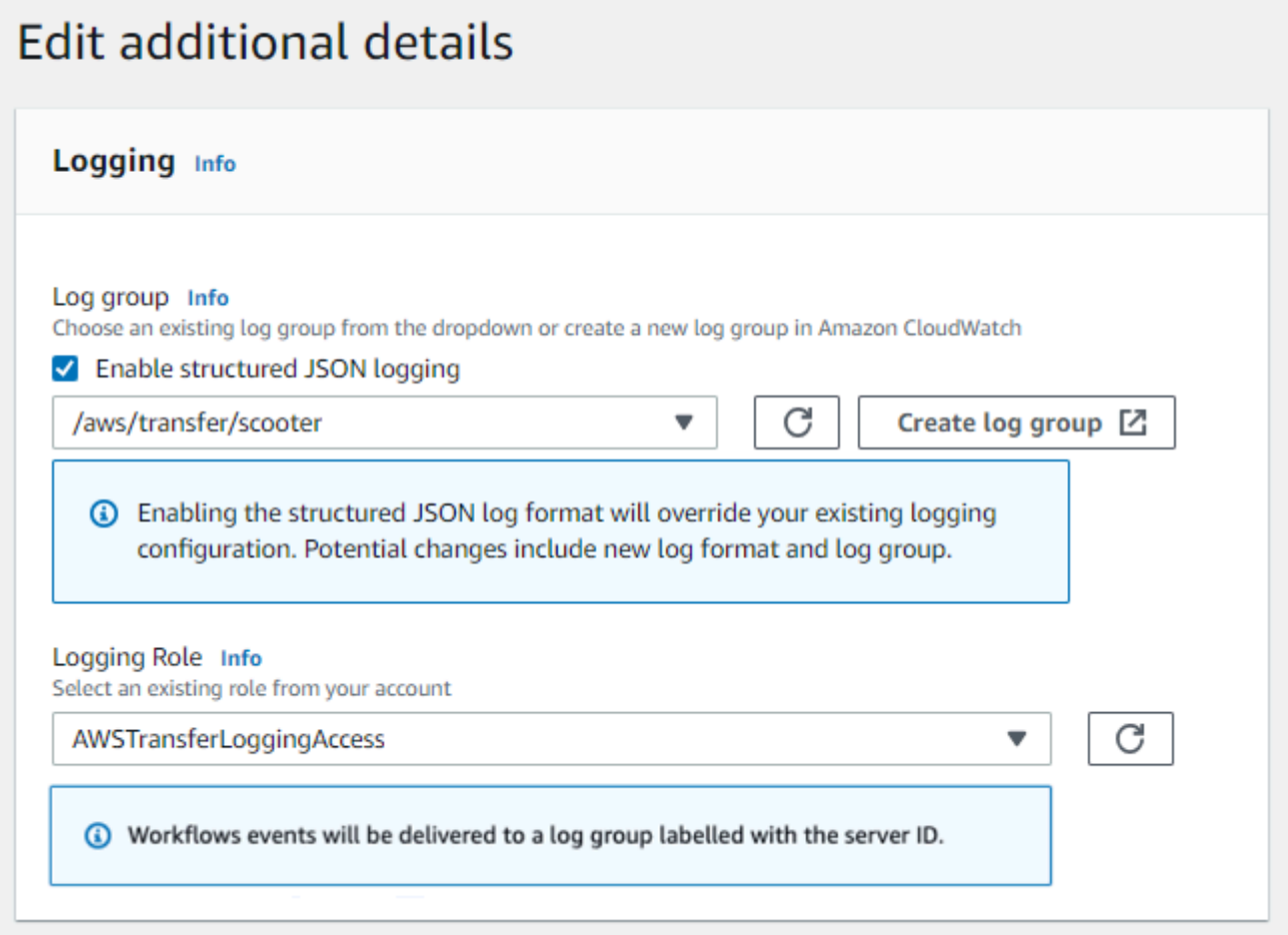
Updating logging for a server

The details for logging depend on the scenario for your update.

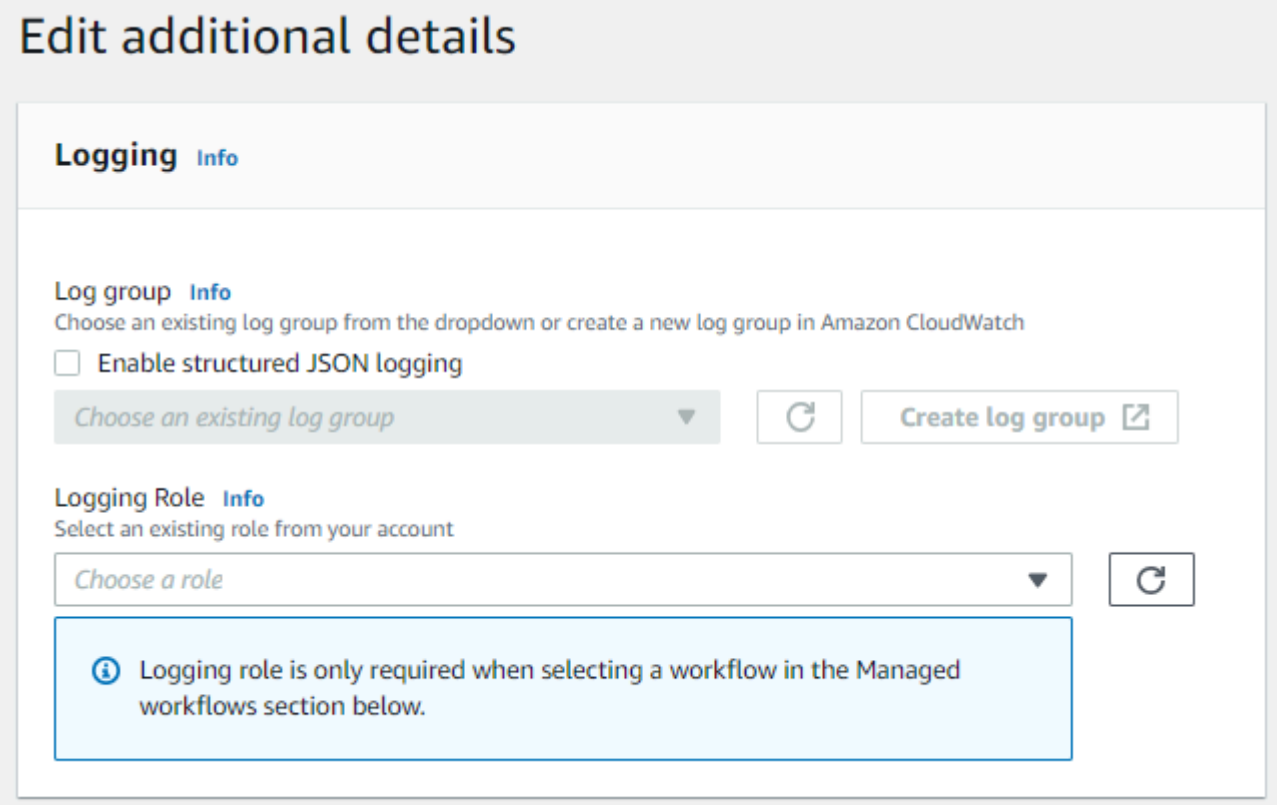
> **Note**
>
> When you opt into structured JSON logging, there can be a delay, in rare cases, where Transfer Family stops logging in the old format, but takes some time to start logging in the new JSON format. This can result in events that don't get logged. There won't be any service disruptions, but you should be careful transferring files during the first hour after changing your logging method, as logs could be dropped.

If you are editing an existing server, your options depend on the state of the server.

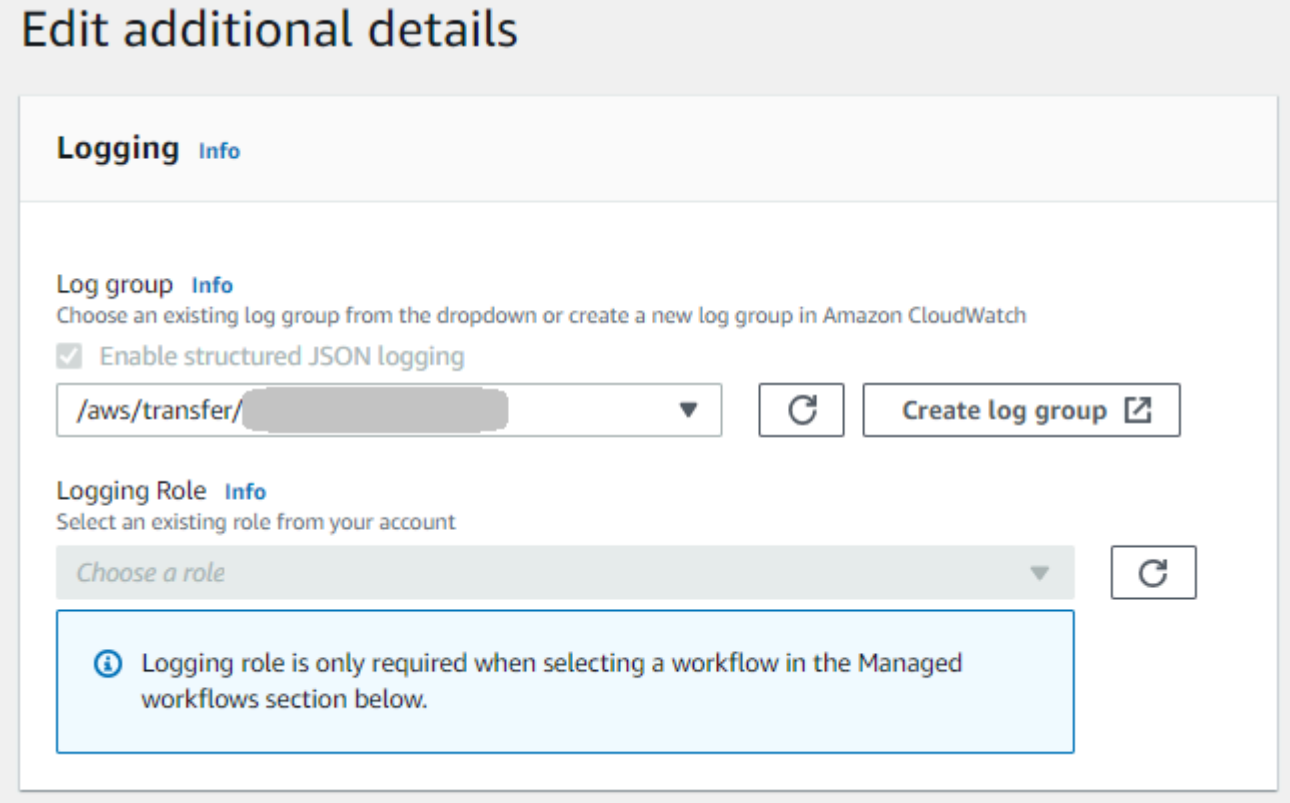
- The server already has a logging role enabled, but does not have Structured JSON logging enabled.



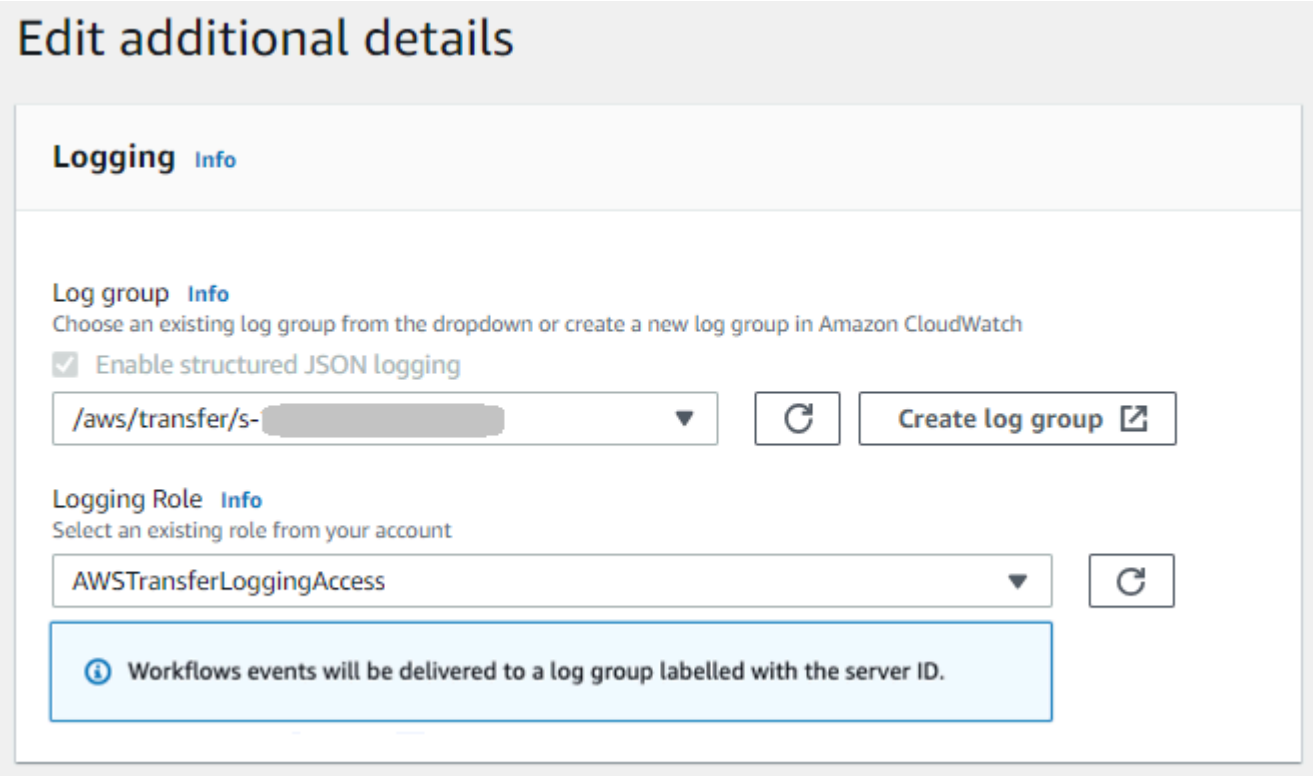
- The server does not have any logging enabled.



- The server already has Structured JSON logging enabled, but does not have a logging role specified.



- The server already has Structured JSON logging enabled, and also has a logging role specified.

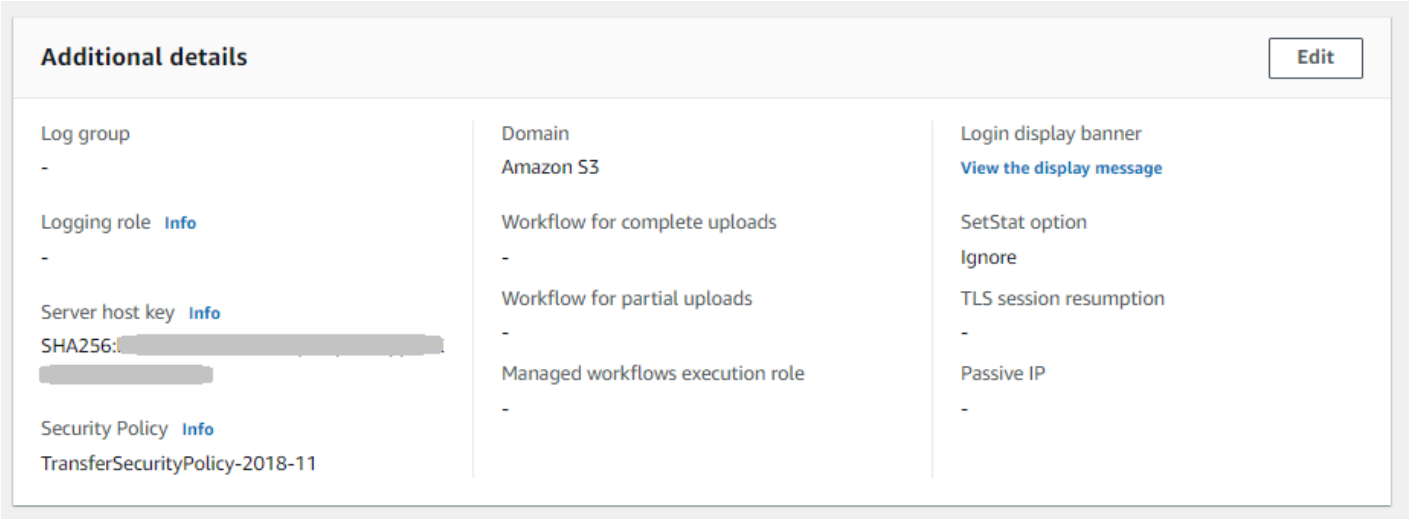


Viewing the server configuration

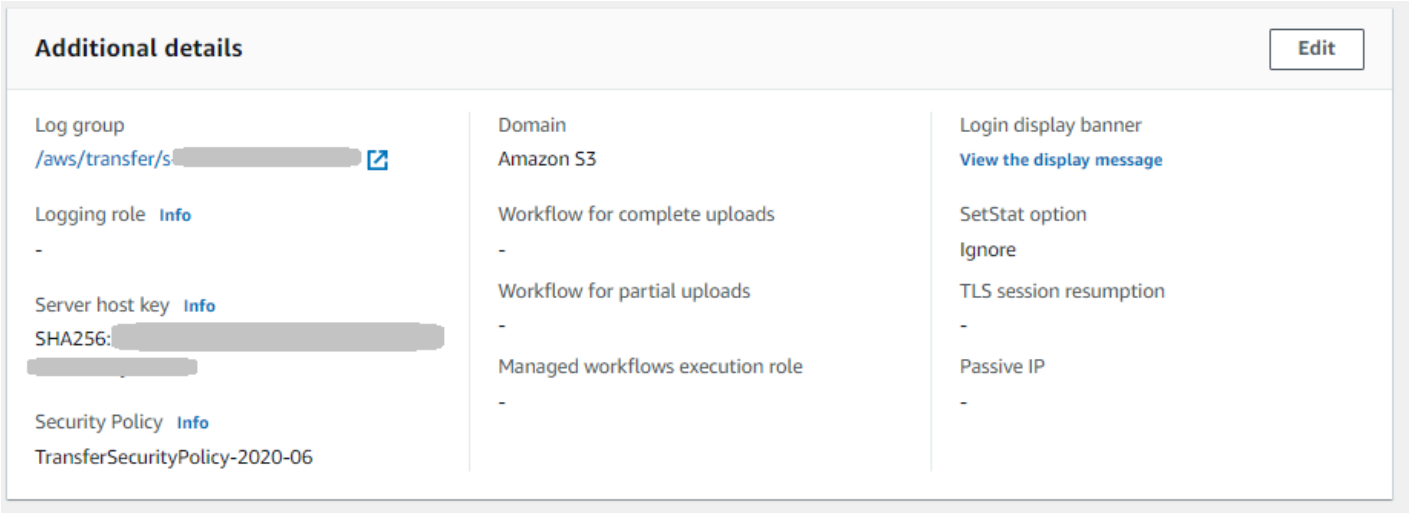
The details for the server configuration page depend on your scenario:

Depending on your scenario, the server configuration page might look like one of the following examples:

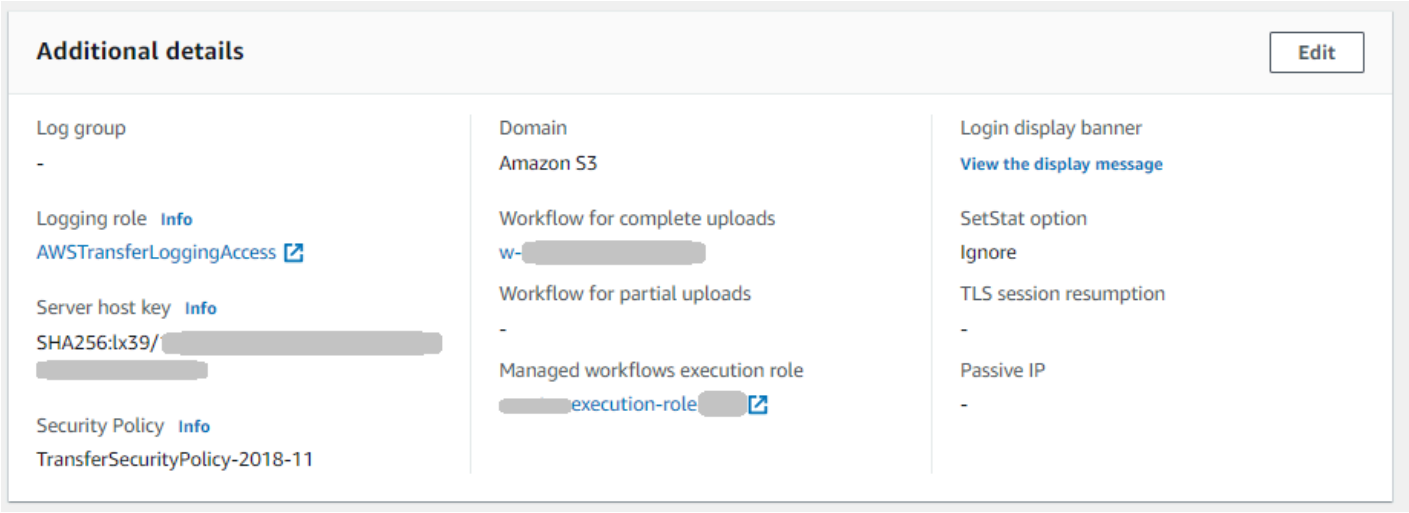
- No logging is enabled.



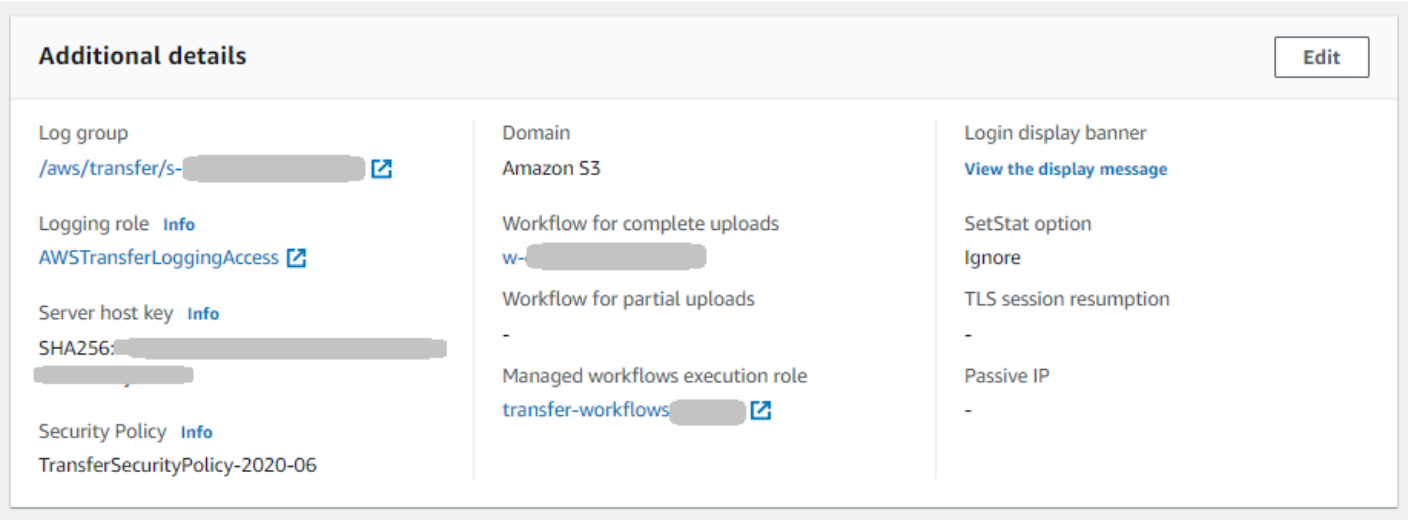
- Structured JSON logging is enabled.



- Logging role is enabled, but structured JSON logging is not enabled.



- Both types of logging (logging role and structured JSON logging) are enabled.



Managing logging for workflows

CloudWatch provides consolidated auditing and logging for workflow progress and results. Additionally, AWS Transfer Family provides several metrics for workflows. You can view metrics for how many workflows executions started, completed successfully, and failed in the previous minute. All of the CloudWatch metrics for Transfer Family are described in Using CloudWatch metrics for Transfer Family servers.

AWS Transfer Family provides two ways to deliver managed workflow execution logs to Amazon CloudWatch:

- **A log group that you specify (structured log destination)** – You specify a CloudWatch Logs log group as a structured log destination for the workflow, and Transfer Family delivers the workflow execution logs to that log group for you. This option does not require a logging role. To use it, provide the Amazon Resource Name (ARN) of a log group in the `StructuredLogDestinations` parameter when you create the workflow (see StructuredLogDestinations in the *AWS Transfer Family API Reference*). The log group must be in the same AWS account and AWS Region as the workflow.

  To configure a structured log destination, the IAM policy for the principal that creates the workflow must contain the following permissions:

  - `logs:CreateLogDelivery`
  - `logs:DeleteLogDelivery`

- `logs:DescribeLogGroups`
- `logs:DescribeResourcePolicies`
- `logs:GetLogDelivery`
- `logs:ListLogDeliveries`
- `logs:PutResourcePolicy`
- `logs:UpdateLogDelivery`

- **A logging role** – You attach an IAM logging role to the server that the workflow is attached to. Transfer Family assumes the role to write the workflow execution logs to a log group named `/aws/transfer/`*`server-id`*. For details, see Configure CloudWatch logging role.

Both mechanisms write the same workflow execution log content in the same JSON format. The difference is only where the logs are delivered: to the log group that you specify, or to the `/aws/transfer/`*`server-id`* log group on the server by way of the logging role.

View Amazon CloudWatch logs for workflows

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Logs**, then choose **Log groups**.
3. On the **Log groups** page, on the navigation bar, choose the correct Region for your AWS Transfer Family server.
4. Choose the log group that contains your workflow logs.

   - If the workflow uses a structured log destination, choose the log group whose ARN you specified in the `StructuredLogDestinations` parameter of the workflow.
   - If the workflow uses a logging role on the server, choose the log group that corresponds to your server. For example, if your server ID is `s-1234567890abcdef0`, your log group is `/aws/transfer/s-1234567890abcdef0`.
5. On the log group details page, the most recent log streams are displayed. The log stream name for the workflow depends on which logging mechanism the workflow uses:

   - For a structured log destination, the workflow log stream is named *`workflowID`*`/`*`executionID`*. For example:

     ```
     w-abcdef01234567890/021345abcdef6789
     ```

- For a logging role, the server log group contains two log streams for the user that you are exploring: one for each Secure Shell (SSH) File Transfer Protocol (SFTP) session, and one for the workflow that is being executed. The format for the workflow log stream is *username*.*workflowID*.*uniqueStreamSuffix*.

For example, if your user is `mary-major`, you have the following log streams:

```
mary-major-east.1234567890abcdef0
mary.w-abcdef01234567890.021345abcdef6789
```

> **Note**
>
> The 16-digit alphanumeric identifiers listed in this example are fictitious. The values that you see in Amazon CloudWatch are different.

The **Log events** page for a workflow log stream contains the details for the workflow execution. If the workflow uses a logging role, the `mary-major-usa-east.1234567890abcdef0` stream displays the details for each user session, and the `mary.w-abcdef01234567890.021345abcdef6789` stream contains the details for the workflow. If the workflow uses a structured log destination, the `w-abcdef01234567890/021345abcdef6789` stream contains the details for the workflow.

The following is a sample log stream for `mary.w-abcdef01234567890.021345abcdef6789`, based on a workflow (`w-abcdef01234567890`) that contains a copy step.

```
{
    "type": "ExecutionStarted",
    "details": {
        "input": {
            "initialFileLocation": {
                "bucket": "amzn-s3-demo-bucket",
                "key": "mary/workflowSteps2.json",
                "versionId": "version-id",
                "etag": "etag-id"
            }
        }
    },
    "workflowId":"w-abcdef01234567890",
```

```
    "executionId":"execution-id",
    "transferDetails": {
        "serverId":"s-server-id",
        "username":"mary",
        "sessionId":"session-id"
    }
},
{
    "type":"StepStarted",
    "details": {
        "input": {
            "fileLocation": {
                "backingStore":"S3",
                "bucket":"amzn-s3-demo-bucket",
                "key":"mary/workflowSteps2.json",
                "versionId":"version-id",
                "etag":"etag-id"
            }
        },
        "stepType":"COPY",
        "stepName":"copyToShared"
    },
    "workflowId":"w-abcdef01234567890",
    "executionId":"execution-id",
    "transferDetails": {
        "serverId":"s-server-id",
        "username":"mary",
        "sessionId":"session-id"
    }
},
{
    "type":"StepCompleted",
    "details":{
        "output":{},
        "stepType":"COPY",
        "stepName":"copyToShared"
    },
    "workflowId":"w-abcdef01234567890",
    "executionId":"execution-id",
    "transferDetails":{
        "serverId":"server-id",
        "username":"mary",
        "sessionId":"session-id"
    }
```

```
},
{
    "type":"ExecutionCompleted",
    "details": {},
    "workflowId":"w-abcdef01234567890",
    "executionId":"execution-id",
    "transferDetails":{
        "serverId":"s-server-id",
        "username":"mary",
        "sessionId":"session-id"
    }
}
```

Configure CloudWatch logging role

To set access, you create a resource-based IAM policy and an IAM role that provides that access information.

To enable Amazon CloudWatch logging, you start by creating an IAM policy that enables CloudWatch logging. You then create an IAM role and attach the policy to it. You can do this when you are creating a server or by editing an existing server. For more information about CloudWatch, see What is Amazon CloudWatch? and What is Amazon CloudWatch logs? in the *Amazon CloudWatch User Guide*.

Use the following example IAM policies to allow CloudWatch logging.

Use a logging role

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/transfer/*"
        }
```

```
    ]
}
```

Use structured logging

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

In the preceding example policy, for the **Resource**, replace the *region-id* and *AWS account* with your values. For example, **"Resource": "arn:aws::logs:us-east-1:111122223333:log-group:/aws/transfer/*"**

You then create a role and attach the CloudWatch Logs policy that you created.

To create an IAM role and attach a policy

1. In the navigation pane, choose **Roles**, and then choose **Create role**.

   On the **Create role** page, make sure that **AWS service** is chosen.
2. Choose **Transfer** from the service list, and then choose **Next: Permissions**. This establishes a trust relationship between AWS Transfer Family and the IAM role. Additionally, add `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the *confused deputy* problem. See the following documentation for more details:

- Procedure for establishing a trust relationship with AWS Transfer Family: To establish a trust relationship
- Description for confused deputy problem: the confused deputy problem

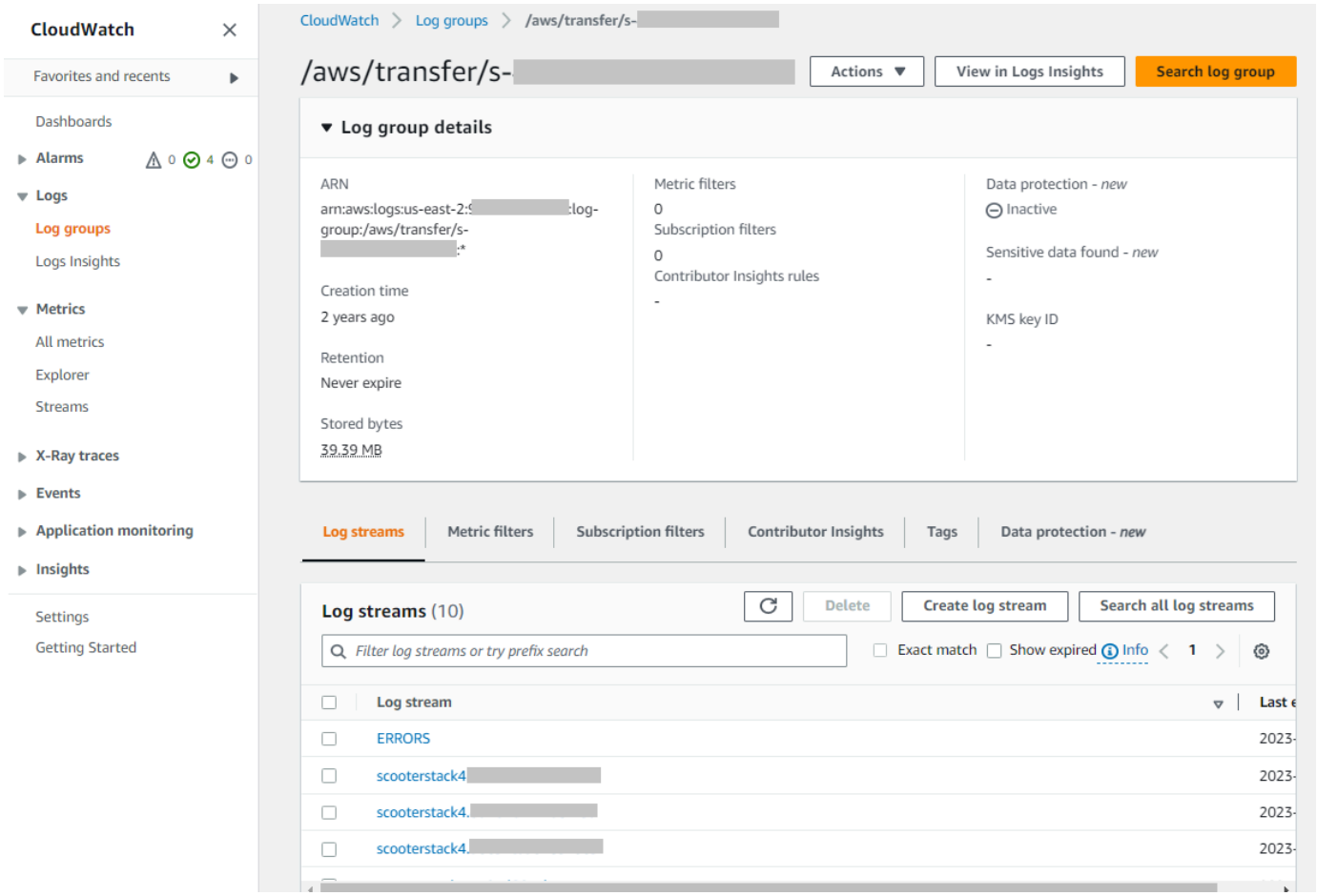
3. In the **Attach permissions policies** section, locate and choose the CloudWatch Logs policy that you just created, and choose **Next: Tags**.
4. (Optional) Enter a key and value for a tag, and choose **Next: Review**.
5. On the **Review** page, enter a name and description for your new role, and then choose **Create role**.
6. To view the logs, choose the **Server ID** to open the server configuration page, and choose **View logs**. You are redirected to the CloudWatch console where you can see your log streams.

On the CloudWatch page for your server, you can see records of user authentication (success and failure), data uploads (PUT operations), and data downloads (GET operations).

Viewing Transfer Family log streams

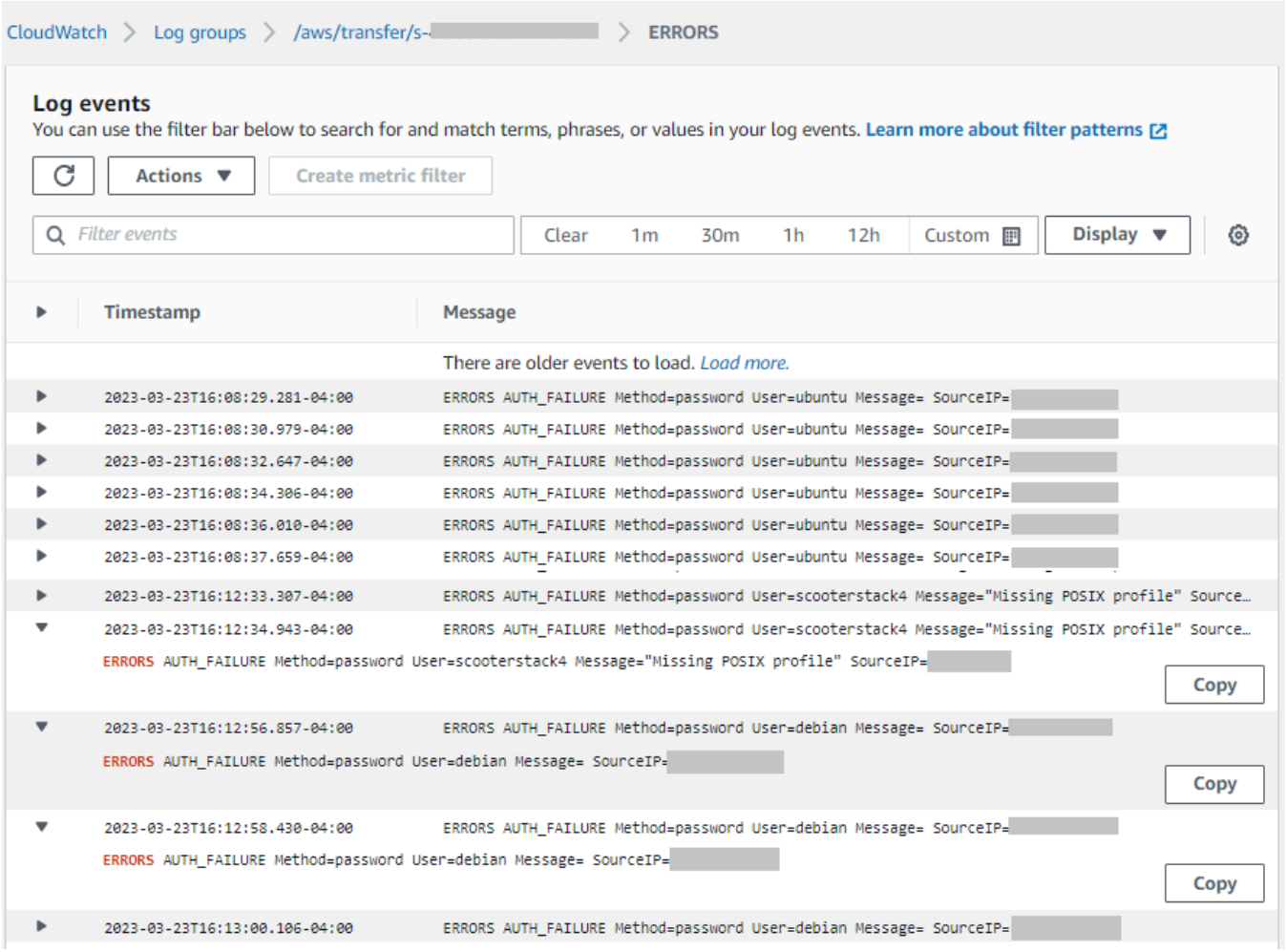
To view your Transfer Family server logs

1. Navigate to the details page for a server.
2. Choose **View logs**. This opens Amazon CloudWatch.
3. The log group for your selected server is displayed.

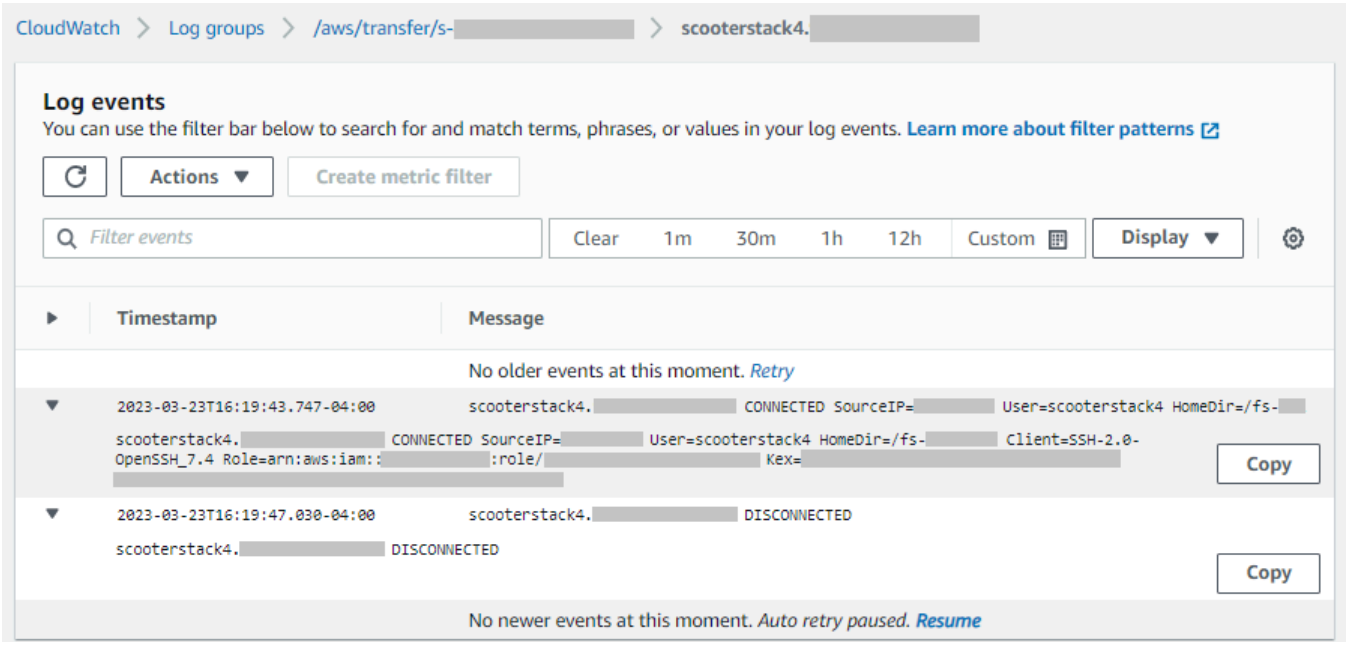


4. You can select a log stream to display details and individual entries for the stream.

   - If there is a listing for **ERRORS**, you can choose it to view details for the latest errors for the server.

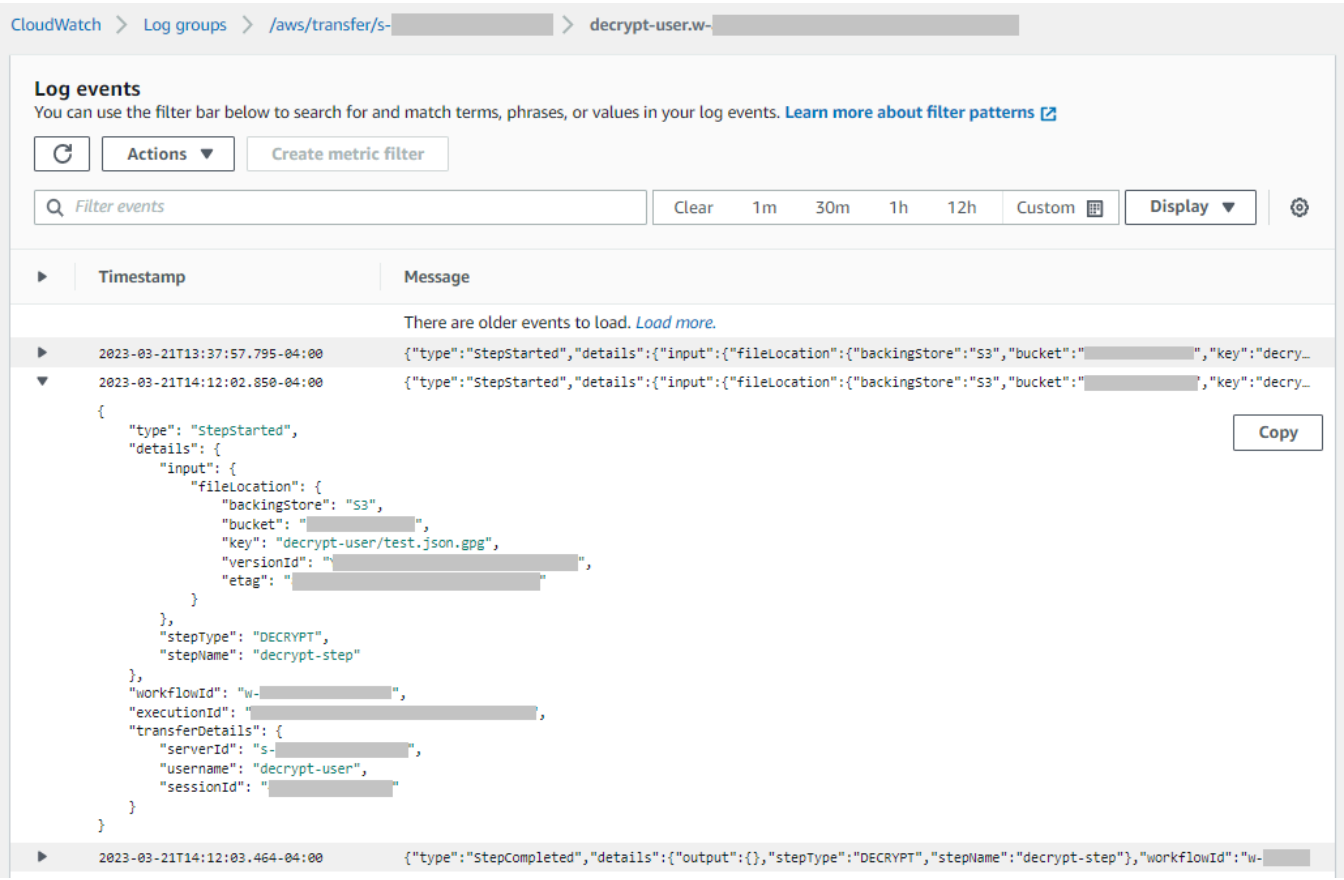


- Choose any other entry to see an example log stream.



- If your server has a managed workflow associated with it, you can view logs for the workflow runs.

> **Note**
>
> The format for the workflow log stream depends on which logging mechanism the workflow uses:
>
> - If the workflow uses a logging role, the format is *username*.*workflowId*.*uniqueStreamSuffix*. For example, **decrypt-user.w-a1111222233334444.aaaa1111bbbb2222** could be the name of a log stream for user **decrypt-user** and workflow **w-a1111222233334444**.
> - If the workflow uses a structured log destination, the format is *workflowId*/*executionId*. For example, **w-a1111222233334444/aaaa1111bbbb2222** could be the name of a log stream for workflow **w-a1111222233334444**.



Note

For any expanded log entry, you can copy the entry to the clipboard by choosing **Copy**. For more details about CloudWatch logs, see Viewing log data.

Creating Amazon CloudWatch alarms

The following example shows how to create Amazon CloudWatch alarms using the AWS Transfer Family metric, `FilesIn`.

CDK

```
new cloudwatch.Metric({
  namespace: "AWS/Transfer",
  metricName: "FilesIn",
  dimensionsMap: { ServerId: "s-00000000000000000" },
  statistic: "Average",
  period: cdk.Duration.minutes(1),
}).createAlarm(this, "AWS/Transfer FilesIn", {
  threshold: 1000,
  evaluationPeriods: 10,
  datapointsToAlarm: 5,
  comparisonOperator:
 cloudwatch.ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD,
});
```

CloudFormation

```
Type: AWS::CloudWatch::Alarm
Properties:
  Namespace: AWS/Transfer
  MetricName: FilesIn
  Dimensions:
    - Name: ServerId
      Value: s-00000000000000000
  Statistic: Average
  Period: 60
  Threshold: 1000
  EvaluationPeriods: 10
  DatapointsToAlarm: 5
```

```
ComparisonOperator: GreaterThanOrEqualToThreshold
```

Logging Amazon S3 API operations to S3 access logs

> **Note**
>
> This section does not apply to Transfer Family web apps.

If you are using Amazon S3 access logs to identify S3 requests made on behalf of your file transfer users, `RoleSessionName` is used to display which IAM role was assumed to service the file transfers. It also displays additional information such as the user name, session id, and server-id used for the transfers. The format is `[AWS:Role Unique Identifier]/username.sessionid@server-id` and is contained in the Requester field. For example, the following are the contents for a sample Requester field from an S3 access log for a file that was copied to the S3 bucket.

`arn:aws:sts::AWS-Account-ID:assumed-role/`*`IamRoleName`*`/username.sessionid@server-id`

In the Requester field above, it shows the IAM Role called `IamRoleName`. For more information about IAM role unique identifiers, see Unique identifiers in the *AWS Identity and Access Management User Guide*.

Examples to limit confused deputy problem

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. For more details, see Cross-service confused deputy prevention.

> **Note**
>
> In the following examples, replace each *`user input placeholder`* with your own information.
> In these examples, you can remove the ARN details for a workflow if your server doesn't have any workflows attached to it.

The following example logging/invocation policy allows any server (and workflow) in the account to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllServersWithWorkflowAttached",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                   "aws:SourceArn": [
                     "arn:aws:transfer:us-west-2:111122223333:server/*",
                     "arn:aws:transfer:us-west-2:111122223333:workflow/*"
                   ]
                }
            }
        }
    ]
}
```

The following example logging/invocation policy allows a specific server (and workflow) to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSpecificServerWithWorkflowAttached",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                   "aws:SourceArn": [
                       "arn:aws:transfer:us-west-2:111122223333:server/server-id",
                       "arn:aws:transfer:us-west-2:111122223333:workflow/workflow-
id"
                   ]
                }
            }
        }
    ]
}
```

CloudWatch log structure for Transfer Family

This topic describes the fields that are populated in Transfer Family logs: both for JSON structured log entries and legacy log entries.

Topics

- JSON structured logs for Transfer Family
- Legacy logs for Transfer Family

JSON structured logs for Transfer Family

The following table contains details for log entry fields for Transfer Family SFTP/FTP/FTPS actions, in the new JSON structured log format.

Field	Description	Example entry
activity-type	The action by the user	The available activity types are as follows: `AUTH_FAILURE` , `CONNECTED` , `CONNECTION_REJECTED` , `DISCONNECTED` , `ERROR`,

Field	Description	Example entry
		`EXIT_REASON` , `CLOSE`, `CREATE_SYMLINK` , `DELETE`, `MKDIR`, `OPEN`, `PARTIAL_CLOSE` , `RENAME`, `RMDIR`, `SETSTAT`, `TLS_RESUME_FAILURE` .
bytes-in	Number of bytes uploaded by the user	29238420042
bytes-out	Number of bytes downloaded by the user	23094032490328
ciphers	Specifies the SSH cipher negotiated for the connection (available ciphers are listed in Cryptographic algorithms)	aes256-gcm@openssh.com
client	The user's client software	SSH-2.0-OpenSSH_7.4
home-dir	The directory that the end user lands on when they connect to the endpoint if their home directory type is `PATH`: if they have a logical home directory, this value is always `/`	/user-home-bucket/test
kex	Specifies the negotiated SSH key exchange (KEX) for the connection (available KEX are listed in Cryptographic algorithms)	diffie-hellman-group14-sha256
message	Provides more information related to the error	*`<string>`*

Field	Description	Example entry
method	The authentication method	publickey
mode	Specifies how a client opens a file	CREATE | TRUNCATE | WRITE
operation	The client operation on a file	OPEN | CLOSE
path	Actual file path affected	/amzn-s3-demo-bucket/test-file-1.pdf
proxy-protocol-v2-header	Present only when the connection included a PROXY protocol v2 (PPv2) header. The value is `ignored` when the server's `SftpMode` is `NONE`, and `applied` when `SftpMode` is `PROXY_PROTOCOL_V2_ENFORCED` and the server honored the header. For more information, see Working with Network Load Balancers.	ignored
ssh-public-key	The public key body for the user that is connecting	AAAAC3NzaC1lZDI1NTE5AAAAIA9OY0qV6XYVHaaOiWAcj2spDJVbgjrqDPY4pxd6GnHl

Field	Description	Example entry
ssh-public-key-fingerprint	The public key fingerprint, as shown in the console for service-managed users when listing their user keys. **Note** In the console, the fingerprint is displayed with the padding characters (if any): from 0 to 3 equal signs (=) at the end. In the log entry, this padding is stripped from the output.	SHA256:BY3gNMHwTfjd4n2VuT4pTyLOk82zWZj4KEYEu7y4r/0
ssh-public-key-type	Type of public key: Transfer Family supports RSA-, ECDSA-, and ED25519-formatted keys	ssh-ed25519
resource-arn	A system-assigned, unique identifier for a specific resource (for example, a server)	arn:aws:transfer:ap-northeast-1:12346789012:server/s-1234567890akeu2js2
role	The IAM role of the user	arn:aws:iam::0293883675:role/testuser-role

Field	Description	Example entry
session-id	A system-assigned, unique identifier for a single session. For SFTP, a session is one SSH connection. All SFTP channels that run over that connection share the same `session-id` .	9ca9a0e1cec6ad9d
source-ip	Client IP address	18.323.0.129
user	The end user's username	myname192
user-policy	The permissions specified for the end user: this field is populated if the user's policy is a session policy.	The JSON code for the session policy that is being used

Legacy logs for Transfer Family

The following table contains details for log entries for various Transfer Family actions.

> **Note**
>
> These entries are not in the new JSON structured log format.

The following table contains details for log entries for various Transfer Family actions, in the new JSON structured log format.

Action	Corresponding logs within Amazon CloudWatch Logs
Authentication failures	ERRORS AUTH_FAILURE Method=publickey User=lhr Message="RSA SHA256:Lf

Action	Corresponding logs within Amazon CloudWatch Logs
	z3R2nmLY4raK+b7Rb1rSvUIbAE+a+Hxg0c7l 1JIZ0" SourceIP=3.8.172.211
COPY/TAG/DELETE/DECRYPT workflow	{"type":"StepStarted","details":{"input":{"fi leLocation":{"backingStore":"EFS","filesystem Id":"fs-12345678","path":"/lhr/regex.py"}},"s tepType":"TAG","stepName":"successfu l_tag_step"},"workflowId":"w-1111aaa a2222bbbb3","executionId":"81234abcd-1234- efgh-5678-ijklmnopqr90","transferDetails": {"serverId":"s-1234abcd5678efghi","usernam e":"lhr","sessionId":"1234567890abcdef0"}}
Custom step workflow	{"type":"CustomStepInvoked","details":{"outpu t":{"token":"MzM4Mjg5YWUtYTEzMy00YjI zLWI3OGMtYzU4OGI2ZjQyMzE5"},"stepTyp e":"CUSTOM","stepName":"efs-s3_copy_ 2"},"workflowId":"w-9283e49d33297c3f 7","executionId":"1234abcd-1234-efgh-5678- ijklmnopqr90","transferDetails":{"serverId":"s- zzzz1111aaaa22223","username":"lhr"," sessionId":"1234567890abcdef0"}}
Deletes	lhr.33a8fb495ffb383b DELETE Path=/bucket/ user/123.jpg
Downloads	lhr.33a8fb495ffb383b OPEN Path=/bucket/ user/123.jpg Mode=READ llhr.33a8fb495ffb383b CLOSE Path=/bucket/ user/123.jpg BytesOut=3618546

Action	Corresponding logs within Amazon CloudWatch Logs
Logins/Logouts	user.914984e553bcddb6 CONNECTED SourceIP=1.22.111.222 User=lhr HomeDir=LOGICAL Client=SSH-2.0-OpenSSH_7.4 Role=arn:aws::iam::123456789012:role/sftp-s3-access ProxyProtocolV2Header=ignored user.914984e553bcddb6 DISCONNECTED The `ProxyProtocolV2Header` field is present only when the connection included a PROXY protocol v2 (PPv2) header. For more information, see the `proxy-protocol-v2-header` field description earlier in this topic.
Renames	lhr.33a8fb495ffb383b RENAME Path=/bucket/user/lambo.png NewPath=/bucket/user/ferrari.png
Sample workflow error log	{"type":"StepErrored","details":{"errorType":"BAD_REQUEST","errorMessage":"Cannot tag Efs file","stepType":"TAG","stepName":"successful_tag_step"},"workflowId":"w-1234abcd5678efghi","executionId":"81234abcd-1234-efgh-5678-ijklmnopqr90","transferDetails":{"serverId":"s-1234abcd5678efghi","username":"lhr","sessionId":"1234567890abcdef0"}}
Symlinks	lhr.eb49cf7b8651e6d5 CREATE_SYMLINK LinkPath=/fs-12345678/lhr/pqr.jpg TargetPath=abc.jpg

Action	Corresponding logs within Amazon CloudWatch Logs
Uploads	lhr.33a8fb495ffb383b OPEN Path=/bucket/user/123.jpg Mode=CREATE|TRUNCATE|WRITE lhr.33a8fb495ffb383b CLOSE Path=/bucket/user/123.jpg BytesIn=3618546
Workflows	{"type":"ExecutionStarted","details":{"input":{"initialFileLocation":{"backingStore":"EFS","filesystemId":"fs-12345678","path":"/lhr/regex.py"}}},"workflowId":"w-1111aaaa2222bbbb3","executionId":"1234abcd-1234-efgh-5678-ijklmnopqr90","transferDetails":{"serverId":"s-zzzz1111aaaa22223","username":"lhr","sessionId":"1234567890abcdef0"}} {"type":"StepStarted","details":{"input":{"fileLocation":{"backingStore":"EFS","filesystemId":"fs-12345678","path":"/lhr/regex.py"}},"stepType":"CUSTOM","stepName":"efs-s3_copy_2"},"workflowId":"w-9283e49d33297c3f7","executionId":"1234abcd-1234-efgh-5678-ijklmnopqr90","transferDetails":{"serverId":"s-18ca49dce5d842e0b","username":"lhr","sessionId":"1234567890abcdef0"}}

Example CloudWatch log entries

This topic presents example log entries.

Topics

- Example transfer sessions log entries

- Example log entries for SFTP connectors
- Example log entries for VPC Lattice connectors
- Example log entries for Key exchange algorithm failures

Example transfer sessions log entries

In this example, an SFTP user connects to a Transfer Family server, uploads a file, then disconnects from the session.

The following log entry reflects an SFTP user connecting to a Transfer Family server.

```
{
   "role": "arn:aws:iam::500655546075:role/transfer-s3",
   "activity-type": "CONNECTED",
   "ciphers": "chacha20-poly1305@openssh.com,chacha20-poly1305@openssh.com",
   "client": "SSH-2.0-OpenSSH_7.4",
   "source-ip": "52.94.133.133",
   "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
   "home-dir": "/test/log-me",
   "ssh-public-key":
 "AAAAC3NzaC1lZDI1NTE5AAAAIA9OY0qV6XYVHaaOiWAcj2spDJVbgjrqDPY4pxd6GnHl",
   "ssh-public-key-fingerprint": "SHA256:BY3gNMHwTfjd4n2VuT4pTyLOk82zWZj4KEYEu7y4r/0",
   "ssh-public-key-type": "ssh-ed25519",
   "user": "log-me",
   "kex": "ecdh-sha2-nistp256",
   "session-id": "9ca9a0e1cec6ad9d"
}
```

The following log entry reflects the SFTP user uploading a file into their Amazon S3 bucket.

```
{
   "mode": "CREATE|TRUNCATE|WRITE",
   "path": "/test/log-me/config-file",
   "activity-type": "OPEN",
   "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
   "session-id": "9ca9a0e1cec6ad9d"
}
```

The following log entries reflect the SFTP user disconnecting from their SFTP session. First, the client closes the connection to the bucket, and then the client disconnects the SFTP session.

```
{
   "path": "/test/log-me/config-file",
   "activity-type": "CLOSE",
   "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
   "bytes-in": "121",
   "session-id": "9ca9a0e1cec6ad9d"
}

{
   "activity-type": "DISCONNECTED",
   "resource-arn": "arn:aws:transfer:us-east-1:500655546075:server/
s-3fe215d89f074ed2a",
   "session-id": "9ca9a0e1cec6ad9d"
}
```

Note

The available activity types are as follows: AUTH_FAILURE, CONNECTED, CONNECTION_REJECTED, DISCONNECTED, ERROR, EXIT_REASON, CLOSE, CREATE_SYMLINK, DELETE, MKDIR, OPEN, PARTIAL_CLOSE, RENAME, RMDIR, SETSTAT, TLS_RESUME_FAILURE.

Example log entries for SFTP connectors

This section contains example logs for both a successful and an unsuccessful transfer. Logs are generated to a log group named `/aws/transfer/`*`connector-id`*, where *`connector-id`* is the identifier for your SFTP connector. Log entries for SFTP connectors are generated when you run either a `StartFileTransfer` or `StartDirectoryListing` command.

This log entry is for a transfer that completed successfully.

```
{
    "operation": "RETRIEVE",
    "timestamp": "2023-10-25T16:33:27.373720Z",
    "connector-id": "connector-id",
    "transfer-id": "transfer-id",
```

```
    "file-transfer-id": "transfer-id/file-transfer-id",
    "url": "sftp://192.0.2.0",
    "file-path": "/remotebucket/remotefilepath",
    "status-code": "COMPLETED",
    "start-time": "2023-10-25T16:33:26.945481Z",
    "end-time": "2023-10-25T16:33:27.159823Z",
    "account-id": "480351544584",
    "connector-arn": "arn:aws:transfer:us-east-1:account-id:connector/connector-id",
    "local-directory-path": "/connectors-localbucket",
    "bytes": 514,
    "egress-type": "SERVICE_MANAGED"
}
```

This log entry is for a transfer that timed out, and thus was not completed successfully.

```
{
    "operation": "RETRIEVE",
    "timestamp": "2023-10-25T22:33:47.625703Z",
    "connector-id": "connector-id",
    "transfer-id": "transfer-id",
    "file-transfer-id": "transfer-id/file-transfer-id",
    "url": "sftp://192.0.2.0",
    "file-path": "/remotebucket/remotefilepath",
    "status-code": "FAILED",
    "failure-code": "TIMEOUT_ERROR",
    "failure-message": "Transfer request timeout.",
    "account-id": "480351544584",
    "connector-arn": "arn:aws:transfer:us-east-1:account-id:connector/connector-id",
    "local-directory-path": "/connectors-localbucket",
    "egress-type": "SERVICE_MANAGED"
}
```

This log entry is for a SEND operation that succeeds.

```
{
    "operation": "SEND",
    "timestamp": "2024-04-24T18:16:12.513207284Z",
    "connector-id": "connector-id",
    "transfer-id": "transfer-id",
    "file-transfer-id": "transfer-id/file-transfer-id",
    "url": "sftp://server-id.server.transfer.us-east-1.amazonaws.com",
    "file-path": "/amzn-s3-demo-bucket/my-test-folder/connector-metrics-us-
east-1-2024-01-02.csv",
```

```
    "status-code": "COMPLETED",
    "start-time": "2024-04-24T18:16:12.295235884Z",
    "end-time": "2024-04-24T18:16:12.461840732Z",
    "account-id": "255443218509",
    "connector-arn": "arn:aws:transfer:us-east-1:account-id:connector/connector-id",
    "bytes": 275,
    "egress-type": "SERVICE_MANAGED"
}
```

Descriptions for some key fields in the previous log examples.

- `timestamp` represents when the log is added to CloudWatch. `start-time` and `end-time` correspond to when the connector actually starts and finishes a transfer.
- `transfer-id` is a unique identifier that is assigned for each `start-file-transfer` request. If the user passes multiple file paths in a single `start-file-transfer` API operation, all the files share the same `transfer-id`.
- `file-transfer-id` is a unique value generated for each file transferred. Note that the initial portion of the `file-transfer-id` is the same as `transfer-id`.

Example log entries for VPC Lattice connectors

This section contains example logs for VPC Lattice connectors. For VPC Lattice connectors, logs include additional fields that provide information about the connector configuration and network setup.

This log entry is for a VPC Lattice connector SEND operation that completed successfully.

```
{
  "operation": "SEND",
  "timestamp": "2025-09-05T14:20:19.577192454Z",
  "connector-id": "connector-id",
  "transfer-id": "transfer-id",
  "file-transfer-id": "transfer-id/file-transfer-id",
  "file-path": ""/amzn-s3-demo-bucket/my-test-folder/connector-vpc-lattice-us-
east-1-2025-03-22.csv"",
  "status-code": "COMPLETED",
  "start-time": "2025-09-05T14:20:19.434072509Z",
  "end-time": "2025-09-05T14:20:19.481453346Z",
  "account-id": "account-id",
  "connector-arn": "arn:aws:transfer:us-east-1:account-id:connector/connector-id",
```

```
  "remote-directory-path": "/test-bucket/test-folder/",
  "bytes": 262,
  "egress-type": "VPC_LATTICE",
  "vpc-lattice-resource-configuration-arn": "arn:aws:vpc-lattice:us-east-1:account-
id:resourceconfiguration/resource-configuration-arn-id,
  "vpc-lattice-port-number": 22
}
```

VPC Lattice connector logs include the following additional fields:

- `egress-type` - Type of egress configuration for the connector
- `vpc-lattice-resource-configuration-arn` - ARN of the VPC Lattice Resource Configuration that defines the target SFTP server location
- `vpc-lattice-port-number` - Port number for connecting to the SFTP server through VPC Lattice

Example log entries for Key exchange algorithm failures

This section contains example logs where the Key exchange algorithm (KEX) failed. These are examples from the **ERRORS** log stream for structured logs.

This log entry is an example where there is a host key type error.

```
{
    "activity-type": "KEX_FAILURE",
    "source-ip": "999.999.999.999",
    "resource-arn": "arn:aws:transfer:us-east-1:999999999999:server/
s-999999999999999999",
    "message": "no matching host key type found",
    "kex": "ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ecdsa-sha2-
nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-
nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-rsa,ssh-dss"
}
```

This log entry is an example where there is a KEX mismatch.

```
{
    "activity-type": "KEX_FAILURE",
    "source-ip": "999.999.999.999",
    "resource-arn": "arn:aws:transfer:us-east-1:999999999999:server/
s-999999999999999999",
```

```
    "message": "no matching key exchange method found",
    "kex": "diffie-hellman-group1-sha1,diffie-hellman-group14-sha1,diffie-hellman-
group14-sha256"
}
```

Using CloudWatch metrics for Transfer Family servers

> **Note**
>
> You can also get metrics for Transfer Family from within the Transfer Family console itself. For details, see Monitoring usage in the console

You can get information about your server using CloudWatch metrics. A *metric* represents a time-ordered set of data points that are published to CloudWatch. When using metrics, you must specify the Transfer Family namespace, metric name, and dimension. For more information about metrics, see Metrics in the *Amazon CloudWatch User Guide*.

The following table describes the CloudWatch metrics for Transfer Family.

Namespace	Metric	Description
`AWS/Transfer`	`BytesIn`	The total number of bytes transferred into the server. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing **Units**: Count **Period**: 5 minutes

Namespace	Metric	Description
	`BytesOut`	The total number of bytes transferred out of the server. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer calls `StartFile Transfer` from their AS2 connector and emitted as soon as the outbound message has finished processin g. **Units**: Count **Period**: 5 minutes
	`FilesIn`	The total number of files transferred into the server. For servers using the AS2 protocol, this metric represent s the number of messages received. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing. **Units**: Count **Period**: 5 minutes

Namespace	Metric	Description
	`FilesOut`	The total number of files transferred out of the server. **Reporting criteria**: • For SFTP/FTP/FTPS: emitted every 5 minutes while a connection is established to the Transfer Family server. If no files or bytes are transferred in the period, "0" is emitted. • For AS2: when the customer calls `StartFile Transfer` from their AS2 connector and emitted as soon as the outbound message has finished processin g. **Units**: Count **Period**: 5 minutes
	`InboundMe ssage`	The total number of AS2 messages successfully received from a trading partner. **Reporting criteria**: When the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing successfully **Units**: Count **Period**: 5 minutes

Namespace	Metric	Description
	`InboundFailedMessage`	The total number of AS2 messages that were unsuccessfully received from a trading partner. That is, a trading partner sent a message, but the Transfer Family server was not able to successfully process it. **Reporting criteria**: When the customer receives a message in their AS2 server and emitted as soon as the inbound message has finished processing unsuccessfully **Units**: Count **Period**: 5 minutes
	`OnUploadExecutionsStarted`	The total number of workflow executions started on the server. **Reporting criteria**: Triggered every time an execution starts **Units**: Count **Period**: 1 minute
	`OnUploadExecutionsSuccess`	The total number of successful workflow executions on the server. **Reporting criteria**: Triggered every time an execution successfully finishes **Units**: Count **Period**: 1 minute

Namespace	Metric	Description
	`OnUploadExecutionsFailed`	The total number of unsuccessful workflow executions on the server. **Reporting criteria**: Triggered every time an execution does not successfully finish **Units**: Count **Period**: 1 minute
	`OutboundMessage`	The total number of AS2 messages successfully sent to a a trading partner. **Reporting criteria**: When the customer calls `StartFileTransfer` from their AS2 connector and emitted as soon as the outbound message has finished processing successfully **Units**: Count **Period**: 5 minutes
	`OutboundFailedMessage`	The total number of AS2 messages that were unsuccessfully sent to a trading partner. **Reporting criteria**: When the customer calls `StartFileTransfer` from their AS2 connector and emitted as soon as the outbound message has finished processing unsuccessfully **Units**: Count **Period**: 5 minutes

Transfer Family dimensions

A *dimension* is a name/value pair that is part of the identity of a metric. For more information about dimensions, see Dimensions in the *Amazon CloudWatch User Guide*.

The following table describes the CloudWatch dimensions for Transfer Family.

Dimension	Description
`ServerId`	The unique ID of the server.
`ConnectorId`	The unique ID of the connector. Used for AS2, for `OutboundMessage` and `OutboundFailedMessage`

Using AWS User Notifications with AWS Transfer Family

To get notified about AWS Transfer Family events, you can use AWS User Notifications to set up various delivery channels. When an event matches a rule that you specify, you receive a notification.

You can receive notifications for events through multiple channels, including email, Amazon Q Developer in chat applications chat notifications, or AWS Console Mobile Application push notifications. You can also see notifications in the Console Notifications Center. User Notifications supports aggregation, which can reduce the number of notifications that you receive during specific events.

For more information, see the Customize file delivery notifications using AWS Transfer Family managed workflows blog post, and What is AWS User Notifications? in the *AWS User Notifications User Guide*.

Using queries to filter log entries

You can use CloudWatch queries to filter and identify log entries for Transfer Family. This section contains some examples.

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. You can create queries or rules.

   - To create a **Logs Insights** query, choose **Logs Insights** from the left navigation panel, and then enter the details for your query.
   - To create a **Contributor Insights** rule, choose Insights > Contributor Insights from the left navigation panel and then enter the details for your rule.

3. Run the query or rule that you created.

View the top authentication failure contributors

In your structured logs, an authentication failure log entry looks similar to the following:

```
{
  "method":"password",
  "activity-type":"AUTH_FAILURE",
  "source-ip":"999.999.999.999",
  "resource-arn":"arn:aws:transfer:us-east-1:999999999999:server/s-0123456789abcdef",
  "message":"Invalid user name or password",
  "user":"exampleUser"
}
```

Run the following query to view the top contributors to authentication failures.

```
filter @logStream = 'ERRORS'
| filter `activity-type` = 'AUTH_FAILURE'
| stats count() as AuthFailures by user, method
| sort by AuthFailures desc
| limit 10
```

Rather than using **CloudWatch Logs Insights**, you can create a **CloudWatch Contributors Insights** rule to view authentication failures. Create a rule similar to the following.

```
{
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.activity-type",
                "In": [
                    "AUTH_FAILURE"
```

```
                ]
            }
        ],
        "Keys": [
            "$.user"
        ]
    },
    "LogFormat": "JSON",
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupARNs": [
        "arn:aws:logs:us-east-1:999999999999:log-group:/customer/structured_logs"
    ]
}
```

View log entries where a file was opened

In your structured logs, a file read log entry looks similar to the following:

```
{
  "mode":"READ",
  "path":"/fs-0df669c89d9bf7f45/avtester/example",
  "activity-type":"OPEN",
  "resource-arn":"arn:aws:transfer:us-east-1:999999999999:server/s-0123456789abcdef",
  "session-id":"0049cd844c7536c06a89"
}
```

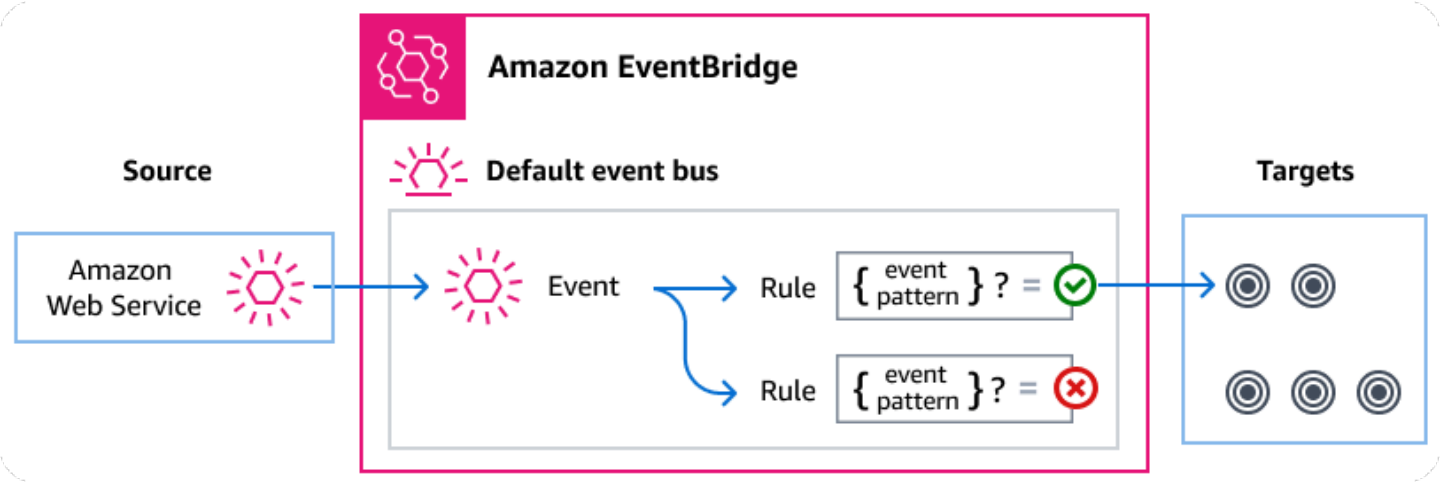
Run the following query to view log entries that indicate a file was opened.

```
filter `activity-type` = 'OPEN'
| display @timestamp, @logStream, `session-id`, mode, path
```

Managing Transfer Family events using Amazon EventBridge

Amazon EventBridge is a serverless service that uses events to connect application components together, which can make 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, Transfer Family generates and sends events to the EventBridge default event bus. Note that the default event bus 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*. You specify rules for the event bus that evaluates events as they arrive. Each rule checks whether an event matches the rule's *event pattern*. If the event matches, the event bus sends the event to one or more specified targets.



Topics

- Transfer Family events
- Sending Transfer Family events by using EventBridge rules
- Amazon EventBridge permissions
- Additional EventBridge resources
- Transfer Family events detail reference

Transfer Family events

Transfer Family automatically sends events to the default EventBridge event bus. You can create rules on the event bus where each rule includes an event pattern and one or more targets.

Events that match a rule's event pattern are delivered to the specified targets on either a *best effort* or *durable* basis (note that some events might be delivered out of order). These delivery levels are described in Delivery level for AWS service events in the *Amazon EventBridge Events Reference*.

- Server-level events for SFTP, FTPS and FTP servers are delivered on a best effort basis.
- The SFTP connector events are delivered on a durable basis.
- The AS2 events are delivered on a durable basis.

The following events are generated by Transfer Family. For more information, see EventBridge events in the *Amazon EventBridge User Guide*.

SFTP, FTPS, and FTP server events

The following tables list the events for SFTP, FTPS, and FTP servers, organized by event type.

File upload and download events

Event detail type	Description
FTP Server File Download Completed	A file has been downloaded successfully for the FTP protocol.
FTP Server File Download Failed	An attempt to download a file has failed for the FTP protocol.
FTP Server File Upload Completed	A file has been uploaded successfully for the FTP protocol.
FTP Server File Upload Failed	An attempt to upload a file has failed for the FTP protocol.
FTPS Server File Download Completed	A file has been downloaded successfully for the FTPS protocol.

Event detail type	Description
FTPS Server File Download Failed	An attempt to download a file has failed for the FTPS protocol.
FTPS Server File Upload Completed	A file has been uploaded successfully for the FTPS protocol.
FTPS Server File Upload Failed	An attempt to upload a file has failed for the FTPS protocol.
SFTP Server File Download Completed	A file has been downloaded successfully for the SFTP protocol.
SFTP Server File Download Failed	An attempt to download a file has failed for the SFTP protocol.
SFTP Server File Upload Completed	A file has been uploaded successfully for the SFTP protocol.
SFTP Server File Upload Failed	An attempt to upload a file has failed for the SFTP protocol.

Other file operations events

Event detail type	Description
FTP Server Directory Create Completed	A directory has been created successfully for the FTP protocol.
FTP Server Directory Create Failed	An attempt to create a directory has failed for the FTP protocol.
FTP Server Directory Delete Completed	A directory has been deleted successfully for the FTP protocol.
FTP Server Directory Delete Failed	An attempt to delete a directory has failed for the FTP protocol.

Event detail type	Description
FTP Server File Delete Completed	A file has been deleted successfully for the FTP protocol.
FTP Server File Delete Failed	An attempt to delete a file has failed for the FTP protocol.
FTP Server File Rename Completed	A file has been renamed successfully for the FTP protocol.
FTP Server File Rename Failed	An attempt to rename a file has failed for the FTP protocol.
FTPS Server Directory Create Completed	A directory has been created successfully for the FTPS protocol.
FTPS Server Directory Create Failed	An attempt to create a directory has failed for the FTPS protocol.
FTPS Server Directory Delete Completed	A directory has been deleted successfully for the FTPS protocol.
FTPS Server Directory Delete Failed	An attempt to delete a directory has failed for the FTPS protocol.
FTPS Server File Delete Completed	A file has been deleted successfully for the FTPS protocol.
FTPS Server File Delete Failed	An attempt to delete a file has failed for the FTPS protocol.
FTPS Server File Rename Completed	A file has been renamed successfully for the FTPS protocol.
FTPS Server File Rename Failed	An attempt to rename a file has failed for the FTPS protocol.
SFTP Server Directory Create Completed	A directory has been created successfully for the SFTP protocol.
SFTP Server Directory Create Failed	An attempt to create a directory has failed for the SFTP protocol.

Event detail type	Description
SFTP Server Directory Delete Completed	A directory has been deleted successfully for the SFTP protocol.
SFTP Server Directory Delete Failed	An attempt to delete a directory has failed for the SFTP protocol.
SFTP Server File Delete Completed	A file has been deleted successfully for the SFTP protocol.
SFTP Server File Delete Failed	An attempt to delete a file has failed for the SFTP protocol.
SFTP Server File Rename Completed	A file has been renamed successfully for the SFTP protocol.
SFTP Server File Rename Failed	An attempt to rename a file has failed for the SFTP protocol.

SFTP connector events

Note

These events are delivered to EventBridge at a durable level, as described in Delivery level for AWS service events in the *Amazon EventBridge Events Reference*.

Event detail type	Description
SFTP Connector File Send Completed	A file transfer from a connector to a remote SFTP server has completed successfully.
SFTP Connector File Send Failed	A file transfer from a connector to a remote SFTP server has failed.
SFTP Connector File Retrieve Completed	A file transfer from a remote SFTP server to a connector has completed successfully.

Event detail type	Description
SFTP Connector File Retrieve Failed	A file transfer from a remote SFTP server to a connector has failed.
SFTP Connector Directory Listing Completed	A start file directory listing call that has completed successfu lly.
SFTP Connector Directory Listing Failed	A start file directory listing that has failed.
SFTP Connector Remote Move Completed	Files or directories have been moved or renamed successfully on the remote server.
SFTP Connector Remote Move Failed	Files or directories have been failed to be moved or renamed on the remote server.
SFTP Connector Remote Delete Completed	Files or directories have been successfully deleted on the remote server.
SFTP Connector Remote Delete Failed	Files or directories have failed to be deleted on the remote server.

AS2 events

Note

These events are delivered to EventBridge at a durable level, as described in Delivery level for AWS service events in the *Amazon EventBridge Events Reference*.

Event detail type	Description
AS2 Payload Receive Completed	The payload for an AS2 message has been received.
AS2 Payload Receive Failed	The payload for an AS2 message has not been received.

Event detail type	Description
AS2 Payload Send Completed	The payload for an AS2 message has been sent successfully.
AS2 Payload Send Failed	The payload for an AS2 message has failed to send.
AS2 MDN Receive Completed	The message disposition notification for an AS2 message has been received.
AS2 MDN Receive Failed	The message disposition notification for an AS2 message has not been received.
AS2 MDN Send Completed	The message disposition notification for an AS2 message has been sent successfully.
AS2 MDN Send Failed	The message disposition notification for an AS2 message has failed to send.

Sending Transfer Family events by using EventBridge rules

If you want the EventBridge default event bus to send Transfer Family events to a target, you must create a rule that contains an event pattern that matches the data in your desired Transfer Family events.

To capture AWS Transfer Family events in Amazon EventBridge

1. Sign in to the AWS Management Console and open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose **Rules**, then choose **Create rule**.
3. Enter a descriptive name for the rule, and optionally enter a description.
4. For **Rule type**, select **Rule with an event pattern** then choose **Next**.
5. In the **Event source** section, select **AWS events or EventBridge partner events**.
6. In the **Creation method** section, choose **Use pattern form**.
7. In the **Event pattern** section, provide the following information.
    a. For **Event source**, choose **AWS services**.
    b. For **AWS service**, choose **Transfer**.

c. For **Event type**, choose the Transfer Family event type that you want to trigger your rule.

   Depending upon your **Event type** selection, you may be presented with an **Event Type Specification 1** section.

d. If you see the **Event Type Specification 1** section, select the specific events you want to capture (or select **Any event** to capture all events for your selected event type).

e. (Optional) Use the **Event pattern** editor to specify filters for event details.

f. Choose **Next**.

8. Choose a target from the choices available in **Select targets**. Choose from the following available targets.

   - **AWS service**. Popular options are Lambda functions for serverless compute, Amazon SQS queues for message processing, Amazon SNS topics for notifications, and AWS Step Functions for orchestrating workflows.
   - **EventBridge API Destination**. If you want to send events to an HTTP endpoint outside of AWS, you can use an API Destination as your target.
   - **EventBridge event bus**. You can send events to another event bus, either in the same account and region or in a different account or region.

     For comprehensive instructions on creating event bus rules, see Creating rules that react to events in the *Amazon EventBridge User Guide*.

   For help in selecting a target, see Select targets in the *Amazon EventBridge User Guide*.

9. Configure any additional options for your target then choose **Next**.
10. (Optional) Add tags to your rule and choose **Next**.
11. In the **Review and create** screen, if everything looks good, choose **Create rule**.

Creating event patterns for Transfer Family events

When Transfer Family delivers an event to the default event bus, EventBridge uses the event pattern defined for each rule to determine if the event should be delivered to the rule's targets. An event pattern matches the data in the desired Transfer Family events. Each event pattern is a JSON object that contains the following:

- A `source` attribute that identifies the service sending the event. For Transfer Family events, the source is `aws.transfer`.
- (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 following event pattern matches against all events from Transfer Family:

```
{
  "source": ["aws.transfer"]
}
```

The following event pattern example matches all of the SFTP connector events:

```
{
  "source": ["aws.transfer"],
  "detail-type": ["SFTP Connector File Send Completed", "SFTP Connector File Retrieve
 Completed",
                  "SFTP Connector File Retrieve Failed", "SFTP Connector File Send
 Failed"]
}
```

The following event pattern example matches all Transfer Family failed events:

```
{
  "source": ["aws.transfer"],
  "detail-type": [{"wildcard", "*Failed"}]
}
```

The following event pattern example matches successful SFTP downloads for user *username*:

```
{
  "source": ["aws.transfer"],
  "detail-type": ["SFTP Server File Download Completed"],
  "detail": {
    "username": [username]
  }
}
```

For more information on writing event patterns, see Event patterns in the *EventBridge User Guide*.

Testing event patterns for Transfer Family events in EventBridge

You can use the EventBridge Sandbox to quickly define and test an event pattern, without having to complete the broader process of creating or editing a rule. Using the Sandbox, you can define an event pattern and use a sample event to confirm that the pattern matches the desired events. EventBridge gives you the option of creating a new rule by using that event pattern directly from the sandbox.

For more information, see Testing an event pattern using the EventBridge Sandbox in the *EventBridge User Guide*.

Amazon EventBridge permissions

Transfer Family doesn't require any additional permissions to deliver events to Amazon EventBridge.

The targets that you specify might require specific permissions or configuration. For more details on using specific services for targets, see Amazon EventBridge targets in the *Amazon EventBridge User Guide*.

Additional EventBridge resources

Refer to the following topics in the *Amazon EventBridge User Guide* for more information on how to use EventBridge to process and manage events.

- For detailed information on how event buses work, see Amazon EventBridge event bus.
- For information on event structure, see Events.
- For information on constructing event patterns for EventBridge to use when matching events against rules, see Event patterns.
- For information on creating rules to specify which events EventBridge processes, see Rules.
- For information on how to specify what services or other destinations to which EventBridge sends matched events, see Targets.

Transfer Family events detail reference

All events from AWS services have a common set of fields containing metadata about the event. These metadata can include the AWS service that is 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 Event structure reference in the *Amazon EventBridge User Guide*.

In addition, each event has a `detail` field that contains data specific to that particular event. The following reference defines the detail fields for the various Transfer Family events.

When you use EventBridge to select and manage Transfer Family events, consider the following:

- The `source` field for all events from Transfer Family is set to `aws.transfer`.
- The `detail-type` field specifies the event type.

  For example, `FTP Server File Download Completed`.
- The `detail` field contains the data that is specific to that particular event.

For information on constructing event patterns that enable rules to match Transfer Family events, see Event patterns in the *Amazon EventBridge User Guide*.

For more information on events and how EventBridge processes them, see Amazon EventBridge events in the *Amazon EventBridge User Guide*.

Topics

- SFTP, FTPS, and FTP server events
- SFTP connector events
- AS2 events

SFTP, FTPS, and FTP server events

The following are the detail fields for SFTP, FTPS, and FTP server events:

- FTP Server Directory Create Completed
- FTP Server Directory Create Failed
- FTP Server Directory Delete Completed
- FTP Server Directory Delete Failed
- FTP Server File Delete Completed
- FTP Server File Delete Failed
- FTP Server File Download Completed

- FTP Server File Download Failed
- FTP Server File Rename Completed
- FTP Server File Rename Failed
- FTP Server File Upload Completed
- FTP Server File Upload Failed
- FTPS Server Directory Create Completed
- FTPS Server Directory Create Failed
- FTPS Server Directory Delete Completed
- FTPS Server Directory Delete Failed
- FTPS Server File Delete Completed
- FTPS Server File Delete Failed
- FTPS Server File Download Completed
- FTPS Server File Download Failed
- FTPS Server File Rename Completed
- FTPS Server File Rename Failed
- FTPS Server File Upload Completed
- FTPS Server File Upload Failed
- SFTP Server Directory Create Completed
- SFTP Server Directory Create Failed
- SFTP Server Directory Delete Completed
- SFTP Server Directory Delete Failed
- SFTP Server File Delete Completed
- SFTP Server File Delete Failed
- SFTP Server File Download Completed
- SFTP Server File Download Failed
- SFTP Server File Rename Completed
- SFTP Server File Rename Failed
- SFTP Server File Upload Completed
- SFTP Server File Upload Failed

The `source` and `detail-type` fields are included below because they contain specific values for Transfer Family events. For definitions of the other metadata fields that are included in all events, see Event structure reference in the *Amazon EventBridge User Guide*.

```
{
  . . .,
  "detail-type": "string",
  "source": "aws.transfer",
  . . .,
  "detail": {
    "failure-code" : "string",
    "status-code" : "string",
    "protocol" : "string",
    "bytes" : "number",
    "client-ip" : "string",
    "failure-message" : "string",
    "end-timestamp" : "string",
    "etag" : "string",
    "file-path" : "string",
    "original-file-path" : "string",
    "renamed-file-path" : "string",
    "directory-path" : "string",
    "server-id" : "string",
    "username" : "string",
    "session-id" : "string",
    "start-timestamp" : "string"
  }
}
```

`detail-type`

Identifies the type of event.

For this event, the value is one of the SFTP, FTPS, or FTP server event names listed previously.

`source`

Identifies the service that generated the event. For Transfer Family events, this value is `aws.transfer`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

For this event, the data includes the following:

`failure-code`

Category for why the transfer failed. Values: `PARTIAL_UPLOAD | PARTIAL_DOWNLOAD | UNKNOWN_ERROR`

`status-code`

Whether the transfer is successful. Values: `COMPLETED | FAILED`.

`protocol`

The protocol used for the transfer. Values: `SFTP | FTPS | FTP`

`bytes`

The number of bytes transferred.

`client-ip`

The IP address for the client involved in the transfer

`failure-message`

For failed transfers, the details for why the transfer failed.

`end-timestamp`

For successful transfers, the timestamp for when the file finishes being processed.

`etag`

The entity tag (only used for Amazon S3 files).

`file-path`

The path to the file being transferred, deleted, or otherwise operated on.

`original-file-path`

For file rename events, the original path of the file before renaming.

`renamed-file-path`

For file rename events, the new path of the file after renaming.

`directory-path`

For directory create and delete events, the path of the directory.

`server-id`

The unique ID for the Transfer Family server.

`username`

The user that is performing the transfer.

`session-id`

The unique identifier for the transfer session. For SFTP, a session is one SSH connection. All SFTP channels that run over that connection share the same `session-id`.

`start-timestamp`

For successful transfers, the timestamp for when file processing begins.

Example SFTP Server File Download Failed example event

The following example shows an event where a download failed on an SFTP server (Amazon EFS is the storage being used).


```
{
    "version": "0",
    "id": "event-ID",
    "detail-type": "SFTP Server File Download Failed",
    "source": "aws.transfer",
    "account": "958412138249",
    "time": "2024-01-29T17:20:27Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:958412138249:server/s-1234abcd5678efghi"
    ],
    "detail": {
        "failure-code": "PARTIAL_DOWNLOAD",
        "status-code": "FAILED",
        "protocol": "SFTP",
        "bytes": 4100,
        "client-ip": "IP-address",
        "failure-message": "File was partially downloaded.",
        "end-timestamp": "2024-01-29T17:20:27.749749117Z",
        "file-path": "/fs-1234abcd5678efghi/user0/test-file",
        "server-id": "s-1234abcd5678efghi",
        "username": "test",
```

```
        "session-id": "session-ID",
        "start-timestamp": "2024-01-29T17:20:16.706282454Z"
    }
}
```

Example FTP Server File Upload Completed example event

The following example shows an event where an upload completed successfully on an FTP server (Amazon S3 is the storage being used).

```
{
    "version": "0",
    "id": "event-ID",
    "detail-type": "FTP Server File Upload Completed",
    "source": "aws.transfer",
    "account": "958412138249",
    "time": "2024-01-29T16:31:43Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:958412138249:server/s-1111aaaa2222bbbb3"
    ],
    "detail": {
        "status-code": "COMPLETED",
        "protocol": "FTP",
        "bytes": 1048576,
        "client-ip": "10.0.0.141",
        "end-timestamp": "2024-01-29T16:31:43.311866408Z",
        "etag": "b6d81b360a5672d80c27430f39153e2c",
        "file-path": "/amzn-s3-demo-bucket/test/1mb_file",
        "server-id": "s-1111aaaa2222bbbb3",
        "username": "test",
        "session-id": "event-ID",
        "start-timestamp": "2024-01-29T16:31:42.462088327Z"
    }
}
```

Example SFTP Server File Delete Completed example event

The following example shows an event where a file was successfully deleted on an SFTP server.

```
{
    "version": "0",
    "id": "event-ID",
```

```
    "detail-type": "SFTP Server File Delete Completed",
    "source": "aws.transfer",
    "account": "958412138249",
    "time": "2025-05-15T14:30:27Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:958412138249:server/s-1234abcd5678efghi"
    ],
    "detail": {
        "status-code": "COMPLETED",
        "protocol": "SFTP",
        "client-ip": "IP-address",
        "end-timestamp": "2025-05-15T14:30:27.749749117Z",
        "file-path": "/fs-1234abcd5678efghi/user0/test-file-to-delete.txt",
        "server-id": "s-1234abcd5678efghi",
        "username": "test",
        "session-id": "session-ID",
        "start-timestamp": "2025-05-15T14:30:26.706282454Z"
    }
}
```

Example SFTP Server File Rename Completed example event

The following example shows an event where a file was successfully renamed on an SFTP server.

```
{
    "version": "0",
    "id": "event-ID",
    "detail-type": "SFTP Server File Rename Completed",
    "source": "aws.transfer",
    "account": "958412138249",
    "time": "2025-05-15T15:45:12Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:958412138249:server/s-1234abcd5678efghi"
    ],
    "detail": {
        "status-code": "COMPLETED",
        "protocol": "SFTP",
        "client-ip": "IP-address",
        "end-timestamp": "2025-05-15T15:45:12.749749117Z",
        "original-file-path": "/fs-1234abcd5678efghi/user0/old-filename.txt",
        "renamed-file-path": "/fs-1234abcd5678efghi/user0/new-filename.txt",
        "server-id": "s-1234abcd5678efghi",
```

```
        "username": "test",
        "session-id": "session-ID",
        "start-timestamp": "2025-05-15T15:45:11.706282454Z"
    }
}
```

Example SFTP Server Directory Create Completed example event

The following example shows an event where a directory was successfully created on an SFTP server.

```
{
    "version": "0",
    "id": "event-ID",
    "detail-type": "SFTP Server Directory Create Completed",
    "source": "aws.transfer",
    "account": "958412138249",
    "time": "2025-05-15T16:20:05Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:958412138249:server/s-1234abcd5678efghi"
    ],
    "detail": {
        "status-code": "COMPLETED",
        "protocol": "SFTP",
        "client-ip": "IP-address",
        "end-timestamp": "2025-05-15T16:20:05.749749117Z",
        "directory-path": "/fs-1234abcd5678efghi/user0/new-directory",
        "server-id": "s-1234abcd5678efghi",
        "username": "test",
        "session-id": "session-ID",
        "start-timestamp": "2025-05-15T16:20:04.706282454Z"
    }
}
```

SFTP connector events

> **Note**
>
> These events are delivered to EventBridge at a durable level, as described in Delivery level for AWS service events in the *Amazon EventBridge Events Reference*.

The following are the detail fields for SFTP connector events:

- SFTP Connector File Send Completed
- SFTP Connector File Send Failed
- SFTP Connector File Retrieve Completed
- SFTP Connector File Retrieve Failed
- SFTP Connector Directory Listing Completed
- SFTP Connector Directory Listing Failed
- SFTP Connector Remote Move Completed
- SFTP Connector Remote Move Failed
- SFTP Connector Remote Delete Completed
- SFTP Connector Remote Delete Failed

The `source` and `detail-type` fields are included below because they contain specific values for Transfer Family events. For definitions of the other metadata fields that are included in all events, see Event structure reference in the *Amazon EventBridge User Guide*.

```
{
  . . .,
  "detail-type": "string",
  "source": "aws.transfer",
  . . .,
  "detail": {
    "operation" : "string",
    "max-items" : "number",
    "connector-id" : "string",
    "output-directory-path" : "string",
    "listing-id" : "string",
    "transfer-id" : "string",
    "file-transfer-id" : "string",
    "url" : "string",
    "file-path" : "string",
    "status-code" : "string",
    "failure-code" : "string",
    "failure-message" : "string",
    "start-timestamp" : "string",
    "end-timestamp" : "string",
    "local-directory-path" : "string",
```

```
    "remote-directory-path" : "string"
    "item-count" : "number"
    "truncated" : "boolean"
    "bytes" : "number",
    "egress-type" : "string",
    "vpc-lattice-resource-configuration-arn" : "string",
    "vpc-lattice-port-number" : "number",
    "local-file-location" : {
      "domain" : "string",
      "bucket" : "string",
      "key" : "string"
    },
    "output-file-location" : {
      "domain" : "string",
      "bucket" : "string",
      "key" : "string"
    }
  }
}
```

`detail-type`

Identifies the type of event.

For this event, the value is one of the SFTP connector event names listed previously.

`source`

Identifies the service that generated the event. For Transfer Family events, this value is `aws.transfer`.

`detail`

A JSON object that contains information about the event. The service that generates the event determines the content of this field.

For this event, the data includes the following:

`max-items`

The maximum number of directory/file names to return.

`operation`

Whether the `StartFileTransfer` request is sending or retrieving a file. Values: `SEND` | `RETRIEVE`.

`connector-id`

The unique identifier for the SFTP connector being used.

`output-directory-path`

The path (bucket and prefix) in Amazon S3 to store the results of the file/directory listing.

`listing-id`

A unique identifier for the `StartDirectoryListing` API operation. This identifier can be used to check CloudWatch logs to see the status of listing request.

`transfer-id`

The unique identifier for the transfer event (a `StartFileTransfer` request).

`file-transfer-id`

The unique identifier for the file being transferred.

`url`

The URL of the partner's AS2 or SFTP endpoint.

`file-path`

The location and file that is being sent or retrieved.

`status-code`

Whether the transfer is successful. Values: `FAILED | COMPLETED`.

`failure-code`

For failed transfers, the reason code for why the transfer failed. Values: `ASSUME_ROLE_ERROR | UNSUPPORTED_OPERATION_ERROR | CONNECTION_ERROR | READ_FILE_ERROR | WRITE_FILE_ERROR | TIMEOUT_ERROR | LIST_DIRECTORY_ERROR | SEND_FILE_NOT_FOUND | RETRIEVE_FILE_NOT_FOUND | DIRECTORY_NOT_FOUND | FILE_NOT_FOUND | PATH_NOT_FOUND | RENAME_ERROR | DELETE_ERROR | INTERNAL_ERROR | UNKNOWN_ERROR`.

`failure-message`

For failed transfers, the details for why the transfer failed.

`start-timestamp`

For successful transfers, the timestamp for when file processing begins.

`end-timestamp`

For successful transfers, the timestamp for when file processing completes.

`local-directory-path`

For `RETRIEVE` requests, the location in which to place the retrieved file.

`remote-directory-path`

For `SEND` requests, the file directory in which to place the file on the partner's SFTP server. This is the value for the `RemoteDirectoryPath` that the user passed to the `StartFileTransfer` request. You can specify a default directory on the partner's SFTP server. If so, this field is empty.

`item-count`

The number of items (directories and files) returned for the listing request.

`truncated`

Whether the list output contains all of the items contained in the remote directory or not.

`bytes`

The number of bytes being transferred. The value is 0 for failed transfers.

`egress-type`

Type of egress configuration for the connector. Values: `SERVICE_MANAGED` or `VPC_LATTICE`.

`vpc-lattice-resource-configuration-arn`

ARN of the VPC_LATTICE Resource Configuration that defines the target SFTP server location. This field is null for service-managed connectors.

`vpc-lattice-port-number`

Port number for connecting to the SFTP server through VPC_LATTICE.

`local-file-location`

This parameter contains the details of the location of the AWS storage file.

`domain`

The storage being used. Currently, the only value is `S3`.

`bucket`

The container for the object in Amazon S3.

`key`

The name assigned to the object in Amazon S3.

`output-file-location`

This parameter contains the details of the location for where to store the results of the directory listing in AWS storage.

`domain`

The storage being used. Currently, the only value is S3.

`bucket`

The container for the object in Amazon S3.

`key`

The name assigned to the object in Amazon S3.

Example SFTP Connector File Send Failed example event

The following example shows an event where an SFTP connector failed while trying to send a file to a remote SFTP server.

```
{
    "version": "0",
    "id": "event-ID",
    "detail-type": "SFTP Connector File Send Failed",
    "source": "aws.transfer",
    "account": "123456789012",
    "time": "2024-01-24T19:30:45Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:123456789012:connector/c-f1111aaaa2222bbbb3"
    ],
    "detail": {
        "operation": "SEND",
        "connector-id": "c-f1111aaaa2222bbbb3",
        "transfer-id": "transfer-ID",
        "file-transfer-id": "file-transfer-ID",
```

```
        "url": "sftp://s-21a23456789012a.server.transfer.us-east-1.amazonaws.com",
        "file-path": "/amzn-s3-demo-bucket/testfile.txt",
        "status-code": "FAILED",
        "failure-code": "CONNECTION_ERROR",
        "failure-message": "Unknown Host",
        "remote-directory-path": "",
        "bytes": 0,
        "start-timestamp": "2024-01-24T18:29:33.658729Z",
        "end-timestamp": "2024-01-24T18:29:33.993196Z",
        "local-file-location": {
            "domain": "S3",
            "bucket": "amzn-s3-demo-bucket",
            "key": "testfile.txt"
        }
    }
}
```

Example SFTP Connector File Retrieve Completed example event

The following example shows an event where an SFTP connector successfully retrieved a file sent from a remote SFTP server.

```
{
    "version": "0",
    "id": "event-ID",
    "detail-type": "SFTP Connector File Retrieve Completed",
    "source": "aws.transfer",
    "account": "123456789012",
    "time": "2024-01-24T18:28:08Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:123456789012:connector/c-f1111aaaa2222bbbb3"
    ],
    "detail": {
        "operation": "RETRIEVE",
        "connector-id": "c-fc68000012345aa18",
        "transfer-id": "file-transfer-ID",
        "file-transfer-id": "file-transfer-ID",
        "url": "sftp://s-21a23456789012a.server.transfer.us-east-1.amazonaws.com",
        "file-path": "testfile.txt",
        "status-code": "COMPLETED",
        "local-directory-path": "/amzn-s3-demo-bucket",
        "bytes": 63533,
```

```
        "start-timestamp": "2024-01-24T18:28:07.632388Z",
        "end-timestamp": "2024-01-24T18:28:07.774898Z",
        "local-file-location": {
            "domain": "S3",
            "bucket": "amzn-s3-demo-bucket",
            "key": "testfile.txt"
        }
    }
}
```

Example SFTP Connector Directory Listing Completed example event

The following example shows an event where a start directory listing call retrieved a listing file from a remote SFTP server.

```
{
    "version": "0",
    "id": "event-ID",
    "detail-type": "SFTP Connector Directory Listing Completed",
    "source": "aws.transfer",
    "account": "123456789012",
    "time": "2024-01-24T18:28:08Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:transfer:us-east-1:123456789012:connector/c-f1111aaaa2222bbbb3"
    ],
    "detail": {
        "max-items": 10000,
        "connector-id": "c-fc68000012345aa18",
        "output-directory-path": "/amzn-s3-demo-bucket/example/file-listing-output",
        "listing-id": "123456-23aa-7980-abc1-1a2b3c4d5e",
        "url": "sftp://s-21a23456789012a.server.transfer.us-east-1.amazonaws.com",

        "status-code": "COMPLETED",
        "remote-directory-path": "/home",
        "item-count": 10000,
        "truncated": true,
        "start-timestamp": "2024-01-24T18:28:07.632388Z",
        "end-timestamp": "2024-01-24T18:28:07.774898Z",
        "output-file-location": {
            "domain": "S3",
            "bucket": "amzn-s3-demo-bucket",
            "key": "c-fc1ab90fd0d047e7a-70987273-49nn-4006-bab1-1a7290cc412ba.json"
```

```
        }
    }
}
```

AS2 events

> **Note**
>
> These events are delivered to EventBridge at a durable level, as described in Delivery level for AWS service events in the *Amazon EventBridge Events Reference*.

The following are the detail fields for AS2 events:

- AS2 Payload Receive Completed
- AS2 Payload Receive Failed
- AS2 Payload Send Completed
- AS2 Payload Send Failed
- AS2 MDN Receive Completed
- AS2 MDN Receive Failed
- AS2 MDN Send Completed
- AS2 MDN Send Failed

The `source` and `detail-type` fields are included below because they contain specific values for Transfer Family events. For definitions of the other metadata fields that are included in all events, see Event structure reference in the *Amazon EventBridge User Guide*.

```
{
   . . .,
  "detail-type": "string",
  "source": "aws.transfer",
  . . .,
  "detail": {
    "s3-attributes" : {
      "file-bucket" : "string",
      "file-key" : "string",
      "json-bucket" : "string",
      "json-key" : "string",
```

```
      "mdn-bucket" : "string",
      "mdn-key" : "string"
      }
    "mdn-subject" : "string",
    "mdn-message-id" : "string",
    "disposition" : "string",
    "bytes" : "number",
    "as2-from" : "string",
    "as2-message-id" : "string",
    "as2-to" : "string",
    "connector-id" : "string",
    "client-ip" : "string",
    "agreement-id" : "string",
    "server-id" : "string",
    "requester-file-name" : "string",
    "message-subject" : "string",
    "start-timestamp" : "string",
    "end-timestamp" : "string",
    "status-code" : "string",
    "failure-code" : "string",
    "failure-message" : "string",
    "transfer-id" : "string"
  }
}
```

`detail-type`

Identifies the type of event.

For this event, the value is one of the AS2 events listed previously.

`source`

Identifies the service that generated the event. For Transfer Family events, this value is `aws.transfer`.

`detail`

A JSON object that contains information about the event. The service generating the event determines the content of this field.

`s3-attributes`

Identifies the Amazon S3 bucket and key for the file being transferred. For MDN events, it also identifies the bucket and key for the MDN file.

`file-bucket`

The container for the object in Amazon S3.

`file-key`

The name assigned to the object in Amazon S3.

`json-bucket`

For COMPLETED or FAILED transfers, the container for the JSON file.

`json-key`

For COMPLETED or FAILED transfers, the name assigned to the JSON file in Amazon S3.

`mdn-bucket`

For MDN events, the container for the MDN file.

`mdn-key`

For MDN events, the name assigned to the MDN file in Amazon S3.

`mdn-subject`

For MDN events, a text description for the message disposition.

`mdn-message-id`

For MDN events, a unique ID for the MDN message.

`disposition`

For MDN events, the category for the disposition.

`bytes`

The number of bytes in the message.

`as2-from`

The AS2 trading partner that is sending the message.

`as2-message-id`

A unique identifier for the AS2 message being transferred.

`as2-to`

The AS2 trading partner that is receiving the message.

`connector-id`

For AS2 messages being sent from a Transfer Family server to a trading partner, the unique identifier for the AS2 connector being used.

`client-ip`

For server events (transfers from a trading partner to a Transfer Family server), the IP address for the client involved in the transfer.

`agreement-id`

For server events, the unique identifier for the AS2 agreement.

`server-id`

For server events, a unique ID only for the Transfer Family server.

`requester-file-name`

For payload events, the original name for the file received during the transfer.

`message-subject`

A text description for the subject of the message.

`start-timestamp`

For successful transfers, the timestamp for when file processing begins.

`end-timestamp`

For successful transfers, the timestamp for when file processing completes.

`status-code`

The code that corresponds to the state of the AS2 message transfer process. Valid values: `COMPLETED | FAILED | PROCESSING`.

`failure-code`

For failed transfers, the category for why the transfer failed.

`failure-message`

For failed transfers, the details for why the transfer failed.

`transfer-id`

The unique identifier for the transfer event.

Example AS2 Payload Receive Completed example event


```
{
    "version": "0",
     "id": "event-ID",
    "detail-type": "AS2 Payload Receive Completed",
    "source": "aws.transfer",
    "account": "076722215406",
    "time": "2024-02-07T06:47:05Z",
    "region": "us-east-1",
    "resources": ["arn:aws:transfer:us-east-1:076722215406:connector/
c-1111aaaa2222bbbb3"],
    "detail": {
        "s3-attributes": {
            "file-key": "/inbound/processed/testAs2Message.dat",
            "file-bucket": "amzn-s3-demo-bucket"
        },
        "client-ip": "client-IP-address",
        "requester-file-name": "testAs2MessageVerifyFile.dat",
        "end-timestamp": "2024-02-07T06:47:06.040031Z",
        "as2-from": "as2-from-ID",
        "as2-message-id": "as2-message-ID",
        "message-subject": "Message from AS2 tests",
        "start-timestamp": "2024-02-07T06:47:05.410Z",
        "status-code": "PROCESSING",
        "bytes": 63,
        "as2-to": "as2-to-ID",
        "agreement-id": "a-1111aaaa2222bbbb3",
        "server-id": "s-1234abcd5678efghi"
    }
}
```


Example AS2 MDN Receive Failed example event


```
{
  "version": "0",
  "id": "event-ID",
  "detail-type": "AS2 MDN Receive Failed",
  "source": "aws.transfer",
  "account": "889901007463",
  "time": "2024-02-06T22:05:09Z",
  "region": "us-east-1",
  "resources": ["arn:aws:transfer:us-east-1:076722215406:server/s-1111aaaa2222bbbb3"],
```

```
  "detail": {
      "mdn-subject": "Your Requested MDN Response re: Test run from Id 123456789abcde
 to partner ijklmnop987654",
      "s3-attributes": {
          "json-bucket": "amzn-s3-demo-bucket1",
          "file-key": "/as2Integ/TestOutboundWrongCert.dat",
          "file-bucket": "amzn-s3-demo-bucket2",
          "json-key": "/as2Integ/failed/TestOutboundWrongCert.dat.json"
      },
      "mdn-message-id": "MDN-message-ID",
      "end-timestamp": "2024-02-06T22:05:09.479878Z",
      "as2-from": "PartnerA",
      "as2-message-id": "as2-message-ID",
      "connector-id": "c-1234abcd5678efghj",
      "message-subject": "Test run from Id 123456789abcde to partner ijklmnop987654",
      "start-timestamp": "2024-02-06T22:05:03Z",
      "failure-code": "VERIFICATION_FAILED_NO_MATCHING_KEY_FOUND",
      "status-code": "FAILED",
      "as2-to": "MyCompany",
      "failure-message": "No public certificate matching message signature could be
 found in profile: p-1234abcd5678efghj",
      "transfer-id": "transfer-ID"
  }
}
```

Security in AWS Transfer Family

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is 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:

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.

This documentation helps you understand how to apply the shared responsibility model when using AWS Transfer Family. The following topics show you how to configure AWS Transfer Family to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Transfer Family resources.

We offer a workshop that provides prescriptive guidance and a hands on lab on how you can build a scalable and secure file transfer architecture on AWS without needing to modify existing applications or manage server infrastructure. You can view the details for this workshop here.

Topics

- VPC connectivity security benefits
- Security policies for AWS Transfer Family servers
- Security policies for AWS Transfer Family SFTP connectors

- Using hybrid post-quantum key exchange with AWS Transfer Family
- Data protection and encryption
- Managing SSH and PGP keys in Transfer Family
- Identity and access management for AWS Transfer Family
- Compliance validation for AWS Transfer Family
- Resilience in AWS Transfer Family
- Create a private connection between a VPC and AWS Transfer Family APIs
- Infrastructure security in AWS Transfer Family
- Add a web application firewall
- Cross-service confused deputy prevention
- AWS managed policies for AWS Transfer Family
- Using service-linked roles for AWS Transfer Family

VPC connectivity security benefits

SFTP connectors with VPC egress type provide enhanced security benefits through Cross-VPC Resource Access:

- **Network isolation**: All traffic remains within your VPC environment, providing complete network isolation from the public internet for private endpoint connections.
- **Source IP control**: Remote SFTP servers only see IP addresses from your VPC CIDR range, giving you full control over the source IP addresses used for connections.
- **Private endpoint access**: Connect directly to SFTP servers in your VPC using private IP addresses, eliminating exposure to the public internet.
- **Hybrid connectivity**: Securely access on-premises SFTP servers through established VPN or Direct Connect connections without additional internet exposure.
- **VPC security controls**: Leverage existing VPC security groups, NACLs, and routing policies to control and monitor SFTP connector traffic.

VPC Lattice security model

VPC connectivity for SFTP connectors uses AWS VPC Lattice with service networks to provide secure multi-tenant access:

- **Confused deputy prevention**: Authentication and authorization checks ensure that connectors can only access the specific resources they are configured for, preventing unauthorized cross-tenant access.
- **IPv6-only service network**: Uses IPv6 addressing to avoid potential IP address conflicts and enhance security isolation.
- **Forward Access Session (FAS)**: Temporary credential handling eliminates the need for long-term credential storage or manual resource sharing.
- **Resource-level access control**: Each connector is associated with a specific Resource Configuration, ensuring granular access control to individual SFTP servers.

Security best practices for VPC connectivity

When using VPC egress type connectors, follow these security best practices:

- **Security groups**: Configure security groups to allow SFTP traffic (port 22) only between necessary resources. Restrict source and destination IP ranges to the minimum required.
- **Resource Gateway placement**: Deploy Resource Gateways in private subnets when possible, and ensure they span at least two Availability Zones for high availability.
- **Network monitoring**: Use VPC Flow Logs and Amazon CloudWatch to monitor network traffic patterns and detect anomalous activity.
- **Access logging**: Enable connector logging to track file transfer activities and maintain audit trails for compliance requirements.
- **Resource Configuration management**: Regularly review and update Resource Configurations to ensure they point to the correct SFTP servers and use appropriate network settings.

Security policies for AWS Transfer Family servers

Server security policies in AWS Transfer Family allow you to limit the set of cryptographic algorithms (message authentication codes (MACs), key exchanges (KEXs), cipher suites, content encryption ciphers, and hash algorithms) associated with your server.

AWS Transfer Family supports post-quantum security policies that use hybrid key exchange algorithms, combining traditional cryptographic methods with post-quantum algorithms to provide enhanced security against future quantum computing threats. For more information, see Using hybrid post-quantum key exchange with AWS Transfer Family.

For a list of supported cryptographic algorithms, see Cryptographic algorithms. For a list of supported key algorithms for use with server host keys and service-managed user keys, see Managing SSH and PGP keys in Transfer Family.

Note

Starting in 2025, all new AWS Transfer Family security policies include post-quantum cryptographic support using hybrid key exchange algorithms. For more information about post-quantum security, see Using hybrid post-quantum key exchange with AWS Transfer Family.

Note

We strongly recommend updating your servers to our latest security policy.

- `TransferSecurityPolicy-2024-01` is the default security policy attached to your server when creating a server using the console, API, or CLI.
- If you create a Transfer Family server using CloudFormation and accept the default security policy, the server is assigned `TransferSecurityPolicy-2018-11`.

If you are concerned about client compatibility, please affirmatively state which security policy you wish to use when creating or updating a server rather than using the default policy, which is subject to change. To change the security policy for a server, see Edit the security policy.

Note

The earlier post quantum policies (**TransferSecurityPolicy-PQ-SSH-Experimental-2023-04** and **TransferSecurityPolicy-PQ-SSH-FIPS-Experimental-2023-04**) are deprecated. We recommend that you use the new policies instead.

For more information on security in Transfer Family, see the following blog posts:

- Six tips to improve the security of your AWS Transfer Family server

- How Transfer Family can help you build a secure, compliant managed file transfer solution

Topics

- Cryptographic algorithms
- Security policy details

Cryptographic algorithms

For host keys, we support the following algorithms:

- `rsa-sha2-256`
- `rsa-sha2-512`
- `ecdsa-sha2-nistp256`
- `ecdsa-sha2-nistp384`
- `ecdsa-sha2-nistp521`
- `ssh-ed25519`

Additionally, the following security policies allow `ssh-rsa`:

- TransferSecurityPolicy-2018-11
- TransferSecurityPolicy-2020-06
- TransferSecurityPolicy-FIPS-2020-06
- TransferSecurityPolicy-FIPS-2023-05
- TransferSecurityPolicy-FIPS-2024-01

> **Note**
>
> It is important to understand the distinction between the RSA key type—which is always `ssh-rsa`—and the RSA host key algorithm, which can be any of the supported algorithms.

The following is a list of supported cryptographic algorithms for each security policy.

Note

In the following table and policies, note the following use of algorithm types.

- SFTP servers only use algorithms in the **SshCiphers**, **SshKexs**, and **SshMacs** sections.
- FTPS servers only use algorithms in the **TlsCiphers** section.
- FTP servers, since they don't use encryption, do not use any of these algorithms.
- AS2 servers only use algorithms in the **ContentEncryptionCiphers** and **HashAlgorithms** sections. These sections define algorithms used for encrypting and signing file content.
- The FIPS-2024-05 and FIPS-2024-01 security policies are identical, except that FIPS-2024-05 doesn't support the `ssh-rsa` algorithm.
- Transfer Family has introduced new restricted policies that closely parallel existing policies:
  - The TransferSecurityPolicy-Restricted-2018-11 and TransferSecurityPolicy-2018-11 security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.
  - The TransferSecurityPolicy-Restricted-2020-06 and TransferSecurityPolicy-2020-06 security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.

*In the following table, the `chacha20-poly1305@openssh.com` cipher is included in the non-restricted policy only,

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2022-03	TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
SshCiphers												
aes128-ctr	♦	♦	♦	♦	♦	♦				♦	♦	♦
aes128-gcm@openssh.com	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
aes192-ctr	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
aes256-ctr	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
aes256-gcm@openssh.com	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
chacha20-poly1305@openssh.com										♦*		♦*

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditConformant-2025-02	TransferSecurityPolicy-AS2Restricted-2025-07	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2022-03	TransferSecurityPolicy-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11	TransferSecurityPolicy-Restricted-2018-11
SshKexs													
mlkem768x25519-sha256	♦			♦									
mlkem768nistp256-sha256	♦			♦									
mlkem1024nistp384-sha384	♦			♦									
curve25519-sha256			♦	♦	♦		♦		♦			♦	
curve25519-sha256@libssh.org			♦	♦	♦		♦		♦			♦	

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2020-06	TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
diffie-hellman-group14-sha1												♦
diffie-hellman-group14-sha256										♦	♦	♦
diffie-hellman-group16-sha512	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted-2025-07	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2020-06	TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
diffie-hellman-group18-sha512	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
diffie-hellman-group-exchange-sha256	♦	♦	♦	♦	♦		♦	♦	♦	♦	♦	♦
ecdh-sha2-nistp256	♦	♦		♦	♦	♦				♦	♦	♦

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted-2025-07	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2022-03	TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
ecdh-sha2-nistp384	♦	♦		♦	♦	♦				♦	♦	♦
ecdh-sha2-nistp521	♦	♦		♦	♦	♦				♦	♦	♦
SshMacs												
hmac-sha1												♦
hmac-sha1-etm@openssh.com												♦
hmac-sha2-256									♦	♦	♦	♦

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2020-06	TransferSecurityPolicy-2022-03	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-2018-11	TransferSecurityPolicy-Restricted-2018-11
hmac-sha2-256-etm@openssh.com	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
hmac-sha2-512										♦	♦	♦	♦	♦
hmac-sha2-512-etm@openssh.com	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
umac-128-etm@openssh.com										♦			♦	♦
umac-128@openssh.com										♦			♦	♦

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2022-03	TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
umac-64-etm@openssh.com												♦
umac-64@openssh.com												♦
ContentEncryptionCiphers												
aes256-cbc	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
aes192-cbc	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
aes128-cbc	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
3des-cbc	♦	♦	♦		♦	♦	♦	♦	♦	♦	♦	♦
HashAlgorithms												
sha256	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2022-03	TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
sha384	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
sha512	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
sha1	♦	♦	♦		♦	♦	♦	♦	♦	♦	♦	♦
TlsCiphers												
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2020-06	TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦

Security policy	TransferSecurityPolicy-2025-03	TransferSecurityPolicy-FIPS-2025-03	TransferSecurityPolicy-SshAuditCompliant-2025-02	TransferSecurityPolicy-AS2Restricted-2025-07	TransferSecurityPolicy-2024-01	TransferSecurityPolicy-FIPS-2024-01/ TransferSecurityPolicy-FIPS-2024-05	TransferSecurityPolicy-2023-05	TransferSecurityPolicy-FIPS-2023-05	TransferSecurityPolicy-2022-03	TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06	TransferSecurityPolicy-FIPS-2020-06	TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TLS_RSA_WITH_AES_128_CBC_SHA256												♦
TLS_RSA_WITH_AES_256_CBC_SHA256												♦

Security policy details

The following sections contain the JSON representation of each security policy.

TransferSecurityPolicy-2025-03

The following shows the TransferSecurityPolicy-2025-03 security policy.

```
{
    "SecurityPolicy": {
```

```
        "Fips": false,
        "SecurityPolicyName": "TransferSecurityPolicy-2025-03",
        "SshCiphers": [
            "aes256-gcm@openssh.com",
            "aes128-gcm@openssh.com",
            "aes128-ctr",
            "aes256-ctr",
            "aes192-ctr"
        ],
        "SshKexs": [
            "mlkem768x25519-sha256",
            "mlkem768nistp256-sha256",
            "mlkem1024nistp384-sha384",
            "ecdh-sha2-nistp256",
            "ecdh-sha2-nistp384",
            "ecdh-sha2-nistp521",
            "curve25519-sha256",
            "curve25519-sha256@libssh.org",
            "diffie-hellman-group16-sha512",
            "diffie-hellman-group18-sha512",
            "diffie-hellman-group-exchange-sha256"
        ],
        "SshMacs": [
            "hmac-sha2-256-etm@openssh.com",
            "hmac-sha2-512-etm@openssh.com"
        ],
        "ContentEncryptionCiphers": [
            "aes256-cbc",
            "aes192-cbc",
            "aes128-cbc",
            "3des-cbc"
        ],
        "HashAlgorithms": [
            "sha256",
            "sha384",
            "sha512",
            "sha1"
        ],
        "TlsCiphers": [
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
```

```
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
        ],
        "Type": "SERVER",
        "Protocols": [
           "SFTP",
           "FTPS"
        ]
    }
}
```

TransferSecurityPolicy-FIPS-2025-03

The following shows the TransferSecurityPolicy-FIPS-2025-03 security policy.

```
{
    "SecurityPolicy": {
        "Fips": true,
        "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2025-03",
        "SshCiphers": [
            "aes256-gcm@openssh.com",
            "aes128-gcm@openssh.com",
            "aes256-ctr",
            "aes192-ctr",
            "aes128-ctr"
        ],
        "SshKexs": [
            "mlkem768x25519-sha256",
            "mlkem768nistp256-sha256",
            "mlkem1024nistp384-sha384",
            "ecdh-sha2-nistp256",
            "ecdh-sha2-nistp384",
            "ecdh-sha2-nistp521",
            "diffie-hellman-group-exchange-sha256",
            "diffie-hellman-group16-sha512",
            "diffie-hellman-group18-sha512"
        ],
        "SshMacs": [
            "hmac-sha2-512-etm@openssh.com",
            "hmac-sha2-256-etm@openssh.com"
        ],
        "ContentEncryptionCiphers": [
```

```
            "aes256-cbc",
            "aes192-cbc",
            "aes128-cbc",
            "3des-cbc"
        ],
        "HashAlgorithms": [
            "sha256",
            "sha384",
            "sha512",
            "sha1"
        ],
        "TlsCiphers": [
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
        ],
        "Type": "SERVER",
        "Protocols": [
           "SFTP",
           "FTPS"
        ]
    }
}
```

TransferSecurityPolicy-AS2Restricted-2025-07

This security policy is designed for AS2 file transfers that require enhanced security by excluding legacy cryptographic algorithms. It supports modern AES encryption and SHA-2 hash algorithms while removing support for weaker algorithms like 3DES and SHA-1.

> **Note**
>
> This security policy is identical to TransferSecurityPolicy-2025-03, except that it does not support 3DES (in ContentEncryptionCiphers) and does not support SHA1 (in HashAlgorithms). It includes all algorithms from 2025-03, including post-quantum cryptographic algorithms (mlkem* KEXs).

```
{
    "SecurityPolicy": {
        "Fips": false,
        "SecurityPolicyName": "TransferSecurityPolicy-AS2Restricted-2025-07",
        "SshCiphers": [
            "aes256-gcm@openssh.com",
            "aes128-gcm@openssh.com",
            "aes128-ctr",
            "aes256-ctr",
            "aes192-ctr"
        ],
        "SshKexs": [
            "mlkem768x25519-sha256",
            "mlkem768nistp256-sha256",
            "mlkem1024nistp384-sha384",
            "ecdh-sha2-nistp256",
            "ecdh-sha2-nistp384",
            "ecdh-sha2-nistp521",
            "curve25519-sha256",
            "curve25519-sha256@libssh.org",
            "diffie-hellman-group16-sha512",
            "diffie-hellman-group18-sha512",
            "diffie-hellman-group-exchange-sha256"
        ],
        "SshMacs": [
            "hmac-sha2-256-etm@openssh.com",
            "hmac-sha2-512-etm@openssh.com"
        ],
        "ContentEncryptionCiphers": [
            "aes256-cbc",
            "aes192-cbc",
            "aes128-cbc"
        ],
        "HashAlgorithms": [
            "sha256",
            "sha384",
            "sha512"
        ],
        "TlsCiphers": [
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
```

```
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
        ],
        "Type": "SERVER",
        "Protocols": [
           "SFTP",
           "FTPS"
        ]
    }
}
```

TransferSecurityPolicy-SshAuditCompliant-2025-02

The following shows the TransferSecurityPolicy-SshAuditCompliant-2025-02 security policy.

> **Note**
>
> This security policy is designed around the recommendations provided by the `ssh-audit` tool, and is 100% compliant with that tool.

```
{
  "SecurityPolicy": {
    "Fips": false,
    "Protocols": [
      "SFTP",
      "FTPS"
    ],
    "SecurityPolicyName": "TransferSecurityPolicy-SshAuditCompliant-2025-02",
    "SshCiphers": [
      "aes128-gcm@openssh.com",
      "aes256-gcm@openssh.com",
      "aes128-ctr",
      "aes256-ctr",
      "aes192-ctr"
    ],
    "SshKexs": [
      "curve25519-sha256",
      "curve25519-sha256@libssh.org",
      "diffie-hellman-group18-sha512",
```

```
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com"
    ],
    "ContentEncryptionCiphers": [
      "aes256-cbc",
      "aes192-cbc",
      "aes128-cbc",
      "3des-cbc"
    ],
    "HashAlgorithms": [
      "sha256",
      "sha384",
      "sha512",
      "sha1"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ],
    "Type": "SERVER"
  }
}
```

TransferSecurityPolicy-2024-01

The following shows the TransferSecurityPolicy-2024-01 security policy.

```
{
    "SecurityPolicy": {
        "Fips": false,
        "SecurityPolicyName": "TransferSecurityPolicy-2024-01",
        "SshCiphers": [
            "aes128-gcm@openssh.com",
```

```
            "aes256-gcm@openssh.com",
            "aes128-ctr",
            "aes256-ctr",
            "aes192-ctr"
        ],
        "SshKexs": [
            "ecdh-sha2-nistp256",
            "ecdh-sha2-nistp384",
            "ecdh-sha2-nistp521",
            "curve25519-sha256",
            "curve25519-sha256@libssh.org",
            "diffie-hellman-group18-sha512",
            "diffie-hellman-group16-sha512",
            "diffie-hellman-group-exchange-sha256"
        ],
        "SshMacs": [
            "hmac-sha2-256-etm@openssh.com",
            "hmac-sha2-512-etm@openssh.com"
        ],
        "ContentEncryptionCiphers": [
            "aes256-cbc",
            "aes192-cbc",
            "aes128-cbc",
            "3des-cbc"
        ],
        "HashAlgorithms": [
            "sha256",
            "sha384",
            "sha512",
            "sha1"
        ],
        "TlsCiphers": [
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
        ]
    }
}
```

TransferSecurityPolicy-FIPS-2024-01/TransferSecurityPolicy-FIPS-2024-05

The following shows the TransferSecurityPolicy-FIPS-2024-01 and TransferSecurityPolicy-FIPS-2024-05 security policies.

> **Note**
>
> The FIPS service endpoint and TransferSecurityPolicy-FIPS-2024-01 and TransferSecurityPolicy-FIPS-2024-05 security policies are only available in some AWS Regions. For more information, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*.
>
> The only difference between these two security policies is that TransferSecurityPolicy-FIPS-2024-01 supports the `ssh-rsa` algorithm, and TransferSecurityPolicy-FIPS-2024-05 doesn't.


```
{
    "SecurityPolicy": {
        "Fips": true,
        "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2024-01",
        "SshCiphers": [
            "aes128-gcm@openssh.com",
            "aes256-gcm@openssh.com",
            "aes128-ctr",
            "aes256-ctr",
            "aes192-ctr"
        ],
        "SshKexs": [
            "ecdh-sha2-nistp256",
            "ecdh-sha2-nistp384",
            "ecdh-sha2-nistp521",
            "diffie-hellman-group18-sha512",
            "diffie-hellman-group16-sha512",
            "diffie-hellman-group-exchange-sha256"
        ],
        "SshMacs": [
            "hmac-sha2-256-etm@openssh.com",
            "hmac-sha2-512-etm@openssh.com"
        ],
        "ContentEncryptionCiphers": [
            "aes256-cbc",
```

```
            "aes192-cbc",
            "aes128-cbc",
            "3des-cbc"
        ],
        "HashAlgorithms": [
            "sha256",
            "sha384",
            "sha512",
            "sha1"
        ],
        "TlsCiphers": [
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
        ]
    }
}
```

TransferSecurityPolicy-2023-05

The following shows the TransferSecurityPolicy-2023-05 security policy.

```
{
    "SecurityPolicy": {
        "Fips": false,
        "SecurityPolicyName": "TransferSecurityPolicy-2023-05",
        "SshCiphers": [
            "aes256-gcm@openssh.com",
            "aes128-gcm@openssh.com",
            "aes256-ctr",
            "aes192-ctr"
        ],
        "SshKexs": [
            "curve25519-sha256",
            "curve25519-sha256@libssh.org",
            "diffie-hellman-group16-sha512",
            "diffie-hellman-group18-sha512",
            "diffie-hellman-group-exchange-sha256"
```

```
        ],
        "SshMacs": [
            "hmac-sha2-512-etm@openssh.com",
            "hmac-sha2-256-etm@openssh.com"
        ],
        "ContentEncryptionCiphers": [
            "aes256-cbc",
            "aes192-cbc",
            "aes128-cbc",
            "3des-cbc"
        ],
        "HashAlgorithms": [
            "sha256",
            "sha384",
            "sha512",
            "sha1"
        ],
        "TlsCiphers": [
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
        ]
    }
}
```

TransferSecurityPolicy-FIPS-2023-05

The FIPS certification details for AWS Transfer Family can be found at https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules/search/all

The following shows the TransferSecurityPolicy-FIPS-2023-05 security policy.

> **Note**
>
> The FIPS service endpoint and TransferSecurityPolicy-FIPS-2023-05 security policy is only available in some AWS Regions. For more information, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*.

```
{
    "SecurityPolicy": {
        "Fips": true,
        "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2023-05",
        "SshCiphers": [
            "aes256-gcm@openssh.com",
            "aes128-gcm@openssh.com",
            "aes256-ctr",
            "aes192-ctr"
        ],
        "SshKexs": [
            "diffie-hellman-group16-sha512",
            "diffie-hellman-group18-sha512",
            "diffie-hellman-group-exchange-sha256"
        ],
        "SshMacs": [
            "hmac-sha2-256-etm@openssh.com",
            "hmac-sha2-512-etm@openssh.com"
        ],
        "ContentEncryptionCiphers": [
            "aes256-cbc",
            "aes192-cbc",
            "aes128-cbc",
            "3des-cbc"
        ],
        "HashAlgorithms": [
            "sha256",
            "sha384",
            "sha512",
            "sha1"
        ],
        "TlsCiphers": [
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
        ]
    }
}
```

TransferSecurityPolicy-2022-03

The following shows the TransferSecurityPolicy-2022-03 security policy.


```
{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2022-03",
    "SshCiphers": [
      "aes256-gcm@openssh.com",
      "aes128-gcm@openssh.com",
      "aes256-ctr",
      "aes192-ctr"
    ],
    "SshKexs": [
      "curve25519-sha256",
      "curve25519-sha256@libssh.org",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group-exchange-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512",
      "hmac-sha2-256"
    ],
    "ContentEncryptionCiphers": [
      "aes256-cbc",
      "aes192-cbc",
      "aes128-cbc",
      "3des-cbc"
    ],
    "HashAlgorithms": [
      "sha256",
      "sha384",
      "sha512",
      "sha1"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
```

```
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
  }
}
```

TransferSecurityPolicy-2020-06 and TransferSecurityPolicy-Restricted-2020-06

The following shows the TransferSecurityPolicy-2020-06 security policy.

> **Note**
>
> The TransferSecurityPolicy-Restricted-2020-06 and TransferSecurityPolicy-2020-06 security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.

```
{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2020-06",
    "SshCiphers": [
      "chacha20-poly1305@openssh.com", //Not included in TransferSecurityPolicy-
Restricted-2020-06
      "aes128-ctr",
      "aes192-ctr",
      "aes256-ctr",
      "aes128-gcm@openssh.com",
      "aes256-gcm@openssh.com"
    ],
    "SshKexs": [
      "ecdh-sha2-nistp256",
      "ecdh-sha2-nistp384",
      "ecdh-sha2-nistp521",
      "diffie-hellman-group-exchange-sha256",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group14-sha256"
    ],
    "SshMacs": [
```

```
      "umac-128-etm@openssh.com",
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com",
      "umac-128@openssh.com",
      "hmac-sha2-256",
      "hmac-sha2-512"
    ],
    "ContentEncryptionCiphers": [
      "aes256-cbc",
      "aes192-cbc",
      "aes128-cbc",
      "3des-cbc"
    ],
    "HashAlgorithms": [
      "sha256",
      "sha384",
      "sha512",
      "sha1"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
  }
}
```

TransferSecurityPolicy-FIPS-2020-06

The FIPS certification details for AWS Transfer Family can be found at https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules/search/all

The following shows the TransferSecurityPolicy-FIPS-2020-06 security policy.

Note

The FIPS service endpoint and TransferSecurityPolicy-FIPS-2020-06 security policy are only available in some AWS Regions. For more information, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*.


```
{
  "SecurityPolicy": {
    "Fips": true,
    "SecurityPolicyName": "TransferSecurityPolicy-FIPS-2020-06",
    "SshCiphers": [
      "aes128-ctr",
      "aes192-ctr",
      "aes256-ctr",
      "aes128-gcm@openssh.com",
      "aes256-gcm@openssh.com"
    ],
    "SshKexs": [
      "ecdh-sha2-nistp256",
      "ecdh-sha2-nistp384",
      "ecdh-sha2-nistp521",
      "diffie-hellman-group-exchange-sha256",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group14-sha256"
    ],
    "SshMacs": [
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha2-256",
      "hmac-sha2-512"
    ],
    "ContentEncryptionCiphers": [
      "aes256-cbc",
      "aes192-cbc",
      "aes128-cbc",
      "3des-cbc"
    ],
    "HashAlgorithms": [
      "sha256",
      "sha384",
```

```
      "sha512",
      "sha1"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
    ]
  }
}
```

TransferSecurityPolicy-2018-11 and TransferSecurityPolicy-Restricted-2018-11

The following shows the TransferSecurityPolicy-2018-11 security policy.

> **Note**
>
> The TransferSecurityPolicy-Restricted-2018-11 and TransferSecurityPolicy-2018-11 security policies are identical, except that the restricted policy doesn't support the `chacha20-poly1305@openssh.com` cipher.

```
{
  "SecurityPolicy": {
    "Fips": false,
    "SecurityPolicyName": "TransferSecurityPolicy-2018-11",
    "SshCiphers": [
      "chacha20-poly1305@openssh.com", //Not included in TransferSecurityPolicy-
Restricted-2018-11
      "aes128-ctr",
      "aes192-ctr",
      "aes256-ctr",
      "aes128-gcm@openssh.com",
      "aes256-gcm@openssh.com"
    ],
    "SshKexs": [
      "curve25519-sha256",
```

```
      "curve25519-sha256@libssh.org",
      "ecdh-sha2-nistp256",
      "ecdh-sha2-nistp384",
      "ecdh-sha2-nistp521",
      "diffie-hellman-group-exchange-sha256",
      "diffie-hellman-group16-sha512",
      "diffie-hellman-group18-sha512",
      "diffie-hellman-group14-sha256",
      "diffie-hellman-group14-sha1"
    ],
    "SshMacs": [
      "umac-64-etm@openssh.com",
      "umac-128-etm@openssh.com",
      "hmac-sha2-256-etm@openssh.com",
      "hmac-sha2-512-etm@openssh.com",
      "hmac-sha1-etm@openssh.com",
      "umac-64@openssh.com",
      "umac-128@openssh.com",
      "hmac-sha2-256",
      "hmac-sha2-512",
      "hmac-sha1"
    ],
    "ContentEncryptionCiphers": [
      "aes256-cbc",
      "aes192-cbc",
      "aes128-cbc",
      "3des-cbc"
    ],
    "HashAlgorithms": [
      "sha256",
      "sha384",
      "sha512",
      "sha1"
    ],
    "TlsCiphers": [
      "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
      "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",
      "TLS_RSA_WITH_AES_128_CBC_SHA256",
```

```
      "TLS_RSA_WITH_AES_256_CBC_SHA256"
    ]
  }
}
```

Security policies for AWS Transfer Family SFTP connectors

SFTP connector security policies in AWS Transfer Family allow you to limit the set of cryptographic algorithms (message authentication codes (MACs), key exchanges (KEXs), and cipher suites) associated with your SFTP connector. The following is a list of supported cryptographic algorithms for each SFTP connector security policy.

> **Note**
>
> `TransferSFTPConnectorSecurityPolicy-2024-03` is the default security policy that is applied to SFTP connectors.

You can change the security policy for your connector. Select **Connectors** from the Transfer Family left navigation pane, and select your connector. Then select **Edit** in the **Sftp configuration** section. In the **Cryptographic algorithm options** section, choose any available security policy from the dropdown list in the **Security Policy** field.

Cryptographic algorithms

For host keys, SFTP connectors support all the algorithms that are supported for Transfer Family servers, except for ed25519:

- `rsa-sha2-256`
- `rsa-sha2-512`
- `ecdsa-sha2-nistp256`
- `ecdsa-sha2-nistp384`
- `ecdsa-sha2-nistp521`

Additionally, for host keys, we support `ssh-rsa`, but only for `TransferSFTPConnectorSecurityPolicy-2023-07`.

For authentication, SFTP connectors support the following key types:

- ssh-rsa
- ecdsa

SFTP connector security policy details

The following table shows the specific cryptographic algorithms supported by each SFTP connector security policy.

Security policy	TransferSFTPConnectorSecurityPolicy-FIPS-2024-10	TransferSFTPConnectorSecurityPolicy-2024-03	TransferSFTPConnectorSecurityPolicy-2023-07
Ciphers			
aes128-ctr			♦
aes128-gcm@openssh.com	♦	♦	♦
aes192-ctr		♦	♦
aes256-ctr		♦	♦
aes256-gcm@openssh.com	♦	♦	♦
Kexs			
curve25519-sha256		♦	♦
curve25519-sha256@libssh.org		♦	♦
diffie-hellman-group14-sha1			♦
diffie-hellman-group16-sha512		♦	♦

Security policy	TransferSFTPConnectorSecurityPolicy-FIPS-2024-10	TransferSFTPConnectorSecurityPolicy-2024-03	TransferSFTPConnectorSecurityPolicy-2023-07
diffie-hellman-group18-sha512		♦	♦
diffie-hellman-group-exchange-sha256		♦	♦
ecdh-sha2-nistp256	♦		
ecdh-sha2-nistp384	♦		
ecdh-sha2-nistp521	♦		
Macs			
hmac-sha2-512-etm@openssh.com		♦	♦
hmac-sha2-256-etm@openssh.com		♦	♦
hmac-sha2-512	♦	♦	♦
hmac-sha2-256	♦	♦	♦
hmac-sha1			♦
hmac-sha1-96			♦
Host Key Algorithms			
rsa-sha2-256	♦	♦	♦
rsa-sha2-512	♦	♦	♦
ecdsa-sha2-nistp256	♦	♦	♦
ecdsa-sha2-nistp384		♦	♦

Security policy	TransferSFTPConnectorSecurityPolicy-FIPS-2024-10	TransferSFTPConnectorSecurityPolicy-2024-03	TransferSFTPConnectorSecurityPolicy-2023-07
ecdsa-sha2-nistp521		♦	♦
ssh-rsa			♦

Using hybrid post-quantum key exchange with AWS Transfer Family

Transfer Family supports a hybrid post-quantum key establishment option for the Secure Shell (SSH) protocol. Post-quantum key establishment is needed because it's already possible to record network traffic and save it for decryption in future by a quantum computer, which is called a *store-now-harvest-later* attack.

You can use this option when you connect to Transfer Family for secure file transfers into and out of Amazon Simple Storage Service (Amazon S3) storage or Amazon Elastic File System (Amazon EFS). Post-quantum hybrid key establishment in SSH introduces post-quantum key establishment mechanisms, which it uses in conjunction with classical key exchange algorithms. SSH keys created with classical cipher suites are safe from brute-force attacks with current technology. However, classical encryption isn't expected to remain secure after the emergence of large-scale quantum computing in the future.

If your organization relies on the long-term confidentiality of data passed over a Transfer Family connection, you should consider a plan to migrate to post-quantum cryptography before large-scale quantum computers become available for use.

To protect data encrypted today against potential future attacks, AWS is participating with the cryptographic community in the development of quantum-resistant or post-quantum algorithms. We've implemented hybrid post-quantum key exchange cipher suites in Transfer Family that combine classic and post-quantum elements.

These hybrid cipher suites are available for use on your production workloads in most AWS Regions. However, because the performance characteristics and bandwidth requirements of hybrid cipher suites are different from those of classic key exchange mechanisms, we recommend that you test them on your Transfer Family connections.

Find out more about post-quantum cryptography in the Post-Quantum Cryptography security blog post.

Contents

- About post-quantum hybrid key exchange in SSH
- How post-quantum hybrid key establishment works in Transfer Family
  - Why ML-KEM?
  - Post-quantum hybrid SSH key exchange and cryptographic requirements (FIPS 140)
- Testing post-quantum hybrid key exchange in Transfer Family
  - Enable post-quantum hybrid key exchange on your SFTP endpoint
  - Set up an SFTP client that supports post-quantum hybrid key exchange
  - Confirm post-quantum hybrid key exchange in SFTP

About post-quantum hybrid key exchange in SSH

Transfer Family supports post-quantum hybrid key exchange cipher suites, which uses both the classical Elliptic Curve Diffie-Hellman (ECDH) key exchange algorithm, and ML-KEM. ML-KEM is a post-quantum public-key encryption and key-establishment algorithm that the National Institute for Standards and Technology(NIST) has designated as its first standard post-quantum key-agreement algorithm.

The client and server still do an ECDH key exchange. Additionally, the server encapsulates a post-quantum shared secret to the client's post-quantum KEM public key, which is advertised in the client's SSH key exchange message. This strategy combines the high assurance of a classical key exchange with the security of the proposed post-quantum key exchanges, to help ensure that the handshakes are protected as long as the ECDH or the post-quantum shared secret cannot be broken.

How post-quantum hybrid key establishment works in Transfer Family

AWS recently announced support for post-quantum key exchange in SFTP file transfers in AWS Transfer Family. Transfer Family securely scales business-to-business file transfers to AWS Storage services using SFTP and other protocols. SFTP is a more secure version of the File Transfer Protocol (FTP) that runs over SSH. The post-quantum key exchange support of Transfer Family raises the security bar for data transfers over SFTP.

The post-quantum hybrid key exchange SFTP support in Transfer Family includes combining post-quantum algorithms ML-KEM-768, and ML-KEM-1024, with ECDH over P256, P384, or Curve25519 curves. The following corresponding SSH key exchange methods are specified in the post-quantum hybrid SSH key exchange draft.

- `mlkem768nistp256-sha256`
- `mlkem1024nistp384-sha384`
- `mlkem768x25519-sha256`

Why ML-KEM?

AWS is committed to supporting standardized, interoperable algorithms. ML-KEM is the only post-quantum key exchange algorithm standardized and approved by the NIST Post-Quantum Cryptography project. Standards bodies are already integrating ML-KEM into protocols. AWS already supports ML-KEM in TLS in some AWS API endpoints.

As part of this commitment, AWS has submitted a draft proposal to the IETF for post-quantum cryptography that combines ML-KEM with NIST-approved curves like P256 for SSH. To help enhance security for our customers, the AWS implementation of the post-quantum key exchange in SFTP and SSH follows that draft. We plan to support future updates to it until our proposal is adopted by the IETF and becomes a standard.

The new key exchange methods (listed in section How post-quantum hybrid key establishment works in Transfer Family) might change as the draft evolves towards standardization.

> **Note**
>
> Post-quantum algorithm support is currently available for post-quantum hybrid key exchange in TLS for AWS KMS (see Using hybrid post-quantum TLS with AWS KMS),AWS Certificate Manager, and AWS Secrets Manager API endpoints.

Post-quantum hybrid SSH key exchange and cryptographic requirements (FIPS 140)

For customers that require FIPS compliance, Transfer Family provides FIPS-approved cryptography in SSH by using the AWS FIPS 140-certified, open-source cryptographic library, AWS-LC. The post-

quantum hybrid key exchange methods supported in the TransferSecurityPolicy-FIPS-2025-03 in Transfer Family are FIPS approved according to NIST's SP 800-56Cr2 (section 2). The German Federal Office for Information Security (BSI) and the Agence nationale de la sécurité des systèmes d'information (ANSSI) of France also recommend such post-quantum hybrid key exchange methods.

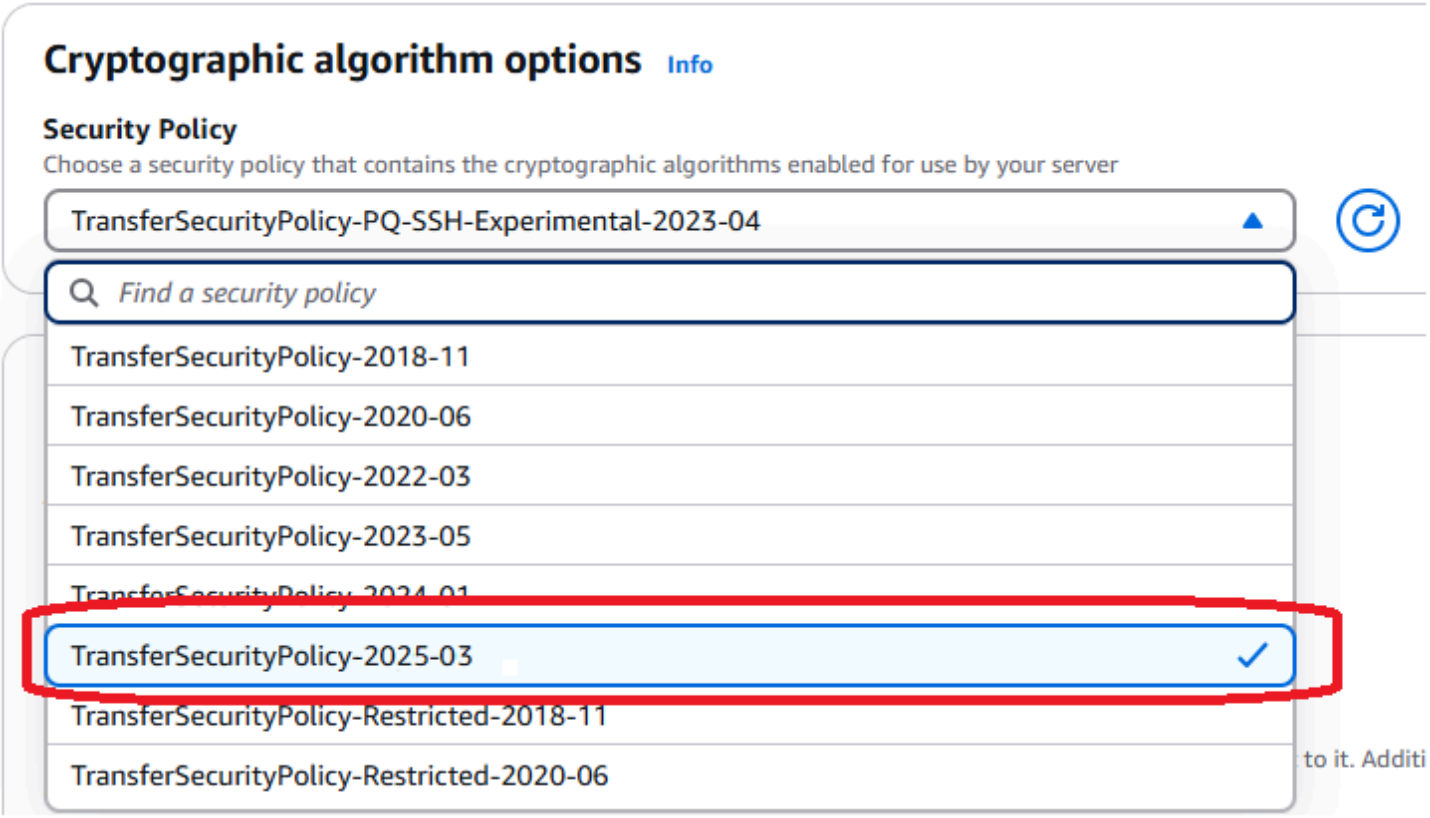
Testing post-quantum hybrid key exchange in Transfer Family

This section describes the steps you take to test post-quantum hybrid key exchange.

1. Enable post-quantum hybrid key exchange on your SFTP endpoint.
2. Use an SFTP client (such as Set up an SFTP client that supports post-quantum hybrid key exchange) that supports post-quantum hybrid key exchange by following the guidance in the aforementioned draft specification.
3. Transfer a file using a Transfer Family server.
4. Confirm post-quantum hybrid key exchange in SFTP.

Enable post-quantum hybrid key exchange on your SFTP endpoint

You can choose the SSH policy when you create a new SFTP server endpoint in Transfer Family, or by editing the Cryptographic algorithm options in an existing SFTP endpoint. The following snapshot shows an example of the AWS Management Console where you update the SSH policy.



The SSH policy names that support post-quantum key exchange are **TransferSecurityPolicy-2025-03** and **TransferSecurityPolicy-FIPS-2025-03**. For more details on Transfer Family policies, see Security policies for AWS Transfer Family servers.

Set up an SFTP client that supports post-quantum hybrid key exchange

After you select the correct post-quantum SSH policy in your SFTP Transfer Family endpoint, you can experiment with post-quantum SFTP in Transfer Family. Install the latest OpenSSH client (such as version 9.9) on your local system to test.

> **Note**
>
> Make sure that your client supports one or more of the ML-KEM algorithms listed earlier. You can view the supported algorithms for your version of OpenSSH by running this command: `ssh -Q kex`.

You can run the example SFTP client to connect to your SFTP endpoint (for example, `s-1111aaaa2222bbbb3.server.transfer.us-west-2.amazonaws.com`) by using the post-quantum hybrid key exchange methods, as shown in the following command.

```
sftp -v -o \
   KexAlgorithms=mlkem768x25519-sha256 \
   -i username_private_key_PEM_file \
   username@server-id.server.transfer.region-id.amazonaws.com
```

In the previous command, replace the following items with your own information:

- Replace *username_private_key_PEM_file* with the SFTP user's private key PEM-encoded file
- Replace *username* with the SFTP user name
- Replace *server-id* with the Transfer Family server ID
- Replace *region-id* with the actual region where your Transfer Family server is located

Confirm post-quantum hybrid key exchange in SFTP

To confirm that post-quantum hybrid key exchange was used during an SSH connection for SFTP to Transfer Family, check the client output. Optionally, you can use a packet capture program. If you use the OpenSSH 9.9 client, the output should look similar to the following (omitting irrelevant information for brevity):

```
% sftp -o KexAlgorithms=mlkem768x25519-sha256 -v -o IdentitiesOnly=yes -
i username_private_key_PEM_file username@s-1111aaaa2222bbbb3.server.transfer.us-
west-2.amazonaws.com
OpenSSH_9.9p2, OpenSSL 3.4.1 11 Feb 2025
debug1: Reading configuration data /Users/username/.ssh/config
debug1: /Users/username/.ssh/config line 146: Applying options for *
debug1: Reading configuration data /Users/username/.ssh/bastions-config
debug1: Reading configuration data /opt/homebrew/etc/ssh/ssh_config
debug1: Connecting to s-1111aaaa2222bbbb3.server.transfer.us-west-2.amazonaws.com
 [xxx.yyy.zzz.nnn] port 22.
debug1: Connection established.
[...]
debug1: Local version string SSH-2.0-OpenSSH_9.9
debug1: Remote protocol version 2.0, remote software version AWS_SFTP_1.1
debug1: compat_banner: no match: AWS_SFTP_1.1
debug1: Authenticating to s-1111aaaa2222bbbb3.server.transfer.us-
west-2.amazonaws.com:22 as 'username'
debug1: load_hostkeys: fopen /Users/username/.ssh/known_hosts2: No such file or
 directory
[...]
debug1: SSH2_MSG_KEXINIT sent
```

```
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: mlkem768x25519-sha256
debug1: kex: host key algorithm: ssh-ed25519
debug1: kex: server->client cipher: aes128-ctr MAC: hmac-sha2-256-etm@openssh.com
 compression: none
debug1: kex: client->server cipher: aes128-ctr MAC: hmac-sha2-256-etm@openssh.com
 compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: SSH2_MSG_KEX_ECDH_REPLY received
debug1: Server host key: ssh-ed25519 SHA256:Ic1Ti0cdDmFdStj06rfU0cmmNccwAha/ASH2unr6zX0
[...]
debug1: rekey out after 4294967296 blocks
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey in after 4294967296 blocks
[...]
Authenticated to s-1111aaaa2222bbbb3.server.transfer.us-west-2.amazonaws.com
 ([xxx.yyy.zzz.nnn]:22) using "publickey".
debug1: channel 0: new session [client-session] (inactive timeout: 0)
[...]
Connected to s-1111aaaa2222bbbb3.server.transfer.us-west-2.amazonaws.com.
sftp>
```

The output shows that client negotiation occurred using the post-quantum hybrid `mlkem768x25519-sha256` method and successfully established an SFTP session.

Data protection and encryption

The AWS shared responsibility model applies to data protection in AWS Transfer Family (Transfer Family). 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. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data privacy FAQ. For information about data protection in Europe, see the AWS shared responsibility model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS IAM Identity Center. 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 support TLS 1.2.
- Set up API and user activity logging with AWS CloudTrail.
- 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 personal data that is stored in Amazon S3.
- If you require FIPS 140-2 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) 140-2.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Transfer Family or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any configuration data that you enter into Transfer Family service configuration, or other services' configurations, might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

In contrast, data from upload and download operations into and out of Transfer Family servers is treated as completely private and never exists outside of encrypted channels—such as an SFTP or FTPS connection. This data is only ever accessible to authorized persons.

Data encryption in Transfer Family

AWS Transfer Family uses the default encryption options you set for your Amazon S3 bucket to encrypt your data. When you enable encryption on a bucket, all objects are encrypted when they are stored in the bucket. The objects are encrypted by using server-side encryption with either Amazon S3 managed keys (SSE-S3) or AWS Key Management Service (AWS KMS) managed keys (SSE-KMS). For information about server-side encryption, see Protecting data using server-side encryption in the *Amazon Simple Storage Service User Guide*.

The following steps show you how to encrypt data in AWS Transfer Family.

To allow encryption in AWS Transfer Family

1. Enable default encryption for your Amazon S3 bucket. For instructions, see Amazon S3 default encryption for S3 buckets in the *Amazon Simple Storage Service User Guide*.

2. Update the AWS Identity and Access Management (IAM) role policy that is attached to the user to grant the required AWS Key Management Service (AWS KMS) permissions.
3. If you are using a session policy for the user, the session policy must grant the required AWS KMS permissions.

The following example shows an IAM policy that grants the minimum permissions required when using AWS Transfer Family with an Amazon S3 bucket that is enabled for AWS KMS encryption. Include this example policy in both the user IAM role policy and session policy, if you are using one.

```
{
   "Sid": "Stmt1544140969635",
   "Action": [
      "kms:Decrypt",
      "kms:Encrypt",
      "kms:GenerateDataKey",
      "kms:GetPublicKey",
      "kms:ListKeyPolicies"
   ],
   "Effect": "Allow",
   "Resource": "arn:aws:kms:region:account-id:key/kms-key-id"
}
```

Note

The KMS key ID that you specify in this policy must be the same as the one specified for the default encryption in step 1.
Root, or the IAM role that is used for the user, must be allowed in the AWS KMS key policy. For information about the AWS KMS key policy, see Using key policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

AWS Transfer Family encryption at rest

Because AWS Transfer Family is a file transfer service, it doesn't manage your storage data at rest. The storage services and systems that AWS Transfer Family supports are responsible for protecting data in that state. However, there is some service-related data that AWS Transfer Family manages at rest.

What's encrypted?

The only data that AWS Transfer Family handles at rest relates to the details it needs to operate your file transfer servers and process transfers. AWS Transfer Family stores the following data with full at-rest encryption in Amazon DynamoDB:

- Server configurations (for example, server settings, protocol configurations, and endpoint details).
- User authentication data, including SSH public keys and user metadata.
- Workflow execution details and step configurations.
- Connector configurations and authentication credentials for third-party systems. These credentials are encrypted using AWS Transfer Family managed encryption keys.

Key management

You can't manage the encryption keys that AWS Transfer Family uses to store information in DynamoDB related to running your servers and processing transfers. This information includes your server configurations, user authentication data, workflow details, and connector credentials.

What's not encrypted?

Though AWS Transfer Family doesn't control how your storage data is encrypted at rest, we still recommend configuring your storage locations with the highest level of security that they support. For example, you can encrypt objects with Amazon S3 managed encryption keys (SSE-S3) or AWS KMS keys (SSE-KMS).

Learn more about how AWS storage services encrypt data at rest:

- Amazon S3
- Amazon EFS

Managing SSH and PGP keys in Transfer Family

In this section, you can find information about SSH keys, including how to generate them and how to rotate them. For details about using Transfer Family with AWS Lambda to manage keys, see the blog post Enabling user self-service key management with AWS Transfer Family and AWS Lambda.

For automated deployment and management of users with multiple SSH keys, see Transfer Family Terraform modules.

> **Note**
>
> AWS Transfer Family accepts RSA, ECDSA, and ED25519 keys for SSH authentication.

This section also covers how to generate and manage Pretty Good Privacy (PGP) keys.

For a comprehensive overview of all supported encryption and key algorithms, including recommendations for different use cases, see Encryption and key algorithms overview.

Encryption and key algorithms overview

AWS Transfer Family supports different types of algorithms for different purposes. Understanding which algorithms to use for your specific use case helps ensure secure and compatible file transfers.

Algorithm Quick Reference

Use Case	Recommended Algorithm	FIPS Compliant	Notes
SSH/SFTP Authentication	RSA (rsa-sha2-256/512), ECDSA, or ED25519	RSA: Yes, ECDSA: Yes, ED25519: No	Compatible with all SSH clients and servers
PGP Key Generation	RSA or ECC (NIST)	Yes	For workflow decryption
PGP File Encryption	AES-256	Yes	Determined by PGP software

SSH authentication algorithms

These algorithms are used for SSH/SFTP authentication between clients and AWS Transfer Family servers. Choose one of these when generating SSH key pairs for user authentication or server host keys.

RSA (Recommended)

Compatible with all SSH clients and servers, and FIPS-compliant. Use with SHA-2 hashing for enhanced security:

- `rsa-sha2-256` - Recommended for most use cases
- `rsa-sha2-512` - Higher security option

ED25519

Modern and efficient. Smaller key sizes with strong security:

- `ssh-ed25519` - Fast and secure, but not FIPS-compliant

ECDSA

Elliptic curve option. Good balance of security and performance:

- `ecdsa-sha2-nistp256` - Standard curve
- `ecdsa-sha2-nistp384` - Higher security curve
- `ecdsa-sha2-nistp521` - Highest security curve

> **Note**
>
> We support `ssh-rsa` with SHA1 for older security policies. For details, see Cryptographic algorithms.

Choosing the right SSH algorithm

- **For most users:** Use RSA with `rsa-sha2-256` or `rsa-sha2-512`
- **For FIPS compliance:** Use RSA or ECDSA algorithms
- **For modern environments:** ED25519 offers excellent security and performance

PGP encryption and decryption algorithms

PGP (Pretty Good Privacy) uses two types of algorithms working together to encrypt and decrypt files in workflows:

1. **Key pair algorithms** - Used to generate the public/private key pairs for encryption and digital signatures

2. **Symmetric algorithms** - Used to encrypt the actual file data (the key pair algorithms encrypt the symmetric key)

PGP key pair algorithms

Choose one of these algorithms when generating PGP key pairs for workflow decryption:

RSA (Recommended)

Recommended for most users. Widely supported, well-established, and FIPS-compliant. Provides good balance of security and compatibility.

ECC (Elliptic Curve Cryptography)

More efficient than RSA with smaller key sizes while maintaining strong security:

- **NIST curves** - Standard curves widely supported and FIPS-compliant
- **BrainPool curves** - Alternative curves for specific compliance requirements
- **Curve25519** - Modern high-performance curve offering strong security with efficient computation

ElGamal

Legacy algorithm. Supported for compatibility with older systems. Use RSA or ECC for new implementations.

For detailed instructions on generating PGP keys, see Generate PGP keys.

PGP symmetric encryption algorithms

These algorithms encrypt your actual file data. The algorithm used depends on how the PGP file was created by your PGP software:

FIPS-compliant algorithms (recommended for regulated environments)

- **AES-128, AES-192, AES-256** - Advanced Encryption Standard (recommended)

Other supported algorithms

- 3DES, IDEA, CAST5, Blowfish, DES, TwoFish, CAMELLIA-128, CAMELLIA-192, CAMELLIA-256

> **Note**
>
> You don't choose the symmetric algorithm directly when using AWS Transfer Family workflows - it's determined by the PGP software used to create the encrypted file. However, you can configure your PGP software to prefer FIPS-compliant algorithms like AES-256.

For more information about supported symmetric algorithms, see Supported symmetric encryption algorithms.

Generate SSH keys for service-managed users

You can set up your server to authenticate users using the service managed authentication method, where usernames and SSH keys are stored within the service. The user's public SSH key is uploaded to the server as a user's property. This key is used by the server as part of a standard key-based authentication process. Each user can have multiple public SSH keys on file with an individual server. For limits on number of keys that can be stored per user, see AWS Transfer Family endpoints and quotas in the *Amazon Web Services General Reference*.

As an alternative to the service managed authentication method, you can authenticate users using a custom identity provider, or AWS Directory Service for Microsoft Active Directory. For more information, see Working with custom identity providers or Using AWS Directory Service for Microsoft Active Directory.

A server can only authenticate users using one method (service managed, directory service, or custom identity provider), and that method cannot be changed after the server is created.

Topics

- Creating SSH keys on macOS, Linux, or Unix
- Creating SSH keys on Microsoft Windows
- Converting an SSH2 key to SSH public key format

Creating SSH keys on macOS, Linux, or Unix

On the macOS, Linux, or Unix operating systems, you use the `ssh-keygen` command to create an SSH public key and SSH private key also known as a key pair.

Note

In the following examples, we do not specify a passphrase: in this case, the tool asks you to enter your passphrase and then repeat it to verify. Creating a passphrase offers better protection for your private key, and might also improve overall system security. You cannot recover your passphrase: if you forget it, you must create a new key.
However, if you are generating a server host key, you *must* specify an empty passphrase, by specifying the `-N ""` option in the command (or by pressing **`Enter`** twice when prompted), because Transfer Family servers cannot request a password at start-up.

To create SSH keys on a macOS, Linux, or Unix operating system

1. On macOS, Linux, or Unix operating systems, open a command terminal.
2. AWS Transfer Family accepts RSA-, ECDSA-, and ED25519-formatted keys. Choose the appropriate command based on the type of key-pair you are generating.

   Tip: Replace *`key_name`* with the actual name of your SSH key pair file.

   - To generate an RSA 4096-bit key pair:

     ```
     ssh-keygen -t rsa -b 4096 -f key_name
     ```

   - To generate an ECDSA 521-bit key-pair (ECDSA has bit sizes of 256, 384, and 521):

     ```
     ssh-keygen -t ecdsa -b 521 -f key_name
     ```

   - To generate an ED25519 key pair:

     ```
     ssh-keygen -t ed25519 -f key_name
     ```

   The following shows an example of the `ssh-keygen` output.

   ```
   ssh-keygen -t rsa -b 4096 -f key_name
   Generating public/private rsa key pair.

   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in key_name.
   ```

```
Your public key has been saved in key_name.pub.
The key fingerprint is:
SHA256:8tDDwPmanTFcEzjTwPGETVWOGW1nVz+gtCCE8hL7PrQ bob.amazon.com
The key's randomart image is:
+---[RSA 4096]----+
|    . ....E      |
| .   = ...       |
|. . . = ..o      |
| . o +  oo =     |
|  + =  .S.= *    |
| . o o ..B + o   |
|     .o.+.* .    |
|     =o*+*.      |
|    ..*o*+.      |
+----[SHA256]-----+
```

Tip: When you run the `ssh-keygen` command as shown preceding, it creates the public and private keys as files in the current directory.

Your SSH key pair is now ready to use. Follow steps 3 and 4 to store the SSH public key for your service-managed users. These users use the keys when they transfer files on Transfer Family server endpoints.

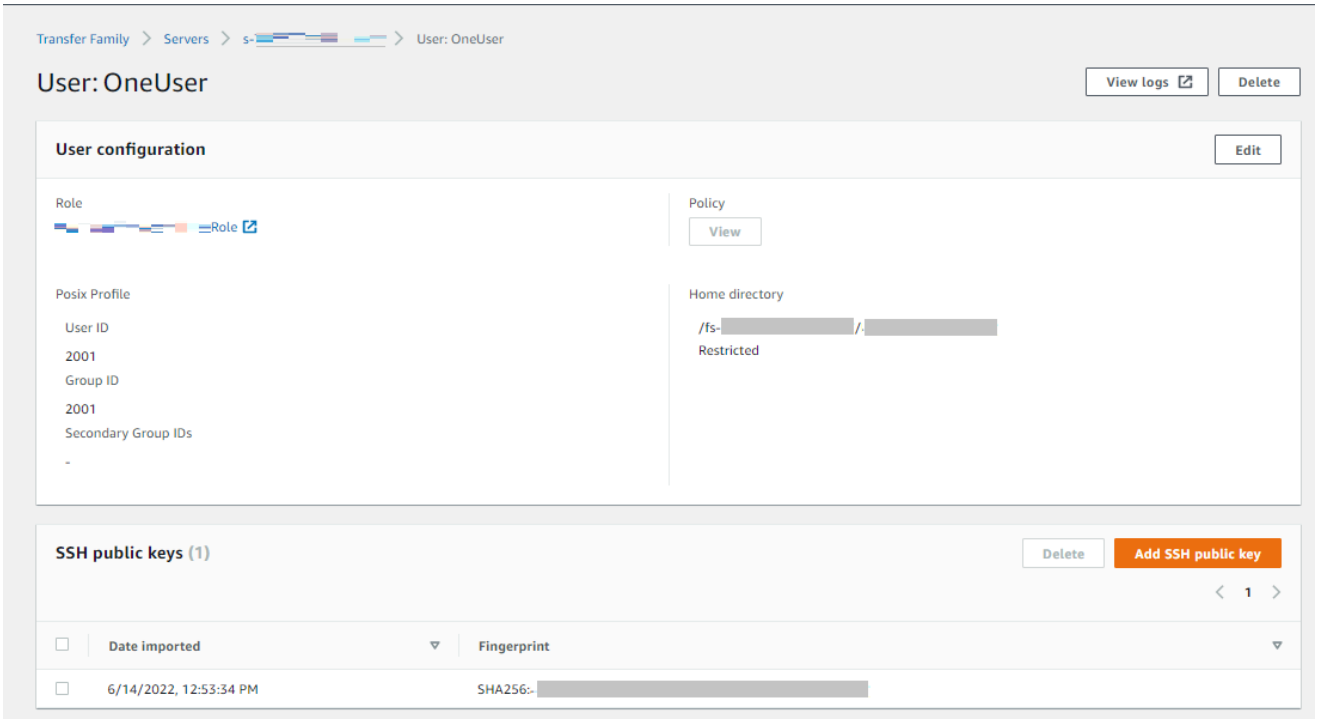
3. Navigate to the *key_name*.pub file and open it.
4. Copy the text and paste it in **SSH public key** for the service-managed user.

   a. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/, then select **Servers** from the navigation pane.

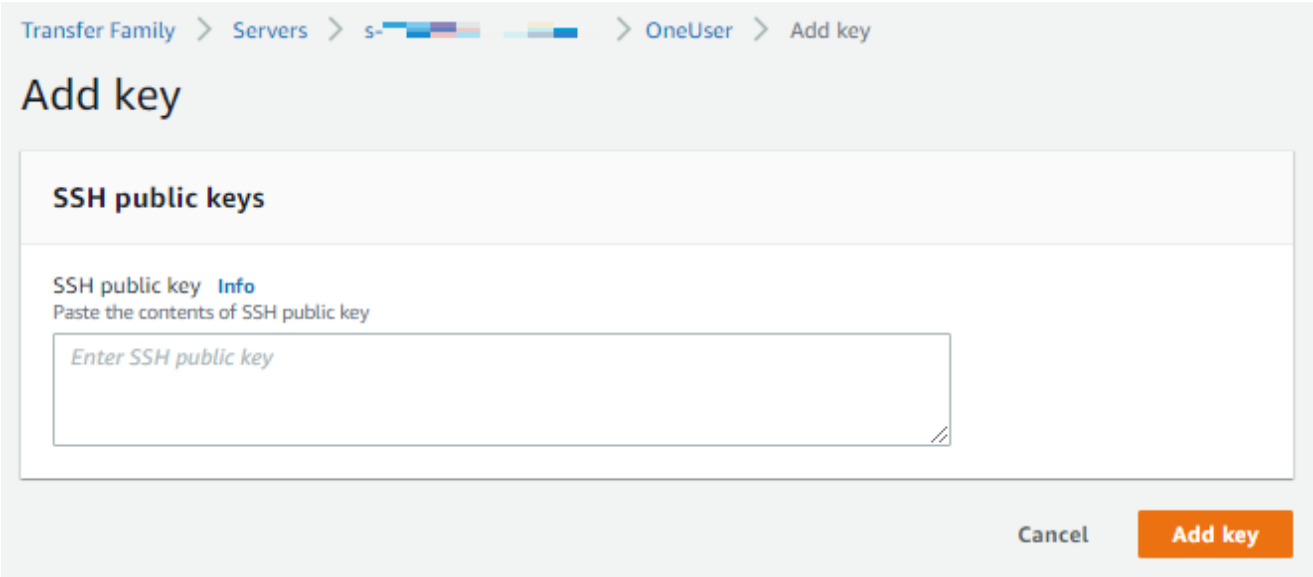
   b. On the **Servers** page, select the **Server ID** for server that contains the user that you want to update.

   c. Select the user for which you are adding a public key.

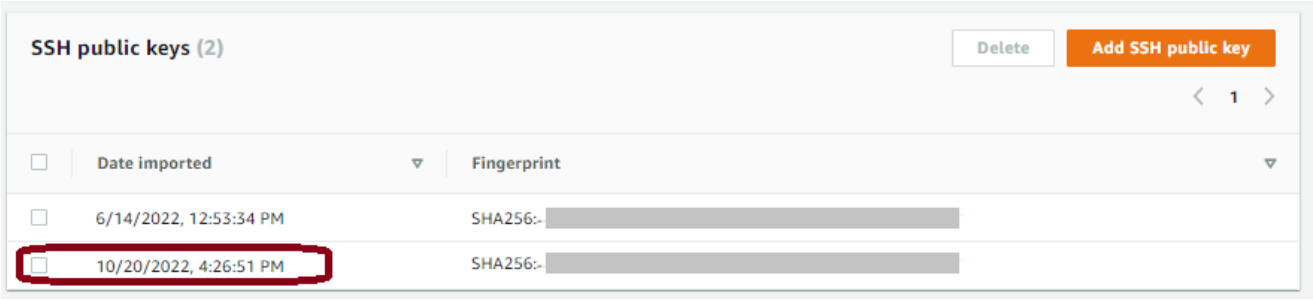
   d. In the **SSH public keys** pane, choose **Add SSH public key**.



e. Paste the text of the public key you generated into the SSH public key text box, and then choose **Add key**.



The new key is listed in the SSH public key pane.



Creating SSH keys on Microsoft Windows

Windows includes OpenSSH as a built-in feature, which you can use to generate SSH keys in the same format as on Linux or macOS. Alternatively, you can use third-party tools like PuTTY's key generator (PuTTYgen).

Using Windows built-in OpenSSH

Recent versions of Windows include OpenSSH by default. You can use the same `ssh-keygen` commands as described in the macOS/Linux section:

1. Open Windows PowerShell or Command Prompt.
2. Run one of the following commands based on the type of key you want to generate:

   - To generate an RSA 4096-bit key pair:

     ```
     ssh-keygen -t rsa -b 4096 -f key_name
     ```

   - To generate an ECDSA 521-bit key-pair:

     ```
     ssh-keygen -t ecdsa -b 521 -f key_name
     ```

   - To generate an ED25519 key pair:

     ```
     ssh-keygen -t ed25519 -f key_name
     ```

3. Follow the same steps as in the macOS/Linux section to upload your public key to AWS Transfer Family.

Using PuTTYgen (third-party tool)

Some third-party SSH clients for Windows, such as PuTTY, use different key formats. PuTTY uses the PPK format for private keys. If you're using PuTTY or related tools like WinSCP, you can use PuTTYgen to create keys in this format.

> **Note**
>
> If you present WinSCP with a private key file not in `.ppk` format, that client offers to convert the key into `.ppk` format for you.

For a tutorial about creating SSH keys by using PuTTYgen, see the SSH.com website.

Converting an SSH2 key to SSH public key format

AWS Transfer Family only accepts SSH-formatted public keys. If you have an SSH2 public key, you need to convert it. An SSH2 public key has the following format:

```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "rsa-key-20160402"
AAAAB3NzaC1yc2EAAAABJQAAAgEAiL0jjDdFqK/kYThqKt7THrjABTPWvXmB3URI
:
:
---- END SSH2 PUBLIC KEY ----
```

An SSH public key has the following format:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAA...
```

Run the following command to convert an SSH2-formatted public key into an SSH-formatted public key. Replace *ssh2-key* with the name of your SSH2 key, and *ssh-key* with the name of your SSH key.

```
ssh-keygen -i -f ssh2-key.pub > ssh-key.pub
```

Rotate SSH keys

For security, we recommend the best practice of rotating your SSH keys. Usually, this rotation is specified as a part of a security policy and is implemented in some automated fashion. Depending

upon the level of security, for a highly sensitive communication, an SSH key pair might be used only once. Doing this eliminates any risk due to stored keys. However, it is much more common to store SSH credentials for a period of time and set an interval that doesn't place undue burden on users. A time interval of three months is common.

> **Note**
>
> For automated SSH key rotation using infrastructure as code, see Transfer Family Terraform modules.

There are two methods used to perform SSH key rotation:

- On the console, you can upload a new SSH public key and delete an existing SSH public key.
- Using the API, you can update existing users by using the DeleteSshPublicKey API to delete a user's Secure Shell (SSH) public key and the ImportSshPublicKey API to add a new Secure Shell (SSH) public key to the user's account.

Console

To perform a key rotation in the console

1. Open the AWS Transfer Family console at https://console.aws.amazon.com/transfer/.
2. Navigate to the **Servers** page.
3. Choose the identifier in the **Server ID** column to see the **Server details** page.
4. Under **Users**, select the check box of the user whose SSH public key that you want to rotate, then choose **Actions**, and then choose **Add key** to see the **Add key** page.

   or

   Choose the username to see the **User details** page, and then choose **Add SSH public key** to see the **Add key** page.
5. Enter the new SSH public key and choose **Add key**.

   > **Important**
   >
   > The format of the SSH public key depends on the type of key you generated.

- For RSA keys, the format is `ssh-rsa` *`string`*.
- For ED25519 keys, the format is `ssh-ed25519` *`string`*.
- For ECDSA keys, the key begins with `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, or `ecdsa-sha2-nistp521`, depending on the size of the key you generated. The beginning string is then followed by *`string`*, similar to the other key types.

You are returned to the **User details** page, and the new SSH public key that you just entered appears in the **SSH public keys** section.

6. Select the check box of the old you key that you want to delete and then choose **Delete**.
7. Confirm the deletion operation by entering the word `delete`, and then choose **Delete**.

API

To perform a key rotation using the API

1. On macOS, Linux, or Unix operating systems, open a command terminal.
2. Retrieve the SSH key that you want to delete by entering the following command. To use this command, replace *`serverID`* with the server ID for your Transfer Family server, and replace *`username`* with your username.

```
aws transfer describe-user --server-id='serverID' --user-name='username'
```

The command returns details about the user. Copy the contents of the `"SshPublicKeyId":` field. You will need to enter this value later in this procedure.

```
"SshPublicKeys": [ { "SshPublicKeyBody": "public-key", "SshPublicKeyId":
 "keyID",
   "DateImported": 1621969331.072 } ],
```

3. Next, import a new SSH key for your user. At the prompt, enter the following command. To use this command, replace *`serverID`* with the server ID for your Transfer Family server, replace *`username`* with your username, and replace *`public-key`* with the fingerprint of your new public key.

```
aws transfer import-ssh-public-key --server-id='serverID' --user-name='username'
   --ssh-public-key-body='public-key'
```

If the command is successful, no output is returned.

4. Finally, delete the old key by running the following command. To use this command, replace *serverID* with the server ID for your Transfer Family server, replace *username* with your username, and replace *keyID-from-step-2* with the key ID value that you copied in step 2 of this procedure

```
aws transfer delete-ssh-public-key --server-id='serverID' --user-name='username'
   --ssh-public-key-id='keyID-from-step-2'
```

5. (Optional) To confirm that the old key no longer exists, repeat step 2.

Generate PGP keys

You can use Pretty Good Privacy (PGP) decryption with the files that Transfer Family processes with workflows. To use decryption in a workflow step, provide a PGP key. For detailed information about PGP key algorithms, including recommendations and FIPS compliance, see PGP key pair algorithms.

The AWS storage blog has a post that describes how to simply decrypt files without writing any code using Transfer Family Managed workflows, Encrypt and decrypt files with PGP and AWS Transfer Family.

The operator that you use to generate your PGP keys depends on your operating system and the version of the key-generation software that you're using.

If you're using Linux or Unix, use your package installer to install gpg. Depending on your Linux distribution, one of the following commands should work for you.

```
sudo yum install gnupg
```

```
sudo apt-get install gnupg
```

For Windows or macOS, you can download what you need from https://gnupg.org/download/.

After you install your PGP key generator software, you run the `gpg --full-gen-key` or `gpg --gen-key` command to generate a key pair.

> **Note**
>
> If you're using GnuPG version 2.3.0 or newer, you must run `gpg --full-gen-key`. When prompted for the type of key to create, choose RSA or ECC. If you choose **ECC**, you can choose from NIST, BrainPool and Curve25519 for the elliptic curve.

Useful gpg subcommands

The following are some useful subcommands for gpg:

- `gpg --help` – This command lists the available options and might include some examples.
- `gpg --list-keys` – This command lists the details for all the key pairs that you have created.
- `gpg --fingerprint` – This command lists the details for all your key pairs, including each key's fingerprint.
- `gpg --export -a` *`user-name`* – This command exports the public key portion of the key for the *`user-name`* that was used when the key was generated.

Manage PGP keys

To manage your PGP keys, use AWS Secrets Manager.

> **Note**
>
> Your secret name includes your Transfer Family server ID. This means you should have already identified or created a server *before* you can store your PGP key information in AWS Secrets Manager.

If you want to use one key and passphrase for all of your users, you can store the PGP key block information under the secret name `aws/transfer/`*`server-id`*`/@pgp-default`, where *`server-id`* is the ID for your Transfer Family server. Transfer Family uses this default key if there is no key where the *`user-name`* matches the user that's executing the workflow.

You can create a key for a specific user. In this case, the format for the secret name is `aws/transfer/`*`server-id`*`/`*`user-name`*, where *`user-name`* matches the user that's running the workflow for a Transfer Family server.

> **Note**
>
> You can store a maximum of 3 PGP private keys, per Transfer Family server, per user.

To configure PGP keys for use with decryption

1. Depending on the version of GPG that you are using, run one of the following commands to generate a PGP key pair.

   - If you are using **GnuPG** version 2.3.0 or newer, run the following command:

     ```
     gpg --full-gen-key
     ```

     You can choose **RSA**, or, if you choose **ECC**, you can choose either **NIST**, **BrainPool** or **Curve25519** for the elliptic curve. If you run gpg `--gen-key` instead, you create a key pair that uses the ECC Curve 25519 encryption algorithm.

   - For versions of **GnuPG** prior to 2.3.0, you can use the following command, since RSA is the default encryption type.

     ```
     gpg --gen-key
     ```

   > **Important**
   >
   > During the key-generation process, you must provide a passphrase and an email address. Make sure to take note of these values. You must provide the passphrase when you enter the key's details into AWS Secrets Manager later in this procedure. And you must provide the same email address to export the private key in the next step.

2. Run the following command to export the private key. To use this command, replace *`private.pgp`* with the name of the file in which to save the private key block, and *`marymajor@example.com`* with the email address that you used when you generated the key pair.

   ```
   gpg --output private.pgp --armor --export-secret-key marymajor@example.com
   ```

3. Use AWS Secrets Manager to store your PGP key.

a. Sign in to the AWS Management Console and open the AWS Secrets Manager console at https://console.aws.amazon.com/secretsmanager/.

b. In the left navigation pane, choose **Secrets**.

c. On the **Secrets** page, choose **Store a new secret**.

d. On the **Choose secret type** page, for **Secret type**, select **Other type of secret**.

e. In the **Key/value pairs** section, choose the **Key/value** tab.

- **Key** – Enter **`PGPPrivateKey`**.

> **Note**
>
> You must enter the **`PGPPrivateKey`** string exactly: do not add any spaces before or between characters.

- **value** – Paste the text of your private key into the value field. You can find the text of your private key in the file (for example, `private.pgp`) that you specified when you exported your key earlier in this procedure. The key begins with `-----BEGIN PGP PRIVATE KEY BLOCK-----` and ends with `-----END PGP PRIVATE KEY BLOCK-----`.

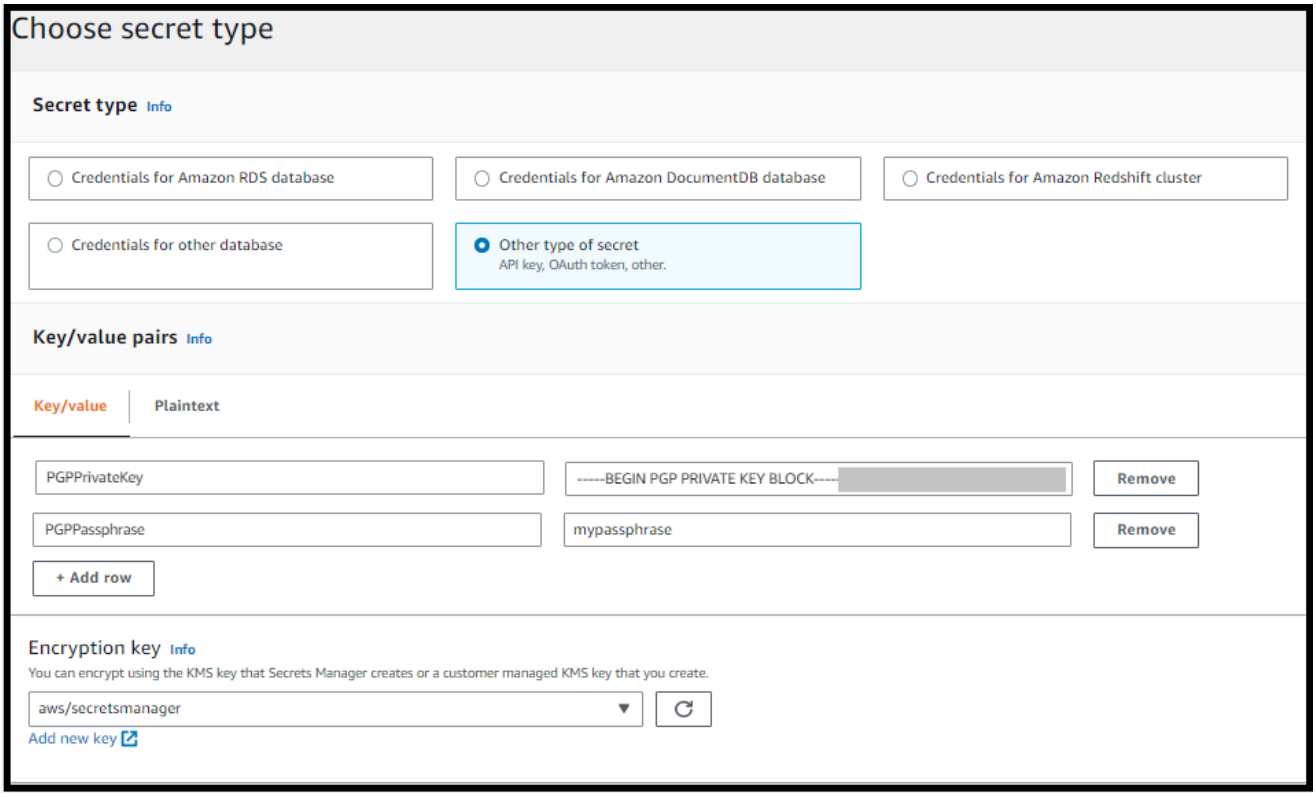
> **Note**
>
> Make sure that the text block contains only the private key and does not contain the public key as well.

f. Select **Add row** and in the **Key/value pairs** section, choose the **Key/value** tab.

- **Key** – Enter **`PGPPassphrase`**.

> **Note**
>
> You must enter the **`PGPPassphrase`** string exactly: do not add any spaces before or between characters.

- **value** – Enter the passphrase you used when you generated your PGP key pair.



> **Note**
>
> You can add up to 3 sets of keys and passphrases. To add a second set, add two new rows, and enter **`PGPPrivateKey2`** and **`PGPPassphrase2`** for the keys, and paste in another private key and passphrase. To add a third set, key values must be **`PGPPrivateKey3`** and **`PGPPassphrase3`**.

g. Choose **Next**.

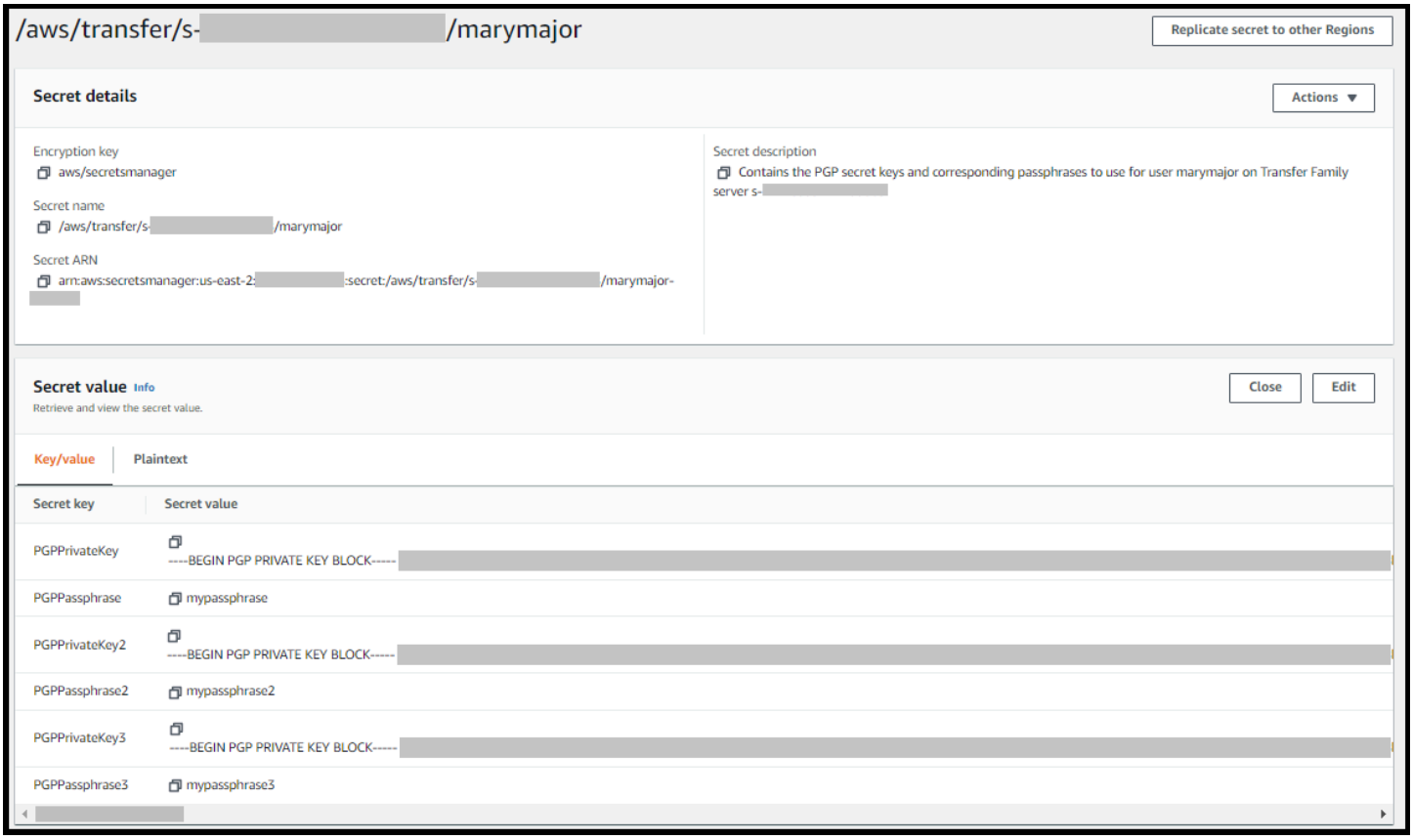
h. On the **Configure secret** page, enter a name and description for your secret.

- If you're creating a default key, that is, a key that can be used by any Transfer Family user, enter **`aws/transfer/`*`server-id`*`/@pgp-default`**. Replace *`server-id`* with the ID of the server that contains the workflow that has a decrypt step.
- If you're creating a key to be used by a specific Transfer Family user, enter **`aws/transfer/`*`server-id`*`/`*`user-name`**. Replace *`server-id`* with the ID of the server that contains the workflow that has a decrypt step, and replace *`user-name`* with the name of the user that's running the workflow. The *`user-name`* is stored in the identity provider that the Transfer Family server is using.

i. Choose **Next** and accept the defaults on the **Configure rotation** page. Then choose **Next**.

j. On the **Review** page, choose **Store** to create and store the secret.

The following screenshot shows the details for the user **`marymajor`** for a specific Transfer Family server. This example shows three keys and their corresponding passphrases.

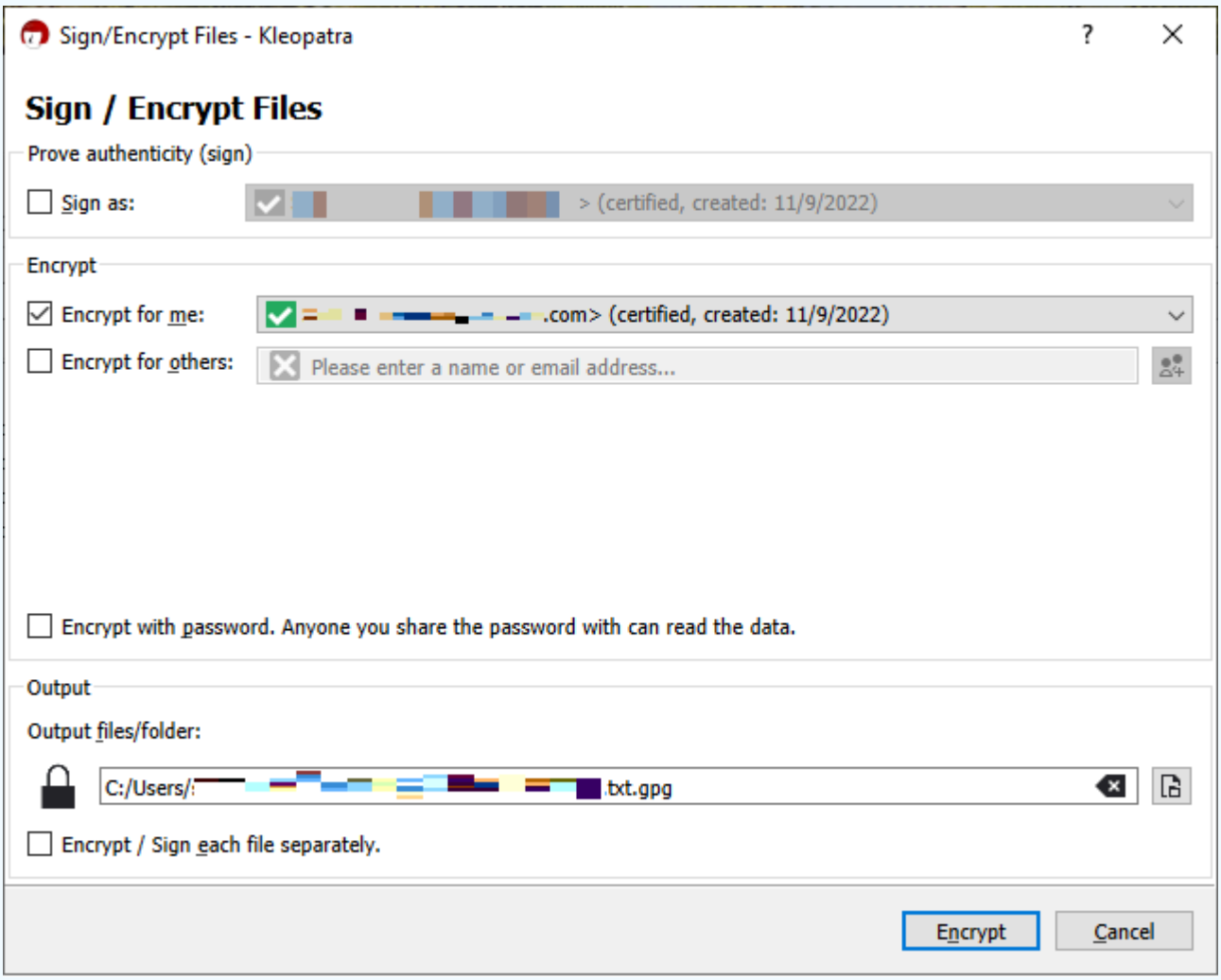


Supported PGP clients

The following clients have been tested with Transfer Family and can be used to generate PGP keys, and to encrypt files that you intend to decrypt with a workflow.

- **Gpg4win + Kleopatra**.

> **Note**
>
> When you select **Sign / Encrypt Files**, make sure to clear the selection for **Sign as**: we do not currently support signing for encrypted files.



If you sign the encrypted file and attempt to upload it to a Transfer Family server with a decryption workflow, you receive the following error:

```
Encrypted file with signed message unsupported
```

- Major **GnuPG** versions: 2.4, 2.3, 2.2, 2.0, and 1.4.

Note that other PGP clients might work as well, but only the clients mentioned here have been tested with Transfer Family.

Identity and access management for AWS Transfer Family

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in)

and *authorized* (have permissions) to use AWS Transfer Family resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS Transfer Family works with IAM
- AWS Transfer Family identity-based policy examples
- AWS Transfer Family tag-based policy examples
- Troubleshooting AWS Transfer Family identity and access
- IAM condition keys for organizational governance

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS Transfer Family identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS Transfer Family works with IAM)
- **IAM administrator** - write policies to manage access (see AWS Transfer Family identity-based policy examples)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Transfer Family works with IAM

Before you use AWS Identity and Access Management (IAM) to manage access to AWS Transfer Family, you should understand what IAM features are available to use with AWS Transfer Family. To

get a high-level view of how AWS Transfer Family and other AWS services work with IAM, see AWS services that work with IAM in the *IAM User Guide*.

Topics

- AWS Transfer Family identity-based policies
- AWS Transfer Family resource-based policies
- Authorization based on AWS Transfer Family tags
- AWS Transfer Family IAM roles

AWS Transfer Family identity-based policies

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. AWS Transfer Family supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON policy elements reference in the *AWS Identity and Access Management User Guide*.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in AWS Transfer Family use the following prefix before the action: `transfer:`. For example, to grant someone permission to create a server, with the Transfer Family `CreateServer` API operation, you include the `transfer:CreateServer` action in their policy. Policy statements must include either an `Action` or `NotAction` element. AWS Transfer Family defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows.

```
"Action": [
      "transfer:action1",
      "transfer:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action.

```
"Action": "transfer:Describe*"
```

To see a list of AWS Transfer Family actions, see Actions defined by AWS Transfer Family in the *Service Authorization Reference*.

Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

The Transfer Family server resource has the following ARN.

```
arn:aws:transfer:${Region}:${Account}:server/${ServerId}
```

For example, to specify the `s-01234567890abcdef` Transfer Family server in your statement, use the following ARN.

```
"Resource": "arn:aws:transfer:us-east-1:123456789012:server/s-01234567890abcdef"
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) in the *Service Authorization Reference*, or IAM ARNs in the *IAM User Guide*.

To specify all instances that belong to a specific account, use the wildcard (*).

```
"Resource": "arn:aws:transfer:us-east-1:123456789012:server/*"
```

Some AWS Transfer Family actions are performed on multiple resources, such as those used in IAM policies. In those cases, you must use the wildcard (*).

```
"Resource": "arn:aws:transfer:*:123456789012:server/*"
```

In some cases you need to specify more than one type of resource, for example, if you create a policy that allows access to Transfer Family servers and users. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
      "resource1",
      "resource2"
            ]
```

To see a list of AWS Transfer Family resources, see Resource types defined by AWS Transfer Family in the *Service Authorization Reference*.

Condition keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

AWS Transfer Family defines its own set of condition keys and also supports using some global condition keys. To see a list of AWS Transfer Family condition keys, see Condition keys for AWS Transfer Family in the *Service Authorization Reference*.

Examples

To view examples of AWS Transfer Family identity-based policies, see AWS Transfer Family identity-based policy examples. For VPC endpoint-specific IAM policies, see Limiting VPC endpoint access for Transfer Family servers.

AWS Transfer Family resource-based policies

Resource-based policies are JSON policy documents that specify what actions a specified principal can perform on the AWS Transfer Family resource and under what conditions. Amazon S3 supports resource-based permissions policies for Amazon S3 *buckets*. Resource-based policies let you grant

usage permission to other accounts on a per-resource basis. You can also use a resource-based policy to allow an AWS service to access your Amazon S3 *buckets*.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, you must also grant the principal entity permission to access the resource. Grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *AWS Identity and Access Management User Guide*.

The Amazon S3 service supports only one type of resource-based policy called a *bucket policy*, which is attached to a *bucket*. This policy defines which principal entities (accounts, users, roles, and federated users) can perform actions on the object.

Examples

To view examples of AWS Transfer Family resource-based policies, see AWS Transfer Family tag-based policy examples.

Authorization based on AWS Transfer Family tags

You can attach tags to AWS Transfer Family resources or pass tags in a request to AWS Transfer Family. To control access based on tags, you provide tag information in the condition element of a policy using the `transfer:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, or `aws:TagKeys` condition keys. For information about how to use tags to control access to AWS Transfer Family resources, see AWS Transfer Family tag-based policy examples.

AWS Transfer Family IAM roles

An IAM role is an entity within your AWS account that has specific permissions.

Using temporary credentials with AWS Transfer Family

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

AWS Transfer Family supports using temporary credentials.

AWS Transfer Family identity-based policy examples

By default, IAM users and roles don't have permission to create or modify AWS Transfer Family resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies on the JSON tab in the *AWS Identity and Access Management User Guide*.

Topics

- Policy best practices
- Using the AWS Transfer Family console
- Allow users to view their own permissions

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Transfer Family resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to

service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the AWS Transfer Family console

To access the AWS Transfer Family console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS Transfer Family resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy. For more information, see Adding permissions to a user in the *AWS Identity and Access Management User Guide*.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Transfer Family tag-based policy examples

The following are examples of how to control access to AWS Transfer Family resources based on tags.

Using tags to control access to AWS Transfer Family resources

Conditions in IAM policies are part of the syntax that you use to specify permissions to AWS Transfer Family resources. You can control access to AWS Transfer Family resources (such as users, servers, roles, and other entities) based on tags on those resources. Tags are key-value pairs. For more information about tagging resources, see Tagging AWS resources in the *AWS General Reference*.

In AWS Transfer Family, resources can have tags, and some actions can include tags. When you create an IAM policy, you can use tag condition keys to control the following:

- Which users can perform actions on an AWS Transfer Family resource, based on tags that the resource has.
- What tags can be passed in an action's request.
- Whether specific tag keys can be used in a request.

By using tag-based access control, you can apply finer control than at the API level. You also can apply more dynamic control than by using resource-based access control. You can create IAM policies that allow or deny an operation based on tags provided in the request (request tags). You can also create IAM policies based on tags on the resource that is being operated on (resource tags). In general, resource tags are for tags that are already on resources, request tags are for when you're adding tags to or removing tags from a resource.

For the complete syntax and semantics of tag condition keys, see Controlling access to AWS resources using resource tags in the *IAM User Guide*. For details about specifying IAM policies with API Gateway, see Control access to an API with IAM permissions in the *API Gateway Developer Guide*.

Example 1: Deny actions based on resource tags

You can deny an action to be performed on a resource based on tags. The following example policy denies `TagResource`, `UntagResource`, `StartServer`, `StopServer`, `DescribeServer`, and `DescribeUser` operations if the user or server resource is tagged with the key `stage` and the value `prod`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "transfer:TagResource",
                "transfer:UntagResource",
                "transfer:StartServer",
                "transfer:StopServer",
                "transfer:DescribeServer",
```

```
                "transfer:DescribeUser"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "prod"
                }
            }
        }
    ]
}
```

Example 2: Allow actions based on resource tags

You can allow an action to be performed on a resource based on tags. The following example policy allows `TagResource`, `UntagResource`, `StartServer`, `StopServer`, `DescribeServer`, and `DescribeUser` operations if the user or server resource is tagged with the key `stage` and the value `prod`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "transfer:TagResource",
                "transfer:UntagResource",
                "transfer:StartServer",
                "transfer:StopServer",
                "transfer:DescribeServer",
                "transfer:DescribeUser"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "prod"
                }
            }
        }
    ]
}
```

Example 3: Deny creation of a user or server based on request tags

The following example policy contains two statements. The first statement denies the `CreateServer` operation on all resources if the cost center key for the tag doesn't have a value.

The second statement denies the `CreateServer` operation if the cost center key for the tag contains any other value besides 1, 2 or 3.

> **Note**
>
> This policy does allow creating or deleting a resource that contains a key called `costcenter` and a value of 1, 2, or 3.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "transfer:CreateServer"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Null":  {
                    "aws:RequestTag/costcenter": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": "transfer:CreateServer",
            "Resource": [
                "*"
            ],
            "Condition": {
                "ForAnyValue:StringNotEquals": {
                    "aws:RequestTag/costcenter": [
                        "1",
```

```
                    "2",
                    "3"
                ]
            }
        }
    ]
}
```

Troubleshooting AWS Transfer Family identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Transfer Family and IAM.

Topics

- I am not authorized to perform an action in AWS Transfer Family
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my AWS Transfer Family resources

I am not authorized to perform an action in AWS Transfer Family

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your sign-in credentials.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a *`widget`* but does not have `transfer:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 transfer:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `transfer;:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS Transfer Family.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS Transfer Family. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

The following example policy contains the permission to pass a role to AWS Transfer Family. Replace **123456789012** with your AWS account ID and **MyTransferRole** with your actual IAM role name.

```
{
   "Version":"2012-10-17",
   "Statement": [
      { "Action": "iam:PassRole",
        "Resource": "arn:aws:iam::123456789012:role/MyTransferRole",
        "Effect": "Allow"
      }
   ]
}
```

I want to allow people outside of my AWS account to access my AWS Transfer Family resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Transfer Family supports these features, see How AWS Transfer Family works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

IAM condition keys for organizational governance

AWS Transfer Family provides IAM condition keys that allow you to restrict resource configurations in any IAM policy. These condition keys can be used in identity-based policies attached to users or roles, or Service Control Policies (SCPs) for organizational governance.

Service Control Policies are IAM policies that apply to an entire AWS organization, providing preventative guardrails across multiple accounts. When used in SCPs, these condition keys help enforce security and compliance requirements organization-wide.

See also

- Actions, resources, and condition keys for Transfer Family
- Service control policies (SCPs)
- View this video that describes how to enforce preventive guardrails using service control policies

Available condition keys

AWS Transfer Family supports the following condition keys for use in IAM policies:

`transfer:RequestServerEndpointType`

Restricts server creation and updates based on endpoint type (PUBLIC, VPC, VPC_ENDPOINT). Commonly used to prevent public-facing endpoints.

`transfer:RequestServerProtocols`

Restricts server creation and updates based on supported protocols (SFTP, FTPS, FTP, AS2).

`transfer:RequestServerDomain`

Restricts server creation based on domain type (S3, EFS).

`transfer:RequestConnectorProtocol`

Restricts connector creation based on protocol (AS2, SFTP).

Supported actions

The condition keys can be applied to the following AWS Transfer Family actions:

- `CreateServer`: Supports `RequestServerEndpointType`, `RequestServerProtocols`, and `RequestServerDomain` condition keys
- `UpdateServer`: Supports `RequestServerEndpointType` and `RequestServerProtocols` condition keys
- `CreateConnector`: Supports `RequestConnectorProtocol` condition key

Example SCP policy

The following example SCP prevents the creation of public AWS Transfer Family servers across your organization:

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "DenyPublicTransferServers",
        "Effect": "Deny",
        "Action": ["transfer:CreateServer", "transfer:UpdateServer"],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "transfer:RequestServerEndpointType": "PUBLIC"
            }
        }
    }]
}
```

Compliance validation for AWS Transfer Family

Third-party auditors assess the security and compliance of AWS Transfer Family as part of multiple AWS compliance programs. These include SOC, PCI, HIPAA, and others. For the complete list, see AWS Services in Scope by Compliance Program.

For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. 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 Transfer Family is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and compliance quick start guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA security and compliance whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS compliance resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub CSPM – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS Transfer Family

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 Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

AWS Transfer Family supports up to 3 Availability Zones and is backed by an auto scaling, redundant fleet for your connection and transfer requests.

For all Transfer Family endpoints:

- Availability Zone-level redundancy is built into the service.
- There are redundant fleets for each AZ.
- This redundancy is provided automatically.

> **Note**
>
> For endpoints in a Virtual Private Cloud (VPC), it is possible to provide a single subnet. However, we recommend that you create endpoints in multiple availability zones within your VPC, to reduce the risk of service disruptions during Availability Zone outages.

See also

- For details on how to create Transfer Family servers in a VPC, see Create a server in a virtual private cloud.
- For more information about AWS Regions and Availability Zones, see AWS global infrastructure.
- For an example on how to build for higher redundancy and minimize network latency by using Latency-based routing, see the blog post Minimize network latency with your AWS Transfer Family servers.

Create a private connection between a VPC and AWS Transfer Family APIs

You can establish a private connection between your VPC and AWS Transfer Family APIs by creating an *interface VPC endpoint*, powered by AWS PrivateLink. You can access AWS Transfer Family APIs as if it were in your VPC, without using an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with AWS Transfer Family APIs.

We create an endpoint network interface in each subnet that you enable for the interface endpoint. For more information, see Access AWS services through AWS PrivateLink in the *AWS*

PrivateLink Guide. Before you set up an interface VPC endpoint for AWS Transfer Family APIs, review Considerations in the *AWS PrivateLink Guide*.

Controlling access using VPC endpoint policies

By default, full access to AWS Transfer Family APIs is allowed through the endpoint. You can control access to the interface endpoint using VPC endpoint policies. You can attach an endpoint policy to your VPC endpoint that controls access to AWS Transfer Family APIs. The policy specifies the following information:

- The **principal** that can perform actions.
- The **actions** that can be performed.
- The **resources** on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

The following is an example of an endpoint policy for AWS Transfer Family APIs. When attached to an endpoint, this policy grants access to all AWS Transfer Family APIs actions on all resources, except those that are tagged with key `Environment` and value `Test`.

```
{
    "Statement": [{
        "Effect": "Deny",
        "Action": "transfer:StartFileTransfer",
        "Principal": "*",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/Environment": "Test"
            }
        }
    }, {
        "Effect": "Allow",
        "Action": "transfer:*",
        "Principal": "*",
        "Resource": "*"
    }]
}
```

Create an interface VPC endpoint for AWS Transfer Family APIs

You can create a VPC endpoint for AWS Transfer Family APIs 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 a VPC endpoint for AWS Transfer Family APIs using one of the following service names:

- `com.amazonaws.region.transfer`
- `com.amazonaws.region.transfer-fips` — To create an interface VPC endpoint that complies with the Federal Information Processing Standard (FIPS) Publication 140-3 US government standard.

If you enable private DNS for the endpoint, you can make API requests to AWS Transfer Family APIs using its default DNS name for the Region, for example, `transfer.us-east-1.amazonaws.com`.

Infrastructure security in AWS Transfer Family

As a managed service, AWS Transfer Family 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 AWS Transfer Family 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.

Avoid placing NLBs and NATs in front of AWS Transfer Family servers

> **Note**
>
> Servers configured with FTP and FTPS protocols only allow a configuration with a VPC: there is no public endpoint available for FTP/FTPS.

Many customers configure a Network Load Balancer (NLB) to route traffic to their AWS Transfer Family server. They typically do this either because they created their server before AWS offered a way to access it from both inside their VPC and from the internet, or to support FTP on the internet. This configuration not only increases costs for customers, but can also cause other issues, which we describe in this section.

For SFTP servers that use a VPC-hosted endpoint, we still recommend avoiding an NLB where possible. If your architecture requires an NLB—for example, to expose SFTP on a custom listener port that AWS Transfer Family does not offer—you can preserve the client's source IP address by enabling PROXY protocol v2 (PPv2). For more information, see Working with Network Load Balancers.

NAT gateways are a mandatory component when clients are connecting from a customer private network behind a corporate firewall. However, you should be aware that when many clients are behind the same NAT gateway, this can impact performance and connection limits. If there's an NLB or NAT in the communication path from the client to the FTP or FTPS server, the server can't accurately recognize the client's IP address, because AWS Transfer Family sees only the IP address of the NLB or NAT.

If you're using the configuration of a Transfer Family server behind an NLB, we recommend that you move to a VPC endpoint and use an Elastic IP address instead of using an NLB. When using NAT gateways, be aware of the connection limitations described below.

If you're using the FTPS protocol, this configuration not only reduces your ability to audit who's accessing your server, but it can also impact performance. AWS Transfer Family uses the source IP address to shard your connections across our data plane. For FTPS, this means that instead of having 10,000 simultaneous connections, Transfer Family servers with NLB or NAT gateways on the communication route are limited to only 300 simultaneous connections.

Although we recommend avoiding Network Load Balancers in front of AWS Transfer Family servers, if your FTP or FTPS implementation requires an NLB or NAT in the communication route from the client, follow these recommendations:

- For an NLB, use port 21 for health checks, instead of ports 8192-8200.
- For the AWS Transfer Family server, enable TLS session resumption by setting `TlsSessionResumptionMode = ENFORCED`.

> **Note**
>
> This is the recommended mode, as it provides enhanced security:
>
> - Requires clients to use TLS session resumption for subsequent connections.
> - Provides stronger security guarantees by ensuring consistent encryption parameters.
> - Helps prevent potential downgrade attacks.
> - Maintains compliance with security standards while optimizing performance.

- If possible, migrate away from using an NLB to take full advantage of AWS Transfer Family performance and connection limits.

For additional guidance on NLB alternatives, contact the AWS Transfer Family Product Management team through AWS Support. For more information about improving your security posture, see the blog post Six tips to improve the security of your AWS Transfer Family server.

VPC connectivity infrastructure security

SFTP connectors with VPC egress type provide enhanced infrastructure security through network isolation and private connectivity:

Network isolation benefits

- **Private network traffic**: All connector traffic to private SFTP servers remains within your VPC, never traversing the public internet.
- **Controlled egress**: For public endpoints accessed via VPC, traffic routes through your NAT gateways, giving you control over egress IP addresses and network policies.
- **VPC security controls**: Leverage existing VPC security groups, network ACLs, and route tables to control connector network access.
- **Hybrid connectivity**: Access on-premises SFTP servers through established VPN or Direct Connect connections without additional internet exposure.

Resource Gateway security considerations

Resource Gateways provide secure ingress points for Cross-VPC Resource Access:

- **Multi-AZ deployment**: Resource Gateways require subnets in at least two Availability Zones for high availability and fault tolerance.
- **Security group controls**: Configure security groups to restrict access to SFTP ports (typically port 22) from authorized sources only.
- **Private subnet placement**: Deploy Resource Gateways in private subnets when connecting to private SFTP servers to maintain network isolation.
- **Connection limits**: Each Resource Gateway supports up to 350 concurrent connections with a 350-second idle timeout for TCP connections.

Add a web application firewall

AWS WAF is a web application firewall that helps protect web applications and APIs from attacks. You can use it to configure a set of rules known as a *web access control list* (web ACL) that allow, block, or count web requests based on customizable web security rules and conditions that you define. For more information, see Using AWS WAF to protect your APIs.

To add AWS WAF

1. Open the API Gateway console at https://console.aws.amazon.com/apigateway/.
2. In the **APIs** navigation pane, and then choose your custom identity provider template.
3. Choose **Stages**.
4. In the **Stages** pane, choose the name of the stage.
5. In the **Stage Editor** pane, choose the **Settings** tab.
6. Do one of the following:
   - Under **Web Application Firewall (WAF)**, for **Web ACL**, choose the web ACL that you want to associate with this stage.
   - If the web ACL you need doesn't exist, you will need to create one by doing the following:
     1. Choose **Create Web ACL**.
     2. On the AWS WAF service homepage, choose **Create web ACL**.
     3. In **Web ACL details**, for **Name**, type the name of the web ACL.
     4. In **Rules**, choose **Add rules**, then choose **Add my own rules and rule groups**.
     5. For **Rule type**, choose IP set to identify a specific list of IP addresses.
     6. For **Rule**, enter the name of the rule.

7. For **IP set**, choose an existing IP set. To create an IP set, see Creating an IP set.
8. For **IP address to use as the originating address**, choose **IP address in header**.
9. For **Header field name**, enter `SourceIP`.
10. For **Position inside header**, choose **First IP address**.
11. For **Fallback for missing IP address**, choose **Match** or **No Match** depending on how you want to handle an invalid (or missing) IP address in the header.
12. For **Action**, choose the action of the IP set.
13. For **Default web ACL action for requests that don't match any rules**, choose **Allow** or **Block** and then click **Next**.
14. For steps 4 and 5, choose **Next**.
15. In **Review and create**, review your choices, and then choose **Create web ACL**.

7. Choose **Save Changes**.
8. Choose **Resources**.
9. For **Actions**, choose **Deploy API**.

For information on how secure Transfer Family with AWS web application firewall, see Securing Transfer Family with AWS application firewall and Amazon API Gateway in the AWS storage blog.

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 that it should not 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. For a detailed description of this problem, see the confused deputy problem in the *IAM User Guide*.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that AWS Transfer Family has for 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 exact Amazon Resource Name (ARN) of the resource you want to allow. If you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcard characters (*) for the unknown portions of the ARN. For example, `arn:aws:transfer::region::account-id:server/*`.

AWS Transfer Family uses the following types of roles:

- **User role** – Allows service-managed users to access the necessary Transfer Family resources. AWS Transfer Family assumes this role in the context of a Transfer Family user ARN.
- **Access role** – Provides access to only the Amazon S3 files that are being transferred. For inbound AS2 transfers, the access role uses the Amazon Resource Name (ARN) for the agreement. For outbound AS2 transfers, the access role uses the ARN for the connector.
- **Invocation role** – For use with Amazon API Gateway as the server's custom identity provider. Transfer Family assumes this role in the context of a Transfer Family server ARN.
- **Logging role** – Used to log entries into Amazon CloudWatch. Transfer Family uses this role to log success and failure details along with information about file transfers. Transfer Family assumes this role in the context of a Transfer Family server ARN. For outbound AS2 transfers, the logging role uses the connector ARN.
- **Execution role** – Allows a Transfer Family user to call and launch workflows. Transfer Family assumes this role in the context of a Transfer Family workflow ARN.

For more information, see Policies and permissions in IAM in the *IAM User Guide*.

> **Note**
>
> In the following examples, replace each *`user input placeholder`* with your own information.

> **Note**
>
> In our examples, we use both `ArnLike` and `ArnEquals`. They are functionally identical, and therefore you may use either when you construct your policies. Transfer Family documentation uses `ArnLike` when the condition contains a wildcard character, and `ArnEquals` to indicate an exact match condition.

AWS Transfer Family user role cross-service confused deputy prevention

The following example policy allows any user of any server in the account to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transfer:us-east-1:123456789012:user/
*"
                }
            }
        }
    ]
}
```

The following example policy allows any user of a specific server to assume the role.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "transfer.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
```

```
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:transfer:us-east-1:123456789012:user/server-id/
*"
        }
      }
    }
  ]
}
```

The following example policy allows a specific user of a specific server to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transfer:us-
east-1:123456789012:user/server-id/user-name"
                }
            }
        }
    ]
}
```

AWS Transfer Family workflow role cross-service confused deputy prevention

The following example policy allows any workflow in the account to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transfer:us-
west-2:111122223333:workflow/*"
                }
            }
        }
    ]
}
```

The following example policy allows a specific workflow to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transfer:us-
west-2:111122223333:workflow/workflow-id"
                }
            }
```

```
        }
    ]
}
```

AWS Transfer Family connector role cross-service confused deputy prevention

The following example policy allows any connector in the account to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transfer:us-
east-1:123456789012:connector/*"
                }
            }
        }
    ]
}
```

The following example policy allows a specific connector to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transfer:us-
east-1:123456789012:connector/connector-id"
                }
            }
        }
    ]
}
```

AWS Transfer Family logging and invocation role cross-service confused deputy prevention

> **Note**
>
> The following examples can be used in both logging and invocation roles.
> In these examples, you can remove the ARN details for a workflow if your server doesn't have any workflows attached to it.

The following example logging/invocation policy allows any server (and workflow) in the account to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllServersWithWorkflowAttached",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                   "aws:SourceArn": [
                     "arn:aws:transfer:us-west-2:111122223333:server/*",
                     "arn:aws:transfer:us-west-2:111122223333:workflow/*"
                   ]
                }
            }
        }
    ]
}
```

The following example logging/invocation policy allows a specific server (and workflow) to assume the role.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSpecificServerWithWorkflowAttached",
            "Effect": "Allow",
            "Principal": {
                "Service": "transfer.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                   "aws:SourceArn": [
                       "arn:aws:transfer:us-west-2:111122223333:server/server-id",
                       "arn:aws:transfer:us-west-2:111122223333:workflow/workflow-
id"
                   ]
                }
            }
        }
    ]
```

```
}
```

AWS managed policies for AWS Transfer Family

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create AWS Identity and Access Management (IAM) customer managed policies that provide your team with only the permissions that they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*. For a detailed listing of all AWS managed policies, see the AWS managed policy reference guide.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the `ReadOnlyAccess` AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS managed policy: AWSTransferConsoleFullAccess

The `AWSTransferConsoleFullAccess` policy provides full access to Transfer Family through the AWS Management Console. For more information, see AWSTransferConsoleFullAccess.

AWS managed policy: AWSTransferFullAccess

The `AWSTransferFullAccess` policy provides full access to Transfer Family services. For more information, see AWSTransferFullAccess.

AWS managed policy: AWSTransferLoggingAccess

The `AWSTransferLoggingAccess` policy grants AWS Transfer Family full access to create log streams and groups and put log events to your account. For more information, see AWSTransferLoggingAccess.

AWS managed policy: AWSTransferReadOnlyAccess

The `AWSTransferReadOnlyAccess` policy provides read-only access to Transfer Family services. For more information, see AWSTransferReadOnlyAccess.

AWS managed policy: AWSTransferServiceRolePolicy

The `AWSTransferServiceRolePolicy` policy is attached to the AWSServiceRoleForTransfer service-linked role and allows Transfer Family to integrate with other AWS services on your behalf. You can't attach this policy to your own IAM entities. For more information, see Using service-linked roles for AWS Transfer Family.

Permissions details

This policy allows Transfer Family to complete the following actions in AWS Secrets Manager:

- Retrieve SFTP host key material from Transfer Family managed secrets to serve to incoming connections.
- List and batch retrieve secrets in the same AWS account to discover the host key secrets that Transfer Family manages.
- Delete orphaned Transfer Family managed secrets that are no longer associated with a Transfer Family host key resource.

For more information, see AWSTransferServiceRolePolicy.

AWS Transfer Family updates to AWS managed policies

View details about updates to AWS managed policies for AWS Transfer Family since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Document history for AWS Transfer Family page.

Change	Description	Date
AWSTransferServiceRolePolicy – New policy	AWS Transfer Family added the `AWSTransferServiceRolePolicy` managed policy, attached to the AWSServiceRoleForTransfer service-linked role, to support storing SFTP host keys in AWS Secrets Manager.	October 6, 2026
Documentation update	Added sections for each of the Transfer Family managed policies.	January 27, 2022
AWSTransferReadOnlyAccess – Update to an existing policy	AWS Transfer Family added new permissions to allow the policy to read AWS Managed Microsoft AD.	September 30, 2021
AWS Transfer Family started tracking changes	AWS Transfer Family started tracking changes for its AWS managed policies.	June 15, 2021

Using service-linked roles for AWS Transfer Family

AWS Transfer Family uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Transfer Family. Service-linked roles are predefined by Transfer Family and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Transfer Family easier because you don't have to manually add the necessary permissions. Transfer Family defines the permissions of its service-linked roles, and unless defined otherwise, only Transfer Family can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy can't be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your Transfer Family resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-linked role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for AWS Transfer Family

AWS Transfer Family uses the AWSServiceRoleForTransfer service-linked role. It allows Transfer Family to integrate with other AWS services on your behalf. The `transfer.amazonaws.com` service can assume this role.

The `AWSTransferServiceRolePolicy` role permissions policy allows Transfer Family to complete the following actions on the specified resources:

- `secretsmanager:BatchGetSecretValue` and `secretsmanager:ListSecrets` on secrets in the same AWS account as the calling principal – Lets Transfer Family discover and read the host key secrets that it manages in AWS Secrets Manager.
- `secretsmanager:DescribeSecret` and `secretsmanager:GetSecretValue` on Transfer Family managed secrets (secrets with an ARN that matches `arn:*:secretsmanager:*:*:secret:transfer!*` and the `aws:secretsmanager:owningService` tag value `transfer`) – Lets Transfer Family retrieve SFTP host key material to serve to incoming connections.
- `secretsmanager:DeleteSecret` on Transfer Family managed secrets (the same `transfer!` ARN and tag scope) – Lets Transfer Family clean up orphaned secrets that are no longer associated with a Transfer Family host key resource.

These are managed secrets: only Transfer Family can create them, and their names include the `transfer!` prefix that identifies Transfer Family as the managing service. For more information, see AWS Secrets Manager secrets managed by other AWS services.

You must configure permissions to allow your users, groups, or roles to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for AWS Transfer Family

You don't need to manually create a service-linked role. Transfer Family creates the service-linked role for you when you create a server, import a host key, or update a host key that uses AWS Secrets Manager storage. This applies whether you use the AWS Management Console, the AWS CLI, or the AWS API.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create a server, import a host key, or update a host key that uses AWS Secrets Manager storage, Transfer Family creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **Transfer** use case. In the IAM CLI or the IAM API, create a service-linked role with the `transfer.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*.

Editing a service-linked role for AWS Transfer Family

AWS Transfer Family does not allow you to edit the AWSServiceRoleForTransfer service-linked role. After you create a service-linked role, you can't change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for AWS Transfer Family

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

To delete the AWSServiceRoleForTransfer service-linked role

1. Delete the host keys that use AWS Secrets Manager storage from your Transfer Family servers, or delete the servers that use them. Removing these resources also removes the AWS Secrets Manager secrets that the service-linked role manages. For more information, see Deleting a server host key.
2. Use the IAM console, the IAM CLI, or the IAM API to delete the AWSServiceRoleForTransfer service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for AWS Transfer Family service-linked roles

AWS Transfer Family supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see AWS Transfer Family endpoints and quotas.

Transfer Family Terraform modules

HashiCorp Terraform is an open-source Infrastructure as Code (IaC) engine developed using the HashiCorp Configuration Language (HCL). Terraform provides a consistent command line interface (CLI) workflow that, in conjunction with AWS Transfer Family for the back-end infrastructure, can manage hundreds of cloud services and codify cloud APIs into declarative configuration files.

You can use Terraform to safely deploy AWS Transfer Family SFTP servers and SFTP connectors, along with associated dependencies and customizations. For the repository that contains Terraform code to create the resources required to run AWS Transfer Family, see the Terraform Transfer Family module source code on GitHub.

> **Note**
>
> The AWS Transfer Family modules for Terraform are a community supported effort. They are not part of an AWS service. Best-effort support is provided by the AWS Storage community.

SFTP servers

This automation provides you with a customizable Terraform module and end-to-end examples to create an SFTP endpoint (`PUBLIC` or `VPC` endpoint types), integrate with Amazon CloudWatch for logging and monitoring, manage user identities for endpoint access, and configure IAM roles for access to Amazon S3 buckets where files are stored. The module supports multiple SSH public keys per user (up to 50 keys) for enhanced security and key rotation capabilities.

SFTP connectors

AWS Transfer Family Terraform module now supports deployment of SFTP connectors to transfer files between Amazon S3 and remote SFTP servers. SFTP connectors provide a fully managed and low-code capability to copy files between Amazon S3 and remote SFTP servers.

You can now use Terraform to programmatically provision your SFTP connectors, associated dependencies and customizations in a single deployment. The module also provides end-to-end examples to automate file transfer workflows based on a schedule or event triggers. Using

Terraform for deployment eliminates the need for time-consuming and error-prone manual configurations, and provides you a fast, repeatable and secure deployment option that can scale.

AS2

To show support for AS2 Terraform templates, add a thumbs up reaction (#) to the Transfer Family Terraform templates feature request. You can also add a comment describing your use case.

B2B Data Interchange

AWS B2B Data Interchange automates the transformation, validation, and generation of Electronic Data Interchange (EDI) documents to and from JSON and XML data formats. To show support for Terraform templates for B2B Data Interchange, add a thumbs up reaction (#) to the feature request. You can also add a comment describing your use case.

Troubleshooting AWS Transfer Family

This chapter provides troubleshooting information for common issues you might encounter when using AWS Transfer Family. Each section focuses on a specific area of functionality to help you quickly find solutions to your problems.

- For issues with IAM in Transfer Family, see Troubleshooting AWS Transfer Family identity and access.
- For issues with your Transfer Family web apps, see Troubleshooting your web apps.

Topics

- Troubleshoot authentication issues
- Troubleshoot SFTP connector issues
- Troubleshoot SFTP connectivity and transfer issues
- Troubleshoot custom identity provider issues
- Troubleshoot workflow issues
- Troubleshoot EFS issues
- Troubleshoot storage and encryption issues
- Troubleshoot monitoring and alerting issues
- Troubleshoot cross-region transfer issues
- Troubleshoot Terraform deployment issues
- Troubleshoot Web Application Firewall integration issues
- Troubleshoot service-managed user issues
- Troubleshoot AS2 issues

Troubleshoot authentication issues

This section describes possible solutions for the following authentication issues.

Topics

- Authentication failures—SSH/SFTP
- Managed AD mismatched realms issue

- Active Directory group limits exceeded
- Miscellaneous authentication issues
- Troubleshoot Amazon API Gateway issues
- Troubleshoot testing your identity provider
- Duplicate Amazon S3 buckets in web app

Authentication failures—SSH/SFTP

Description

When you try to connect to your server using Secure Shell (SSH) File Transfer Protocol (SFTP), you receive a message similar to the following:

```
Received disconnect from 3.130.115.105 port 22:2: Too many authentication failures
  Authentication failed.
```

> **Note**
>
> If you are using an API Gateway and receive this error, see Too many authentication failures.

Cause

You have not added an RSA key pair for your user, so you must authenticate using a password instead.

Solution

When you run the `sftp` command, specify the `-o PubkeyAuthentication=no` option. This option forces the system to request your password. For example:

```
sftp -o PubkeyAuthentication=no sftp-user@server-id.server.transfer.region-
id.amazonaws.com
```

Managed AD mismatched realms issue

Description

A user's realm and their group realm must match. They must both be in the default realm, or they must both be in the trusted realm.

Cause

If a user and their group do not match, the user cannot be authenticated by Transfer Family. If you test the identity provider for the user, you receive the error No associated access found for user's groups.

Solution

Reference a group in the user's realm that matches the group realm (either default or trusted).

Active Directory group limits exceeded

Description

When attempting to add more Active Directory groups to your AWS Transfer Family server, you receive an error indicating you've reached the maximum number of groups allowed.

Cause

AWS Transfer Family has a default limit of 100 Active Directory groups per server.

Solution

These are two possible solutions:

- Consolidate your Active Directory groups to reduce the total number needed.
- If your use case requires more than 100 groups, consider using a custom identity provider solution as described in Simplify Active Directory authentication with a custom identity provider for AWS Transfer Family.

Miscellaneous authentication issues

Description

You receive an authentication error and none of the other troubleshooting works

Cause

You might have specified a target for a logical directory that contains a leading or trailing slash (/).

Solution

Update your logical directory target, to make sure it begins with a slash, and does not contain a trailing slash. For example, `/amzn-s3-demo-bucket/images` is acceptable, but `amzn-s3-demo-bucket/images` and `/amzn-s3-demo-bucket/images/` are not.

Troubleshoot Amazon API Gateway issues

This section describes possible solutions for the following API Gateway issues.

Topics

- Too many authentication failures
- Connection closed

Too many authentication failures

Description

When you try to connect to your server using Secure Shell (SSH) File Transfer Protocol (SFTP), you get the following error:

```
Received disconnect from 3.15.127.197 port 22:2: Too many authentication failures
Authentication failed.
Couldn't read packet: Connection reset by peer
```

Cause

You might have entered an incorrect password for your user. Try again to enter the correct password.

If the password is correct, the issue might be caused by a role Amazon Resource Name (ARN) that is not valid. To confirm that this is the issue, test the identity provider for your server. If you see a response similar to the following, the role ARN is a placeholder only, as indicated by the role ID value of all zeros:

```
{
    "Response": "{\"Role\": \"arn:aws:iam::000000000000:role/MyUserS3AccessRole\",
\"HomeDirectory\": \"/\"}",
```

```
    "StatusCode": 200,
    "Message": "",
    "Url": "https://api-gateway-ID.execute-api.us-east-1.amazonaws.com/prod/
servers/transfer-server-ID/users/myuser/config"
}
```

Solution

Replace the placeholder role ARN with an actual role that has permission to access the server.

To update the role

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the left navigation pane, choose **Stacks**.
3. In the **Stacks** list, choose your stack, and then choose the **Parameters** tab.
4. Choose **Update**. On the **Update stack** page, choose **Use current template**, and then choose **Next**.
5. Replace **UserRoleArn** with a role ARN that has sufficient permissions for accessing your Transfer Family server.

   > **Note**
   >
   > To grant the necessary permissions, you can add the `AmazonAPIGatewayAdministrator` and the `AmazonS3FullAccess` managed policies to your role.

6. Choose **Next**, and then choose **Next** again. On the **Review *stack*** page, select **I acknowledge that AWS CloudFormation might create IAM resources**, and then choose **Update stack**.

Connection closed

Description

When you try to connect to your server using Secure Shell (SSH) File Transfer Protocol (SFTP), you get the following error:

```
Connection closed
```

Cause

One possible cause for this issue is that your Amazon CloudWatch logging role does not have a trust relationship with Transfer Family.

Solution

Make sure that the logging role for the server has a trust relationship with Transfer Family. For more information, see To establish a trust relationship.

Troubleshoot testing your identity provider

Description

If you test your identity provider using the console or the `TestIdentityProvider` API operation, the `Response` field is empty. For example:

```
{
    "Response": "{}",
    "StatusCode": 200,
    "Message": ""
}
```

Cause

The most likely cause is that the authentication failed because of an incorrect user name or password.

Solution

Make sure that you are using the correct credentials for your user, and make updates to the username or password, if necessary.

Duplicate Amazon S3 buckets in web app

Description

The same Amazon S3 bucket appears multiple times in the Transfer Family web app interface.

Cause

This occurs when a user belongs to multiple Active Directory groups that have grants to the same Amazon S3 bucket. The web app lists all top-level grants associated with the user's UID or GID, including duplicate grants to the same bucket location.

Solution

To prevent duplicate listings, consolidate grants so each user has only one grant per Amazon S3 location. Review your Amazon S3 Access Grants configuration and remove redundant grants for the same bucket across different Active Directory groups.

Troubleshoot SFTP connector issues

This section describes possible solutions for issues with SFTP connectors.

Topics

- Troubleshoot adding trusted host keys for your SFTP connector
- Key negotiation fails
- SFTP connector throttling
- Optimizing SFTP connector performance
- Troubleshoot VPC connectivity issues
- Miscellaneous SFTP connector issues

Troubleshoot adding trusted host keys for your SFTP connector

Description

When you are creating or editing an SFTP connector, and you are adding a trusted host key, you receive the following error: `Failed to edit connector details (Invalid host key format.)`

Cause

If you paste in a correct public key, the issue might be that you included the `comment` portion of the key. AWS Transfer Family does not currently accept the comment portion of the key.

Solution

Delete the comment portion of the key, when you paste it into the text field. For example, assume your key looks similar to the following:

```
ssh-rsa AAAA...== marymajor@dev-dsk-marymajor-1d-c1234567.us-east-1.amazon.com
```

Remove the text that follows the == characters and only paste in the portion of the key up to and including the ==.

```
ssh-rsa AAAA...==
```

Key negotiation fails

Description

You receive an error where the key exchange negotiation fails. For example:

```
Key exchange negotiation failed due to incompatible host key algorithms.
Client offered: [ecdsa-sha2-nistp256, ecdsa-sha2-nistp384,
ecdsa-sha2-nistp521, rsa-sha2-512, rsa-sha2-256] Server offered: [ssh-rsa]
```

Cause

This error is because there's no overlap between the host key algorithms supported by the server and those supported by the connector.

Solution

Ensure that the remote server supports at least one of the Client host key algorithms listed in the error message. For the list of supported algorithms, see Security policies for AWS Transfer Family SFTP connectors.

SFTP connector throttling

Description

When using SFTP connectors for file transfers, you encounter errors such as:

```
{"type":"ExecutionThrottled","details":{},"connectorId":"c-1234567890abcdef0"}
```

Or you notice that file transfers are being delayed or failing intermittently during high-volume operations.

Cause

SFTP connectors have service quotas that limit the number of concurrent file transfers and API operations. When these limits are exceeded, throttling occurs to protect the service and ensure fair usage across all customers.

Solution

To address SFTP connector issues, try the following solutions:

1. Implement exponential backoff and retry logic in your applications. For example, create a function that automatically retries failed operations with increasing wait times between attempts.
2. Implement rate limiting in your application:
   - Limit the number of concurrent transfers.
   - Add delays between batches of transfers.
3. Monitor your usage against service quotas:
   - Use CloudWatch metrics to track API usage.
   - Set up alarms to notify you when approaching quota limits.
4. For options to scale your SFTP connectors, see Scaling your SFTP connectors.
5. If throttling persists and impacts your business operations, request a quota increase through the Service Quotas console.

Optimizing SFTP connector performance

Description

Your SFTP connector transfers are slower than expected or you experience inconsistent performance.

Cause

SFTP connector performance can be affected by various factors including network conditions, file sizes, remote server configuration, and concurrent transfer limits.

Solution

To optimize SFTP connector performance:

- Configure your remote SFTP server for optimal performance:
  - Increase maximum sessions and transfers per session
  - Optimize TCP window sizes for high-latency connections
  - Use compression if supported by both ends
- Consider network optimization, by placing your Transfer Family connector in a region close to your remote SFTP server.
- Implement a monitoring strategy to identify performance bottlenecks:
  - Monitor network throughput and latency
  - Analyze logs for patterns in slow transfers

Troubleshoot VPC connectivity issues

This section describes solutions for common issues with VPC_LATTICE-enabled SFTP connectors.

Connector stuck in PENDING status

Description

Your VPC_LATTICE-enabled SFTP connector remains in `PENDING` status for an extended period (more than 10 minutes).

Cause

This can occur due to DNS resolution delays, Resource Gateway configuration issues, or VPC Lattice service network association problems.

Solution

Contact AWS Support at Contact AWS.to help analyze the root cause of the issue. You can also try the following work-around.

1. Verify that your Resource Gateway is in `ACTIVE` status:

```
aws vpc-lattice get-resource-gateway --resource-gateway-
identifier rgw-1234567890abcdef0
```

2. Check that your Resource Configuration is properly configured and active:

```
aws vpc-lattice get-resource-configuration --resource-configuration-
identifier rcfg-1234567890abcdef0
```

3. Ensure your Resource Gateway has subnets in at least two Availability Zones that support VPC Lattice.
4. If the issue persists, delete and recreate the connector with the same configuration.

Connector in ERRORED status

Description

Your VPC_LATTICE-enabled SFTP connector shows ERR0RED status with error details.

Cause

Common causes include invalid Resource Configuration ARN, insufficient IP addresses in VPC subnets, or cross-region resource sharing attempts.

Solution

1. Check the error details using `describe-connector`:

```
aws transfer describe-connector --connector-id c-1234567890abcdef0
```

2. Verify the Resource Configuration ARN is correct and in the same region as your connector.
3. Ensure your VPC subnets have sufficient available IP addresses for the Resource Gateway.
4. Check that your Resource Configuration target (IP address or DNS name) is reachable from your VPC.

Public IP address not supported error

Description

You receive an error when trying to create a Resource Configuration with a public IP address: `ValidationException: IP address x.x.x.x is not in allowed ranges.`

Cause

Resource Configurations for public endpoints must use DNS names, not IP addresses.

Solution

Use the public DNS name of your SFTP server instead of its IP address when creating the Resource Configuration:

```
aws vpc-lattice create-resource-configuration \
    --name my-public-server-config \
    --resource-gateway-identifier rgw-1234567890abcdef0 \
    --resource-configuration-definition dnsResource={domainName="my.sftp.server.com"} \
    --port-ranges 22
```

Availability Zone not supported error

Description

You receive an error when creating a Resource Gateway: `Subnet subnet-xxx is not valid because it is not in a supported Availability Zone.`

Cause

Not all Availability Zones support VPC Lattice Cross-VPC Resource Access. The error message lists the supported AZs for your region.

Solution

1. Create subnets in the supported Availability Zones listed in the error message.
2. Update your Resource Gateway to use subnets from supported AZs only.
3. Ensure you have at least two subnets in different supported AZs.

Connection timeouts with VPC_LATTICE connectors

Description

File transfers through VPC connectors time out or fail intermittently.

Cause

VPC Lattice has connection limits (350 connections per resource) and idle timeouts (350 seconds for TCP).

Solution

1. Monitor concurrent connections to stay within the 350 connection limit per resource.
2. Implement connection pooling and reuse in your applications.
3. Configure appropriate timeout values in your SFTP client applications (less than 350 seconds).
4. Consider creating multiple Resource Configurations for the same target to distribute load.

Miscellaneous SFTP connector issues

Description

You receive an error after you run `StartFileTransfer`, but do not know the cause of the issue, and only the connector ID is returned after the API call.

Cause

This error can have several causes. To troubleshoot, we recommend that you test your connector and search your CloudWatch logs.

Solution

- **Test your connector**: See Test an SFTP connector. If the test fails, the system provides an error message based on the reason the test failed. That section describes how to test your connector from either the console or by using the TestConnection API command.
- **View CloudWatch logs for your connector**: See Example log entries for SFTP connectors. This topic provides examples for SFTP connector log entries, and the naming convention to help you find the appropriate logs.

Troubleshoot SFTP connectivity and transfer issues

This section describes possible solutions for SFTP connectivity and file transfer issues.

Topics

- Troubleshoot SFTP connectivity issues
- Troubleshoot SFTP client issues
- Troubleshoot file upload issues
- Troubleshoot VPC egress type SFTP connector issues

Troubleshoot SFTP connectivity issues

Description

Your SFTP client cannot initiate the connection. This issue can happen continuously or intermittently. For example, you might see the following sequence of events in your SFTP client debug logs:

```
sftp -vvv username@1.1.1.1
.................................
debug1: Local version string ...........
kex_exchange_identification: read: Connection reset by peer
Connection reset by 1.1.1.1 port 22
Connection closed.
```

Cause

There is an edge case where the zero-byte TCP ACK (ACK without data), also known as the three-way handshake, is either dropped or delayed.

Solution

As a workaround, Transfer Family offers a solution that uses a different configuration to solve this issue, but may cause compatibility issues with older clients. For that reason, this solution is available only on port 2223.

In the procedure for creating a Transfer Family server in a VPC (Create a server in a virtual private cloud), when you specify a security group, configure SSH traffic to use port 2223.

Troubleshoot SFTP client issues

SFTP client side messages are described in SFTP messages. The best way to troubleshoot SFTP client issues is to check the SFTP client logs and, if necessary, reach out to your network administrator.

Troubleshoot file upload issues

This section describes possible solutions for the following file upload issues.

Topics

- Troubleshoot Amazon S3 file upload errors

- Troubleshoot unreadable file names

Troubleshoot Amazon S3 file upload errors

Description

When you are attempting to upload a file to Amazon S3 storage using Transfer Family, you receive the following error message: AWS Transfer does not support random access writes to S3 objects.

Cause

When you're using Amazon S3 for your server's storage, Transfer Family does not support multiple connections for a single transfer.

Solution

If your Transfer Family server is using Amazon S3 for its storage, disable any options for your client software that mention using multiple connections for a single transfer.

Troubleshoot unreadable file names

Description

You see corrupted file names in some of your uploaded files. Users sometimes encounter problems with FTP and SFTP transfers that garble certain characters in file names, such as umlauts, accented letters, or certain scripts, such as Chinese or Arabic.

Cause

Although the FTP and SFTP protocols can allow for character encoding of files names to be negotiated by clients, Amazon S3 and Amazon EFS do not. Instead, they require UTF-8 character encoding. As a result, certain characters are not rendered correctly.

Solution

To solve this problem, review your client application for file name character encoding and make sure it is set to UTF-8.

Troubleshoot VPC egress type SFTP connector issues

If you're experiencing issues with VPC egress type SFTP connectors, check the following:

Connector status is PENDING

Description

Your VPC egress type connector remains in PENDING status for several minutes after creation, and TestConnection returns "Connector not available".

Cause

DNS resolution for VPC connectors can take several minutes to complete after creation.

Solution

Wait for the connector status to become ACTIVE before attempting file transfers. This is normal behavior for VPC egress type connectors.

Connection timeouts

Description

Your VPC egress type connector times out when attempting to connect to the SFTP server.

Cause

Security groups may not allow traffic on port 22 between your Resource Gateway subnets and the target SFTP server.

Solution

Verify that security groups allow traffic on port 22 between your Resource Gateway subnets and the target SFTP server.

Resource Configuration errors

Description

Your VPC egress type connector fails to connect due to Resource Configuration issues.

Cause

The Resource Configuration may point to an incorrect IP address or DNS name, or the Resource Gateway may not be in the same VPC as your SFTP server (for private endpoints).

Solution

Ensure your Resource Configuration points to the correct IP address or DNS name, and that the Resource Gateway is in the same VPC as your SFTP server (for private endpoints). For more information, see Resource configurations in the *Amazon VPC Lattice User Guide*.

Public endpoint issues

Description

Your VPC egress type connector cannot connect to public SFTP endpoints.

Cause

For public endpoints, you must use a DNS name (not an IP address) in your Resource Configuration, and your VPC must have a NAT Gateway for outbound internet access.

Solution

Ensure you're using a DNS name, not an IP address, in your Resource Configuration. Verify that your VPC has a NAT Gateway for outbound internet access.

Availability Zone issues

Description

You cannot create a Resource Gateway due to Availability Zone limitations.

Cause

Resource Gateways require subnets in at least 2 Availability Zones, and not all AZs support VPC Lattice.

Solution

Check the supported Availability Zones for VPC Lattice in your region and ensure you have subnets in at least 2 supported AZs.

Troubleshoot custom identity provider issues

This section describes possible solutions for issues related to custom identity providers with Transfer Family.

Topics

- Troubleshoot API Gateway integration errors
- Troubleshoot Lambda function timeouts
- Troubleshoot consistent Lambda timeout issues
- Troubleshoot KeyError exceptions

Troubleshoot API Gateway integration errors

Description

Users cannot authenticate with your Transfer Family server, and when testing your identity provider, you see errors such as:

```
{
    "Response": "",
    "StatusCode": 500,
    "Message": "Internal server error"
}
```

Cause

API Gateway integration errors can occur due to:

- Incorrect API Gateway configuration
- Lambda function errors not being properly handled
- Permission issues between API Gateway and Lambda
- Malformed responses from the Lambda function

Solution

To troubleshoot API Gateway integration errors:

1. Check your Lambda function logs for detailed error information:
   - In the CloudWatch console, navigate to **Log groups** > **/aws/lambda/your-function-name**
   - Look for error messages or stack traces that indicate the root cause
2. Verify that your Lambda function returns properly formatted responses:

```
{
    "Role": "arn:aws:iam::123456789012:role/TransferUserRole",
    "HomeDirectory": "/mybucket/home/username"
}
```

3. Enable detailed CloudWatch logging for API Gateway:
   - In the API Gateway console, select your API and choose **Stages**
   - Select your stage and under **Logs/Tracing**, enable **CloudWatch Logs**
   - Set the log level to **ERROR** or **INFO**
4. Test your API Gateway endpoint directly:

```
curl -X POST https://your-api-id.execute-api.region.amazonaws.com/prod/servers/your-server-id/users/username/config \
    -H "Content-Type: application/json" \
    -d '{"Password": "password"}'
```

5. Verify permissions between API Gateway and Lambda:
   - Ensure API Gateway has permission to invoke your Lambda function
   - Check that the execution role for your Lambda function has necessary permissions

Troubleshoot Lambda function timeouts

Description

When users attempt to authenticate with your Transfer Family server using a custom identity provider, they experience long delays followed by authentication failures. In your Lambda logs, you see timeout errors.

Cause

Lambda functions used for custom identity providers have a default timeout of 3 seconds. If your authentication logic takes longer than this timeout (for example, when querying external databases or making API calls to third-party identity providers), the function will time out and authentication will fail.

Solution

To resolve Lambda timeout issues:

1. Increase the Lambda function timeout:
   - In the Lambda console, navigate to your function and select the **Configuration** tab
   - Under **General configuration**, click **Edit**
   - Increase the timeout value (up to 15 seconds is recommended for authentication functions)
2. Optimize your Lambda function code:
   - Use connection pooling for database queries
   - Implement caching for frequently accessed data
   - Minimize external API calls during authentication
3. Consider using Lambda Provisioned Concurrency to eliminate cold starts:

```
aws lambda put-provisioned-concurrency-config \
    --function-name my-authentication-function \
    --qualifier prod \
    --provisioned-concurrent-executions 5
```

4. Monitor Lambda performance using CloudWatch metrics and set up alarms for duration thresholds

Troubleshoot consistent Lambda timeout issues

Description

Users experience consistent timeouts when using a Lambda function for authentication.

Cause

Lambda can't reach the corresponding AWS service used to authenticate (such as DynamoDB, Secrets Manager, or other identity provider).

Solution

Verify that subnets can reach AWS services. Or, if connecting to an internet identity provider (such as Okta), verify that the Lambda function's subnet can reach the internet via a NAT Gateway

Troubleshoot `KeyError` exceptions

Description

In your Transfer Family log entries, you notice 'KeyError' exceptions.

Cause

The most likely cause is that user or `identity_provider` record is malformed or missing required fields.

Solution

Review the ERRORS log, located in `/aws/transfer/`*`your-server-id`* log group for clues.

Troubleshoot workflow issues

This section describes possible solutions for issues with managed workflows.

Topics

- Troubleshoot managed workflows issues
- Troubleshoot workflow decryption issues

Troubleshoot managed workflows issues

This section describes possible solutions for the following workflow issues.

Topics

- Troubleshoot workflow-related errors using Amazon CloudWatch
- Troubleshoot workflow copy errors

Troubleshoot workflow-related errors using Amazon CloudWatch

Description

If you are having issues with your workflows, you can use Amazon CloudWatch to investigate the cause.

Cause

There can be several causes. Use Amazon CloudWatch Logs to investigate.

Solution

Transfer Family emits workflow execution status into CloudWatch Logs. The following types of workflow errors can appear in CloudWatch Logs:

- `"type": "StepErrored"`
- `"type": "ExecutionErrored"`
- `"type": "ExecutionThrottled"`
- `"Service failure on starting workflow"`

You can filter your workflow's execution logs using different filter and pattern syntax. For example, you can create a log filter in your CloudWatch logs to capture workflow execution logs that contain the **ExecutionErrored** message. For details, see Real-time processing of log data with subscriptions and Filter and pattern syntax in the *Amazon CloudWatch Logs User Guide*.

StepErrored

```
2021-10-29T12:57:26.272-05:00
            {"type":"StepErrored","details":
{"errorType":"BAD_REQUEST","errorMessage":"Cannot
            tag Efs file","stepType":"TAG","stepName":"successful_tag_step"},
            "workflowId":"w-
abcdef01234567890","executionId":"1234abcd-56ef-78gh-90ij-1234klmno567",
            "transferDetails":
{"serverId":"s-1234567890abcdef0","username":"lhr","sessionId":"1234567890abcdef0"}
```

Here, `StepErrored` indicates that a step within the workflow has generated an error. In a single workflow, you can have multiple steps configured. This error tells you in which step the error occurred and provides an error message. In this particular example, the step was configured to tag a file; however, tagging a file in an Amazon EFS file system is not supported, so the step generated an error.

ExecutionErrored

```
2021-10-29T12:57:26.618-05:00
            {"type":"ExecutionErrored","details":{},"workflowId":"w-w-
abcdef01234567890",
            "executionId":"1234abcd-56ef-78gh-90ij-1234klmno567","transferDetails":
{"serverId":"s-1234567890abcdef0",
```

```
            "username":"lhr","sessionId":"1234567890abcdef0"}}
```

When a workflow is not able to execute any step, it generates an `ExecutionErrored` message. For example, if you have configured a single step in a given workflow, and if the step is not able to execute, the overall workflow fails.

Executionthrottled

Execution is throttled if a workflow is getting triggered at a rate that is faster than the system can support. This log message indicates that you must slow down your execution rate for workflows. If you are not able to scale down your workflow-execution rate, contact AWS Support at Contact AWS.

Service failure on starting workflow

Anytime you remove a workflow from a server and replace it with a new one, or update server configuration (which impacts a workflow's execution role), you must wait approximately 10 minutes before executing the new workflow. The Transfer Family server caches the workflow details, and it takes 10 minutes for the server to refresh its cache.

Additionally, you must log out of any active SFTP sessions, and then log back in after the 10-minute waiting period to see the changes.

Troubleshoot workflow copy errors

Description

If you're executing a workflow that contains a step to copy the uploaded file, you could encounter the following error:

```
{
  "type": "StepErrored", "details": {
    "errorType": "BAD_REQUEST", "errorMessage": "Bad Request (Service: Amazon S3;
 Status Code: 400; Error Code: 400 Bad Request;
    Request ID: request-ID; S3 Extended Request ID: request-ID Proxy: null)",
 "stepType": "COPY", "stepName": "copy-step-name" },
  "workflowId": "workflow-ID",
  "executionId": "execution-ID",
  "transferDetails": {
     "serverId": "server-ID",
     "username": "user-name",
```

```
    "sessionId": "session-ID"
  }
}
```

Cause

The source file is in an Amazon S3 bucket that is in a different AWS Region than the destination bucket.

Solution

If you're executing a workflow that includes a copy step, make sure that the source and destination buckets are in the same AWS Region.

Troubleshoot workflow decryption issues

This section describes possible solutions for the following issues with encrypted workflows.

Topics

- Troubleshoot anonymous recipient encryption issues
- Troubleshoot error for signed encryption file
- Troubleshoot error for a FIPS algorithm

Troubleshoot anonymous recipient encryption issues

Description

Your decrypt workflow fails when processing certain encrypted files, but works with others.

Cause

Files may have been encrypted without specifying a recipient (anonymous encryption), making it difficult for the workflow to determine which key to use for decryption.

Solution

Always encrypt files with a non-anonymous recipient using the `-r` parameter. For example:

```
gpg -e -r user@example.com --openpgp file.txt
```

To check if a file was encrypted with a specific recipient or anonymously, use the `--list-packets` command:

```
gpg --list-packets file.txt.gpg
```

This command displays the packet structure of the GPG-encrypted file without decrypting its contents. Look for output that includes recipient information, such as:

```
:pubkey enc packet: version 3, algo 1, keyid 1A2B3C4D5E6F7G8H
```

If you see keyid information, the file was encrypted for a specific recipient. If this information is missing, the file may have been encrypted anonymously, which can cause decryption failures in Transfer Family workflows.

You can also use this command to verify:

- Which encryption algorithms were used
- Compression methods (if any)
- Creation timestamps

To view the list of supported cipher algorithms for your GPG installation, use:

```
gpg --version
```

This information can help you ensure that you're using encryption algorithms compatible with Transfer Family workflows, especially when FIPS compliance is required.

Troubleshoot error for signed encryption file

Description

Your decrypt workflow fails and you receive the following error:

```
"Encrypted file with signed message unsupported"
```

Cause

Transfer Family does not currently support signing for encrypted files.

Solution

In your PGP client, if there is an option to sign the encrypted file, make sure to clear the selection, as Transfer Family does not currently support signing for encrypted files.

Troubleshoot error for a FIPS algorithm

Description

Your decrypt workflow fails, and the log message resembles the following:

```
{
   "type": "StepErrored",
   "details": {
      "errorType": "BAD_REQUEST",
      "errorMessage": "File encryption algorithm not supported with FIPS mode
 enabled.",
      "stepType": "DECRYPT",
      "stepName": "step-name"
   },
   "workflowId": "workflow-ID",
   "executionId": "execution-ID",
   "transferDetails": {
      "serverId": "server-ID",
      "username": "user-name",
      "sessionId": "session-ID"
   }
}
```

Cause

Your Transfer Family server has FIPS mode enabled and an associated Decrypt workflow step. When encrypting the files before uploading to your Transfer Family server, the encryption client might generate encrypted files that use non-FIPS approved symmetric encryption algorithms. In such a scenario, the workflow is unable to decrypt files. In the following example, **GnuPG** version 2.4.0 is using OCB (a non-FIPS block cipher mode) to encrypt files: this causes the workflow to fail.

Solution

You must edit the GPG key that you used to encrypt your files, and then re-encrypt them. The following procedure describes the steps you must take.

To edit your PGP keys

1. Identify the key that you must edit by running `gpg --list-keys`

   This returns a list of keys. Each key has details similar to the following:

   ```
   pub   ed25519 2022-07-07 [SC]
         wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   uid           [ultimate] Mary Major <marymajor@example.com>
   sub   cv25519 2022-07-07 [E]
   ```

2. Identify the key that you want to edit. In the example shown in the previous step, the ID is `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`.

3. Run `gpg --edit-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`.

   The system responds with details about the **GnuPG** program and the specified key.

4. At the `gpg>` prompt, enter `showpref`. The following details are returned:

   ```
   [ultimate] (1). Mary Major <marymajor@example.com>
      Cipher: AES256, AES192, AES, 3DES
      AEAD: OCB
      Digest: SHA512, SHA384, SHA256, SHA224, SHA1
      Compression: ZLIB, BZIP2, ZIP, Uncompressed
      Features: MDC, AEAD, Keyserver no-modify
   ```

   Note that the preferred algorithms that are stored on the key are listed.

5. We want to edit the key to retain all algorithms except for **OCB**. Run the `setpref` command, specifying all the algorithms to retain:

   ```
   gpg> setpref AES256, AES192,AES,3DES,SHA512, SHA384, SHA256, SHA224, SHA1,ZLIB,
    BZIP2, ZIP, Uncompressed
   ```

   This returns the following details:

   ```
   Set preference list to:
        Cipher: AES256, AES192, AES, 3DES
        AEAD:
        Digest: SHA512, SHA384, SHA256, SHA224, SHA1
        Compression: ZLIB, BZIP2, ZIP, Uncompressed
        Features: MDC, Keyserver no-modify
   ```

```
Really update the preferences? (y/N)
```

6. Enter y to update, then enter your password when prompted to confirm the change.
7. Save the changes.

```
gpg> save
```

Before re-running your decrypt workflow, you must re-encrypt your files, using the edited key.

Troubleshoot EFS issues

This section describes possible solutions for issues with Amazon EFS storage.

Topics

- Troubleshoot Amazon EFS issues

Troubleshoot Amazon EFS issues

This section describes possible solutions for the following Amazon EFS issues.

Topics

- Troubleshoot Amazon EFS service-managed users
- Troubleshoot missing POSIX profile
- Troubleshoot logical directories with Amazon EFS

Troubleshoot Amazon EFS service-managed users

Description

You run the `sftp` command and the prompt doesn't appear, and instead you see the following message:

```
Couldn't canonicalize: Permission denied
           Need cwd
```

Cause

Your AWS Identity and Access Management (IAM) user's role does not have permission to access Amazon Elastic File System (Amazon EFS).

Solution

Increase the policy permissions for your user's role. You can add an AWS managed policy, such as `AmazonElasticFileSystemClientFullAccess`.

Troubleshoot missing POSIX profile

Description

If you're using Amazon EFS storage for your server and you're using a custom identity provider, you must provide your AWS Lambda function with a POSIX profile.

Cause

One possible cause is that the templates that we provide for creating an AWS Lambda-backed Amazon API Gateway method do not currently contain POSIX information.

If you did provide POSIX information, the format that you used for providing the POSIX information might not be getting parsed correctly by Transfer Family.

Solution

Make sure that you are providing a JSON element to Transfer Family for the `PosixProfile` parameter.

For example, if you're using Python, you could add the following line where you parse the `PosixProfile` parameter:

```
if PosixProfile:
        response_data["PosixProfile"] = json.loads(PosixProfile)
```

Or, in JavaScript, you could add the following line, where the *`uid-value`* and *`gid-value`* are integers, 0 or greater, that represent the User ID (UID) and Group ID (GID) respectively:

```
PosixProfile: {"Uid": uid-value, "Gid": gid-value},
```

These code examples send the `PosixProfile` parameter to Transfer Family as a JSON object, rather than as a string.

Also, within AWS Secrets Manager, you must store the `PosixProfile` parameter as follows. Replace *`your-uid`* and *`your-gid`* with your actual values for the GID and UID.

```
{"Uid": your-uid, "Gid": your-gid, "SecondaryGids": []}
```

Troubleshoot logical directories with Amazon EFS

Description

If the user's home directory does not exist, and they run an `ls` command, the system responds as follows:

```
sftp> ls
remote readdir ("/"): No such file or directory
```

Cause

If your Transfer Family server uses Amazon EFS, the home directory for the user must be created with read and write access before the user can work in their logical home directory. The user cannot create this directory themselves, as they would lack permissions for `mkdir` on their logical home directory.

Solution

A user with administrative access to the parent directory needs to create the user's logical home directory.

Troubleshoot storage and encryption issues

This section describes possible solutions for issues with storage and encryption.

Topics

- Troubleshoot policies for encrypted Amazon S3 buckets
- Troubleshoot ResourceNotFound exception

Troubleshoot policies for encrypted Amazon S3 buckets

Description

You have an encrypted Amazon S3 bucket that you are using as storage for your Transfer Family server. If you try to upload a file to the server, you receive the error `Couldn't close file: Permission denied`.

And if you view the server logs, you see the following errors:

```
ERROR Message="Access denied" Operation=CLOSE Path=/bucket/user/test.txt BytesIn=13
ERROR Message="Access denied"
```

Cause

The policy for your IAM user does not have permission to access the encrypted bucket.

Solution

You must specify additional permissions in your policy to grant the required AWS Key Management Service (AWS KMS) permissions. For details, see Data protection and encryption.

Troubleshoot `ResourceNotFound` exception

Description

You receive an error where the resource cannot be found. For example, if you run `UpdateServer`, you might get the following error:

```
An error occurred (ResourceNotFoundException) when calling the UpdateServer operation:
 Unknown server
```

Cause

There are several reasons for receiving a ResourceNotFoundException message. In most cases, the resource that you specified in your API command does not exist. If you did specify an existing resource, then the most probable cause is that your default region is different than the region for your resource. For example, if your default region is **us-east-1**, and your Transfer Family server is in **us-east-2**, you will receive an Unknown resource exception.

For details about setting a default region, see Quick configuration with `aws configure`.

Solution

Add a region parameter to your API command to explicitly specify where to find a particular resource.

```
aws transfer -describe-server --server-id server-id --region us-east-2
```

Troubleshoot monitoring and alerting issues

This section provides information about troubleshooting monitoring and alerting issues with AWS Transfer Family, including missing or incomplete CloudWatch metrics and missing EventBridge events.

Topics

- Troubleshoot missing or incomplete CloudWatch metrics
- Troubleshoot missing EventBridge events

Troubleshoot missing or incomplete CloudWatch metrics

Description

CloudWatch metrics for your Transfer Family servers are missing, incomplete, or not updating as expected.

Cause

Missing or incomplete metrics can be caused by several factors:

- Logging configuration issues
- Low activity levels that don't generate metrics within the expected timeframe
- Viewing metrics with incorrect dimensions or time ranges

Solution

To resolve issues with missing or incomplete CloudWatch metrics:

1. Ensure that logging is properly configured for your Transfer Familyserver:
    - In the Transfer Family console, check that logging is enabled under **Server details > Additional details > Logging role**.

- Very that the logging role has the necessary permissions and trust relationships.

2. When viewing metrics in the CloudWatch console:

   - Use the correct dimensions, for example **ServerId** for server-level metrics
   - Adjust the time range to ensure it covers periods of activity
   - Check that you're in the correct AWS Region

3. Generate test activity on your Transfer Family server to ensure metrics are being produced.

Troubleshoot missing EventBridge events

Description

You've configured Amazon EventBridge rules to capture Transfer Family events, but events are not being delivered to your target destinations or triggering expected actions.

Cause

Missing EventBridge events can be caused by:

- Incorrectly configured event patterns
- Permission issues with event targets
- Service limits or throttling
- Events not being generated due to server configuration

Solution

To troubleshoot missing EventBridge events:

1. Verify your event pattern is correctly formatted to match Transfer Family events:

```
{
  "source": ["aws.transfer"],
  "detail-type": ["Transfer State Change"],
  "detail": {
    "serverId": ["s-1234567890abcdef0"]
  }
}
```

2. Check that your event target has the necessary permissions:
   - For Lambda targets, ensure the Lambda function's resource policy allows EventBridge to invoke it
   - For SQS targets, verify the queue policy allows EventBridge to send messages
   - For SNS targets, confirm the topic policy permits EventBridge to publish to it
3. Test your rule by generating sample events:
   - Use the EventBridge console to create a test event that matches your pattern
   - Perform actions on your Transfer Family server that should generate events
4. Enable EventBridge rule metrics to monitor rule invocations and failures:

```
aws events put-rule --name "TransferStateChangeRule" --event-pattern '{...}' --state
 ENABLED --metrics-enabled
```

5. Check CloudWatch Logs for any error messages related to event delivery failures

Troubleshoot cross-region transfer issues

This section describes possible solutions for issues related to transferring files across AWS Regions.

Topics

- Troubleshoot cross-region transfer permission issues
- Troubleshoot cross-region transfer performance issues

Troubleshoot cross-region transfer permission issues

Description

When attempting to transfer files between Amazon S3 buckets in different regions using Transfer Family workflows, you encounter errors such as:

```
{
  "type": "StepErrored",
  "details": {
    "errorType": "BAD_REQUEST",
    "errorMessage": "Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied)",
    "stepType": "COPY",
```

```
    "stepName": "cross_region_copy"
  }
}
```

Cause

Cross-region transfers require specific IAM permissions for both the source and destination buckets. The IAM role used by your Transfer Family server or workflow may not have sufficient permissions to access buckets in other regions.

Solution

To resolve cross-region transfer permission issues:

1. Ensure your IAM role has permissions for both source and destination buckets:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::source-bucket-name/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::destination-bucket-name/*"
    }
  ]
}
```

2. If using KMS encryption, add permissions for both the source and destination KMS keys:

```
{
  "Effect": "Allow",
  "Action": [
```

```
      "kms:Decrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": [
      "arn:aws:kms:source-region:account-id:key/source-key-id",
      "arn:aws:kms:destination-region:account-id:key/destination-key-id"
    ]
}
```

3. Verify that bucket policies in both regions allow access from your Transfer Family server's IAM role
4. For cross-account transfers, ensure proper cross-account permissions are configured

Troubleshoot cross-region transfer performance issues

Description

Cross-region transfers are significantly slower than expected or time out during large file transfers.

Cause

Cross-region transfers inherently involve greater latency and may be affected by network conditions, file sizes, and service limits. Large files or high volumes of small files can experience performance degradation.

Solution

To improve cross-region transfer performance:

- For large files, consider using Amazon S3 Transfer Acceleration:

```
aws s3 cp --source-region us-east-1 --region us-west-2 \
    s3://source-bucket/large-file.zip s3://destination-bucket/large-file.zip \
    --acl bucket-owner-full-control --s3-accelerate
```

- For multiple small files, batch them together before transfer:
  - Use compression to combine multiple files
  - Use Amazon S3 batch operations for large-scale transfers
- Consider using Transfer Family SFTP connectors with appropriate timeout settings for large transfers

- For recurring transfers, consider replicating data using Amazon S3 Cross-Region Replication (CRR) instead of ad-hoc transfers
- Monitor transfer performance using Amazon CloudWatch metrics to identify bottlenecks

Troubleshoot Terraform deployment issues

This section describes possible solutions for issues related to deploying Transfer Family resources using Terraform. For general information about Terraform modules for Transfer Family, see Transfer Family Terraform modules.

Topics

- Troubleshoot Terraform resource creation failures
- Troubleshoot Terraform state management issues

Troubleshoot Terraform resource creation failures

Description

When attempting to create Transfer Family resources using Terraform, you encounter errors such as:

```
Error: error creating Transfer Server: InvalidRequestException: The request is not
 valid.
Error: error creating Transfer User: InvalidRequestException: Unable to create the user
 because the server endpoint type is incompatible with the home directory type.
```

Cause

These errors typically occur due to incompatible configuration parameters or missing dependencies in your Terraform configuration. Common causes include:

- Incompatible endpoint type and storage configurations
- Missing required IAM roles or policies
- Incorrect security policy specifications
- VPC endpoint configuration issues

Solution

To resolve Terraform deployment issues:

- Ensure your Terraform configuration uses compatible parameter combinations:
  - For public endpoints, ensure you're using Amazon S3 for storage.
  - For VPC endpoints, verify proper VPC and security group configuration.
- Use explicit dependencies with the `depends_on` attribute to ensure resources are created in the correct order.
- Verify that all IAM roles have the necessary trust relationships and permissions.
- Use the latest version of the AWS provider for Terraform to ensure compatibility with all Transfer Family features.
- For complex deployments and simple use cases, consider using the official Transfer Family Terraform modules available on GitHub at https://github.com/aws-ia/terraform-aws-transfer-family. These modules provide extensive examples covering simple and complex customer use cases, follow AWS best practices, and can simplify deployment for customers who need help with infrastructure as code (IaC) configuration.

Troubleshoot Terraform state management issues

Description

After making changes to your Transfer Family resources outside of Terraform (via console or AWS CLI), you encounter state drift or errors when running `terraform plan` or `terraform apply`.

Cause

Terraform maintains a state file that tracks the resources it manages. When changes are made outside of Terraform, the state file becomes out of sync with the actual resources, causing errors or unexpected behavior during subsequent Terraform operations.

Solution

To resolve Terraform state management issues with Transfer Family resources:

1. Use `terraform import` to bring existing resources under Terraform management:

```
terraform import <transfer_family_server.example> s-<server-id>
terraform import <transfer_family_server.example> s-<server-id>/username
```

2. Use `terraform refresh` to update the state file with the current real-world infrastructure

3. For resources that can't be imported or have complex state issues, consider using `terraform state rm` to remove them from the state file, then recreate them with Terraform
4. Implement a policy to manage Transfer Family resources exclusively through Terraform to prevent future state drift
5. Use remote state storage with locking to prevent concurrent modifications when working in teams

Troubleshoot Web Application Firewall integration issues

This section describes possible solutions for issues related to integrating AWS WAF with Transfer Family.

Topics

- Troubleshoot WAF blocking legitimate traffic
- Troubleshoot WAF integration with custom identity providers

Troubleshoot WAF blocking legitimate traffic

Description

After configuring AWS WAF with your Transfer Family endpoint, legitimate users are unable to connect or experience intermittent connection failures. You may see HTTP 403 (Forbidden) responses in your logs.

Cause

Your AWS WAF rules may be too restrictive or incorrectly configured, causing false positives that block legitimate traffic. Common causes include:

- IP-based rules that inadvertently block corporate networks or VPNs
- Rate-based rules with thresholds that are too low for your normal traffic patterns
- Managed rule groups that are overly aggressive for your use case

Solution

To resolve false positive issues:

1. Enable AWS WAF logging to identify which rules are triggering the blocks. For instructions, see Logging AWS WAF web ACL traffic.
2. Review your logs to identify patterns in the blocked requests.
3. Adjust your rules by:
   - Adding IP addresses or ranges to an allowlist
   - Increasing rate limits for rate-based rules
   - Setting specific rules to Count mode instead of Block mode to monitor without blocking
   - Creating exceptions for specific rules using rule group exclusions
4. Test the updated configuration with a representative sample of legitimate traffic before fully deploying.

Troubleshoot WAF integration with custom identity providers

Description

After configuring AWS WAF with your Transfer Family server that uses a custom identity provider, authentication fails or users experience intermittent authentication issues.

Cause

When using a custom identity provider with API Gateway, AWS WAF rules may interfere with the API calls between Transfer Family and your identity provider. This can happen because AWS WAF is inspecting and potentially blocking the API traffic based on its rule sets.

Solution

To resolve issues with AWS WAF and custom identity providers:

- Ensure that your AWS WAF configuration includes exceptions for the API Gateway endpoints used by your custom identity provider.
- Add the Transfer Family service principal (transfer.amazonaws.com) to an allowlist in your AWS WAF rules.
- If using managed rule groups, review them for rules that might affect API authentication flows and consider disabling those specific rules.
- Test your identity provider directly using the `TestIdentityProvider` API operation to verify it works correctly without AWS WAF interference.

Troubleshoot service-managed user issues

This section describes possible solutions for issues with service-managed users.

Topics

- Troubleshoot service-managed users

Troubleshoot service-managed users

This section describes possible solutions for the following issues.

Topics

- Troubleshoot public key body too long
- Troubleshoot failed to add SSH public key

Troubleshoot public key body too long

Description

When you try to create a service-managed user, you receive the following error:

```
Failed to create user (1 validation error detected:
'sshPublicKeyBody' failed to satisfy constraint: Member must have length less than or
 equal to 2048)
```

Cause

You might be entering a PGP key for the public key body, and AWS Transfer Family does not support PGP keys for service-managed users.

Solution

If the PGP key is RSA-based, you can convert it to PEM format. For example, Ubuntu provides a conversion tool here: https://manpages.ubuntu.com/manpages/jammy/en/man1/openpgp2ssh.1.html

Troubleshoot failed to add SSH public key

Description

When you try to add a public key for a service-managed user, you receive the following error:

```
Failed to add SSH public key (Unsupported or invalid SSH public key format)
```

Cause

You might be attempting to import an SSH2-formatted public key, and AWS Transfer Family does not support SSH2-formatted public keys for service-managed users.

Solution

You need to convert the key into OpenSSH format. This process is described in Converting an SSH2 key to SSH public key format.

Troubleshoot AS2 issues

This section describes possible solutions for issues with AS2 transfers.

Topics

- Troubleshoot AS2 issues
- AS2 certificate issues
- AS2 MDN receipt issues
- Certificate expiration monitoring issues

Troubleshoot AS2 issues

AS2 error codes, messages and troubleshooting tips for Applicability Statement 2 (AS2)-enabled servers are described in the AS2 failure codes section in this guide.

AS2 certificate issues

Description

You're experiencing certificate-related errors with AS2 transfers.

Cause

Common causes include expired certificates, incorrect certificate formats, or mismatched certificate chains.

Solution

Try the following solutions:

- Verify that your certificates are not expired
- Ensure certificates are in the correct format (PEM for AWS Transfer Family)
- Check that the certificate chain is complete and valid
- Confirm that the signing and encryption certificates match between trading partners
- Rotate certificates well before expiration to avoid interruptions

AS2 MDN receipt issues

Description

You're not receiving expected Message Disposition Notifications (MDNs) for AS2 transfers.

Cause

MDN issues can occur due to network connectivity problems, incorrect endpoint configurations, or MDN format mismatches.

Solution

Consider these solutions:

- Verify that the MDN URL is correctly configured and accessible
- Check network connectivity between the AS2 server and the MDN endpoint
- Ensure that both trading partners are configured for the same MDN type (synchronous or asynchronous)
- Review AS2 logs for any errors related to MDN processing
- If using synchronous MDNs, verify that timeouts are set appropriately

Certificate expiration monitoring issues

This section provides solutions for common issues related to certificate expiration monitoring and the DaysUntilExpiry metric.

DaysUntilExpiry metric not appearing

Problem: The DaysUntilExpiry metric is not visible in Amazon CloudWatch after importing a certificate.

Solution:

- Wait up to 24 hours after importing the certificate. It can take up to a full day for Transfer Family to emit the metric to your account.
- Ensure you're looking in the correct AWS region and under the `AWS/Transfer` namespace in CloudWatch.

Certificate expiration alarms not triggering

Problem: CloudWatch alarms for certificate expiration are not triggering when expected.

Solution:

- Verify that the alarm is configured with the `Maximum` statistic and a period of 1 day.
- Check that the threshold comparison is set to `Less than or equal to` the desired number of days.
- Ensure that `Treat missing data as good (not breaching)` is selected in the alarm configuration.
- Verify that the alarm dimensions match your certificate's CertificateId and Description (if provided).
- Check that the alarm actions (SNS topics, email notifications) are properly configured and active.

AWS Transfer Family API Reference

The complete API Reference guide for Transfer Family is available at AWS Transfer Family API Reference.

AWS Transfer Family is a secure transfer service that you can use to transfer files into and out of Amazon Simple Storage Service (Amazon S3) storage over the following protocols:

- Secure Shell (SSH) File Transfer Protocol (SFTP)
- File Transfer Protocol Secure (FTPS)
- File Transfer Protocol (FTP)
- Applicability Statement 2 (AS2)

Servers, users, and roles are all identified by their Amazon Resource Name (ARN). You can assign tags, which are key-value pairs, to entities with an ARN. Tags are metadata that can be used to group or search for these entities. One example where tags are useful is for accounting purposes.

The following conventions are observed in AWS Transfer Family ID formats:

- `ServerId` values take the form `s-01234567890abcdef`.
- `SshPublicKeyId` values take the form `key-01234567890abcdef`.

Amazon Resource Name (ARN) formats take the following form:

- For servers, ARNs take the form `arn:aws:transfer:`*`region`*`:`*`account-id`*`:server/`*`server-id`*.

  An example of a server ARN is: `arn:aws:transfer:us-east-1:123456789012:server/s-01234567890abcdef`.
- For users, ARNs take the form `arn:aws:transfer:`*`region`*`:`*`account-id`*`:user/`*`server-id`*`/`*`username`*.

  An example is `arn:aws:transfer:us-east-1:123456789012:user/s-01234567890abcdef/user1`.

DNS entries (endpoints) in use are as follows:

- API endpoints take the form `transfer.region.amazonaws.com`.
- Server endpoints take the form `server-id.server.transfer.region.amazonaws.com`.

This API interface reference for AWS Transfer Family contains documentation for a programming interface that you can use to manage AWS Transfer Family. The reference structure is as follows:

- For the alphabetical list of API actions, see Actions.
- For the alphabetical list of data types, see Types.
- For a list of common query parameters, see Common Parameters.
- For descriptions of the error codes, see Common Errors.

> **Tip**
>
> Rather than actually running a command, you can use the `--generate-cli-skeleton` parameter with any API call to generate and display a parameter template. You can then use the generated template to customize and use as input on a later command. For details, see Generate and use a parameter skeleton file.

Making API requests

In addition to using the console, you can use the AWS Transfer Family API to programmatically configure and manage your servers. This section describes the AWS Transfer Family operations, request signing for authentication and the error handling. For information about the regions and endpoints available for Transfer Family, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*

> **Note**
>
> You can also use the AWS SDKs when developing applications with Transfer Family;. The AWS SDKs for Java, .NET, and PHP wrap the underlying Transfer Family API, simplifying your programming tasks. For information about downloading the SDK libraries, see Sample code libraries.

Transfer Family required request headers

This section describes the required headers that you must send with every POST request to AWS Transfer Family. You include HTTP headers to identify key information about the request including the operation you want to invoke, the date of the request, and information that indicates the authorization of you as the sender of the request. Headers are case insensitive and the order of the headers is not important.

The following example shows headers that are used in the ListServers operation.

```
POST / HTTP/1.1
Host: transfer.us-east-1.amazonaws.com
x-amz-target: TransferService.ListServers
x-amz-date: 20220507T012034Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIDEXAMPLE/20220507/us-east-1/transfer/
aws4_request,
    SignedHeaders=content-type;host;x-amz-date;x-amz-target,
    Signature=13550350a8681c84c861aac2e5b440161c2b33a3e4f302ac680ca5b686de48de
Content-Type: application/x-amz-json-1.1
Content-Length: 17

{"MaxResults":10}
```

The following are the headers that must include with your POST requests to Transfer Family. Headers shown below that begin with "x-amz" are specific for AWS. All other headers listed are common header used in HTTP transactions.

Transfer Family request inputs and signing

All request inputs must be sent as part of JSON payload in request body. For Actions in which all request fields are optional, for example `ListServers`, you still need to provide an empty JSON object in the request body, such as `{}`. The structure of Transfer Family payload request/response is documented in existing the API reference, for example DescribeServer.

Transfer Family supports authentication using AWS Signature Version 4. For details, see Signing AWS API requests.

Error responses

When there is an error, the response header information contains:

- Content-Type: `application/x-amz-json-1.1`
- An appropriate `4xx` or `5xx` HTTP status code

The body of an error response contains information about the error that occurred. The following sample error response shows the output syntax of response elements common to all error responses.

```
{
    "__type": "String",
    "Message": "String", <!-- Message is lowercase in some instances -->
    "Resource": "String",
    "ResourceType": "String",
    "RetryAfterSeconds": "String"
}
```

The following table explains the JSON error response fields shown in the preceding syntax.

__type

One of the exceptions from a Transfer Family API call.

Type: String

Message or **message**

One of the operation error code messages.

> **Note**
>
> Some exceptions use `message`, and others use `Message`. You can check the code for your interface to determine the proper case. Alternatively, you can test each option to see which works.

Type: String

Resource

The resource for which the error is invoked. For example, if you try to create a user that already exists, the `Resource` is the username for the existing user.

Type: String

ResourceType

The resource type for which the error is invoked. For example, if you try to create a user that already exists, the `ResourceType` is `User`.

Type: String

RetryAfterSeconds

The number of seconds to wait before retrying the command.

Type: String

Error response examples

The following JSON body is returned if you call the `DescribeServer` API and specify a server that does not exist.

```
{
  "__type": "ResourceNotFoundException",
  "Message": "Unknown server",
  "Resource": "s-11112222333344444",
  "ResourceType": "Server"
}
```

The following JSON body is returned if executing an API causes throttling to occur.

```
{
   "__type":"ThrottlingException",
   "RetryAfterSeconds":"1"
}
```

The following JSON body is returned if you use the `CreateServer` API and you do not have sufficient permissions to create a Transfer Family server.

```
{
  "__type": "AccessDeniedException",
  "Message": "You do not have sufficient access to perform this action."
}
```

The following JSON body is returned if you use the `CreateUser` API and specify a user that already exists.

```
{
  "__type": "ResourceExistsException",
  "Message": "User already exists",
  "Resource": "Alejandro-Rosalez",
  "ResourceType": "User"
}
```

Available libraries

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of the command-line tools and Query API. These libraries provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. See Tools to build on AWS

For libraries and sample code in all languages, see Sample code & libraries.

Identity Providers

AWS Transfer Family supports multiple identity provider types to authenticate and manage users. Each server can use only one authentication method, which must be selected when the server is created.

Service Managed

With the `SERVICE_MANAGED` authentication method, user credentials are stored and managed within AWS Transfer Family. Users are authenticated using SSH public keys that are associated with their username on the server.

Each user can have one or more SSH public keys stored in the service. When a client requests a file operation, it provides the username and SSH private key, which is authenticated against the stored public key.

Directory Service

The `AWS_DIRECTORY_SERVICE` authentication method allows you to integrate with AWS Directory Service for Microsoft Active Directory (AWS Directory Service for Microsoft Active Directory).

This option enables you to manage user authentication and access through your existing Active Directory groups. Users can authenticate using their Active Directory credentials.

There is a default limit of 100 Active Directory groups per server, which can be increased to a maximum of 150 groups through a service limit increase.

Lambda

The `AWS_LAMBDA` authentication method allows you to connect to a custom identity provider using AWS Lambda.

This option provides flexibility to integrate with your existing identity management systems. The Lambda function is responsible for authenticating users and returning the appropriate access policies.

Custom (API Gateway)

The `API_GATEWAY` authentication method (displayed as **Custom** in the console) allows you to use a custom authentication method that provides both user authentication and access control.

This method relies on the Amazon API Gateway to use your API call from your identity provider to validate user requests. You might use this custom method to authenticate users against a directory service, a database name/password pair, or some other mechanism.

For all authentication methods, users are assigned policies that define their access to Amazon S3 buckets or Amazon Elastic File System file systems. The server inherits the trust relationship from the user through an IAM role with an `AssumeRole` action, allowing it to perform file operations on behalf of the user.

Naming Conventions

AWS Transfer Family uses standardized formats for resource identifiers and Amazon Resource Names (ARNs). Understanding these conventions is important when working with the AWS Transfer Family API.

ID Formats

The following conventions are observed in AWS Transfer Family ID formats:

Server IDs

`ServerId` values take the form `s-01234567890abcdef`.

SSH Public Key IDs

`SshPublicKeyId` values take the form `key-01234567890abcdef`.

Connector IDs

`ConnectorId` values take the form `c-01234567890abcdef`.

Workflow IDs

`WorkflowId` values take the form `w-01234567890abcdef`.

Profile IDs

`ProfileId` values take the form `p-01234567890abcdef`.

WebApp IDs

`WebAppId` values take the form `webapp-01234567890abcdef`.

ARN Formats

Amazon Resource Name (ARN) formats take the following form:

Server ARNs

For servers, ARNs take the form `arn:aws:transfer:`*`region`*`:`*`account-id`*`:server/`*`server-id`*.

Example: `arn:aws:transfer:us-east-1:123456789012:server/s-01234567890abcdef`.

User ARNs

For users, ARNs take the form `arn:aws:transfer:`*`region`*`:`*`account-id`*`:user/`*`server-id`*`/`*`username`*.

Example: `arn:aws:transfer:us-east-1:123456789012:user/s-01234567890abcdef/user1`.

Connector ARNs

For connectors, ARNs take the form `arn:aws:transfer:`*`region`*`:`*`account-id`*`:connector/`*`connector-id`*.

Example: `arn:aws:transfer:us-east-1:123456789012:connector/c-01234567890abcdef`.

Workflow ARNs

For workflows, ARNs take the form `arn:aws:transfer:`*`region`*`:`*`account-id`*`:workflow/`*`workflow-id`*.

Example: `arn:aws:transfer:us-east-1:123456789012:workflow/w-01234567890abcdef`.

WebApp ARNs

For web applications, ARNs take the form `arn:aws:transfer:`*`region`*`:`*`account-id`*`:webapp/`*`webapp-id`*.

Example: `arn:aws:transfer:us-east-1:123456789012:webapp/webapp-01234567890abcdef`.

You can assign tags, which are key-value pairs, to entities with an ARN. Tags are metadata that can be used to group or search for these entities. One example where tags are useful is for accounting purposes.

DNS and Endpoints

AWS Transfer Family uses standardized DNS naming conventions for both API endpoints and server endpoints. Understanding these endpoints is essential for configuring clients and making API calls.

API Endpoints

API endpoints are used for making API calls to manage AWS Transfer Family resources. These endpoints take the following forms:

Standard API Endpoints

Standard API endpoints take the form `transfer.`*`region`*`.amazonaws.com`.

Example: `transfer.us-east-1.amazonaws.com`

Dual-Stack API Endpoints

AWS Transfer Family offers dual-stack API endpoints that can be accessed using either IPv4 or IPv6 requests:

- https://transfer.*region-code*.api.aws
- https://transfer-fips.*region-code*.api.aws

Server Endpoints

Server endpoints are used by file transfer clients to connect to AWS Transfer Family servers. These endpoints take the following forms:

Standard Server Endpoints

Standard server endpoints take the form *server-id*.server.transfer.*region*.amazonaws.com.

Example: s-01234567890abcdef.server.transfer.us-east-1.amazonaws.com

Custom Hostnames

You can also configure custom hostnames for your AWS Transfer Family servers. Custom hostnames can be used to provide a more user-friendly or branded experience for your users.

To use a custom hostname, you must:

1. Own the domain name
2. Provide a valid certificate
3. Configure DNS records to point to your AWS Transfer Family server

For a complete list of AWS Transfer Family endpoints by AWS Region, see the AWS Transfer Family endpoints and quotas in the *AWS General Reference*.

Document history for AWS Transfer Family

The following table describes the documentation for this release of AWS Transfer Family.

- **API version:** transfer-2018-11-05
- **Latest documentation update:** October 14, 2026

Change	Description	Date
AWS Secrets Manager storage for SFTP server host keys	AWS Transfer Family SFTP-enabled servers can now store host keys in AWS Secrets Manager, so that you choose the AWS Key Managemen t Service key that encrypts the private key material and manage the secret alongside your other secrets. For details, see Storing SFTP server host keys in AWS Secrets Manager.	October 14, 2026
Configurable SFTP ports for public server endpoints	You can now choose the ports that an SFTP server with a public endpoint listens on. Use the `SftpPorts` property to configure up to three ports when you create or update a server. For details, see the `SftpPorts` documentation in the ProtocolDetails topic.	September 29, 2026
Source IP preservation for SFTP servers behind a Network Load Balancer	AWS Transfer Family SFTP servers behind a Network Load Balancer (NLB) can use PROXY protocol v2 to	September 15, 2026

Change	Description	Date
	preserve the client's source IP address. IP-based access policies in your custom identity provider then see the real client address instead of the NLB private IP. For details, see Working with Network Load Balancers.	
VPC endpoint support for Transfer Family web apps	AWS Transfer Family web apps now support VPC hosted endpoints, allowing you to host your web app endpoint inside a virtual private cloud for private data transfers to and from Amazon S3 without traversing the public internet. For details, see Create a Transfer Family web app in a VPC.	November 19, 2025
IPv6 support	AWS Transfer Family now supports dual-stack (IPv4 and IPv6) endpoints for SFTP public endpoints, VPC-internal endpoints for SFTP/FTPS/FTP/AS2, SFTP and AS2 connectors, and API endpoints. For details, see IPv6 support for Transfer Family servers.	June 30, 2025

Change	Description	Date
Enhancements to the SFTP connectors user experience	• Ability to self-serve concurrent connections setting for your connectors • Ability to provide SSH private key in OpenSSH format that is used for authenticating connections • Ability to discover the public host key of remote servers using their SFTP connectors For details, see Create an SFTP connector with service-managed egress.	April 9, 2025
Ability for SFTP connectors to delete, rename, and move files on the remote SFTP server	You can now organize your files stored on remote SFTP servers by deleting, renaming or moving source files to archive locations after they have been copied from the remote server. For details, see Move or rename files or directories on the remote SFTP server.	April 7, 2025

Change	Description	Date
Support for AWS Transfer Family web apps	AWS Transfer Family web apps are a new resource customers can use to create a simple interface for accessing their data in Amazon S3 through a web browser. For details, see Transfer Family web apps.	December 1, 2024
Moved API Reference into a separate guide	To improve the customer experience, the API Reference is now published separately from the user guide. For the separate API reference, see Welcome to the AWS Transfer Family API.	July 31, 2024
Ability for SFTP connectors to list remote files and directories	Transfer Family has added the ability for our customers to use SFTP connectors to list files stored in remote SFTP servers. For details, see List contents of a remote directory	April 23, 2024
Ability to use a trading partner's self-signed TLS certificate with AS2 message exchange	AWS Transfer Family has added the option to import and use a trading partner's public, self-signed TLS certificate for sending Applicability Statement 2 (AS2) messages to their server over HTTPS .	April 12, 2024

Change	Description	Date
Addition of security policies for SFTP connectors	AWS Transfer Family has added security policies for use with SFTP connectors. For details, see Security policies for AWS Transfer Family SFTP connectors.	April 5, 2024
Integrate with Amazon EventBridge	AWS Transfer Family now automatically publishes events to Amazon EventBridge for all file transfer operations. For details, see Managing Transfer Family events using Amazon EventBridge.	February 8, 2024
Addition of new security policies	AWS Transfer Family has added new FIPS and non-FIPS security policies. Also, the default security policy assigned to servers is always the latest security policy. For details, see Security policies for AWS Transfer Family servers.	February 5, 2024

Change	Description	Date
Support for static IP addresses for SFTP connectors and AS2	Transfer Family now provides static IP addresses for SFTP connectors and AS2. This enables connection with remote SFTP servers that are secured by IP allowlisting controls. For AS2, we're introducing static IP addresses for asynchronous MDN responses from AS2 servers.	January 16, 2024
The user guide has been reorganized to align more closely with the latest version of AWS Transfer Family.	Transfer Family has added multiple features since the guide originated, necessitating a restructuring of the guide.	January 3, 2024
Logical directory mappings enhancements Amazon S3 list performance optimization	Transfer Family now supports logical directory mappings up to 2.1 MB. You can also now declare whether a user mapping is to a file. For more information, see Rules for using logical directories. When creating or updating a server that uses Amazon S3 for storage, you can now optimize the performance of listing your S3 directories (or folders). For more information, see Configuring an SFTP, FTPS, or FTP server endpoint.	November 17, 2023

Change	Description	Date
Alternate port for SFTP servers with virtual private cloud (VPC) endpoints	You can now enable an alternate nonstandard port for your SFTP Transfer Family servers that have VPC endpoints. For more informati on, see Create a server in a virtual private cloud.	November 17, 2023
Support for SFTP connectors	SFTP Connectors extend the capabilities of AWS Transfer Family to communicate with remote servers both in the cloud and on-premises. For more information, see Using SFTP connectors.	July 25, 2023
Support for AS2 Basic authentication	Transfer Family now supports using Basic authentication for servers that use the Applicabi lity Statement 2 (AS2) protocol. For more informati on, see Basic authentication for AS2 connectors.	June 30, 2023
Support for structured JSON logging	Transfer Family now supports delivering structured JSON logs to Amazon CloudWatc h, grouping log steams into custom log groups, and performing common log queries across protocols. For more information, see Amazon CloudWatch logging for AWS Transfer Family servers.	June 24, 2023

Change	Description	Date
Support for multiple methods of authentication	Transfer Family has support for authenticating by using a password, a public/private key pair, or both. This is available for servers that use the SFTP protocol and a custom identity provider. For more information, see Create an SFTP-enabled server.	May 17, 2023
Support for Pretty Good Privacy (PGP) decryption with files that Transfer Family processes with workflows	Transfer Family has built-in support for Pretty Good Privacy (PGP) decryption. You can use PGP decryption on files that are uploaded over SFTP, FTPS, or FTP to Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS). For more information, see Generate PGP keys and Use PGP decryption in your workflow.	December 21, 2022
Fully managed support for Applicability Statement 2 (AS2) file transfer protocol with Transfer Family servers	You can create servers that use the AS2 protocol for sending and receiving information to and from trading partners who are inside or outside the AWS environment. For more information, see Configuring AS2.	July 25, 2022

Change	Description	Date
Support for display banners when creating a server	You can add customized messages when you create servers. You can display a pre-authentication message (all protocols), and a post-auth entication message (for FTP and FTPS servers). For more information, see Create an SFTP-enabled server, Create an FTPS-enabled server, or Create an FTP-enabled server.	February 17, 2022
Support for AWS Lambda as an identity provider	You can now connect to a custom identity provider using AWS Lambda with their Transfer Family servers. Previously, you had to supply an Amazon API Gateway URL to integrate a custom identity provider. For more informati on, see Using AWS Lambda to integrate your identity provider.	November 16, 2021
Support for Managed File Transfer Workflows	Managed File Transfer Workflows provide you with post-upload processing abstractions for the common tasks that you currently perform manually. For more information, see AWS Transfer Family managed workflows.	September 2, 2021

Change	Description	Date
Support for AWS Directory Service for Microsoft Active Directory	In addition to service managed and custom identity providers, you can now use AWS Directory Service for Microsoft Active Directory to manage user access for authentication and authorization. For more information, see Using AWS Directory Service for Microsoft Active Directory.	May 24, 2021
New AWS Regions	AWS Transfer Family is now available in the Africa (Cape Town) Region. For more information about Transfer Family endpoints, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*.	February 24, 2021
New AWS Regions	AWS Transfer Family is now available in the Asia Pacific (Hong Kong) and Middle East (Bahrain) Regions. For more information about Transfer Family endpoints, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*.	February 17, 2021

Change	Description	Date
Support for Amazon EFS as a data store	Transfer Family now supports file transfers into and out of Amazon Elastic File System (Amazon EFS). Amazon EFS is a simple, scalable, fully managed elastic NFS file system. For more information, see Configure an Amazon EFS file system.	January 06, 2021
Support for AWS WAF	Transfer Family now supports AWS WAF, a web application firewall that helps protect web applications and API operations from attacks. For more information, see Add a web application firewall.	November 24, 2020
Support for multiple security groups in a virtual private cloud (VPC)	You can now attach multiple security groups to a server in a VPC. For more information, see Create a server in a virtual private cloud.	October 15, 2020

Change	Description	Date
New AWS Regions	Transfer Family is now available in the AWS GovCloud (US) Regions. For more information about Transfer Family endpoints for AWS GovCloud (US) Regions, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*. For information about using Transfer Family in the AWS GovCloud (US) Regions, see AWS Transfer Family in the *AWS GovCloud (US) User Guide*.	September 30, 2020
A security policy with supported cryptographic algorithms can now be attached to your server	You can now attach a security policy that contains a set of supported cryptographic algorithms to your server. For more information, see Security policies for AWS Transfer Family servers.	August 12, 2020

Change	Description	Date
Support for Federal Information Processing Standard (FIPS) endpoints	FIPS-enabled endpoints are now available in North American AWS Regions. For available Regions, see AWS Transfer Family endpoints and quotas in the *AWS General Reference*. To enable FIPS for an SFTP-enabled server endpoint, see Create an SFTP-enabled server. To enable FIPS for an FTPS-enabled server endpoint, see Create an FTPS-enabled server. To enable FIPS for an FTP-enabled server endpoint, see Create an FTP-enabled server.	August 12, 2020
Username character-length increase and additional allowed characters	Usernames can now contain at signs (@) and periods (.), and can be a maximum length of 100 characters. To add a user, see Managing users for server endpoints.	August 12, 2020
Support for automatic Amazon CloudWatch logging AWS Identity and Access Management (IAM) role creation	Transfer Family now supports automatic creation of a CloudWatch logging IAM role to view end-user activity. For more information, see Create an SFTP-enabled server, Create an FTPS-enabled server, or Create an FTP-enabled server.	July 30, 2020

Change	Description	Date
AWS Transfer Family now supports Source IP as a factor for authorization.	Transfer Family adds support for using end users' source IP addresses as a factor for authorization, enabling you to apply an additiona l layer of security when authorizing access over Secure File Transfer Protocol (SFTP), File Transfer Protocol over SSL (FTPS), or File Transfer Protocol (FTP). For more information, see Working with custom identity providers.	June 9, 2020
AWS Transfer for SFTP is now AWS Transfer Family and adds support for FTP and FTPS.	You can now use two additional protocols for your users' file transfers: File Transfer Protocol Secure (FTPS) and File Transfer Protocol (FTP). Users can move, run, secure, and integrate FTP over SSL (FTPS) and plaintext FTP based workflows in AWS, in addition to existing Secure File Transfer Protocol (SFTP) support.	April 23, 2020

Change	Description	Date
Support for virtual private cloud (VPC) security groups and Elastic IP addresses	You can now create an allowlist for incoming IP addresses using security groups, providing an additional layer of security for servers. You can also associate Elastic IP addresses with your server's endpoint. By doing this, you can enable users behind firewalls to allow access to that endpoint. For more information, see Create a server in a virtual private cloud.	January 10, 2020
Support for working in a VPC	You can now create a server in a VPC. You can use your server to transfer data over your client to and from an Amazon S3 bucket without going over the public internet. For more informati on, see Create a server in a virtual private cloud.	March 27, 2019
First version of AWS Transfer Family released.	This initial release includes setting up directions, describes how to get started, and provides information on client configuration, user configuration, and monitoring activity.	November 25, 2018