User Guide

Amazon Elastic File System

Amazon Elastic File System: User Guide

Table of Contents

What is Amazon Elastic File System? 1
Are you a first-time user of Amazon EFS? 2
How it works 3
How Amazon EFS works with Amazon EC2 4
How Amazon EFS works with AWS Direct Connect and AWS Managed VPN 7
How Amazon EFS works with AWS Backup 8
Features 9
Authentication and access control 9
Data consistency in Amazon EFS 9
Availability and durability 10
Replication 16
Getting started 17
Prerequisites 17
Create file system and launch EC2 instance 17
Transfer files to your file system 18
Prerequisites 18
Clean up resources 19
Installing the EFS client 21
Dependencies for EFS tools 21
Supported distributions 22
Manually installing the EFS client 24
Installing the Amazon EFS client on Amazon EC2 Linux instances 24
Installing the Amazon EFS client on other Linux distributions 25
Installing the EFS client on EC2 Mac instances 25
Automatically installing the EFS client 26
What the Amazon EFS client does during installation 26
Systems Manager supported operating systems 27
Configuring AWS Systems Manager to install the EFS client 28
Installing and upgrading `botocore` 30
Upgrading `stunnel` 30
Resolving issues with installing stunnel 32
Enabling FIPS mode 33
Creating and managing resources 35
Implementation summary 36

Resource IDs ... 37
Creation token and idempotency ... 38
Creating file systems ... 38
Required IAM permissions for creating file systems ... 39
Configuration options for file systems ... 39
Deleting file systems ... 55
Creating file system policies ... 56
Creating access points ... 57
Deleting access points ... 60
Tagging resources ... 61
Tag basics ... 61
Tag restrictions ... 62
Using tags for access control ... 62
Tag your resources ... 63
Tutorial: Creating writable per-user subdirectories ... 63
Mounting file systems ... 66
Mounting considerations for Linux ... 66
Using the EFS mount helper ... 68
Mount settings used by EFS mount helper ... 70
Getting support logs ... 71
Prerequisites ... 71
Mounting on EC2 Linux ... 73
Mounting on EC2 Mac ... 75
Mounting from a different Region ... 77
Mounting One Zone file systems ... 77
Mounting with IAM authorization ... 81
Mounting with EFS access points ... 82
Mounting multiple EC2 instances ... 82
Mounting from another account or VPC ... 84
Using NFS ... 87
Prerequisites ... 88
NFS support ... 88
Installing the NFS client ... 89
Recommended NFS mount settings ... 91
Mounting on EC2 with DNS ... 93
Mounting with an IP address ... 95

Automatically mounting file systems 97
New EC2 Linux instances 97
Existing EC2 Linux instances 99
Linux and Mac instances using NFS 103
Unmounting file systems 106
Tutorial: Create and mount a file system using the AWS CLI 107
Prerequisites 108
Setting up the AWS CLI 109
Step 1: Create EC2 resources 110
Step 2: Create EFS resources 115
Step 3: Mount and test the file system 118
Step 4: Clean up 121
Tutorial: Mounting with on-premises clients 123
Prerequisites 125
Step 1: Create your EFS resources 127
Step 2: Install the NFS client 128
Step 3: Mount the Amazon EFS file system on your on-premises Client 129
Step 4: Clean up resources and protect your AWS account 131
Optional: Encrypting data in transit 131
Tutorial: Mount a file system from a different VPC 135
Prerequisites 135
Step 1: Determine the ID of the mount target's Availability Zone 136
Step 2: Determine the mount target IP address 137
Step 3: Add a host entry for the mount target 138
Step 4: Mount your file system using the EFS mount helper 138
Step 5: Clean up resources and protect your AWS account 140
Troubleshooting mount issues 141
File system mount on Windows instance fails 141
Access denied by server 141
Automatic mounting fails and the instance is unresponsive 141
Mounting multiple Amazon EFS file systems in /etc/fstab fails 142
Mount command fails with "wrong fs type" error message 143
Mount command fails with "incorrect mount option" error message 143
Mounting with access point fails 144
File system mount fails immediately after file system creation 144
File system mount hangs and then fails with timeout error 144

File system mount with NFS using DNS name fails 145
File system mount fails with "nfs not responding" 146
Mount target lifecycle state is stuck 146
Mount target lifecycle state shows error 146
Mount does not respond 147
Mounted client gets disconnected 147
Operations on newly mounted file system return "bad file handle" Error 148
Unmounting a file system fails 148
Transferring data 150
Using AWS DataSync 150
Using AWS Transfer Family 151
Prerequisites for using AWS Transfer Family with Amazon EFS 152
Configuring your EFS file system for AWS Transfer Family 152
Managing file systems 157
Understanding file system status 157
Managing mount targets 158
Mount targets and Availability Zones 159
Creating mount targets 162
Deleting mount targets 168
Changing mount target VPC 169
Changing mount target security groups 170
Managing throughput 171
Managing storage lifecycle 173
File system operations for lifecycle management 174
Configuring lifecycle policies 175
Managing access to encrypted file systems 177
Managing KMS keys for EFS file systems 178
Monitoring 180
Monitoring tools 181
Automated tools 181
Manual monitoring tools 182
File system metering 182
Metering objects 182
Metered file system size 183
Metering throughput 185
Viewing storage class size 186

Monitoring metrics with CloudWatch ... 188
CloudWatch metrics ... 188
Accessing CloudWatch metrics ... 194
Using CloudWatch metrics ... 196
Using metric math with CloudWatch metrics ... 197
Monitoring mount attempt successes and failures ... 203
Creating alarms ... 205
Logging API calls with CloudTrail ... 206
Amazon EFS information in CloudTrail ... 207
Understanding Amazon EFS log file entries ... 208
Amazon EFS log file entries for encrypted-at-rest file systems ... 215
Using billing and usage reports ... 216
Using cost allocation tags ... 216
User-defined cost allocation tags ... 217
Billing reports ... 217
Usage reports ... 218
Understanding billing and usage reports ... 219
Tracking operations in your usage reports ... 224
Performance specifications ... 225
Performance summary ... 225
Storage classes ... 227
Performance modes ... 227
Throughput modes ... 228
Choosing a throughput mode ... 229
Elastic throughput ... 229
Provisioned throughput ... 230
Restrictions on switching throughput and changing provisioned amount ... 233
Performance tips ... 233
Average I/O size ... 233
Optimizing workloads that demand high throughput and IOPS ... 233
Simultaneous connections ... 234
Request model ... 234
NFS client mount settings ... 234
Optimizing small-file performance ... 235
Optimizing directory performance ... 236
Optimizing the NFS read_ahead_kb size ... 236

- Troubleshooting performance issues 237
  - Unable to create an EFS file system 238
  - Access denied to allowed files on NFS file system 238
  - Errors when accessing the Amazon EFS console 238
  - Amazon EC2 instance hangs 239
  - Application writing large amounts of data hangs 239
  - Poor performance when opening many files in parallel 240
  - Custom NFS settings causing write delays 240
  - Creating backups with Oracle Recovery Manager is slow 241
- Troubleshooting AMI and kernel issues 241
  - Unable to chown 242
  - File system keeps performing operations repeatedly due to client bug 242
  - Deadlocked client 242
  - Listing files in a large directory takes a long time 243

Backing up and replicating data 244

- Backing up file systems 244
  - How AWS Backup works with Amazon EFS 245
  - Required IAM permissions 247
  - Backup performance 247
  - Managing automatic backups 247
- Replicating file systems 249
  - Costs 250
  - Replication performance 250
  - Required IAM permissions 251
  - Configuring replication to new file system 252
  - Configuring replication to existing file system 256
  - Replicating across AWS accounts 262
  - Viewing replication details 268
  - Deleting replication configurations 272
  - Using the replica 274

Securing data 276

- Encrypting data 277
  - AWS KMS 277
  - Encrypting data at rest 280
  - Encrypting data in transit 282
  - Troubleshooting encryption 284

- Identity and access management 286
  - Audience 286
  - Authenticating with identities 287
  - Managing access using policies 290
  - How Amazon Elastic File System works with IAM 293
  - Identity-based policy examples 299
  - Resource-based policy examples 304
  - AWS managed policies 306
  - Using tags with Amazon EFS 312
  - Using service-linked roles for Amazon EFS 315
  - Troubleshooting 319
- Controlling file system data access 321
  - Default File System Policy 322
  - EFS actions for clients 322
  - EFS condition keys for clients 322
  - File system policy examples 323
- Controlling network access 323
  - Using VPC security groups 323
  - Working with VPC endpoints 326
- NFS-level users, groups, and permissions 327
  - File and directory permissions 328
  - Example EFS file system use cases and permissions 329
  - User and group ID permissions for files and directories within a file system 330
  - No root squashing 331
  - Permissions caching 334
  - Changing file system object ownership 334
  - EFS access points 334
- Working with access points 335
  - Enforcing a user identity 336
  - Enforcing a root directory 336
  - Using access points in IAM policies 338
- Blocking public access to file systems 340
  - Blocking public access with AWS Transfer Family 340
  - The meaning of "public" 341
- Compliance validation 343
- Resilience 344

Network isolation ... 345
Quotas ... 346
Amazon EFS quotas that you can increase ... 346
Requesting a quota increase ... 349
Amazon EFS resource quotas that you cannot change ... 349
Quotas for NFS clients ... 351
Quotas for Amazon EFS file systems ... 351
Unsupported NFSv4.0 and 4.1 features ... 352
Additional considerations ... 354
Troubleshooting file operation errors related to quotas ... 354
Command fails with “Disk quota exceeded” error ... 354
Command fails with "I/O error" ... 355
Command fails with "File name is too long" error ... 355
Command fails with "File not found" error ... 356
Command fails with "Too many links" error ... 356
Command fails with "File too large" error ... 356
Amazon EFS API ... 357
API endpoints ... 357
IPv4 endpoints ... 358
Dual-stack (IPv4 and IPv6) endpoints ... 358
Specifying endpoints ... 359
API version ... 359
Related topics ... 359
Working with the query API request rate for Amazon EFS ... 359
Polling ... 360
Retries or batch processing ... 360
Calculating the sleep interval ... 360
Actions ... 360
CreateAccessPoint ... 362
CreateFileSystem ... 370
CreateMountTarget ... 386
CreateReplicationConfiguration ... 401
CreateTags ... 407
DeleteAccessPoint ... 410
DeleteFileSystem ... 412
DeleteFileSystemPolicy ... 416

DeleteMountTarget 419
DeleteReplicationConfiguration 422
DeleteTags 425
DescribeAccessPoints 428
DescribeAccountPreferences 433
DescribeBackupPolicy 436
DescribeFileSystemPolicy 439
DescribeFileSystems 443
DescribeLifecycleConfiguration 449
DescribeMountTargets 453
DescribeMountTargetSecurityGroups 459
DescribeReplicationConfigurations 463
DescribeTags 467
ListTagsForResource 472
ModifyMountTargetSecurityGroups 476
PutAccountPreferences 480
PutBackupPolicy 483
PutFileSystemPolicy 486
PutLifecycleConfiguration 492
TagResource 500
UntagResource 504
UpdateFileSystem 507
UpdateFileSystemProtection 515
Data Types 518
AccessPointDescription 520
BackupPolicy 523
CreationInfo 524
Destination 526
DestinationToCreate 529
FileSystemDescription 533
FileSystemProtectionDescription 538
FileSystemSize 539
LifecyclePolicy 541
MountTargetDescription 543
PosixUser 547
ReplicationConfigurationDescription 549

ResourceIdPreference 551
RootDirectory 552
Tag 554
Document history 555

What is Amazon Elastic File System?

Amazon Elastic File System (Amazon EFS) provides serverless, fully elastic file storage so that you can share file data without provisioning or managing storage capacity and performance. Amazon EFS is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files. Because Amazon EFS has a simple web services interface, you can create and configure file systems quickly and easily. The service manages all the file storage infrastructure for you, meaning that you can avoid the complexity of deploying, patching, and maintaining complex file system configurations.

Amazon EFS supports the Network File System version 4 (NFSv4.1 and NFSv4.0) protocol, so the applications and tools that you use today work seamlessly with Amazon EFS. Amazon EFS is accessible across most types of Amazon Web Services compute instances, including Amazon EC2, Amazon ECS, Amazon EKS, AWS Lambda, and AWS Fargate.

The service is designed to be highly scalable, highly available, and highly durable. Amazon EFS offers the following file system types to meet your availability and durability needs:

- *Regional* (Recommended) – Regional file systems (recommended) store data redundantly across multiple geographically separated Availability Zones within the same AWS Region. Storing data across multiple Availability Zones provides continuous availability to the data, even when one or more Availability Zones in an AWS Region are unavailable.
- *One Zone* – One Zone file systems store data within a single Availability Zone. Storing data in a single Availability Zone provides continuous availability to the data. In the unlikely case of the loss or damage to all or part of the Availability Zone, however, data that is stored in these types of file systems might be lost.

For more information about file system types, see EFS file system types.

Amazon EFS provides the throughput, IOPS, and low latency needed for a broad range of workloads. EFS file systems can grow to petabyte scale, drive high levels of throughput, and allow massively parallel access from compute instances to your data. For most workloads, we recommend using the default modes, which are the General Purpose performance mode and the Elastic throughput modes.

- *General Purpose* – The General Purpose performance mode is ideal for latency-sensitive applications, like web-serving environments, content-management systems, home directories, and general file serving.
- *Elastic* – The Elastic throughput mode is designed to automatically scale throughput performance up or down to meet the needs of your workload activity.

For more information about EFS performance and throughput modes, see Amazon EFS performance specifications.

Amazon EFS provides file-system-access semantics, such as strong data consistency and file locking. For more information, see Data consistency in Amazon EFS. Amazon EFS also supports controlling access to your file systems through Portable Operating System Interface (POSIX) permissions. For more information, see Securing your data in Amazon EFS.

Amazon EFS supports authentication, authorization, and encryption capabilities to help you meet your security and compliance requirements. Amazon EFS supports two forms of encryption for file systems: encryption in transit and encryption at rest. You can enable encryption at rest when creating an EFS file system. If you do, all of your data and metadata is encrypted. You can enable encryption in transit when you mount the file system. NFS client access to Amazon EFS is controlled by both AWS Identity and Access Management (IAM) policies and network security policies, such as security groups. For more information, see Encrypting data in Amazon EFS, Identity and access management for Amazon EFS, and Controlling network access to EFS file systems for NFS clients.

> **Note**
>
> Using Amazon EFS with Microsoft Windows–based Amazon EC2 instances is not supported.

Are you a first-time user of Amazon EFS?

If you are a first-time user of Amazon EFS, we recommend that you read the following sections in order:

1. For an Amazon EFS product and pricing overview, see Amazon EFS.
2. For an Amazon EFS technical overview, see How Amazon EFS works.
3. Try the Getting started exercise.

If you want to learn more about Amazon EFS, the following topics discuss the service in greater detail:

- Creating and managing EFS resources
- Managing EFS file systems
- Amazon EFS API

How Amazon EFS works

Amazon Elastic File System (EFS) provides a simple, serverless, set-and-forget elastic file system. With Amazon EFS, you can create a file system, mount the file system on an Amazon EC2 instance, and then read and write data to and from your file system. You can mount an EFS file system in your virtual private cloud (VPC), through the Network File System versions 4.0 and 4.1 (NFSv4) protocol. We recommend using a current generation Linux NFSv4.1 client, such as those found in the latest Amazon Linux, Amazon Linux 2, Red Hat, Ubuntu, and macOS Big Sur AMIs, in conjunction with the EFS mount helper. For instructions, see Installing the Amazon EFS client.

For a list of Amazon EC2 Linux and macOS Amazon Machine Images (AMIs) that support this protocol, see NFS support. For some AMIs, you must install an NFS client to mount your file system on your Amazon EC2 instance. For instructions, see Installing the NFS client.

You can access your EFS file system concurrently from multiple NFS clients, so applications that scale beyond a single connection can access a file system. Amazon EC2 and other AWS compute instances running in multiple Availability Zones within the same AWS Region can access the file system, so that many users can access and share a common data source.

For a list of AWS Regions where you can create an EFS file system, see the Amazon Web Services General Reference.

To access your EFS file system in a VPC, you create one or more mount targets in the VPC. A *mount target* provides an IP address for an NFSv4 endpoint at which you can mount an EFS file system. You mount your file system using its Domain Name Service (DNS) name, which resolves to the IP address of the EFS mount target in the same Availability Zone as your EC2 instance. You can create one mount target in each Availability Zone in an AWS Region. If there are multiple subnets in an Availability Zone in your VPC, you create a mount target in one of the subnets. Then all EC2 instances in that Availability Zone share that mount target.

> **Note**
>
> An EFS file system can have mount targets in only one VPC at a time.

Mount targets themselves are designed to be highly available. As you design for high availability and failover to other Availability Zones, keep in mind that while the IP addresses and DNS for your mount targets in each Availability Zone are static, they are redundant components backed by multiple resources. For more information about mount targets, see Managing mount targets.

After mounting the file system by using its DNS name, you use it like any other POSIX-compliant file system. For information about NFS-level permissions and related considerations, see Network File System (NFS) level users, groups, and permissions.

You can mount your EFS file systems on your on-premises data center servers when connected to your Amazon VPC with AWS Direct Connect or AWS VPN. You can mount your EFS file systems on on-premises servers to migrate datasets to EFS, enable cloud bursting scenarios, or back up your on-premises data to Amazon EFS.

Following, you can find a description about how Amazon EFS works with other services.

Topics

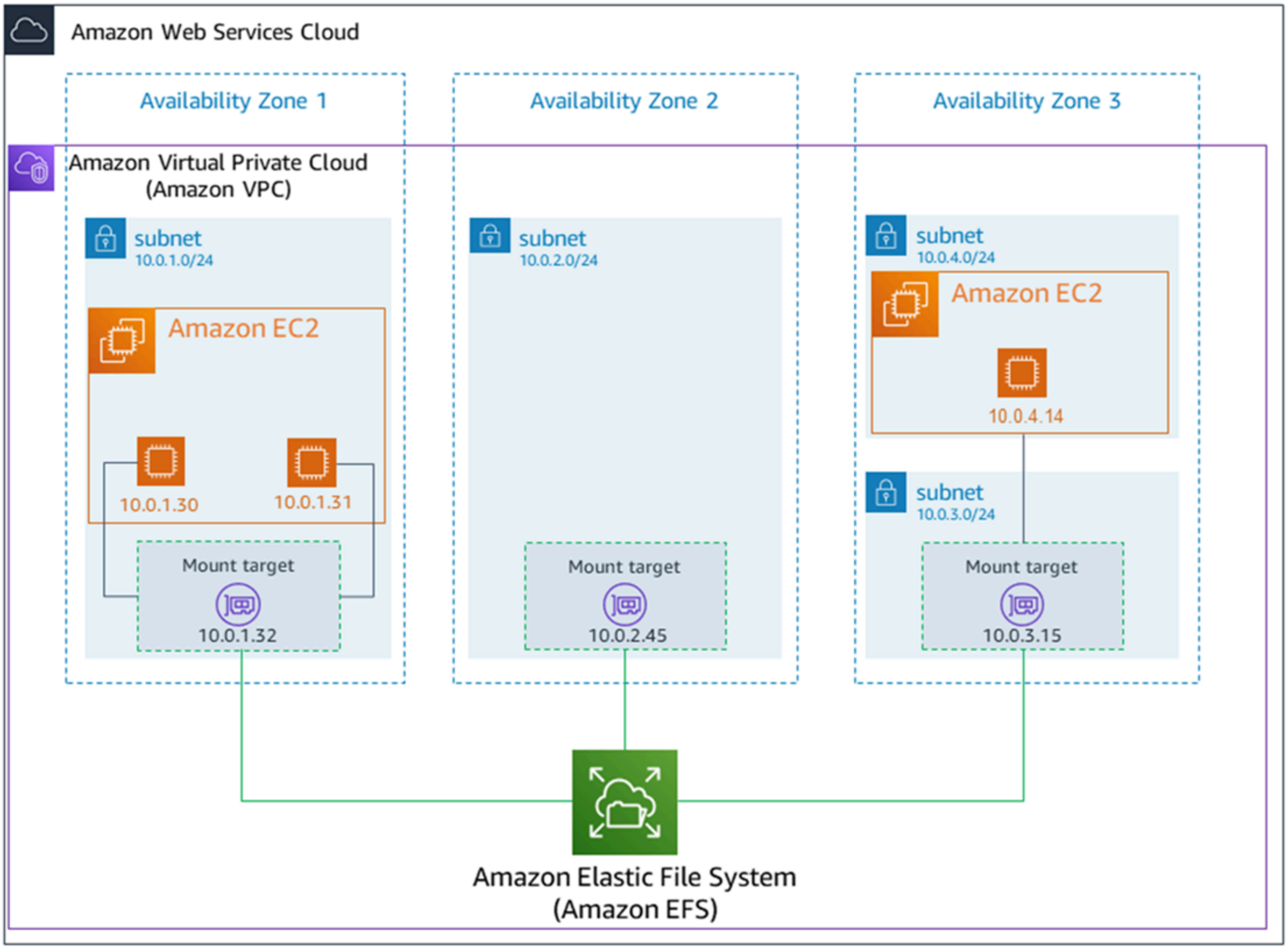
- How Amazon EFS works with Amazon EC2
- How Amazon EFS works with AWS Direct Connect and AWS Managed VPN
- How Amazon EFS works with AWS Backup

How Amazon EFS works with Amazon EC2

This section explains how Amazon EFS Regional and One Zone file systems are mounted to EC2 instances in an Amazon VPC.

Regional EFS file systems

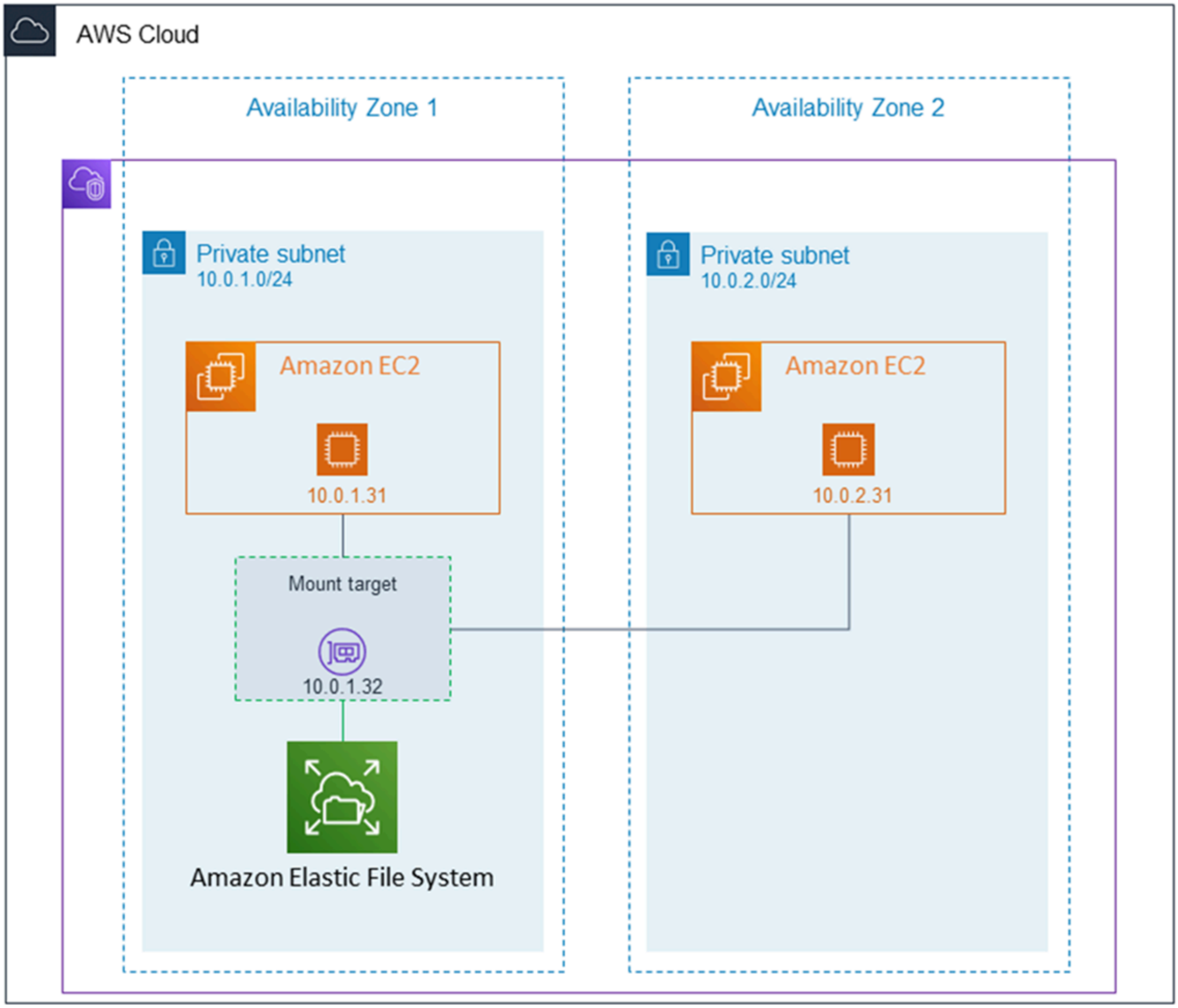
The following illustration shows multiple EC2 instances accessing an Amazon EFS file system that is configured for multiple Availability Zones in an AWS Region.



In this illustration, the virtual private cloud (VPC) has three Availability Zones. Because the file system is Regional, a mount target was created in each Availability Zone. We recommend that you access the file system from a mount target within the same Availability Zone for performance and cost reasons. One of the Availability Zones has two subnets. However, a mount target is created in only one of the subnets. For more information, see Mounting EFS file systems using the EFS mount helper.

One Zone EFS file systems

The following illustration shows multiple EC2 instances accessing a One Zone file system from different Availability Zones in a single AWS Region.



In this illustration, the VPC has two Availability Zones, each with one subnet. Because the file system type is One Zone, it can only have a single mount target. For better performance and cost, we recommend that you access the file system from a mount target in the same Availability Zone as the EC2 instance that you're mounting it on.

In this example, the EC2 instance in the us-west-2c Availability Zone will pay EC2 data access charges for accessing a mount target in a different Availability Zone. For more information, see [Mounting One Zone file systems](#).

How Amazon EFS works with AWS Direct Connect and AWS Managed VPN

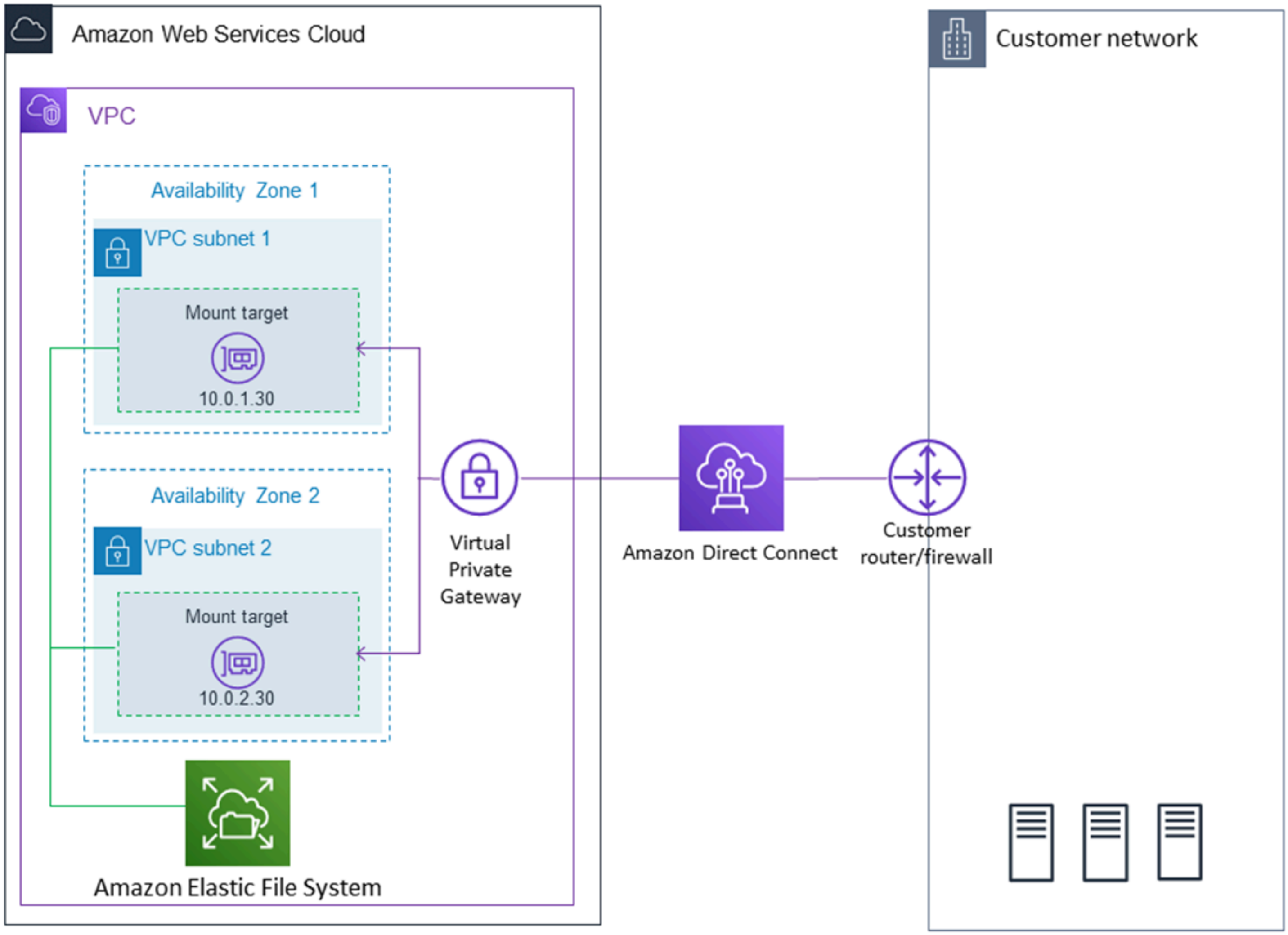
By using an Amazon EFS file system mounted on an on-premises server, you can migrate on-premises data into the AWS Cloud hosted in an Amazon EFS file system. You can also take advantage of bursting. In other words, you can move data from your on-premises servers into Amazon EFS and analyze it on a fleet of Amazon EC2 instances in your Amazon VPC. You can then store the results permanently in your file system or move the results back to your on-premises server.

Keep the following considerations in mind when using Amazon EFS with an on-premises server:

- Your on-premises server must have a Linux-based operating system. We recommend Linux kernel version 4.0 or later.
- For the sake of simplicity, we recommend mounting an Amazon EFS file system on an on-premises server using a mount target IP address instead of a DNS name.

There is no additional cost for on-premises access to your Amazon EFS file systems. You are charged for the AWS Direct Connect connection to your Amazon VPC. For more information, see AWS Direct Connect pricing.

The following illustration shows an example of how to access an Amazon EFS file system from on-premises (the on-premises servers have the file systems mounted).



You can use any mount target in your VPC if you can reach that mount target's subnet by using an AWS Direct Connect connection between your on-premises server and VPC. To access Amazon EFS from an on-premises server, add a rule to your mount target security group to allow inbound traffic to the NFS port (2049) from your on-premises server. For more information, including detailed procedures, see Prerequisites.

How Amazon EFS works with AWS Backup

For a comprehensive backup implementation for your file systems, you can use Amazon EFS with AWS Backup. AWS Backup is a fully managed backup service that makes it easy to centralize and automate data backup across AWS services in the cloud and on-premises. Using AWS Backup, you can centrally configure backup policies and monitor backup activity for your AWS resources. Amazon EFS always prioritizes file system operations over backup operations. To learn more about backing up EFS file systems using AWS Backup, see Backing up EFS file systems.

Features of Amazon EFS

Following are features of Amazon EFS.

Topics

- Authentication and access control
- Data consistency in Amazon EFS
- Availability and durability of EFS file systems
- Replication

Authentication and access control

You must have valid credentials to use the Amazon EFS management console and to make Amazon EFS API requests, such as create a file system. In addition, you must also have permissions to create or access other EFS and AWS resources.

Users and roles that you create in AWS Identity and Access Management (IAM) must be granted permissions to create or access resources. For more information about permissions, see Identity and access management for Amazon EFS.

IAM authorization for NFS clients is an additional security option for Amazon EFS that uses IAM to simplify access management for Network File System (NFS) clients at scale. With IAM authorization for NFS clients, you can use IAM to manage access to an EFS file system in an inherently scalable way. IAM authorization for NFS clients is also optimized for cloud environments. For more information on using IAM authorization for NFS clients, see Using IAM to control access to file systems.

Data consistency in Amazon EFS

Amazon EFS provides the close-to-open consistency semantics that applications expect from NFS.

In Amazon EFS, write operations for Regional file systems are durably stored across Availability Zones in these situations:

- An application performs a synchronous write operation (for example, using the `open` Linux command with the `O_DIRECT` flag, or the `fsync` Linux command).
- An application closes a file.

Depending on the access pattern, Amazon EFS can provide stronger consistency guarantees than close-to-open semantics. Applications that perform synchronous data access and perform non-appending writes have read-after-write consistency for data access.

File locking

NFS client applications can use NFS version 4 file locking (including byte-range locking) for read and write operations on Amazon EFS files.

Remember the following about how Amazon EFS locks files:

- Amazon EFS only supports advisory locking and read/write operations don't check for conflicting locks before executing. For example, to avoid file synchronization issues with atomic operations, your application must be aware of NFS semantics (such as close-to-open consistency).
- Any one particular file can have up to 512 locks across all instances connected and users accessing the file.

Availability and durability of EFS file systems

This section describes the file system types and storage class options for Amazon Elastic File System (Amazon EFS) file systems.

EFS file system types

Amazon EFS offers Regional and One Zone file system types.

- **Regional –** Regional file systems (recommended) store data redundantly across multiple geographically separated Availability Zones within the same AWS Region. Storing data across multiple Availability Zones provides continuous availability to the data, even when one or more Availability Zones in an AWS Region are unavailable.
- **One Zone –** One Zone file systems store data within a single Availability Zone. Storing data in a single Availability Zone provides continuous availability to the data. In the unlikely case of the loss or damage to all or part of the Availability Zone, however, data that is stored in these types of file systems might be lost.

  In the unlikely case of the loss or damage to all or part of an AWS Availability Zone, data in a One Zone storage class may be lost. For example, events like fire and water damage could result in data loss. Apart from these types of events, our One Zone storage classes use similar engineering designs as our Regional storage classes to protect objects from independent disk, host, and rack-level failures, and each are designed to deliver 99.999999999% data durability.

For added data protection, Amazon EFS automatically backs up One Zone file systems with AWS Backup. You can restore file system backups to any operational Availability Zone within an AWS Region, or you can restore them to a different AWS Region. EFS file system backups that are created and managed using AWS Backup are replicated to three Availability Zones and are designed for durability. For more information, see Resilience in AWS Backup.

> **Note**
>
> One Zone file systems are available to only certain Availability Zones. For a table that lists the Availability Zones in which you can use One Zone file systems, see Supported Availability Zones for One Zone file systems.

The following table compares the file system types, including their availability, durability, and other considerations.

File system type	Designed for	Durability (designed for)	Availability	Availability Zones	Other considera tions
Regional	Data requiring the highest durability and availability.	99.999999 999% (11 9s)	99.99%	>=3	None
One Zone	Data that doesn't require the highest durability and availability.	99.999999 999% (11 9s)	99.99%	1	Not resilient to the loss of the Availability Zone

Supported Availability Zones for One Zone file systems

One Zone file systems are available to only certain Availability Zones. The following table lists the AWS Region and the AZ IDs for each Availability Zone in which you can use One Zone file systems. To see the mapping of AZ IDs to Availability Zones in your account, see Availability Zone IDs for your AWS Resources in the *AWS Resource Access Manager User Guide*.

Availability Zones that support One Zone file systems

AWS Region Name	AWS Region Code	Supported AZ IDs
US East (Ohio)	us-east-2	use2-az1, use2-az2, use2-az3
US East (N. Virginia)	us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
US West (N. California)	us-west-1	usw1-az1, usw1-az3
US West (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Africa (Cape Town)	af-south-1	afs1-az1,afs1-az2,afs1-az3
Asia Pacific (Hong Kong)	ap-east-1	ape1-az1, ape1-az2,ape1-az3
Asia Pacific (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Asia Pacific (Osaka)	ap-northeast-3	apne3-az1, apne3-az2, apne3-az3
Asia Pacific (Seoul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asia Pacific (Singapore)	ap-southeast-1	apse1-az1, apse1-az2
Asia Pacific (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3
Asia Pacific (Tokyo)	ap-northeast-1	apne1-az1,apne1-az4
Canada (Central)	ca-central-1	cac1-az1, cac1-az2

AWS Region Name	AWS Region Code	Supported AZ IDs
China (Beijing)	cn-north-1	cnn1-az1, cnn1-az2
China (Ningxia)	cn-northwest-1	cnnw1-az1, cnnw1-az2, cnnw1-az3
Europe (Frankfurt)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Europe (Ireland)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Europe (London)	eu-west-2	euw2-az1, euw2-az2
Europe (Milan)	eu-south-1	eus1-az1, eus1-az2, eus1-az3
Europe (Paris)	eu-west-3	euw3-az1, euw3-az3
Europe (Stockholm)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Middle East (Bahrain)	me-south-1	mes1-az1, mes1-az2, mes1-az3
South America (São Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3
AWS GovCloud (US-East)	us-gov-east-1	usge1-az1, usge1-az2, usge1-az3
AWS GovCloud (US-West)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

EFS storage classes

Amazon EFS offers different storage classes that are designed for the most effective storage depending on use cases.

- **EFS Standard –** The EFS Standard storage class uses solid state drive (SSD) storage to deliver the lowest levels of latency for frequently accessed files. New file system data is first written to the

EFS Standard storage class and then can be tiered to the EFS Infrequent Access and EFS Archive storage classes by using lifecycle management.

- **EFS Infrequent Access (IA)** – A cost-optimized storage class for data that is accessed only a few times each quarter.
- **EFS Archive** – A cost-optimized storage class for data that is accessed a few times each year or less.

  The EFS Archive storage class is supported on EFS file systems with Elastic throughput. You cannot update your file system's throughput to Bursting or Provisioned once the file system has data in the Archive storage class.

Comparing storage classes

The following table compares the storage classes. For more details about the performance of each storage class, see Amazon EFS performance specifications.

Storage class	Designed for	First byte read latency	Durability (designed for)[1]	Availability SLA	Availability Zones	Minimum billing charge per file[2]	Minir stora durat
EFS Standard	Active data requiring fast sub-millisecond latency performance	Sub-millisecond		99.99% (Regional) 99.9% (One Zone)	=>3 (Regional) 1 (One Zone)	Not applicable	Not applic e
EFS Infrequent Access	Inactive data that is accessed only a few times each quarter.	Tens of milliseconds	99.999999999% (11 9's)			128 KiB	Not applic e
EFS Archive	Inactive data that is accessed a few times each year or less	Tens of milliseconds		99.9% (Regional)	=>3 (Regional)	128 KiB	90 days

[1]Because One Zone file systems store data in a single AWS Availability Zone, data that is stored in these types of file systems might be lost in the event of a disaster or other fault that affects all copies of the data within the Availability Zone, or in the event of Availability Zone destruction.

[2]Lifecycle policies updated on or after 12 PM PT, November 26, 2023 will tier files that are smaller than 128 KiB to the IA class. For more information about how Amazon EFS meters and bills for individual files and metadata, see How Amazon EFS reports file system and object sizes.

Storage class billing

You are billed for the amount of data in each storage class. You are also billed data access charges when files in IA or Archive storage are read, or for data that transitions between storage classes using lifecycle management. The AWS bill displays the capacity for each storage class and the metered access against the file system's storage class. To learn more, see Amazon EFS Pricing.

Additionally, Infrequent Access (IA) and Archive storage classes have a minimum billing charge per file of 128 KiB. Support for files smaller than 128 KiB is only available for lifecycle policies updated on or after 12:00 PM PT, November 26, 2023. For more information on how Amazon EFS meters and bills for individual files and metadata, see How Amazon EFS reports file system and object sizes.

Additional billing considerations apply depending on throughput mode.

- For file systems using Elastic throughput, you are billed for the total monthly amount of metadata and data transferred for your file systems, independent of the storage charges.
- For file systems using Provisioned throughput, you are billed for the throughput provisioned above what you are provided based on the amount of data that is in the EFS Standard storage class.
- For file systems using Bursting throughput, the allowed throughput is determined based on the amount of the data stored in the EFS Standard storage class only.

For more information about EFS throughput modes, see Throughput modes.

> **Note**
>
> You don't incur data access charges when using AWS Backup to back up lifecycle management-enabled EFS file systems. To learn more about AWS Backup with Amazon EFS, see Backing up EFS file systems.

Lifecycle management

To manage your file systems so that they are stored cost effectively throughout their lifecycle, use lifecycle management. Lifecycle management automatically transitions data between storage classes according to the lifecycle configuration defined for the file system. The lifecycle configuration is a set of *lifecycle policies* that define when to transition the file system data to another storage class. For more information, see Managing storage lifecycle.

Replication

You can create a replica of your Amazon EFS file system in the AWS Region of your preference using replication. Replication automatically and transparently replicates the data and metadata on your EFS file system to a new destination EFS file system that is created in an AWS Region that you choose. EFS automatically keeps the source and destination file systems synchronized. Replication is continual and designed to provide a recovery point objective (RPO) and a recovery time objective (RTO) of minutes. These features assist you in meeting your compliance and business continuity goals. For more information, see Replicating EFS file systems.

Getting started with Amazon EFS

If you are using Amazon Elastic File System (Amazon EFS) for the first time, complete the following steps to get started with your first EFS file system.

1. Review the prerequisites for getting started
2. Create your EFS file system and launch your EC2 instance
3. Transfer files to your EFS file system using AWS DataSync
4. Clean up resources and protect your AWS account

Prerequisites

Before completing the getting started steps, make sure you have the following requirements:

- You are set up with Amazon EC2 and are familiar with launching EC2 instances. You need an AWS account, a user with administrative access, a key pair, and a security group. For more information, see Get started with Amazon EC2 in the *Amazon EC2 User Guide*.
- Your Amazon Virtual Private Cloud (Amazon VPC), EC2, and EFS resources are all in the same AWS Region and that you have a default VPC in the Region. If you don't have a default VPC, or if you want to mount your file system from a new VPC with new or existing security groups, see Using VPC security groups.
- You haven't changed the default inbound access rule for the default security group.

You can also perform a similar getting started exercise using AWS Command Line Interface (AWS CLI) commands to make the EFS API calls. For more information, see Tutorial: Create an EFS file system and mount it on an EC2 instance using the AWS CLI.

Create your EFS file system and launch your EC2 instance

After making sure that you have the prerequisites for this getting started exercise, you can create your EFS file system and launch your EC2 instance. The quickest way to complete all of the necessary steps to get started with your first EFS file system is to use the EC2 new launch wizard during instance launch.

> **Note**
>
> You can't use Amazon EFS with Microsoft Windows–based EC2 instances.

To create your EFS file system and launch your EC2 instance using the EC2 launch wizard

For instructions on creating and mounting your EFS file system when creating an EC2 instance launch, see Use Amazon EFS with Amazon EC2 in the *Amazon EC2 User Guide*.

The following are the steps that you'll perform when creating an EFS file system during instance launch.

1. Create an EC2 instance running on a Linux operating system using the key pair and network settings you choose.
2. Create a shared EFS file system that has the recommended settings and is automatically mounted to the EC2 instance.
3. Launch the EC2 instance so that the EFS file system is readily available for file transfers.

Alternatively, in the Amazon EFS console, you can create file systems with recommended settings or custom settings. You can also use the AWS CLI and API to create file systems. For more information about all of your options for creating a file system, see Creating EFS file systems.

Transfer files to your EFS file system using AWS DataSync

After creating an EFS file system, you can transfer files to it from an existing file system by using AWS DataSync. DataSync is a data transfer service that simplifies, automates, and accelerates moving and replicating data between on-premises storage systems and AWS storage services over the internet or AWS Direct Connect. DataSync can transfer your file data, and also file system metadata such as ownership, timestamps, and access permissions.

For more information about DataSync, see AWS DataSync.

Prerequisites

Before transferring files to the EFS file system, make sure you have the following:

- A source NFS file system that you can transfer files from. This source system needs to be accessible over NFS version 3, version 4, or 4.1. Example file systems include those located in an on-premises data center, self-managed in-cloud file systems, and EFS file systems.
- You are set up to use DataSync. To learn more, see Getting started with AWS DataSync in the *AWS DataSync User Guide*.

To transfer files to your EFS file system using AWS DataSync

For instructions on using DataSync to transfer files to an EFS file system, see Transferring your data with AWS DataSync in the *AWS DataSync User Guide*.

The following are the steps that you'll perform when transferring files to the EFS file system using DataSync.

1. Connect to your EC2 instance. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Download, deploy, and activate an agent in your environment.
3. Create and configure a source and destination location.
4. Create and configure a task.
5. Run the task to transfer files from the source to the destination.

Clean up resources and protect your AWS account

When you're finished with this getting started exercise, perform the following steps to clean up your resources and protect your AWS account.

To clean up resources and protect your account

1. Connect to your EC2 instance. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Unmount the EFS file system with the following command.

   ```
   $ sudo umount efs
   ```

3. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
4. Delete the EFS file system that you created in the first step of the getting started exercise.

   a. Choose the EFS file system that you want to delete from the list of file systems.
   b. For **Actions**, choose **Delete file system**.
   c. In the **Permanently delete file system** dialog box, type the file system ID for the EFS file system that you want to delete, and then choose **Delete File System**.

5. Terminate the EC2 instance that you launched for this getting started exercise. For instructions, see Terminate Amazon EC2 instances in the *AWS IAM Identity Center User Guide*.
6. If you created a security group for this getting started exercise, then delete it. For instructions, see Delete a security group in the *AWS IAM Identity Center User Guide*.

> **Warning**
>
> Don't delete the default security group for your VPC.

Installing the Amazon EFS client

We recommend that you install the Amazon EFS client (`amazon-efs-utils`), an open-source collection of tools for Amazon EFS. The Amazon EFS client includes a *mount helper*, which is a program that helps simplify mounting EFS file systems. The client also enables the ability to use Amazon CloudWatch to monitor an EFS file system's mount status, and it includes tooling that makes it easier to perform encryption of data in transit for Amazon EFS file systems.

You can manually install the Amazon EFS client on Amazon EC2 (EC2) instances running supported distributions. For certain supported operating systems, you can alternatively configure AWS Systems Manager to automatically install or update the package. For a list of distributions that you can use with AWS Systems Manager, see Systems Manager supported operating systems.

> **Important**
>
> We recommend that you always use the most current version of `amazon-efs-utils` to ensure access to its full capabilities. For example, mounting with IPv6 addresses is supported in versions 2.3 or later, but not in earlier versions.

Topics

- Dependencies for EFS tools
- Supported distributions
- Manually installing the Amazon EFS client
- Automatically installing or updating Amazon EFS client using AWS Systems Manager
- Installing and upgrading botocore
- Upgrading stunnel
- Enabling FIPS mode

Dependencies for EFS tools

The following dependencies exist for `amazon-efs-utils` and are installed when you install the `amazon-efs-utils` package:

- NFS client

  - `nfs-utils` for RHEL, CentOS, Amazon Linux, and Fedora distributions
  - `nfs-common` for Debian and Ubuntu distributions
- Network relay (stunnel package, version 4.56 or later)
- Python (version 3.4 or later)
- OpenSSL 1.0.2 or newer

> **Note**
>
> By default, when using the EFS mount helper with Transport Layer Security (TLS), the mount helper enforces certificate hostname checking. The EFS mount helper uses the `stunnel` program for its TLS functionality. Some versions of Linux don't include a version of `stunnel` that supports these TLS features by default. When using one of those Linux versions, mounting an EFS file system using TLS fails.

After you've installed the `amazon-efs-utils` package, upgrade stunnel. See Upgrading `stunnel`.

You can use AWS Systems Manager to manage Amazon EFS clients and automate the tasks required to install or update the amazon-efs-utils package on your EC2 instances. For more information, see Automatically installing or updating Amazon EFS client using AWS Systems Manager.

For issues with encryption, see Troubleshooting encryption.

Supported distributions

The Amazon EFS client has been verified against the following Linux and Mac distributions:

Distribution	Package type	`init` system
Amazon Linux 2023 (AL2023)	rpm	`systemd`
Amazon Linux 2 (AL2)	rpm	`systemd`
CentOS 8	rpm	`systemd`

Distribution	Package type	init system
Amazon Linux 1 (AL1) 2017.09 **Note** AL1 AMI reached its end-of-life on December 31, 2023 and is not supported for `amazon-efs-utils` packages released in April 2024 or later (version 2.0 and later).	rpm	`upstart`
Debian 11	deb	`systemd`
Fedora 29 - 32	rpm	`systemd`
macOS Big Sur		`launchd`
macOS Monterey		`launchd`
macOS Ventura		`launchd`
macOS Sonoma		`launchd`
OpenSUSE Leap, Tumbleweed	rpm	`systemd`
Oracle8	rpm	`systemd`
Red Hat Enterprise Linux (RHEL) 7, 8, 9	rpm	`systemd`
SUSE Linux Enterprise Server (SLES) 12, 15	rpm	`systemd`
Ubuntu 16.04 LTS, 18.04 LTS, 20.04 LTS, 22.04 LTS	deb	`systemd`

For a complete list of supported distributions that the package has been verified against, see the `amazon-efs-utils` README on Github.

Manually installing the Amazon EFS client

You can manually install the Amazon EFS client on Amazon EC2 (EC2) Linux instances and on EC2 Mac instances running macOS Big Sur, macOS Monterey, and macOS Ventura. For a list of the distributions that support Amazon EFS client, see Supported distributions

The installation procedures for supported operating systems are described in the following sections.

Topics

- Installing the Amazon EFS client on Amazon EC2 Linux instances
- Installing the Amazon EFS client on other Linux distributions
- Installing the Amazon EFS client on EC2 Mac instances running macOS Big Sur, macOS Monterey, or macOS Ventura

Installing the Amazon EFS client on Amazon EC2 Linux instances

The `amazon-efs-utils` package for installing on Amazon EC2 Linux instances from the following locations:

- The Amazon Machine Images (AMI) package repositories for Amazon Linux. The following instructions are for installing the `amazon-efs-utils` package from the AMI package repositories.
- The AWS efs-utils GitHub repository. For more information about installing the `amazon-efs-utils` package from GitHub, see Installing the Amazon EFS client on other Linux distributions.

> **Note**
>
> - If you're using AWS Direct Connect, you can find installation instructions in Prerequisites.
> - The Amazon Linux 1 (AL1) AMI reached its end-of-life on December 31, 2023 and is not supported for `amazon-efs-utils` packages released in April 2024 and later (version 2.0 and later). We recommend that you upgrade applications to Amazon Linux 2023 (AL2023), which includes long-term support until 2028.

To install the `amazon-efs-utils` package from the AMI package repository on EC2 Linux instances

1. Make sure that you've created an AL2023 or Amazon Linux 2 (AL2) EC2 instance. For information on how to do this, see Step 1: Launch an instance.
2. Access the terminal for your instance through Secure Shell (SSH), and log in with the appropriate user name. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
3. Run the following command to install the `amazon-efs-utils` package.

```
sudo yum install -y amazon-efs-utils
```

Installing the Amazon EFS client on other Linux distributions

If you don't want to get the `amazon-efs-utils` package from the Amazon Linux AMI package repositories, it is also available on GitHub.

After you clone the package, you can build and install `amazon-efs-utils` using one of the following methods, depending on the package type supported by your Linux distribution:

- **RPM –** This package type is supported by AL2023, Amazon Linux 2, Red Hat Linux, CentOS, and similar.
- **DEB –** This package type is supported by Ubuntu, Debian, and similar.

For instructions on installing the `amazon-efs-utils` package for other Linux distributions, see On other Linux distributions in the `amazon-efs-utils` README on Github.

Installing the Amazon EFS client on EC2 Mac instances running macOS Big Sur, macOS Monterey, or macOS Ventura

The `amazon-efs-utils` package is available for installation on EC2 Mac instances running macOS Big Sur, macOS Monterey, or macOS Ventura.

For instructions on installing the `amazon-efs-utils` package on Mac instances, see On MacOS Big Sur, macOS Monterey, macOS Sonoma and macOS Ventura distribution in the `amazon-efs-utils` README on Github.

Next steps

After installs `amazon-efs-utils` on your EC2 instance, proceed to the next steps for mounting your file system:

- [Install `botocore`] so that you can use Amazon CloudWatch to monitor your file system's mount status.
- [Upgrade to the latest version of `stunnel`] to enable encryption of data in transit.
- [Mount your file system] using the EFS mount helper.

Automatically installing or updating Amazon EFS client using AWS Systems Manager

You can use AWS Systems Manager to simplify the management of the Amazon EFS client (`amazon-efs-utils`). AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. With AWS Systems Manager you can automate the tasks required to install or update the `amazon-efs-utils` package on your Amazon EC2 (EC2) instances. The Systems Manager capabilities like Distributor and State Manager enable you to automate the following processes:

- Maintaining version control over the Amazon EFS client.
- Centrally storing and systematically distributing the Amazon EFS client to your Amazon EC2 instances.
- Automate the process of keeping your EC2 instances in a defined state.

For more information, see the *AWS Systems Manager User Guide*.

What the Amazon EFS client does during installation

You use the Amazon EFS client to automate monitoring Amazon CloudWatch logs for file system mount status and upgrade `stunnel` to the latest version for selected Linux distributions. When you install the Amazon EFS client on your Amazon EC2 instances using Systems Manager, it takes the following actions:

- Installs the `botocore` package using the same steps described in Installing and upgrading `botocore`. The Amazon EFS client uses `botocore` to monitor the EFS file system mount status.

- Enables the monitoring of EFS file system mount status in CloudWatch logs by updating `efs-utils.conf`. For more information, see Monitoring mount attempt successes and failures.
- For EC2 instances running `RHEL7` or `CentOS7`, the Amazon EFS client automatically upgrades `stunnel` as described in Upgrading `stunnel`. Upgrading `stunnel` is required in order to successfully mount an EFS file system using TLS, and the `stunnel` version shipped with `RHEL7` and `CentOS7` does not support the Amazon EFS client (`amazon-efs-utils`).

Systems Manager supported operating systems

Your EC2 instances must be running one of the following operating systems in order to be used with AWS Systems Manager to automatically update or install the Amazon EFS client.

Platform	Platform version	Architecture
Amazon Linux 2023 (AL2023)	AL2023	x86_64, arm64 (Graviton2 or later processors)
Amazon Linux 2 (AL2)	2.0	x86_64, arm64 (Amazon Linux 2, A1 instance types)
Amazon Linux 1 (AL1) **Note** Amazon Linux 1 (AL1) AMI reached its end-of-life on December 31, 2023 and is not supported for `amazon-efs-utils` packages released in April 2024 and later (version 2.0 and later). We recommend that you upgrade applications to Amazon Linux 2023 (AL2023), which	2017.09, 2018.03	x86_64

Platform	Platform version	Architecture
includes long-term support until 2028.		
CentOS	7, 8	x86_64
Red Hat Enterprise Linux (RHEL)	7, 8	x86_64, arm64 (RHEL 7.6 and later, A1 instance types)
SUSE Linux Enterprise Server (SLES)	12, 15	x86_64
Ubuntu Server	16.04, 18.04, 20.04	x86_64, arm64 (Ubuntu Server 16 and later, A1 instance types)

Configuring AWS Systems Manager to install the EFS client

There are two one-time configurations required to set up Systems Manager to automatically install or update the `amazon-efs-utils` package.

1. Configure an AWS Identity and Access Management (IAM) instance profile with the required permissions.
2. Configure an Association (including the schedule) used for installation or updates by the State Manager.

Step 1: Configure an IAM instance profile with the required permissions

By default, AWS Systems Manager doesn't have permission to manage your Amazon EFS clients and install or update the amazon-efs-utils package. You must grant access to Systems Manager by using an AWS Identity and Access Management (IAM) instance profile. An instance profile is a container that passes IAM role information to an Amazon EC2 (EC2) instance at launch.

Use the `AmazonElasticFileSystemsUtils` AWS managed permission policy to assign the appropriate permissions to roles. You can create a new role for your instance profile or add the `AmazonElasticFileSystemsUtils` permission policy to an existing role. You must then use this instance profile to launch your EC2 instances. For more information, see Configure instance permissions required for Systems Manager.

Step 2: Configure an association used by State Manager

The `amazon-efs-utils` package is included with Distributor and is ready for you to deploy to managed EC2 instances. To view the latest version of `amazon-efs-utils` that is available for installation, you can use the AWS Systems Manager console or your preferred AWS command line tool. To access Distributor, open the https://console.aws.amazon.com/systems-manager/ and choose **Distributor** in the left navigation pane. Locate **AmazonEFSUtils** in the **Owned by Amazon** section. Choose **AmazonEFSUtils** to see the package details. For more information, see View packages.

Using State Manager, you can install or update the `amazon-efs-utils` package on your managed EC2 instances immediately or on a schedule. Additionally, you can ensure that `amazon-efs-utils` is automatically installed on new EC2 instances. For more information about installation or updating packages using Distributor and State Manager, see Working with Distributor.

To automatically install or update the amazon-efs-utils package on instances using the Systems Manager console, see Scheduling a package installation or update (console). This will prompt you to create an association for State Manager, which defines the state you want to apply to a set of instances. Use the following inputs when you create your association:

- For **Parameters** choose **Action** > **Install** and **Installation Type** > **In-place update**.
- For **Targets** the recommended setting is **Choose all instances** to register all new and existing EC2 instances as targets to automatically install or update **AmazonEFSUtils**. Alternatively, you can specify instance tags, select instances manually, or choose a resource group to apply the association to a subset of instances. If you specify instance tags, you must launch your EC2 instances with the tags to allows AWS Systems Manager to automatically install or update the Amazon EFS client.
- For **Specify schedule** the recommended setting for **AmazonEFSUtils** is every 30 days. You can use controls to create a cron or rate schedule for the association.

To use AWS Systems Manager to mount EFS file systems to multiple EC2 instances, see Mounting EFS to multiple EC2 instances .

Installing and upgrading botocore

The Amazon EFS client uses botocore to interact with other AWS services. It is required if you want to monitor mount attempt success or failure for your EFS file systems in CloudWatch Logs. For more information, see Monitoring mount attempt successes and failures.

For instructions on installing and upgrading botocore, see Installing botocore in the amazon-efs-utils README on Github.

Upgrading stunnel

Encryption of data in transit with the EFS mount helper requires OpenSSL version 1.0.2 or newer, and a version of stunnel that supports both Online Certificate Status Protocol (OCSP) and certificate hostname checking. The EFS mount helper uses the stunnel program for its TLS functionality. Note that some versions of Linux don't include a version of stunnel that supports these TLS features by default. When using one of those Linux distributions, mounting an EFS file system using TLS fails.

After installing the EFS mount helper, you can upgrade your system's version of stunnel with the following instructions.

To upgrade stunnel on Amazon Linux, Amazon Linux 2, and other supported Linux distributions (except for SLES 12)

1. In a web browser, go to the stunnel downloads page https://stunnel.org/downloads.html.
2. Locate the latest stunnel version that is available in tar.gz format. Note the name of the file as you will need it in the following steps.
3. Open a terminal on your Linux client, and run the following commands in the order presented.

   a. For RPM:

   ```
   sudo yum install -y gcc openssl-devel tcp_wrappers-devel
   ```

   For DEB:

   ```
   sudo apt-get install build-essential libwrap0-dev libssl-dev
   ```

b. Replace *latest-stunnel-version* with the name of the file you noted previously in Step 2.

```
sudo curl -o latest-stunnel-version.tar.gz https://www.stunnel.org/
downloads/latest-stunnel-version.tar.gz
```

c.

```
sudo tar xvfz latest-stunnel-version.tar.gz
```

d.

```
cd latest-stunnel-version/
```

e.

```
sudo ./configure
```

f.

```
sudo make
```

g. The current `stunnel` package is installed in `bin/stunnel`. So that the new version can be installed, remove that directory with the following command.

```
sudo rm /bin/stunnel
```

h. Install the latest version:

```
sudo make install
```

i. Create a symlink:

```
sudo ln -s /usr/local/bin/stunnel /bin/stunnel
```

To upgrade stunnel on macOS

- Open a terminal on your EC2 Mac instance, and run the following command to upgrade to the latest version of stunnel.

```
brew upgrade stunnel
```

Upgrading stunnel for SLES 12

- Run the following commands and follow the zypper package manager instructions to upgrade stunnel on your compute instance running SLES12.

  ```
  sudo zypper addrepo https://download.opensuse.org/repositories/security:Stunnel/
  SLE_12_SP5/security:Stunnel.repo
  sudo zypper refresh
  sudo zypper install -y stunnel
  ```

After you've installed a version of stunnel with the required features, you can mount your file system using TLS with the Amazon EFS recommended settings.

Resolving issues with installing stunnel

If you are unable to install stunnel, try disabling certificate hostname checking. Additionally, provide the strongest security possible by enabling Online Certificate Status Protocol (OCSP).

Topics

- Disabling Certificate Hostname Checking
- Enabling Online Certificate Status Protocol

Disabling Certificate Hostname Checking

If you are unable to install the required dependencies, you can optionally disable certificate hostname checking inside the Amazon EFS mount helper configuration. We do not recommend that you disable this feature in production environments. To disable certificate host name checking, do the following:

1. Using your text editor of choice, open the `/etc/amazon/efs/efs-utils.conf` file.
2. Set the `stunnel_check_cert_hostname` value to false.
3. Save the changes to the file and close it.

For more information on using encryption of data in transit, see Mounting EFS file systems.

Enabling Online Certificate Status Protocol

In order to maximize file system availability in the event that the CA is not reachable from your VPC, the Online Certificate Status Protocol (OCSP) is not enabled by default when you choose to encrypt data in transit. Amazon EFS uses an Amazon certificate authority (CA) to issue and sign its TLS certificates, and the CA instructs the client to use OCSP to check for revoked certificates. The OCSP endpoint must be accessible over the Internet from your Virtual Private Cloud in order to check a certificate's status. Within the service, Amazon EFS continuously monitors certificate status, and issues new certificates to replace any revoked certificates it detects.

In order to provide the strongest security possible, you can enable OCSP so that your Linux clients can check for revoked certificates. OCSP protects against malicious use of revoked certificates, which is unlikely to occur within your VPC. In the event that an EFS TLS certificate is revoked, Amazon publishes a security bulletin and release a new version of EFS mount helper that rejects the revoked certificate.

To enable OCSP on your Linux client for all future TLS connections to EFS

1. Open a terminal on your Linux client.
2. Using your text editor of choice, open the `/etc/amazon/efs/efs-utils.conf` file.
3. Set the `stunnel_check_cert_validity` value to true.
4. Save the changes to the file and close it.

To enable OCSP as part of the `mount` command

- Use the following mount command to enable OCSP when mounting the file system.

```
$ sudo mount -t efs -o tls,ocsp fs-12345678:/ /mnt/efs
```

Enabling FIPS mode

If your operating system is using Federal Information Processing Standards (FIPS) endpoints when mounting your file system, then you must enable FIPS mode in the Amazon EFS client. Enabling the FIPS mode involves modifying the `efs-utils.conf` file on the operating system.

> **Note**
>
> FIPS mode requires that the installed version of OpenSSL is compiled with FIPS. For more information on how to configure OpenSSL with FIPS see the OpenSSL FIPS README.

To enable FIPS mode in the Amazon EFS client

1. Access the terminal for your Amazon EC2 instance through Secure Shell (SSH), and log in with the appropriate user name. For more information, see Connect to your EC2 instance in the *Amazon Elastic Compute Cloud User Guide*.
2. Using your text editor of choice, open the `/etc/amazon/efs/efs-utils.conf` file.
3. Find the line containing the following text:

```
"fips_mode_enabled = false"
```

4. Change the text to the following:

```
"fips_mode_enabled = true"
```

5. Save your changes.

Creating and managing EFS resources

Amazon EFS provides elastic, shared file storage that is POSIX-compliant. The file system that you create supports concurrent read and write access from multiple Amazon EC2 instances. The file system is also accessible from all of the Availability Zones in the AWS Region where it is created.

You can mount an Amazon EFS file system on EC2 instances in your virtual private cloud (VPC) based on Amazon VPC by using the Network File System versions 4.0 and 4.1 protocol (NFSv4). For more information, see How Amazon EFS works.

As an example, suppose that you have one or more EC2 instances launched in your VPC. Now you want to create and use a file system on these instances. Following are the typical steps that you must perform to use Amazon EFS file systems in the VPC:

- **Create an Amazon EFS file system –** When creating a file system, we recommend using the **Name** tag. The **Name** tag value appears in the console and makes it easier to identify the file system. You can also add other optional tags to the file system.
- **Create mount targets for the file system –** To access the file system in your VPC and mount the file system to your Amazon EC2 instance, you must create mount targets in the VPC subnets.
- **Create security groups –** Both an Amazon EC2 instance and a mount target must have associated security groups. These security groups act as a virtual firewall that controls the traffic between them. You can use the security group that you associated with the mount target to control inbound traffic to your file system. To do this, add an inbound rule to the mount target security group that allows access from a specific EC2 instance. Then, you can mount the file system only on that EC2 instance.

Topics

- Implementation summary
- Resource IDs
- Creation token and idempotency
- Creating EFS file systems
- Deleting EFS file systems
- Creating file system policies
- Creating access points

- Deleting access points
- Tagging EFS resources
- Tutorial: Creating writable per-user subdirectories

Implementation summary

In Amazon EFS, a file system is the primary resource. Each file system has properties such as ID, creation token, creation time, file system size in bytes, number of mount targets created for the file system, and the file system lifecycle policies.

Amazon EFS also supports other resources to configure the primary resource. These include mount targets and access points:

- **Mount target** – To access your file system, you must create mount targets in your VPC. Each mount target has the following properties: the mount target ID, the subnet ID in which it is created, the file system ID for which it is created, an IP address at which the file system may be mounted, VPC security groups, and the mount target state. You can use the IP address or the DNS name in your `mount` command.

  Each file system has a DNS name of the following form.

  ```
  file-system-id.efs.aws-region.amazonaws.com
  ```

  You can specify this DNS name in your `mount` command to mount the Amazon EFS file system. Suppose you create an `efs-mount-point` subdirectory off of your home directory on your EC2 instance or on-premises server. Then, you can use the mount command to mount the file system. For example, on an Amazon Linux AMI, you can use the following `mount` command.

  ```
  $ sudo mount -t nfs -o
   nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport file-
  system-DNS-name:/ ~/efs-mount-point
  ```

  For more information, see Managing mount targets.

- **Access Points** – An access point applies an operating system user, group, and file system path to any file system request made using the access point. The access point's operating system user and group override any identity information provided by the NFS client. The file system path is exposed to the client as the access point's root directory. This ensures that each application always uses the correct operating system identity and the correct directory when accessing shared file-based datasets. Applications using the access point can only access data in its own directory and below. For more information, see Working with access points.

Mount targets and tags are *subresources* that are associated with a file system. You can only create them within the context of an existing file system.

Amazon EFS provides API operations for you to create and manage these resources. In addition to the create and delete operations for each resource, Amazon EFS supports a describe operation that enables you to retrieve resource information. You have the following options for creating and managing these resources:

- Use the Amazon EFS console – For an example, see Getting started.
- Use the Amazon EFS command line interface (CLI) – For an example, see Tutorial: Create an EFS file system and mount it on an EC2 instance using the AWS CLI.
- You can also manage these resources programmatically as follows:
  - Use the AWS SDKs – The AWS SDKs simplify your programming tasks by wrapping the underlying Amazon EFS API. The SDK clients also authenticate your requests by using access keys that you provide. For more information, see Sample Code and Libraries.
  - Call the Amazon EFS API directly from your application – If you cannot use the SDKs for some reason, you can make the Amazon EFS API calls directly from your application. However, you need to write the necessary code to authenticate your requests if you use this option. For more information about the Amazon EFS API, see Amazon EFS API.

Resource IDs

Amazon EFS assigns unique resource identifiers (IDs) to all EFS resources when they are created. All EFS resource IDs consist of a resource identifier and a combination of digits 0–9 and lowercase letters a–f.

Before October 2021, the IDs assigned to newly created file system and mount target resources used 8 characters after the hyphen (for example, `fs-12345678`). From May 2021 to October 2021, we changed the IDs of these resource types to use 17 characters after the hyphen (for example, `fs-1234567890abcdef0`). Depending on when your account was created, you might have file system and mount target resources with short IDs, though any new resources of these types receive the longer IDs. The Resource ID never changes.

Creation token and idempotency

Idempotency ensures that an API request completes only once. With idempotent requests, if the original request completes successfully, subsequent requests have no additional effect. This is useful to prevent duplicate jobs from being created when you interact with the Amazon EFS API.

The Amazon EFS API supports idempotency with client request tokens. A *client request token* is a unique string that you specify when you make a create job request.

A client request token can be any string that includes up to 64 ASCII characters. If you reuse a client request token within one minute of a successful request, the API returns the job details of the original request.

If you use the console, it generates the token for you. If you use the **Custom Create** flow in the console, the creation token that is generated for you has the following format:

```
"CreationToken": "console-d215fa78-1f83-4651-b026-facafd8a7da7"
```

If you use Quick Create to create a file system with the service recommended settings, the creation token has the following format:

```
"CreationToken": "quickCreated-d7f56c5f-e433-41ca-8307-9d9c0f8a77a2"
```

Creating EFS file systems

You can create a file system when you create a new EC2 launch instance, as explained in the Getting started exercise. However, you can also create file systems by using the Amazon EFS console, the AWS Command Line Interface (AWS CLI), or the Amazon EFS API.

When creating the file system using the Amazon EFS console, you have different options, depending on whether you want the file system to use the recommended settings or if you want to customize the settings.

- You can use Quick create to quickly create a file system with the following recommended settings.
  - Regional availability
  - Lifecycle policies to transition the file system to EFS Infrequent Access (IA) storage after 30 days, to EFS Archive storage after 90 days, and not to transition to EFS Standard storage

    - Encryption of data at rest enabled
    - Elastic throughput mode
    - General Purpose performance mode
    - Mount targets configured in each Availability Zone in the AWS Region in which the file system is created using an available IPv4 on the specified subnet
- You can use Customize to create a file system with the settings you choose.

For a table that lists the file system settings and the recommended values, see Configuration options for file systems.

Required IAM permissions for creating file systems

To create EFS resources, such as a file system and access points, you must have AWS Identity and Access Management (IAM) permissions for the corresponding API operation and resource.

Create IAM users and grant them permissions for Amazon EFS actions with user policies. You can also use roles to grant cross-account permissions. Amazon Elastic File System also uses an IAM service-linked role that includes permissions required to call other AWS services on your behalf. For more information about managing permissions for API operations, see Identity and access management for Amazon EFS.

Configuration options for file systems

EFS file systems are configured with the settings listed in the following table.

- If you use Quick create to create the file system, the file system is created with the indicated recommended value for the setting.
- If use Customize to create a custom file system, you can change the recommended value for the setting.

After you create the file system, you can customize the file system's settings with the exception of file system type, encryption, and performance mode.

EFS file system settings

Setting	Description
File system type	*Recommended: Regional*

Setting	Description
	The file system type determines the availability and durability with which an EFS file system stores data within an AWS Region. • **Regional** file systems store data and metadata redundantly across all Availability Zones within an AWS Region. You can also create mount targets in each Availability Zone in the AWS Region. Regional offers the highest levels of availability and durability. • **One Zone** file systems store data and metadata redundantly within a single Availability Zone. One Zone file systems can have only a single mount target. This mount target must be located in the same Availability Zone in which the file system is created. **Note** One Zone file systems are available to only certain Availability Zones. For more information about file system types, see Availability and durability of EFS file systems.

Setting	Description
Lifecycle management	*Recommended: Transition into IA storage after 30 days and into Archive storage after 90 days. Do not transition into Standard storage.* Lifecycle management uses lifecycle policies to automatically move files in to and out of the lower-cost Infrequent Access (IA) storage class based on access patterns. When you create a file system by using the AWS Management Console, the file system's lifecycle policy is configured with the following default settings: • **Transition into IA** set to **30 days since last access**. • **Transition into Archive** set to **90 days since last access.** • **Transition into Standard** set to **None**. When you create a file system by using the AWS CLI, Amazon EFS API, or AWS SDKs, you cannot set a lifecycle policy at the same time. You must wait until the file system is created, and then use the PutLifecycleConfiguration API operation to update the lifecycle policy. For more information about lifecycle management, see Managing storage lifecycle.

Setting	Description
Encryption at rest	*Recommended: Enabled* Amazon EFS uses your AWS Key Managemen t Service (AWS KMS) EFS service key (`aws/elasticfilesystem`) to encrypt data at rest by default. With encryption, all data and metadata stored on it are encrypted. Once you create an EFS file system, you cannot change its encryption setting. This means that you cannot modify an unencrypted file system to make it encrypted. Instead, you need to create a new, encrypted file system. For more information about Amazon EFS encryption, see Encrypting data in Amazon EFS.

Setting	Description
Throughput mode	*Recommended: Elastic* You can choose from the following throughpu t modes: • **Elastic** – Provides throughput that scales up and down automatically in real time, to meet your workload's performance needs. • **Provisioned** – Specify the throughput amount that you want, independent of the file system's size. • **Bursting –** Provides throughput that scales with the amount of data in Standard storage. **Note** Additional charges are associated with using Elastic and Provisioned throughput. For more information, see Amazon EFS pricing. For more information about throughput modes, see Throughput modes.

Setting	Description
Performance mode	*Recommended: General Purpose* The **General Purpose** performance mode has the lowest per-operation latency and is recommended for all file systems. **Max I/O** is a previous generation performance type that is designed for highly parallelized workloads that can tolerate higher latencies than the General Purpose mode. Max I/O mode is not supported for One Zone file systems or file systems that use Elastic throughput. **Important** Due to the higher per-operation latencies with Max I/O, we recommend using General Purpose performance mode for all file systems. For more information, see Performance modes.

Setting	Description
Network access	*Recommended: Mount target created in each Availability Zone in which the file system is available* The mount target settings are as follows: • **VPC** – You can create mount targets for the file system in one VPC at a time. By default, the EC2 instance's default subnet is selected. Change the VPC if you need to access the file system from a different VPC. If you want to change the VPC after creating the after the file system, you need to first delete the mount targets from the current VPC. For more information, see Changing the mount target VPC. • **Availability Zone** and **Subnet ID** – You can change the Availability Zone or subnet for the mount target. • **IP address type** – By default, Amazon EFS uses an IPv4 address at which to create the mount target. You can alternatively choose **IPv6 only** to support IPv6 addresses only, or **Dual-stack** to support both IPv4 and IPv6 addresses. **Note** The IP address type must match the IP type of the subnet. Additionally, the IP address type overrides the IP addressing attribute of your subnet. For example, if the IP address type is IPv4-only and the IPv6 addressing

Setting	Description
	attribute is enabled for your subnet, network interfaces created in the subnet receive an IPv4 address from the range of the subnet. For more information, see Modify the IP addressing attributes of your subnet. • **IP address** – By default, Amazon EFS creates the mount target at an available address on the specified subnet. Alternately, you can specify the IP address for the mount target. You can't change the IP address of a mount target after it's created. You need to delete the mount target and create a new one with the new address. • **Security group** – The security group assigned to the VPC target is used by default. You can add or remove security groups. For more information about security groups, see Changing mount target security groups. When you create a file system by using the AWS CLI, Amazon EFS API, or AWS SDKs, you cannot create a mount target at the same time. You must wait until the file system is created, and then use the CreateMountTarget API operation to create the mount targets. For more information about mount targets, see Managing mount targets.

Quick create using the console

Use the Amazon EFS console to create an Amazon EFS file system that has the recommended settings. If you want to create a file system with a customized configuration, see Custom create using the console.

To quick create an Amazon EFS file system that has the recommended settings

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. Choose **Create file system** to open the **Create file system** dialog box.
3. (Optional) Enter a **Name** for your file system.
4. For **Virtual Private Cloud (VPC)**, choose your VPC, or keep it set to your default VPC.
5. Choose **Create** to create a file system that uses the following service recommended settings:

   - Regional availability
   - Lifecycle policies to transition the file system to EFS Infrequent Access (IA) storage after 30 days, to EFS Archive storage after 90 days, and not to transition to EFS Standard storage
   - Encryption of data at rest enabled
   - Elastic throughput mode
   - General Purpose performance mode
   - Mount targets configured in each Availability Zone in the AWS Region in which the file system is created, using an available IPv4 on the specified subnet

   The **File systems** page appears with a banner across the top showing the status of the file system you created. A link to access the file system details page appears in the banner when the file system becomes available.

   For more information about file system status, see Understanding file system status.

Custom create using the console

This section describes the process of using the Amazon EFS console to create an EFS file system with customized settings instead of using the service-recommended settings. For more information about creating a file system by using the recommended settings, see Quick create using the console.

Creating an EFS file system with custom settings by using the console is a four-step process:

- Step 1 – Configure general file system settings, including the storage class and throughput mode.
- Step 2 – Configure file system network settings, including the virtual private cloud (VPC) and mount targets. For each mount target, set the Availability Zone, subnet, IP address, and security groups.
- Step 3 – (Optional) Create a file system policy to control NFS client access to the file system.
- Step 4 – Review the file system settings, make any changes, and then create the file system.

Step 1: Configure file system settings

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. Choose **Create file system** to open the **Create file system** dialog box.
3. Choose **Customize** to create a customized file system instead of creating a file system by using the recommended settings. The **File system settings** page opens.
4. For **General** settings, do the following.

   a. (Optional) Enter a **Name** for the file system.

   b. For **File system type**, **Regional** is selected by default. Choose **One Zone** if you want to create a file system that stores file system data and metadata redundantly within a single Availability Zone. If you choose **One Zone**, choose the **Availability Zone** that you want the file system created in, or keep the default value.

   c. For **Lifecycle management**, change the lifecycle policies, if necessary.

      - **Transition into IA** – Select when to transition files into the Infrequent Access (IA) storage class, based on the time since they were last accessed in Standard storage.
      - **Transition into Archive** – Select when to transition files into the Archive storage class, based on the time since they were last accessed in Standard storage.
      - **Transition into Standard** – Select whether to transition the file system to the storage class.

      For more information about lifecycle policies, see Managing storage lifecycle.

   d. Amazon EFS uses your AWS Key Management Service (AWS KMS) EFS service key (`aws/elasticfilesystem`) to encrypt data at rest by default. To choose a different KMS key to use for encryption, expand **Customize encryption settings** and choose a key from the list. Or, enter a KMS key ID or Amazon Resource Name (ARN) for the KMS key that you want to use.

If you need to create a new key, choose **Create an AWS KMS key** to launch the AWS KMS console and create a new key.

You can turn off encryption of data at rest by clearing the check box.

You cannot change the encryption setting after the file system is created. For more information, see Encrypting data in Amazon EFS.

5. For **Performance** settings, do the following:

   a. For **Throughput mode**, the **Elastic** mode is selected by default.

      - To use provisioned throughput, choose **Provisioned**, and, in **Provisioned Throughput (MiB/s)**, enter the amount of throughput to provision for file system requests. The amount of **Maximum Read Throughput** is displayed at three times the amount of the throughput that you enter.
      - To use bursting throughput, choose **Bursting**.

      After you choose the throughput mode, an estimate of the monthly cost for the file system is shown. You can change the throughput mode after the file system becomes available.

      For more information about choosing the correct throughput mode for your performance needs, see Encrypting data in Amazon EFS.

   b. For **Performance mode**, the default is **General Purpose**. To change the performance mode, expand **Additional settings**, and then choose **Max I/O**.

      You cannot change the performance mode after the file system becomes available. For more information, see Performance modes.

      > **Important**
      > Due to the higher per-operation latencies with Max I/O, we recommend using General Purpose performance mode for all file systems.

6. (Optional) Add tag key-value pairs to your file system.
7. Choose **Next** to configure network access for the file system.

Step 2: Configure network access

In Step 2, you configure the file system's network settings, including the VPC and mount targets.

1. Choose the **Virtual Private Cloud (VPC)** where you want EC2 instances to connect to your file system. For more information, see Managing mount targets.
2. For **Mount targets**, you create one or more mount targets for your file system. For each mount target, set the following properties:

   - **Availability Zone** – By default, a mount target is configured in each Availability Zone in an AWS Region. If you don't want a mount target in a particular Availability Zone, choose **Remove** to delete the mount target for that zone. Create a mount target in every Availability Zone that you plan to access your file system from – there is no cost to do so.
   - **Subnet ID** – Choose from the available subnets in an Availability Zone. The default subnet is preselected.
   - **IP address type** – Choose **IPv4 only** to support IPv4 addresses only, **IPv6 only** to support IPv6 addresses only, or **Dual-stack** to support both IPv4 and IPv6 addresses.
   - **IPv4 or IPv6 address** – If you know the IP address where you want to place the mount target, then enter it in the IP address box that matches the **IP address type**. If you don't specify a value, Amazon EFS selects an unused IP address from the specified subnet.
   - **Security groups** – By default, Amazon EFS chooses the default security group for the VPC. To change the security group, delete the assigned group and then choose the group from the **Choose security groups** list. You can specify one or more security groups for the mount target. For more information, see Using VPC security groups.
3. Choose **Add mount target** to create a mount target for an Availability Zone that doesn't have one. If a mount target is configured for each Availability Zone, this choice is not available.
4. Choose **Next** to set the file system policy.

Step 3: Create a file system policy (optional)

Optionally, you can create a file system policy for your file system. An EFS file system policy is an IAM resource policy that you use to control NFS client access to the file system. For more information, see Using IAM to control access to file systems.

1. In **Policy options**, you can choose any combination of the preconfigured file system policies:

   - **Prevent root access by default –** This option removes `ClientRootAccess` from the set of allowed EFS actions.
   - **Enforce read-only access by default –** This option removes `ClientWriteAccess` from the set of allowed EFS actions.
   - **Prevent anonymous access –** This option removes `ClientMount` from the set of allowed EFS actions.
   - **Enforce in-transit encryption for all clients –** This option denies access to unencrypted clients.

   When you choose a preconfigured policy, the policy JSON object is displayed in the **Policy editor** pane.
2. Use **Grant additional permissions** to grant file system permissions to additional IAM principals, including another AWS account. Choose **Add**, and enter the principal ARN of the entity that you are granting permissions to. Then choose the **Permissions** that you want to grant. The additional permissions are shown in the **Policy editor**.
3. You can use the **Policy editor** to customize a preconfigured policy or to create your own file system policy. When you use the editor, the preconfigured policy options become unavailable. To clear the current file system policy and start creating a new policy, choose **Clear**.
4. Choose **Next** to review and create the file system.

Step 4: Review and create

1. Review each of the file system configuration groups. You can make changes to each group at this time by choosing **Edit**.
2. Choose **Create file system** to create your file system and return to the **File systems** page.

   A banner across the top shows that the new file system is being created. A link to access the new file system details page appears in the banner when the file system becomes available.

Create using the AWS CLI

When you're using the AWS CLI, you create these resources in order. First, you create a file system. Then, you can create mount targets and any additional optional tags for the file system by using corresponding AWS CLI commands.

The following examples use `adminuser` for the `--profile` parameter values. You must use an appropriate user profile to provide your credentials. For information, see Prerequisites to use the AWS CLI in the *AWS Command Line Interface User Guide*.

- To create an encrypted file system with automatic backups enabled, use the Amazon EFS `create-file-system` CLI command (the corresponding operation is CreateFileSystem), as shown following.

```
aws efs create-file-system \
--creation-token creation-token \
--encrypted \
--backup \
--performance-mode generalPurpose \
--throughput-mode elastic \
--region aws-region \
--tags Key=key,Value=value Key=key1,Value=value1 \
--profile adminuser
```

  For example, the following `create-file-system` command creates a file system using Elastic throughput in the `us-west-2` AWS Region. The command specifies `MyFirstFS` as the creation token. For a list of the AWS Regions where you can create an Amazon EFS file system, see Amazon EFS endpoints and quotas in the *Amazon Web Services General Reference*.

```
aws efs create-file-system \
--creation-token MyFirstFS \
--backup \
--encrypted \
--performance-mode generalPurpose \
--throughput-mode elastic \
--region us-west-2 \
--tags Key=Name,Value="Test File System" Key=developer,Value=rhoward \
--profile adminuser
```

After successfully creating the file system, Amazon EFS returns the file system description as JSON, as shown in the following example.

```
{
    "OwnerId": "123456789abcd",
    "CreationToken": "MyFirstFS",
    "Encrypted": true,
    "FileSystemId": "fs-c7a0456e",
    "CreationTime": 1422823614.0,
    "LifeCycleState": "creating",
    "Name": "Test File System",
    "NumberOfMountTargets": 0,
    "SizeInBytes": {
        "Value": 6144,
        "ValueInIA": 0,
        "ValueInStandard": 6144
        "ValueInArchive": 0
    },
    "PerformanceMode": "generalPurpose",
    "ThroughputMode": "elastic",
    "Tags": [
      {
         "Key": "Name",
         "Value": "Test File System"
      }
   ]
}
```

- The following example creates a file system that uses Bursting throughput in the `us-west-2a` Availability Zone by using the `availability-zone-name` property.

```
aws efs create-file-system \
--creation-token MyFirstFS \
--availability-zone-name us-west-2a \
--backup \
--encrypted \
--performance-mode generalPurpose \
--throughput-mode bursting \
--region us-west-2 \
--tags Key=Name,Value="Test File System" Key=developer,Value=rhoward \
--profile adminuser
```

After successfully creating the file system, Amazon EFS returns the file system description as JSON, as shown in the following example.

```
{
    "AvailabilityZoneId": "usw-az1",
    "AvailabilityZoneName": "us-west-2a",
    "OwnerId": "123456789abcd",
    "CreationToken": "MyFirstFS",
    "Encrypted": true,
    "FileSystemId": "fs-c7a0456e",
    "CreationTime": 1422823614.0,
    "LifeCycleState": "creating",
    "Name": "Test File System",
    "NumberOfMountTargets": 0,
    "SizeInBytes": {
        "Value": 6144,
        "ValueInIA": 0,
        "ValueInStandard": 6144
        "ValueInArchive": 0
    },
    "PerformanceMode": "generalPurpose",
    "ThroughputMode": "bursting",
    "Tags": [
      {
         "Key": "Name",
         "Value": "Test File System"
      }
   ]
}
```

Amazon EFS also provides the `describe-file-systems` CLI command (the corresponding API operation is [DescribeFileSystems]), which you can use to retrieve a list of file systems in your account, as shown following.

```
aws efs describe-file-systems \
--region aws-region \
--profile adminuser
```

Amazon EFS returns a list of the file systems in your AWS account created in the specified Region.

Deleting EFS file systems

File system deletion is a destructive action that you can't undo. You lose the file system and any data you have in it. Any data that you delete from a file system is gone, and you can't restore the data. When users delete data from a file system, that data is immediately rendered unusable. EFS force-overwrites the data in an eventual manner.

> **Note**
>
> You cannot delete a file system that is part of a replication configuration. You must delete the replication configuration first. For more information, see Deleting replication configurations.

> **Important**
>
> You should always unmount a file system before you delete it.

Using the console

To delete a file system

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. Select the file system that you want to delete in the **File systems** page.
3. Choose **Delete**.
4. In the **Delete file system** dialog box, enter the file system ID shown, and choose **Confirm** to confirm the delete.

   The console simplifies the file system deletion for you. First it deletes the associated mount targets, and then it deletes the file system.

Using the AWS CLI

Before you can use the AWS CLI command to delete a file system, you must delete all of the mount targets and access points that were created for the file system.

For example AWS CLI commands, see Step 4: Clean up.

Creating file system policies

You can create a file system policy by using the Amazon EFS console or by using the AWS CLI. You can also create a file system policy programmatically by using AWS SDKs or the Amazon EFS API directly. EFS file system policies have a 20,000 character limit. For more information about using an EFS file system policy and examples, see Using IAM to control access to file systems.

> **Note**
>
> Amazon EFS file system policy changes can take several minutes to take effect.

Using the console

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. Choose **File Systems**.
3. On the **File systems** page, choose the file system that you want to edit or create a file system policy for.
4. Choose **File system policy**, then choose **Edit**.
5. In **Policy options**, you can choose any combination of the preconfigured file system policies:

    - **Prevent root access by default** – This option removes `ClientRootAccess` from the set of allowed EFS actions.
    - **Enforce read-only access by default** – This option removes `ClientWriteAccess` from the set of allowed EFS actions.
    - **Prevent anonymous access** – This option removes `ClientMount` from the set of allowed EFS actions.
    - **Enforce in-transit encryption for all clients** – This option denies access to unencrypted clients.

    When you choose a preconfigured policy, the policy JSON object is displayed in the **Policy editor** pane.
6. Use **Grant additional permissions** to grant file system permissions to additional IAM principals, including another AWS account. Choose **Add**, and enter the principal ARN of the entity that you are granting permissions to. Then choose the **Permissions** that you want to grant. The additional permissions are shown in the **Policy editor**.

7. You can use the **Policy editor** to customize a preconfigured policy or to create your own file system policy. When you use the editor, the preconfigured policy options become unavailable. To clear the current file system policy and start creating a new policy, choose **Clear**.

   When you clear the editor, the preconfigured policies become available once again.

8. After you complete editing the policy, choose **Save**.

Using the AWS CLI

In the following example, the put-file-system-policy CLI command creates a file system policy that allows the specified AWS account read-only access to the EFS file system. The equivalent API command is PutFileSystemPolicy.

```
aws efs put-file-system-policy --file-system-id fs-01234567 --policy '{
    "Id": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientMount"
            ],
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            }
        }

    ]
}'
```

Creating access points

You can create and delete Amazon EFS access points using the AWS Management Console, the AWS Command Line Interface (AWS CLI), and the Amazon EFS API and SDKs. You cannot modify an access point once it is created. A file system can have a maximum of 10,000 access points unless you request an increase.

> **Note**
>
> If multiple requests to create access points on the same file system are sent in quick succession, and the file system is nearing the access points limit, you may experience a throttling response for these requests. This is to ensure that the file system does not exceed the stated access point quota.

For more information about EFS access points, see Working with access points.

Using the console

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. Choose **Access points** to open the **Access points** window.
3. Choose **Create access point** to display the **Create access point** page.

   You can also open the **Create access point** page by choosing **File Systems**. Choose a file system **Name** or **File system ID** and then choose **Access points** and **Create access point** to create an access point for that file system.

   a. Enter the following information in the **Details** panel:

      - **File system –** Enter a file system name or ID and choose the matching file system. You can also choose the file system from the list that appears when you choose the input field.
      - (Optional) **Name –** Enter a name for the access point.
      - (Optional) **Root directory path –** You can specify a root directory for the access point; the default access point root is `/`. To enter a root directory path, use the format `/foo/bar`. For more information, see Enforcing a root directory with an access point.

   b. (Optional) In the **POSIX user** panel, you can specify the full POSIX identity to use to enforce user and group information for all file operations by NFS clients that are using the access point. For more information, see Enforcing a user identity using an access point.

      - **User ID –** Enter a numeric POSIX user ID for the user.
      - **Group ID –** Enter a numeric POSIX group ID for the user.
      - **Secondary group IDs –** Enter an optional comma-separated list of secondary group IDs.

c. (Optional) For **Root directory creation permissions**, you can specify the permissions to use when Amazon EFS creates the root directory path, if specified and the root directory doesn't already exist. For more information, see Enforcing a root directory with an access point.

> **Note**
>
> If you don't specify any root directory ownership and permissions, and the root directory does not already exist, EFS will not create the root directory. Any attempts to mount the file system by using the access point will fail.

- **Owner user ID** – Enter the numeric POSIX user ID to use as the root directory owner.
- **Owner group ID** – Enter the numeric POSIX group ID to use as the root directory owner group.
- **Permissions** – Enter the Unix mode of the directory. A common configuration is 755. Ensure that the execute bit is set for the access point user so that they are able to mount.

4. Choose **Create access point** to create the access point by using this configuration.

Using the AWS CLI

In the following example, the `create-access-point` CLI command creates an access point for an EFS file system. The equivalent API command is CreateAccessPoint.

```
aws efs create-access-point --file-system-id fs-abcdef0123456789a --client-token
 010102020-3 \
--root-directory “Path=/efs/mobileapp/
east,CreationInfo={OwnerUid=0,OwnerGid=11,Permissions=775}” \
--posix-user “Uid=22,Gid=4” \
--tags Key=Name,Value=east-users
```

If the request is successful, the CLI responds with the access point description.

```
{
    "ClientToken": "010102020-3",
    "Name": "east-users",
    "AccessPointId": "fsap-abcd1234ef5678901",
```

```
    "AccessPointArn": "arn:aws:elasticfilesystem:us-east-2:111122223333:access-point/
fsap-abcd1234ef5678901",
    "FileSystemId": "fs-01234567",
    "LifeCycleState": "creating",
    "OwnerId": "111122223333",
    "PosixUser": {
      "Gid": 4,
      "Uid": 22
    },
    "RootDirectory": {
    "CreationInfo": {
         "OwnerGid": 0,
         "OwnerUid": 11,
         "Permissions": "775"
      },
        "Path": "/efs/mobileapp/east",
    },
    "Tags": []
}
```

Note

If multiple requests to create access points on the same file system are sent in quick succession, and the file system is nearing the access points limit, you may experience a throttling response for these requests. This is to ensure that the file system does not exceed the stated access point quota.

Deleting access points

When you delete an access point, any clients that are using the access point lose access to the Amazon EFS file system that it's configured for.

Using the console

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the left navigation pane, choose **Access points** to open the **Access points** page.
3. Select the access point to delete.
4. Choose **Delete**.

5. Choose **Confirm** to confirm the action and delete the access point.

Using the AWS CLI

In the following example, the `delete-access-point` CLI command deletes the specified access point. The equivalent API command is [DeleteAccessPoint](). If the command is successful, the service returns an HTTP 204 response with an empty HTTP body.

```
aws efs delete-access-point --access-point-id fsap-092e9f80b3fb5e6f3 --client-token
 010102020-3
```

Tagging EFS resources

To help you manage your EFS resources, you can assign your own metadata to each resource in the form of *tags*. With tags, you can categorize your AWS resources in different ways, for example, by purpose, owner, or environment. This categorization is useful when you have many resources of the same type—you can quickly identify a specific resource based on the tags that you've assigned to it. This topic describes tags and shows you how to create them.

Tag basics

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value, both of which you define.

Tags enable you to categorize your AWS resources in different ways, for example, by purpose, owner, or environment. For example, you could define a set of tags for your account's Amazon EFS file systems that helps you track each file system's owner.

We recommend that you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys makes it easier for you to manage your resources. You can search and filter the resources based on the tags you add.

Tags don't have any semantic meaning to Amazon EFS and are interpreted strictly as a string of characters. Also, tags are not automatically assigned to your resources. You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags for the resource are also deleted.

Tag restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource – 50
- For each resource, each tag key must be unique, and each tag key can have only one value.
- Maximum key length – 128 Unicode characters in UTF-8
- Maximum value length – 256 Unicode characters in UTF-8
- Although Amazon EFS allows for any character in its tags, other services are more restrictive. The allowed characters across services are: letters, numbers, and spaces representable in UTF-8, and the following characters: + - = . _ : / @.
- Tag keys and values are case-sensitive.
- The `aws:` prefix is reserved for AWS use. If a tag has a tag key with this prefix, then you can't edit or delete the tag's key or value. Tags with the `aws:` prefix do not count against your tags per resource limit.

You can't update or delete a resource based solely on its tags; you must specify the resource identifier. For example, to delete file systems that you tagged with a tag key called `DeleteMe`, you must use the `DeleteFileSystem` action with the resource identifiers of the file system, such as `fs-1234567890abcdef0`.

When you tag public or shared resources, the tags that you assign are available only to your AWS account. No other AWS account will have access to those tags. For tag-based access control to shared resources, each AWS account must assign its own set of tags to control access to the resource.

You can tag Amazon EFS file system and access point resources.

Using tags for access control

You can use tags to control access to Amazon EFS resources and to implement attribute-based access control (ABAC).

> **Note**
>
> Replication does not support using tags for attribute-based access control (ABAC).

Tag your resources

You can tag Amazon EFS file system and access point resources that already exist in your account.

Using the console

- You can use the Amazon EFS console to apply tags to existing resources by using the **Tags** tab on the resource details screen. In the Amazon EFS console, you can specify tags for a resource when you create the resource. For example, you can add a tag with a key of `Name` and a value that you specify.

  In most cases, the console applies the tags immediately after the resource is created (rather than during resource creation). Although the console organizes resources according to the `Name` tag, this tag doesn't have any semantic meaning to the Amazon EFS service.

Using the AWS CLI

- If you're using the Amazon EFS API, the AWS CLI, or an AWS SDK, you can use the `TagResource` EFS API action to apply tags to existing resources. Additionally, some resource-creating actions enable you to specify tags for a resource when the resource is created.

  The AWS CLI commands for managing tags, and the equivalent Amazon EFS API actions, are listed in the following table.

CLI command	Description	Equivalent API operation
tag-resource	Add new tags or update existing tags	TagResource
list-tags-for-resource	Retrieve existing tags	ListTagsForResource
untag-resource	Delete existing tags	UntagResource

Tutorial: Creating writable per-user subdirectories

After you create an EFS file system and mount it locally on your Amazon EC2 (EC2) instance, it exposes an empty directory called the *`file system root`*. One common use case for this file system root directory is to create a "writable" subdirectory for each user you create on the EC2 instance and mount the subdirectory on the user's home directory. All files and subdirectories the user creates in their home directory are then created on the EFS file system.

> **Note**
>
> You can follow the Getting started exercise to create and mount an EFS file system on your EC2 instance.

In the following steps, you create a user, create a subdirectory for the user, make the user the owner of the subdirectory, and then mount the Amazon EFS subdirectory on the user's home directory.

1. Create user mike:

   - Log in to your EC2 instance. Using root privileges (in this case, using the `sudo` command), create the user and assign a password.

     For example, the following command creates the user `mike`.

     ```
     $ sudo useradd -c "Mike Smith" mike
     $ sudo passwd mike
     ```

     A home directory is also created for the user. For example, `/home/mike`.

2. Create a subdirectory under *`EFSroot`* for the user.

   For example, the following command creates subdirectory `mike` under *`EFSroot`*.

   ```
   $  sudo mkdir /EFSroot/mike
   ```

   You will need to replace *`EFSroot`* with your local directory name.

3. The root user and root group are the owners of the subdirectory (you can verify this by using the `ls -l` command). To enable full permissions for the user on this subdirectory, grant ownership of the directory to the user.

   For example:

   ```
   $ sudo chown mike:mike /EFSroot/mike
   ```

4. Use the `mount` command to mount the subdirectory onto the user's home directory.

   For example:

```
$  sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-
target-DNS:/mike  /home/mike
```

   The *`mount-target-DNS`* address identifies the remote EFS file system root.

If you unmount this mount target, the user can't access the directory without remounting, which requires root permissions.

Mounting EFS file systems

To mount EFS file systems, we recommend that you use the EFS mount helper. The EFS mount helper helps you mount your EFS file systems on EC2 Linux and Mac instances running the supported distributions. The mount helper is part of the open-source collection of tools that is installed when you install the Amazon EFS client (`amazon-efs-utils`). For more information about the Amazon EFS client and the supported distributions, see Installing the Amazon EFS client.

Alternatively, you can manually mount EFS file systems using the standard Linux NFS client. Amazon EFS supports the Network File System versions 4.0 and 4.1 (NFSv4) protocols when mounting your file systems on Amazon EC2 instances.

Additionally, you can use the EFS mount helper or NFS to configure an EC2 instance to automatically mount an EFS file system when the instance starts.

Topics

- Mounting considerations for Linux
- Mounting EFS file systems using the EFS mount helper
- Using Network File System to mount EFS file systems
- Automatically mounting EFS file systems
- Unmounting file systems
- Tutorial: Create an EFS file system and mount it on an EC2 instance using the AWS CLI
- Tutorial: Mounting with on-premises Linux clients
- Tutorial: Mount a file system from a different VPC
- Troubleshooting mount issues

Mounting considerations for Linux

We recommend the following values for mount options on Linux:

- `rsize=1048576` – Sets the maximum number of bytes of data that the NFS client can receive for each network READ request. This value applies when reading data from a file on an EFS file system. We recommend that you use the largest size possible (up to `1048576`) to avoid diminished performance.

- `wsize=1048576` – Sets the maximum number of bytes of data that the NFS client can send for each network WRITE request. This value applies when writing data to a file on an EFS file system. We recommend that you use the largest size possible (up to `1048576`) to avoid diminished performance.
- `hard` – Sets the recovery behavior of the NFS client after an NFS request times out, so that NFS requests are retried indefinitely until the server replies. We recommend that you use the hard mount option (`hard`) to ensure data integrity. If you use a `soft` mount, set the `timeo` parameter to at least `150` deciseconds (15 seconds). Doing so helps minimize the risk of data corruption that is inherent with soft mounts.
- `timeo=600` – Sets the timeout value that the NFS client uses to wait for a response before it retries an NFS request to 600 deciseconds (60 seconds). If you must change the timeout parameter (`timeo`), we recommend that you use a value of at least `150`, which is equivalent to 15 seconds. Doing so helps avoid diminished performance.
- `retrans=2` – Sets to 2 the number of times the NFS client retries a request before it attempts further recovery action.
- `noresvport` – Tells the NFS client to use a new non-privileged Transmission Control Protocol (TCP) source port when a network connection is reestablished. Doing this helps make sure that the EFS file system has uninterrupted availability after a network recovery event.
- `_netdev` – When present in `/etc/fstab`, prevents the client from attempting to mount the EFS file system until the network has been enabled.

In general, avoid setting any other mount options that are different from the defaults, which can cause reduced performance and other issues. If you don't use the preceding defaults, be aware of the following:

- Changing read or write buffer sizes or disabling attribute caching can result in reduced performance.
- Amazon EFS ignores source ports. If you change Amazon EFS source ports, it doesn't have any effect.
- Amazon EFS doesn't support any of the Kerberos security variants. For example, the following mount command fails.

```
$ mount -t nfs4 -o krb5p <DNS_NAME>:/ /efs/
```

- We recommend that you mount your file system using its DNS name. Amazon EFS resolves this name to the IP address of the EFS mount target in the same Availability Zone as your Amazon EC2 instance without calling external resources. If you use a mount target in an Availability Zone different from that of your EC2 instance, you incur standard EC2 charges for data sent across Availability Zones. You also might see increased latencies for file system operations.
- For more mount options, and detailed explanations of the defaults, see the Linux documentation.

> **Note**
>
> If your EC2 instance needs to start regardless of the status of your mounted EFS file system, add the `nofail` option to your file system's entry in your `/etc/fstab` file.

Mounting EFS file systems using the EFS mount helper

After you install the Amazon EFS client (`amazon-efs-utils`), you can use the EFS mount helper to mount EFS file systems on your EC2 Linux and Mac instances running a supported distributions. Amazon EFS does not support mounting from EC2 Windows instances.

> **Important**
>
> We recommend that you always use the most current version of `amazon-efs-utils` to ensure successful mounting. For example, versions of `amazon-efs-utils` prior to 2.3 do not support mounting with IPv6 addresses.

When mounting a file system, the mount helper defines a new network file system type, called `efs`, which is fully compatible with the standard `mount` command in Linux. The mount helper also supports mounting an EFS file system at instance boot time automatically by using entries in the `/etc/fstab` configuration file on EC2 Linux instances.

> **Warning**
>
> Use the `_netdev` option, used to identify network file systems, when mounting your file system automatically. If `_netdev` is missing, your EC2 instance might stop responding. This result is because network file systems need to be initialized after the compute instance

starts its networking. For more information, see Automatic mounting fails and the instance is unresponsive.

You can mount a file system by specifying one of the following properties:

- **File system DNS name –** If you use the file system DNS name, and the mount helper cannot resolve it, for example when you are mounting a file system in a different VPC, it will fall back to using the mount target IP address. For more information, see Mounting EFS file systems from another AWS account or VPC.
- **File system ID –** If you use the file system ID, the mount helper resolves it to the local IP address of the mount target elastic network interface (ENI) without calling external resources.
- **Mount target IP address –** You can use the IP address of one of the file systems mount targets.

You can find the value for all of these properties in the Amazon EFS console. The file system DNS name is found in the **Attach** screen.

When encryption of data in transit is declared as a mount option for your EFS file system, the mount helper initializes a client `stunnel` process, and a supervisor process called `amazon-efs-mount-watchdog`. The `amazon-efs-mount-watchdog` process monitors the health of TLS mounts, and is started automatically the first time an EFS file system is mounted over TLS. If your client is running on Linux, this process is managed by either `upstart` or `systemd` depending on your Linux distribution. For clients running on a supported macOS, it is managed by `launchd`.

`Stunnel` is an open-source multipurpose network relay. The client `stunnel` process listens on a local port for inbound traffic, and the mount helper redirects NFS client traffic to this local port.

The mount helper uses TLS version 1.2 to communicate with your file system. Using TLS requires certificates, and these certificates are signed by a trusted Amazon Certificate Authority. For more information on how encryption works, see Encrypting data in Amazon EFS.

Topics

- Mount settings used by EFS mount helper
- Getting support logs
- Prerequisites for using the EFS mount helper
- Mounting on EC2 Linux instances using the EFS mount helper
- Mounting on EC2 Mac instances using the EFS mount helper

- Mounting EFS file systems from a different AWS Region
- Mounting One Zone file systems
- Mounting with IAM authorization
- Mounting with EFS access points
- Mounting EFS to multiple EC2 instances
- Mounting EFS file systems from another AWS account or VPC

Mount settings used by EFS mount helper

The Amazon EFS mount helper client uses the following mount options that are optimized for Amazon EFS:

- `nfsvers=4.1` – used when mounting on EC2 Linux instances

  `nfsvers=4.0` – used when mounting on supported EC2 Mac instances running macOS Big Sur, Monterey, and Ventura
- `rsize=1048576` – Sets the maximum number of bytes of data that the NFS client can receive for each network READ request to 1048576, the largest available, to avoid diminished performance.
- `wsize=1048576` – Sets the maximum number of bytes of data that the NFS client can send for each network WRITE request to `1048576`, the largest available, to avoid diminished performance.
- `hard` – Sets the recovery behavior of the NFS client after an NFS request times out, so that NFS requests are retried indefinitely until the server replies, to ensure data integrity.
- `timeo=600` – Sets the timeout value that the NFS client uses to wait for a response before it retries an NFS request to 600 deciseconds (60 seconds) to avoid diminished performance.
- `retrans=2` – Sets to 2 the number of times the NFS client retries a request before it attempts further recovery action.
- `noresvport` – Tells the NFS client to use a new non-privileged Transmission Control Protocol (TCP) source port when a network connection is reestablished. Using the `noresvport` option helps to ensure that your EFS file system has uninterrupted availability after a reconnection or network recovery event.
- `mountport=2049` – only used when mounting on EC2 Mac instances running macOS Big Sur, Monterey, and Ventura.

Getting support logs

The EFS mount helper has built-in logging for your EFS file system. You can share these logs with AWS Support for troubleshooting purposes. You can find the logs stored in `/var/log/amazon/efs` on clients using the EFS mount helper. These logs are for the EFS mount helper, the stunnel process (disabled by default), and for the `amazon-efs-mount-watchdog` process that monitors the stunnel process.

> **Note**
>
> The `amazon-efs-mount-watchdog` process ensures that each mount's stunnel process is running, and stops the stunnel process when the EFS file system is unmounted. If for some reason a stunnel process is terminated unexpectedly, the watchdog process will restart it.

You can change the log configuration in `/etc/amazon/efs/efs-utils.conf`. In order for any log changes to take effect, you need to unmount and remount the file system using the EFS mount helper. Log capacity for the mount helper and watchdog logs is limited to 20 MiB. Logs for the stunnel process are disabled by default.

> **Important**
>
> You can enable logging for the stunnel process logs. However, enabling the stunnel logs can use up a nontrivial amount of space on your file system.

Prerequisites for using the EFS mount helper

You can mount an EFS file system on an Amazon EC2 instance using the Amazon EFS mount helper. To use the mount helper, you need the following:

- **File system ID of the file system to mount** - The EFS mount helper resolves the file system ID to the local IP address of the mount target elastic network interface (ENI) without calling external resources.
- **An EFS mount target** – You create mount targets in your virtual private cloud (VPC). If you create your file system in the console using the service recommended settings, a mount target is created in each Availability Zone in the AWS Region that the file system is in. For instructions to create mount targets, see Managing mount targets.

> **Note**
>
> We recommend that you wait 60 seconds after the newly created mount target's lifecycle state is **available** before mounting the file system via DNS. This wait lets the DNS records propagate fully in the AWS Region where the file system resides.

If you use a mount target in an Availability Zone different from that of your EC2 instance, you incur standard EC2 charges for data sent across Availability Zones. You also might see increased latencies for file system operations.

- For mounting One Zone file systems from a different Availability Zone:
  - **The name of the file system's Availability Zone –** If you are mounting an EFS One Zone file system that is located in a different Availability Zone than the EC2 instance.
  - **Mount target DNS name –** Alternatively, you can specify the mount target's DNS name instead of the Availability Zone.
- **An EC2 instance running one of the supported Linux or macOS distributions –** The supported distributions for mounting your file system with the mount helper are the following:
  - Amazon Linux 2
  - Amazon Linux 2023
  - Amazon Linux 2017.09 and newer
  - macOS Big Sur
  - Red Hat Enterprise Linux (and derivatives such as CentOS) version 7 and newer
  - Ubuntu 16.04 LTS and newer

> **Note**
>
> EC2 Mac instances running macOS Big Sur support NFS 4.0 only.

- **The EFS mount helper is installed on the EC2 instance –** The mount helper is a tool in the `amazon-efs-utils` package of utilities. For information about installing `amazon-efs-utils`, see Installing the Amazon EFS client.
- **The EC2 instance is in a VPC –** The connecting EC2 instance must be in a virtual private cloud (VPC) based on the Amazon VPC service. It also must be configured to use the DNS server

provided by AWS. For information about the Amazon DNS server, see DHCP option sets in Amazon VPC in the *Amazon VPC User Guide*.

- **VPC has DNS hostnames enabled –** The VPC of the connecting EC2 instance must have DNS hostnames enabled. For more information, see DNS attributes for your VPC in the *Amazon VPC User Guide*.
- **For EC2 instances and file systems in different AWS Regions –** If the EC2 instance and the file system you are mounting are located in different AWS Regions, you will need to edit the `region` property in the `efs-utils.conf` file. For more information, see Mounting EFS file systems from a different AWS Region.

Mounting on EC2 Linux instances using the EFS mount helper

This procedure requires the following:

- You have installed the `amazon-efs-utils` package on the Amazon EC2 instance. For more information, see Manually installing the Amazon EFS client.
- You have created mount targets for the file system. For more information, see Managing mount targets.

To mount your EFS file system using the mount helper on EC2 Linux instances

1. Open a terminal window on your EC2 instance through Secure Shell (SSH), and log in with the appropriate user name. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Create a directory `efs` that you will use as the file system mount point using the following command:

   ```
   sudo mkdir efs
   ```

3. Run one of the following commands to mount your file system.

   > **Note**
   >
   > If the EC2 instance and the file system you are mounting are located in different AWS Regions, see Mounting EFS file systems from a different AWS Region to edit the `region` property in the `efs-utils.conf` file.

- To mount using the file system id:

```
sudo mount -t efs file-system-id efs-mount-point/
```

Use the ID of the file system you are mounting in place *`file-system-id`* and `efs` in place of *`efs-mount-point`*.

```
sudo mount -t efs fs-abcd123456789ef0 efs/
```

Alternatively, if you want to use encryption of data in transit, you can mount your file system with the following command.

```
sudo mount -t efs -o tls fs-abcd123456789ef0:/ efs/
```

- To mount using the file system DNS name:

```
sudo mount -t efs -o tls file-system-dns-name efs-mount-point/
```

```
sudo mount -t efs -o tls fs-abcd123456789ef0.efs.us-east-2.amazonaws.com efs/
```

- To mount using the mount target IP address:

```
sudo mount -t efs -o tls,mounttargetip=mount-target-ip file-system-id efs-mount-point/
```

```
sudo mount -t efs -o tls,mounttargetip=192.0.2.0 fs-abcd123456789ef0 efs/
```

You can view and copy the exact commands to mount your file system in the **Attach** dialog box.

a. In the Amazon EFS console, choose the file system that you want to mount to display its details page.

b. To display the mount commands to use for this file system, choose **Attach** in the upper right.

The **Attach** screen displays the exact commands to use for mounting the file system in the following ways:

- (**Mount via DNS**) Using the file system's DNS name with the EFS mount helper or an NFS client.
- (**Mount via IP**) Using the mount target IP address in the selected Availability Zone with an NFS client.

Mounting on EC2 Mac instances using the EFS mount helper

This procedure requires the following:

- You have installed the `amazon-efs-utils` package on the Amazon EC2 Mac instance. For more information, see Installing the Amazon EFS client on EC2 Mac instances running macOS Big Sur, macOS Monterey, or macOS Ventura.
- You have created mount targets for the file system. You can create mount targets at file system creation and add them to existing file systems. For more information, see Managing mount targets.
- You are mounting the file system on an EC2 Mac instance running macOS Big Sur, Monterey, or Ventura. Other macOS versions are not supported.

> **Note**
>
> Only EC2 Mac instances running macOS Big Sur, Monterey, and Ventura are supported. Other macOS versions are not supported for use with Amazon EFS.

To mount your EFS file system using the EFS mount helper on EC2 Mac instances running macOS Big Sur, Monterey, or Ventura

1. Open a terminal window on your EC2 Mac instance through Secure Shell (SSH), and log in with the appropriate user name. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Create a directory to use as the file system mount point using the following command:

```
sudo mkdir efs
```

3. Run the following command to mount your file system.

   Note

   By default, the EFS mount helper uses encryption in transit when mounting on EC2 Mac instances, whether or not you use the `tls` option in the mount command.

   ```
   sudo mount -t efs file-system-id efs-mount-point/
   ```

   ```
   sudo mount -t efs fs-abcd123456789ef0 efs/
   ```

   You can also use the `tls` option when mounting.

   ```
   sudo mount -t efs -o tls fs-abcd123456789ef0:/ efs
   ```

   To mount a file system on an EC2 Mac instance without using encryption in transit, use the `notls` option, as shown in the following command.

   ```
   sudo mount -t efs -o notls file-system-id efs-mount-point/
   ```

   You can view and copy the exact commands to mount your file system in the management console's **Attach** dialog box, described as follows.

   a. In the Amazon EFS console, choose the file system that you want to mount to display its details page.

   b. To display the mount commands to use for this file system, choose **Attach** in the upper right.

      The **Attach** screen displays the exact commands to use for mounting the file system in the following ways:

      - (**Mount via DNS**) Using the file system's DNS name with the EFS mount helper or an NFS client.

- (**Mount via IP**) Using the mount target IP address in the selected Availability Zone with an NFS client.

Mounting EFS file systems from a different AWS Region

To mount your EFS file system from an EC2 instance that is in a different AWS Region than the file system, you must edit the `region` property value in the `efs-utils.conf` file.

To edit the `region` property in `efs-utils.conf`

1. Access the terminal for your EC2 instance through Secure Shell (SSH), and log in with the appropriate user name. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Locate the `efs-utils.conf` file, and open it using your preferred editor.
3. Locate the following line:

```
#region = us-east-1
```

   a. Uncomment the line.
   b. If the file system is not located in the `us-east-1` region, replace `us-east-1` with the ID of the region in which the file system is located.
   c. Save the changes.
4. Add a host entry for the cross region mount. For more information on how to do this, see Step 3: Add a host entry for the mount target.
5. Mount the file system using the EFS mount helper for Linux or Mac instances.

Mounting One Zone file systems

EFS One Zone file systems support only a single mount target which is located in the same Availability Zone as the file system. You cannot add additional mount targets. This section describes things to consider when mounting One Zone file systems.

You can avoid data transfer charges between Availability Zones and achieve better performance by accessing an EFS file system using an Amazon EC2 compute instance that is located in the same Availability Zone as that of the file system's mount target.

The procedures in this section require the following:

- You have installed the `amazon-efs-utils package` on the EC2 instance. For more information, see Installing the Amazon EFS client.
- You have created a mount target for the file system. For more information, see Managing mount targets.

Mounting One Zone file systems on EC2 in a different Availability Zone

If you are mounting a One Zone file system on an Amazon EC2 instance that is located in a different Availability Zone, you have to specify the file system's Availability Zone name or the DNS name of the file system's mount target in the mount helper mount command.

Create a directory called `efs` to use as the file system mount point using the following command:

```
sudo mkdir efs
```

Use the following command to mount the file system using the EFS mount helper. The command specifies the file system's Availability Zone name.

```
sudo mount -t efs -o az=availability-zone-name,tls file-system-id mount-point/
```

This is the command with sample values:

```
sudo mount -t efs -o az=us-east-1a,tls fs-abcd1234567890ef efs/
```

The following command mounts the file system, specifying the DNS name of the file system's mount target.

```
sudo mount -t efs -o tls mount-target-dns-name mount-point/
```

This is the command with an example mount target DNS name.

```
sudo mount -t efs -o tls us-east-1a.fs-abcd1234567890ef9.efs.us-east-1.amazonaws.com
 efs/
```

Mounting One Zone file systems in a different Availability Zone automatically with EFS mount helper

If you are using /etc/fstab to mount an EFS One Zone file system on an EC2 instance that is located in a different Availability Zone, you have to specify the file system's Availability Zone name or the DNS name of the file system's mount target in the /etc/fstab entry.

```
availability-zone-name.file-system-id.efs.aws-region.amazonaws.com:/ efs-mount-point
 efs defaults,_netdev,noresvport,tls 0 0
```

```
us-east-1a.fs-abc123def456a7890.efs.us-east-1.amazonaws.com:/ efs-one-zone efs
 defaults,_netdev,noresvport,tls 0 0
```

Mounting One Zone file systems automatically with NFS

If you are using /etc/fstab to mount an EFS file system using One Zone storage on an EC2 instance that is located in a different Availability Zone, you have to specify the file system's Availability Zone name with the file system's DNS name in the /etc/fstab entry.

```
availability-zone-name.file-system-id.efs.aws-region.amazonaws.com:/ efs-mount-point
 nfs4
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport,_netdev 0
 0
```

```
us-east-1a.fs-abc123def456a7890.efs.us-east-1.amazonaws.com:/ efs-one-zone nfs4
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport,_netdev 0
 0
```

For more information about how to edit the /etc/fstab file, and the values used in this command, see Automatically mounting EFS file systems.

Mounting file systems with One Zone file system on other AWS compute instances

When you use a One Zone file system with Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, or AWS Lambda, you need to configure the service to use the same Availability Zone that the EFS file system is located in, illustrated as follows, and described in the following sections.

Connecting from Amazon Elastic Container Service

You can use EFS file systems with Amazon ECS to share file system data across your fleet of container instances so your tasks have access to the same persistent storage, no matter the instance on which they land. To use EFS One Zone file systems with Amazon ECS you should choose only subnets that are in the same Availability Zone as your file system when launching your task. For more information, see [Amazon EFS volumes](#) in the *Amazon Elastic Container Service Developer Guide*.

Connecting from Amazon Elastic Kubernetes Service

When mounting an One Zone file system from Amazon EKS, you can use the Amazon EFS [Container Storage Interface](#) (CSI) driver, which supports EFS access points, to share a file system between multiple pods in an Amazon EKS or self-managed Kubernetes cluster. The Amazon EFS CSI driver is installed in the Fargate stack. When using the Amazon EFS CSI driver with EFS One Zone file systems, you can use the `nodeSelector` option when launching your pod to ensure it gets scheduled within the same Availability Zone as your file system.

Connecting from AWS Lambda

You can use Amazon EFS with AWS Lambda to share data across function invocations, read large reference data files, and write function output to a persistent and shared store. Lambda securely connects the function instances to the EFS mount targets that are in the same Availability Zone and subnet. When you use Lambda with One Zone file systems, configure your function to only launch invocations into subnets that are in the same Availability Zone as your file system.

Mounting with IAM authorization

To mount your EFS file system on Linux instances using AWS Identity and Access Management (IAM) authorization, use the EFS mount helper. For more information about IAM authorization for NFS clients, see Using IAM to control access to file systems.

You need to create a directory to use as the file system mount point in the following sections. You can use the following command to create a mount point directory `efs`:

```
sudo mkdir efs
```

You can then replace instances of *efs-mount-point* with `efs`.

Mounting with IAM using an EC2 instance profile

If you are mounting with IAM authorization to an Amazon EC2 instance with an instance profile, use the `tls` and `iam` mount options, shown following.

```
$ sudo mount -t efs -o tls,iam file-system-id efs-mount-point/
```

To automatically mount with IAM authorization to an EC2 instance that has an instance profile, add the following line to the `/etc/fstab` file on the EC2 instance.

```
file-system-id:/ efs-mount-point efs _netdev,tls,iam 0 0
```

Mounting with IAM using a named profile

You can mount with IAM authorization using the IAM credentials located in the AWS CLI credentials file `~/.aws/credentials`, or the AWS CLI config file `~/.aws/config`. If `"awsprofile"` is not specified, the "default" profile is used.

To mount with IAM authorization to a Linux instance using a credentials file, use the `tls`, `awsprofile`, and `iam` mount options, shown following.

```
$ sudo mount -t efs -o tls,iam,awsprofile=namedprofile file-system-id efs-mount-point/
```

To automatically mount with IAM authorization to a Linux instance using a credentials file, add the following line to the `/etc/fstab` file on the EC2 instance.

```
file-system-id:/ efs-mount-point efs _netdev,tls,iam,awsprofile=namedprofile 0 0
```

Mounting with EFS access points

You can mount an EFS file system using an EFS access point only by using the EFS mount helper.

> **Note**
>
> You must configure one or more mount targets for your file system when mounting a file system using EFS access points.

When you mount a file system using an access point, the mount command includes the `access-point-id` and the `tls` mount option in addition to the regular mount options. An example is shown following.

```
$ sudo mount -t efs -o tls,accesspoint=access-point-id file-system-id efs-mount-point
```

To automatically mount a file system using an access point, add the following line to the `/etc/fstab` file on the EC2 instance.

```
file-system-id efs-mount-point efs _netdev,tls,accesspoint=access-point-id 0 0
```

For more information about EFS access points, see Working with access points.

Mounting EFS to multiple EC2 instances

You can mount EFS file systems to multiple Amazon EC2 instances remotely and securely without having to log in to the instances by using the AWS Systems Manager Run Command. For more information about AWS Systems Manager Run Command, see AWS Systems Manager Run Command in the *AWS Systems Manager User Guide*. The following prerequisites are required before mounting EFS file systems using this method:

1. The EC2 instances are launched with an instance profile that includes the `AmazonElasticFileSystemsUtils` permissions policy. For more information, see Step 1: Configure an IAM instance profile with the required permissions.
2. Version 1.28.1 or later of the Amazon EFS client (amazon-efs-utils package) is installed on the EC2 instances. You can use AWS Systems Manager to automatically install the package on your instances. For more information, see Step 2: Configure an association used by State Manager.

To mount multiple EFS file systems to multiple EC2 instances using the console

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.
3. Choose **Run a command**.
4. Enter **`AWS-RunShellScript`** in the **Commands** search field.
5. Select **AWS-RunShellScript**.
6. In **Command parameters** enter the mount command to use for each EFS file system that you want to mount. For example:

   ```
   sudo mount -t efs -o tls fs-12345678:/ /mnt/efs
   sudo mount -t efs -o tls,accesspoint=fsap-12345678 fs-01233210 /mnt/efs
   ```

   For more information about EFS mount commands using the Amazon EFS client, see Mounting on EC2 Linux instances using the EFS mount helper or Mounting on EC2 Mac instances using the EFS mount helper.
7. Select the target AWS Systems Manager managed EC2 instances that you want the command to run on.
8. Make any other additional settings you would like. Then choose **Run** to run the command and mount the EFS file systems specified in the command.

   Once you run the command, you can see its status in the command history.

Mounting EFS file systems from another AWS account or VPC

You can mount your EFS file system using IAM authorization for NFS clients and EFS access points using the EFS mount helper. By default, the EFS mount helper uses domain name service (DNS) to resolve the IP address of your EFS mount target. If you are mounting the file system from a different account or virtual private cloud (VPC), you need to resolve the EFS mount target manually.

Following, you can find instructions for determining the correct EFS mount target IP address to use for your NFS client. You can also find instructions for configuring the client to mount the EFS file system using that IP address.

Topics

- Mounting EFS file systems from another AWS account
- Mounting EFS file systems from another VPC

Mounting EFS file systems from another AWS account

Using shared VPCs, you can mount an EFS file system that is owned by one AWS account from Amazon EC2 instances that are owned by a different AWS account. For more information about setting up a shared VPC, see Share your VPC with other accounts in the *Amazon VPC Peering Guide*.

After you set up VPC sharing, the EC2 instances can mount the EFS file system using Domain Name System (DNS) name resolution or the EFS mount helper. We recommend using the EFS mount helper to mount your EFS file systems.

Mounting EFS file systems from another VPC

When you use a VPC peering connection or transit gateway to connect VPCs, Amazon EC2 instances that are in one VPC can access EFS file systems in another VPC, even if the VPCs belong to different accounts.

You can't use DNS name resolution for EFS mount points in another VPC. To mount your EFS file system, use the IP address of the mount points in the corresponding Availability Zone.

Alternatively, you can use Amazon Route 53 as your DNS service. In Route 53, you can resolve the EFS mount target IP addresses from another VPC by creating a private hosted zone and resource record set. For more information on how to do so, see Working with private hosted zones in the *Amazon Route 53 Developer Guide*.

Prerequisites

Before using the following the procedure, take these steps:

- Install the Amazon EFS client, part of the `amazon-efs-utils` set of utilities on the compute instance you're mounting the EFS file system on. You use the EFS mount helper, which is included in `amazon-efs-utils`, to mount the file system. For instructions on installing `amazon-efs-utils`, see [Installing the Amazon EFS client](#).
- Allow the `ec2:DescribeAvailabilityZones` action in the IAM policy for the IAM role you attached to the instance. We recommend that you attach the AWS managed policy `AmazonElasticFileSystemsUtils` to an IAM entity to provide the necessary permissions for the entity.
- When mounting from another AWS account, update the file system resource policy to allow the `elasticfilesystem:DescribeMountTarget` action for the principal ARN of other AWS account. For example:

```
{
    "Id": "access-point-example03",
    "Statement": [
        {
            "Sid": "access-point-statement-example03",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::555555555555:root"},
            "Action": "elasticfilesystem:DescribeMountTargets",
            "Resource": "arn:aws:elasticfilesystem:us-east-2:111122223333:file-
system/fs-12345678"
        }
    ]
}
```

  For more information about EFS file system resource policies, see [Resource-based policies within Amazon EFS](#).
- Install botocore. The EFS client uses botocore to retrieve the mount target IP address when the file system DNS name cannot be resolved when mounting a file system in another VPC. For more information, see [Install botocore](#) in the `amazon-efs-utils` README file.
- Set up either a VPC peering connection or a VPC transit gateway.

  You connect the client's VPC and your EFS file system's VPC using either a VPC peering connection or a VPC transit gateway. When you use a VPC peering connection or transit gateway to connect VPCs, Amazon EC2 instances that are in one VPC can access EFS file systems in another VPC, even if the VPCs belong to different accounts.

A *transit gateway* is a network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information about using VPC transit gateways, see Getting Started with transit gateways in the *Amazon VPC Transit Gateways Guide*.

A *VPC peering connection* is a networking connection between two VPCs. This type of connection enables you to route traffic between them using private Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6) addresses. You can use VPC peering to connect VPCs within the same AWS Region or between AWS Regions. For more information on VPC peering, see What is VPC Peering? in the *Amazon VPC Peering Guide*.

To ensure high availability of your file system, we recommend that you always use an EFS mount target IP address that is in the same Availability Zone as your NFS client. If you're mounting an EFS file system that is in another account, ensure that the NFS client and EFS mount target are in the same Availability Zone ID. This requirement applies because AZ names can differ from one account to another.

To mount an EFS file system in another VPC using IAM or an access point

1. Connect to your EC2 instance. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Create a directory for mounting the file system using the following command.

```
$ sudo mkdir /mnt/efs
```

3. To mount the file system using IAM authorization, use the following command:

```
$ sudo mount -t efs -o tls,iam file-system-dns-name /mnt/efs/
```

   For more information about using IAM authorization with EFS, see Using IAM to control access to file systems.

   To mount the file system using an EFS access point, use the following command:

```
$ sudo mount -t efs -o tls,accesspoint=access-point-id file-system-dns-name /mnt/
efs/
```

For more information about EFS access points, see Working with access points.

Mounting EFS file systems from a different AWS Region

If you are mounting your EFS file system from another VPC that is in a different AWS Region than the file system, you will need to edit the `efs-utils.conf` file. In `/dist/efs-utils.conf`, locate the following lines:

```
#region = us-east-1
```

Uncomment the line, and replace the value for the ID of the region in which the file system is located, if it is not in `us-east-1`.

Using Network File System to mount EFS file systems

Following, learn how to install the Network File System (NFS) client and how to mount your Amazon EFS file system on an Amazon EC2 instance. You also can find an explanation of the `mount` command and the available options for specifying your file system's Domain Name System (DNS) name in the `mount` command. In addition, you can find how to use the file `fstab` to automatically remount your file system after any system restarts.

> **Note**
>
> In this section, you can learn how to mount your Amazon EFS file system without the amazon-efs-utils package. To use encryption of data in transit with your file system, you must mount your file system with Transport Layer Security (TLS). To do so, we recommend using the amazon-efs-utils package. For more information, see Installing the Amazon EFS client.

Topics

- Prerequisites
- NFS support
- Installing the NFS client
- Recommended NFS mount settings

- Mounting on Amazon EC2 with a DNS name
- Mounting with an IP address

Prerequisites

Before you can mount a file system, make sure you meet the following requirements:

- Create, configure, and launch your related AWS resources. For instructions, see Getting started with Amazon EFS.
- Create VPC security groups for your Amazon EC2 instances and mount targets with the required inbound and outbound access. For more information, see Using VPC security groups.

NFS support

Amazon EFS supports the Network File System versions 4.0 and 4.1 (NFSv4) protocols when mounting your file systems on Amazon EC2 instances. Although NFSv4.0 is supported, we recommend that you use NFSv4.1. Mounting your Amazon EFS file system on your Amazon EC2 instance also requires an NFS client that supports your chosen NFSv4 protocol. Amazon EC2 Mac instances running macOS Big Sur only support NFS v4.0.

Amazon EFS does not support the `nconnect` mount option.

> **Note**
>
> For Linux kernel versions 5.4.*, the Linux NFS client uses a default `read_ahead_kb` value of 128 KB. We recommend increasing this value to 15 MB. For more information, see Optimizing the NFS read_ahead_kb size.

For optimal performance and to avoid a variety of known NFS client bugs, we recommend working with a recent Linux kernel. If you are using an enterprise Linux distribution, we recommend the following:

- Amazon Linux 2
- Amazon Linux 2017.09 or newer
- Red Hat Enterprise Linux (and derivatives such as CentOS) version 7 and newer
- Ubuntu 16.04 LTS and newer

- SLES 12 Sp2 or later

If you are using another distribution or a custom kernel, we recommend kernel version 4.3 or newer. To troubleshoot issues related to certain AMI or kernel versions when using Amazon EFS from an EC2 instance, see Troubleshooting AMI and kernel issues.

> **Note**
>
> Mounting EFS file systems with Amazon EC2 instances running Microsoft Windows is not supported.

Installing the NFS client

To mount your EFS file system on your Amazon EC2 instance, first you need to install an NFS client. To connect to your EC2 instance and install an NFS client, you need the public DNS name of the EC2 instance and a user name to log in. That user name for your instance is typically `ec2-user`.

To connect your EC2 instance and install the NFS client

1. Connect to your EC2 instance. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.

   The key file cannot be publicly viewable for SSH. You can use the **chmod 400 *filename*.pem** command to set these permissions. For more information, see Create a key pair for your Amazon EC2 instance.
2. (Optional) Get updates and reboot.

```
$ sudo yum -y update
            $  sudo reboot
```

3. After the reboot, reconnect to your EC2 instance.
4. Install the NFS client.

   If you're using an Amazon Linux AMI or Red Hat Linux AMI, install the NFS client with the following command.

```
$ sudo yum -y install nfs-utils
```

If you're using an Ubuntu Amazon EC2 AMI, install the NFS client with the following command.

```
$ sudo apt-get -y install nfs-common
```

5. Start the NFS service using the following commands. For RHEL 7:

```
$ sudo service nfs start
```

For RHEL 8:

```
$ sudo service nfs-server start
```

6. Verify that the NFS service started, as follows.

```
$ sudo service nfs status
Redirecting to /bin/systemctl status nfs.service
# nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor
 preset: disabled)
   Active: active (exited) since Wed 2019-10-30 16:13:44 UTC; 5s ago
  Process: 29446 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/
SUCCESS)
  Process: 29441 ExecStartPre=/bin/sh -c /bin/kill -HUP `cat /run/gssproxy.pid`
 (code=exited, status=0/SUCCESS)
  Process: 29439 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
 Main PID: 29446 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/nfs-server.service
```

If you use a custom kernel (that is, if you build a custom AMI), you need to include at a minimum the NFSv4.1 client kernel module and the right NFS4 userspace mount helper.

Note

If you choose **Amazon Linux AMI 2016.03.0** or **Amazon Linux AMI 2016.09.0** when launching your Amazon EC2 instance, you don't need to install `nfs-utils` because it's already included in the AMI by default.

Next: Mount your file system

Use one of the following procedures to mount your file system.

- Mounting on Amazon EC2 with a DNS name
- Mounting with an IP address
- Automatically mounting EFS file systems

Recommended NFS mount settings

We recommend the following values for mount options on Linux:

- `noresvport` – Tells the NFS client to use a new non-privileged Transmission Control Protocol (TCP) source port when a network connection is reestablished. NFS client software included in older versions of the Linux kernel (versions v5.4 and below) include a behavior that causes NFS clients to, upon disconnection, attempt reconnecting on the same TCP source port. This behavior does not comply with the TCP RFC, and can prevent these clients from quickly re-establishing connections to an EFS file system.

  Using `noresvport` option helps to ensure that NFS clients reconnect transparently to your EFS file system, maintaining uninterrupted availability when reconnecting after a network recovery event.

  > **Important**
  >
  > We strongly recommend using the `noresvport` mounting option to help ensure that your EFS file system has uninterrupted availability after a reconnection or network recovery event.
  > Consider using the EFS mount helper to mount your file systems. The EFS mount helper uses NFS mount options optimized for Amazon EFS file systems.

- `rsize=1048576` – Sets the maximum number of bytes of data that the NFS client can receive for each network READ request. This value applies when reading data from a file on an EFS file system. We recommend that you use the largest size possible (up to `1048576`) to avoid diminished performance.
- `wsize=1048576` – Sets the maximum number of bytes of data that the NFS client can send for each network WRITE request. This value applies when writing data to a file on an EFS file system. We recommend that you use the largest size possible (up to `1048576`) to avoid diminished performance.

- `hard` – Sets the recovery behavior of the NFS client after an NFS request times out, so that NFS requests are retried indefinitely until the server replies. We recommend that you use the hard mount option (`hard`) to ensure data integrity. If you use a `soft` mount, set the `timeo` parameter to at least `150` deciseconds (15 seconds). Doing so helps minimize the risk of data corruption that is inherent with soft mounts.
- `timeo=600` – Sets the timeout value that the NFS client uses to wait for a response before it retries an NFS request to 600 deciseconds (60 seconds). If you must change the timeout parameter (`timeo`), we recommend that you use a value of at least `150`, which is equivalent to 15 seconds. Doing so helps avoid diminished performance.
- `retrans=2` – Sets to 2 the number of times the NFS client retries a request before it attempts further recovery action.
- `_netdev` – When present in `/etc/fstab`, prevents the client from attempting to mount the EFS file system until the network has been enabled.
- `nofail` – If your EC2 instance needs to start regardless of the status of your mounted EFS file system, add the `nofail` option to your file system's entry in your `/etc/fstab` file.

If you don't use the preceding defaults, be aware of the following:

- In general, avoid setting any other mount options that are different from the defaults, which can cause reduced performance and other issues. For example, changing read or write buffer sizes or disabling attribute caching can result in reduced performance.
- Amazon EFS ignores source ports. If you change Amazon EFS source ports, it doesn't have any effect.
- Amazon EFS does not support the `nconnect` mount option.
- Amazon EFS doesn't support any of the Kerberos security variants. For example, the following mount command fails.

```
$ mount -t nfs4 -o krb5p <DNS_NAME>:/ /efs/
```

- We recommend that you mount your file system using its DNS name. This name resolves to the IP address of the Amazon EFS mount target in the same Availability Zone as your Amazon EC2 instance. If you use a mount target in an Availability Zone different from that of your Amazon EC2 instance, you incur standard EC2 charges for data sent across Availability Zones. You also might see increased latencies for file system operations.

- For more mount options, and detailed explanations of the defaults, see the Linux documentation.

Mounting on Amazon EC2 with a DNS name

> **Note**
>
> Prior to mounting your file system, you need to add a rule to the mount target security group that allows inbound NFS access from the EC2 security group. For more information, see Using VPC security groups.

- **File system DNS name –** Using the file system's DNS name is your simplest mounting option. The file system DNS name automatically resolves to the mount target’s IP address in the Availability Zone of the connecting Amazon EC2 instance. You can get the DNS name from the console, or if you have the file system ID, you can construct it using the following convention.

  ```
  file-system-id.efs.aws-region.amazonaws.com
  ```

  > **Note**
  >
  > DNS resolution for file system DNS names requires that the Amazon EFS file system has a mount target in the same Availability Zone as the client instance.

  - Using the file system DNS name, you can mount a file system on your Amazon EC2 Linux instance with the following command.

    ```
    sudo mount -t nfs -o
     nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport file-
    system-id.efs.aws-region.amazonaws.com:/ /efs-mount-point
    ```

  - Using the file system DNS name, you can mount a file system on your Amazon EC2 Mac instance running a supported macOS version (Big Sur, Monterey, Ventura) with the following command.

```
sudo mount -t nfs -o
 nfsvers=4.0,rsize=65536,wsize=65536,hard,timeo=600,retrans=2,noresvport,mountport=2049 fil
system-id.efs.aws-region.amazonaws.com:/ /efs
```

> **Important**
>
> You must use `mountport=2049` in order to successfully connect to the EFS file system when mounting on EC2 Mac instances running support macOS versions.

- **Mount target DNS name** – In December 2016, we introduced file system DNS names. We continue to provide a DNS name for each Availability Zone mount target for backward compatibility. The generic form of a mount target DNS name is as follows.

  ```
  availability-zone.file-system-id.efs.aws-region.amazonaws.com
  ```

  > **Note**
  >
  > Mount target DNS name resolution across Availability Zones is supported.

  In some cases, you might delete a mount target and then create a new one in the same Availability Zone. In such a case, the DNS name for that new mount target in that Availability Zone is the same as the DNS name for the old mount target.

You can view and copy the exact commands to mount your file system in the **Attach** dialog box.

To view the mount commands for your file system

1. In the Amazon EFS console, choose the file system that you want to mount to display its details page.
2. To display the mount commands to use for this file system, choose **Attach** in the upper right.

   The **Attach** screen displays the exact commands to use for mounting the file system.
3. The default **Mount via DNS** view displays the command to mount the file system using the file system's DNS name when mounting with the EFS mount helper or an NFS client.

For a list of AWS Regions that support Amazon EFS, see Amazon Elastic File System in the AWS General Reference.

To be able to use a DNS name in the `mount` command, the following must be true:

- The connecting EC2 instance must be inside a VPC and must be configured to use the DNS server provided by Amazon. For information about Amazon DNS server, see DHCP option sets in Amazon VPC in the *Amazon VPC User Guide*.
- The VPC of the connecting EC2 instance must have both **DNS Resolution** and **DNS Hostnames** enabled. For more information, see DNS attributes for your VPC in the *Amazon VPC User Guide*.
- The connecting EC2 instance must be inside the same VPC as the EFS file system. For more information on accessing and mounting a file system from another location or from a different VPC, see Prerequisites and Tutorial: Mount a file system from a different VPC.

> **Note**
>
> We recommend that you wait 90 seconds after creating a mount target before you mount your file system. This wait lets the DNS records propagate fully in the AWS Region where the file system is.

Mounting with an IP address

As an alternative to mounting your Amazon EFS file system with the DNS name, Amazon EC2 instances can mount a file system using a mount target's IP address. Mounting by IP address works in environments where DNS is disabled, such as VPCs with DNS hostnames disabled.

You can also configure mounting a file system using the mount target IP address as a fallback option for applications configured to mount the file system using its DNS name by default. When connecting to a mount target IP address, EC2 instances should mount using the mount target IP address in the same Availability Zone as the connecting instance.

You can view and copy the exact commands to mount your file system in the **Attach** dialog box.

> **Note**
>
> Prior to mounting your file system, you need to add a rule for the mount target security group to allow inbound NFS access from the EC2 security group. For more information, see Using VPC security groups.

To view and copy the exact commands to mount your EFS file system using the mount target IP address

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the Amazon EFS console, choose the file system that you want to mount to display its details page.
3. To display the mount commands to use for this file system, choose **Attach** in the upper right.
4. The **Attach** screen displays the exact commands to use for mounting the file system.

   Choose **Mount via IP** to display the command to mount the file system using the mount target IP address in the selected Availability Zone with an NFS client.

- Using the IP address of a mount target in the `mount` command, you can mount a file system on your Amazon EC2 Linux instance with the following command.

```
sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-
target-IP:/   /efs
```

- Using the IP address of a mount target in the `mount` command, you can mount a file system on your Amazon EC2 Mac instance running macOS Big Sur with the following command.

```
sudo mount -t nfs -o
 nfsvers=4.0,rsize=65536,wsize=65536,hard,timeo=600,retrans=2,noresvport,mountport=2049 mount-
target-IP:/ /efs
```

> **Important**
>
> You must use `mountport=2049` in order to successfully connect to the EFS file system when mounting on EC2 Mac instances running macOS Big Sur.

Mounting with an IP address in AWS CloudFormation

You can also mount your file system using an IP address in an AWS CloudFormation template. For more information, see storage-efs-mountfilesystem-ip-addr.config in the **awsdocs/elastic-beanstalk-samples** repository for community-provided configuration files on GitHub.

Automatically mounting EFS file systems

You can use the EFS mount helper or NFS to configure an Amazon EC2 instance to automatically mount an EFS file system when the instance starts.

- Using the EFS mount helper:
  - Attach an EFS file system when you create a new EC2 Linux instance using the EC2 launch instance wizard.
  - Update the EC2 `/etc/fstab` file with an entry for the EFS file system.
- Using NFS without the EFS mount helper to update the EC2 `/etc/fstab` file, for EC2 Linux and Mac instances.

> **Note**
>
> The EFS mount helper does not support automatic mounting on EC2 Mac instances running macOS Big Sur or Monterey. Instead, you can use NFS to configure the /etc/fstab file on an EC2 Mac instance to automatically mount an EFS file system.

Topics

- Enabling automatic mounting on new EC2 Linux instances
- Enabling automatic mounting on existing EC2 Linux instances
- Enabling automatic mounting on EC2 Linux or Mac instances using NFS

Enabling automatic mounting on new EC2 Linux instances

When you create a new EC2 Linux instance using the Amazon EC2 launch instance wizard, you can configure it to mount your Amazon EFS file system automatically. The EC2 instance mounts the file system automatically the instance first launched and also whenever it restarts.

This method uses the EFS mount helper to mount the file system update the /etc/fstab file on the EC2 instance. The mount helper is part of the `amazon-efs-utils` set of tools.

> **Note**
>
> EFS file systems do not support mounting on EC2 Mac instances running macOS Big Sur or Monterey at instance launch.

> **Note**
>
> You can't use Amazon EFS with Microsoft Windows–based EC2 instances.

Before you can launch and connect to an EC2 instance, you need to create a key pair. For more information, see [Amazon EC2 key pairs and Amazon EC2 instances](#) in the *Amazon EC2 User Guide* to create a key pair.

To configure your EC2 instance to mount an EFS file system automatically at launch

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose **Launch Instance**.
3. In **Step 1: Choose an Amazon Machine Image (AMI)**, find an Amazon Linux AMI at the top of the list and choose **Select**.
4. In **Step 2: Choose an Instance Type**, choose **Next: Configure Instance Details**.
5. In **Step 3: Configure Instance Details**, provide the following information:
   - For **Network**, choose the entry for the same VPC that the EFS file system you're mounting is in.
   - For **Subnet**, choose a default subnet in any Availability Zone.
   - For **File systems**, choose the EFS file system that you want to mount. The path shown next the file system ID is the mount point that the EC2 instance will use, which you can change.
   - Under **Advanced Details**, the **User data** is automatically generated, and includes the commands needed to mount the EFS file systems you specified under **File systems**.
6. Choose **Next: Add Storage**.
7. Choose **Next: Add Tags**.

8. Name your instance and choose **Next: Configure Security Group**.
9. In **Step 6: Configure Security Group**, set **Assign a security group** to **Select an existing security group**. Choose the default security group to make sure that it can access your EFS file system.

   You can't access your EC2 instance by Secure Shell (SSH) using this security group. For access by SSH, later you can edit the default security and add a rule to allow SSH or a new security group that allows SSH. You can use the following settings:

   - **Type:** SSH
   - **Protocol:** TCP
   - **Port Range:** 22
   - **Source:** Anywhere 0.0.0.0/0
10. Choose **Review and Launch**.
11. Choose **Launch**.
12. Select the check box for the key pair that you created, and then choose **Launch Instances**.

Your EC2 instance is now configured to mount the EFS file system at launch and whenever it's rebooted.

Enabling automatic mounting on existing EC2 Linux instances

The `/etc/fstab` file contains information about file systems. The command `mount -a`, which runs during instance start-up, mounts all the file systems listed in `/etc/fstab`. In this procedure, you will manually update the `/etc/fstab` on an Amazon EC2 Linux instance so that the instance uses the EFS mount helper to automatically remount an EFS file system when the instance restarts.

> **Note**
>
> EFS file systems do not support automatic mounting using `/etc/fstab` with the EFS mount helper on EC2 Mac instances running macOS Big Sur or Monterey. Instead, you can use NFS with `/etc/fstab` to automatically mount your file system on EC2 Mac instances running macOS Big Sur and Monterey.

This method uses the EFS mount helper to mount the file system. The mount helper is part of the `amazon-efs-utils` set of tools.

The `amazon-efs-utils` tools are available for installation on Amazon Linux and Amazon Linux 2 Amazon Machine Images (AMIs). For more information about `amazon-efs-utils`, see Installing the Amazon EFS client. If you are using another Linux distribution, such as Red Hat Enterprise Linux (RHEL), manually build and install `amazon-efs-utils`. For more information, see Installing the Amazon EFS client on other Linux distributions.

Prerequisites

The following requirements need to be in place before you can successfully implement this procedure:

- You have already created the EFS file system that you want to be automatically remounted. For more information, see Quick create using the console.
- You have already created the EC2 Linux instance that you want to configure to automatically remount an EFS file system.
- The EFS mount helper is installed on the EC2 Linux instance. For more information, see Installing the Amazon EFS client.

Update the /etc/fstab file

Perform the following steps to update the /etc/fstab on an EC2 Linux instance so that the instance uses the EFS mount helper to automatically remount an EFS file system when the instance restarts.

To update the /etc/fstab file on your EC2 instance

1. Connect to your EC2 instance. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Open the `/etc/fstab` file in an editor.
3. For automatic mounting using either IAM authorization or an EFS access point:

    - To automatically mount with IAM authorization to an EC2 instance that has an instance profile, add the following line to the `/etc/fstab` file.

    ```
    file-system-id:/ efs-mount-point efs _netdev,noresvport,tls,iam 0 0
    ```

    - To automatically mount with IAM authorization to a Linux instance using a credentials file, add the following line to the `/etc/fstab` file.

```
file-system-id:/ efs-mount-point efs
 _netdev,noresvport,tls,iam,awsprofile=namedprofile 0 0
```

- To automatically mount a file system using an EFS access point, add the following line to the `/etc/fstab` file.

```
file-system-id:/ efs-mount-point efs _netdev,noresvport,tls,accesspoint=access-
point-id 0 0
```

> **Warning**
>
> Use the `_netdev` option, used to identify network file systems, when mounting your file system automatically. If `_netdev` is missing, your EC2 instance might stop responding. This result is because network file systems need to be initialized after the compute instance starts its networking. For more information, see Automatic mounting fails and the instance is unresponsive.

For more information, see Mounting with IAM authorization and Mounting with EFS access points.

4. Save the changes to the file.

> **Note**
>
> In some cases, your EC2 instance might need to start regardless of the status of your mounted EFS file system. In such cases, add the `nofail` option to your file system's entry in your `/etc/fstab` file.

The line of code you added to the `/etc/fstab` file does the following.

Field	Description
file-system-id :/	The ID for your EFS file system. You can get this ID from the console or programmatically from the CLI or an AWS SDK.

Field	Description
efs-mount-point	The mount point for the EFS file system on your EC2 instance.
efs	The type of file system. When you're using the mount helper, this type is always efs.
mount options	Mount options for the file system. This is a comma-separated list of the following options: • _netdev – This option tells the operating system that the file system resides on a device that requires network access. This option prevents the instance from mounting the file system until the network has been enabled on the client. • noresvport – Tells the NFS client to use a new Transmission Control Protocol (TCP) source port when a network connection is reestablished. Doing this helps make sure that the EFS file system has uninterrupted availability after a network recovery event. • tls – Enables encryption of data in transit. • iam – Use this option to mount with IAM authorization to an EC2 instance that has an instance profile. Using the iam mount option requires also using the tls option. For more information, see Using IAM to control access to file systems. • awsprofile= *namedprofile* – Use this option with the iam and tls options to mount with IAM authorization to a Linux instance using a credentials file. For more information about EFS access points, see Using IAM to control access to file systems. • accesspoint= *access-point-id* – Use this option with the tls option to mount using an EFS access point. For more informati on about EFS access points, see Working with access points.
0	A nonzero value indicates that the file system should be backed up by dump. For EFS, this value should be 0.

Field	Description
`0`	The order in which `fsck` checks file systems at boot. For EFS file systems, this value should be `0` to indicate that `fsck` should not run at start-up.

Enabling automatic mounting on EC2 Linux or Mac instances using NFS

Using NFS without the EFS mount helper to update the Amazon EC2 `/etc/fstab` file, for EC2 Linux and Mac instances.

To update the `/etc/fstab` file on your EC2 instance

1. Connect to your EC2 instance. For more information, see Connect to your EC2 instance in the *Amazon EC2 User Guide*.
2. Open the `/etc/fstab` file in an editor.
3. To automatically mount a file system using NFS instead of the EFS mount helper, add the following line to the `/etc/fstab` file.
   - Replace *`file_system_id`* with the ID of the file system you are mounting.
   - Replace *`aws-region`* with the AWS Region that the file system in, such as `us-east-1`.
   - Replace *`mount_point`* with the file system's mount point.

```
file_system_id.efs.aws-region.amazonaws.com:/ mount_point nfs4
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport,_netdev
 0 0
```

The line of code you added to the `/etc/fstab` file does the following.

Field	Description
`file-system-id` `:/`	The ID for your EFS file system. You can get this ID from the console or programmatically from the CLI or an AWS SDK.
`efs-mount-point`	The mount point for the EFS file system on your EC2 instance.

Field	Description
`nfs4`	Specifies the file system type.

Field	Description
`mount options`	The comma-separated list of mount options for the file system: • `nfsvers=4.1` – specifies using NFS v4.1. • `rsize=1048576` – For improved performance, sets the maximum number of bytes of data that the NFS client can receive for each network READ request when reading data from a file on an EFS file system. `1048576` is the largest size possible. • `wsize=1048576` – For improved performance, sets the maximum number of bytes of data that the NFS client can send for each network WRITE request when writing data to a file on an EFS file system. `1048576` is the largest size possible. • `hard` – Sets the recovery behavior of the NFS client after an NFS request times out, so that NFS requests are retried indefinitely until the server replies. We recommend that you use the hard mount option (`hard`) to ensure data integrity. If you use a `soft` mount, set the `timeo` parameter to at least `150` deciseconds (15 seconds). Doing so helps minimize the risk of data corruption that is inherent with soft mounts. • `timeo=600` – Sets the timeout value that the NFS client uses to wait for a response before it retries a request to 600 decisecon ds (60 seconds). If you must change the timeout parameter (`timeo`), we recommend that you use a value of at least `150`, which is equivalent to 15 seconds. Doing so helps avoid diminished performance. • `retrans=2` – Sets to 2 the number of times the NFS client retries a request before it attempts further recovery action. • `noresvport` – Tells the NFS client to use a new Transmission Control Protocol (TCP) source port when a network connection is reestablished. Doing this helps make sure that the EFS file system has uninterrupted availability after a network recovery event. • `_netdev` – Prevents the client from attempting to mount the EFS file system until the network has been enabled.

Field	Description
`0`	Specifies the dump value; `0` tells the dump utility to not back up the file system.
`0`	Tells the `fsck` utility to not run at start-up.

Unmounting file systems

Before you delete a file system, we recommend that you unmount it from every Amazon EC2 instance that it's connected to. You can unmount a file system on your Amazon EC2 instance by running the `umount` command on the instance itself. You can't unmount an EFS file system through the AWS CLI, the AWS Management Console, or through any of the AWS SDKs. To unmount an EFS file system connected to an EC2 instance running Linux, use the `umount` command as follows:

```
umount /mnt/efs
```

We recommend that you do not specify any other `umount` options. Avoid setting any other `umount` options that are different from the defaults.

You can verify that your EFS file system has been unmounted by running the `df` command. This command displays the disk usage statistics for the file systems currently mounted on your Linux-based Amazon EC2 instance. If the EFS file system that you want to unmount isn't listed in the `df` command output, this means that the file system is unmounted.

Example – Identify the mount status of an EFS file system and unmount it

```
$ df -T
Filesystem Type 1K-blocks Used Available Use% Mounted on
/dev/sda1 ext4 8123812 1138920 6884644 15% /
availability-zone.file-system-id.efs.aws-region.amazonaws.com :/ nfs4 9007199254740992
 0 9007199254740992 0% /mnt/efs
```

```
$ umount /mnt/efs
```

```
$ df -T
```

```
Filesystem Type 1K-blocks Used Available Use% Mounted on
/dev/sda1 ext4 8123812 1138920 6884644 15% /
```

Tutorial: Create an EFS file system and mount it on an EC2 instance using the AWS CLI

Create an encrypted EFS file system, mount it on an EC2 instance in your VPC, and test the setup using the AWS CLI.

> **Note**
>
> In the Getting started tutorial, you use the console to create Amazon EC2 and EFS resources. In this tutorial, you use the AWS CLI to do the same—primarily to familiarize yourself with the Amazon EFS API.

In this tutorial, you create the following AWS resources in your account:

- Amazon EC2 resources:
  - Two security groups (for your EC2 instance and EFS file system).

    You add rules to these security groups to authorize appropriate inbound/outbound access. Doing this allows your EC2instance to connect to the file system through the mount target by using a standard NFSv4.1 TCP port.
  - An EC2 instance in your VPC.
- Amazon EFS resources:
  - A file system.
  - A mount target for your file system.

    To mount your file system on an EC2 instance you need to create a mount target in your VPC. You can create one mount target in each of the Availability Zones in your VPC. For more information, see How Amazon EFS works.

Then, you test the file system on your EC2 instance. The cleanup step at the end of the tutorial provides information for you to remove these resources.

The tutorial creates all these resources in the US West (Oregon) Region (`us-west-2`). Whichever AWS Region you use, be sure to use it consistently. All of your resources—your VPC, EC2 resources, and EFS resources—must be in the same AWS Region.

Topics

- Prerequisites
- Setting up the AWS CLI
- Step 1: Create EC2 resources
- Step 2: Create EFS resources
- Step 3: Mount the file system on the EC2 instance and test
- Step 4: Clean up

Prerequisites

- You can use the root credentials of your AWS account to sign in to the console and try the getting started exercise. However, AWS Identity and Access Management (IAM) recommends that you do not use the root credentials of your AWS account. Instead, create an administrator user in your account and use those credentials to manage resources in your account. Instead, create an administrator user in your account and use those credentials to manage resources in your account. For more information, see Assign AWS account access for an IAM Identity Center user in the *AWS IAM Identity Center User Guide*.
- You can use a default VPC or a custom VPC that you have created in your account. For this walkthrough, the default VPC configuration works. However, if you use a custom VPC, verify the following:
  - DNS hostnames are enabled. For more information, see DNS attributes in your VPC in the *Amazon VPC User Guide*.
  - The internet gateway is attached to your VPC. For more information, see Connect to the internet using an internet gateway in the *Amazon VPC User Guide*.
  - The VPC subnets are configured to request public IP addresses for instances launched in the VPC subnets. For more information, see IP addressing for your VPCs and subnets in the *Amazon VPC User Guide*.
  - The VPC route table includes a rule to send all Internet-bound traffic to the Internet gateway.
- You need to set up the AWS CLI and add the adminuser profile.

Setting up the AWS CLI

Use the following instructions to set up the AWS CLI and user profile.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see [Get started with the AWS CLI] in the *AWS Command Line Interface User Guide*.
2. Set profiles.

   You store user credentials in the AWS CLI `config` file. The example CLI commands in this tutorial specify the adminuser profile. Create the adminuser profile in the `config` file. You can also set the administrator user profile as the default in the `config` file as shown.

   ```
   [profile adminuser]
   aws_access_key_id = admin user access key ID
   aws_secret_access_key = admin user secret access key
   region = us-west-2

   [default]
   aws_access_key_id = admin user access key ID
   aws_secret_access_key = admin user secret access key
   region = us-west-2
   ```

   The preceding profile also sets the default AWS Region. If you don't specify a Region in the CLI command, the us-west-2 region is assumed.
3. Verify the setup by entering the following command at the command prompt. Both of these commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try the help command

     You can also specify the user profile explicitly by adding the `--profile` parameter.

     ```
     aws help
     ```

     ```
     aws help \
     --profile adminuser
     ```

Step 1: Create EC2 resources

In this step, you do the following:

- Create two security groups.
- Add rules to the security groups to authorize additional access.
- Launch an EC2 instance. You create and mount an EFS file system on this instance in the next step.

Step 1.1: Create two security groups

In this section, you create security groups in your VPC for your EC2 instance and EFS mount target. Later in the tutorial, you assign these security groups to an EC2 instance and an EFS mount target. For information about security groups, see Amazon EC2 security groups for Linux instances .

To create security groups

1. Create two security groups using the `create-security-group` CLI command:

   a. Create a security group (`efs-walkthrough1-ec2-sg`) for your EC2 instance, and provide your VPC ID.

   ```
   $ aws ec2 create-security-group \
   --region us-west-2 \
   --group-name efs-walkthrough1-ec2-sg \
   --description "Amazon EFS walkthrough 1, SG for EC2 instance" \
   --vpc-id vpc-id-in-us-west-2 \
   --profile adminuser
   ```

   Write down the security group ID. The following is an example response.

   ```
   {
       "GroupId": "sg-aexample"
   }
   ```

   You can find the VPC ID using the following command.

   ```
   $ aws  ec2 describe-vpcs
   ```

b. Create a security group (`efs-walkthrough1-mt-sg`) for your EFS mount target. You need to provide your VPC ID.

```
$ aws ec2 create-security-group \
--region us-west-2 \
--group-name efs-walkthrough1-mt-sg \
--description "Amazon EFS walkthrough 1, SG for mount target" \
--vpc-id vpc-id-in-us-west-2 \
--profile adminuser
```

Write down the security group ID. The following is an example response.

```
{
    "GroupId": "sg-aexample"
}
```

2. Verify the security groups.

```
aws ec2 describe-security-groups \
--group-ids list of security group IDs separated by space \
--profile adminuser \
--region us-west-2
```

Both should have only one outbound rule that allows all traffic to leave.

In the next section, you authorize additional access that enables the following:

- Enable you to connect to your EC2 instance.
- Enable traffic between an EC2 instance and an EFS mount target (with which you associate these security groups later in this tutorial).

Step 1.2: Add rules to the security groups to authorize inbound/outbound access

In this step, you add rules to the security groups to authorize inbound/outbound access.

To add rules

1. Authorize incoming Secure Shell (SSH) connections to the security group for your EC2 instance (`efs-walkthrough1-ec2-sg`) so you can connect to your EC2 instance using SSH from any host.

```
$ aws ec2 authorize-security-group-ingress \
--group-id id of the security group created for EC2 instance \
--protocol tcp \
--port 22 \
--cidr 0.0.0.0/0 \
--profile adminuser \
--region us-west-2
```

Verify that the security group has the inbound and outbound rule you added.

```
aws ec2 describe-security-groups \
--region us-west-2 \
--profile adminuser \
--group-id security-group-id
```

2. Authorize inbound access to the security group for the EFS mount target (`efs-walkthrough1-mt-sg`).

   At the command prompt, run the following AWS CLI `authorize-security-group-ingress` command using the adminuser profile to add the inbound rule.

```
$ aws ec2 authorize-security-group-ingress \
--group-id ID of the security group created for Amazon EFS mount target \
--protocol tcp \
--port 2049 \
--source-group ID of the security group created for EC2 instance \
--profile adminuser \
--region us-west-2
```

3. Verify that both security groups now authorize inbound access.

```
aws ec2 describe-security-groups \
--group-names efs-walkthrough1-ec2-sg   efs-walkthrough1-mt-sg \
--profile adminuser \
--region us-west-2
```

Step 1.3: Launch an EC2 instance

In this step, you launch an EC2 instance.

To launch an EC2 instance

1. Gather the following information that you need to provide when launching an EC2 instance:

   - Key pair name. For instructions to create a key pair, see [Create a key pair for your Amazon EC2 instance]() in the *Amazon EC2 User Guide*.
   - The ID of the Amazon Machine Image (AMI) you want to launch.

     The AWS CLI command that you use to launch an EC2 instance requires the ID of the Amazon Machine Image (AMI) that you want to deploy as a parameter. The exercise uses the Amazon Linux HVM AMI.

     > **Note**
     >
     > You can use most general purpose Linux-based AMIs. If you use another Linux AMI, make sure that you use your distribution's package manager to install the NFS client on the instance. Also, you might need to add software packages as you need them.

     For the Amazon Linux HVM AMI, you can find the latest IDs at [Amazon Linux AMI](). You choose the ID value from the Amazon Linux AMI IDs table as follows:

     - Choose the **US West Oregon** region. This walkthrough assumes you are creating all resources in the US West (Oregon) Region (us-west-2).
     - Choose the **EBS-backed HVM 64-bit** type (because in the CLI command you specify the `t2.micro` instance type, which does not support instance store).

   - ID of the security group you created for an EC2 instance.
   - AWS Region. This walkthrough uses the us-west-2 region.
   - Your VPC subnet ID where you want to launch the instance. You can get list of subnets using the `describe-subnets` command.

     ```
     $ aws ec2 describe-subnets \
     --region us-west-2 \
     --filters "Name=vpc-id,Values=vpc-id" \
     --profile adminuser
     ```

After you choose the subnet ID, write down the following values from the `describe-subnets` result:

- **Subnet ID** – You need this value when you create a mount target. In this exercise, you create a mount target in the same subnet where you launch an EC2 instance.
- **Availability Zone of the subnet** – You need this value to construct your mount target DNS name, which you use to mount a file system on the EC2 instance.

2. Run the following AWS CLI `run-instances` command to launch an EC2 instance.

```
$ aws ec2 run-instances \
--image-id AMI ID \
--count 1 \
--instance-type t2.micro \
--associate-public-ip-address \
--key-name key-pair-name \
--security-group-ids ID of the security group created for EC2 instance \
--subnet-id VPC subnet ID \
--region us-west-2 \
--profile adminuser
```

3. Write down the instance ID returned by the `run-instances` command.

4. The EC2 instance you created must have a public DNS name that you use to connect to the EC2 instance and mount the file system on it. The public DNS name is of the form:

```
ec2-xx-xx-xx-xxx.compute-1.amazonaws.com
```

Run the following CLI command and write down the public DNS name.

```
aws ec2 describe-instances \
--instance-ids EC2 instance ID \
--region us-west-2 \
--profile adminuser
```

If you don't find the public DNS name, check the configuration of the VPC in which you launched the EC2 instance. For more information, see Prerequisites.

5. (Optional) Assign a name to the EC2 instance that you created. To do so, add a tag with the key name and value set to the name that you want to assign to the instance. You do this by running the following AWS CLI `create-tags` command.

```
$  aws ec2 create-tags \
--resources  EC2-instance-ID \
--tags Key=Name,Value=Provide-instance-name  \
--region us-west-2 \
--profile adminuser
```

Step 2: Create EFS resources

In this step, you do the following:

- Create an encrypted EFS file system.
- Enable lifecycle management.
- Create a mount target in the Availability Zone where you have your EFS instance launched.

Step 2.1: Create an EFS file system

In this step, you create an EFS file system. Write down the `FileSystemId` to use later when you create mount targets for the file system in the next step.

To create a file system

- Create a file system with the optional `Name` tag.

  a. At the command prompt, run the following AWS CLI `create-file-system` command.

```
$  aws efs create-file-system \
--encrypted \
--creation-token FileSystemForWalkthrough1 \
--tags Key=Name,Value=SomeExampleNameValue \
--region us-west-2 \
--profile adminuser
```

  You get the following response.

```
{
    "OwnerId": "111122223333",
    "CreationToken": "FileSystemForWalkthrough1",
    "FileSystemId": "fs-c657c8bf",
```

```
    "CreationTime": 1548950706.0,
    "LifeCycleState": "creating",
    "NumberOfMountTargets": 0,
    "SizeInBytes": {
        "Value": 0,
        "ValueInIA": 0,
        "ValueInStandard": 0
    },
    "PerformanceMode": "generalPurpose",
    "Encrypted": true,
    "KmsKeyId": "arn:aws:kms:us-west-2:111122223333:a5c11222-7a99-43c8-9dcc-
abcdef123456",
    "ThroughputMode": "bursting",
    "Tags": [
        {
            "Key": "Name",
            "Value": "SomeExampleNameValue"
        }
    ]
}
```

b. Note the `FileSystemId` value. You need this value when you create a mount target for this file system in Step 2.3: Create a mount target.

Step 2.2: Enable lifecycle management

In this step, you enable lifecycle management on your file system in order to use the EFS Infrequent Access (IA) storage class. To learn more, see Managing storage lifecycle and EFS storage classes.

To enable lifecycle management

- At the command prompt, run the following AWS CLI `put-lifecycle-configuration` command.

```
$  aws efs put-lifecycle-configuration \
--file-system-id fs-c657c8bf \
--lifecycle-policies TransitionToIA=AFTER_30_DAYS \
--region us-west-2 \
--profile adminuser
```

You get the following response.

```
{
  "LifecyclePolicies": [
    {
        "TransitionToIA": "AFTER_30_DAYS"
    }
  ]
}
```

Step 2.3: Create a mount target

In this step, you create a mount target for your file system in the Availability Zone where you have your EC2 instance launched.

1. Make sure you have the following information:

   - ID of the file system (for example, `fs-example`) for which you are creating the mount target.
   - VPC subnet ID where you launched the EC2 instance in Step 1: Create EC2 resources.

     For this tutorial, you create the mount target in the same subnet in which you launched the EC2 instance, so you need the subnet ID (for example, `subnet-example`).
   - ID of the security group you created for the mount target in the preceding step.

2. At the command prompt, run the following AWS CLI `create-mount-target` command.

```
$ aws efs create-mount-target \
--file-system-id file-system-id \
--subnet-id  subnet-id \
--security-group ID-of-the security-group-created-for-mount-target \
--region us-west-2 \
--profile adminuser
```

   You get the following response.

```
{
    "MountTargetId": "fsmt-example",
```

```
    "NetworkInterfaceId": "eni-example",
    "FileSystemId": "fs-example",
    "PerformanceMode" : "generalPurpose",
    "LifeCycleState": "available",
    "SubnetId": "fs-subnet-example",
    "OwnerId": "account-id",
    "IpAddress": "xxx.xx.xx.xxx"
}
```

3. You can also use the `describe-mount-targets` command to get descriptions of mount targets you created on a file system.

```
$ aws efs describe-mount-targets \
--file-system-id file-system-id \
--region us-west-2 \
--profile adminuser
```

Step 3: Mount the file system on the EC2 instance and test

In this step, you do the following:

- Gather required information.
- Install the NFS client on your EC2 instance.
- Mount the file system on your EC2 instance and test.

Topics

- Step 3.1: Gather Information
- Step 3.2: Install the NFS client on your EC2 instance
- Step 3.3: Mount the file system on your EC2 instance and test

Step 3.1: Gather Information

Make sure you have the following information as you follow the steps in this section:

- Public DNS name of your EC2 instance in the following format:

```
ec2-xx-xxx-xxx-xx.aws-region.compute.amazonaws.com
```

- DNS name of your file system. You can construct this DNS name using the following generic form:

```
file-system-id.efs.aws-region.amazonaws.com
```

  The EC2 instance on which you mount the file system by using the mount target can resolve the file system's DNS name to the mount target's IP address.

> **Note**
>
> Amazon EFS doesn't require that your EC2 instance have either a public IP address or public DNS name. The requirements listed preceding are just for this walkthrough example to ensure that you can connect by using SSH into the instance from outside the VPC.

Step 3.2: Install the NFS client on your EC2 instance

You can connect to your EC2 instance from Windows or from a computer running Linux, or macOS X, or any other Unix variant.

To install an NFS client

1. Connect to your EC2 instance. For more information, see [Connect to your EC2 instance](#) in the *Amazon EC2 User Guide*.
2. Execute the following commands on the EC2 instance by using the SSH session:

   a. (Optional) Get updates and reboot.

   ```
   $  sudo yum -y update
   $  sudo reboot
   ```

   After the reboot, reconnect to your EC2 instance.

   b. Install the NFS client.

   ```
   $ sudo yum -y install nfs-utils
   ```

> **Note**
>
> If you choose the **Amazon Linux AMI 2016.03.0** Amazon Linux AMI when launching your EC2 instance, you don't need to install `nfs-utils` because it is already included in the AMI by default.

Step 3.3: Mount the file system on your EC2 instance and test

Now you mount the file system on your EC2 instance.

1. Make a directory ("efs-mount-point").

```
$ mkdir ~/efs-mount-point
```

2. Mount the EFS file system.

```
$ sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-
target-DNS:/   ~/efs-mount-point
```

The EC2 instance can resolve the mount target DNS name to the IP address. You can optionally specify the IP address of the mount target directly.

```
$ sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-
target-ip:/  ~/efs-mount-point
```

3. Now that you have the EFS file system mounted on your EC2 instance, you can create files.

   a. Change the directory.

   ```
   $ cd ~/efs-mount-point
   ```

   b. List the directory contents.

   ```
   $ ls -al
   ```

   It should be empty.

```
drwxr-xr-x 2 root     root     4096 Dec 29 22:33 .
drwx------ 4 ec2-user ec2-user 4096 Dec 29 22:54 ..
```

c. The root directory of a file system, upon creation, is owned by and is writable by the root user, so you need to change permissions to add files.

```
$ sudo chmod go+rw .
```

Now, if you try the `ls -al` command you see that the permissions have changed.

```
drwxrwxrwx 2 root     root     4096 Dec 29 22:33 .
drwx------ 4 ec2-user ec2-user 4096 Dec 29 22:54 ..
```

d. Create a text file.

```
$ touch test-file.txt
```

e. List directory content.

```
$ ls -l
```

You now have successfully created and mounted an EFS file system on your EC2 instance in your VPC.

The file system you mounted doesn't persist across reboots. To automatically remount the directory, you can use the `fstab` file. If you are using an Auto Scaling group to launch EC2 instances, you can also set scripts in a launch configuration.

Step 4: Clean up

If you no longer need the resources you created, you should remove them. You can do this with the CLI.

- Remove EC2 resources (the EC2 instance and the two security groups). Amazon EFS deletes the network interface when you delete the mount target.
- Remove EFS resources (file system, mount target).

To delete AWS resources created in this walkthrough

1. Terminate the EC2 instance you created for this tutorial.

```
$ aws ec2 terminate-instances \
--instance-ids instance-id \
--profile adminuser
```

   You can also delete EC2 resources using the console. For instructions, see Terminate an instance.

2. Delete the mount target.

   You must delete the mount targets created for the file system before deleting the file system. You can get a list of mount targets by using the `describe-mount-targets` CLI command.

```
$  aws efs describe-mount-targets \
--file-system-id file-system-ID \
--profile adminuser \
--region aws-region
```

   Then delete the mount target by using the `delete-mount-target` CLI command.

```
$ aws efs delete-mount-target \
--mount-target-id ID-of-mount-target-to-delete \
--profile adminuser \
--region aws-region
```

3. (Optional) Delete the two security groups you created. You don't pay for creating security groups.

   You must delete the mount target's security group first, before deleting the EC2 instance's security group. The mount target's security group has a rule that references the EC2 security group. Therefore, you cannot first delete the EC2 instance's security group.

   For instructions, see Delete your security group in the *Amazon EC2 User Guide*.

4. Delete the file system by using the `delete-file-system` CLI command. You can get a list of your file systems by using the `describe-file-systems` CLI command. You can get the file system ID from the response.

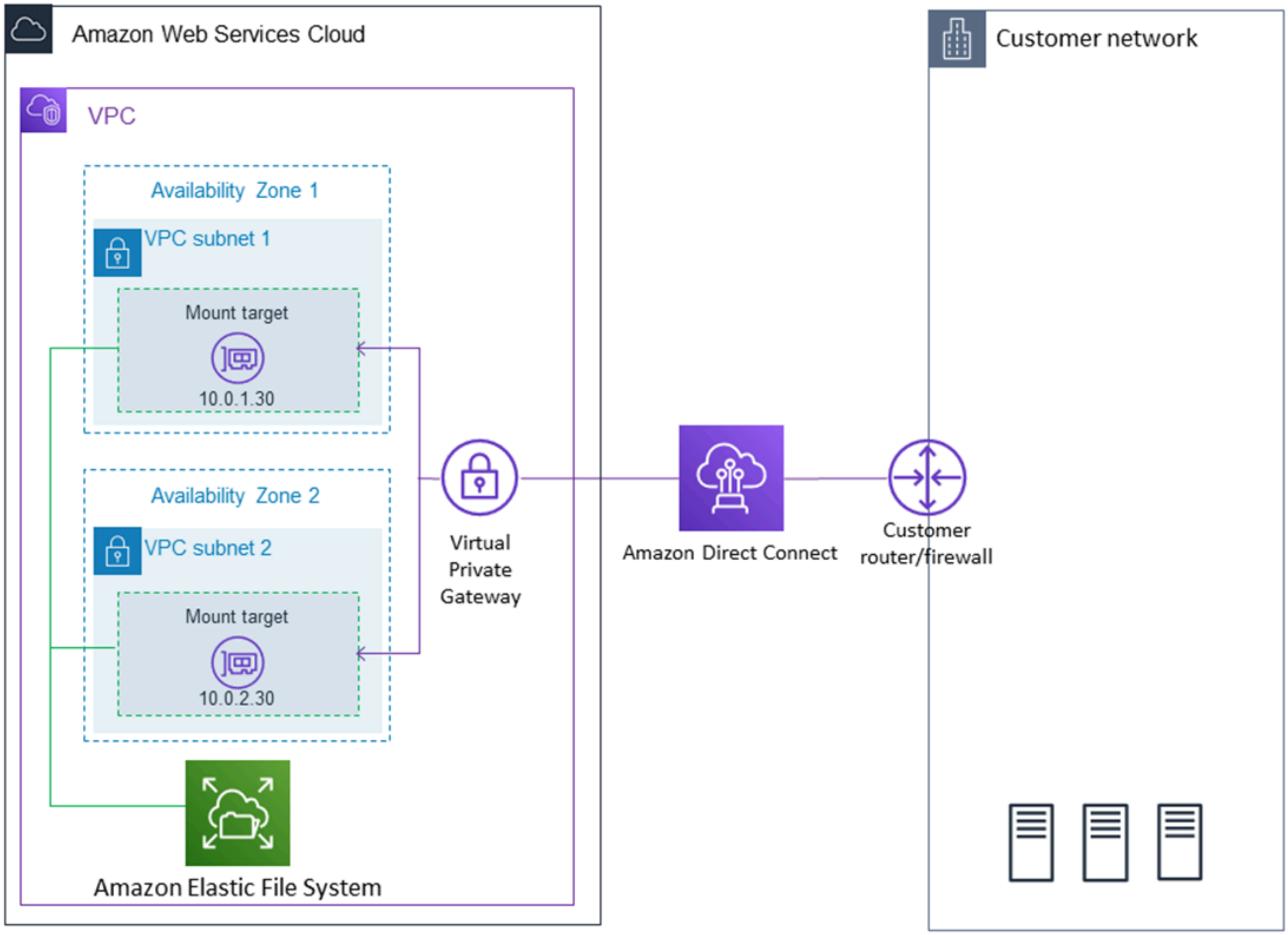
```
aws efs describe-file-systems \
--profile adminuser \
--region aws-region
```

Delete the file system by providing the file system ID.

```
$ aws efs delete-file-system \
--file-system-id ID-of-file-system-to-delete \
--region aws-region \
--profile adminuser
```

Tutorial: Mounting with on-premises Linux clients

You can mount your Amazon EFS file systems on your on-premises data center servers when connected to your Amazon VPC with AWS Direct Connect or VPN. The following graphic shows a high-level schematic diagram of the AWS services required in mounting Amazon EFS file systems from on-premises.



> **Note**
>
> Using Amazon EFS with Microsoft Windows–based clients isn't supported.

Topics

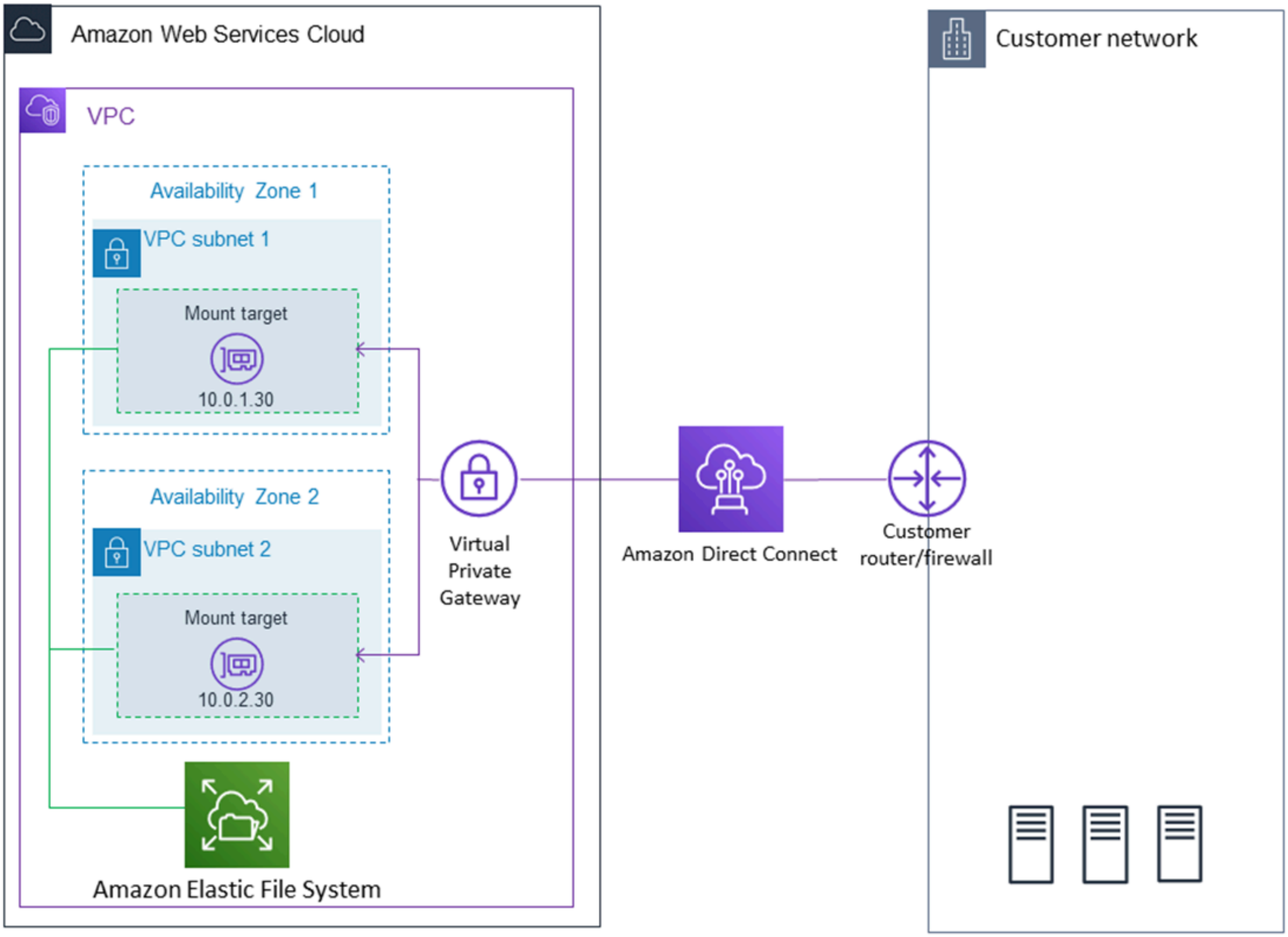
- Prerequisites
- Step 1: Create your EFS resources
- Step 2: Install the NFS client
- Step 3: Mount the Amazon EFS file system on your on-premises Client
- Step 4: Clean up resources and protect your AWS account
- Optional: Encrypting data in transit

Prerequisites

Make sure that you already have an AWS Direct Connect or VPN connection. For more information on AWS Direct Connect, see the AWS Direct Connect User Guide. For more information on setting up a VPN connection, see Connect your VPC to remote networks using AWS Virtual Private Network in the *Amazon VPC User Guide*.

After you have an AWS Direct Connect or VPN connection, create an EFS file system and a mount target in your Amazon VPC. After that, you download and install the amazon-efs-utils tools. Then, you test the file system from your on-premises client. Finally, the clean-up step at the end of the walkthrough provides information for you to remove these resources.

The walkthrough creates all these resources in the US West (Oregon) Region (`us-west-2`). Whichever AWS Region you use, be sure to use it consistently. All of your resources—your VPC, your mount target, and your Amazon EFS file system—must be in the same AWS Region, as shown in the following diagram.



> **Note**
>
> In some cases, your local application might need to know if the EFS file system is available. In these cases, your application should be able to point to a different mount point IP address if the first mount point becomes temporarily unavailable. In this scenario, we recommend that you have two on-premises clients connected to your file system through different Availability Zones (AZs) for higher availability.

You can use the root credentials of your AWS account to sign in to the console and try this exercise. However, AWS Identity and Access Management (IAM) best practices recommend that you don't use the root credentials of your AWS account. Instead, create an administrator user in your account and use those credentials to manage resources in your account. For more information, see [Single sign-on access to AWS accounts] in the *AWS IAM Identity Center User Guide*.

You can use a default VPC or a custom VPC that you have created in your account. For this walkthrough, the default VPC configuration works. However, if you use a custom VPC, verify the following:

- The internet gateway is attached to your VPC. For more information, see Enable internet access for a VPC using an internet gateway in the *Amazon VPC User Guide*.
- The VPC route table includes a rule to send all internet-bound traffic to the Internet gateway.

Step 1: Create your EFS resources

In this step, you create your EFS file system and mount targets.

To create your EFS file system

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. Choose **Create File System**.
3. Choose your default VPC from the **VPC** list.
4. Select the checkboxes for all of the Availability Zones. Make sure that they all have the default subnets, automatic IP addresses, and the default security groups chosen. These are your mount targets. For more information, see Managing mount targets.
5. Choose **Next Step**.
6. Name your file system, keep **general purpose** selected as your default performance mode, and choose **Next Step**.
7. Choose **Create File System**.
8. Choose your file system from the list and make a note of the **Security group** value. You need this value for the next step.

The file system you just created has mount targets. Each mount target has an associated security group. The security group acts as a virtual firewall that controls network traffic. If you didn't provide a security group when creating a mount target, Amazon EFS associates the default security group of the VPC with it. If you followed the preceding steps exactly, then your mount targets are using the default security group.

Next, you add a rule to the mount target's security group to allow inbound traffic to the Network File System (NFS) port (2049). You can use the AWS Management Console to add the rule to your mount target's security groups in your VPC.

To allow inbound traffic to the NFS port

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Under **NETWORK & SECURITY**, choose **Security Groups**.
3. Choose the security group associated with your file system. You made a note of this at the end of Step 1: Create your EFS resources.
4. In the tabbed pane that appears below the list of security groups, choose the **Inbound** tab.
5. Choose **Edit**.
6. Choose **Add Rule**, and choose a rule of the following type:

   - **Type – NFS**
   - **Source – Anywhere**

   We recommend that you only use the **Anywhere** source for testing. You can create a custom source set to the IP address of the on-premises client, or use the console from the client itself, and choose **My IP**.

   Note

   You don't need to add an outbound rule, because the default outbound rule allows all traffic to leave. If you don't have this default outbound rule, add an outbound rule to open a TCP connection on the NFS port, identifying the mount target security group as the destination.

Step 2: Install the NFS client

In this step, you install the NFS client.

To install the NFS client on your on-premises server

> **Note**
>
> If you require data to be encrypted in transit, use the Amazon EFS mount helper, `amazon-efs-utils`, instead of the NFS client. For information on installing amazon-efs-utils, see the section *Optional: Encrypting Data in Transit*.

1. Access the terminal for your on-premises client.
2. Install NFS.

   If you're using Red Hat Linux, install NFS with the following command.

   ```
   $ sudo yum -y install nfs-utils
   ```

   If you're using Ubuntu, install NFS with the following command.

   ```
   $ sudo apt-get -y install nfs-common
   ```

Step 3: Mount the Amazon EFS file system on your on-premises Client

To create a mount directory

1. Make a directory for the mount point with the following command.

   Example

   ```
   mkdir ~/efs
   ```

2. Choose your preferred IP address of the mount target in the Availability Zone. You can measure the latency from your on-premises Linux clients. To do so, use a terminal-based tool like `ping` against the IP address of your EC2 instances in different Availability Zones to find the one with the lowest latency.

- Run the mount command to mount the file system using the IP address of the mount target.

```
$ sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-
target-IP:/   ~/efs
```

Now that you've mounted your Amazon EFS file system, you can test it out with the following procedure.

To test the Amazon EFS file system connection

1. Change directories to the new directory that you created with the following command.

```
$ cd ~/efs
```

2. Make a subdirectory and change the ownership of that subdirectory to your EC2 instance user. Then, navigate to that new directory with the following commands.

```
$ sudo mkdir getting-started
$ sudo chown ec2-user getting-started
$ cd getting-started
```

3. Create a text file with the following command.

```
$ touch test-file.txt
```

4. List the directory contents with the following command.

```
$ ls -al
```

As a result, the following file is created.

```
-rw-rw-r-- 1 username username 0 Nov 15 15:32 test-file.txt
```

Warning

Use the `_netdev` option, used to identify network file systems, when mounting your file system automatically. If `_netdev` is missing, your EC2 instance might stop responding. This result is because network file systems need to be initialized after the compute instance starts its networking. For more information, see Automatic mounting fails and the instance is unresponsive.

Step 4: Clean up resources and protect your AWS account

After you have finished this walkthrough, or if you don't want to explore the walkthroughs, you should follow these steps to clean up your resources and protect your AWS account.

To clean up resources and protect your AWS account

1. Unmount the Amazon EFS file system with the following command.

```
$ sudo umount ~/efs
```

2. Open the Amazon EFS console at Amazon EFS Console.
3. Choose the Amazon EFS file system that you want to delete from the list of file systems.
4. For **Actions**, choose **Delete file system**.
5. In the **Permanently delete file system** dialog box, type the file system ID for the Amazon EFS file system that you want to delete, and then choose **Delete File System**.
6. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
7. In the navigation pane, choose **Security Groups**.
8. Select the name of the security group that you added the rule to for this walkthrough.

> **Warning**
>
> Don't delete the default security group for your VPC.

9. For **Actions**, choose **Edit inbound rules**.
10. Choose the X at the end of the inbound rule you added, and choose **Save**.

Optional: Encrypting data in transit

To encrypt data in transit, use the Amazon EFS mount helper, amazon-efs-utils, instead of the NFS client.

The *amazon-efs-utils* package is an open-source collection of Amazon EFS tools. The amazon-efs-utils collection comes with a mount helper and tooling that makes it easier to encrypt data in transit for Amazon EFS. For more information on this package, see Installing the Amazon EFS client. This package is available as a free download from GitHub, which you can get by cloning the package's repository.

To clone amazon-efs-utils from GitHub

1. Access the terminal for your on-premises client.
2. From the terminal, clone the amazon-efs-utils tool from GitHub to a directory of your choice, with the following command.

```
git clone https://github.com/aws/efs-utils
```

Now that you have the package, you can install it. This installation is handled differently depending on the Linux distribution of your on-premises client. The following distributions are supported:

- Amazon Linux 2
- Amazon Linux
- Red Hat Enterprise Linux (and derivatives such as CentOS) version 7 and newer
- Ubuntu 16.04 LTS and newer

To build and install amazon-efs-utils as an RPM package

1. Open a terminal on your client and navigate to the directory that has the cloned amazon-efs-utils package from GitHub.
2. Build the package with the following command.

```
make rpm
```

> **Note**
>
> If you haven't already, install the rpm-builder package with the following command.
>
> ```
> sudo yum -y install rpm-build
> ```

3. Install the package with the following command.

```
sudo yum -y install build/amazon-efs-utils*rpm
```

To build and install amazon-efs-utils as an deb package

1. Open a terminal on your client and navigate to the directory that has the cloned amazon-efs-utils package from GitHub.
2. Build the package with the following command.

```
./build-deb.sh
```

3. Install the package with the following command.

```
sudo apt-get install build/amazon-efs-utils*deb
```

After the package is installed, configure amazon-efs-utils for use in your AWS Region with AWS Direct Connect or VPN.

To configure amazon-efs-utils for use in your AWS Region

1. Using your text editor of choice, open `/etc/amazon/efs/efs-utils.conf` for editing.
2. Find the line "`dns_name_format = {fs_id}.efs.{region}.amazonaws.com`".
3. Change *`{region}`* with the ID for your AWS Region, for example `us-west-2`.

To mount the EFS file system on your on-premises client, first open a terminal on your on-premises Linux client. To mount the system, you need the file system ID, the mount target IP address for one of your mount targets, and the file system's AWS Region. If you created multiple mount targets for your file system, then you can choose any one of these.

When you have that information, you can mount your file system in three steps:

To create a mount directory

1. Make a directory for the mount point with the following command.

Example

```
mkdir ~/efs
```

2. Choose your preferred IP address of the mount target in the Availability Zone. You can measure the latency from your on-premises Linux clients. To do so, use a terminal-based tool like `ping` against the IP address of your EC2 instances in different Availability Zones to find the one with the lowest latency.

To update `/etc/hosts`

- Add an entry to your local `/etc/hosts` file with the file system ID and the mount target IP address, in the following format.

```
mount-target-IP-Address file-system-ID.efs.region.amazonaws.com
```

Example

```
192.0.2.0 fs-12345678.efs.us-west-2.amazonaws.com
```

To make a mount directory

1. Make a directory for the mount point with the following command.

Example

```
mkdir ~/efs
```

2. Run the mount command to mount the file system.

Example

```
sudo mount -t efs fs-12345678 ~/efs
```

If you want to use encryption of data in transit, your mount command looks something like the following.

Example

```
sudo mount -t efs -o tls fs-12345678 ~/efs
```

Tutorial: Mount a file system from a different VPC

In this tutorial, you set up an EC2 instance to mount an EFS file system that is in a different virtual private cloud (VPC). You do this using the EFS mount helper. The mount helper is part of the `amazon-efs-utils` set of tools. For more information about `amazon-efs-utils`, see Installing the Amazon EFS client.

The client's VPC and your EFS file system's VPC must be connected using either a VPC peering connection or a VPC transit gateway. When you use a VPC peering connection or transit gateway to connect VPCs, EC2 instances that are in one VPC can access EFS file systems in another VPC, even if the VPCs belong to different accounts.

> **Note**
> Using Amazon EFS with Microsoft Windows–based clients isn't supported.

Topics

- Prerequisites
- Step 1: Determine the ID of the mount target's Availability Zone
- Step 2: Determine the mount target IP address
- Step 3: Add a host entry for the mount target
- Step 4: Mount your file system using the EFS mount helper
- Step 5: Clean up resources and protect your AWS account

Prerequisites

To complete this tutorial, you must have the following:

- The `amazon-efs-utils` set of tools is installed on the EC2 instance before using this procedure. For instructions on installing `amazon-efs-utils`, see [Installing the Amazon EFS client](#).
- One of the following:
  - A VPC peering connection between the VPC where the EFS file system resides and the VPC where the EC2 instance resides. A *VPC peering connection* is a networking connection between two VPCs. This type of connection enables you to route traffic between them using private Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6) addresses. You can use VPC peering to connect VPCs within the same AWS Region or between AWS Regions. For more information, see [Creating and Accepting a VPC Peering Connection](#) in the *Amazon VPC Peering Guide*.
  - A transit gateway connecting the VPC where the EFS file system resides and the VPC where the EC2 instance resides. A *transit gateway* is a network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see [Getting Started with using Amazon VPC Transit Gateways](#) in the *Amazon VPC Transit Gateways Guide*.

Step 1: Determine the ID of the mount target's Availability Zone

To ensure high availability of your file system, we recommend that you always use an EC2 mount target IP address that is in the same Availability Zone as your NFS client. If you are mounting an EFS file system that is in another account, ensure that the NFS client and EFS mount target are in the same Availability Zone ID. This requirement applies because Availability Zone names can differ between accounts.

To determine the Availability Zone ID of the EC2 instance

1. Connect to your EC2 instance. For more information, see [Connect to your EC2 instance](#) in the *Amazon EC2 User Guide*.
2. Determine the Availability Zone ID that the EC2 instance is in using the `describe-availability-zones` CLI command as follows.

```
[ec2-user@ip-10.0.0.1] $ aws ec2 describe-availability-zones --zone-name
{
    "AvailabilityZones": [
        {
            "State": "available",
            "ZoneName": "us-east-2b",
```

```
            "Messages": [],
            "ZoneId": "use2-az2",
            "RegionName": "us-east-2"
        }
    ]
}
```

The Availability Zone ID is returned in the `ZoneId` property, `use2-az2`.

Step 2: Determine the mount target IP address

Now that you know the Availability Zone ID of the EC2 instance, you can now retrieve the IP address of the mount target that is in the same Availability Zone ID.

To determine the mount target IP address in the same Availability Zone ID

- Retrieve the mount target IP address for your file system in the `use2-az2` AZ ID using the `describe-mount-targets` CLI command, as follows.

```
$ aws efs describe-mount-targets --file-system-id file_system_id
{
    "MountTargets": [
        {
            "OwnerId": "111122223333",
            "MountTargetId": "fsmt-11223344",
  =====>    "AvailabilityZoneId": "use2-az2",
            "NetworkInterfaceId": "eni-048c09a306023eeec",
            "AvailabilityZoneName": "us-east-2b",
            "FileSystemId": "fs-01234567",
            "LifeCycleState": "available",
            "SubnetId": "subnet-06eb0da37ee82a64f",
            "OwnerId": "958322738406",
  =====>    "IpAddress": "10.0.2.153"
        },
...
        {
            "OwnerId": "111122223333",
            "MountTargetId": "fsmt-667788aa",
            "AvailabilityZoneId": "use2-az3",
            "NetworkInterfaceId": "eni-0edb579d21ed39261",
            "AvailabilityZoneName": "us-east-2c",
            "FileSystemId": "fs-01234567",
```

```
            "LifeCycleState": "available",
            "SubnetId": "subnet-0ee85556822c441af",
            "OwnerId": "958322738406",
            "IpAddress": "10.0.3.107"
        }
    ]
}
```

The mount target in the `use2-az2` Availability Zone ID has an IP address of 10.0.2.153.

Step 3: Add a host entry for the mount target

You can now make an entry in the `/etc/hosts` file on the EC2 instance that maps the mount target IP address to your EFS file system's hostname.

To add a host entry for the mount target

1. Add a line for the mount target IP address to the EC2 instance's `/etc/hosts` file. The entry uses the format `mount-target-IP-Address file-system-ID.efs.region.amazonaws.com`. Use the following command to add the line to the file.

```
echo "10.0.2.153 fs-01234567.efs.us-east-2.amazonaws.com" | sudo tee -a /etc/hosts
```

2. Make sure that the VPC security groups for the EC2 instance and mount target have rules that allow access to the EFS file system, as needed. For more information, see Using VPC security groups.

Step 4: Mount your file system using the EFS mount helper

To mount your EFS file system, you first create a mount directory on the EC2 instance. Then, using the EFS mount helper, you can mount the file system with either AWS Identity and Access Management (IAM) authorization or an EFS access point. For more information, see Using IAM to control access to file systems and Working with access points.

To create a mount directory

- Create a directory for mounting the file system using the following command.

```
$ sudo mkdir /mnt/efs/
```

To mount the file system using IAM authorization

- Use the following command to mount the file system using IAM authorization.

```
$ sudo mount -t efs -o tls,iam file-system-id /mnt/efs/
```

To mount the file system using an EFS access point

- Use the following command to mount the file system using an EFS access point.

```
$ sudo mount -t efs -o tls,accesspoint=access-point-id file-system-id /mnt/efs/
```

Now that you've mounted your EFS file system, you can test it with the following procedure.

To test the EFS file system connection

1. Change directories to the new directory that you created with the following command.

```
$ cd ~/mnt/efs
```

2. Make a subdirectory and change the ownership of that subdirectory to your EC2 instance user. Then navigate to that new directory with the following commands.

```
$ sudo mkdir getting-started
$ sudo chown ec2-user getting-started
$ cd getting-started
```

3. Create a text file with the following command.

```
$ touch test-file.txt
```

4. List the directory contents with the following command.

```
$ ls -al
```

As a result, the following file is created.

```
-rw-rw-r-- 1 username username 0 Nov 15 15:32 test-file.txt
```

You can also mount your file system automatically by adding an entry to the /etc/fstab file. For more information, see Enabling automatic mounting on existing EC2 Linux instances.

> **Warning**
>
> Use the _netdev option, used to identify network file systems, when mounting your file system automatically. If _netdev is missing, your EC2 instance might stop responding. This result is because network file systems need to be initialized after the compute instance starts its networking. For more information, see Automatic mounting fails and the instance is unresponsive.

Step 5: Clean up resources and protect your AWS account

After you have finished this tutorial, perform the following steps to clean up your resources and protect your AWS account.

To clean up resources and protect your AWS account

1. Unmount the EFS file system with the following command.

   ```
   $ sudo umount ~/efs
   ```

2. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
3. Choose the EFS file system that you want to delete from the list of file systems.
4. For **Actions**, choose **Delete file system**.
5. In the **Permanently delete file system** dialog box, type the file system ID for the EFS file system that you want to delete, and then choose **Delete File System**.
6. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
7. In the navigation pane, choose **Security Groups**.
8. Select the name of the security group that you added the rule to for this tutorial.

   > **Warning**
   >
   > Don't delete the default security group for your VPC.

9. For **Actions**, choose **Edit inbound rules**.

10. Choose the X at the end of the inbound rule you added, and choose **Save**.

Troubleshooting mount issues

Following, you can find information about troubleshooting EFS file system mounting issues.

File system mount on Windows instance fails

A file system mount on an Amazon EC2 instance on Microsoft Windows fails.

Action to take

Don't use Amazon EFS with Windows EC2 instances, which isn't supported.

Access denied by server

A file system mount fails with the following message:

```
/efs mount.nfs4: access denied by server while mounting 127.0.0.1:/
```

This issue can occur if your NFS client does not have permission to mount the file system.

Action to take

If you are attempting to mount the file system using IAM, make sure you are using the `-o iam` or -o tls option in your mount command. This tells the EFS mount helper to pass your credentials to the EFS mount target. If you still don't have access, check your file system policy and your identity policy to ensure there are no DENY clauses that apply to your connection, and that there is at least one ALLOW clause that applies to the connection. For more information, see Using IAM to control access to file systems and Creating file system policies.

Automatic mounting fails and the instance is unresponsive

This issue can occur if the file system was mounted automatically on an instance and the `_netdev` option wasn't declared. If `_netdev` is missing, your EC2 instance might stop responding. This result is because network file systems need to be initialized after the compute instance starts its networking.

Action to take

If this issue occurs, contact AWS Support.

Mounting multiple Amazon EFS file systems in /etc/fstab fails

For instances that use the systemd init system with two or more Amazon EFS entries at `/etc/fstab`, there might be times where some or all of these entries are not mounted. In this case, the `dmesg` output shows one or more lines similar to the following.

```
NFS: nfs4_discover_server_trunking unhandled error -512. Exiting with error EIO
```

Action to take

In this case, we recommend that you create a new systemd service file in `/etc/systemd/system/mount-nfs-sequentially.service`. The code to include in the file depends on whether you are manually mounting the file systems or using the Amazon EFS mount helper.

- If you are manually mounting the file systems, then the `ExecStart` command must point to Network File System (NFS4). Include the following code in the file:

```
[Unit]
Description=Workaround for mounting NFS file systems sequentially at boot time
After=remote-fs.target

[Service]
Type=oneshot
ExecStart=/bin/mount -avt nfs4
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

- If you are using the Amazon EFS mount helper, then the `ExecStart` command must point to EFS instead of NFS4 to use Transport Layer Security (TLS). Include the following code in the file:

```
[Unit]
Description=Workaround for mounting NFS file systems sequentially at boot time
After=remote-fs.target

[Service]
Type=oneshot
ExecStart=/bin/mount -avt efs
RemainAfterExit=yes
```

```
[Install]
WantedBy=multi-user.target
```

After you create the file, run the following two commands:

1. `sudo systemctl daemon-reload`
2. `sudo systemctl enable mount-nfs-sequentially.service`

Then restart your Amazon EC2 instance. The file systems are mounted on demand, generally within a second.

Mount command fails with "wrong fs type" error message

The mount command fails with the following error message.

```
mount: wrong fs type, bad option, bad superblock on 10.1.25.30:/,
missing codepage or helper program, or other error (for several filesystems
(e.g. nfs, cifs) you might need a /sbin/mount.<type> helper program)
In some cases useful info is found in syslog - try dmesg | tail or so.
```

Action to take

If you receive this message, install the `nfs-utils` (or `nfs-common` on Ubuntu) package. For more information, see Installing the NFS client.

Mount command fails with "incorrect mount option" error message

The mount command fails with the following error message.

```
mount.nfs: an incorrect mount option was specified
```

Action to take

This error message most likely means that your Linux distribution doesn't support Network File System versions 4.0 and 4.1 (NFSv4). To confirm this is the case, you can run the following command.

```
$ grep CONFIG_NFS_V4_1 /boot/config*
```

If the preceding command returns `# CONFIG_NFS_V4_1 is not set`, NFSv4.1 is not supported on your Linux distribution. For a list of the Amazon Machine Images (AMIs) for Amazon Elastic Compute Cloud (Amazon EC2) that support NFSv4.1, see NFS support.

Mounting with access point fails

The mount command fails when mounting with an access point, with the following error message:

```
mount.nfs4: mounting access_point failed, reason given by server: No such file or
 directory
```

Action to take

This error message indicates that the specified EFS path does not exist. Make sure that you provide the ownership and permissions for the access point root directory. EFS will not create the root directory without this information. For more information, see Working with access points.

If you don't specify any root directory ownership and permissions, and the root directory does not already exist, EFS will not create the root directory. When this happens, any attempts to mount the file system using the access point will fail.

File system mount fails immediately after file system creation

It can take up to 90 seconds after creating a mount target for the Domain Name Service (DNS) records to propagate fully in an AWS Region.

Action to take

If you're programmatically creating and mounting file systems, for example with an AWS CloudFormation template, we recommend that you implement a wait condition.

File system mount hangs and then fails with timeout error

The file system mount command hangs for a minute or two, and then fails with a timeout error. The following code shows an example.

```
$ sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-
target-ip:/ mnt
```

```
[2+ minute wait here]
mount.nfs: Connection timed out
$Â
```

Action to take

This error can occur because either the Amazon EC2 instance or the mount target security groups are not configured properly. Make sure that the mount target security group has an inbound rule that allows NFS access from the EC2 security group. For more information, see Using VPC security groups.

Verify that the mount target IP address that you specified is valid. If you specify the wrong IP address and there is nothing else at that IP address to reject the mount, you might experience this issue.

File system mount with NFS using DNS name fails

Attempts to mount a file system using an NFS client (not using the `amazon-efs-utils` client) using the file system's DNS name fails, as shown in the following example:

```
$ sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport file-
system-id.efs.aws-region.amazonaws.com:/ mnt
mount.nfs: Failed to resolve server file-system-id.efs.aws-region.amazonaws.com:
  Name or service not known.

$
```

Action to take

Check your VPC configuration. If you are using a custom VPC, make sure that DNS settings are enabled. For more information, see DNS attributes for your VPC in the *Amazon VPC User Guide*. Also, file system and mount target DNS names are not resolvable from outside the VPC where they exist.

Before you can mount a file system using its DNS name in the `mount` command, you must do the following:

- Ensure that there's an Amazon EFS mount target in the same Availability Zone as the Amazon EC2 instance.

- Ensure that there's a mount target in the same VPC as the Amazon EC2 instance. Otherwise, you can't use DNS name resolution for EFS mount targets that are in another VPC. For more information, see Mounting EFS file systems from another AWS account or VPC.
- Connect your Amazon EC2 instance inside an Amazon VPC configured to use the DNS server provided by Amazon. For more information, see DHCP option sets in Amazon VPC in the *Amazon VPC User Guide*.
- Ensure that the Amazon VPC of the connecting Amazon EC2 instance has DNS hostnames enabled. For more information, see DNS attributes in your VPC in the *Amazon VPC User Guide*.

File system mount fails with "nfs not responding"

An Amazon EFS file system mount fails on a Transmission Control Protocol (TCP) reconnection event with "`nfs: server_name still not responding`".

Action to take

Use the `noresvport` mount option to make sure that the NFS client uses a new TCP source port when a network connection is reestablished. Doing this helps ensure uninterrupted availability after a network recovery event.

Mount target lifecycle state is stuck

The mount target lifecycle state is stuck in the **creating** or **deleting** state.

Action to take

Retry the `CreateMountTarget` or `DeleteMountTarget` call.

Mount target lifecycle state shows error

The mount target lifecycle state shows as **error**.

Action to take

Amazon EFS cannot create the necessary Domain Name System (DNS) records for new file system mount targets if the virtual private cloud (VPC) has conflicting hosted zones. Amazon EFS cannot create new records within a customer-owned hosted zone. If you need to maintain a hosted zone with a conflicting `efs.<region>.amazonaws.com` DNS range, create the hosted zone in a separate VPC. For more information about DNS considerations for VPC, see DNS attributes for your VPC.

To resolve this issue, delete the conflicting `efs.<region>.amazonaws.com` host from the VPC and create the mount target again. For more information about deleting the mount target, see Managing mount targets.

Mount does not respond

An Amazon EFS mount appears unresponsive. For example, commands like `ls` hang.

Action to take

This error can occur if another application is writing large amounts of data to the file system. Access to the files that are being written might be blocked until the operation is complete. In general, any commands or applications that attempt to access files that are being written to might appear to hang. For example, the `ls` command might hang when it gets to the file that is being written. This result is because some Linux distributions alias the `ls` command so that it retrieves file attributes in addition to listing the directory contents.

To resolve this issue, verify that another application is writing files to the Amazon EFS mount, and that it is in the `Uninterruptible sleep` (D) state, as in the following example:

```
$ ps aux | grep large_io.py
root 33253 0.5 0.0 126652 5020 pts/3 D+ 18:22 0:00 python large_io.py /efs/large_file
```

After you've verified that this is the case, you can address the issue by waiting for the other write operation to complete, or by implementing a workaround. In the example of `ls`, you can use the `/bin/ls` command directly, instead of an alias. Doing this allows the command to proceed without hanging on the file being written. In general, if the application writing the data can force a data flush periodically, perhaps by using `fsync(2)`, doing so can help improve the responsiveness of your file system for other applications. However, this improvement might be at the expense of performance when the application writes data.

Mounted client gets disconnected

A client mounted to an Amazon EFS file system can occasionally become disconnected due to any number of causes. NFS clients are designed to automatically reconnect in case of interruption to minimize the impact of routine disconnections on application performance and availability. In most cases, clients transparently reconnect within seconds.

However, the NFS client software included in older versions of the Linux kernel (versions v5.4 and below) include a behavior that causes NFS clients to, upon disconnection, attempt reconnecting on the same TCP source port. This behavior does not comply with the TCP RFC, and can prevent these clients from quickly re-establishing connections to their NFS server (in this case, an EFS file system).

To resolve this issue, we strongly recommend that you use the Amazon EFS mount helper to mount your EFS file systems. The EFS mount helper uses mount settings that are optimized for Amazon EFS file systems. For more information about the EFS client and mount helper, see Installing the Amazon EFS client.

If you can't use the EFS mount helper, we strongly recommend using the `noresvport` NFS mount option, which instructs NFS clients to re-establish connections using new TCP source ports to avoid this issue. For more information, see Recommended NFS mount settings.

Operations on newly mounted file system return "bad file handle" Error

Operations performed on a newly mounted file system return a `bad file handle` error.

This error can happen if an Amazon EC2 instance was connected to one file system and one mount target with a specified IP address, and then that file system and mount target were deleted. If you create a new file system and mount target to connect to that Amazon EC2 instance with the same mount target IP address, this issue can occur.

Action to take

You can resolve this error by unmounting the file system, and then remounting the file system on the Amazon EC2 instance. For more information about unmounting your Amazon EFS file system, see Unmounting file systems.

Unmounting a file system fails

If your file system is busy, you can't unmount it.

Action to take

You can resolve this issue in the following ways:

- Use lazy unmount, **umount -l** which detaches the file system from the file system hierarchy when run, then cleans up all references to the file system as soon as it is not busy anymore.

- Wait for all read and write operations to finish, and then attempt the **umount** command again.
- Force an unmount using the **umount -f** command.

> **Warning**
>
> Forcing an unmount interrupts any data read or write operations that are currently in process for the file system. See the umount man page for more information and guidance when using this option.

Transferring data in and out of Amazon EFS

You can use AWS DataSync and AWS Transfer Family to transfer data in and out of your Amazon EFS file systems. AWS DataSync is an online data transfer service that can copy data between Network File System (NFS), Server Message Block (SMB) file servers, self-managed object storage, and also between AWS services. For more information about using DataSync with Amazon EFS, see Using AWS DataSync to transfer data.

AWS Transfer Family is a fully managed AWS service that you can use to transfer files into and out of Amazon EFS file systems over the Secure File Transfer Protocol (SFTP), File Transfer Protocol (FTP), and FTP over Secure Sockets Layer (FTPS) protocol. Using Transfer Family, you can provide your business partners access to files stored in your Amazon EFS file systems for use cases such as data distribution, supply chain, content management, and web serving applications. For more information about using Transfer Family with Amazon EFS, see Using AWS Transfer Family to transfer data.

Topics

- Using AWS DataSync to transfer data
- Using AWS Transfer Family to transfer data

Using AWS DataSync to transfer data

AWS DataSync is an online data transfer service that simplifies, automates, and accelerates moving and replicating data between on-premises storage systems, and also between AWS storage services. DataSync can copy data between Network File System (NFS), Server Message Block (SMB) file servers, self-managed object storage, AWS Snowball Edge, Amazon S3 buckets, EFS file systems, and FSx for Windows File Server file systems.

You can also use DataSync to transfer files between two EFS file systems, including file systems in different AWS Regions and file systems owned by different AWS accounts. Using DataSync to copy data between EFS file systems, you can perform one-time data migrations, periodic data ingestion for distributed workloads, and automate replication for data protection and recovery.

For more information, see Getting started with Amazon EFS and the AWS DataSync User Guide.

Using AWS Transfer Family to transfer data

AWS Transfer Family is a fully managed AWS service that you can use to transfer files into and out of Amazon EFS file systems over the following protocols:

- Secure Shell (SSH) File Transfer Protocol (SFTP) (AWS Transfer for SFTP)
- File Transfer Protocol Secure (FTPS) (AWS Transfer for FTPS)
- File Transfer Protocol (FTP) (AWS Transfer for FTP)

Using Transfer Family, you can securely enable third parties such as your vendors, partners, or customers access to your files over the supported protocols at scale globally, without needing to manage any infrastructure. Additionally, you can now easily access your EFS file systems from Windows, macOS, and Linux environments using SFTP, FTPS, and FTP clients. This helps expand the accessibility of your data beyond NFS clients and access points, to users across multiple environments.

Using Transfer Family to transfer data in Amazon EFS file systems is accounted for in the same manner as other client usage. For more information, see Throughput modes and Amazon EFS quotas.

To learn more about AWS Transfer Family, see the AWS Transfer Family User Guide.

> **Note**
>
> Using Transfer Family with Amazon EFS is disabled by default for AWS accounts that have Amazon EFS file systems with policies that allow public access that were created before January 6, 2021. To enable using Transfer Family to access your file system, contact Support.

Topics

- Prerequisites for using AWS Transfer Family with Amazon EFS
- Configuring your EFS file system for AWS Transfer Family

Prerequisites for using AWS Transfer Family with Amazon EFS

To use Transfer Family to access files your Amazon EFS file system, your configuration must meet the following conditions:

- The Transfer Family server and your Amazon EFS file system are located in the same AWS Region.
- IAM policies are configured to enable access to the IAM role used by Transfer Family. For more information, see Create an IAM role and policy in the *AWS Transfer Family User Guide*.
- (Optional) If the Transfer Family server is owned by a different account, enable cross-account access.
  - Ensure that your file system policy does not allow public access. For more information, see Blocking public access to EFS file systems.
  - Modify the file system policy to enable cross-account access. For more information, see Configuring cross-account access for Transfer Family.

Configuring your EFS file system for AWS Transfer Family

Configuring an Amazon EFS file system to work with Transfer Family requires the following steps:

- **Step 1.** Get the list of POSIX IDs that are allocated to the Transfer Family users.
- **Step 2.** Ensure that your file system's directories are accessible to the Transfer Family users by using the POSIX IDs allocated to the Transfer Family users.
- **Step 3.** Configure IAM to enable access to the IAM role used by Transfer Family.

Setting file and directory permissions for Transfer Family users

Make sure that the Transfer Family users have access to the necessary file and directories on your EFS file system. Assign access permissions to the directory using the list of POSIX IDs allocated to the Transfer Family users. In this example, a user creates a directory named `transferFam` under the Amazon EFS mount point. Creating a directory is optional, depending on your use case. If needed, you can choose its name and location on the EFS file system.

To assign file and directory permissions to POSIX users for Transfer Family

1. Connect to your Amazon EC2 instance. Amazon EFS only supports mounting by Linux-based EC2 instances.
2. Mount your EFS file system if it is not already mounted on the EC2 instance. For more information, see [Mounting EFS file systems].
3. The following example creates the directory on the EFS file system, and changes its group to the POSIX group ID for the Transfer Family users, which is 1101 in this example.

    a. Create the directory `efs/transferFam` using the following commands. In practice, you can use a name and location on the file system of your choosing.

    ```
    [ec2-user@ip-192-0-2-0 ~]$ ls
    efs  efs-mount-point  efs-mount-point2
    [ec2-user@ip-192-0-2-0 ~]$ ls efs
    [ec2-user@ip-192-0-2-0 ~]$ sudo mkdir efs/transferFam
    [ec2-user@ip-192-0-2-0 ~]$ ls -l efs
    total 0
    drwxr-xr-x 2 root root 6 Jan  6 15:58 transferFam
    ```

    b. Use the following command to change the group of `efs/transferFam` to the POSIX GID assigned to the Transfer Family users.

    ```
    [ec2-user@ip-192-0-2-0 ~]$ sudo chown :1101 efs/transferFam/
    ```

    c. Confirm the change.

    ```
    [ec2-user@ip-192-0-2-0 ~]$ ls -l efs
    total 0
    drwxr-xr-x 2 root 1101 6 Jan  6 15:58 transferFam
    ```

Enable access to the IAM role used by Transfer Family

In Transfer Family, you create a resource-based IAM policy and an IAM role that define user access to the EFS file system. For more information, see [Create an IAM role and policy] in the *AWS Transfer Family User Guide*. You must grant that Transfer Family IAM role access to your EFS file system using either an IAM identity policy or a file system policy.

The following is an example file system policy that grants `ClientMount` (read) and `ClientWrite` access to the IAM role `EFS-role-for-transfer`.

JSON

```
{
    "Version": "2012-10-17",
    "Id": "efs-policy-wizard-8698b356-4212-4d30-901e-ad2030b57762",
    "Statement": [
        {
            "Sid": "Grant-transfer-role-access",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EFS-role-for-transfer"
            },
            "Action": [
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientMount"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/*"
        }
    ]
}
```

For more information about creating a file system policy, see Creating file system policies. For more information about using identity-based IAM policies to manage access to EFS resources, see Identity-based policies for Amazon EFS.

Configuring cross-account access for Transfer Family

If the Transfer Family server used to access your file system belongs to a different AWS account, you must grant that account access to your file system. Also, your file system policy has to be non-public. For more information about blocking public access to your file system, see Blocking public access to EFS file systems.

You can grant a different AWS account access to your file system in the file system policy. In the Amazon EFS console, use the **Grant additional permissions** section of the **File system policy editor** to specify the AWS account and the level of file system access you are granting. For more information about creating or editing a file system policy, see Creating file system policies.

You can specify the account using the account ID or the account Amazon Resource Name (ARN). For more information about ARNs, see IAM ARNs in the *IAM User Guide*.

The following example is a non-public file system policy that grants cross-account access to the file system. It has the following two statements:

1. The first statement, `NFS-client-read-write-via-fsmt`, grants read, write, and root privileges to NFS clients accessing the file system using a file system mount target.
2. The second statement, `Grant-cross-account-access`, grants only read and write privileges to the AWS account 111122223333, which is the account that owns the Transfer Family server that needs access to this EFS file system in your account.

```
{
    "Statement": [
        {
            "Sid": "NFS-client-read-write-via-fsmt",
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientRootAccess",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientMount"
            ],
            "Condition": {
                "Bool": {
                    "elasticfilesystem:AccessedViaMountTarget": "true"
                }
            }
        },
        {
            "Sid": "Grant-cross-account-access",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientMount"
            ]
        }
    ]
}
```

The following file system policy adds a statement granting access to the IAM role used by Transfer Family.

```
{
    "Statement": [
        {
            "Sid": "NFS-client-read-write-via-fsmt",
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientRootAccess",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientMount"
            ],
            "Condition": {
                "Bool": {
                    "elasticfilesystem:AccessedViaMountTarget": "true"
                }
            }
        },
        {
            "Sid": "Grant-cross-account-access",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientMount"
            ]
        },
        {
            "Sid": "Grant-transfer-role-access",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EFS-role-for-transfer"
            },
            "Action": [
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientMount"
            ]
        }
    ]
}
```

Managing EFS file systems

File system management tasks include managing a file system's network accessibility with mount targets, changing its throughput mode, updating its lifecycle policies, and managing its encryption.

You can perform these file system management tasks using the AWS Management Console, or programmatically using the AWS Command Line Interface (AWS CLI) or API, as discussed in the following sections.

Topics

- Understanding file system status
- Managing mount targets
- Managing file system throughput
- Managing storage lifecycle
- Managing access to encrypted file systems

Understanding file system status

You can view the status of Amazon EFS file systems using the Amazon EFS console or the AWS CLI. An Amazon EFS file system can have one of the status values described in the following table.

File system state	Description
AVAILABLE	The file system is in a healthy state, and is reachable and available for use.
CREATING	Amazon EFS is in the process of creating the new file system.
DELETING	Amazon EFS is deleting the file system in response to a user-initiated delete request. For more information, see Deleting EFS file systems.
DELETED	Amazon EFS has deleted the file system in response to a user-initiated delete request. For more information, see Deleting EFS file systems.
UPDATING	The file system is undergoing an update in response to a user-initiated update request.

File system state	Description
ERROR	Applicable for One Zone file systems, including file systems in a replication configuration. The file system is in a failed state and is unrecoverable. To access the file system data, restore a backup of this file system to a new file system. For more information, see Backing up and replicating data in Amazon EFS

Managing mount targets

You mount your file system on Amazon EC2 or other AWS compute instance in your virtual private cloud (VPC) using one or more mount targets that you create for the file system. You can create the mount target when you create the file system or after you create the file system.

After a mount target is created for a file system, you can create additional mount targets, delete mount targets, and modify the security groups for mount targets. If you want to modify the VPC for mount targets, then you need to first delete the existing mount targets.

> **Note**
>
> You can't change the IP address of an existing mount target. To change an IP address, you need to delete the mount target and create a new one with the new address.

Topics

- Mount targets and Availability Zones
- Creating mount targets
- Deleting mount targets
- Changing the mount target VPC
- Changing mount target security groups

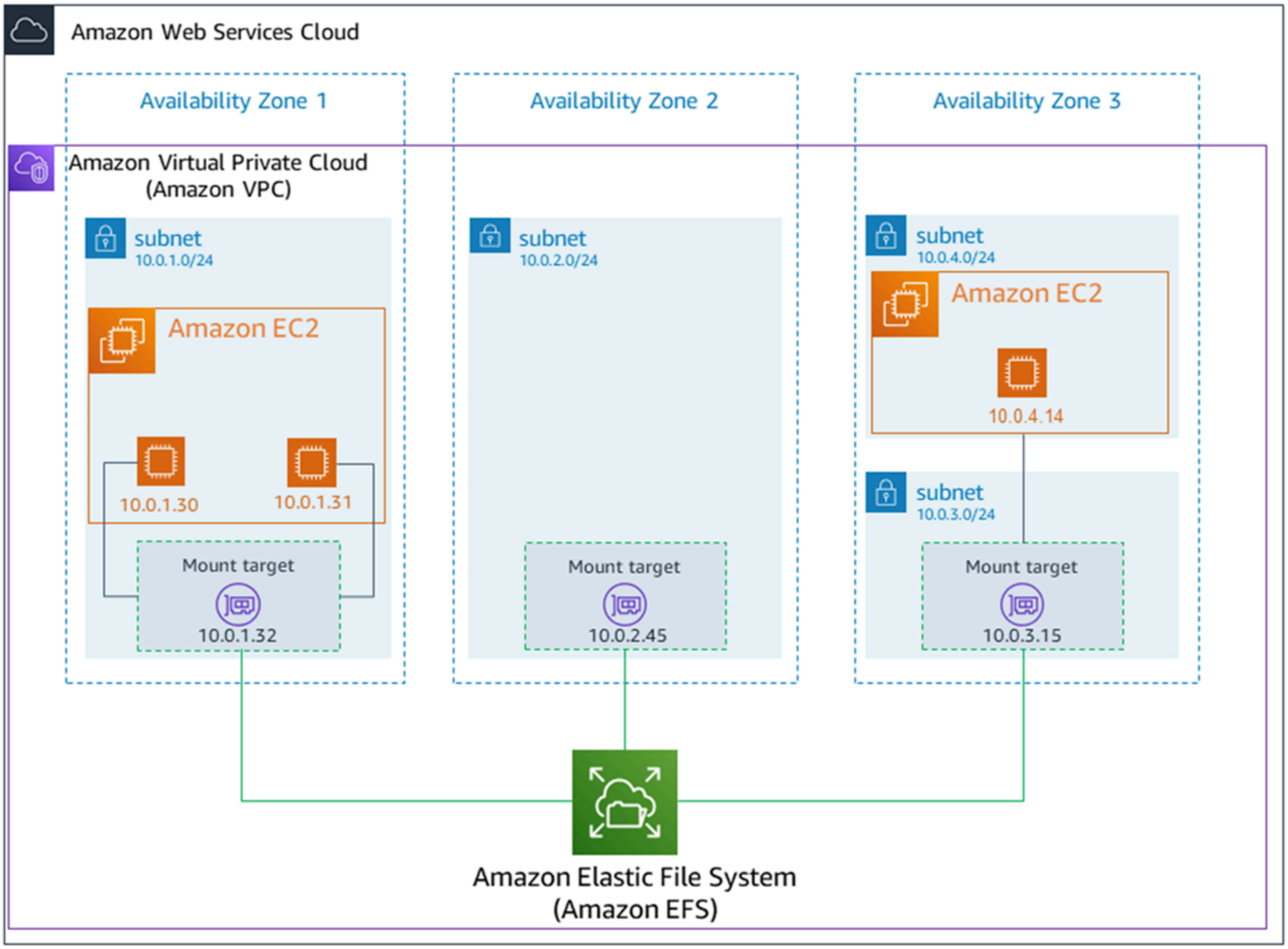
Mount targets and Availability Zones

For EFS file systems that use Regional storage classes, you can create a mount target in each Availability Zone in an AWS Region.

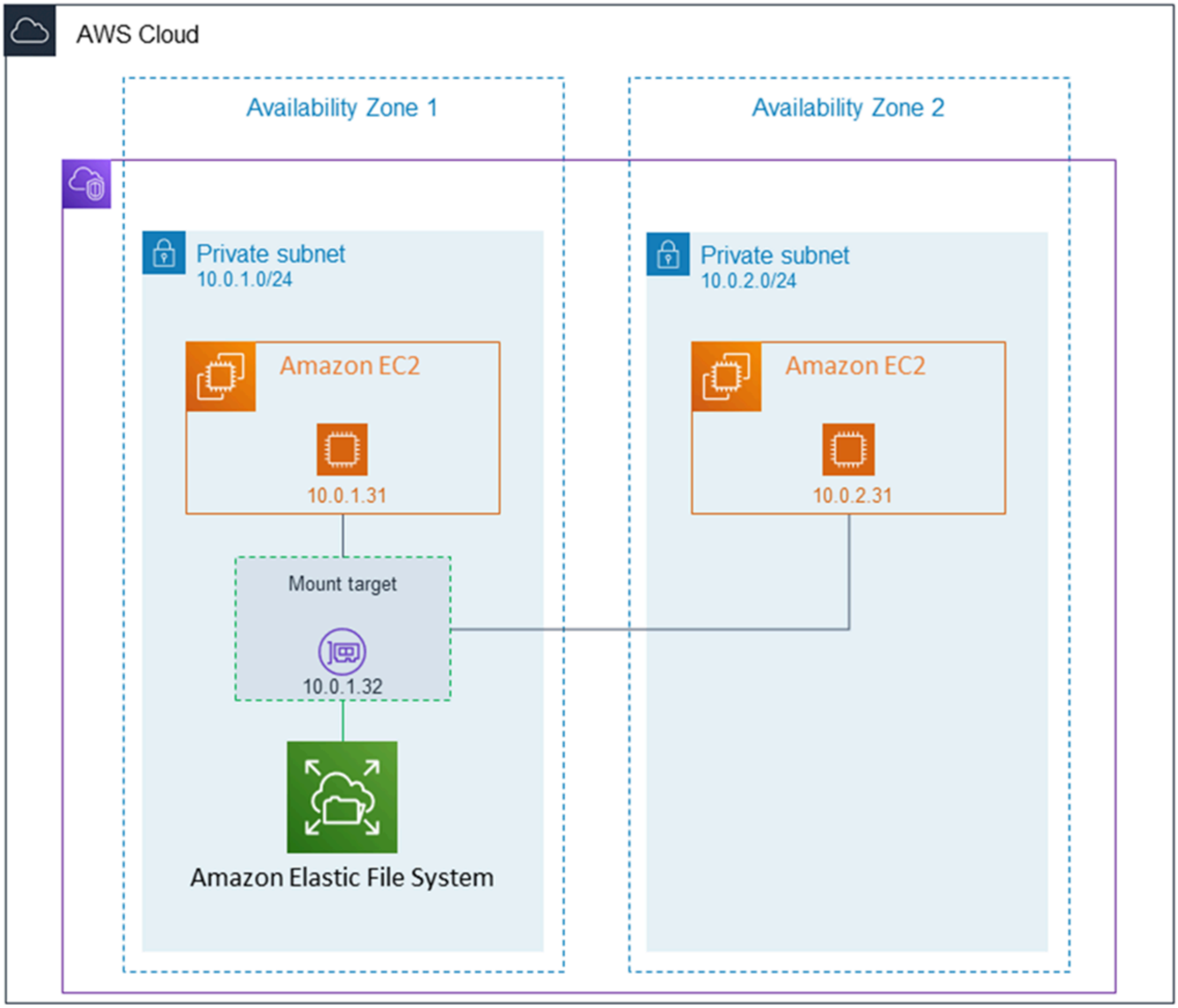
For One Zone file systems, you can only create a single mount target in the same Availability Zone as the file system. Then you can mount the file system on compute instances, including Amazon EC2, Amazon ECS, and AWS Lambda in your virtual private cloud (VPC).

The following diagram shows a Regional file system with mount targets created in all Availability Zones in the VPC. The illustration shows three EC2 instances launched in different VPC subnets accessing an EFS file system. The illustration also shows one mount target in each of the Availability Zones (regardless of the number of subnets in each Availability Zone).

You can create only one mount target per Availability Zone. If an Availability Zone has multiple subnets, as shown in one of the zones in the illustration, you create a mount target in only one of the subnets. As long as you have one mount target in an Availability Zone, the EC2 instances launched in any of its subnets can share the same mount target.



The following diagram shows a One Zone file system, with a single mount target created in the same Availability Zone as the file system. Accessing the file system by using the EC2 instance in the `us-west2c` Availability Zone incurs data access charges because it is located in a different Availability Zone than the mount target.



The mount target security group acts as a virtual firewall that controls the traffic. For example, it determines which clients can access the file system. This section explains the following:

- Managing mount target security groups and enabling traffic.
- Mounting the file system on your clients.
- NFS-level permissions considerations.

  Initially, only the root user on the Amazon EC2 instance has read-write-execute permissions on the file system. This topic discusses NFS-level permissions and provides examples that show you how to grant permissions in common scenarios. For more information, see Network File System (NFS) level users, groups, and permissions.

Creating mount targets

To access an EFS file system in a VPC, you need to create mount targets for the file system.

For an EFS file system, the following is true:

- You can create mount targets for the file system in one VPC at a time. If you want to access the file system from another VPC, you need to delete the mount targets from the current VPC and then create new mount targets in the other VPC. For more information, see Changing the mount target VPC.
- If the VPC has multiple subnets in an Availability Zone, you can create a mount target in only one of those subnets. All EC2 instances in the Availability Zone can share the single mount target.
- At a minimum, you should create a mount target in each Availability Zone from which you want to access the file system.

> **Note**
>
> There are cost considerations for mounting a file system on an EC2 instance in an Availability Zone through a mount target created in another Availability Zone. For more information, see Amazon EFS Pricing. In addition, by always using a mount target local to the instance's Availability Zone, you remove a partial failure scenario. If the mount target's zone goes down, you can't access your file system through that mount target.

You can create mount targets for a file system by using the AWS Management Console, AWS CLI, or programmatically by using the AWS SDKs. In the console, you can create mount targets when you create the file system or after the file system is created. For instructions on creating mount targets when creating a file system, see Custom create using the console.

Using the console

Use the following procedure to add mount targets to an existing EFS file system.

To create a mount target on an EFS file system

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the left navigation pane, choose **File systems**, and then select the file system for which you want to change the VPC.
3. Choose **Network** and then choose **Manage** to display the mount targets for the file system.
4. Choose the file system that you want to add mount targets for by choosing its **Name** or the **File system ID**.

   > **Note**
   >
   > For One Zone file systems, you can only create a single mount target that is in the same Availability Zone as the file system.

5. For file systems that use EFS Regional storage classes, choose **Add mount target** for each mount target you want to create for the file system.
6. Define the mount target settings:

   a. Choose the Availability Zone and subnet ID for the mount target.

   b. For **IP address type**, choose **IPv4 only** to support IPv4 addresses only, **IPv6 only** to support IPv6 addresses only, or **Dual-stack** to support both IPv4 and IPv6 addresses.

      > **Note**
      >
      > The IP address type must match the IP type of the subnet. Additionally, the IP address type overrides the IP addressing attribute of your subnet. For example, if the IP address type is IPv4-only and the IPv6 addressing attribute is enabled for your subnet, network interfaces created in the subnet receive an IPv4 address from the range of the subnet. For more information, see Modify the IP addressing attributes of your subnet.

   c. If you know the IP address where you want to place the mount target, then enter it in the IP address box that matches the **IP address type**. If you don't specify a value, Amazon EFS selects an unused IP address from the specified subnet.

> **Note**
>
> You can't change the IP address of a mount target after it's created. To change an IP address, you need to delete the mount target and create a new one with the new address.

7. Choose at least one security group to associate with the mount target. You can modify the security groups later.
8. Choose **Save**.

Using the AWS CLI

This section provide examples for creating a mount target in the AWS CLI using the `create-mount-target` command. The equivalent API command is CreateMountTarget.

- If you don't specify an IP address type for the mount target, then IPv4-only is used.
- If you don't specify an IP address for the mount target, then Amazon EFS assigns an available address on the specified subnet.
- The IP address type overrides the IP addressing attribute of your subnet. For example, if the IP address type is IPv4-only and the IPv6 addressing attribute is enabled for your subnet, network interfaces created in the subnet receive an IPv4 address from the range of the subnet. For more information, see Modify the IP addressing attributes of your subnet.

> **Note**
>
> For One Zone file systems, you can only create a single mount target that is in the same Availability Zone as the file system.

Example: Create mount target at an available IPv4 address on a subnet

The following command specifies the file system, subnet, and security group for the mount target. The target is created at an available IPv4 address on the specified subnet.

```
$ aws efs create-mount-target \
--file-system-id file-system-id \
```

```
--subnet-id  subnet-id \
--security-group ID-of-the-security-group-created-for-mount-target \
--region aws-region \
```

The following example shows the command with sample data.

```
$ aws efs create-mount-target \
--file-system-id fs-0123456789abcdef1 \
--subnet-id  subnet-b3983dc4 \
--security-group sg-01234567 \
--region us-east-2 \
```

After successfully creating the mount target, Amazon EFS returns the mount target description as JSON as shown in the following example.

```
{
    "OwnerID": "111122223333"
    "MountTargetId": "fsmt-f9a14450",
    "FileSystemId": "fs-0123456789abcdef1",
    "SubnetId": "subnet-b3983dc4",
    "LifeCycleState": "available",
    "IpAddress": "10.0.1.24",
    "NetworkInterfaceId": "eni-3851ec4e",
    "AvailabilityZoneId": "use2-az1",
    "AvailabilityZoneName": "us-east-2a",
    "VpcId": "vpc-3c39ef57"
}
```

Example: Create a mount target at a specific IPv4 address

The following command specifies the file system, subnet, security group, and IPv4 address to use for the mount target. The target is created at the specified IPv4 address on the specified subnet.

```
$ aws efs create-mount-target \
--file-system-id file-system-id \
--subnet-id  subnet-id \
--security-group ID-of-the-security-group-created-for-mount-target \
--ip-address IPv4-address
--region aws-region \
```

The following example shows the command with sample data.

```
$ aws efs create-mount-target \
--file-system-id fs-0123456789abcdef1 \
--subnet-id  subnet-b3983dc4 \
--security-group sg-01234567 \
--ip-address 10.0.1.24 \
--region us-east-2 \
```

After successfully creating the mount target, Amazon EFS returns the mount target description as JSON as shown in the following example.

```
{
    "OwnerID": "111122223333"
    "MountTargetId": "fsmt-f9a14450",
    "FileSystemId": "fs-0123456789abcdef1",
    "SubnetId": "subnet-b3983dc4",
    "LifeCycleState": "available",
    "IpAddress": "10.0.1.24",
    "NetworkInterfaceId": "eni-3851ec4e",
    "AvailabilityZoneId": "use2-az1",
    "AvailabilityZoneName": "us-east-2a",
    "VpcId": "vpc-3c39ef57"
}
```

Example: Create a mount target at a specific IPv6 address

The following command specifies the file system, subnet, security group, and IPv6 address to use for the mount target. The target is created at the specified IPv6 address on the specified subnet.

```
$ aws efs create-mount-target \
--file-system-id file-system-id \
--subnet-id  subnet-id \
--security-group ID-of-the-security-group-created-for-mount-target \
--ip-address-type IP-address-type \
--ipv6-address IPv6-address \
--region aws-region \
```

The following example shows the command with sample data.

```
$ aws efs create-mount-target \
--file-system-id fs-0123456789abcdef1 \
--subnet-id  subnet-b3983dc4 \
```

```
--security-group sg-01234567 \
--ip-address-type IPV6_ONLY \
--ipv6-address 2001:0db8:85a3:0000:0000:8a2e:0370:7334 \
--region us-east-2 \
```

After successfully creating the mount target, Amazon EFS returns the mount target description as JSON as shown in the following example.


```
{
    "OwnerID": "111122223333"
    "MountTargetId": "fsmt-f9a14450",
    "FileSystemId": "fs-0123456789abcdef1",
    "SubnetId": "subnet-b3983dc4",
    "LifeCycleState": "available",
    "Ipv6Address": "2001:0db8:85a3:0000:0000:8a2e:0370:7334",
    "NetworkInterfaceId": "eni-3851ec4e",
    "AvailabilityZoneId": "use2-az1",
    "AvailabilityZoneName": "us-east-2a",
    "VpcId": "vpc-3c39ef57"
}
```


Example: Create a mount target at an available IPv4 address on dual-stack subnet

The command specifies the file system, subnet, security group, dual-stack IP address type, and IPv6 address for the mount target. The target is created at an available IPv4 address and the specified IPv6 address on the dual-stack subnet.

```
$ aws efs create-mount-target \
--file-system-id file-system-id \
--subnet-id  subnet-id \
--security-group ID-of-the-security-group-created-for-mount-target \
--ip-address-type IP-address-type
--ipv6-address IPv6-address \
--region aws-region \
```

The following example shows the command with sample data.

```
$ aws efs create-mount-target \
--file-system-id fs-0123456789abcdef1 \
--subnet-id  subnet-b3983dc4 \
--security-group sg-01234567 \
```

```
--ip-address-type DUAL_STACK \
--ipv6-address 2001:0db8:85a3:0000:0000:8a2e:0370:7334 \
--region us-east-2 \
```

After successfully creating the mount target, Amazon EFS returns the mount target description as JSON as shown in the following example.

```
{
    "OwnerID": "111122223333"
    "MountTargetId": "fsmt-f9a14450",
    "FileSystemId": "fs-0123456789abcdef1",
    "SubnetId": "subnet-b3983dc4",
    "LifeCycleState": "available",
    "IpAddress": "10.0.1.24",
    "Ipv6Address": "2001:0db8:85a3:0000:0000:8a2e:0370:7334",
    "NetworkInterfaceId": "eni-3851ec4e",
    "AvailabilityZoneId": "use2-az1",
    "AvailabilityZoneName": "us-east-2a",
    "VpcId": "vpc-3c39ef57"
}
```

Deleting mount targets

When you delete a mount target, the operation forcibly breaks any mounts of the file system, which might disrupt instances or applications using those mounts. To avoid application disruption, stop applications and unmount the file system before deleting the mount target. For more information, see Unmounting file systems.

You can delete mount targets for a file system by using the AWS Management Console, AWS CLI, or programmatically by using the AWS SDKs.

Using the console

Use the following procedure to delete mount targets for an existing EFS file system.

To delete mount targets on an EFS file system

1. Unmount the file system. For instructions, see Unmounting file systems.
2. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
3. In the left navigation pane, choose **File systems**, and then select the file system for which you want to delete the mount target.

4. Choose **Network** and then choose **Manage** to display the mount targets for the file system.
5. For each mount target you want to delete, choose **Remove**.
6. Choose **Save**.

Using the AWS CLI

To delete an existing mount target, use the `delete-mount-target` AWS CLI command (corresponding operation is DeleteMountTarget), as shown following.

> **Note**
>
> Before deleting a mount target, first unmount the file system.

```
$ aws efs delete-mount-target \
--mount-target-id mount-target-ID-to-delete \
--region aws-region-where-mount-target-exists
```

The following is an example with sample data.

```
$ aws efs delete-mount-target \
--mount-target-id fsmt-5751852e \
--region us-east-2 \
```

Changing the mount target VPC

You can use an EFS file system in one VPC based on the Amazon VPC service at a time. That is, you create mount targets in a VPC for your file system, and use those mount targets to provide access to the file system.

You can mount the EFS file system from these targets:

- Amazon EC2 instances in the same VPC
- EC2 instances in a VPC connected by VPC peering
- On-premises servers by using AWS Direct Connect
- On-premises servers over an AWS virtual private network (VPN) by using Amazon VPC

A *VPC peering connection* is a networking connection between two VPCs that enables you to route traffic between them. The connection can use private Internet Protocol version 4 (IPv4) or version 6 (IPv6). For more information on how Amazon EFS works with VPC peering, see Mounting EFS file systems from another AWS account or VPC.

To change the VPC for a file system

The following are the steps that you'll perform to change the VPC for an EFS file system's network configuration.

1. Delete each mount target assigned to the file system. For instructions, see Deleting mount targets.
2. When the **Mount targets status** for each mount target is **Deleted**, assign the new VPC and create new mount targets for the file system. For instructions, see Creating mount targets.

Changing mount target security groups

Security groups define inbound and outbound access. When you change security groups associated with a mount target, make sure that you authorize necessary inbound and outbound access. Doing so enables your EC2 instance to communicate with the file system. For more information about security groups, see Using VPC security groups.

You can add or remove security groups for a file system's mount target by using the AWS Management Console, AWS CLI, or programmatically by using the AWS SDKs.

Using the console

To modify security groups for mount targets

Use the following procedure to add or remove mount target security groups for an existing EFS file system.

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the left navigation pane, choose **File systems**, and then select the file system for which you want to manage mount targets.
3. Choose **Network** and then choose **Manage** to display the mount targets for the file system.
4. To remove a security group from a mount target, choose **X** next to the security group ID.
5. To add a security group to a mount target, choose the security from the **Security groups** list.

6. Choose **Save**.

Using the AWS CLI

To modify security groups that are in effect for a mount target, use the `modify-mount-target-security-group` AWS CLI command (the corresponding operation is ModifyMountTargetSecurityGroups) to replace any existing security groups, as shown following.

```
$ aws efs modify-mount-target-security-groups \
--mount-target-id mount-target-ID-whose-configuration-to-update \
--security-groups  security-group-ids-separated-by-space \
--region aws-region-where-mount-target-exists \
--profile adminuser
```

The following is an example with sample data.

```
$ aws efs modify-mount-target-security-groups \
--mount-target-id fsmt-5751852e \
--security-groups  sg-1004395a sg-1114433a \
--region us-east-2
```

Managing file system throughput

Elastic throughput is the default throughput mode in the Amazon EFS console and is recommended for most use cases. With Elastic throughput, performance automatically scales up or down to meet the needs of your workload activity. If, however, you know the specific access patterns for your workloads (including throughput, latency, and storage needs), then you can choose to change the throughput mode.

> **Note**
>
> While Elastic throughput is designed to scale elastically with your throughput, we recommend implementing proper governance through monitoring metrics with CloudWatch (MeteredIOBytes) and usage alerts as part of your operational best practices. This helps you maintain optimal resource utilization and stay within your planned operational parameters. For more information, see Monitoring metrics with Amazon CloudWatch.

Other throughput modes you can choose include:

- **Provisioned throughput** – You specify a level of throughput that the file system can drive independent of the file system's size or burst credit balance.
- **Bursting throughput** – Throughput scales with the amount of storage in your file system and supports bursting to higher levels for up to 12 hours per day.

For more information about Amazon EFS throughput modes, see Throughput modes.

> **Note**
>
> You can change the throughput mode and the provisioned throughput amount after the file system is available. Changing throughput mode does not cause application downtime. However, any time that you change the file system to Provisioned throughput or increase the provisioned throughput amount, you must wait at least 24 hours before you can change the throughput mode again or decrease the provisioned amount.

You can manage the file system throughput mode by using the Amazon EFS console, the AWS Command Line Interface (AWS CLI), and Amazon EFS API.

Using the console

To manage file system throughput

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the left navigation pane, choose **File systems** to display the list of EFS file systems in your account.
3. Choose the file system that you want to change the throughput mode for.
4. On the file system details page, in the **General** section, choose **Edit**. The **Edit** page displays.
5. Modify the **Throughput mode** setting.

   - To use Elastic throughput or Provisioned throughput, choose **Enhanced**, and then choose **Elastic** or **Provisioned**.

     If you choose **Provisioned**, then, in **Provisioned Throughput (MiB/s)**, enter the amount of throughput to provision for file system requests. The amount of **Maximum Read Throughput** is displayed at three times the amount of the throughput that you enter. EFS file systems meter read requests at one-third the rate of other requests. After you enter the throughput, an estimate of the monthly cost for the file system is shown.

> **Note**
>
> You can change the throughput mode and the provisioned throughput amount after the file system is available. However, any time that you change the file system throughput to Provisioned or increase the provisioned throughput amount, you must wait at least 24 hours before you can change the throughput mode again or decrease the provisioned amount.

- To use Bursting throughput, choose **Bursting**.

For more information about choosing the correct throughput mode for your performance needs, see Throughput modes.

6. Choose **Save changes** to implement your changes.

Using the AWS CLI

Use the update-file-system CLI command, or the UpdateFileSystem API action to change a file system's throughput mode.

Managing storage lifecycle

You can manage your file systems so that they have cost-effective storage throughout their lifecycle. Use lifecycle management to automatically transition data between storage classes according to the lifecycle configuration for the file system. The lifecycle configuration comprises of three *lifecycle policies* that you set for the file system.

Lifecycle policies instruct lifecycle management when to transition files into and out of the EFS Infrequent Access (IA) and EFS Archive storage classes. Transition time is based on when the files were last accessed in the Standard storage class. To determine last accessed time in the Standard storage class, an internal timer tracks when a file was last accessed (not the POSIX file system attributes that are publicly viewable). Whenever a file in Standard is accessed, the lifecycle management timer is reset.

Lifecycle policies apply to the entire EFS file system.

The EFS lifecycle policies are:

- **Transition into IA** – Instructs lifecycle management when to move files into the Infrequent Access storage, which is cost-optimized for data that is accessed only a few times each quarter. By default, files that are not accessed in Standard storage for 30 days are transitioned into IA.
- **Transition into Archive** – Instructs lifecycle management when to move files into the Archive storage class, which is cost-optimized for data that is accessed only a few times each year or less. By default, files that are not accessed in Standard storage for 90 days are transitioned into Archive.
- **Transition into Standard** – Instructs lifecycle management whether to transition files out of IA or Archive and back into Standard storage when the files are accessed in the IA or Archive storage. By default, files are not moved back to Standard storage, and they remain in the IA or Archive storage class when they are accessed.

  For performance-sensitive use cases that demand the fastest latency performance (such as applications that work with a large volume of small files), choose to transition files into Standard storage **On first access**.

For more information about configuring the lifecycle policies for a file system, see Configuring lifecycle policies.

File system operations for lifecycle management

File system operations for lifecycle management have a lower priority than operations for EFS file system workloads. The time required to transition files in to or out of IA and Archive storage varies depending on the file size and file system workload. For example, transitioning millions of small files may take longer than transitioning fewer larger files of the same total storage size.

File metadata, including file names, ownership information, and file system directory structure, is always stored in Standard to help ensure consistent metadata performance.

Metadata operations for file systems in IA or Archive storage, such as listing the contents of a directory, don't count as file access. During the process of transitioning a file's content to the IA or Archive storage classes, the file is stored in the Standard storage class and is billed at that storage rate.

All write operations to files in the file system's IA or Archive storage classes are first written to Standard storage classes, and are then eligible to be transitioned to the applicable storage class after 24 hours.

Configuring lifecycle policies

When you create an EFS file system that has the recommended settings using the AWS Management Console, the file system is automatically configured with the following default lifecycle configuration:

- **Transition into IA** is set to **30 days since last access**.
- **Transition into Archive** is set to **90 days since last access**.
- **Transition into Standard** is set to **None**.

You can change the default lifecycle policies when creating a file system with customized settings using the AWS Management Console or when creating a file system using the AWS CLI. Alternately, you can change the policies after the file system is created, as described in the following procedures.

Using the console

You can use the AWS Management Console to set the lifecycle policies for an existing file system.

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. Choose **File systems** to display the list of file systems in your account.
3. Choose the file system on which you want to modify lifecycle policies.
4. On the file system details page, in the **General** section, choose **Edit**. The **Edit** page displays.
5. For Lifecycle management, configure the lifecycle policies:
   - Set **Transition into IA** to one of the available options. To stop moving files into IA storage, choose **None**.
   - Set **Transition into Archive** to one of the available options. To stop moving files into Archive storage, choose **None**.
   - Set **Transition into Standard** to **On first access** to move files that are in IA storage to standard storage when they're accessed for non-metadata operations.

To stop moving files from IA or Archive to Standard storage on first access, select **None**.

6. Choose **Save changes** to save your changes.

Using the AWS CLI

You can use the AWS CLI to set or modify a file system's lifecycle policies.

- Run the put-lifecycle-configuration AWS CLI command or the PutLifecycleConfiguration API command, specifying the file system ID of the file system for which you are managing lifecycle management.

```
$  aws efs put-lifecycle-configuration \
--file-system-id File-System-ID \
--lifecycle-policies "[{\"TransitionToIA\":\"AFTER_60_DAYS\"},
{\"TransitionToPrimaryStorageClass\":\"AFTER_1_ACCESS\"},{\"TransitionToArchive\":
\"AFTER_90_DAYS\"}]" \
--region us-west-2 \
--profile adminuser
```

You get the following response.

```
{
    "LifecyclePolicies": [
        {
            "TransitionToIA": "AFTER_60_DAYS"
        },
        {
            "TransitionToPrimaryStorageClass": "AFTER_1_ACCESS"
        },
        {
            "TransitionToArchive": "AFTER_90_DAYS"
        }
    ]
}
```

To stop lifecycle management for an existing file system (CLI)

- Run the `put-lifecycle-configuration` command specifying the file system ID of the file system for which you are stopping lifecycle management. Keep the `--lifecycle-policies` property empty.

  ```
  $  aws efs put-lifecycle-configuration \
  --file-system-id File-System-ID \
  --lifecycle-policies \
  --region us-west-2 \
  --profile adminuser
  ```

  You get the following response.

  ```
  {
      "LifecyclePolicies": []
  }
  ```

Managing access to encrypted file systems

Using Amazon EFS, you can create encrypted file systems. Amazon EFS supports two forms of encryption for file systems, encryption in transit and encryption at rest. Any key management that you need to perform is related only to encryption at rest. Amazon EFS automatically manages the keys for encryption in transit.

If you create a file system that uses encryption at rest, data and metadata are encrypted at rest. Amazon EFS uses AWS Key Management Service (AWS KMS) for key management. When you create a file system using encryption at rest, you specify an AWS KMS key. The KMS key can be `aws/elasticfilesystem` (the AWS managed key for Amazon EFS), or it can be a customer managed key that you manage.

File data—the contents of your files—is encrypted at rest using the KMS key that you specified when you created your file system. Metadata—file names, directory names, and directory contents—is encrypted using a key that Amazon EFS manages.

The EFS AWS managed key for your file system is used as the KMS key for encrypting the metadata in your file system, for example file names, directory names, and directory contents. You own the customer managed key used to encrypt file data (the contents of your files) at rest.

You manage who has access to your KMS keys and the contents of your encrypted file systems. This access is controlled by both AWS Identity and Access Management (IAM) policies and AWS KMS. IAM policies control a user's access to Amazon EFS API actions. AWS KMS key policies control a user's access to the KMS key that you specified when the file system was created. For more information, see the following:

- IAM Users in the *IAM User Guide*
- Key policies in AWS KMS in the *AWS Key Management Service Developer Guide*
- Grants in AWS KMS in the *AWS Key Management Service Developer Guide*.

As a key administrator, you can import external keys. You can also modify keys by enabling them, disabling them, or deleting them. The state of the KMS key that you specified (when you created the file system with encryption at rest) affects access to its contents. The KMS key must be in the `enabled` state for users to have access to the contents of an encrypted-at-rest file system that is encrypted using that key.

Managing KMS keys for EFS file systems

You can disable or delete your customer managed KMS keys, or you can revoke Amazon EFS access to your KMS keys. Disabling and revoking access for Amazon EFS to your keys are reversible actions. Exercise significant caution when deleting KMS keys. Deleting a KMS key is an irreversible action.

If you disable or delete the KMS key used for your mounted file system, the following is true:

- That KMS key can't be used as the key for new encrypted-at-rest file systems.
- Existing encrypted-at-rest file systems that use that KMS key stop working after a period of time.

If you revoke Amazon EFS access to a grant for any existing mounted file system, the behavior is the same as if you disabled or deleted the associated KMS key. In other words, the encrypted-at-rest file system continues to function, but stops working after a period of time.

You can prevent access to a mounted encrypted-at-rest file system that has a KMS key that you disabled, deleted, or revoked Amazon EFS access to. To do this, unmount the file system and delete your Amazon EFS mount targets.

You can't immediately delete an AWS KMS key, but you can schedule it for deletion in 7-30 days. While a KMS key is scheduled for deletion, you can't use it for cryptographic operations. You can also cancel a KMS key's scheduled deletion.

To learn how to disable and re-enable customer managed KMS keys, see Enabling and disabling keys in the AWS Key Management Service Developer Guide. To learn how to schedule deletion of customer managed KMS keys, see Deleting KMS keys in the AWS Key Management Service Developer Guide.

Monitoring Amazon EFS

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon EFS and your AWS solutions. We recommend that you collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring Amazon EFS, however, create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

The next step is to establish a baseline for normal Amazon EFS performance in your environment, by measuring performance at various times and under different load conditions. As you monitor Amazon EFS, consider storing historical monitoring data. This stored data will give you a baseline to compare against with current performance data, identify normal performance patterns and performance anomalies, and devise methods to address issues.

For example, with Amazon EFS, you can monitor network throughput, I/O for read, write, and metadata operations, client connections, and burst credit balances for your file systems. If performance falls outside your established baseline, you might need to change the size of your file system or the number of connected clients to optimize the file system for your workload.

To establish a baseline, you should, at a minimum, monitor the following items:

- Your file system's network throughput.
- The number of client connections to a file system.
- The number of bytes for each file system operation, including data read, data write, and metadata operations.

Topics

- Monitoring tools

- How Amazon EFS reports file system and object sizes
- Viewing storage class size
- Monitoring metrics with Amazon CloudWatch
- Logging Amazon EFS API calls with AWS CloudTrail

Monitoring tools

AWS provides various tools that you can use to monitor Amazon EFS. You can configure some of these tools to do the monitoring for you, but some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

Automated monitoring tools

You can use the following automated monitoring tools to watch Amazon EFS and report when something is wrong:

- **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch alarms do not invoke actions only because they are in a particular state; the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring metrics with Amazon CloudWatch.
- **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see What is Amazon CloudWatch Logs? in the *Amazon CloudWatch User Guide*.
- **Amazon CloudWatch Events** – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action. For more information, see What is Amazon CloudWatch Events in the *Amazon CloudWatch User Guide*.
- **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see Working with CloudTrail log files in the *AWS CloudTrail User Guide*.

Manual monitoring tools

Another important part of monitoring Amazon EFS involves manually monitoring those items that the Amazon CloudWatch alarms don't cover. The Amazon EFS, CloudWatch, and other AWS Management Console dashboards provide an at-a-glance view of the state of your AWS environment. We recommend that you also check the log files for the file systems.

- From the Amazon EFS console, you can find the following items for your file systems:
  - The current metered size
  - The number of mount targets
  - The lifecycle state
- CloudWatch home page shows:
  - Current alarms and status
  - Graphs of alarms and resources
  - Service health status

  In addition, you can use CloudWatch to do the following:
  - Create customized dashboards to monitor the services that you use.
  - Graph metric data to troubleshoot issues and discover trends.
  - Search and browse all your AWS resource metrics.
  - Create and edit alarms to be notified of problems.

How Amazon EFS reports file system and object sizes

The following sections describe how Amazon EFS reports file system sizes, sizes of objects within a file system, and file system throughput.

Metering EFS file system objects

Objects that you can view in an EFS file system include regular files, directories, symbolic links, and special files (FIFOs and sockets). Each of these objects is metered for 2 kibibytes (KiB) of metadata (for its inode) and one or more increments of 4 KiB of data. The following list explains the metered data size for different types of file system objects:

- **Regular files** – The metered data size of a regular file is the logical size of the file rounded to the next 4-KiB increment, except that it might be less for sparse files.

A *sparse file* is a file to which data is not written to all positions of the file before its logical size is reached. For a sparse file, in some cases the actual storage used is less than the logical size rounded to the next 4-KiB increment. In these cases, Amazon EFS reports actual storage used as the metered data size.

- **Directories** – The metered data size of a directory is the actual storage used for the directory entries and the data structure that holds them, rounded to the next 4-KiB increment. The metered data size doesn't include the actual storage used by the file data.
- **Symbolic links and special files** – The metered data size for these objects is always 4 KiB.

When Amazon EFS reports the space used for an object, through the NFSv4.1 `space_used` attribute, it includes the object's current metered data size but not its metadata size. You can use two utilities for measuring the disk usage of a file, the `du` and `stat` utilities. Following is an example of how to use the `du` utility on an empty file that includes the `-k` option to return the output in kilobytes.

```
$ du -k file
4      file
```

Following example shows how to use the `stat` utility on an empty file to return the file's disk usage.

```
$ /usr/bin/stat --format="%b*%B" file | bc
4096
```

To measure the size of a directory, use the `stat` utility. Find the `Blocks` value, and then multiply that value by the block size. Following is an example of how to use the `stat` utility on an empty directory:

```
$ /usr/bin/stat --format="%b*%B" . | bc
4096
```

Metered size of an EFS file system

The metered size of an EFS file system includes the sum of the sizes of all current objects in all of the EFS storage classes. The size of each object is calculated from a representative sampling of the size of the object during the metered hour, for example from 8 AM to 9 AM.

An empty file contributes 6 KiB (2 KiB metadata + 4 KiB data) to the metered size of a file system. Upon creation, a file system has a single empty root directory and therefore has a metered size of 6 KiB.

The metered sizes of a particular file system define the usage for which the owner account is billed for that file system for that hour.

> **Note**
>
> The computed metered size doesn't represent a consistent snapshot of the file system at any particular time during that hour. Instead, it represents the sizes of the objects that existed in the file system at varying times within each hour, or possibly the hour before it. These sizes are summed to determine the file system's metered size for the hour. The metered size of a file system is thus eventually consistent with the metered sizes of the objects stored when there are no writes to the file system.

You can see the metered size for an EFS file system in the following ways:

- Using the describe-file-systems AWS CLI command and the DescribeFileSystem API operation, the response includes the following:

  ```
  "SizeInBytes":{
              "Timestamp": 1403301078,
              "Value": 29313744866,
              "ValueInIA": 675432,
              "ValueInStandard": 29312741784
              "ValueInArchive": 327650
           }
  ```

  Where the metered size of `ValueInStandard` is also used to determine your I/O throughput baseline and burst rates for file systems using the Bursting throughput.
- View the `StorageBytes` CloudWatch metric, which displays the total metered size of data in each storage classes. For more information about the `StorageBytes` metric, see CloudWatch metrics for Amazon EFS.
- Run the `df` command in Linux at the terminal prompt of an EC2 instance.

  Don't use the **du** command on the root of the file system for storage metering purposes because the response does not reflect the full set data used for metering your file system.

> **Note**
>
> The metered size of `ValueInStandard` is also used to determine your I/O throughput baseline and burst rates. For more information, see Bursting throughput.

Metering Infrequent Access and Archive storage classes

The EFS Infrequent Access (IA) and Archive storage classes are metered in 4 KiB increments and have a minimum billing charge per file of 128 KiB. IA and Archive file metadata (2 KiB per file) is always stored and metered in the Standard storage class. Support for files smaller than 128 KiB is only available for lifecycle policies updated on or after 12:00 PM PT, November 26, 2023. Data access for IA and Archive storage is metered in 128 KiB increments.

You can use the `StorageBytes` CloudWatch metric to view the metered size of data in each of the storage classes. The metric also displays the total number of bytes that are consumed by small-file rounding within the IA and Archive storage classes. For more information about viewing CloudWatch metrics, see Accessing CloudWatch metrics for Amazon EFS. For more information about the `StorageBytes` metric, see CloudWatch metrics for Amazon EFS.

Metering throughput

Amazon EFS meters the throughput for read requests at one-third the rate of the other file system I/O operations. For example, if you are driving 30 mebibytes per second (MiBps) of both read and write throughput, the read portion counts as 10 MiBps of effective throughput, the write portion counts as 30 MiBps, and the combined metered throughput is 40 MiBps. This combined throughput adjusted for consumption rates is reflected in the `MeteredIOBytes` CloudWatch metric.

Metering Elastic throughput

When Elastic throughput mode is enabled for a file system, you pay only for the amount of metadata and data read from or written to the file system. EFS file systems using Elastic throughput mode meter and bill metadata reads as read operations and metadata writes as write operations. Metadata operations are metered in 1 KiB increments after the first 4 KiB. Data operations are metered in 1 KiB increments after the first 32 KiB.

> **Note**
>
> While Elastic throughput is designed to scale elastically with your throughput, we recommend implementing proper governance through monitoring metrics with CloudWatch (MeteredIOBytes) and usage alerts as part of your operational best practices. This helps you maintain optimal resource utilization and stay within your planned operational parameters. For more information, see Monitoring metrics with Amazon CloudWatch.

Metering Provisioned throughput

For file systems that use Provisioned throughput mode, you pay only for the amount of time that throughput is enabled. Amazon EFS meters file systems with Provisioned throughput mode enabled once every hour. For metering when Provisioned throughput mode is set for less than one hour, Amazon EFS calculates the time-average using millisecond precision.

Viewing storage class size

You can view how much data is stored in each storage class of your file system using the Amazon EFS console, the AWS CLI, or the EFS API.

Using the console

The **Metered size** tab on the **File system details** page displays the current metered size of the file system in binary multiples of bytes (kibibytes, mebibytes, gibibytes, and tebibytes). The metric is emitted every 15 minutes and lets you view your file system's metered size over time. **Metered size** displays the following information for the file system storage size:

- **Total size** is the size (in binary bytes) of data stored in the file system, including all storage classes.
- **Size in Standard** is the size (in binary bytes) of data stored in the EFS Standard storage class.
- **Size in IA** is the size (in binary bytes) of data stored in the EFS Infrequent Access storage class. Files smaller than 128KiB are rounded up to 128KiB.
- **Size in Archive** is the size (in binary bytes) of data stored in the EFS Archive storage class. Files smaller than 128KiB are rounded up to 128KiB.

You can also view the `Storage bytes` metric on the **Monitoring** tab on the **File system details** page in the Amazon EFS console. For more information, see Accessing CloudWatch metrics for Amazon EFS.

Using the AWS CLI

You can view how much data is stored in each storage class of your file system using the AWS CLI or EFS API. View data storage details by calling the `describe-file-systems` CLI command (the corresponding API operation is DescribeFileSystems).

```
$  aws efs describe-file-systems \
--region us-west-2 \
--profile adminuser
```

In the response, `ValueInIA` displays the last metered size in bytes in the file system's Infrequent Access storage class. `ValueInStandard` displays the last metered size in bytes in the Standard storage class. `ValueInArchive` displays the last metered size in bytes in the Archive storage class. The sum of the three values equals the size of the entire file system, which is displayed in `Value`.

```
{
   "FileSystems":[
      {
         "OwnerId":"251839141158",
         "CreationToken":"MyFileSystem1",
         "FileSystemId":"fs-47a2c22e",
         "PerformanceMode" : "generalPurpose",
         "CreationTime": 1403301078,
         "LifeCycleState":"created",
         "NumberOfMountTargets":1,
         "SizeInBytes":{
            "Value": 29313746702,
            "ValueInIA": 675432,
            "ValueInStandard": 29312741784,
            "ValueInArchive":329486
        },
        "ThroughputMode": "elastic"
      }
   ]
}
```

For additional ways to view and measure disk usage, see Metering EFS file system objects.

Monitoring metrics with Amazon CloudWatch

You can monitor file systems using Amazon CloudWatch, which collects and processes raw data from Amazon EFS into readable, near real-time metrics. These statistics are recorded for a period of 15 months, so that you can gain a better perspective on how your web application or service is performing.

By default, Amazon EFS metric data is automatically sent to CloudWatch at 1-minute periods, unless noted for some individual metrics. The Amazon EFS console displays a series of graphs based on the raw data from Amazon CloudWatch. Depending on your needs, you might prefer to get data for your file systems from CloudWatch instead of the graphs in the console.

For more information about Amazon CloudWatch, see What is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*.

Amazon EFS CloudWatch metrics are reported as raw *bytes*. Bytes are not rounded to either a decimal or binary multiple of the unit.

Topics

- CloudWatch metrics for Amazon EFS
- Accessing CloudWatch metrics for Amazon EFS
- Using CloudWatch metrics for Amazon EFS
- Using metric math with CloudWatch metrics
- Monitoring mount attempt successes and failures
- Creating CloudWatch alarms to monitor Amazon EFS

CloudWatch metrics for Amazon EFS

Amazon EFS metrics use the `EFS` namespace. The `AWS/EFS` namespace includes the following metrics. All metrics except for `TimeSinceLastSync` are for a single dimension, `FileSystemId`. A file system's ID can be found in the Amazon EFS console, and it takes the form of `fs-abcdef0123456789a`.

`TimeSinceLastSync`

Shows the amount of time that has passed since the last successful sync to the destination file system in a replication configuration. Any changes to data on the source file system that occurred before the `TimeSinceLastSync` value have been successfully replicated. Any changes on the source that occurred after `TimeSinceLastSync` might not be fully replicated.

This metric uses two dimensions:

- `FileSystemId` dimension – ID of the source file system in the replication configuration.
- `DestinationFileSystemId` dimension – ID of the destination file system in the replication configuration.

Units: Seconds

Valid statistics: `Minimum`, `Maximum`, `Average`

`PercentIOLimit`

Shows how close a file system is to reaching the I/O limit of the General Purpose performance mode.

Units: Percent

Valid statistics: `Minimum`, `Maximum`, `Average`

`BurstCreditBalance`

The number of burst credits that a file system has. Burst credits allow a file system to burst to throughput levels above a file system's baseline level for periods of time.

The `Minimum` statistic is the smallest burst credit balance for any minute during the period. The `Maximum` statistic is the largest burst credit balance for any minute during the period. The `Average` statistic is the average burst credit balance during the period.

Units: Bytes

Valid statistics: `Minimum`, `Maximum`, `Average`

`PermittedThroughput`

The maximum amount of throughput that a file system can drive.

- For file systems using Elastic throughput, this value reflects the maximum write throughput of the file system.
- For file systems using Provisioned throughput, if the amount of data stored in the EFS Standard storage class allows your file system to drive a higher throughput than you provisioned, this metric reflects the higher throughput instead of the provisioned amount.

- For file systems in Bursting throughput, this value is a function of the file system size and `BurstCreditBalance`.

The `Minimum` statistic is the smallest throughput permitted for any minute during the period. The `Maximum` statistic is the highest throughput permitted for any minute during the period. The `Average` statistic is the average throughput permitted during the period.

> **Note**
>
> Read operations are metered at one-third the rate of other operations.

Units: Bytes per second

Valid statistics: `Minimum`, `Maximum`, `Average`

`MeteredIOBytes`

The number of metered bytes for each file system operation, including data read, data write, and metadata operations, with read operations discounted according to the throughput limit.

You can create a [CloudWatch metric math expression]() that compares `MeteredIOBytes` to `PermittedThroughput`. If these values are equal, then you are consuming the entire amount of throughput allocated to your file system. In this situation, you might consider changing the file system's throughput mode to get higher throughput.

The `Sum` statistic is the total number of metered bytes associated with all file system operations. The `Minimum` statistic is the size of the smallest operation during the period. The `Maximum` statistic is the size of the largest operation during the period. The `Average` statistic is the average size of an operation during the period. The `SampleCount` statistic provides a count of all operations.

Units:

- Bytes for `Minimum`, `Maximum`, `Average`, and `Sum` statistics.
- Count for `SampleCount`.

Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`

TotalIOBytes

The actual number of bytes for each file system operation processed by Amazon EFS, without any read discounts. This number may differ from the actual amount requested by your applications because it includes minimums. This number may also be higher than the numbers shown in `PermittedThroughput`.

Data operations are metered at 32 KiB and other operations are metered at 4 KiB. After the minimum, all operations are metered per KiB.

The `Sum` statistic is the total number of bytes associated with all file system operations. The `Minimum` statistic is the size of the smallest operation during the period. The `Maximum` statistic is the size of the largest operation during the period. The `Average` statistic is the average size of an operation during the period. The `SampleCount` statistic provides a count of all operations.

> **Note**
>
> To calculate the average operations per second for a period, divide the `SampleCount` statistic by the number of seconds in the period. To calculate the average throughput (bytes per second) for a period, divide the `Sum` statistic by the number of seconds in the period.

Units:

- Bytes for `Minimum`, `Maximum`, `Average`, and `Sum` statistics.
- Count for `SampleCount`.

Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`

`DataReadIOBytes`

The actual number of bytes for each file system read operation.

The `Sum` statistic is the total number of bytes associated with read operations. The `Minimum` statistic is the size of the smallest read operation during the period. The `Maximum` statistic is the size of the largest read operation during the period. The `Average` statistic is the average size of read operations during the period. The `SampleCount` statistic provides a count of read operations.

Units:

- Bytes for `Minimum`, `Maximum`, `Average`, and `Sum`.
- Count for `SampleCount`.

Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`

`DataWriteIOBytes`

The actual number of bytes for each file system write operation.

The `Sum` statistic is the total number of bytes associated with write operations. The `Minimum` statistic is the size of the smallest write operation during the period. The `Maximum` statistic is the size of the largest write operation during the period. The `Average` statistic is the average size of write operations during the period. The `SampleCount` statistic provides a count of write operations.

Units:

- Bytes are the units for the `Minimum`, `Maximum`, `Average`, and `Sum` statistics.
- Count for `SampleCount`.

Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`

`MetadataIOBytes`

The actual number of bytes for each metadata operation.

The `Sum` statistic is the total number of bytes associated with metadata operations. The `Minimum` statistic is the size of the smallest metadata operation during the period. The `Maximum` statistic is the size of the largest metadata operation during the period. The `Average` statistic is the size of the average metadata operation during the period. The `SampleCount` statistic provides a count of metadata operations.

Units:

- Bytes are the units for the `Minimum`, `Maximum`, `Average`, and `Sum` statistics.
- Count for `SampleCount`.

Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`

`MetadataReadIOBytes`

The actual number of bytes for each metadata read operation.

The `Sum` statistic is the total number of bytes associated with metadata read operations. The `Minimum` statistic is the size of the smallest metadata read operation during the period. The `Maximum` statistic is the size of the largest metadata read operation during the period. The

`Average` statistic is the average size of metadata read operations during the period. The `SampleCount` statistic provides a count of metadata read operations.

Units:

- Bytes are the units for the `Minimum`, `Maximum`, `Average`, and `Sum` statistics.
- Count for `SampleCount`.

Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`

`MetadataWriteIOBytes`

The actual number of bytes for each metadata write operation.

The `Sum` statistic is the total number of bytes associated with metadata write operations. The `Minimum` statistic is the size of the smallest metadata write operation during the period. The `Maximum` statistic is the size of the largest metadata write operation during the period. The `Average` statistic is the average size of metadata write operations during the period. The `SampleCount` statistic provides a count of metadata write operations.

Units:

- Bytes are the units for the `Minimum`, `Maximum`, `Average`, and `Sum` statistics.
- Count for `SampleCount`.

Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`

`ClientConnections`

The number of client connections to a file system. When using a standard client, there is one connection per mounted Amazon EC2 instance.

> **Note**
>
> To calculate the average `ClientConnections` for periods greater than one minute, divide the `Sum` statistic by the number of minutes in the period.

Units: Count of client connections

Valid statistics: `Sum`

`StorageBytes`

The size of the file system in bytes, including the amount of data stored in the EFS storage classes. This metric is emitted to CloudWatch every 15 minutes.

The `StorageBytes` metric has the following dimensions:

- `Total` is the metered size (in bytes) of data stored in the file system, in all storage classes. For EFS Infrequent Access (IA) and EFS Archive storage classes, files smaller than 128KiB are rounded to 128KiB.
- `Standard` is the metered size (in bytes) of data stored in the EFS Standard storage class.
- `IA` is the actual size (in bytes) of data stored in the EFS Infrequent Access storage class.
- `IASizeOverhead` is the difference (in bytes) between the actual size of data in the EFS Infrequent Access storage class (indicated in the `IA` dimension) and the metered size of the storage class, after rounding small files to 128KiB.
- `Archive` is the actual size (in bytes) of data stored in the EFS Archive storage class.
- `ArchiveSizeOverhead` is the difference (in bytes) between the actual size of data in the EFS Archive storage class (indicated in the `Archive` dimension) and the metered size of the storage class, after rounding small files to 128KiB.

Units: Bytes

Valid statistics: `Minimum`, `Maximum`, `Average`

> **Note**
>
> `StorageBytes` is displayed on the Amazon EFS console **File system metrics** page using base 1024 units (kibibytes, mebibytes, gibibytes, and tebibytes).

Accessing CloudWatch metrics for Amazon EFS

You can view Amazon EFS metrics for CloudWatch in several ways:

- In the Amazon EFS console
- In the CloudWatch console
- Using the CloudWatch CLI

- Using the CloudWatch API

To view CloudWatch metrics and alarms (Amazon EFS console)

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. Choose **File systems**.
3. Choose the file system that you want to view CloudWatch metrics for.
4. Choose **Monitoring** to display the **File system metrics** page.

   The **File system metrics** page displays a default set of CloudWatch metrics for the file system. Any CloudWatch alarms that you have configured also display with these metrics. For file systems that use Max I/O performance mode, the default set of metrics includes Burst Credit balance in place of Percent IO limit. You can override the default settings using the **Metrics settings** dialog box, accessed by opening the settings.

   > **Note**
   >
   > The Throughput utilization (%) metric is not a CloudWatch metric; it is derived using CloudWatch metric math.

5. You can adjust the way metrics and alarms are displayed using the controls on the **File system metric** page, as follows.

   - Toggle the **Display mode** between **Time series** or **Single value**.
   - Show or hide any CloudWatch alarms configured for the file system.
   - Choose **See more in CloudWatch** to view the metrics in CloudWatch.
   - Choose **Add to dashboard** to open your CloudWatch dashboard and add the displayed metrics.
   - Adjust the metric time window displayed from 1 hour to 1 week.

To view CloudWatch metrics and alarms (CloudWatch console)

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select the **EFS** namespace.

4. (Optional) To view a metric, enter its name in the search field.
5. (Optional) To filter by dimension, select **FileSystemId**.

To access metrics from the AWS CLI

- Use the `list-metrics` command with the `--namespace "AWS/EFS"` namespace. For more information, see the AWS CLI Command Reference.

To access metrics from the CloudWatch API

- Call `GetMetricStatistics`. For more information, see the Amazon CloudWatch API Reference.

Using CloudWatch metrics for Amazon EFS

The metrics reported by Amazon EFS provide information that you can analyze in different ways. The following list shows some common uses for the metrics. These are suggestions to get you started, not a comprehensive list.

How do I?	Relevant metrics
How can I determine my throughput?	You can monitor the daily `Sum` statistic of the `TotalIOBytes` metric to see your throughput.
How can I determine the total amount of data transferred?	You can monitor the daily `Sum` statistic of `MeteredIOBytes` to see your total data transferred.
How can I track the number of Amazon EC2 instances that are connected to a file system?	You can monitor the `Sum` statistic of the `ClientConnections` metric. To calculate the average `ClientConnections` for periods greater than one minute, divide the sum by the number of minutes in the period.
How can I see my burst credit balance?	You can see your balance by monitoring the `BurstCreditBalance` metric for your file system. For more information on bursting and burst credits, see Bursting throughput.

Monitoring throughput performance

The CloudWatch metrics for throughput monitoring—`TotalIOBytes`, `ReadIOBytes`, `WriteIOBytes`, and `MetadataIOBytes`— represent the actual throughput that you are driving on your file system. The metric `MeteredIOBytes` represents the calculation of the overall metered throughput that you are driving. You can use the **Throughput utilization (%)** graph in the Amazon EFS console **Monitoring** section to monitor your throughput utilization. If you use custom CloudWatch dashboards or another monitoring tool, you can create a CloudWatch metric math expression that compares `MeteredIOBytes` to `PermittedThroughput`.

`PermittedThroughput` measures the amount of allowed throughput for the file system. This value is based on one of the following methods:

- For file systems in Elastic throughput, this value reflects the maximum write throughput of the file system.
- For file systems using Provisioned throughput, if the amount of data stored in the EFS Standard storage class allows your file system to drive a higher throughput than you provisioned, this metric reflects the higher throughput instead of the provisioned amount.
- For file systems using Bursting throughput, this value is a function of the file system size and `BurstCreditBalance`. Monitor `BurstCreditBalance` to ensure that your file system is operating at its burst rate rather than its base rate. If the balance is consistently at or near zero, consider switching to Elastic throughput or Provisioned throughput to get additional throughput.

When the values for `MeteredIOBytes` and `PermittedThroughput` are equal, your file system is consuming all available throughput. For file systems using Provisioned throughput, you can provision additional throughput.

Using metric math with CloudWatch metrics

Using metric math, you can query multiple Amazon CloudWatch metrics and use math expressions to create new time series based on these metrics. You can visualize the resulting time series in the CloudWatch console and add them to dashboards. For example, you can use Amazon EFS metrics to take the sample count of `DataRead` operations divided by 60. The result is the average number of reads per second on your file system for a given 1-minute period. For more information on metric math, see Using math expressions with CloudWatch metrics in the *Amazon CloudWatch User Guide.*

Following, find some useful metric math expressions for Amazon EFS.

Topics

- Metric math: Throughput in MiBps
- Metric math: Percent throughput
- Metric math: Percentage of permitted throughput utilization
- Metric math: Throughput IOPS
- Metric math: Percentage of IOPS
- Metric math: Average I/O size in KiB
- Using metric math through an AWS CloudFormation template for Amazon EFS

Metric math: Throughput in MiBps

To calculate the average throughput (in MiBps) for a time period, first choose a sum statistic (`DataReadIOBytes`, `DataWriteIOBytes`, `MetadataIOBytes`, or `TotalIOBytes`). Then convert the value to MiB, and divide that by the number of seconds in the period.

Suppose that your example logic is this: (sum of `TotalIOBytes` ÷ 1,048,576 (to convert to MiB)) ÷ seconds in the period

Then your CloudWatch metric information is the following.

ID	Usable metrics	Statistic	Period
m1	• `DataReadIOBytes` • `DataWriteIOBytes` • `MetadataIOBytes` • `TotalIOBytes`	sum	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	`(m1/1048576)/PERIOD(m1)`

Metric math: Percent throughput

This metric math expression calculates the percent of overall throughput used for the different I/O types—for example, the percentage of total throughput that is driven by read requests. To calculate the percent of overall throughput used by one of the I/O types (`DataReadIOBytes`, `DataWriteIOBytes`, or `MetadataIOBytes`) for a time period, first multiply the respective sum statistic by 100. Then divide the result by the sum statistic of `TotalIOBytes` for the same period.

Suppose that your example logic is this: (sum of `DataReadIOBytes` x 100 (to convert to percentage)) ÷ sum of `TotalIOBytes`

Then your CloudWatch metric information is the following.

ID	Usable metric or metrics	Statistic	Period
m1	• `TotalIOBytes`	sum	1 minute
m2	• `DataReadIOBytes`	sum	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	`(m2*100)/m1`

Metric math: Percentage of permitted throughput utilization

To calculate the percentage of permitted throughput utilization (`MeteredIOBytes`) for a time period, first multiply the throughput in MiBps by 100. Then divide the result by the a average statistic of `PermittedThroughput` converted to MiB for the same period.

Suppose that your example logic is this: (metric math expression for throughput in MiBps x 100 (to convert to percentage)) ÷ (sum of `PermittedThroughput` ÷ 1,048,576 (to convert bytes to MiB))

Then your CloudWatch metric information is the following.

ID	Usable metric or metrics	Statistic	Period
m1	`MeteredIOBytes`	sum	1 minute
m2	`Permitted Throughput`	average	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	(m1/1048576)/PERIOD(m1)
e2	m2/1048576
e3	`((e1)*100)/(e2)`

Metric math: Throughput IOPS

To calculate the average operations per second (IOPS) for a time period, divide the sample count statistic (`DataReadIOBytes`, `DataWriteIOBytes`, `MetadataIOBytes`, or `TotalIOBytes`) by the number of seconds in the period.

Suppose that your example logic is this: sample count of `DataWriteIOBytes` ÷ seconds in the period

Then your CloudWatch metric information is the following.

ID	Usable metrics	Statistic	Period
m1	• `DataReadIOBytes` • `DataWriteIOBytes` • `MetadataIOBytes`	sample count	1 minute

ID	Usable metrics	Statistic	Period
	• `TotalIOBytes`		

Your metric math ID and expression are the following.

ID	Expression
e1	`m1/PERIOD(m1)`

Metric math: Percentage of IOPS

To calculate the percentage of IOPS per second of the different I/O types (`DataReadIOBytes`, `DataWriteIOBytes`, or `MetadataIOBytes`) for a time period, first multiply the respective sample count statistic by 100. Then divide that value by the sample count statistic of `TotalIOBytes` for the same period.

Suppose that your example logic is this: (sample count of `MetadataIOBytes` x 100 (to convert to percentage)) ÷ sample count of `TotalIOBytes`

Then your CloudWatch metric information is the following.

ID	Usable metrics	Statistic	Period
m1	• `TotalIOBytes`	sample count	1 minute
m2	• `DataReadIOBytes` • `DataWriteIOBytes` • `MetadataIOBytes`	sample count	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	(m2*100)/m1

Metric math: Average I/O size in KiB

To calculate the average I/O size (in KiB) for a period, divide the respective sum statistic for the `DataReadIOBytes`, `DataWriteIOBytes`, or `MetadataIOBytes` metric by the same sample count statistic of that metric.

Suppose that your example logic is this: (sum of `DataReadIOBytes` ÷ 1,024 (to convert to KiB)) ÷ sample count of `DataReadIOBytes`

Then your CloudWatch metric information is the following.

ID	Usable metrics	Statistic	Period
m1	• `DataReadIOBytes` • `DataWriteIOBytes` • `MetadataIOBytes`	sum	1 minute
m2	• `DataReadIOBytes` • `DataWriteIOBytes` • `MetadataIOBytes`	sample count	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	(m1/1024)/m2

Using metric math through an AWS CloudFormation template for Amazon EFS

You can also create metric math expressions through AWS CloudFormation templates. One such template is available for you to download and customize for use from the Amazon EFS tutorials on GitHub. For more information about using AWS CloudFormation templates, see Working with AWS CloudFormation templates in the *AWS CloudFormation User Guide.*

Monitoring mount attempt successes and failures

You can use Amazon CloudWatch Logs to monitor and report the success or failure of mount attempts for your EFS file systems remotely without having to log into the clients. Use the following procedure to configure your EC2 instance to use CloudWatch Logs to monitor the success or failure of its file system mount attempts.

To enable mount attempt success or failure notification in CloudWatch logs

1. Install `amazon-efs-utils` on the EC2 instance mounting the file system. For more information, see Automatically installing or updating Amazon EFS client using AWS Systems Manager or Manually installing the Amazon EFS client.
2. Install `botocore` on the EC2 instance that will mount the file system. For more information, see Installing and upgrading `botocore`.
3. Enable the CloudWatch Logs feature in `amazon-efs-utils`. When you use AWS Systems Manager to install and configure `amazon-efs-utils`, CloudWatch logging is automatically done for you. When you install the `amazon-efs-utils` package manually, you have to manually update the `/etc/amazon/efs/efs-utils.conf` configuration file by uncommenting the `# enabled = true` line in the `cloudwatch-log` section. Use one of the following commands to enable CloudWatch Logs manually.

   For Linux instances:

   ```
   sudo sed -i -e '/\[cloudwatch-log\]/{N;s/# enabled = true/enabled = true/}' /etc/
   amazon/efs/efs-utils.conf
   ```

For MacOS instances:

```
EFS_UTILS_VERSION= efs-utils-version
sudo sed -i -e '/\[cloudwatch-log\]/{N;s/# enabled = true/enabled = true/;}' /usr/
local/Cellar/amazon-efs-utils/${EFS_UTILS_VERSION}/libexec/etc/amazon/efs/efs-
utils.conf
```

For Mac2 instances:

```
EFS_UTILS_VERSION= efs-utils-version
sudo sed -i -e '/\[cloudwatch-log\]/{N;s/# enabled = true/enabled = true/;}' /opt/
homebrew/Cellar/amazon-efs-utils/${EFS_UTILS_VERSION}/libexec/etc/amazon/efs/efs-
utils.conf
```

4. Optionally, you can configure CloudWatch Logs group names and set the log retention days in the `efs-utils.conf` file. If you want to have separate log groups in CloudWatch for each mounted file system, add `/{fs_id}` to the end of the `log_group_name` field in `efs-utils.conf` file, as follows:

```
[cloudwatch-log]
log_group_name = /aws/efs/utils/{fs_id}
```

5. Attach the `AmazonElasticFileSystemsUtils` AWS managed policy to the IAM role that you have attached to the EC2 instance, or to the AWS credentials configured on your instance. You can use Systems Manager to do this, for more information, see Step 1: Configure an IAM instance profile with the required permissions.

The following are examples of mount attempt status log entries:

```
Successfully mounted fs-12345678.efs.us-east-1.amazonaws.com at /home/ec2-user/efs
Mount failed, Failed to resolve "fs-01234567.efs.us-east-1.amazonaws.com"
```

To view mount status in CloudWatch Logs

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Log groups** in the left-hand navigation bar.
3. Choose the **/aws/efs/utils** log group. You will see a log stream for each Amazon EC2 instance and EFS file system combination.

4. Choose a log stream to view specific log events including mount attempt success or failure status.

Creating CloudWatch alarms to monitor Amazon EFS

You can create a CloudWatch alarm that sends an Amazon SNS message when the alarm changes state. An alarm watches a single metric over a time period that you specify. The alarm then performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or Auto Scaling policy.

Alarms invoke actions for sustained state changes only. CloudWatch alarms don't invoke actions only because they are in a particular state; the state must have changed and been maintained for a specified number of periods.

One important use of CloudWatch alarms for Amazon EFS is to enforce encryption at rest for your file system. You can enable encryption at rest for an Amazon EFS file system when it's created. To enforce data encryption-at-rest policies for Amazon EFS file systems, you can use Amazon CloudWatch and AWS CloudTrail to detect the creation of a file system and verify that encryption at rest is enabled.

> **Note**
>
> Currently, you can't enforce encryption in transit.

The following procedures outline how to create alarms for Amazon EFS.

Using the console

To set alarms using the CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Create Alarm**. This launches the **Create Alarm Wizard**.
3. Choose **EFS Metrics** and scroll through the Amazon EFS metrics to locate the metric you want to place an alarm on. To display only the Amazon EFS metrics in this dialog box, search for the file system ID of your file system. Select the metric to create an alarm on, and choose **Next**.

4. Fill in the **Name**, **Description**, **Whenever** values for the metric.
5. If you want CloudWatch to send you an email when the alarm state is reached, in the **Whenever this alarm:** field, choose **State is ALARM**. In the **Send notification to:** field, choose an existing SNS topic. If you select **Create topic**, you can set the name and email addresses for a new email subscription list. This list is saved and appears in the field for future alarms.

> **Note**
>
> If you use **Create topic** to create a new Amazon SNS topic, the email addresses must be verified before they receive notifications. Emails are only sent when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, they do not receive a notification.

6. At this point, the **Alarm Preview** area gives you a chance to preview the alarm you're about to create. Choose **Create Alarm**.

Using the AWS CLI

To set an alarm using the AWS CLI

- Call `put-metric-alarm`. For more information, see the AWS CLI Command Reference.

Using the CloudWatch API

To set an alarm using the CloudWatch API

- Call `PutMetricAlarm`. For more information, see the Amazon CloudWatch API Reference.

Logging Amazon EFS API calls with AWS CloudTrail

Amazon EFS is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon EFS. CloudTrail captures all API calls for Amazon EFS as events, including calls from the Amazon EFS console and from code calls to Amazon EFS API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon EFS. 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 Amazon EFS, the IP address from which the request was made, who made the request, when it was made, and additional details.

For more information, see the What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

Amazon EFS information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon EFS, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Working with CloudTrail event history.

For an ongoing record of events in your AWS account, including events for Amazon EFS, 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 AWS 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 topics in the *AWS CloudTrail User Guide:*

- Creating a trail for your AWS account
- AWS service integrations with CloudTrail logs
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions and Receiving CloudTrail log files from multiple accounts

All Amazon EFS Amazon EFS API are logged by CloudTrail. For example, calls to the `CreateFileSystem`, `CreateMountTarget` and `CreateTags` operations 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 user or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.

- Whether the request was made by another AWS service.

For more information, see CloudTrail userIdentity element in the *AWS CloudTrail User Guide.*

Understanding Amazon EFS log file entries

A *trail* is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An *event* represents a single request from any source and includes information about the requested 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 calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateTags` operation when a tag for a file system is created from the console.


```
{
 "eventVersion": "1.06",
 "userIdentity": {
  "type": "Root",
  "principalId": "111122223333",
  "arn": "arn:aws:iam::111122223333:root",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2017-03-01T18:02:37Z"
   }
  }
 },
 "eventTime": "2017-03-01T19:25:47Z",
 "eventSource": "elasticfilesystem.amazonaws.com",
 "eventName": "CreateTags",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "console.amazonaws.com",
 "requestParameters": {
  "fileSystemId": "fs-00112233",
  "tags": [{
    "key": "TagName",
    "value": "AnotherNewTag"
   }
```

```
  ]
 },
 "responseElements": null,
 "requestID": "dEXAMPLE-feb4-11e6-85f0-736EXAMPLE75",
 "eventID": "eEXAMPLE-2d32-4619-bd00-657EXAMPLEe4",
 "eventType": "AwsApiCall",
 "apiVersion": "2015-02-01",
 "recipientAccountId": "111122223333"
}
```

The following example shows a CloudTrail log entry that demonstrates the `DeleteTags` action when a tag for a file system is deleted from the console.

```
{
 "eventVersion": "1.06",
 "userIdentity": {
  "type": "Root",
  "principalId": "111122223333",
  "arn": "arn:aws:iam::111122223333:root",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2017-03-01T18:02:37Z"
   }
  }
 },
 "eventTime": "2017-03-01T19:25:47Z",
 "eventSource": "elasticfilesystem.amazonaws.com",
 "eventName": "DeleteTags",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "console.amazonaws.com",
 "requestParameters": {
  "fileSystemId": "fs-00112233",
  "tagKeys": []
 },
 "responseElements": null,
 "requestID": "dEXAMPLE-feb4-11e6-85f0-736EXAMPLE75",
 "eventID": "eEXAMPLE-2d32-4619-bd00-657EXAMPLEe4",
 "eventType": "AwsApiCall",
```

```
 "apiVersion": "2015-02-01",
 "recipientAccountId": "111122223333"
}
```

Log entries for EFS service-linked roles

The Amazon EFS service-linked role makes API calls to AWS resources. You will see CloudTrail log entries with `username: AWSServiceRoleForAmazonElasticFileSystem` for calls made by the EFS service-linked role. For more information about EFS and service-linked roles, see Using service-linked roles for Amazon EFS.

The following example shows a CloudTrail log entry that demonstrates a `CreateServiceLinkedRole` action when Amazon EFS creates the AWSServiceRoleForAmazonElasticFileSystem service-linked role.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/user1",
        "accountId": "111122223333",
        "accessKeyId": "A111122223333",
        "userName": "user1",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-10-23T22:45:41Z"
            }
        },
        "invokedBy": "elasticfilesystem.amazonaws.com”
    },
    "eventTime": "2019-10-23T22:45:41Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "CreateServiceLinkedRole",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "user_agent",
    "requestParameters": {
        "aWSServiceName": "elasticfilesystem.amazonaws.com”
    },
```

```
    "responseElements": {
        "role": {
            "assumeRolePolicyDocument":
 "111122223333-10-111122223333Statement111122223333Action111122223333AssumeRole111122223333Effe
%22%3A%20%22Allow%22%2C%20%22Principal%22%3A%20%7B%22Service%22%3A%20%5B%22
 elasticfilesystem.amazonaws.com%22%5D%7D%7D%5D%7D",
            "arn": "arn:aws:iam::111122223333:role/aws-service-role/
elasticfilesystem.amazonaws.com/AWSServiceRoleForAmazonElasticFileSystem",
            "roleId": "111122223333",
            "createDate": "Oct 23, 2019 10:45:41 PM",
            "roleName": "AWSServiceRoleForAmazonElasticFileSystem",
            "path": "/aws-service-role/elasticfilesystem.amazonaws.com/“
        }
    },
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

The following example shows a CloudTrail log entry that demonstrates a `CreateNetworkInterface` action made by the `AWSServiceRoleForAmazonElasticFileSystem` service-linked role, noted in the `sessionContext`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:sts::0123456789ab:assumed-role/
AWSServiceRoleForAmazonElasticFileSystem/0123456789ab",
        "accountId": "0123456789ab",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::0123456789ab:role/aws-service-role/
elasticfilesystem.amazonaws.com/AWSServiceRoleForAmazonElasticFileSystem",
                "accountId": "0123456789ab",
                "userName": "AWSServiceRoleForAmazonElasticFileSystem"
            },
            "webIdFederationData": {},
```

```
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-10-23T22:50:05Z"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "20You 19-10-23T22:50:05Z",
    "eventSource": "ec2.amazonaws.com",
    "eventName": "CreateNetworkInterface",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "elasticfilesystem.amazonaws.com”,
    "userAgent": "elasticfilesystem.amazonaws.com",
    "requestParameters": {
        "subnetId": "subnet-71e2f83a",
        "description": "EFS mount target for fs-1234567 (fsmt-1234567)",
        "groupSet": {},
        "privateIpAddressesSet": {}
    },
    "responseElements": {
        "requestId": "0708e4ad-03f6-4802-b4ce-4ba987d94b8d",
        "networkInterface": {
            "networkInterfaceId": "eni-0123456789abcdef0",
            "subnetId": "subnet-12345678",
            "vpcId": "vpc-01234567",
            "availabilityZone": "us-east-1b",
            "description": "EFS mount target for fs-1234567 (fsmt-1234567)",
            "ownerId": "666051418590",
            "requesterId": "0123456789ab",
            "requesterManaged": true,
            "status": "pending",
            "macAddress": "00:bb:ee:ff:aa:cc",
            "privateIpAddress": "192.0.2.0",
            "privateDnsName": "ip-192-0-2-0.ec2.internal",
            "sourceDestCheck": true,
            "groupSet": {
                "items": [
                    {
                        "groupId": "sg-c16d65b6",
                        "groupName": "default"
                    }
                ]
            },
            "privateIpAddressesSet": {
```

```
                "item": [
                    {
                        "privateIpAddress": "192.0.2.0",
                        "primary": true
                    }
                ]
            },
            "tagSet": {}
        }
    },
    "requestID": "11112222-3333-4444-5555-666666777777",
    "eventID": "aaaabbbb-1111-2222-3333-444444555555",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


Log entries for EFS authentication

Amazon EFS authorization for NFS clients emits `NewClientConnection` and `UpdateClientConnection` CloudTrail events. A `NewClientConnection` event is emitted when a connection is authorized immediately after an initial connection, and immediately after a re-connection. An `UpdateClientConnection` is emitted when a connection is reauthorized and the list of permitted actions has changed. The event is also emitted when the new list of permitted actions doesn't include `ClientMount`. For more information about EFS authorization, see Using IAM to control access to file systems.

The following example shows a CloudTrail log entry that demonstrates a `NewClientConnection` event.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:sts::0123456789ab:assumed-role/abcdef0123456789",
        "accountId": "0123456789ab",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE ",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::0123456789ab:role/us-east-2",
```

```
                "accountId": "0123456789ab",
                "userName": "username"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-12-23T17:50:16Z"
            },
            "ec2RoleDelivery": "1.0"
        }
    },
    "eventTime": "2019-12-23T18:02:12Z",
    "eventSource": "elasticfilesystem.amazonaws.com",
    "eventName": "NewClientConnection",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "elasticfilesystem",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "27859ac9-053c-4112-aee3-f3429719d460",
    "readOnly": true,
    "resources": [
        {
            "accountId": "0123456789ab",
            "type": "AWS::EFS::FileSystem",
            "ARN": "arn:aws:elasticfilesystem:us-east-2:0123456789ab:file-system/
fs-01234567"
        },
        {
            "accountId": "0123456789ab",
            "type": "AWS::EFS::AccessPoint",
            "ARN": "arn:aws:elasticfilesystem:us-east-2:0123456789ab:access-point/
fsap-0123456789abcdef0"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "0123456789ab",
    "serviceEventDetails": {
        "permissions": {
            "ClientRootAccess": true,
            "ClientMount": true,
            "ClientWrite": true
        },
        "sourceIpAddress": "10.7.3.72"
```

```
    }
}
```

Amazon EFS log file entries for encrypted-at-rest file systems

Amazon EFS gives you the option of using encryption at rest, encryption in transit, or both, for your file systems. For more information, see Encrypting data in Amazon EFS.

Amazon EFS sends Encryption context when making AWS KMS API requests to generate data keys and decrypt Amazon EFS data. The file system ID is the encryption context for all file systems that are encrypted at rest. In the `requestParameters` field of a CloudTrail log entry, the encryption context looks similar to the following.

```
"EncryptionContextEquals": {}
"aws:elasticfilesystem:filesystem:id" : "fs-4EXAMPLE"
```

Using AWS billing and usage reports

When using Amazon EFS, you pay only for the resources you use and there are no minimum fees or setup charges. Like other AWS services, while pricing is quoted on a monthly basis, your usage is prorated by the hour, and you are billed for your average usage over a month.

AWS provides the following reports for Amazon EFS:

- **Billing reports –** Multiple reports that provide high-level views of all of the activity for the AWS services that you're using, including Amazon EFS. AWS always bills the owner of the EFS file system for Amazon EFS fees. For more information about billing reports, see AWS Billing reports for Amazon EFS.
- **Usage reports –** A summary of activity for a specific service, aggregated by hour, day, or month. You can choose which usage type and operation to include. You can also choose how the data is aggregated. For more information, see AWS usage reports for Amazon EFS.

  The following topics provide information about billing and usage reporting for Amazon EFS.

Topics

- Using cost allocation EFS file system tags
- AWS Billing reports for Amazon EFS
- AWS usage reports for Amazon EFS
- Understanding billing and usage reports for Amazon EFS

Using cost allocation EFS file system tags

To track the storage cost or other criteria for individual projects or groups of projects, label your EFS file systems using cost allocation tags. A *cost allocation tag* is a key-value pair that you associate with a file system. After you activate cost allocation tags, AWS uses the tags to organize your resource costs on your cost allocation report.

The *cost allocation report* lists the AWS usage for your account by product category and linked account user. The report contains the same line items as the detailed billing report (see Understanding billing and usage reports for Amazon EFS) and additional columns for your tag keys.

AWS provides two types of cost allocation tags, *AWS-generated tags* and *user-defined tags*. AWS defines, creates, and applies the AWS-generated `aws:createdBy` tag key for you after an Amazon EFS CreateFileSystem event. You define, create, and apply *user-defined* tags to your file systems.

You must activate both types of tags separately in the Billing and Cost Management console before they can appear in your billing reports. For more information about AWS-generated tags, see Organizing and tracking costs using AWS cost allocation tags in the *AWS Billing User Guide*.

- For more information about creating user-defined tags for EFS resources, see Tagging EFS resources.
- For more information about activating user-defined tags in the Billing and Cost Management console, see Activating user-defined cost allocation tags in the *AWS Billing User Guide*.
- For more information about activating AWS-generated tags in the Billing and Cost Management console, see Activating AWS-generated cost allocation tags in the *AWS Billing User Guide*.

User-defined cost allocation tags

A user-defined cost allocation tag has the following components:

- The tag key. The tag key is the name of the tag. For example, in the tag project/Trinity, project is the key. The tag key is a case-sensitive string that can contain 1 to 128 Unicode characters.

- The tag value. The tag value is a required string. For example, in the tag project/Trinity, Trinity is the value. The tag value is a case-sensitive string that can contain from 0 to 256 Unicode characters.

For more information about user-defined tags, see Using user-defined cost allocation tags in the *AWS Billing User Guide*.

AWS Billing reports for Amazon EFS

Your monthly bill from AWS separates your usage information and cost by AWS service and function. There are several AWS billing reports available: the monthly report, the cost allocation report, and detailed billing reports. For more information, see Understanding your bill in the *AWS Billing* User Guide.

For more detailed reports about your Amazon EFS storage usage, use AWS Data Exports to create exports of the AWS Cost and Usage Report (AWS CUR). With AWS CUR 2.0, you publish your AWS billing reports to an Amazon Simple Storage Service (Amazon S3) bucket that you own. The Amazon EFS usage report lists operations by usage type and AWS Region. For more information, see Cost and Usage Report (CUR) 2.0 in the *AWS Data Exports User Guide*.

EFS file systems are billed based on the following categories of usage.

- Storage (per gigabyte [GB]) – The amount of data stored in your EFS file systems per month, with different prices depending on the storage class (EFS Standard, EFS Infrequent Access (IA) and EFS Archive).
- Throughput (per GB or mebibytes per second [MiBps]) – The amount of data transferred (read/write operations) or the amount that exceeds your provisioned amount, depending on the file system's throughput, per month.
- Data access and transition (per MiBps) – The amount of data read and transferred to the EFS Infrequent Access (IA) and EFS Archive storage classes per month.
- Backup storage (per GB) – The amount of storage space consumed by backups per month. Increasing your backup retention period or taking additional user-initiated volume backups increases the amount of backup storage your file system consumes. For more information, see Backing up EFS file systems
- Replication (per GB) – The prevailing storage and read, write, and tiering activity rates for your source and destination EFS file systems, as well as any applicable charges for data transfer activity between them, per month.

For detailed information about Amazon EFS charges, see Amazon EFS Pricing.

For information about understanding the codes and abbreviations used in the billing and usage reports for Amazon EFS, see Understanding billing and usage reports for Amazon EFS.

AWS usage reports for Amazon EFS

When you download a usage report, you can choose to aggregate usage data by hour, day, or month. The Amazon EFS usage report lists operations by usage type and AWS Region. For more detailed reports about your Amazon EFS storage usage, download dynamically generated AWS usage reports. You can choose which usage type, operation, and time period to include. You can also choose how the data is aggregated. For more information about downloading usage reports, see Downloading an AWS Usage Report in the *AWS Data Exports User Guide*.

The Amazon EFS usage report includes the following information:

- **Service** – Amazon EFS
- **UsageType** – One of the following values:
  - A code that identifies the type of storage
  - A code that identifies the type of request
  - A code that identifies the type of data transfer
  - A code that identifies the throughput mode
  - A code that identifies the backup usage

  For a detailed explanation of Amazon EFS usage types, see Understanding billing and usage reports for Amazon EFS.
- **Resource** – The name of the resource associated with the listed usage.
- **StartTime** – Start time of the day that the usage applies to, in Coordinated Universal Time (UTC).
- **EndTime** – End time of the day that the usage applies to, in Coordinated Universal Time (UTC).
- **UsageValue** – One of the following volume values. The typical unit of measurement for data is gigabytes (GB). However, depending on the service and the report, terabytes (TB) might appear instead.
  - The number of requests during the specified time period
  - The amount of data transferred
  - The amount of data stored in a given hour

For information about understanding the codes and abbreviations used in the billing and usage reports for Amazon EFS, see Understanding billing and usage reports for Amazon EFS.

Understanding billing and usage reports for Amazon EFS

Amazon EFS billing and usage reports use codes and abbreviations. For usage types in the table that follows, replace *region* with abbreviations from this list:

- **APE1:** Asia Pacific (Hong Kong)
- **APN1:** Asia Pacific (Tokyo)
- **APN2:** Asia Pacific (Seoul)
- **APN3:** Asia Pacific (Osaka)

- **APS1:** Asia Pacific (Singapore)
- **APS2:** Asia Pacific (Sydney)
- **APS3:** Asia Pacific (Mumbai)
- **APS4:** Asia Pacific (Jakarta)
- **APS5:** Asia Pacific (Hyderabad)
- **APS6:** Asia Pacific (Melbourne)
- **CAN1:** Canada (Central)
- **CAN2:** Canada West (Calgary)
- **CNN1:** China (Beijing)
- **CNW1:** China (Ningxia)
- **AFS1:** Africa (Cape Town)
- **EUC2:** Europe (Zurich)
- **EUN1:** Europe (Stockholm)
- **EUS2:** Europe (Spain)
- **EUC1:** Europe (Frankfurt)
- **EU:** Europe (Ireland)
- **EUS1:** Europe (Milan)
- **EUW2:** Europe (London)
- **EUW3:** Europe (Paris)
- **ILC1:** Israel (Tel Aviv)
- **MEC1:** Middle East (UAE)
- **MES1:** Middle East (Bahrain)
- **SAE1:** South America (São Paulo)
- **UGW1:** AWS GovCloud (US-West)
- **UGE1:** AWS GovCloud (US-East)
- **USE1 (or no prefix):** US East (N. Virginia)
- **USE2:** US East (Ohio)
- **USW1:** US West (N. California)
- **USW2:** US West (Oregon)

For information about pricing by AWS Region, see [Amazon EFS Pricing](#).

The first column in the following table lists usage types that appear in your billing and usage reports. The typical unit of measurement for data is gigabytes (GB). For provisioned throughput amount, however, mebibytes per second (MiBps) is used instead.

Usage types for Amazon EFS

Usage type	CloudWa h metric	Units	Granula ty	Description
`*region*-ArchiveDataAccess-Bytes`	N/A	GB	N/A	The amount of data that was transferred to or accessed in EFS Archive storage.
`*region*-ArchiveEarlyDelete-ByteHrs`	N/A	GB-Mo	N/A	Prorated storage usage for files deleted from EFS Archive storage before the 90-day minimum commitment ended.
`*region*-ArchiveEarlyDelete-SmallFiles`	N/A	GB-Mo	N/A	Prorated storage usage for small files (smaller than 128 KB) that were deleted from EFS Archive storage before the 90-day minimum commitment ended.
`*region*-ArchiveTimedStorage-ByteHrs`	[StorageE tes.Archive](#)	GB-Mo	Hourly	The number of GB-months that data was stored in EFS Archive storage.
`*region*-ArchiveTimedStorage-SmallFiles`	[StorageE tes.ArchiveSizeOv rhead](#)	GB-Mo	Hourly	The number of GB-months that small objects (smaller than 128 KB) were stored in EFS Archive storage.

Usage type	CloudWatch metric	Units	Granularity	Description
region-ETDataAccess-Bytes	TotalIOBytes	GB	Hourly	The amount of data that was written or read for files using Elastic throughput.
region-IADataAccess-Bytes	N/A	GB	Hourly	The amount of data that was transferred to or accessed in EFS Infrequent Access (IA) storage.
region-IATimedStorage-ByteHrs	StorageBytes.IA	GB-Mo	Hourly	The number of GB-months that data was stored in EFS IA storage.
region-IATimedStorage-ET-ByteHrs	StorageBytes.IA	GB-Mo	Hourly	The number of GB-months that data for files using Elastic throughput was stored in EFS IA storage.
region-IATimedStorage-ET-SmallFiles	StorageBytes.IASizeOverhead	GB-Mo	Hourly	The number of GB-months that small files (smaller than 128 KB) were stored in EFS IA storage, for files using Elastic throughput.
region-IATimedStorage-SmallFiles	StorageBytes.IASizeOverhead	GB-Mo	Hourly	The number of GB-months that small files (smaller than 128 KB) were stored in EFS IA storage.
region-IATimedStorage-Z-ByteHrs	StorageBytes.IA	GB-Mo	Hourly	The actual size of data stored in EFS IA storage for One Zone file systems.

Usage type	CloudWatch metric	Units	Granularity	Description
region-IATimedStorage-Z-SmallFiles	StorageBytes.IASizeOverhead	GB-Mo	Hourly	The number of GB-months that small files (smaller than 128 KB) were stored in EFS IA storage for One Zone file systems.
region-IncludedTP-MiBpsHrs	PermittedThroughput	MiBps-Mo	Hourly	The maximum amount of throughput that a file system can drive.
region-ProvisionedTP-MiBpsHrs	PermittedThroughput	MiBps-Mo	Hourly	For file systems using Provisioned throughput, the throughput provisioned or maximum permitted throughput.
region-TimedStorage-ByteHrs	StorageBytes.Standard	GB-Mo	Hourly	The actual total amount of data stored in EFS Standard storage for regional file systems.
region-TimedStorage-Z-ByteHrs	StorageBytes.Standard	GB-Mo	Hourly	The actual total amount of data stored in EFS Standard storage for One Zone file systems.

Notes

- EFS Archive storage is charged for a minimum storage duration of 90 days, even if the data is deleted before 90 days.
- A tiering charge applies to data tiered from EFS Standard storage to EFS Infrequent Access (IA) storage, and from EFS IA storage to EFS Standard storage.
- EFS IA and EFS Archive storage have a minimum billable file size of 128 KiB. Files smaller than 128 KiB can be tiered to these storage classes but are charged for 128 KiB of storage at the appropriate storage class rate.

- For file systems using Elastic throughput, metadata operations are metered in 1 KiB increments after the first 4 KiB and data operations are metered in 1 KiB increments after the first 32 KiB.
- Storage usage is calculated in binary gigabytes (GB), where 1 GB is 2^{30} bytes. See How Amazon EFS reports file system and object sizes for additional details on how storage and throughput usage are metered to calculate your Amazon EFS bill.
- GB-Month is derived by taking the total number of GB-hours, aggregating these over the course of a month, and then dividing by the number of hours in that month.

Tracking operations in your usage reports

Operations describe the action taken on your EFS file system by the specified usage type. Operations are indicated by self-explanatory codes, such as `Read` or `Delete`. To see which actions on your file system generated a specific type of usage, use these codes. When you create a usage report, you can choose to include **All Operations**, or a specific operation, for example, `Write`, to report on.

Amazon EFS performance specifications

The following sections provide an overview of Amazon EFS performance, and describe how your file system configuration impacts key performance dimensions. We also provide some important tips and recommendations for optimizing the performance of your file system.

Topics

- Performance summary
- Storage classes
- Performance modes
- Throughput modes
- Amazon EFS performance tips
- Troubleshooting Amazon EFS performance issues
- Troubleshooting AMI and kernel issues

Performance summary

File system performance is typically measured by using the dimensions of latency, throughput, and Input/Output operations per second (IOPS). Amazon EFS performance across these dimensions depends on your file system's configuration. The following configurations impact the performance of an Amazon EFS file system:

- **File system type** – Regional or One Zone
- **Performance mode** – General Purpose or Max I/O

> **Important**
>
> Max I/O performance mode has higher per-operation latencies than General Purpose performance mode. For faster performance, we recommend always using General Purpose performance mode. For more information, see Performance modes.

- **Throughput mode** – Elastic, Provisioned, or Bursting

The following table outlines performance specifications for file systems using General Purpose performance mode and the possible different combinations of file system type and throughput mode.

Performance specifications for file systems using General Purpose performance mode

Storage and throughput configuration		Latency		Maximum IOPS		Maximum throughput		
File system type	**Throughput mode**	**Read operations**	**Write operations**	**Read operations**	**Write operations**	**Per-file-system read**[1]	**Per-file-system write**[1]	**Per-client read/write**
Regional	Elastic	As low as 250 microseconds (µs)	As low as 2.7 milliseconds (ms)	900,000–2,500,000[2]	500,000[2]	20–60 gibibytes per second (GiBps)	1–5 GiBps	1,500 mebibytes per second (MiBps)[3]
Regional	Provisioned	As low as 250 µs	As low as 2.7 ms	55,000	25,000	3–10 GiBps	1–3.33 GiBps	500 MiBps
Regional	Bursting	As low as 250 µs	As low as 2.7 ms	35,000	7,000	3–5 GiBps	1–3 GiBps	500 MiBps
One Zone	Elastic, Provisioned, Bursting	As low as 250 µs	As low as 1.6 ms	35,000	7,000	3 GiBps[4]	1 GiBps[4]	500 MiBps

1. Maximum read and write throughput depend on the AWS Region. Throughput in excess of an AWS Region's maximum throughput requires a throughput quota increase. Any request for additional throughput is considered on a case-by-case basis by the Amazon EFS service

team. Approval might depend on your type of workload. To learn more about requesting quota increases, see Amazon EFS quotas.

2. By default, file systems that use Elastic throughput drive a maximum of 90,000 read IOPS for infrequently accessed data, 250,000 read IOPS for frequently accessed data, and 50,000 write IOPS. If your workload requires more IOPS, then you can request an increase of up to 10 times these numbers. For more information, see Amazon EFS quotas that you can increase. Additional recommendations apply to achieve maximum IOPS. For more information, see the section called "Optimizing workloads that demand high throughput and IOPS".
3. The maximum combined read and write throughput is 1,500 MiBps for file systems using Elastic throughput and mounted using version 2.0 or later of the Amazon EFS client (amazon-efs-utils version) or the Amazon EFS CSI Driver (aws-efs-csi-driver). For all other file systems, the throughput limit is 500 MiBps. For more information about the Amazon EFS client, see Installing the Amazon EFS client.
4. One Zone file systems that use Bursting throughput can drive the same per-file-system read and write throughput amounts as Regional file systems using Bursting throughput (maximum read of 5 GiBps for read and 3 GiBps for write).

Storage classes

Amazon EFS storage classes are designed for the most effective storage depending on use cases.

- EFS Standard storage class uses solid state drive (SSD) storage to deliver the lowest levels of latency for frequently accessed files. This storage class provides first-byte latencies as low as 250 microseconds for reads and 2.7 milliseconds for writes.
- EFS Infrequent Access (IA) and EFS Archive storage classes store less frequently accessed data that doesn't require the latency performance that frequently accessed data requires. These storage classes provide first-byte latencies of tens of milliseconds.

For more information about EFS storage classes, see the section called "EFS storage classes".

Performance modes

Amazon EFS offers two performance modes, General Purpose and Max I/O.

- **General Purpose mode** has the lowest per-operation latency and is the default performance mode for file systems. One Zone file systems always use the General Purpose performance mode. For faster performance, we recommend always using General Purpose performance mode.
- **Max I/O mode** is a previous generation performance type that is designed for highly parallelized workloads that can tolerate higher latencies than the General Purpose mode. Max I/O mode is not supported for One Zone file systems or file systems that use Elastic throughput.

> **Important**
>
> Due to the higher per-operation latencies with Max I/O, we recommend using General Purpose performance mode for all file systems.

To help ensure that your workload stays within the IOPS limit available to file systems using General Purpose performance mode, you can monitor the `PercentIOLimit` CloudWatch metric. For more information, see CloudWatch metrics for Amazon EFS.

Applications can scale their IOPS elastically up to the limit associated with the performance mode. You are not billed separately for IOPS; they are included in a file system's throughput accounting. Every Network File System (NFS) request is accounted for as 4 kilobyte (KB) of throughput, or its actual request and response size, whichever is larger.

Throughput modes

A file system's throughput mode determines the throughput available to your file system. Amazon EFS offers three throughput modes: Elastic, Provisioned, and Bursting. Read throughput is discounted to allow you to drive higher read throughput than write throughput. The maximum throughput available with each throughput mode depends on the AWS Region. For more information about the maximum file system throughput in the different regions, see Amazon EFS quotas.

Your file system can achieve a combined 100% of its read and write throughput. For example, if your file system is using 33% of its read throughput limit, the file system can simultaneously achieve up to 67% of its write throughput limit. You can monitor your file system’s throughput usage in the **Throughput utilization (%)** graph on the on the **File System Detail** page of the console. For more information, see Monitoring throughput performance.

Choosing the correct throughput mode for a file system

Choosing the correct throughput mode for your file system depends on your workload's performance requirements.

- **Elastic throughput** (Recommended) – Use the default Elastic throughput when you have spiky or unpredictable workloads and performance requirements that are difficult to forecast, or when your application drives throughput at an average-to-peak ratio of 5% or less. For more information, see Elastic throughput.
- **Provisioned throughput** – Use Provisioned throughput if you know your workload's performance requirements, or when your application drives throughput at an average-to-peak ratio of 5% or more. For more information, see Provisioned throughput.
- **Bursting throughput** – Use Bursting throughput when you want throughput that scales with the amount of storage in your file system.

  If, after using Bursting throughput, you find that your application is throughput-constrained (for example, it uses more than 80% of the permitted throughput or you have used all of your burst credits), then you should use either Elastic or Provisioned throughput. For more information, see Bursting throughput.

For more information about Amazon EFS metrics, see CloudWatch metrics for Amazon EFS.

Elastic throughput

For file systems that are using Elastic throughput, Amazon EFS automatically scales throughput performance up or down to meet the needs of your workload activity. Elastic throughput is the best throughput mode for spiky or unpredictable workloads with performance requirements that are difficult to forecast, or for applications that drive throughput at 5% or less of the peak throughput on average (the average-to-peak ratio).

Because throughput performance for file systems with Elastic throughput scales automatically, you don't need to specify or provision the throughput capacity to meet your application needs. You pay only for the amount of metadata and data read or written, and you don't accrue or consume burst credits while using Elastic throughput.

> **Note**
>
> While Elastic throughput is designed to scale elastically with your throughput, we recommend implementing proper governance through monitoring metrics with CloudWatch (MeteredIOBytes) and usage alerts as part of your operational best practices. This helps you maintain optimal resource utilization and stay within your planned operational parameters. For more information, see Monitoring metrics with Amazon CloudWatch.

For information about per-Region Elastic throughput limits, see Amazon EFS quotas that you can increase.

Provisioned throughput

With Provisioned throughput, you specify a level of throughput that the file system can drive independent of the file system's size or burst credit balance. Use Provisioned throughput if you know your workload's performance requirements, or if your application drives throughput at 5% or more of the average-to-peak ratio.

For file systems using Provisioned throughput, you are charged for the amount of throughput enabled for the file system. The throughput amount billed in a month is based on the throughput provisioned in excess of your file system's included baseline throughput from Standard storage, up to the prevailing Bursting baseline throughput limits in the AWS Region.

If the file system's baseline throughput exceeds the Provisioned throughput amount, then it automatically uses the Bursting throughput allowed for the file system (up to the prevailing Bursting baseline throughput limits in that AWS Region).

For information about per-Region Provisioned throughput limits, see Amazon EFS quotas that you can increase.

Bursting throughput

Bursting throughput is recommended for workloads that require throughput that scales with the amount of storage in your file system. With Bursting throughput, the base throughput is proportionate to the file system's size in the Standard storage class, at a rate of 50 KiBps per each GiB of storage. Burst credits accrue when the file system consumes below its base throughput rate, and are deducted when throughput exceeds the base rate.

When burst credits are available, a file system can drive throughput up to 100 MiBps per TiB in Standard storage (50 KiBps per GiB), up to the AWS Region limit, with a minimum of 100 MiBps. If no burst credits are available, a file system can drive up to 50 MiBps per TiB of storage, with a minimum of 1 MiBps.

For information about per-Region Bursting throughput, see General resource quotas that cannot be changed.

Understanding Amazon EFS burst credits

With Bursting throughput, each file system earns burst credits over time at a baseline rate that is determined by the size of the file system that is stored in the EFS Standard storage class. The baseline rate is 50 MiBps per tebibyte [TiB] of storage (equivalent to 50 KiBps per GiB of storage). Amazon EFS meters read operations up to one-third the rate of write operations, permitting the file system to drive a baseline rate up to 150 KiBps per GiB of read throughput, or 50 KiBps per GiB of write throughput.

A file system can drive throughput at its baseline metered rate continuously. A file system accumulates burst credits whenever it is inactive or driving throughput below its baseline metered rate. Accumulated burst credits give the file system the ability to drive throughput above its baseline rate.

For example, a file system with 100 GiB of metered data in the Standard storage class has a baseline throughput of 5 MiBps. Over a 24-hour period of inactivity, the file system earns 432,000 MiB worth of credit (5 MiB × 86,400 seconds = 432,000 MiB), which can be used to burst at 100 MiBps for 72 minutes (432,000 MiB ÷ 100 MiBps = 72 minutes).

File systems larger than 1 TiB can always burst for up to 50 percent of the time if they are inactive for the remaining 50 percent of the time.

The following table provides examples of bursting behavior.

File system size	Burst throughput	Baseline throughput
100 GiB of metered data in Standard storage	• Burst to 300 (MiBps) read-only for up to 72 minutes per day, or • Burst to 100 MiBps write-only for up to 72 minutes per day	• Drive up to 15 MiBps read-only continuously • Drive up to 5 MiBps write-only continuously

File system size	Burst throughput	Baseline throughput
1 TiB of metered data in Standard storage	• Burst to 300 MiBps read-only for 12 hours per day, or • Burst to 100 MiBps write-only for 12 hours per day	• Drive 150 MiBps read-only continuously • Drive 50 MiBps write-only continuously
10 TiB of metered data in Standard storage	• Burst to 3 GiBps read-only for 12 hours per day, or • Burst to 1 GiBps write-only for 12 hours per day	• Drive 1.5 GiBps read-only continuously • Drive 500 MiBps write-only continuously
Generally, larger file systems	• Burst to 300 MiBps read-only per TiB of storage for 12 hours per day, or • Burst to 100 MiBps write-only per TiB of storage for 12 hours per day	• Drive 150 MiBps read-only per TiB of storage continuously • Drive 50 MiBps write-only per TiB of storage continuously

> **Note**
>
> Amazon EFS provides a metered throughput of 1 MiBps to all file systems, even if the baseline rate is lower.
> The file system size used to determine the baseline and burst rates is the `ValueInStandard` metered size available through the DescribeFileSystems API operation.
> File systems can earn credits up to a maximum credit balance of 2.1 TiB for file systems smaller than 1 TiB, or 2.1 TiB per TiB stored for file systems larger than 1 TiB. This behavior means that file systems can accumulate enough credits to burst for up to 12 hours continuously.

Restrictions on switching throughput and changing provisioned amount

You can switch an existing file system's throughput mode and change the throughput amount. However, after switching the throughput mode to Provisioned throughput or changing the Provisioned throughput amount, the following actions are restricted for a 24-hour period:

- Switching from Provisioned throughput mode to Elastic or Bursting throughput mode.
- Decreasing the Provisioned throughput amount.

Amazon EFS performance tips

When using Amazon EFS, keep the following performance tips in mind.

Average I/O size

The distributed nature of Amazon EFS enables high levels of availability, durability, and scalability. This distributed architecture results in a small latency overhead for each file operation. Because of this per-operation latency, overall throughput generally increases as the average I/O size increases, because the overhead is amortized over a larger amount of data.

Optimizing workloads that demand high throughput and IOPS

For workloads that require high throughput and IOPS, use Regional file systems configured with the General Purpose performance mode and Elastic throughput.

> **Note**
>
> To achieve the maximum read IOPS for frequently accessed data, the file system must use Elastic throughput.

To achieve the highest levels of performance, you must leverage parallelization by configuring your application or workload as follows.

1. Distribute the workload evenly across all clients and directories, with at least the same number of directories as the number of utilized clients.
2. Minimize contention by aligning individual threads to distinct datasets or files.

3. Distribute the workload across 10 or more NFS clients, with at least 64 threads per client in a single mount target.

Simultaneous connections

You can mount Amazon EFS file systems on up to thousands of Amazon EC2 and other AWS compute instances concurrently. You can drive higher throughput levels on your file system in aggregate across compute instances if you can parallelize your application across more instances.

Request model

If you enable asynchronous writes to your file system, pending write operations are buffered on the Amazon EC2 instance before they're written to Amazon EFS asynchronously. Asynchronous writes typically have lower latencies. When performing asynchronous writes, the kernel uses additional memory for caching.

A file system that has enabled synchronous writes, or one that opens files using an option that bypasses the cache (for example, `O_DIRECT`), issues synchronous requests to Amazon EFS. Every operation goes through a round trip between the client and Amazon EFS.

> **Note**
>
> Your chosen request model has tradeoffs in consistency (if you're using multiple Amazon EC2 instances) and speed. Using synchronous writes provides increased data consistency by completing each write request transaction before processing the next request. Using asynchronous writes provides increased throughput by buffering pending write operations.

NFS client mount settings

Verify that you're using the recommended mount options as outlined in Mounting EFS file systems and in Mounting considerations for Linux.

When mounting your file systems on Amazon EC2 instances, Amazon EFS supports the Network File System version 4.0 and 4.1 (NFSv4) protocols. NFSv4.1 provides better performance for parallel small-file read operations (greater than 10,000 files per second) compared to NFSv4.0 (less than 1,000 files per second). For Amazon EC2 macOS instances running macOS Big Sur, only NFSv4.0 is supported.

Don't use the following mount options:

- `noac`, `actimeo=0`, `acregmax=0`, `acdirmax=0` – These options disable the attribute cache, which has a very large performance impact.
- `lookupcache=pos`, `lookupcache=none` – These options disable the file name lookup cache, which has a very large impact on performance.
- `fsc` – This option enables local file caching, but does not change NFS cache coherency, and does not reduce latencies.

> **Note**
>
> When you mount your file system, consider increasing the size of the read and write buffers for your NFS client to 1 MB.

Optimizing small-file performance

You can improve small-file performance by minimizing file reopens, increasing parallelism, and bundling reference files where possible.

- Minimize the number of round trips to the server.

  Don't unnecessarily close files if you will need them later in a workflow. Keeping file descriptors open enables direct access to the local copy in the cache. File open, close, and metadata operations generally cannot be made asynchronously or through a pipeline.

  When reading or writing small files, the two additional round trips are significant.

  Each round trip (file open, file close) can take as much time as reading or writing megabytes of bulk data. It's more efficient to open an input or output file once, at the beginning of your compute job, and hold it open for the entire length of the job.
- Use parallelism to reduce the impact of round-trip times.
- Bundle reference files in a `.zip` file. Some applications use a large set of small, mostly read-only reference files. Bundling these in a `.zip` file allows you to read many files with one open-close round trip.

  The `.zip` format allows for random access to individual files.

Optimizing directory performance

When performing a listing (**ls**) on very large directories (over 100k files) that are being modified concurrently, Linux NFS clients can hang, not returning a response. This issue is fixed in kernel 5.11, which has been ported to Amazon Linux 2 kernels 4.14, 5.4, and 5.10.

We recommend keeping the number of directories on your file system to less than 10,000, if possible. Use nested subdirectories as much as possible.

When listing a directory, avoid getting file attributes if they are not required, because they are not stored in the directory itself.

Optimizing the NFS read_ahead_kb size

The NFS `read_ahead_kb` attribute defines the number of kilobytes for the Linux kernel to read ahead or prefetch during a sequential read operation.

For Linux kernel versions prior to 5.4.*, the `read_ahead_kb` value is set by multiplying `NFS_MAX_READAHEAD` by the value for `rsize` (the client configured read buffer size set in the mount options). When using the recommended mount options, this formula sets `read_ahead_kb` to 15 MB.

> **Note**
>
> Starting with Linux kernel versions 5.4.*, the Linux NFS client uses a default `read_ahead_kb` value of 128 KB. We recommend increasing this value to 15 MB.

The Amazon EFS mount helper that is available in `amazon-efs-utils` version 1.33.2 and later automatically modifies the `read_ahead_kb` value to equal 15 * `rsize`, or 15 MB, after mounting the file system.

For Linux kernels 5.4 or later, if you do not use the mount helper to mount your file systems, consider manually setting `read_ahead_kb` to 15 MB for improved performance. After mounting the file system, you can reset the `read_ahead_kb` value by using the following command. Before using this command, replace the following values:

- Replace *`read-ahead-value-kb`* with the desired size in kilobytes.
- Replace *`efs-mount-point`* with the file system's mount point.

```
device_number=$(stat -c '%d' efs-mount-point)
((major = ($device_number & 0xFFF00) >> 8))
((minor = ($device_number & 0xFF) | (($device_number >> 12) & 0xFFF00)))
sudo bash -c "echo read-ahead-value-kb > /sys/class/bdi/$major:$minor/read_ahead_kb"
```

The following example sets the `read_ahead_kb` size to 15 MB.

```
device_number=$(stat -c '%d' efs)
((major = ($device_number & 0xFFF00) >> 8))
((minor = ($device_number & 0xFF) | (($device_number >> 12) & 0xFFF00)))
sudo bash -c "echo 15000 > /sys/class/bdi/$major:$minor/read_ahead_kb"
```

Troubleshooting Amazon EFS performance issues

In general, if you encounter issues with Amazon EFS that you have trouble resolving, confirm that you're using a recent Linux kernel. If you are using an enterprise Linux distribution, we recommend the following:

- Amazon Linux 2 with kernel 4.3 or newer
- Amazon Linux 2015.09 or newer
- RHEL 7.3 or newer
- All versions of Ubuntu 16.04
- Ubuntu 14.04 with kernel 3.13.0-83 or newer
- SLES 12 Sp2 or later

If you are using another distribution or a custom kernel, we recommend kernel version 4.3 or newer.

> **Note**
>
> RHEL 6.9 might be suboptimal for certain workloads due to Poor performance when opening many files in parallel.

Topics

- Unable to create an EFS file system

- Access denied to allowed files on NFS file system
- Errors when accessing the Amazon EFS console
- Amazon EC2 instance hangs
- Application writing large amounts of data hangs
- Poor performance when opening many files in parallel
- Custom NFS settings causing write delays
- Creating backups with Oracle Recovery Manager is slow

Unable to create an EFS file system

A request to create an EFS file system fails with the following message:

```
User: arn:aws:iam::111122223333:user/username is not authorized to
perform: elasticfilesystem:CreateFileSystem on the specified resource.
```

Action to take

Check your AWS Identity and Access Management (IAM) policy to confirm that you are authorized to create EFS file systems with the specified resource conditions. For more information, see Identity and access management for Amazon EFS.

Access denied to allowed files on NFS file system

When a user who is assigned more than 16 access group IDs (GIDs) attempts to perform an operation on an NFS file system, they could be denied access to allowed files on the file system. This issue occurs because the NFS protocol supports a maximum of 16 GIDs per user, and any additional GIDs are truncated from the NFS client request, as defined in RFC 5531.

Action to take

Restructure your NFS user and group mappings so that each user is assigned no more than 16 access groups (GIDs).

Errors when accessing the Amazon EFS console

This section describes errors users might experience when accessing the Amazon EFS management console.

Error authenticating credentials for `ec2:DescribeVPCs`

The following error message displays when accessing the Amazon EFS console:

```
AuthFailure: An error occurred authenticating your credentials for ec2:DescribeVPCs.
```

This error indicates that your login credentials did not successfully authenticate with the Amazon EC2 service. The Amazon EFS console calls the Amazon EC2 service on your behalf when creating EFS file systems in the VPC that you choose.

Action to take

Ensure that the time on the client accessing the Amazon EFS console is set correctly.

Amazon EC2 instance hangs

An Amazon EC2 instance can hang because you deleted a file system mount target without first unmounting the file system.

Action to take

Before you delete a file system mount target, unmount the file system. For more information about unmounting your Amazon EFS file system, see Unmounting file systems.

Application writing large amounts of data hangs

An application that writes a large amount of data to Amazon EFS hangs and causes the instance to reboot.

Action to take

If an application takes too long to write all of its data to Amazon EFS, Linux might reboot because it appears that the process has become unresponsive. Two kernel configuration parameters define this behavior, `kernel.hung_task_panic` and `kernel.hung_task_timeout_secs`.

In the example following, the state of the hung process is reported by the `ps` command with `D` before the instance reboot, indicating that the process is waiting on I/O.

```
$ ps aux | grep large_io.py
root 33253 0.5 0.0 126652 5020 pts/3 D+ 18:22 0:00 python large_io.py
/efs/large_file
```

To prevent a reboot, increase the timeout period or disable kernel panics when a hung task is detected. The following command disables hung task kernel panics on most Linux systems.

```
$ sudo sysctl -w kernel.hung_task_panic=0
```

Poor performance when opening many files in parallel

Applications that open multiple files in parallel do not experience the expected increase in performance of I/O parallelization.

Action to take

This issue occurs on Network File System version 4 (NFSv4) clients and on RHEL 6 clients using NFSv4.1 because these NFS clients serialize NFS OPEN and CLOSE operations. Use NFS protocol version 4.1 and one of the suggested Linux distributions that does not have this issue.

If you can't use NFSv4.1, be aware that the Linux NFSv4.0 client serializes open and close requests by user ID and group IDs. This serialization happens even if multiple processes or multiple threads issue requests at the same time. The client only sends one open or close operation to an NFS server at a time, when all of the IDs match. To work around these issues, you can perform any of the following actions:

- You can run each process from a different user ID on the same Amazon EC2 instance.
- You can leave the user IDs the same across all open requests, and modify the set of group IDs instead.
- You can run each process from a separate Amazon EC2 instance.

Custom NFS settings causing write delays

You have custom NFS client settings, and it takes up to three seconds for an Amazon EC2 instance to see a write operation performed on a file system from another Amazon EC2 instance.

Action to take

If you encounter this issue, you can resolve it in one of the following ways:

- If the NFS client on the Amazon EC2 instance that's reading data has attribute caching activated, unmount your file system. Then remount it with the `noac` option to disable attribute caching. Attribute caching in NFSv4.1 is enabled by default.

> **Note**
>
> Disabling client-side caching can potentially reduce your application's performance.

- You can also clear your attribute cache on demand by using a programming language compatible with the NFS procedures. To do this, you can send an ACCESS procedure request immediately before a read request.

  For example, using the Python programming language, you can construct the following call.

```
# Does an NFS ACCESS procedure request to clear the attribute cache, given a path to
 the file
import os
os.access(path, os.W_OK)
```

Creating backups with Oracle Recovery Manager is slow

Creating backups with Oracle Recovery Manager can be slow if Oracle Recovery Manager pauses for 120 seconds before starting a backup job.

Action to take

If you encounter this issue, disable Oracle Direct NFS, as described in Enabling and Disabling Direct NFS Client Control of NFS in the Oracle Help Center.

> **Note**
>
> Amazon EFS doesn't support Oracle Direct NFS.

Troubleshooting AMI and kernel issues

Following, you can find information about troubleshooting issues related to certain Amazon Machine Image (AMI) or kernel versions when using Amazon EFS from an Amazon EC2 instance.

Topics

- Unable to chown

- File system keeps performing operations repeatedly due to client bug
- Deadlocked client
- Listing files in a large directory takes a long time

Unable to chown

You're unable to change the ownership of a file/directory using the Linux `chown` command.

Kernel versions with this bug

2.6.32

Action to take

You can resolve this error by doing the following:

- If you're performing `chown` for the one-time setup step necessary to change ownership of the EFS root directory, you can run the `chown` command from an instance running a newer kernel. For example, use the newest version of Amazon Linux.
- If `chown` is part of your production work flow, you must update the kernel version to use `chown`.

File system keeps performing operations repeatedly due to client bug

A file system gets stuck performing repeated operations due to a client bug.

Action to take

Update the client software to the latest version.

Deadlocked client

A client becomes deadlocked.

Kernel versions with this bug

- CentOS-7 with kernel Linux 3.10.0-229.20.1.el7.x86_64
- Ubuntu 15.10 with kernel Linux 4.2.0-18-generic

Action to take

Do one of the following:

- Upgrade to a newer kernel version. For CentOS-7, kernel version **Linux 3.10.0-327** or later contains the fix.
- Downgrade to an older kernel version.

Listing files in a large directory takes a long time

This can happen if the directory is changing while your NFS client iterates through the directory to finish the list operation. Whenever the NFS client notices that the contents of the directory changed during this iteration, the NFS client restarts iterating from the beginning. As a result, the ls command can take a long time to complete for a large directory with frequently changing files.

Kernel versions with this bug

CentOS and RHEL kernel versions lower than 2.6.32-696.el6

Action to take

To resolve this issue, upgrade to a newer kernel version.

Backing up and replicating data in Amazon EFS

To protect your data, Amazon EFS automatically backs up your EFS file systems. For further resilience and data protection, you can replicate your EFS file system in an AWS Region. Backing up and replicating your EFS file systems ensure that you can provide continued operations or services in the event something happens to the EFS file system data. For example, in the case of data corruption or data loss.

Topics

- Backing up EFS file systems
- Replicating EFS file systems

Backing up EFS file systems

Amazon EFS is natively integrated with AWS Backup, a fully managed, policy-based service that you can use to create and manage backup policies to protect your data in Amazon EFS. File systems that you create using the Amazon EFS console are automatically backed up by AWS Backup by default. When you use the CLI or API to create a file system, automatic backups are enabled by default only for One Zone file systems.

Using AWS Backup for Amazon EFS, you can perform the following actions:

- Manage automatic backup scheduling and retention by configuring backup plans. You specify the backup frequency, when to back up, how long to retain backups, and a lifecycle policy for backups.
- Restore backups of Amazon EFS data. You can restore file system data to either a new or existing file system. You also can choose whether to perform a full restore or an item-level restore.

For more information about using AWS Backup, see Getting started with AWS Backup in the *AWS Backup Developer Guide*.

Topics

- How AWS Backup works with Amazon EFS
- Required IAM permissions
- Backup performance

- Managing automatic backups of EFS file systems

How AWS Backup works with Amazon EFS

File systems that you create using the Amazon EFS console are automatically backed up by AWS Backup by default. When you use the CLI or API to create a file system, automatic backups are enabled by default only for One Zone file systems. You can turn on automatic backups after creating your EFS file system using the AWS CLI or API. The default EFS backup plan uses the AWS Backup recommended settings for automatic backups—daily backups with a 35-day retention period. The backups created using the default EFS backup plan are stored in a default EFS backup vault, which is also created by Amazon EFS on your behalf. The default backup plan and backup vault cannot be deleted.

All data in an EFS file system is backed up, whatever storage class the data is in. You don't incur data access charges when backing up an EFS file system that has lifecycle management enabled and has data in the Infrequent Access (IA) or Archive storage class. When you restore a recovery point, all files are restored to the Standard storage class.

Incremental backups

AWS Backup performs incremental backups of EFS file systems. During the initial backup, a copy of the entire file system is made. During subsequent backups of that file system, only files and directories that have been changed, added, or removed are copied. With each incremental backup, AWS Backup retains the necessary reference data to allow a full restore. This approach minimizes the time required to complete the backup and saves on storage costs by not duplicating data.

Backup consistency

Amazon EFS is designed to be highly available. You can access and modify your EFS file systems while your backup is occurring in AWS Backup. However, inconsistencies, such as duplicated, skewed, or excluded data, can occur if you make modifications to your file system while the backup is occurring. These modifications include write, rename, move, or delete operations. To ensure consistent backups, we recommend that you pause applications or processes that are modifying the file system for the duration of the backup process. Or, schedule your backups to occur during periods when the file system is not being modified.

Backup completion window

You can optionally specify a completion window for a backup. This window defines the period of time in which a backup needs to be completed. If you specify a completion window, make sure that you consider the expected performance and the size and makeup of your file system. Doing this helps make sure that your backup can be completed during the window.

Backups that aren't completed during the specified window are flagged with an incomplete status. During the next scheduled backup, AWS Backup resumes at the point that it left off. You can see the status of all of your backups on the AWS Backup Management Console.

On-demand backups

With AWS Backup, you can save a single resource to a backup vault on-demand. Unlike with scheduled backups, you don't need to create a backup plan to initiate an on-demand backup. You can still assign a lifecycle to your backup, which automatically moves the recovery point to the cold storage tier and notes when to delete it.

Additionally, AWS Backup automatically transitions data to cold storage only for data that no longer exists in the most recent warm backup. For example, your file system has 100 files when you create a backup and you delete two files the day after you created the backup (100 files - 2 files = 98 files on second day). When you transition the data to cold storage, only the two deleted files move to cold storage and the remaining 98 files are billed as warm storage.

Concurrent backups

AWS Backup limits backups to one concurrent backup per resource. Therefore, scheduled or on-demand backups might fail if a backup job is already in progress. For more information about AWS Backup limits, see AWS Backup quotas in the *AWS Backup Developer Guide*.

Backup deletions

The default EFS backup vault Access policy is set to deny deleting recovery points. To delete existing backups of your EFS file systems, you must change the vault access policy. If you attempt to delete an EFS recovery point without modifying the vault access policy, you receive the following error message:

```
"Access Denied: Insufficient privileges to perform this action. Please consult with the
 account administrator for necessary permissions."
```

To edit the default backup vault access policy, you must have permissions to edit policies. For more information, see Allow all IAM actions (admin access) in the *IAM User Guide*.

Required IAM permissions

AWS Backup creates a service-linked role on your behalf in your account. This role has the permissions required to perform Amazon EFS backups.

You can use the `elasticfilesystem:backup` and `elasticfilesystem:restore` actions to allow or deny an IAM entity (such as a user, group, or role) the ability to create or restore backups of an EFS file system. You can use these actions in a file system policy or in an identity-based IAM policy. For more information, see Identity and access management for Amazon EFS and Using IAM to control access to file systems.

Backup performance

In general, you can expect the following backup and restore rates with AWS Backup. The rates may be less for some workloads, such as those containing a large file or directory.

- Backup rate of 2,000 files per second or 400 megabytes per second (MBps), whichever is slower.
- Restore rate of 1,500 files per second or 200 MBps, whichever is slower.

The maximum duration for a backup operation in AWS Backup is 30 days.

Using AWS Backup doesn't consume accumulated burst credits, and it doesn't count against the General Purpose performance mode file operation limits. For more information, see Quotas for Amazon EFS file systems.

Managing automatic backups of EFS file systems

When you create a file system using the Amazon EFS console, automatic backups are turned on by default. You can turn on automatic backups after creating your file system using the AWS CLI or API.

You can edit the default backup plan settings using the AWS Backup console. For more information, see Managing backup plans in the *AWS Backup Developer Guide*. You can see all of your automatic backups, and edit the default EFS backup plan settings using the AWS Backup console.

Amazon EFS applies the `aws:elasticfilesystem:default-backup` system tag key with a value of `enabled` to EFS file systems when automatic backups are enabled.

After you create a file system, you can turn automatic backups on or off using the console, the AWS CLI, or the EFS API.

Using the console

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the **File systems** page, choose the file system that you want to turn automatic backups on or off for and display the **File system details** page.
3. Choose **Edit** in the **General** settings panel.
4. - To turn on automatic backups, select **Enable automatic backups**.
   - To turn off automatic backups, clear **Enable automatic backups**.
5. Choose **Save changes**.

Using the AWS CLI

- Use the `put-backup-policy` CLI command (the corresponding API operation is PutBackupPolicy) turn automatic backups on or off for an existing file system.
  - Use the following command to turn on automatic backups.

    ```
    $ aws efs put-backup-policy --file-system-id fs-01234567 \
    --backup-policy Status="ENABLED"
    ```

    Amazon EFS responds with the new backup policy.

    ```
    {
       "BackupPolicy": {
          "Status": "ENABLING"
       }
    }
    ```
  - Use the following command to turn off automatic backups.

    ```
    $ aws efs put-backup-policy --file-system-id fs-01234567 \
    --backup-policy Status="DISABLED"
    ```

Amazon EFS responds with the new backup policy.

```
{
   "BackupPolicy": {
      "Status": "DISABLING"
   }
}
```

Replicating EFS file systems

For expanded resilience and data protection, you can replicate your EFS file system in an AWS Region. When you enable replication on an EFS file system, Amazon EFS automatically and transparently replicates the data and metadata on the source file system to a destination file system. In the event of a disaster or when performing game day exercises, you can fail over to your replica file system. To resume operations, you can then fail back to the primary file system.

To manage the process of creating the destination file system and keeping it synced with the source file system, Amazon EFS uses a *replication configuration*.

After you create the replication configuration, Amazon EFS automatically keeps the source and destination file systems synchronized. Changes made to the source file system are not transferred to the destination file system in a point-in-time consistent manner. Instead they're transferred based on the **Last synced time** for the replication. The **Last sync time** indicates when the last successful sync between the source and destination was completed. Changes made to your source file system as of the last synced time are replicated to the destination file system, while changes made to the source file system after the last synced time may not be replicated. For more information, see [Viewing replication details](#).

Replication is available in all AWS Regions in which Amazon EFS is available. To replicate an EFS file system in a Region that is disabled by default, you must first opt in to the Region. For more information, see [Specify which AWS Regions your account can use](#) in the *AWS General Reference Guide*. If you opt out of a Region later, Amazon EFS pauses all replication activities for the Region. To resume replication activities for the Region, opt in to the AWS Region again.

> **Note**
>
> Replication does not support using tags for attribute-based access control (ABAC).

Topics

- Costs
- Replication performance
- Required IAM permissions
- Configuring replication to new EFS file system
- Configuring replication to an existing EFS file system
- Replicating EFS file systems across AWS accounts
- Viewing replication details
- Deleting replication configurations
- Using the replica

Costs

To facilitate replication, Amazon EFS creates hidden directories and metadata on the destination file system. These equate to approximately 12 mebibytes (MiB) of metered data for which you are billed. For more information about metering file system storage, see How Amazon EFS reports file system and object sizes.

Replication performance

When you create new replications or reverse the direction of existing replications during the failback process, Amazon EFS performs an initial sync, which includes a series of one-time setup actions to support the replication. The amount of time that the initial sync takes to finish depends on factors such as the size of the source file system and the number of files in it.

After the initial replication is finished, Amazon EFS maintains a Recovery Point Objective (RPO) of 15 minutes for most file systems. However, if the source file system has files that change very frequently and has either more than 100 million files or files that are larger than 100 GB, replication may take longer than 15 minutes. For information about monitoring when the last replication successfully finished, see Viewing replication details.

You can monitor when the last successful sync occurred using the console, the AWS Command Line Interface (AWS CLI), the API, and Amazon CloudWatch. In CloudWatch, use the TimeSinceLastSync EFS metric. For more information, see Viewing replication details.

Required IAM permissions

Amazon EFS uses either the EFS service-linked role named `AWSServiceRoleForAmazonElasticFileSystem` or the IAM role that you specify to synchronize replication between the source and destination file systems. To provide an IAM role, the IAM user or role creating the replication configuration must have `iam:PassRole` permission. For more information, see Grant a user permissions to pass a role to an AWS service in the *AWS Identity and Access Management User Guide*.

- For more information about the `iam:CreateServiceLinkedRole`, see the example in Using service-linked roles for Amazon EFS.
- For more information about a custom IAM role – see Create an IAM role with a custom trust policy.

> **Note**
>
> If you are performing cross-account replication, then you must provide an IAM role when you create the replication configuration. Using the service-linked role is not permitted. For more information, see Replicating EFS file systems across AWS accounts.

The service-linked role or IAM role that you provide when creating the replication configuration must have the following permissions for replication.

- `elasticfilesystem:DescribeFileSystem`
- `elasticfilesystem:CreateFileSystem`
- `elasticfilesystem:CreateReplicationConfiguration`
- `elasticfilesystem:DeleteReplicationConfiguration`
- `elasticfilesystem:DescribeReplicationConfigurations`

You can use the `AmazonElasticFileSystemFullAccess` managed policy to automatically get all required EFS permissions. For more information, see AWS managed policy: AmazonElasticFileSystemFullAccess.

Configuring replication to new EFS file system

Amazon EFS automatically creates a new file system and copies the data and metadata on the source file system to a new read-only destination file system in the AWS Region that you choose. When you replicate to a new file system, you choose the file system type and the AWS Key Management Service (AWS KMS) key to use for encryption. Additionally, Amazon EFS does not create any mount targets when it creates the destination file system. After you create the replication configuration, you must create one or more mount targets to mount a destination file system.

> **Note**
>
> A file system can be part of only one replication configuration. You cannot use a destination file system as the source file system in another replication configuration.

- **File system type** – The file system type determines the availability and durability with which the Amazon EFS file system stores data within an AWS Region.
  - Choose **Regional** to create a file system that stores data and metadata redundantly across all Availability Zones within the AWS Region.
  - Choose **One Zone** to create a file system that stores data and metadata redundantly within a single Availability Zone.

  For more information about file system types, see EFS file system types.
- **Encryption** – All destination file systems are created with encryption at rest enabled. You can specify the AWS KMS key that is used to encrypt the destination file system. If you don't specify a KMS key, your service-managed KMS key for Amazon EFS is used.

  > **Important**
  >
  > After the destination file system is created, you cannot change the KMS key.

The destination file system is created with default settings based on your source file system. Additional settings can be changed after creation.

- **Automatic backups –** For destination file systems using One Zone storage, automatic backups are enabled by default. After the file system is created, you can change the automatic backup setting. For more information, see Managing automatic backups of EFS file systems.
- **Performance mode –** The destination file system's **Performance mode** matches that of the source file system, unless the destination file system uses One Zone storage. In that case, the **General Purpose** mode is used. The performance mode cannot be changed.
- **Throughput mode –** The destination file system's **Throughput mode** matches that of the source file system. After the file system is created, you can change the mode.

  If the source file system's throughput mode is **Provisioned**, then the destination file system's provisioned throughput amount matches that of the source file system, unless the source file's provisioned amount exceeds the limit for the destination file system's Region. If the source file system's provisioned amount exceeds the Region limit for the destination file system, then the destination file system's provisioned throughput amount is the Region limit. For more information, see Amazon EFS quotas that you can increase.

- **Lifecycle management –** Lifecycle management is not enabled on the destination file system. After the destination file system is created, you can enable it. For more information, see Managing storage lifecycle.

Step 1: Create the replication configuration

The first step in replicating to a new file system is to create the replication configuration.

Using the console

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. Open the file system that you want to replicate:
   a. In the left navigation pane, choose **File systems**.
   b. In the **File systems** list, choose the file system that you want to replicate. The file system that you choose cannot be a source or destination file system in an existing replication configuration.
3. Choose the **Replication** tab.
4. In the **Replication** section, choose **Create replication**.

5. In the **Replication settings** section, define the replication settings:

   a. For **Replication configuration**, choose whether to replicate to a new or existing file system.

   b. For **Destination AWS Region**, choose the AWS Region in which to replicate the file system.

6. In the **Destination file system settings** section, define the destination file system settings.

   a. For **File system type**, choose a storage option for the file system:

      - To create a file system that stores data redundantly across multiple geographically separated Availability Zones within an AWS Region, choose **Regional**.
      - To create a file system that stores data redundantly within a single Availability Zone in an AWS Region, choose **One Zone**, and then select the Availability Zone.

      For more information, see EFS file system types.

      > **Note**
      >
      > One Zone file systems are not available in all Availability Zones in the AWS Regions where Amazon EFS is available.

   b. For **Encryption**, encryption of data at rest is automatically enabled on the destination file system. By default, Amazon EFS uses your AWS Key Management Service (AWS KMS) service key (`aws/elasticfilesystem`). To use a different KMS key, choose the KMS key or enter the key's Amazon Resource Name (ARN).

      > **Important**
      >
      > After the file system is created, you cannot change the KMS key.

Create the replication configuration (AWS CLI)

This section provides examples for creating a replication configuration in the AWS CLI using the `create-replication-configuration` command. The equivalent API command is CreateReplicationConfiguration.

Example : Create a replication configuration for a Regional destination file system

The following example creates a replication configuration for the file system *fs-0123456789abcdef1*. The example uses the `Region` parameter to create a destination file system in the *eu-west-2* AWS Region. The `KmsKeyId` parameter specifies the KMS key ID to use when encrypting the destination file system:

```
aws efs create-replication-configuration \
--source-file-system-id fs-0123456789abcdef1 \
--destinations "[{\"Region\":\"eu-west-2\", \"KmsKeyId\":\"arn:aws:kms:us-
east-2:111122223333:key\/abcd1234-ef56-ab78-cd90-1111abcd2222\"}]"
```

The AWS CLI responds as follows:

```
         {
    "SourceFileSystemArn": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/fs-0123456789abcdef1",
    "SourceFileSystemRegion": "us-east-1",
    "Destinations": [
        {
            "Status": "ENABLING",
            "FileSystemId": "fs-0123456789abcde22",
            "Region": "eu-west-2"
        }
    ],
    "SourceFileSystemId": "fs-0123456789abcdef1",
    "CreationTime": 1641491892.0,
    "OriginalSourceFileSystemArn": "arn:aws:elasticfilesystem:us-
east-1:111122223333:file-system/fs-0123456789abcdef1"
}
```

Example : Create a replication configuration for a One Zone destination file system

The following example creates a replication configuration for the file system *fs-0123456789abcdef1*. The example uses the `AvailabilityZoneName` parameter to create a One Zone destination file system in the *us-west-2a* Availability Zone. Because no KMS key is specified, the destination file system is encrypted using the account's default AWS KMS service key (`aws/elasticfilesystem`).

```
aws efs create-replication-configuration \
--source-file-system-id fs-0123456789abcdef1 \
--destinations AvailabilityZoneName=us-west-2a
```

Step 2: Mount the destination file system

Amazon EFS does not create any mount targets when it creates the destination file system. To mount the destination file system, you must create one or more mount targets. For more information, see Mounting EFS file systems.

Configuring replication to an existing EFS file system

Amazon EFS replicates the data and metadata on the source file system to the destination file system and AWS Region that you choose. During replication, Amazon EFS identifies data differences between the file systems and applies the differences to the destination file system.

To replicate to an existing file system, perform the following steps.

Topics

- Step 1: Disable the file system's replication overwrite protection
- Step 2: Create the replication configuration

> **Note**
>
> A file system can be part of only one replication configuration. You cannot use a destination file system as the source file system in another replication configuration.

Step 1: Disable the file system's replication overwrite protection

When you create an Amazon EFS file system, its replication overwrite protection is enabled by default. Replication overwrite protection prevents the file system from being used as the destination in a replication configuration. Before you can use the file system as the destination in a replication configuration, you must disable the protection. If you delete the replication configuration, the file system's replication overwrite protection is re-enabled and the file system becomes writeable.

The status of the replication overwrite protection for an Amazon EFS file system can have one of the values described in the following table.

File system state	Description
ENABLED	The file system cannot be used as the destination file system in a replication configuration. The file system is writeable. Replication overwrite protection is `ENABLED` by default.
DISABLED	The file system can be used as the destination file system in a replication configuration.
REPLICATING	The file system is being used as the destination file system in a replicati on configuration. The file system is read-only and is only modified only by Amazon EFS during replication.

Required permission

Disabling replication overwrite protection requires permissions for the `elasticfilesystem:UpdateFileSystemProtection` action. For more information, see [AWS managed policy: AmazonElasticFileSystemFullAccess].

Using the console

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. In the left navigation pane, choose **File systems**.
3. In the **File systems** list, choose the Amazon EFS file system that you want to use as the destination file system in a replication configuration.
4. In the **File system protection** section, turn off **Replication Overwrite Protection**.

To disable replication overwrite protection (AWS CLI)

In the following example, the `update-file-system-protection` CLI command disables the replication overwrite protection for the specified file system. The equivalent API command is UpdateFileSystemProtection.

```
aws efs update-file-system-protection
 --file-system-id fs-0a8b2be428114d97c
 --replication-overwrite-protection DISABLED
```

The AWS CLI responds as follows.

```
{
    "ReplicationOverwriteProtection": "DISABLED"
}
```

Step 2: Create the replication configuration

After you disable replication overwrite protection on the destination file system, you can create the replication configuration. When replicating to an existing file system, the destination file system can be in the same account or in a different account than the source file system.

If the source file system is encrypted, then the destination file system must also be encrypted. Additionally, if the source file is unencrypted and the destination file system is encrypted, then you cannot fail back to the source destination after performing failover. For more information about encryption, see Encrypting data in Amazon EFS.

> **Note**
>
> When you initially configure replication to an existing file system, Amazon EFS writes data to or removes existing data from the destination file system to match data in the source file system. If you don't want to change data in the destination file system, then you should replicate to a new file system instead. For more information, see Configuring replication to new EFS file system.

Prerequisites

Have a copy of the destination file system ID (for same-account replication) or the destination file system ARN (for cross-account replication) that you want to use.

If the destination file system is in a different AWS account than the source file system, create an IAM role that allows Amazon EFS to perform replication and assign resource policies to the file systems. For more information, see Replicating EFS file systems across AWS accounts.

Using the console

1. Sign in to the AWS Management Console and open the Amazon EFS console at https://console.aws.amazon.com/efs/.
2. Open the file system that you want to replicate:
   a. In the left navigation pane, choose **File systems**.
   b. In the **File systems** list, choose the Amazon EFS file system that you want to replicate. The file system that you choose cannot be a source or destination file system in an existing replication configuration.
3. Choose the **Replication** tab.
4. In the **Replication** section, choose **Create replication**.
5. For **Replication configuration**, choose existing file system.
6. Choose the destination file system.
   - To replicate to a file system that's in the same AWS account as the source file system:
     1. Select **Choose a file system in this account** and, for **Destination AWS Region**, select the AWS Region to which to replicate the file system.
     2. Choose **Browse EFS**, and then select the file system. The path to your destination file system appears in the **Destination** box.
   - To replicate to a file system that’s in a different AWS account than the source file system:
     1. Choose **Specify a file system in another account**.
     2. For **Destination file system ARN**, enter the Amazon Resource Name (ARN) of the destination file system.

   > **Note**
   >
   > If replication overwrite protection is enabled on the file system, then a warning displays. Choose **Disable protection** to open the file system in a new tab and turn off its **Replication overwrite protection**. After disabling the protection, return to the **Create replication** tab and click the **Refresh** button to clear the message.

7. For **IAM role**, enter the ARN of the IAM role that allows Amazon EFS to replicate to the destination file system. This is optional for same-account replication, but required for cross-

account replication. For more information, see Replicating EFS file systems across AWS accounts.

8. Choose **Create replication**, type **confirm** in the confirmation message input box, and then choose **Create replication**. The **Replication** section shows the replication details.

To create the replication configuration (AWS CLI)

This section provides examples for creating a replication configuration in the AWS CLI using the `create-replication-configuration` command. The equivalent API command is CreateReplicationConfiguration.

Example : Create a replication configuration to an existing destination file system in another Region

The following example creates a replication configuration where the file system ID *fs-0123456789abcdef1* is replicated to file system ID *fs-0a8b2be428114d97c* in the *eu-west-2* AWS Region.

```
aws efs create-replication-configuration \
--source-file-system-id fs-0123456789abcdef1 \
--destinations "[{\"Region\":\"eu-west-2\",\"FileSystemId\":\"fs-0a8b2be428114d97c\"}]"
```

The AWS CLI responds as follows:

```
{
    "SourceFileSystemId": "fs-0123456789abcdef1",
    "SourceFileSystemRegion": "us-east-1",
    "SourceFileSystemArn": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/fs-0123456789abcdef1",
    "OriginalSourceFileSystemArn": "arn:aws:elasticfilesystem:us-
east-1:111122223333:file-system/fs-0123456789abcdef1",
    "CreationTime": "2024-10-20T20:40:13+00:00",
    "Destinations": [
        {
            "Status": "ENABLING",
            "FileSystemId": "fs-0a8b2be428114d97c",
            "Region": "eu-west-2",
            "OwnerId": "123456789012,
         }
    ],
    "SourceFileSystemOwnerId": "123456789012"
```

```
}
```

Example : Create a cross-account replication configuration

The following example creates a replication configuration where the source and destination file systems are in different AWS accounts. The source file system ID *fs-0123456789abcdef1* in account *555666777888* is replicated to file system ID *fs-0a8b2be428114d97c* in account *123456789012*. The example specifies the Amazon Resource Name (ARN) of the destination file system and the ARN of the IAM role in the source account that allows Amazon EFS to perform replication on its behalf. Because no KMS key is specified, the destination file system is encrypted using the account's default AWS KMS service key (`aws/elasticfilesystem`).

```
aws efs
--region $REGION
--endpoint $ENDPOINT create-replication-configuration
--source-file-system-id fs-0123456789abcdef1
--destinations Region=eu-west-2,FileSystemId=arn:aws:elasticfilesystem:eu-
west-2:123456789012:file-system/
fs-0a8b2be428114d97c,RoleArn=arn:aws:iam::555666777888:role/cross-account-replication
```

The AWS CLI responds as follows:

```
{
    "SourceFileSystemId": "fs-0123456789abcdef1",
    "SourceFileSystemRegion": "us-east-1",
    "SourceFileSystemArn": "arn:aws:elasticfilesystem:us-east-1:555666777888:file-
system/fs-0123456789abcdef1",
    "OriginalSourceFileSystemArn": "arn:aws:elasticfilesystem:us-
east-1:555666777888:file-system/fs-0123456789abcdef1",
    "CreationTime": "2024-10-20T20:40:13+00:00",
    "Destinations": [
        {
            "Status": "ENABLING",
            "FileSystemId": "fs-0a8b2be428114d97c",
            "Region": "eu-west-2",
            "OwnerId": "123456789012,
            "RoleArn": "arn:aws:iam::555666777888:role/cross-account-replication"
        }
    ],
    "SourceFileSystemOwnerId": "555666777888"
}
```

Replicating EFS file systems across AWS accounts

You can replicate EFS file systems across AWS accounts. Replicating across accounts enhances the overall resilience and reliability of your disaster recovery (DR) strategies and can help you meet corporate compliance mandates.

For example, you might be required by compliance policies to use different accounts for different environments (such as production, staging, and disaster recovery (DR)). Or you may find that replication across different AWS accounts provides stronger isolation, more granular control over permissions and access policies, and more straightforward auditing of resources. If the production account is compromised (such as by security breaches, misconfiguration, or insider threats), having the DR servers in a separate account can prevent the attacker from accessing them, reduce the blast radius of security incidents, and minimize the risk of unauthorized changes.

Replicating across AWS accounts requires additional security and policy setup. Instead of using service-linked roles to perform cross-account replication, you must create an IAM role that gives Amazon EFS permission to perform replication in the destination account. You also need to create policies on the file systems that you want to share across accounts. After the IAM role and file system policies are created, you create the replication configuration.

Topics

- Create an IAM role with a custom trust policy
- Create policies on the source and destination file systems
- Create the replication configuration

Create an IAM role with a custom trust policy

For Amazon EFS to perform cross-account replication on the source account's behalf, an IAM role must be created on the source account. The role must have the `elasticfilesystem.amazonaws.com` trust policy to allow Amazon EFS to assume the role and act as the service principal. The role must contain all of the IAM permissions required to perform replication (see Required IAM permissions) and grant explicit permission to replicate to the file system in the destination account.

Prerequisites

You must create both the source file system and the destination file system in the replication configuration before you can create the IAM role for the source account. Amazon EFS cannot create the destination file system for you during replication. Additionally, you must know and provide the Amazon Resource Name (ARN) for each file system.

To create the IAM role for cross-account replication

The following are the general steps for creating an IAM role with custom trust policies for cross-account replication with Amazon EFS. For step-by-step instructions for creating an IAM role, see [Create a role using custom trust policies]() in the *AWS Identity and Access Management User Guide*.

1. In the AWS Identity and Access Management console for the source account, create an IAM role that uses the following trust policy. For instructions, see [Create a role using custom trust policies]() in the *AWS Identity and Access Management User Guide*.

   JSON

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

2. After you create the role, assign the following permissions for the role. Replace `arn:aws:elasticfilesystem:us-east-1:111122223333:file-system/fs-0123456789abcdef1` with the ARN of the destination file system and replace `arn:aws:elasticfilesystem:us-east-1:444455556666:file-system/fs-5678910112hijkqr1` with the ARN of the source file system. For instructions on assigning permissions to the role, see [Creating policies using the JSON editor]().

   JSON

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

```
        {
            "Effect": "Allow",
            "Action":[
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:CreateReplicationConfiguration",
                "elasticfilesystem:DescribeReplicationConfigurations",
                "elasticfilesystem:DeleteReplicationConfiguration",
                "elasticfilesystem:ReplicationWrite"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-
east-1:111122223333:file-system/fs-0123456789abcdef1"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ReplicationRead",
                "elasticfilesystem:DescribeFileSystems"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-
east-1:444455556666:file-system/fs-5678910112hijkqr1"
        }
    ]
}
```

3. Copy or write down the ARN for the IAM role. You need to provide the ARN when you create the replication configuration.

Create policies on the source and destination file systems

To share file systems cross-account in Amazon EFS, you must assign policies to both the destination and source file systems. The policies grant or restrict access across accounts to the file system to which they are applied. Only account owners with permission to edit file systems can assign policies to the file system in their account.

In addition to granting or restricting access across accounts, the policies need to grant other permissions required for clients to work with the file systems, such as `elasticfilesystem:ClientMount`. Otherwise, the file system might be inaccessible to clients.

> **Important**
>
> You cannot restrict access to resources over TLS connection. If you include the `"aws:SecureTransport": "false"` condition in your statement, the NFS client connection will fail.

Policy for the destination file system

To allow the source account permission to replicate to the destination file system and to delete the replication configuration from the destination account, the following policy must be created on the destination file system. Replace `arn:aws:iam::444455556666:root` with the ID of the account that owns the source file system. Replace `arn:aws:elasticfilesystem:us-east-1:111122223333:file-system/fs-0123456789abcdef1` with the ARN of the destination file system

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
        "Sid": "AllowSourceAccountReplicationActions",
        "Effect": "Allow",
        "Principal": {
        "AWS": "arn:aws:iam::444455556666:root"
      },
    "Action": [
         "elasticfilesystem:DescribeFileSystems",
         "elasticfilesystem:CreateReplicationConfiguration",
         "elasticfilesystem:DescribeReplicationConfigurations",
         "elasticfilesystem:DeleteReplicationConfiguration",
         "elasticfilesystem:ReplicationWrite"
         ],
     "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-system/
fs-0123456789abcdef1"
     },
     {
        "Sid": "AllowReadOnlyClientAccess",
        "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EfsReadOnly"
            },
            "Action": [
                "elasticfilesystem:ClientMount"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-system/fs-0123456789abcdef1"
        }
    ]
}
```

Policy for the source file system

To allow the destination account permission to delete the replication configuration from the source account, you must assign the following policy to the source file system. Replace `arn:aws:iam::111122223333:root` with the ID of the account that owns the destination file system. Replace `arn:aws:elasticfilesystem:us-east-1:444455556666:file-system/fs-5678910112hijkqr1` with the ARN of the source file system.

JSON

```
{
    "Version": "2012-10-17",
    "Id": "efs-policy-wizard-15ad9567-2546-4bbb-8168-5541b6fc0e55",
    "Statement": [
        {
            "Sid": "AllowDestinationAccountToDeleteReplication",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "elasticfilesystem:DeleteReplicationConfiguration",
            "Resource": "arn:aws:elasticfilesystem:us-east-1:444455556666:file-system/fs-5678910112hijkqr1"
        },
        {
            "Sid": "AllowClientAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EfsReadOnly"
```

```
            },
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientRootAccess"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:444455556666:file-
system/fs-5678910112hijkqr1",
            "Condition": {
               "Bool": {
                   "elasticfilesystem:AccessedViaMountTarget": "true"
               }
           }
        }
    ]
}
```

To create the file system policy

Perform the following steps for both the destination and source file system, using the policies in the previous section.

1. Sign in to the AWS Management Console with the account that owns the file system, and then open the Amazon EFS console at Amazon EFS Console.
2. Open the file system:
   a. In the left navigation pane, choose **File systems**.
   b. In the **File systems** list, choose the file system.
3. On the **File system policy** tab, choose **Edit**.
4. Paste the policy in **Policy editor {Json}** and then choose **Save**.

Create the replication configuration

After you have created the IAM role and added the file system policies to the source and destination file systems, follow the instructions in Configuring replication to an existing EFS file system to create the replication configuration.

Viewing replication details

You can monitor the time when the last successful sync was completed in a replication configuration. Any changes to data on the source file system that occurred before this time have been successfully replicated to the destination file system. Any changes that occurred after this time might not be fully replicated. To monitor when the last replication successfully finished, you can use the Amazon EFS console, AWS CLI, API, or Amazon CloudWatch.

- **In the EFS console** – The **Last synced** property in the **File system details** > **Replication** section shows the time when the last successful sync between the source and destination was completed.
- **In the AWS CLI or API** – The `LastReplicatedTimestamp` property in the `Destination` object shows the time that the last successful sync was completed. To access this property, use the `describe-replication-configurations` CLI command. DescribeReplicationConfigurations is the equivalent API operation.
- **In CloudWatch** – The `TimeSinceLastSync` CloudWatch metric for Amazon EFS shows the time that has elapsed since the last successful sync was completed. For more information, see CloudWatch metrics for Amazon EFS.

A replication configuration can have one of the status values described in the following table.

Replication state	Description
`ENABLED`	The replication configuration is in a healthy state and available for use.
`ENABLING`	Amazon EFS is in the process of creating the replication configuration.
`DELETING`	Amazon EFS is deleting the replication configuration in response to a user-initiated delete request.
`PAUSING`	Amazon EFS is in the process of pausing replication.
`PAUSED`	Replication is paused due to a problem with the configuration. Additional information about the problem is provided. Some problems that cause the replication to be paused include:

Replication state	Description
	• Authorization error. Permissions issues prevent Amazon EFS from performing replication. Make sure that the IAM role used to create the replication configuration has permission to perform replication. Also make sure that the file system policies are correct. • AWS account is unavailable. Make sure that the source and destinati on accounts are opted in to the appropriate AWS Regions and are not suspended. For more information, see Specify which AWS Regions your account can use in the *AWS General Reference Guide*. • KMS key for the source or destination file system is not accessible. Make sure that the KMS key assigned to each file system is accessible. For more information, see Managing KMS keys for EFS file systems.
`ERROR`	The replication configuration is in a failed state and is unrecoverable. You must delete the replication configuration and create a new one. Additional information about the problem is provided. For cross-account or cross-Region replication, the error may be caused because the replication configuration was deleted from the other AWS account or AWS Region.

Using the console

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the left navigation pane, choose **File systems**.
3. Choose a file system from the list.
4. Choose the **Replication** tab to display the **Replication** section.

   In the **Replication** section, you can see the following information for the replication configuration:

   - **Replication state** may be **Enabling**, **Enabled**, **Deleting**, **Pausing**, **Paused**, or **Error**. Amazon EFS displays details about the cause for the **Paused** or **Error** state.
   - **Replication direction** shows the direction in which data is being replicated. The first file system listed is the source, and its data is being replicated *to* the second file system listed, which is the destination.

- **Last synced** shows when the last successful sync occurred on the destination file system. Any changes to data on the source file system that occurred before this time were successfully replicated to the destination file system. Any changes that occurred after this time might not be fully replicated.
- **Replication file systems** lists each file system in the replication configuration by its file system ID, the role it has in the replication configuration (either source or destination), the AWS Region in which it's located, and its **Permission**. A source file system has a permission of **Writable**, and a destination file system has a permission of **Read-only**.

Using the AWS CLI

To view a replication configuration, use the `describe-replication-configurations` command. You can view the replication configuration for either a specific file system, or all replication configurations for a particular AWS account in an AWS Region. The equivalent API command is DescribeReplicationConfigurations.

If the status of the replication configuration is `PAUSED` or `ERROR`, information about the cause of the issue and how to fix it is returned in the `StatusMessage` parameter.

Example : View the replication configuration for a specific file system

The following example describes the replication configuration for the file system `fs-0123456789abcdef1`.

```
aws efs describe-replication-configurations --file-system-id fs-0123456789abcdef1
```

The AWS CLI responds as follows.

```
{
    "Replications": [
        {
            "SourceFileSystemArn": "arn:aws:elasticfilesystem:eu-
west-1:111122223333:file-system/fs-abcdef0123456789a",
            "CreationTime": 1641491892.0,
            "SourceFileSystemRegion": "eu-west-1",
            "OriginalSourceFileSystemArn": "arn:aws:elasticfilesystem:eu-
west-1:111122223333:file-system/fs-abcdef0123456789a",
            "SourceFileSystemId": "fs-abcdef0123456789a",
            "Destinations": [
```

```
                {
                    "Status": "ENABLED",
                    "FileSystemId": "fs-0123456789abcdef1",
                    "Region": "us-east-1"
                }
            ]
        }
    ]
}
```

Example : View the replication configuration for all the replication configurations in an account

The following example describes the replication configuration for all the replication configurations for an account in an AWS Region the file system.

```
aws efs describe-replication-configurations
```

The AWS CLI responds as follows.

```
     {
    "Replications": [
        {
            "SourceFileSystemArn": "arn:aws:elasticfilesystem:eu-
west-1:555555555555:file-system/fs-0123456789abcdef1",
            "CreationTime": 1641491892.0,
            "SourceFileSystemRegion": "eu-west-1",
            "OriginalSourceFileSystemArn": "arn:aws:elasticfilesystem:eu-
west-1:555555555555:file-system/fs-0123456789abcdef1",
            "SourceFileSystemId": "fs-0123456789abcdef1",
            "Destinations": [
                {
                    "Status": "ENABLED",
                    "FileSystemId": "fs-abcdef0123456789a",
                    "Region": "us-east-1",
                    "LastReplicatedTimestamp": 1641491802.375
                }
            ]
        },
```

```
        {
            "SourceFileSystemArn": "arn:aws:elasticfilesystem:eu-
west-1:555555555555:file-system/fs-021345abcdef6789a",
            "CreationTime": 1641491822.0,
            "SourceFileSystemRegion": "eu-west-1",
            "OriginalSourceFileSystemArn": "arn:aws:elasticfilesystem:eu-
west-1:555555555555:file-system/fs-021345abcdef6789a",
            "SourceFileSystemId": "fs-021345abcdef6789a",
            "Destinations": [
                {
                    "Status": "ENABLED",
                    "FileSystemId": "fs-012abc3456789def1",
                    "Region": "us-east-1",
                    "LastReplicatedTimestamp": 1641491823.575
                }
            ]
        }
    ]
}
```

Deleting replication configurations

If you need to fail over to the destination file system, delete the replication configuration of which it is a member. After you delete a replication configuration, the destination file system becomes writeable and its replication overwrite protection is re-enabled. For more information, see Using the replica.

Deleting a replication configuration and changing the destination file system to be writeable can take several minutes to complete. After the configuration is deleted, Amazon EFS might write some data to a `lost+found` directory in the root directory of the destination file system, using the following naming convention:

```
efs-replication-lost+found-source-file-system-id-TIMESTAMP
```

Note

You cannot delete a file system that is part of a replication configuration. You must delete the replication configuration before deleting the file system.

You can delete an existing replication configuration from either the source or the destination file system by using the Amazon EFS console, the AWS CLI, or the API.

For cross-account or cross-Region replications, Amazon EFS deletes the replication configuration from both the source and destination accounts or Regions. If there's a configuration or permissions issue that prevents Amazon EFS from deleting the replication configuration from both sides, you can delete the configuration from only the local side (the account or Region from which the delete is performed). Deleting the local configuration leaves the configuration in the other account or Region unrecoverable.

Using the console

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. In the left navigation pane, choose **File systems**.
3. Choose either the source or the destination file system that is in the replication configuration that you want to delete.
4. Choose the **Replication** tab to display the **Replication** section.
5. Choose **Delete replication** to delete the replication configuration. When prompted, confirm your choice.

   If you are deleting a cross-account replication configuration, and there's a problem that prevents you from deleting the configuration from both the source and destination side, then you can choose the option to delete this file system's configuration only.

   > **Note**
   >
   > Delete the file system's configuration only if Amazon EFS is unable to delete the replication configuration in both the source and destination account or Region. Deleting the local configuration leaves the configuration in the other account or Region unrecoverable.

Using the AWS CLI

To delete a replication configuration, use the `delete-replication-configuration` CLI. The equivalent API command is [DeleteReplicationConfiguration](#).

The following example deletes the replication configuration for source file system `fs-0123456789abcdef1`.

```
aws efs --region us-west-2 delete-replication-configuration \
--source-file-system-id fs-0123456789abcdef1
```

If a configuration or permissions issue prevents Amazon EFS from deleting the replication configuration from both sides, you can delete the configuration from only the local side (the account or Region from which the delete is performed). Deleting the local configuration leaves the configuration in the other account or Region unrecoverable. The equivalent API parameter is `DeletionMode` and the value is `LOCAL_CONFIGURATION_ONLY`.

The following example deletes the replication configuration for source file system *fs-0123456789abcdef1* from the local side only.

```
aws efs --region us-west-2 delete-replication-configuration \
--source-file-system-id fs-0123456789abcdef1
--deletion-mode LOCAL_CONFIGURATION_ONLY
```

Using the replica

In the event of a disaster or when performing game day exercises, you can fail over to your replica file system by deleting its replication configuration. After the replication configuration is deleted, the replica becomes writeable and you can start using it in your application workflow. When the disaster is mitigated or the game day exercise is over, you can continue using the replica as the primary file system or you can perform a failback to resume operations on your original primary file system.

During the failback process, you can choose to discard the changes made to your replica file system or preserve them by copying them back to your primary.

- To discard the changes made to your replica during failover, re-create the original replication configuration on your primary file system, where the replica file system is the replication destination. During replication, Amazon EFS synchronizes the file systems by updating your replica file system's data to match that of your primary.
- To replicate the changes made to your replica during failover, create a replication configuration on the replica file system, where the primary file system is the replication destination. During replication, Amazon EFS identifies and transfers the differences from your replica file system back to the primary file system. Once the replication is complete, you can resume replicating the primary file system by re-creating the original replication configuration or creating a new configuration.

The amount of time it takes for Amazon EFS to complete the replication process varies and depends on factors such as the size of the file system and the number of files in it. For more information, see Replication performance.

Securing your data in Amazon EFS

The AWS shared responsibility model applies to data protection in Amazon Elastic File System. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see 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 users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with EFS or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Topics

- Encrypting data in Amazon EFS
- Identity and access management for Amazon EFS
- Using IAM to control access to file systems
- Controlling network access to EFS file systems for NFS clients
- Network File System (NFS) level users, groups, and permissions
- Working with access points
- Blocking public access to EFS file systems
- Compliance validation for Amazon EFS
- Resilience in Amazon EFS
- Network isolation for Amazon EFS

Encrypting data in Amazon EFS

Amazon EFS supports two forms of encryption for file systems, encryption of data in transit and encryption at rest. You can enable encryption of data at rest when creating an Amazon EFS file system. You can enable encryption of data in transit when you mount the file system.

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.

If your organization is subject to corporate or regulatory policies that require encryption of data and metadata at rest, we recommend creating a file system that is encrypted at rest, and mounting your file system using encryption of data in transit.

Topics

- AWS KMS
- Encrypting data at rest
- Encrypting data in transit
- Troubleshooting encryption

AWS KMS

Amazon EFS integrates with AWS Key Management Service (AWS KMS) for key management. Amazon EFS uses customer managed keys to encrypt your file system in the following way:

- **Encrypting metadata at rest –** Amazon EFS uses the AWS managed key for Amazon EFS, `aws/elasticfilesystem`, to encrypt and decrypt file system metadata (that is, file names, directory names, and directory contents).
- **Encrypting file data at rest –** You choose the customer managed key used to encrypt and decrypt file data (that is, the contents of your files). You can enable, disable, or revoke grants on this customer managed key. This customer managed key can be one of the two following types:
  - **AWS managed key for Amazon EFS –** This is the default customer managed key, `aws/elasticfilesystem`. You're not charged to create and store a customer managed key, but there are usage charges. To learn more, see AWS Key Management Service pricing.
  - **Customer managed key –** This is the most flexible KMS key to use, because you can configure its key policies and grants for multiple users or services. For more information on creating customer managed keys, see Creating keys in the *AWS Key Management Service Developer Guide.*

    If you use a customer managed key for file data encryption and decryption, you can enable key rotation. When you enable key rotation, AWS KMS automatically rotates your key once per year. Additionally, with a customer managed key, you can choose when to disable, re-enable, delete, or revoke access to your customer managed key at any time. For more information, see Managing access to encrypted file systems.

> **Important**
>
> Amazon EFS accepts only symmetric customer managed keys. You cannot use asymmetric customer managed keys with Amazon EFS.

Data encryption and decryption at rest are handled transparently. However, AWS account IDs specific to Amazon EFS appear in your AWS CloudTrail logs related to AWS KMS actions. For more information, see Amazon EFS log file entries for encrypted-at-rest file systems.

Amazon EFS key policies for AWS KMS

Key policies are the primary way to control access to customer managed keys. For more information on key policies, see Key policies in AWS KMS in the *AWS Key Management Service Developer Guide.* The following list describes all the AWS KMS–related permissions that are required or otherwise supported by Amazon EFS for encrypted at rest file systems:

- **kms:Encrypt** – (Optional) Encrypts plaintext into ciphertext. This permission is included in the default key policy.
- **kms:Decrypt** – (Required) Decrypts ciphertext. Ciphertext is plaintext that has been previously encrypted. This permission is included in the default key policy.
- **kms:ReEncrypt** – (Optional) Encrypts data on the server side with a new customer managed key, without exposing the plaintext of the data on the client side. The data is first decrypted and then re-encrypted. This permission is included in the default key policy.
- **kms:GenerateDataKeyWithoutPlaintext** – (Required) Returns a data encryption key encrypted under a customer managed key. This permission is included in the default key policy under **kms:GenerateDataKey***.
- **kms:CreateGrant** – (Required) Adds a grant to a key to specify who can use the key and under what conditions. Grants are alternate permission mechanisms to key policies. For more information on grants, see Grants in AWS KMS in the *AWS Key Management Service Developer Guide.* This permission is included in the default key policy.
- **kms:DescribeKey** – (Required) Provides detailed information about the specified customer managed key. This permission is included in the default key policy.
- **kms:ListAliases** – (Optional) Lists all of the key aliases in the account. When you use the console to create an encrypted file system, this permission populates the **Select KMS key** list. We recommend using this permission to provide the best user experience. This permission is included in the default key policy.

AWS managed key for Amazon EFS KMS policy

The KMS policy JSON for the AWS managed key for Amazon EFS, `aws/elasticfilesystem` is as follows:

JSON

```
{
    "Version": "2012-10-17",
    "Id": "auto-elasticfilesystem-1",
    "Statement": [
        {
            "Sid": "Allow access to EFS for all principals in the account that
 are authorized to use EFS",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "*"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "elasticfilesystem.us-
east-2.amazonaws.com",
                    "kms:CallerAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "Allow direct access to key metadata to the account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "kms:Describe*",
                "kms:Get*",
                "kms:List*",
                "kms:RevokeGrant"
            ],
            "Resource": "*"
        }
    ]
}
```

Encrypting data at rest

You can create encrypted file systems using the AWS Management Console, the AWS CLI, or programmatically through the Amazon EFS API or one of the AWS SDKs. Your organization might require the encryption of all data that meets a specific classification or is associated with a particular application, workload, or environment.

Once you create an EFS file system, you cannot change its encryption setting. This means that you cannot modify an unencrypted file system to make it encrypted. Instead, you need to create a new, encrypted file system.

> **Note**
>
> The AWS key management infrastructure uses Federal Information Processing Standards (FIPS) 140-2 approved cryptographic algorithms. The infrastructure is consistent with National Institute of Standards and Technology (NIST) 800-57 recommendations.

Enforcing the creation of EFS file systems encrypted at rest

You can use the `elasticfilesystem:Encrypted` IAM condition key in AWS Identity and Access Management (IAM) identity-based policies to control whether users can create Amazon EFS file systems that are encrypted at rest. For more information about using the condition key, see Example: Enforce the creation of encrypted file systems.

You can also define service control policies (SCPs) inside AWS Organizations to enforce EFS encryption for all AWS accounts in your organization. For more information about service control policies in AWS Organizations, see Service control policies in the *AWS Organizations User Guide*.

Encrypting a file system at rest using the console

When you create a new file system using the Amazon EFS console, encryption at rest is enabled by default.

> **Note**
>
> Encryption at rest is not enabled by default when creating a new file system using the AWS CLI, API, and SDKs. For more information, see Create using the AWS CLI.

How encryption at rest works

In an encrypted file system, data and metadata are automatically encrypted before being written to the file system. Similarly, as data and metadata are read, they are automatically decrypted before being presented to the application. These processes are handled transparently by Amazon EFS, so you don't have to modify your applications.

Amazon EFS uses industry-standard AES-256 encryption algorithm to encrypt EFS data and metadata at rest. For more information, see AWS KMS cryptography essentials in the *AWS Key Management Service Developer Guide*.

Encrypting data in transit

Enabling encryption of data in transit for your EFS file system is done by enabling Transport Layer Security (TLS) when you mount your file system using the Amazon EFS mount helper. For more information, see Mounting EFS file systems using the EFS mount helper.

When encryption of data in transit is declared as a mount option for your Amazon EFS file system, the mount helper initializes a client stunnel process. Stunnel is an open source multipurpose network relay. The client stunnel process listens on a local port for inbound traffic, and the mount helper redirects Network File System (NFS) client traffic to this local port. The mount helper uses TLS version 1.2 to communicate with your file system.

How encrypting in transit works

To enable encryption of data in transit, you connect to Amazon EFS using TLS. We recommend using the EFS mount helper to mount your file system because it simplifies the mounting process as compared to mounting with NFS `mount`. The EFS mount helper manages the process using `stunnel` for TLS. If you're not using the mount helper, you can still enable encryption of data in transit. At a high level, the following are the steps to do so.

To enable encryption of data in transit without using the EFS mount helper

1. Download and install `stunnel`, and note the port that the application is listening on. For instructions to do so, see Upgrading `stunnel`.
2. Run `stunnel` to connect to your EFS file system on port 2049 using TLS.
3. Using the NFS client, mount `localhost:port`, where *`port`* is the port that you noted in the first step.

Because encryption of data in transit is configured on a per-connection basis, each configured mount has a dedicated `stunnel` process running on the instance. By default, the `stunnel` process used by the EFS mount helper listens on a local port ranging from 20049 to 21049, and it connects to Amazon EFS on port 2049.

Note

By default, when using the Amazon EFS mount helper with TLS, the mount helper enforces certificate hostname checking. The Amazon EFS mount helper uses the `stunnel` program for its TLS functionality. Some versions of Linux don't include a version of stunnel that supports these TLS features by default. When using one of those Linux versions, mounting an EFS file system using TLS fails.

After you've installed the amazon-efs-utils package, to upgrade your system's version of stunnel see Upgrading `stunnel`.

For issues with encryption, see Troubleshooting encryption.

When using encryption of data in transit, your NFS client setup is changed. When you inspect your actively mounted file systems, you see one mounted to 127.0.0.1, or `localhost`, as in the following example.

```
$ mount | column -t
127.0.0.1:/  on  /home/ec2-user/efs        type  nfs4
 (rw,relatime,vers=4.1,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,port=20127,timeo=6
```

When mounting with TLS and the Amazon EFS mount helper, you are reconfiguring your NFS client to mount to a local port. The EFS mount helper starts a client `stunnel` process that is listening on this local port, and `stunnel` is opening an encrypted connection to the EFS file system using TLS. The EFS mount helper is responsible for setting up and maintaining this encrypted connection and the associated configuration.

To determine which EFS file system ID corresponds to which local mount point, you can use the following command. Replace *`efs-mount-point`* with the local path where you mounted your file system.

```
grep -E "Successfully mounted.*efs-mount-point" /var/log/amazon/efs/mount.log | tail -1
```

When you use the mount helper for encryption of data in transit, it also creates a process called `amazon-efs-mount-watchdog`. This process ensures that each mount's stunnel process is running, and stops the stunnel when the EFS file system is unmounted. If for some reason a stunnel process is terminated unexpectedly, the watchdog process restarts it.

Troubleshooting encryption

Following, you can find information about troubleshooting encryption issues for Amazon EFS.

- Mounting with encryption of data in transit fails
- Mounting with encryption of data in transit is interrupted
- Encrypted-at-rest file system can't be created
- Unusable encrypted file system

Mounting with encryption of data in transit fails

By default, when you use the Amazon EFS mount helper with Transport Layer Security (TLS), it enforces hostname checking. Some systems don't support this feature, such as when you use Red Hat Enterprise Linux or CentOS. In these cases, mounting an EFS file system using TLS fails.

Action to take

We recommend that you upgrade the version of stunnel on your client to support hostname checking. For more information, see Upgrading `stunnel`.

Mounting with encryption of data in transit is interrupted

It's possible, however unlikely, that your encrypted connection to your Amazon EFS file system can hang or be interrupted by client-side events.

Action to take

If your connection to your Amazon EFS file system with encryption of data in transit is interrupted, take the following steps:

1. Ensure that the stunnel service is running on the client.
2. Confirm that the watchdog application `amazon-efs-mount-watchdog` is running on the client. You can find out whether this application is running with the following command:

```
ps aux | grep [a]mazon-efs-mount-watchdog
```

3. Check your support logs. For more information, see Getting support logs.
4. Optionally, you can enable your stunnel logs and check the information in those as well. You can change the configuration of your logs in `/etc/amazon/efs/efs-utils.conf` to enable the

stunnel logs. However, doing so requires unmounting and then remounting the file system with the mount helper for the changes to take effect.

> **Important**
> Enabling the stunnel logs can use up a nontrivial amount of space on your file system.

If the interruptions continue, contact AWS Support.

Encrypted-at-rest file system can't be created

You've tried to create a new encrypted-at-rest file system. However, you get an error message saying that AWS KMS is unavailable.

Action to take

This error can occur in the rare case that AWS KMS becomes temporarily unavailable in your AWS Region. If this happens, wait until AWS KMS returns to full availability, and then try again to create the file system.

Unusable encrypted file system

An encrypted file system consistently returns NFS server errors. These errors can occur when EFS can't retrieve your master key from AWS KMS for one of the following reasons:

- The key was disabled.
- The key was deleted.
- Permission for Amazon EFS to use the key was revoked.
- AWS KMS is temporarily unavailable.

Action to take

First, confirm that the AWS KMS key is enabled. You can do so by viewing the keys in the console. For more information, see Viewing Keys in the *AWS Key Management Service Developer Guide*.

If the key is not enabled, enable it. For more information, see Enabling and Disabling Keys in the *AWS Key Management Service Developer Guide*.

If the key is pending deletion, then this status disables the key. You can cancel the deletion, and re-enable the key. For more information, see Scheduling and Canceling Key Deletion in the *AWS Key Management Service Developer Guide*.

If the key is enabled, and you're still experiencing an issue, or if you encounter an issue re-enabling your key, contact AWS Support.

Identity and access management for Amazon EFS

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 Amazon EFS resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How Amazon Elastic File System works with IAM
- Identity-based policy examples for Amazon Elastic File System
- Resource-based policy examples for Amazon EFS
- AWS managed policies for Amazon EFS
- Using tags with Amazon EFS
- Using service-linked roles for Amazon EFS
- Troubleshooting Amazon Elastic File System identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon EFS.

Service user – If you use the Amazon EFS service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon EFS features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon EFS, see Troubleshooting Amazon Elastic File System identity and access.

Service administrator – If you're in charge of Amazon EFS resources at your company, you probably have full access to Amazon EFS. It's your job to determine which Amazon EFS features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon EFS, see How Amazon Elastic File System works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon EFS. To view example Amazon EFS identity-based policies that you can use in IAM, see Identity-based policy examples for Amazon Elastic File System.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.

  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Use an IAM role to grant permissions to applications running on Amazon EC2 instances 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 is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles 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 further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, 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 of resource-based policies are 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. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

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, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's

permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. 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 Amazon Elastic File System works with IAM

Before you use IAM to manage access to Amazon EFS, learn what IAM features are available to use with Amazon EFS.

IAM features you can use with Amazon Elastic File System

IAM feature	Amazon EFS support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how Amazon EFS and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Amazon EFS

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles 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*.

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. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon EFS

To view examples of Amazon EFS identity-based policies, see Identity-based policy examples for Amazon Elastic File System.

Resource-based policies within Amazon EFS

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are 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. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

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, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They 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 Cross account resource access in IAM in the *IAM User Guide*.

To learn about using a resource policy to control file system data access, see Using IAM to control access to file systems. To learn how to attach a resource-based policy to a file system, see Creating file system policies.

Resource-based policy examples within Amazon EFS

To view examples of Amazon EFS resource-based policies, see Resource-based policy examples for Amazon EFS.

Policy actions for Amazon EFS

Supports policy actions: Yes

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. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon EFS actions, see Actions defined by Amazon Elastic File System in the *Service Authorization Reference*.

Policy actions in Amazon EFS use the following prefix before the action:

```
elasticfilesystem
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "elasticfilesystem:action1",
      "elasticfilesystem:action2"
```

```
        ]
```

To view examples of Amazon EFS identity-based policies, see Identity-based policy examples for Amazon Elastic File System.

Policy resources for Amazon EFS

Supports policy resources: Yes

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. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Amazon EFS resource types and their ARNs, see Resources defined by Amazon Elastic File System in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon Elastic File System.

To view examples of Amazon EFS identity-based policies, see Identity-based policy examples for Amazon Elastic File System.

Policy condition keys for Amazon EFS

Supports service-specific policy condition keys: Yes

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 (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. 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.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Amazon EFS condition keys, see Condition keys for Amazon Elastic File System in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon Elastic File System.

To view examples of Amazon EFS identity-based policies, see Identity-based policy examples for Amazon Elastic File System.

ACLs in Amazon EFS

Supports ACLs: No

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.

ABAC with Amazon EFS

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Amazon EFS

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Amazon EFS

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).

Service roles for Amazon EFS

Supports service roles: Yes

A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Create a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon EFS functionality. Edit service roles only when Amazon EFS provides guidance to do so.

Service-linked roles for Amazon EFS

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Amazon EFS service-linked roles, see [Using service-linked roles for Amazon EFS](#).

Identity-based policy examples for Amazon Elastic File System

By default, users and roles don't have permission to create or modify Amazon EFS resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see [Create IAM policies (console)](#) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon EFS, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Elastic File System in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the Amazon EFS console
- Example: Allow users to view their own permissions
- Example: Enforce the creation of encrypted file systems
- Example: Enforce the creation of unencrypted file systems

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon EFS 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 AWS 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 Amazon EFS console

To access the Amazon Elastic File System console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon EFS 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 (users or roles) with that policy.

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 they're trying to perform.

To ensure that users and roles can still use the Amazon EFS console, also attach the Amazon EFS `AmazonElasticFileSystemReadOnlyAccess` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

You can see the `AmazonElasticFileSystemReadOnlyAccess` and other Amazon EFS managed service policies in AWS managed policies for Amazon EFS.

Example: 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": "*"
        }
    ]
}
```

Example: Enforce the creation of encrypted file systems

The following example illustrates an identity-based policy that authorizes principals to create only encrypted file systems.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticfilesystem:CreateFileSystem",
            "Condition": {
```

```
                "Bool": {
                    "elasticfilesystem:Encrypted": "true"
                }
            },
            "Resource": "*"
        }
    ]
}
```

If this policy is assigned to a user who tries to create an unencrypted file system, the request fails. The user sees a message similar to the following, whether they are using the AWS Management Console, the AWS CLI, or the AWS API or SDK:

```
User: arn:aws:iam::111122223333:user/username is not authorized to
      perform: elasticfilesystem:CreateFileSystem on the specified resource.
```

Example: Enforce the creation of unencrypted file systems

The following example illustrates an identity-based policy that authorizes principals to create only unencrypted file systems.

```
{
      "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticfilesystem:CreateFileSystem",
            "Condition": {
                "Bool": {
                    "elasticfilesystem:Encrypted": "false"
                }
            },
            "Resource": "*"
        }
    ]
}
```

If this policy is assigned to a user who tries to create an encrypted file system, the request fails. The user sees a message similar to the following, whether they are using the AWS Management Console, the AWS CLI, or the AWS API or SDK:

```
User: arn:aws:iam::111122223333:user/username is not authorized to
```

```
    perform: elasticfilesystem:CreateFileSystem on the specified resource.
```

You can also enforce the creation of encrypted or unencrypted EFS file systems by creating an AWS Organizations service control policy (SCP). For more information about service control policies in AWS Organizations, see Service control policies in the *AWS Organizations User Guide*.

Resource-based policy examples for Amazon EFS

In this section, you can find example file system policies that grant or deny permissions for various Amazon EFS actions. EFS file system policies have a 20,000 character limit. For information about the elements of a resource-based policy, see Resource-based policies within Amazon EFS.

> **Important**
>
> If you grant permission to an individual IAM user or role in a file system policy, don't delete or recreate that user or role while the policy is in effect on the file system. If this happens, that user or role is effectively locked out from file system and will not be able to access it. For more information, see Specifying a Principal in the *IAM User Guide*.

For information about how to create a file system policy, see Creating file system policies.

Topics

- Example: Grant read and write access to a specific AWS role
- Example: Grant read-only access
- Example: Grant access to an EFS access point

Example: Grant read and write access to a specific AWS role

In this example, the EFS file system policy has the following characteristics:

- The effect is `Allow`.
- The principal is set to the Testing_Role in the AWS account.
- The action is set to `ClientMount` (read), and `ClientWrite`.
- The condition for granting permissions is set to `AccessedViaMountTarget`.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/Testing_Role"
            },
            "Action": [
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientMount"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-2:111122223333:file-system/
fs-1234abcd",
            "Condition": {
                "Bool": {
                    "elasticfilesystem:AccessedViaMountTarget": "true"
                }
            }
        }
    ]
}
```

Example: Grant read-only access

The following file system policy only grants `ClientMount`, or read-only, permissions to the `EfsReadOnly` IAM role.

```
{
    "Id": "read-only-example-policy02",
    "Statement": [
        {
            "Sid": "efs-statement-example02",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/EfsReadOnly"
            },
            "Action": [
                "elasticfilesystem:ClientMount"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-2:111122223333:file-system/
fs-12345678"
        }
    ]
```

```
}
```

To learn how to set additional file system policies, including denying root access to all IAM principals, except for a specific management workstation, see Enable root squashing using IAM authorization for NFS clients.

Example: Grant access to an EFS access point

You use an EFS access policy to provide an NFS client with an application-specific view into shared file-based datasets on an EFS file system. You grant the access point permissions on the file system using a file system policy.

This file policy example uses a condition element to grant a specific access point that is identified by its ARN full access to the file system.

For more information about using EFS access points, see Working with access points.

```
{
    "Id": "access-point-example03",
    "Statement": [
        {
            "Sid": "access-point-statement-example03",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::555555555555:role/
EfsAccessPointFullAccess"},
            "Action": "elasticfilesystem:Client*",
            "Resource": "arn:aws:elasticfilesystem:us-east-2:111122223333:file-system/
fs-12345678",
            "Condition": {
                "StringEquals": {
                    "elasticfilesystem:AccessPointArn":"arn:aws:elasticfilesystem:us-
east-2:555555555555:access-point/fsap-12345678" }
            }
        }
    ]
}
```

AWS managed policies for Amazon EFS

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSServiceRoleForAmazonElasticFileSystem

Amazon EFS uses the service-linked role named `AWSServiceRoleForAmazonElasticFileSystem` to allow Amazon EFS to manage AWS resources on your behalf. This role trusts the `elasticfilesystem.amazonaws.com` service to assume the role. For more information, see Using service-linked roles for Amazon EFS.

AWS managed policy: AmazonElasticFileSystemFullAccess

You can attach the `AmazonElasticFileSystemFullAccess` policy to your IAM identities.

This policy grants administrative permissions that allow full access to Amazon EFS and access to related AWS services via the AWS Management Console.

Permissions details

This policy includes the following permissions.

- `elasticfilesystem` – Allows principals to perform all actions in the Amazon EFS console. It also allows principals to create (`elasticfilesystem:Backup`) and restore (`elasticfilesystem:Restore`) backups using AWS Backup.
- `cloudwatch` – Allows principals to describe Amazon CloudWatch file system metrics and alarms for a metric in the Amazon EFS console.
- `ec2` – Allows principals to create, delete, and describe network interfaces, describe and modify network interface attributes, describe Availability Zones, security groups, subnets, virtual private clouds (VPCs) and VPC attributes associated with an EFS file system in the Amazon EFS console.

- `kms` – Allows principals to list aliases for AWS Key Management Service (AWS KMS) keys and to describe KMS keys in the Amazon EFS console.
- `iam` – Grants permission to create a service linked role that allows Amazon EFS to manage AWS resources on the user's behalf.
- `iam:PassRole` – Grants permission to pass an IAM role to Amazon EFS.

To view the permissions for this policy, see AmazonElasticFileSystemFullAccess in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonElasticFileSystemReadOnlyAccess

You can attach the `AmazonElasticFileSystemReadOnlyAccess` policy to your IAM identities.

This policy grants read only access to Amazon EFS via the AWS Management Console.

Permissions details

This policy includes the following permissions.

- `elasticfilesystem` – Allows principals to describe attributes of Amazon EFS file systems, including account preferences, backup and file system policies, lifecycle configuration, mount targets and their security groups, tags, and access points in the Amazon EFS console.
- `cloudwatch` – Allows principals to retrieve CloudWatch metrics and describe alarms for metrics in the Amazon EFS console.
- `ec2` – Allows principals to view Availability Zones, network interfaces and their attributes, security groups, subnets, VPCs and their attributes in the Amazon EFS console.
- `kms` – Allows principals to list aliases for AWS KMS keys in the Amazon EFS console.

To view the permissions for this policy, see AmazonElasticFileSystemReadOnlyAccess in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonElasticFileSystemClientFullAccess

You can attach the `AmazonElasticFileSystemClientFullAccess` policy to an IAM entity.

This policy grants read and write client access to EFS file systems. This policy allows NFS clients to mount, read and write to EFS file systems.

To view the permissions for this policy, see AmazonElasticFileSystemFullAccess in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonElasticFileSystemClientReadWriteAccess

You can attach the `AmazonElasticFileSystemClientReadWriteAccess` policy to an IAM entity.

This policy grants read and write client access to EFS file systems. This policy allows NFS clients to mount, read and write to EFS file systems.

To view the permissions for this policy, see AmazonElasticFileSystemClientReadWriteAccess in the *AWS Managed Policy Reference Guide*.

Amazon EFS updates to AWS managed policies

View details about updates to AWS managed policies for Amazon EFS since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon EFS Document history page.

Change	Description	Date
Update to an existing policy	Policy: AmazonElasticFileSystemFullAccess Amazon EFS added the following: • `ReplicationRead` and `ReplicationWrite` to give permission to read and write file system data for replication. • `iam:PassRole` to give permission for Amazon EFS to create replication configurations.	November 7, 2024
Update to an existing policy	Policy: AmazonElasticFileSystemServiceRolePolicy Amazon EFS added `ReplicationRead` and `ReplicationWrite` to give permission to read and write file system data for replication.	November 7, 2024
Update to an existing policy	Policy: AmazonElasticFileSystemReadOnlyAccess	November 7, 2024

Change	Description	Date
	Amazon EFS added the `ReplicationRead` action to give permission to read file system data for replication.	
Update to an existing policy	Policy: AmazonElasticFileSystemReadOnlyAccess Amazon EFS added new permissions that give source and destination accounts access to file systems for cross-account replications.	August 7, 2024
Update to an existing policy	Policy: AmazonElasticFileSystemFullAccess Amazon EFS added a new permission to allow principals to disable and enable protection on a file system. The permissions are required to allow Amazon EFS to replicate to an existing file system.	November 27, 2023
Update to an existing policy	Policy: AmazonElasticFileSystemServiceRolePolicy Amazon EFS added new permissions to allow principals to create, describe, and delete Amazon EFS replications, and to create Amazon EFS file systems. The permissions are required to allow Amazon EFS to manage files system replication configurations on the user's behalf.	January 25, 2022
Update to an existing policy	Policy: AmazonElasticFileSystemReadOnlyAccess Amazon EFS added a new permission to allow principals to describe Amazon EFS replications. The permissions are required to allow users to view files system replication configurations.	January 25, 2022

Change	Description	Date
Update to an existing policy	Policy: AmazonElasticFileSystemFullAccess Amazon EFS added new permissions to allow principals to create, describe, and delete Amazon EFS replications. The permissions are required to allow users to manage files system replication configurations.	January 25, 2022
Started tracking policy	Policy: AmazonElasticFileSystemClientReadWriteAccess Grants read and write privileges on Amazon EFS file systems to NFS clients.	January 3, 2022
Started tracking policy	Policy: AmazonElasticFileSystemServiceRolePolicy The service-linked role permissions for Amazon EFS.	October 8, 2021
Update to an existing policy	Policy: AmazonElasticFileSystemFullAccess Amazon EFS added new permissions to allow principals to modify and describe Amazon EFS account preferences. The permissions are required to allow users to view and set account preferences settings in the Amazon EFS console.	May 7, 2021
Update to an existing policy	Policy: AmazonElasticFileSystemReadOnlyAccess Amazon EFS added new permissions to allow principals to describe Amazon EFS account preferences. The permissions are required to allow users to view account preferences settings in the Amazon EFS console.	May 7, 2021
Amazon EFS started tracking changes	Amazon EFS started tracking changes for its AWS managed policies.	May 7, 2021

Using tags with Amazon EFS

You can use tags to control access to Amazon EFS resources and to implement attribute-based access control (ABAC). For more information, see:

- Tagging EFS resources
- Controlling access based on tags on a resource
- What is ABAC for AWS? in the *IAM User Guide*

> **Note**
>
> Amazon EFS replication does not support using tags for attribute-based access control (ABAC).

To apply tags to Amazon EFS resources during creation, users must have certain AWS Identity and Access Management (IAM) permissions.

Granting permissions to tag resources during creation

The following tag-on create Amazon EFS API actions allow you to specify tags when you create the resource.

- `CreateAccessPoint`
- `CreateFileSystem`

To enable users to tag resources on creation, they must have permissions to use the action that creates the resources, such as `elasticfilesystem:CreateAccessPoint` or `elasticfilesystem:CreateFileSystem`. If tags are specified in the resource-creating action, AWS performs additional authorization on the `elasticfilesystem:TagResource` action to verify if users have permission to create tags. Therefore, users must also have explicit permissions to use the `elasticfilesystem:TagResource` action.

In the IAM policy definition for the `elasticfilesystem:TagResource` action, use the `Condition` element with the `elasticfilesystem:CreateAction` condition key to give tagging permissions to the action that creates the resource.

Example policy: Allow adding tags to file systems only at the time of creation

The following example policy allows users to create file systems and apply tags to them only during creation. Users are not permitted to tag any existing resources (they cannot call the `elasticfilesystem:TagResource` action directly).

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "elasticfilesystem:CreateFileSystem"
      ],
      "Resource": "arn:aws:elasticfilesystem:region:account-id:file-system/*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "elasticfilesystem:TagResource"
      ],
      "Resource": "arn:aws:elasticfilesystem:region:account-id:file-system/*",
      "Condition": {
         "StringEquals": {
             "elasticfilesystem:CreateAction": "CreateFileSystem"
          }
       }
    }
  ]
}
```

Using tags to control access to your Amazon EFS resources

To control access to Amazon EFS resources and actions, you can use IAM policies based on tags. You can provide this control in two ways:

- You can control access to Amazon EFS resources based on the tags on those resources.
- You can control which tags can be passed in an IAM request condition.

For information about how to use tags to control access to AWS resources, see Controlling access using tags in the *IAM User Guide*.

Controlling access based on tags on a resource

To control which actions a user or role can perform on an Amazon EFS resource, you can use tags on the resource. For example, you might want to allow or deny specific API operations on a file system resource based on the key-value pair of the tag on the resource.

Example policy: Create a file system only when a specific tag is used

The following example policy allows the user to create a file system only when they tag it with a specific tag key-value pair, in this example, `key=Department`, `value=Finance`.

```
{
    "Effect": "Allow",
    "Action": [
        "elasticfilesystem:CreateFileSystem",
        "elasticfilesystem:TagResource"
    ],
    "Resource": "arn:aws:elasticfilesystem:region:account-id:file-system/*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Department": "Finance"
        }
    }
}
```

Example policy: Delete file systems with specific tags

The following example policy allows a user to delete only file systems that are tagged with `Department=Finance`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:DeleteFileSystem"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/*",
```

```
                "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/Department": "Finance"
                    }
                }
            }
        ]
}
```

Using service-linked roles for Amazon EFS

Amazon Elastic File System uses an AWS Identity and Access Management (IAM) service-linked role. The Amazon EFS service-linked role is a unique type of IAM role that is linked directly to Amazon EFS. The predefined Amazon EFS service-linked role includes permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon EFS easier because you don't have to manually add the necessary permissions. Amazon EFS defines the permissions of its service-linked role, and only Amazon EFS can assume its role. 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 the Amazon EFS service-linked role only after first deleting your Amazon EFS file systems. This protects your Amazon EFS resources because you can't inadvertently remove permission to access the resources.

The service-linked role enables all API calls to be visible through AWS CloudTrail. This helps with monitoring and auditing requirements because you can track all actions that Amazon EFS performs on your behalf. For more information, see Log entries for EFS service-linked roles.

For more information, see Service-linked role permissions in the *IAM User Guide*.

Service-linked role permissions for Amazon EFS

Amazon EFS uses the service-linked role named **AWSServiceRoleForAmazonElasticFileSystem** to allow Amazon EFS to call and manage AWS resources on behalf of your EFS file systems.

The AWSServiceRoleForAmazonElasticFileSystem service-linked role trusts the `elasticfilesystem.amazonaws.com` to assume the role.

The role permissions policy allows Amazon EFS to complete the actions included in the policy definition JSON:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "backup-storage:MountCapsule",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:ModifyNetworkInterfaceAttribute",
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:*:*:key/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "backup:CreateBackupVault",
                "backup:PutBackupVaultAccessPolicy"
            ],
            "Resource": [
                "arn:aws:backup:*:*:backup-vault:aws/efs/automatic-backup-vault"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "backup:CreateBackupPlan",
                "backup:CreateBackupSelection"
            ],
            "Resource": [
```

```
                "arn:aws:backup:*:*:backup-plan:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "backup.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/backup.amazonaws.com/
AWSServiceRoleForBackup"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "backup.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:DescribeFileSystems",
                "elasticfilesystem:CreateReplicationConfiguration",
                "elasticfilesystem:DescribeReplicationConfigurations",
                "elasticfilesystem:DeleteReplicationConfiguration",
                "elasticfilesystem:ReplicationRead",
                "elasticfilesystem:ReplicationWrite"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

> **Note**
>
> You must manually configure IAM permissions for AWS KMS when creating a new EFS file system that is encrypted at rest. To learn more, see Encrypting data at rest.

You must configure permissions to allow an IAM entity (such as a user, group, or role) 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 Amazon EFS

In most cases, you don't need to manually create a service-linked role. When you create mount targets or a replication configuration for your EFS file system in the AWS Management Console, the AWS CLI, or the AWS API, Amazon EFS creates the service-linked role for you.

Additionally, if you manually 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 mount targets or a replication configuration for your EFS file system, Amazon EFS creates the service-linked role for you.

If, however, Amazon EFS does not create the service-linked-role or if you started using Amazon EFS before it supported service-linked roles, then you can manually create the service-linked role. For instructions, see Creating a service-linked role in the *IAM User Guide*.

Editing a service-linked role for Amazon EFS

Amazon EFS doesn't allow you to edit the `AWSServiceRoleForAmazonElasticFileSystem` service-linked role. After you create a service-linked role, you cannot 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 Update a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for Amazon EFS

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. For more information, see Clean up resources and protect your AWS account.

> **Note**
>
> If the Amazon EFS service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForAmazonElasticFileSystem service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for Amazon EFS service-linked roles

Amazon EFS supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see AWS service endpoints in the *AWS General Reference User Guide*.

Troubleshooting Amazon Elastic File System identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon EFS and IAM.

Topics

- I am not authorized to perform an action in Amazon EFS
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon EFS resources

I am not authorized to perform an action in Amazon EFS

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `elasticfilesystem:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 elasticfilesystem:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `elasticfilesystem:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

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 Amazon EFS.

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 Amazon EFS. 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.

I want to allow people outside of my AWS account to access my Amazon EFS 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 Amazon EFS supports these features, see How Amazon Elastic File System 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*.

Using IAM to control access to file systems

You can use both IAM identity policies and resource policies to control client access to Amazon EFS resources in a way that is scalable and optimized for cloud environments. Using IAM, you can permit clients to perform specific actions on a file system, including read-only, write, and root access. An "allow" permission on an action in *either* an IAM identity policy *or* a file system resource policy allows access for that action. The permission does not need to be granted in *both* an identity *and* a resource policy.

NFS clients can identify themselves using an IAM role when connecting to an EFS file system. When a client connects to a file system, Amazon EFS evaluates the file system's IAM resource policy, which is called a file system policy, along with any identity-based IAM policies to determine the appropriate file system access permissions to grant.

When you use IAM authorization for NFS clients, client connections and IAM authorization decisions are logged to AWS CloudTrail. For more information about how to log Amazon EFS API calls with CloudTrail, see Logging Amazon EFS API calls with AWS CloudTrail.

> **Important**
>
> You must use the EFS mount helper to mount your EFS file systems in order to use IAM authorization to control client access. For more information, see Mounting with IAM authorization.

Default EFS file system policy

The default EFS file system policy does not use IAM to authenticate, and grants full access to any anonymous client that can connect to the file system using a mount target. The default policy is in effect whenever a user-configured file system policy is not in effect, including at file system creation. Whenever the default file system policy is in effect, a `DescribeFileSystemPolicy` API operation returns a `PolicyNotFound` response.

EFS actions for clients

You can specify the following actions for clients accessing a file system using a file system policy.

Action	Description
`elasticfilesystem:ClientMount`	Provides read-only access to a file system.
`elasticfilesystem:ClientWrite`	Provides write permissions on a file system.
`elasticfilesystem:ClientRootAccess`	Provides use of the root user when accessing a file system.

EFS condition keys for clients

To express conditions, you use predefined condition keys. Amazon EFS has the following predefined condition keys for NFS clients. Any other condition keys are not enforced when using IAM controls to secure access to EFS file systems.

EFS Condition Key	Description	Operator
`aws:SecureTransport`	Use this key to require clients to use TLS when connecting to an EFS file system.	Boolean
`aws:SourceIp`	Use this key to compare the requester's IP address with the IP address that you specify in the policy. The `aws:SourceIp` condition	String

EFS Condition Key	Description	Operator
	key can only be used for public IP address ranges.	
`elasticfilesystem:AccessPointArn`	ARN of the EFS access point to which the client is connecting.	String
`elasticfilesystem:AccessedViaMountTarget`	Use this key to prevent access to an EFS file system by clients that are not using file system mount targets.	Boolean

File system policy examples

To view examples of Amazon EFS file system policies, see Resource-based policy examples for Amazon EFS.

Controlling network access to EFS file systems for NFS clients

You can control access by NFS clients to Amazon EFS file systems using network layer security and EFS file system policies. You can use the network layer security mechanisms available with Amazon EC2, such as VPC security group rules and network ACLs. You can also use AWS IAM to control NFS access with an EFS file system policy and identity-based policies.

Topics

- Using VPC security groups
- Working with interface VPC endpoints in Amazon EFS

Using VPC security groups

When using Amazon EFS, you specify VPC security groups for your EC2 instances and security groups for the EFS mount targets associated with the file system. A security group acts as a firewall, and the rules that you add define the traffic flow. In the Getting started exercise, you created one security group when you launched the EFS instance. You then associated another with the EFS mount target (that is, the default security group for your default VPC). That approach works for the Getting started exercise. However, for a production system, you should set up security groups with minimal permissions for use with Amazon EFS.

You can authorize inbound and outbound access to your EFS file system. To do so, you add rules that allow EFS instances to connect to your EFS file system through the mount target using the Network File System (NFS) port.

- Each EC2 instance that mounts the file system must have a security group with a rule that allows outbound access to the mount target on the NFS port.
- The EFS mount target needs to have a security group with a rule that allows inbound access from each EC2 instance on which you want to mount the file system.

Source ports for working with Amazon EFS

To support a broad set of NFS clients, Amazon EFS allows connections from any source port. If you require that only privileged users can access Amazon EFS, we recommend using the following client firewall rule. Connect to your file system using SSH and run the following command:

```
iptables -I OUTPUT 1 -m owner --uid-owner 1-4294967294 -m tcp -p tcp --dport 2049 -j
 DROP
```

This command inserts a new rule at the start of the OUTPUT chain (`-I OUTPUT 1`). The rule prevents any unprivileged, nonkernel process (`-m owner --uid-owner 1-4294967294`) from opening a connection to the NFS port (`-m tcp -p tcp –dport 2049`).

Security considerations for network access

An NFS version 4.1 (NFSv4.1) client can only mount a file system if it can make a network connection to the NFS port (TCP port 2049) of one of the file system's mount targets. Similarly, an NFSv4.1 client can only assert a user and group ID when accessing a file system if it can make this network connection.

Whether you can make this network connection is governed by a combination of the following:

- **Network isolation provided by the mount targets' VPC** – File system mount targets can't have public IP addresses associated with them. The only targets that can mount file systems are the following:
  - Amazon EC2 instances in the local Amazon VPC
  - EC2 instances in connected VPCs

  - On-premises servers connected to an Amazon VPC by using AWS Direct Connect and an AWS Virtual Private Network (VPN)
- **Network access control lists (ACLs) for the VPC subnets of the client and mount targets, for access from outside the mount target's subnets** – To mount a file system, the client must be able to make a TCP connection to the NFS port of a mount target and receive return traffic.
- **Rules of the client's and mount targets' VPC security groups, for all access** – For an EC2 instance to mount a file system, the following security group rules must be in effect:
  - The file system must have a mount target whose network interface has a security group with a rule that enables inbound connections on the NFS port from the instance. You can enable inbound connections either by IP address (CIDR range) or security group. The source of the security group rules for the inbound NFS port on mount target network interfaces is a key element of file system access control. Inbound rules other than the one for the NFS port, and any outbound rules, aren't used by network interfaces for file system mount targets.
  - The mounting instance must have a network interface with a security group rule that enables outbound connections to the NFS port on one of the file system's mount targets. You can enable outbound connections either by IP address (CIDR range) or security group.

For more information, see Managing mount targets.

Creating security groups

To create security groups for EC2 instances and EFS mount targets

The following are the general steps that you'll perform when creating the security groups for Amazon EFS. For instructions on creating the security groups, see Create a security group in the *Amazon VPC User Guide*.

1. For your EC2 instances, create a security group with the following rules:
   - An inbound rule that allows inbound access that allows using Secure Shell (SSH) from any host. Optionally, restrict the **Source** address.
   - An outbound rule that allows all traffic to leave. When you create a security group, it is created with an outbound rule by default so you shouldn't need to add one.
2. For your EFS mount target, create a security group with the following rules:
   - An inbound rule that allows access from the EC2 security group. Identify the EC2 security group as the source.

- An outbound rule to open the TCP connection on all of the NFS ports. Identify the EC2 security group as the destination.

Working with interface VPC endpoints in Amazon EFS

To establish a private connection between your virtual private cloud (VPC) and the Amazon EFS API, you can create an interface VPC endpoint. The endpoint provides secure connectivity to the Amazon EFS API without requiring an internet gateway, NAT instance, or virtual private network (VPN) connection. For more information, see Access an AWS service using an interface VPC endpoint in the *Amazon VPC User Guide*.

Interface VPC endpoints are powered by AWS PrivateLink, a feature that enables private communication between AWS services using private IP addresses. To use AWS PrivateLink, create an interface VPC endpoint for Amazon EFS in your VPC using the Amazon VPC console, API, or CLI. Doing this creates an elastic network interface in your subnet with a private IP address that serves Amazon EFS API requests. You can also access a VPC endpoint from on-premises environments or from other VPCs using AWS VPN, AWS Direct Connect, or VPC peering. To learn more, see Connect your VPC to services using AWS PrivateLink in the *Amazon VPC User Guide*.

Creating an interface endpoint for Amazon EFS

To create an interface VPC endpoint for Amazon EFS, use one of the following:

- **`com.amazonaws.`*`region`*`.elasticfilesystem`** – Creates an endpoint for Amazon EFS API operations.
- **`com.amazonaws.`*`region`*`.elasticfilesystem-fips`** – Creates an endpoint for the Amazon EFS API that complies with Federal Information Processing Standard (FIPS) 140-2.

For a complete list of Amazon EFS endpoints, see Amazon Elastic File System endpoints and quotas in the *Amazon Web Services General Reference*.

For more information about how to create an interface endpoint, see Access an AWS service using an interface VPC endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for Amazon EFS

To control access to the Amazon EFS API, you can attach an AWS Identity and Access Management (IAM) policy to your VPC endpoint. The policy specifies the following:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Control access to VPC endpoints using endpoint policies in the *Amazon VPC User Guide*.

The following example shows a VPC endpoint policy that denies everyone permission to create an EFS file system through the endpoint. The example policy also grants everyone permission to perform all other actions.

```
{
   "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "elasticfilesystem:CreateFileSystem",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

Network File System (NFS) level users, groups, and permissions

After creating a file system, by default only the root user (UID 0) has read, write, and execute permissions. For other users to modify the file system, the root user must explicitly grant them access. You can use access points to automate the creation of directories that a nonroot user can write from. For more information, see Working with access points.

EFS file system objects have a Unix-style mode associated with them. This mode value defines the permissions for performing actions on that object. Users familiar with Unix-style systems can easily understand how Amazon EFS behaves with respect to these permissions.

Additionally, on Unix-style systems, users and groups are mapped to numeric identifiers, which Amazon EFS uses to represent file ownership. For Amazon EFS, file system objects (that is, files, directories, and so on) 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.

> **Note**
>
> The NFS protocol supports a maximum of 16 group IDs (GIDs) per user and any additional GIDs are truncated from NFS client requests. For more information, see Access denied to allowed files on NFS file system.

Following, you can find examples of permissions and a discussion about NFS permissions considerations for Amazon EFS.

Topics

- File and directory permissions
- Example EFS file system use cases and permissions
- User and group ID permissions for files and directories within a file system
- No root squashing
- Permissions caching
- Changing file system object ownership
- EFS access points

File and directory permissions

Files and directories in an EFS file system support standard Unix-style read, write, and execute permissions based on the user and group ID asserted by the mounting NFSv4.1 client, unless overridden by an EFS access point. For more information, see Network File System (NFS) level users, groups, and permissions.

> **Note**
>
> By default, this layer of access control depends on trusting the NFSv4.1 client in its assertion of the user and group ID. You can use AWS Identity and Access Management

(IAM) resource-based policies and identity policies to authorize NFS clients and provide read-only, write, and root access permissions. You can use EFS access points to override the operating system user and group identity information provided by the NFS client. For more information, see Using IAM to control access to file systems and Creating access points.

As an example of read, write, and execute permissions for files and directories, Alice might have permissions to read and write to any files that she wants to in her personal directory on a file system, `/alice`. However, in this example Alice is not allowed to read or write to any files in Mark's personal directory on the same file system, `/mark`. Both Alice and Mark are allowed to read but not write files in the shared directory `/share`.

Example EFS file system use cases and permissions

After you create an EFS file system and mount targets for the file system in your VPC, you can mount the remote file system locally on your Amazon EC2 instance. The `mount` command can mount any directory in the file system. However, when you first create the file system, there is only one root directory at `/`. The root user and root group own the mounted directory.

The following `mount` command mounts the root directory of an Amazon EFS file system, identified by the file system DNS name, on the `/efs-mount-point` local directory.

```
sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport file-
system-id.efs.aws-region.amazonaws.com:/ efs-mount-point
```

The initial permissions mode allows:

- `read-write-execute` permissions to the owner *root*
- `read-execute` permissions to the group *root*
- `read-execute` permissions to others

Only the root user can modify this directory. The root user can also grant other users permissions to write to this directory, for example:

- Create writable per-user subdirectories. For step-by-step instructions, see Tutorial: Creating writable per-user subdirectories.

- Allow users to write to the EFS file system root. A user with root privileges can grant other users access to the file system.
  - To change the EFS file system ownership to a non-*root* user and group, use the following:

    ```
    $ sudo chown user:group /EFSroot
    ```

  - To change permissions of the file system to something more permissive, use the following:

    ```
    $ sudo chmod 777 /EFSroot
    ```

    This command grants read-write-execute privileges to all users on all EC2 instances that have the file system mounted.

User and group ID permissions for files and directories within a file system

Files and directories in an EFS file system support standard Unix-style read, write, and execute permissions based on the user ID and group IDs. When an NFS client mounts an EFS file system without using an access point, the user ID and group ID provided by the client is trusted. You can use EFS access points to override user ID and group IDs used by the NFS client. When users attempt to access files and directories, Amazon EFS checks their user IDs and group IDs to verify that each user has permission to access the objects. Amazon EFS also uses these IDs to indicate the owner and group owner for new files and directories that the user creates. Amazon EFS doesn't examine user or group names—it only uses the numeric identifiers.

> **Note**
>
> When you create a user on an EC2 instance, you can assign any numeric user ID (UID) and group ID (GID) to the user. The numeric user IDs are set in the `/etc/passwd` file on Linux systems. The numeric group IDs are in the `/etc/group` file. These files define the mappings between names and IDs. Outside of the EC2 instance, Amazon EFS doesn't perform any authentication of these IDs, including the root ID of 0.

If a user accesses an EFS file system from two different EC2 instances, depending on whether the UID for the user is the same or different on those instances you see different behavior, as follows:

- If the user IDs are the same on both EC2 instances, Amazon EFS considers them to indicate the same user, regardless of the EC2 instance used. The user experience when accessing the file system is the same from both EC2 instances.
- If the user IDs aren't the same on both EC2 instances, Amazon EFS considers the users to be different users. The user experience isn't the same when accessing the EFS file system from the two different EC2 instances.
- If two different users on different EC2 instances share an ID, Amazon EFS considers them to be the same user.

You might consider managing user ID mappings across EC2 instances consistently. Users can check their numeric ID using the `id` command.

```
$ id

uid=502(joe) gid=502(joe) groups=502(joe)
```

Turn Off the ID Mapper

The NFS utilities in the operating system include a daemon called an ID Mapper that manages mapping between user names and IDs. In Amazon Linux, the daemon is called `rpc.idmapd` and on Ubuntu is called `idmapd`. It translates user and group IDs into names, and vice versa. However, Amazon EFS deals only with numeric IDs. We recommend that you turn this process off on your EC2 instances. On Amazon Linux, the ID mapper is usually disabled, and if it is don't enable it. To turn off the ID mapper, use the commands shown following.

```
$  service rpcidmapd status
$  sudo service rpcidmapd stop
```

No root squashing

By default, root squashing is disabled on EFS file systems. Amazon EFS behaves like a Linux NFS server with `no_root_squash`. If a user or group ID is 0, Amazon EFS treats that user as the `root` user, and bypasses permissions checks (allowing access and modification to all file system objects). Root squashing can be enabled on a client connection when the AWS Identity and Access Management (AWS IAM) identity or resource policy does not allow access to the `ClientRootAccess` action. When root squashing is enabled, the root user is converted to a user with limited permissions on the NFS server.

For more information, see Using IAM to control access to file systems.

Enable root squashing using IAM authorization for NFS clients

You can configure Amazon EFS to prevent root access to your EFS file system for all AWS principals except for a single management workstation. You do this by configuring AWS Identity and Access Management (IAM) authorization for Network File System (NFS) clients.

To do this requires configuring two IAM permissions policies, as follows:

- Create an EFS file system policy that explicitly allows read and write access to the file system, and implicitly denies root access.
- Assign an IAM identity to the Amazon EC2 management workstation that requires root access to the file system by using an EC2 instance profile. For more information about Amazon EC2 instance profiles, see Use instance profiles in the *AWS Identity and Access Management User Guide*.
- Assign the `AmazonElasticFileSystemClientFullAccess` AWS managed policy to the IAM role of the management workstation. For more information about AWS managed policies for Amazon EFS, see Identity and access management for Amazon EFS.

To enable root squashing using IAM authorization for NFS clients, use the following procedures.

To prevent root access to the file system

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.
2. Choose **File systems**.
3. Choose the file system that you want to enable root squashing on.
4. On the file system details page, choose **File system policy**, and then choose **Edit**. The **File system policy** page appears.
5. Choose **Prevent root access by default*** under **Policy options**. The policy JSON object appears in the **Policy editor**.
6. Choose **Save** to save the file system policy.

Clients that aren't anonymous can get root access to the file system through an identity-based policy. When you attach the `AmazonElasticFileSystemClientFullAccess` managed policy to the workstation's role, IAM grants root access to the workstation based on its identity policy.

To enable root access from the management workstation

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. Create a role for Amazon EC2 called `EFS-client-root-access`. IAM creates an instance profile with the same name as the EC2 role you created.
3. Assign the AWS managed policy `AmazonElasticFileSystemClientFullAccess` to the EC2 role you created. The contents of this policy is shown following.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientRootAccess",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:DescribeMountTargets"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Attach the instance profile to the EC2 instance that you are using as the management workstation, as described following. For more information, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide for Linux Instances*.
   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   b. In the navigation pane, choose **Instances**.
   c. Choose the instance. For **Actions**, choose **Instance Settings**, and then choose **Attach/ Replace IAM role**.
   d. Choose the IAM role that you created in the first step, `EFS-client-root-access`, and choose **Apply**.

5. Install the EFS mount helper on the management workstation. For more information about the EFS mount helper mount helper and the amazon-efs-utils package, see Installing the Amazon EFS client.

6. Mount the EFS file system on the management workstation by using the following command with the `iam` mount option.

   ```
   $ sudo mount -t efs -o tls,iam file-system-id:/ efs-mount-point
   ```

   You can configure the Amazon EC2 instance to automatically mount the file system with IAM authorization. For more information about mounting an EFS file system with IAM authorization, see Mounting with IAM authorization.

Permissions caching

Amazon EFS caches file permissions for a small time period. As a result, there might be a brief window where a user whose access was revoked recently can still access that object.

Changing file system object ownership

Amazon EFS enforces the POSIX `chown_restricted` attribute. This means only the root user can change the owner of a file system object. The root or the owner user can change the owner group of a file system object. However, unless the user is root, the group can only be changed to one that the owner user is a member of.

EFS access points

An *access point* applies an operating system user, group, and file system path to any file system request made using the access point. The access point's operating system user and group override any identity information provided by the NFS client. The file system path is exposed to the client as the access point's root directory. This approach ensures that each application always uses the correct operating system identity and the correct directory when accessing shared file-based datasets. Applications using the access point can only access data in its own directory and below. For more information about access points, see Working with access points.

Working with access points

EFS *access points* are application-specific entry points into an EFS file system that make it easier to manage application access to shared datasets. Access points can enforce a user identity, including the user's POSIX groups, for all file system requests that are made through the access point. Access points can also enforce a different root directory for the file system so that clients can only access data in the specified directory or its subdirectories.

You can use AWS Identity and Access Management (IAM) policies to enforce that specific applications use a specific access point. By combining IAM policies with access points, you can easily provide secure access to specific datasets for your applications.

> **Note**
>
> You need to create at least one mount target on your EFS file system to use access points.

You can create access points for an existing EFS file system using the AWS Management Console, the AWS Command Line Interface (AWS CLI), and the Amazon EFS API. An EFS file system can have a maximum of 10,000 access points. You cannot modify an existing access point after it's created.

For step-by-step procedures to create an access point, see Creating access points.

You use the EFS mount helper when mounting a file system using an access point. In the mount command, include file system ID, the access point ID, and the `tls` mount option, as shown in the following example.

```
$ mount -t efs -o tls,iam,accesspoint=fsap-abcdef0123456789a fs-
abc0123def456789a: /localmountpoint
```

For more information on mounting file systems using an access point, see Mounting with EFS access points.

Topics

- Enforcing a user identity using an access point
- Enforcing a root directory with an access point
- Using access points in IAM policies

Enforcing a user identity using an access point

You can use an access point to enforce user and group information for all file system requests made through the access point. To enable this feature, you need to specify the operating system identity to enforce when you create the access point.

As part of this, you provide the following:

- User ID – The numeric POSIX user ID for the user.
- Group ID – The numeric POSIX group ID for the user.
- Secondary group IDs – An optional list of secondary group IDs.

When user enforcement is enabled, Amazon EFS replaces the NFS client's user and group IDs with the identity configured on the access point for all file system operations. User enforcement also does the following:

- The owner and group for new files and directories are set to the user ID and group ID of the access point.
- EFS considers the user ID, group ID, and secondary group IDs of the access point when evaluating file system permissions. EFS ignores the NFS client's IDs.

> **Important**
>
> Enforcing a user identity is subject to the `ClientRootAccess` IAM permission. For example, in some cases you might configure the access point user ID, group ID, or both to be root (that is, setting the UID, GID, or both to 0). In such cases, you must grant the `ClientRootAccess` IAM permission to the NFS client.

Enforcing a root directory with an access point

You can use an access point to override the root directory for a file system. When you enforce a root directory, the NFS client using the access point uses the root directory configured on the access point instead of the file system's root directory.

You enable this feature by setting the access point `Path` attribute when creating an access point. The `Path` attribute is the full path of the root directory of the file system for all file system requests made through this access point. The full path can't exceed 100 characters in length. It can include up to four subdirectories.

When you specify a root directory on an access point, it becomes the root directory of the file system for the NFS client mounting the access point. For example, suppose that the root directory of your access point is `/data`. In this case, mounting `fs-12345678:/` using the access point has the same effect as mounting `fs-12345678:/data` without using the access point.

When specifying a root directory in your access point, ensure that the directory permissions are configured to allow the user of the access point to successfully mount the file system. Specifically, make sure that the execute bit is set for the access point user or group, or for everyone. For example, a directory permission value of 755 allows the directory user owner to list files, create files, and mount, and all other users to list files and mount.

Creating the root directory for an access point

If a root directory path for an access point doesn't exist on the file system, Amazon EFS automatically creates that root directory with the ownership and permissions specified. Amazon EFS will not create the root directory if you do not specify the directory ownership and permissions at creation. This approach makes it possible to provision file system access for a specific user or application without mounting your file system from a Linux host. To create a root directory, you have to configure the root directory ownership and permission by using the following attributes when creating an access point:

- `OwnerUid` – The numeric POSIX user ID to use as the root directory owner.
- `OwnerGiD` – The numeric POSIX group ID to use as the root directory owner group.
- Permissions – The Unix mode of the directory. A common configuration is 755. Ensure that the execute bit is set for the access point user so they are able to mount. This configuration gives the directory owner permission to enter, list, and write new files in the directory. It gives all other users permission to enter and list files. For more information on working with Unix file and directory modes, see [Network File System (NFS) level users, groups, and permissions](#).

Amazon EFS creates an access point root directory only if the OwnUid, OwnGID, and permissions are specified for the directory. If you do not provide this information, Amazon EFS does not create the root directory. If the root directory does not exist, attempts to mount using the access point will fail.

When you mount a file system with an access point, the root directory for the access point is created if the directory doesn't already exist, provided that the root directory's OwnerUid and Permissions were specified when the access point was created. If the access point's root directory already exists before mount time, the existing permissions aren't overwritten by the access point. If you delete the root directory, EFS recreates it the next time that the file system is mounted using the access point.

> **Note**
>
> If you do not specify the ownership and permissions for an access point root directory, Amazon EFS will not create the root directory. All attempts to mount the access point will fail.

Security model for access point root directories

When a root directory override is in effect, Amazon EFS behaves like a Linux NFS server with the `no_subtree_check` option enabled.

In the NFS protocol, servers generate file handles that are used by clients as unique references when accessing files. EFS securely generates file handles that are unpredictable and specific to an EFS file system. When a root directory override is in place, Amazon EFS doesn't disclose file handles for files outside the specified root directory. However, in some cases a user might get a file handle for a file outside of their access point by using an out-of-band mechanism. For example, they might do so if they have access to a second access point. If they do this, they can perform read and write operations on the file.

File ownership and access permissions are always enforced, for access to files within and outside of a user's access point root directory.

Using access points in IAM policies

You can use an IAM policy to enforce that a specific NFS client, identified by its IAM role, can only access a specific access point. To do this, you use the `elasticfilesystem:AccessPointArn` IAM condition key. The `AccessPointArn` is the Amazon Resource Name (ARN) of the access point that the file system is mounted with.

Following is an example of a file system policy that allows the IAM role app1 to access the file system using access point fsap-01234567. The policy also allows app2 to use the file system using access point fsap-89abcdef.

JSON

```
{
    "Version": "2012-10-17",
    "Id": "MyFileSystemPolicy",
    "Statement": [
        {
            "Sid": "App1Access",
            "Effect": "Allow",
            "Principal": { "AWS": "arn:aws:iam::111122223333:role/app1" },
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/*",
            "Condition": {
                "StringEquals": {
                    "elasticfilesystem:AccessPointArn" :
 "arn:aws:elasticfilesystem:us-east-1:222233334444:access-point/fsap-01234567"
                }
            }
        },
        {
            "Sid": "App2Access",
            "Effect": "Allow",
            "Principal": { "AWS": "arn:aws:iam::111122223333:role/app2" },
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/*",
            "Condition": {
                "StringEquals": {
                    "elasticfilesystem:AccessPointArn" :
 "arn:aws:elasticfilesystem:us-east-1:222233334444:access-point/fsap-89abcdef"
                }
```

```
            }
        }
    ]
}
```

Blocking public access to EFS file systems

The Amazon EFS block public access feature provides settings to help you manage public access to EFS file systems. By default, new EFS file systems don't allow public access. However, you can modify file system policies to allow public access.

> **Important**
>
> Enabling Block Public Access access helps protect your resources by preventing public access from being granted through the resource policies that are directly attached to the file system. In addition to enabling Block Public Access, carefully inspect the following policies to confirm that they do not grant public access:
>
> - Identity-based policies attached to associated AWS principals (for example, IAM roles)
> - Resource-based policies attached to associated AWS resources (for example,AWS Key Management Service (KMS) keys)

Topics

- Blocking public access with AWS Transfer Family
- The meaning of "public"

Blocking public access with AWS Transfer Family

When you use Amazon EFS with AWS Transfer Family, file system access requests received from a Transfer Family server that is owned by a different account than the file system are blocked if the file system allows public access. Amazon EFS evaluates the file system's IAM policies, and if the policy is public, it blocks the request. To permit AWS Transfer Family access to your file system, update your file system policy so that it is not considered public.

> **Note**
>
> Using Transfer Family with Amazon EFS is disabled by default for AWS accounts that have EFS file systems with policies that allow public access that were created before January 6, 2021. To enable using Transfer Family to access your file system, contact AWS Support.

The meaning of "public"

When evaluating whether a file system allows public access, Amazon EFS assumes that the file system policy is public. It then evaluates the file system policy to determine if it qualifies as non-public. To be considered non-public, a file system policy must grant access only to fixed values (values that don't contain a wild card) of one or more of the following:

- A set of Classless Inter-Domain Routings (CIDRs), using `aws:SourceIp`. For more information about CIDR, see RFC 4632 on the RFC Editor website.
- An AWS principal, user, role, or service principal (for example, `aws:PrincipalOrgID`)
- `aws:SourceArn`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:SourceOwner`
- `aws:SourceAccount`
- `elasticfilesystem:AccessedViaMountTarget`
- `aws:userid, outside the pattern "AROLEID:*"`

Under these rules, the following example policy is considered public.

JSON

```
{
    "Version": "2012-10-17",
    "Id": "efs-policy-wizard-15ad9567-2546-4bbb-8168-5541b6fc0e55",
    "Statement": [
        {
            "Sid": "efs-statement-14a7191c-9401-40e7-a388-6af6cfb7dd9c",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientRootAccess"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/*"
            }
     ]
}
```

You can make this file system policy non-public by using the EFS condition key `elasticfilesystem:AccessedViaMountTarget` set to true. You can use `elasticfilesystem:AccessedViaMountTarget` to allow the specified EFS actions to clients accessing the EFS file system using a file system mount target. The following non-public policy uses the `elasticfilesystem:AccessedViaMountTarget` condition key set to true.

JSON

```
{
    "Version": "2012-10-17",
    "Id": "efs-policy-wizard-15ad9567-2546-4bbb-8168-5541b6fc0e55",
    "Statement": [
        {
            "Sid": "efs-statement-14a7191c-9401-40e7-a388-6af6cfb7dd9c",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite",
                "elasticfilesystem:ClientRootAccess"
            ],
            "Resource": "arn:aws:elasticfilesystem:us-east-1:111122223333:file-
system/*",
            "Condition": {
                "Bool": {
```

```
                    "elasticfilesystem:AccessedViaMountTarget": "true"
                }
            }
        }
    ]
}
```

For more information about Amazon EFS condition keys, see EFS condition keys for clients. For more information about creating file system policies, see Creating file system policies.

Compliance validation for Amazon EFS

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. AWS provides the following resources to help with compliance:

- Security Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.

- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Amazon EFS

The AWS global infrastructure is built around AWS Regions and Availability Zones (AZs). AWS Regions provide multiple physically separated and isolated AZs, which are connected with low-latency, high-throughput, and highly redundant networking. With AZs, you can design and operate applications and databases that automatically fail over between zones without interruption. AZs are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

Amazon EFS file systems are resilient to one or more Availability Zone failures within an AWS Region. Mount targets themselves are designed to be highly available. As you design for high availability and failover to other AZs, keep in mind that while the IP addresses and DNS for your mount targets in each AZ are static, they are redundant components backed by multiple resources. For more information, see How Amazon EFS works with Amazon EC2 .

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Network isolation for Amazon EFS

As a managed service, Amazon Elastic File System 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 Amazon EFS 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.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

These APIs are callable from any network location, but Amazon EFS does support resource-based access policies which can include restrictions based on the source IP address. You can also use Amazon EFS policies to control access from specific Amazon Virtual Private Cloud (Amazon VPC) endpoints, or specific VPCs. Effectively, this isolates network access to a given Amazon EFS resource from only the specific VPC within the AWS network.

Amazon EFS quotas

Following, you can find out about quotas when working with Amazon EFS.

Topics

- Amazon EFS quotas that you can increase
- Amazon EFS resource quotas that you cannot change
- Quotas for NFS clients
- Quotas for Amazon EFS file systems
- Unsupported NFSv4.0 and 4.1 features
- Additional considerations
- Troubleshooting file operation errors related to quotas

Amazon EFS quotas that you can increase

Service Quotas is an AWS service that helps you manage your quotas, or limits, from one location. In the Service Quotas console, you can view Amazon EFS limit values and request a quota increase for the number of EFS file systems in an AWS Region and the read IOPS for frequently accessed data.

You can also request an increase for the following Amazon EFS quotas by contacting AWS Support. To learn more, see Requesting a quota increase. The Amazon EFS service team reviews each request individually.

- Number of file systems for each customer account.
- Number of access points for each file system.
- Maximum read IOPS per file system using Elastic throughput. When the read IOPS for frequently accessed file systems is increased, both the read IOPS for infrequently accessed file systems and the write IOPS are also increased.
- Elastic throughput per Regional file system for all connected clients in an AWS Region.
- Provisioned throughput per Regional file system for all connected clients in an AWS Region.

The following table lists general default resource quotas that you can request to change.

Resource	Default quota
Number of file systems for each customer account in an AWS Region	1,000
Number of access points for each file system	10,000
Maximum IOPS per file system using Elastic throughput You can request an increase of up to 10 times the default quota of read IOPS for frequently accessed data. Increases in the read IOPS for frequently accessed data also results in an increase of the read IOPS for infrequently accessed data and the write IOPS.	Infrequently accessed data read: 90,000 Frequently accessed data read: 250,000 Write: 50,000

The following table lists Elastic throughput quotas per file system for all connected clients in each AWS Region.

Regional file systems – Total default Elastic throughput per file system for all connected clients in each AWS Region

AWS Region	Maximum read throughput	Maximum write throughput (metered throughput)
US East (Ohio) Region US East (N. Virginia) Region US West (Oregon) Region	60 gibibytes per second (GiBps)	5 GiBps

AWS Region	Maximum read throughput	Maximum write throughput (metered throughput)
Asia Pacific (Mumbai) Region Asia Pacific (Seoul) Region Asia Pacific (Singapore) Region Asia Pacific (Sydney) Region Asia Pacific (Tokyo) Region Europe (Frankfurt) Region Europe (Ireland) Region Europe (London) Region		
All other AWS Regions	20 GiBps	1 GiBps

The following table lists the Provisioned throughput quotas per file system for all connected clients in each AWS Region.

Regional file systems – Total default Provisioned throughput per file system for all connected clients in each AWS Region

AWS Region	Maximum read throughput	Maximum write throughput (metered throughput)
US East (Ohio) Region US East (N. Virginia) Region US West (Oregon) Region Europe (Ireland) Region	10 GiBps	3.33 GiBps
All other AWS Regions	3 GiBps	1 GiBps

Requesting a quota increase

To request an increase for these quotas through AWS Support, take the following steps. The Amazon EFS team reviews each quota increase request.

To request a quota increase through AWS Support

1. Open the AWS Support Center page, and sign in if necessary. Then choose **Create Case**.
2. Under **Create case**, choose **Service Limit Increase**.
3. For **Limit Type**, choose the type of limit to increase. Fill in the necessary fields in the form, and then choose your preferred method of contact.

Amazon EFS resource quotas that you cannot change

Quotas for several Amazon EFS resources cannot be changed, including:

- Quotas for general resources, such as the number of connections for each file system.
- Elastic and Provisioned throughput quotas per One Zone file system for all connected clients in an AWS Region.
- Bursting throughput quotas per Regional or One Zone file system for all connected clients in an AWS Region.

The following table lists the general resource quotas that cannot be changed.

General resource quotas that cannot be changed

Resource	Quota
Number of connections for each file system	25,000
Number of mount targets for each file system in an Availability Zone	1
Number of mount targets for each virtual private cloud (VPC)	1,400
Number of security groups for each mount target	5

Resource	Quota
Number of tags for each file system	50
Number of VPCs for each file system	1

Note

Clients can also connect to mount targets that are in an account or VPC that is different from that of the file system. For more information, see Mounting EFS file systems from another AWS account or VPC.

The following table lists the total default Elastic and Provisioned throughput limits per file system for all connected clients in each AWS Region.

One Zone file systems – Total default Elastic and Provisioned throughput per file system for all connected clients in each AWS Region

AWS Region	Maximum read throughput	Maximum write throughput (metered throughput)
All AWS Regions	3 GiBps	1 GiBps

The following table lists the total Bursting throughput limits per file system for all connected clients in each AWS Region.

Regional and One Zone file systems – Total Bursting throughput per file system for all connected clients in each AWS Region

AWS Region	Maximum read throughput	Maximum write throughput
US East (Ohio) Region US East (N. Virginia) Region US West (Oregon) Region Asia Pacific (Sydney) Region	5 GiBps	3 GiBps

AWS Region	Maximum read throughput	Maximum write throughput
Europe (Ireland) Region		
All other AWS Regions	3 GiBps	1 GiBps

Quotas for NFS clients

The following quotas for NFS clients apply, assuming a Linux NFSv4.1 client:

- Maximum combined read and write throughput is 1,500 mebibytes per second (MiBps) for file systems using Elastic throughput and mounted using version 2.0 or later of the Amazon EFS client (amazon-efs-utils version) or the Amazon EFS CSI Driver (aws-efs-csi-driver). Maximum throughput for all other file systems is 500 MiBps. For more information about performance, see Performance summary. NFS client throughput is calculated as the total number of bytes that are sent and received, with a minimum NFS request size of 4 KB (after applying a 1/3 metering rate for read requests).
- Up to 65,536 active users for each client can have files open at the same time.
- Up to 65,536 files open at the same time on the instance. Listing directory contents doesn't count as opening a file.
- Each unique mount on the client can acquire up to a total of 65,536 locks per connection.
- When connecting to Amazon EFS, NFS clients located on-premises or in another AWS Region can observe lower throughput than when connecting to EFS from the same AWS Region. This effect is because of increased network latency. Network latency of 1 ms or less is required to achieve maximum per-client throughput. Use the DataSync data migration service when migrating large datasets from on-premises NFS servers to EFS.
- The NFS protocol supports a maximum of 16 group IDs (GIDs) per user and any additional GIDs are truncated from NFS client requests. For more information, see Access denied to allowed files on NFS file system.
- Using Amazon EFS with Microsoft Windows isn't supported.

Quotas for Amazon EFS file systems

The following quotas are specific to Amazon EFS file systems.

Resource	Quota
File name length, in bytes	255
Symbolic link (symlink) length, in bytes	4,080
Number of hard links to a file	177
Size of a single file	52,673,613,135,872 bytes (47.9 TiB)
Number of levels for directory depth	1,000
Number of locks on a single file across all instances and users	512
Character limit for each file system policy	20,000
*Number of file operations per second for General Purpose mode	250,000

*For more information about the number of file operations per second for General Purpose mode, see Performance summary.

Unsupported NFSv4.0 and 4.1 features

Although Amazon EFS doesn't support NFSv2, or NFSv3, it does support both NFSv4.1 and NFSv4.0, except for the following features:

- pNFS
- Exended attributes
- Client delegation or callbacks of any type
  - Operation OPEN always returns `OPEN_DELEGATE_NONE` as the delegation type.
  - The operation OPEN returns `NFSERR_NOTSUPP` for the `CLAIM_DELEGATE_CUR` and `CLAIM_DELEGATE_PREV` claim types.
- Mandatory locking

All locks in Amazon EFS are advisory, which means that read and write operations don't check for conflicting locks before the operation is executed.

- Deny share

  NFS supports the concept of a share deny. A *share deny* is primarily used by Windows clients for users to deny others access to a particular file that has been opened. Amazon EFS doesn't support this, and returns the NFS error `NFS4ERR_NOTSUPP` for any OPEN commands specifying a share deny value other than `OPEN4_SHARE_DENY_NONE`. Linux NFS clients don't use anything other than `OPEN4_SHARE_DENY_NONE`.
- Access control lists (ACLs)
- Amazon EFS doesn't update the `time_access` attribute on file reads. Amazon EFS updates `time_access` in the following events:
  - When a file is created (an inode is created)
  - When an NFS client makes an explicit `setattr` call
  - On a write to the inode caused by, for example, file size changes or file metadata changes
  - Any inode attribute is updated
- Namespaces
- Persistent reply cache
- Kerberos based security
- NFSv4.1 data retention
- SetUID on directories
- Unsupported file types when using the CREATE operation: Block devices (NF4BLK), character devices (NF4CHR), attribute directory (NF4ATTRDIR), and named attribute (NF4NAMEDATTR).
- Unsupported attributes: FATTR4_ARCHIVE, FATTR4_FILES_AVAIL, FATTR4_FILES_FREE, FATTR4_FILES_TOTAL, FATTR4_FS_LOCATIONS, FATTR4_MIMETYPE, FATTR4_QUOTA_AVAIL_HARD, FATTR4_QUOTA_AVAIL_SOFT, FATTR4_QUOTA_USED, FATTR4_TIME_BACKUP, and FATTR4_ACL.

  An attempt to set these attributes results in an `NFS4ERR_ATTRNOTSUPP` error that is sent back to the client.

Additional considerations

In addition, note the following:

- For a list of AWS Regions where you can create Amazon EFS file systems, see the AWS General Reference User Guide.
- Amazon EFS does not support the `nconnect` mount option.
- You can mount an Amazon EFS file system from on-premises data center servers using AWS Direct Connect and VPN. For more information, see Tutorial: Mounting with on-premises clients.

Troubleshooting file operation errors related to quotas

When you access EFS file systems, certain limits on the files in the file system apply. Exceeding these limits causes file operation errors. For more information about file-based limits in Amazon EFS, see Amazon EFS quotas.

Following, you can find some common file operation errors and the limits associated with each error.

Topics

- Command fails with “Disk quota exceeded” error
- Command fails with "I/O error"
- Command fails with "File name is too long" error
- Command fails with "File not found" error
- Command fails with "Too many links" error
- Command fails with "File too large" error

Command fails with “Disk quota exceeded” error

Amazon EFS doesn't currently support user disk quotas. This error can occur if any of the following limits have been exceeded:

- Up to 65,536 active users can have files open at the same time. A user account that is logged in multiple times counts as one active user.
- Up to 65,536 files can be open at once for an instance. Listing directory contents doesn't count as opening a file.

- Each unique mount on the client can acquire up to a total of 65,536 locks per connection.

Action to take

If you encounter this issue, you can resolve it by identifying which of the preceding limits you are exceeding, and then making changes to meet that limit. For more information, see Quotas for NFS clients.

Command fails with "I/O error"

This error occurs when you encounter one of the following issues:

- More than 65,536 active user accounts for each instance have files open at once.

  Action to take

  If you encounter this issue, you can resolve it by meeting the supported limit of open files on your instances. To do so, reduce the number of active users that have files from your Amazon EFS file system open simultaneously on your instances.
- The AWS KMS key encrypting your file system was deleted.

  Action to take

  If you encounter this issue, you can no longer decrypt the data that was encrypted under that key, which means that data becomes unrecoverable.

Command fails with "File name is too long" error

This error occurs when the size of a file name or its symbolic link (symlink) is too long. File names have the following limits:

- A name can be up to 255 bytes long.
- A symlink can be up to 4080 bytes in size.

Action to take

If you encounter this issue, you can resolve it by reducing the size of your file name or symlink length to meet the supported limits.

Command fails with "File not found" error

This error occurs because some older 32-bit versions of Oracle E-Business suite use 32-bit file I/O interfaces, and EFS uses 64-bit inode numbers. System calls that may fail include `stat()` and `readdir()`.

Action to take

If you encounter this error, you can resolve it by using the **nfs.enable_ino64=0 kernel** boot option. This option compresses the 64-bit EFS inode numbers to 32 bits. Kernel boot options are handled differently for different Linux distributions. On Amazon Linux, turn on this option by adding `nfs.enable_ino64=0 kernel` to the `GRUB_CMDLINE_LINUX_DEFAULT` variable in `/etc/default/grub`. Please consult your distribution for specific documentation on how to turn on kernel boot options.

Command fails with "Too many links" error

This error occurs when there are too many hard links to a file. You can have up to 177 hard links in a file.

Action to take

If you encounter this issue, you can resolve it by reducing the number of hard links to a file to meet the supported limit.

Command fails with "File too large" error

This error occurs when a file is too large. A single file can be up to 52,673,613,135,872 bytes (47.9 TiB) in size.

Action to take

If you encounter this issue, you can resolve it by reducing the size of a file to meet the supported limit.

Amazon EFS API

The Amazon EFS API is a network protocol based on HTTP (RFC 2616). For each API call, you make an HTTP request to the region-specific Amazon EFS API endpoint for the AWS Region where you want to manage file systems. The API uses JSON (RFC 4627) documents for HTTP request/response bodies.

The Amazon EFS API is an RPC model. In this model, there is a fixed set of operations and the syntax for each operation is known to clients without any prior interaction. In the following section, you can find a description of each API operation using an abstract RPC notation. Each has an operation name that doesn't appear on the wire. For each operation, the topic specifies the mapping to HTTP request elements.

The specific Amazon EFS operation to which a given request maps is determined by a combination of the request's method (GET, PUT, POST, or DELETE) and which of the various patterns its Request-URI matches. If the operation is PUT or POST, Amazon EFS extracts call arguments from the Request-URI path segment, query parameters, and the JSON object in the request body.

> **Note**
>
> Although operation names, such as `CreateFileSystem`, don't appear on the wire, these names are meaningful in AWS Identity and Access Management (IAM) policies. For more information, see Identity and access management for Amazon EFS.
> The operation name is also used to name commands in command-line tools and elements of the AWS SDK APIs. For example, there is a AWS CLI command named `create-file-system` that maps to the `CreateFileSystem` operation.
> The operation name also appears in AWS CloudTrail logs for Amazon EFS API calls.

API endpoints

An endpoint is a URL that serves as an entry point for an AWS web service. Amazon EFS supports the following endpoint types:

- IPv4 endpoints
- Dual-stack (both IPv4 and IPv6) endpoints
- FIPS endpoints

When you make a request, you can specify the endpoint to use. If you do not specify an endpoint, the IPv4 endpoint is used by default. To use a different endpoint type, you must specify it in your request. For examples of how to do this, see Specifying endpoints. For a table of available endpoints, see Amazon EFS endpoints and quotas.

IPv4 endpoints

IPv4 endpoints support IPv4 traffic only. IPv4 endpoints are available for all Regions. The format of the IPv4 endpoint depends on the Region.

- For IPv4 endpoints in the AWS commercial Regions, the format is `elasticfilesystem.region.amazonaws.com`.

  For example, if you specify `elasticfilesystem.us-west-2.amazonaws.com` as the endpoint, we direct your request to the US West (Oregon) Region (us-west-2) endpoint.

- For IPv4 endpoints in the China Regions, the format is `elasticfilesystem.region.amazonaws.com.cn`.

  For example, the Amazon EFS API endpoint for the China (Beijing) Region is `elasticfilesystem.cn-north-1.amazonaws.com.cn`.

Dual-stack (IPv4 and IPv6) endpoints

Dual-stack endpoints support both IPv4 and IPv6 traffic. When you make a request to a dual-stack endpoint, the endpoint URL resolves to an IPv6 or an IPv4 address, depending on the protocol used by your network and client. The format of the dual-stack endpoint depends on the Region.

- For dual-stack endpoints in the AWS commercial Regions, the format is `efs.aws-region.api.aws`.

  For example, the dual-stack endpoint for the US West (Oregon) Region (us-west-2) is `efs.eu-west-2.api.aws`.

- For dual-stack endpoints in the China Regions, the format is `efs.region.api.amazonwebservices.com.cn`.

  For example, the dual-stack endpoint for the China (Beijing) Region is `efs.cn-north-1.api.amazonwebservices.com.cn`.

Specifying endpoints

The following examples show how to specify an endpoint for the US West (N. California) Region (us-west-1) using the AWS CLI for Amazon EFS.

IPv4

```
aws efs get-rest-apis --region us-west-1 --endpoint-url https://elasticfilesystem.us-west-1.amazonaws.com
```

Dual-stack

```
aws efs get-rest-apis --region us-west-1 --endpoint-url https://efs.us-west-1.api.aws
```

API version

The version of the API being used for a call is identified by the first path segment of the request URI, and its form is an ISO 8601 date. For example, see CreateFileSystem.

The documentation describes API version 2015-02-01.

Related topics

The following sections provide descriptions of the API operations, how to create a signature for request authentication, and how to grant permissions for these API operations using the IAM policies.

- Identity and access management for Amazon EFS
- Actions
- Data Types

Working with the query API request rate for Amazon EFS

Amazon EFS API requests are throttled for each AWS account on a per-region basis to help service performance. All Amazon EFS API calls together, whether they originate from an application, the AWS CLI, or the Amazon EFS console, must not exceed the maximum allowed API request rate. The maximum API request rate can vary across AWS Regions. API requests made are attributed to the underlying AWS account.

If an API request exceeds the API request rate for its category, the request returns the `ThrottlingException` error code. To prevent this error, ensure that your application doesn't retry API requests at a high rate. You can do this by using care when polling and by using exponential backoff retries.

Polling

Your application might need to call an API operation repeatedly to check for an update in status. Before you start polling, give the request time to potentially complete. When you begin polling, use an appropriate sleep interval between successive requests. For best results, use an increasing sleep interval.

Retries or batch processing

Your application might need to retry an API request after it fails, or to process multiple resources (for example, all of your Amazon EFS file systems). To lower the rate of API requests, use an appropriate sleep interval between successive requests. For best results, use an increasing or variable sleep interval.

Calculating the sleep interval

When you have to poll or retry an API request, we recommend using an exponential backoff algorithm to calculate the sleep interval between API calls. The idea behind exponential backoff is to use progressively longer waits between retries for consecutive error responses. For more information, and implementation examples of this algorithm, see Retry behavior in the *Amazon Web Services General Reference.*

Actions

The following actions are supported:

- CreateAccessPoint
- CreateFileSystem
- CreateMountTarget
- CreateReplicationConfiguration

- CreateTags
- DeleteAccessPoint
- DeleteFileSystem
- DeleteFileSystemPolicy
- DeleteMountTarget
- DeleteReplicationConfiguration
- DeleteTags
- DescribeAccessPoints
- DescribeAccountPreferences
- DescribeBackupPolicy
- DescribeFileSystemPolicy
- DescribeFileSystems
- DescribeLifecycleConfiguration
- DescribeMountTargets
- DescribeMountTargetSecurityGroups
- DescribeReplicationConfigurations
- DescribeTags
- ListTagsForResource
- ModifyMountTargetSecurityGroups
- PutAccountPreferences
- PutBackupPolicy
- PutFileSystemPolicy
- PutLifecycleConfiguration
- TagResource
- UntagResource
- UpdateFileSystem
- UpdateFileSystemProtection

CreateAccessPoint

Creates an EFS access point. An access point is an application-specific view into an EFS file system that applies an operating system user and group, and a file system path, to any file system request made through the access point. The operating system user and group override any identity information provided by the NFS client. The file system path is exposed as the access point's root directory. Applications using the access point can only access data in the application's own directory and any subdirectories. A file system can have a maximum of 10,000 access points unless you request an increase. To learn more, see Mounting a file system using EFS access points.

> **Note**
>
> If multiple requests to create access points on the same file system are sent in quick succession, and the file system is near the limit of access points, you may experience a throttling response for these requests. This is to ensure that the file system does not exceed the stated access point limit.

This operation requires permissions for the `elasticfilesystem:CreateAccessPoint` action.

Access points can be tagged on creation. If tags are specified in the creation action, IAM performs additional authorization on the `elasticfilesystem:TagResource` action to verify if users have permissions to create tags. Therefore, you must grant explicit permissions to use the `elasticfilesystem:TagResource` action. For more information, see Granting permissions to tag resources during creation.

Request Syntax

```
POST /2015-02-01/access-points HTTP/1.1
Content-type: application/json

{
   "ClientToken": "string",
   "FileSystemId": "string",
   "PosixUser": {
      "Gid": number,
      "SecondaryGids": [ number ],
      "Uid": number
   },
   "RootDirectory": {
```

```
      "CreationInfo": {
         "OwnerGid": number,
         "OwnerUid": number,
         "Permissions": "string"
      },
      "Path": "string"
   },
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ClientToken

A string of up to 64 ASCII characters that Amazon EFS uses to ensure idempotent creation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: Yes

FileSystemId

The ID of the EFS file system that the access point provides access to.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

PosixUser

The operating system user and group applied to all file system requests made using the access point.

Type: PosixUser object

Required: No

RootDirectory

Specifies the directory on the EFS file system that the access point exposes as the root directory of your file system to NFS clients using the access point. The clients using the access point can only access the root directory and below. If the `RootDirectory` > `Path` specified does not exist, Amazon EFS creates it and applies the `CreationInfo` settings when a client connects to an access point. When specifying a `RootDirectory`, you must provide the `Path`, and the `CreationInfo`.

Amazon EFS creates a root directory only if you have provided the CreationInfo: OwnUid, OwnGID, and permissions for the directory. If you do not provide this information, Amazon EFS does not create the root directory. If the root directory does not exist, attempts to mount using the access point will fail.

Type: RootDirectory object

Required: No

Tags

Creates tags associated with the access point. Each tag is a key-value pair, each key must be unique. For more information, see Tagging AWS resources in the *AWS General Reference Guide*.

Type: Array of Tag objects

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "AccessPointArn": "string",
   "AccessPointId": "string",
   "ClientToken": "string",
   "FileSystemId": "string",
   "LifeCycleState": "string",
   "Name": "string",
   "OwnerId": "string",
   "PosixUser": {
      "Gid": number,
      "SecondaryGids": [ number ],
      "Uid": number
   },
   "RootDirectory": {
      "CreationInfo": {
         "OwnerGid": number,
         "OwnerUid": number,
         "Permissions": "string"
      },
      "Path": "string"
   },
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

AccessPointArn

The unique Amazon Resource Name (ARN) associated with the access point.

Type: String

Length Constraints: Maximum length of 128.

Pattern: ^arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:access-point/fsap-[0-9a-f]{8,40}$

AccessPointId

The ID of the access point, assigned by Amazon EFS.

Type: String

Length Constraints: Maximum length of 128.

Pattern: ^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:access-point/fsap-[0-9a-f]{8,40}|fsap-[0-9a-f]{8,40})$

ClientToken

The opaque string specified in the request to ensure idempotent creation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: .+

FileSystemId

The ID of the EFS file system that the access point applies to.

Type: String

Length Constraints: Maximum length of 128.

Pattern: ^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$

LifeCycleState

Identifies the lifecycle phase of the access point.

Type: String

Valid Values: creating | available | updating | deleting | deleted | error

Name

The name of the access point. This is the value of the Name tag.

Type: String

OwnerId

Identifies the AWS account that owns the access point resource.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

PosixUser

The full POSIX identity, including the user ID, group ID, and secondary group IDs on the access point that is used for all file operations by NFS clients using the access point.

Type: PosixUser object

RootDirectory

The directory on the EFS file system that the access point exposes as the root directory to NFS clients using the access point.

Type: RootDirectory object

Tags

The tags associated with the access point, presented as an array of Tag objects.

Type: Array of Tag objects

Errors

AccessPointAlreadyExists

Returned if the access point that you are trying to create already exists, with the creation token you provided in the request.

HTTP Status Code: 409

AccessPointLimitExceeded

Returned if the AWS account has already created the maximum number of access points allowed per file system. For more informaton, see https://docs.aws.amazon.com/efs/latest/ug/limits.html#limits-efs-resources-per-account-per-region.

HTTP Status Code: 403

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ThrottlingException

Returned when the `CreateAccessPoint` API action is called too quickly and the number of Access Points on the file system is nearing the limit of 120.

HTTP Status Code: 429

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateFileSystem

Creates a new, empty file system. The operation requires a creation token in the request that Amazon EFS uses to ensure idempotent creation (calling the operation with same creation token has no effect). If a file system does not currently exist that is owned by the caller's AWS account with the specified creation token, this operation does the following:

- Creates a new, empty file system. The file system will have an Amazon EFS assigned ID, and an initial lifecycle state `creating`.
- Returns with the description of the created file system.

Otherwise, this operation returns a `FileSystemAlreadyExists` error with the ID of the existing file system.

> **Note**
>
> For basic use cases, you can use a randomly generated UUID for the creation token.

The idempotent operation allows you to retry a `CreateFileSystem` call without risk of creating an extra file system. This can happen when an initial call fails in a way that leaves it uncertain whether or not a file system was actually created. An example might be that a transport level timeout occurred or your connection was reset. As long as you use the same creation token, if the initial call had succeeded in creating a file system, the client can learn of its existence from the `FileSystemAlreadyExists` error.

For more information, see Creating a file system in the *Amazon EFS User Guide*.

> **Note**
>
> The `CreateFileSystem` call returns while the file system's lifecycle state is still `creating`. You can check the file system creation status by calling the DescribeFileSystems operation, which among other things returns the file system state.

This operation accepts an optional `PerformanceMode` parameter that you choose for your file system. We recommend `generalPurpose` `PerformanceMode` for all file systems. The `maxIO` mode is a previous generation performance type that is designed for highly parallelized workloads that can tolerate higher latencies than the `generalPurpose` mode. `MaxIO` mode is not supported for One Zone file systems or file systems that use Elastic throughput.

The `PerformanceMode` can't be changed after the file system has been created. For more information, see Amazon EFS performance modes.

You can set the throughput mode for the file system using the `ThroughputMode` parameter.

After the file system is fully created, Amazon EFS sets its lifecycle state to `available`, at which point you can create one or more mount targets for the file system in your VPC. For more information, see CreateMountTarget. You mount your Amazon EFS file system on an EC2 instances in your VPC by using the mount target. For more information, see Amazon EFS: How it Works.

This operation requires permissions for the `elasticfilesystem:CreateFileSystem` action.

File systems can be tagged on creation. If tags are specified in the creation action, IAM performs additional authorization on the `elasticfilesystem:TagResource` action to verify if users have permissions to create tags. Therefore, you must grant explicit permissions to use the `elasticfilesystem:TagResource` action. For more information, see Granting permissions to tag resources during creation.

Request Syntax

```
POST /2015-02-01/file-systems HTTP/1.1
Content-type: application/json

{
   "AvailabilityZoneName": "string",
   "Backup": boolean,
   "CreationToken": "string",
   "Encrypted": boolean,
   "KmsKeyId": "string",
   "PerformanceMode": "string",
   "ProvisionedThroughputInMibps": number,
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ],
   "ThroughputMode": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

AvailabilityZoneName

For One Zone file systems, specify the AWS Availability Zone in which to create the file system. Use the format `us-east-1a` to specify the Availability Zone. For more information about One Zone file systems, see EFS file system types in the *Amazon EFS User Guide*.

> **Note**
>
> One Zone file systems are not available in all Availability Zones in AWS Regions where Amazon EFS is available.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: No

Backup

Specifies whether automatic backups are enabled on the file system that you are creating. Set the value to `true` to enable automatic backups. If you are creating a One Zone file system, automatic backups are enabled by default. For more information, see Automatic backups in the *Amazon EFS User Guide*.

Default is `false`. However, if you specify an `AvailabilityZoneName`, the default is `true`.

> **Note**
>
> AWS Backup is not available in all AWS Regions where Amazon EFS is available.

Type: Boolean

Required: No

CreationToken

A string of up to 64 ASCII characters. Amazon EFS uses this to ensure idempotent creation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: Yes

Encrypted

A Boolean value that, if true, creates an encrypted file system. When creating an encrypted file system, you have the option of specifying an existing AWS Key Management Service key (KMS key). If you don't specify a KMS key, then the default KMS key for Amazon EFS, `/aws/elasticfilesystem`, is used to protect the encrypted file system.

Type: Boolean

Required: No

KmsKeyId

The ID of the KMS key that you want to use to protect the encrypted file system. This parameter is required only if you want to use a non-default KMS key. If this parameter is not specified, the default KMS key for Amazon EFS is used. You can specify a KMS key ID using the following formats:

- Key ID - A unique identifier of the key, for example `1234abcd-12ab-34cd-56ef-1234567890ab`.
- ARN - An Amazon Resource Name (ARN) for the key, for example `arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`.
- Key alias - A previously created display name for a key, for example `alias/projectKey1`.
- Key alias ARN - An ARN for a key alias, for example `arn:aws:kms:us-west-2:444455556666:alias/projectKey1`.

If you use `KmsKeyId`, you must set the [CreateFileSystem:Encrypted] parameter to true.

> **Important**
>
> EFS accepts only symmetric KMS keys. You cannot use asymmetric KMS keys with Amazon EFS file systems.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `^([0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|mrk-[0-9a-f]{32}|alias/[a-zA-Z0-9/_-]+|(arn:aws[-a-z]*:kms:[a-z0-9-]+:\d{12}:((key/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})|(key/mrk-[0-9a-f]{32})|(alias/[a-zA-Z0-9/_-]+))))$`

Required: No

PerformanceMode

The performance mode of the file system. We recommend `generalPurpose` performance mode for all file systems. File systems using the `maxIO` performance mode can scale to higher levels of aggregate throughput and operations per second with a tradeoff of slightly higher latencies for most file operations. The performance mode can't be changed after the file system has been created. The `maxIO` mode is not supported on One Zone file systems.

> **Important**
>
> Due to the higher per-operation latencies with Max I/O, we recommend using General Purpose performance mode for all file systems.

Default is `generalPurpose`.

Type: String

Valid Values: `generalPurpose | maxIO`

Required: No

ProvisionedThroughputInMibps

The throughput, measured in mebibytes per second (MiBps), that you want to provision for a file system that you're creating. Required if `ThroughputMode` is set to `provisioned`. Valid values are 1-3414 MiBps, with the upper limit depending on Region. To increase this limit, contact Support. For more information, see Amazon EFS quotas that you can increase in the *Amazon EFS User Guide*.

Type: Double

Valid Range: Minimum value of 1.0.

Required: No

Tags

Use to create one or more tags associated with the file system. Each tag is a user-defined key-value pair. Name your file system on creation by including a `"Key":"Name","Value":"{value}"` key-value pair. Each key must be unique. For more information, see Tagging AWS resources in the *AWS General Reference Guide*.

Type: Array of Tag objects

Required: No

ThroughputMode

Specifies the throughput mode for the file system. The mode can be `bursting`, `provisioned`, or `elastic`. If you set `ThroughputMode` to `provisioned`, you must also set a value for `ProvisionedThroughputInMibps`. After you create the file system, you can decrease your file system's Provisioned throughput or change between the throughput modes, with certain time restrictions. For more information, see Specifying throughput with provisioned mode in the *Amazon EFS User Guide*.

Default is `bursting`.

Type: String

Valid Values: `bursting | provisioned | elastic`

Required: No

Response Syntax

```
HTTP/1.1 201
Content-type: application/json

{
   "AvailabilityZoneId": "string",
   "AvailabilityZoneName": "string",
   "CreationTime": number,
   "CreationToken": "string",
   "Encrypted": boolean,
   "FileSystemArn": "string",
   "FileSystemId": "string",
   "FileSystemProtection": {
      "ReplicationOverwriteProtection": "string"
   },
   "KmsKeyId": "string",
   "LifeCycleState": "string",
   "Name": "string",
   "NumberOfMountTargets": number,
   "OwnerId": "string",
   "PerformanceMode": "string",
   "ProvisionedThroughputInMibps": number,
   "SizeInBytes": {
      "Timestamp": number,
      "Value": number,
      "ValueInArchive": number,
      "ValueInIA": number,
      "ValueInStandard": number
   },
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ],
   "ThroughputMode": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 201 response.

The following data is returned in JSON format by the service.

AvailabilityZoneId

The unique and consistent identifier of the Availability Zone in which the file system is located, and is valid only for One Zone file systems. For example, `use1-az1` is an Availability Zone ID for the us-east-1 AWS Region, and it has the same location in every AWS account.

Type: String

AvailabilityZoneName

Describes the AWS Availability Zone in which the file system is located, and is valid only for One Zone file systems. For more information, see Using EFS storage classes in the *Amazon EFS User Guide*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

CreationTime

The time that the file system was created, in seconds (since 1970-01-01T00:00:00Z).

Type: Timestamp

CreationToken

The opaque string specified in the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Encrypted

A Boolean value that, if true, indicates that the file system is encrypted.

Type: Boolean

FileSystemArn

The Amazon Resource Name (ARN) for the EFS file system, in the format `arn:aws:elasticfilesystem:`*`region`*`:`*`account-id`*`:file-system/`*`file-system-id`* . Example with sample data: `arn:aws:elasticfilesystem:us-west-2:1111333322228888:file-system/fs-01234567`

Type: String

FileSystemId

The ID of the file system, assigned by Amazon EFS.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

FileSystemProtection

Describes the protection on the file system.

Type: FileSystemProtectionDescription object

KmsKeyId

The ID of an AWS KMS key used to protect the encrypted file system.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `^([0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|mrk-[0-9a-f]{32}|alias/[a-zA-Z0-9/_-]+|(arn:aws[-a-z]*:kms:[a-z0-9-]+:\d{12}:((key/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})|(key/mrk-[0-9a-f]{32})|(alias/[a-zA-Z0-9/_-]+))))$`

LifeCycleState

The lifecycle phase of the file system.

Type: String

Valid Values: `creating | available | updating | deleting | deleted | error`

Name

You can add tags to a file system, including a `Name` tag. For more information, see CreateFileSystem. If the file system has a `Name` tag, Amazon EFS returns the value in this field.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

NumberOfMountTargets

The current number of mount targets that the file system has. For more information, see CreateMountTarget.

Type: Integer

Valid Range: Minimum value of 0.

OwnerId

The AWS account that created the file system.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

PerformanceMode

The performance mode of the file system.

Type: String

Valid Values: `generalPurpose | maxIO`

ProvisionedThroughputInMibps

The amount of provisioned throughput, measured in MiBps, for the file system. Valid for file systems using `ThroughputMode` set to `provisioned`.

Type: Double

Valid Range: Minimum value of 1.0.

SizeInBytes

The latest known metered size (in bytes) of data stored in the file system, in its `Value` field, and the time at which that size was determined in its `Timestamp` field. The `Timestamp` value is the integer number of seconds since 1970-01-01T00:00:00Z. The `SizeInBytes` value doesn't represent the size of a consistent snapshot of the file system, but it is eventually consistent when there are no writes to the file system. That is, `SizeInBytes` represents actual size only if the file system is not modified for a period longer than a couple of hours. Otherwise, the value is not the exact size that the file system was at any point in time.

Type: FileSystemSize object

Tags

The tags associated with the file system, presented as an array of `Tag` objects.

Type: Array of Tag objects

ThroughputMode

Displays the file system's throughput mode. For more information, see Throughput modes in the *Amazon EFS User Guide*.

Type: String

Valid Values: `bursting | provisioned | elastic`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemAlreadyExists

Returned if the file system you are trying to create already exists, with the creation token you provided.

HTTP Status Code: 409

FileSystemLimitExceeded

Returned if the AWS account has already created the maximum number of file systems allowed per account.

HTTP Status Code: 403

InsufficientThroughputCapacity

Returned if there's not enough capacity to provision additional throughput. This value might be returned when you try to create a file system in provisioned throughput mode, when you attempt to increase the provisioned throughput of an existing file system, or when you attempt to change an existing file system from Bursting Throughput to Provisioned Throughput mode. Try again later.

HTTP Status Code: 503

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ThroughputLimitExceeded

Returned if the throughput mode or amount of provisioned throughput can't be changed because the throughput limit of 1024 MiB/s has been reached.

HTTP Status Code: 400

UnsupportedAvailabilityZone

Returned if the requested Amazon EFS functionality is not available in the specified Availability Zone.

HTTP Status Code: 400

Examples

Create an encrypted EFS file system

The following example sends a POST request to create a file system in the `us-west-2` Region with automatic backups enabled. The request specifies `myFileSystem1` as the creation token for idempotency.

Sample Request

```
POST /2015-02-01/file-systems HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T215117Z
Authorization: <...>
Content-Type: application/json
Content-Length: 42

{
  "CreationToken" : "myFileSystem1",
  "PerformanceMode" : "generalPurpose",
  "Backup": true,
  "Encrypted": true,
  "Tags":[
      {
         "Key": "Name",
         "Value": "Test Group1"
      }
   ]
}
```

Sample Response

```
HTTP/1.1 201 Created
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 319

{
   "ownerId":"251839141158",
   "CreationToken":"myFileSystem1",
   "Encrypted": true,
   "PerformanceMode" : "generalPurpose",
   "fileSystemId":"fs-01234567",
   "CreationTime":"1403301078",
   "LifeCycleState":"creating",
   "numberOfMountTargets":0,
   "SizeInBytes":{
       "Timestamp": 1403301078,
       "Value": 29313618372,
       "ValueInArchive": 201156,
       "ValueInIA": 675432,
```

```
        "ValueInStandard": 29312741784
    },
    "Tags":[
       {
         "Key": "Name",
         "Value": "Test Group1"
       }
    ],
    "ThroughputMode": "elastic"
}
```

Create an encrypted EFS file system with One Zone availability

The following example sends a POST request to create a file system in the `us-west-2` Region with automatic backups enabled. The file system will have One Zone storage in the `us-west-2b` Availability Zone.

Sample Request

```
POST /2015-02-01/file-systems HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T215117Z
Authorization: <...>
Content-Type: application/json
Content-Length: 42

{
  "CreationToken" : "myFileSystem2",
  "PerformanceMode" : "generalPurpose",
  "Backup": true,
  "AvailabilityZoneName": "us-west-2b",
  "Encrypted": true,
  "ThroughputMode": "elastic",
  "Tags":[
      {
         "Key": "Name",
         "Value": "Test Group1"
      }
   ]
}
```

Sample Response

```
HTTP/1.1 201 Created
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 319

{
   "ownerId":"251839141158",
   "CreationToken":"myFileSystem1",
   "Encrypted": true,
   "AvailabilityZoneId": "usew2-az2",
   "AvailabilityZoneName": "us-west-2b",
   "PerformanceMode" : "generalPurpose",
   "fileSystemId":"fs-01234567",
   "CreationTime":"1403301078",
   "LifeCycleState":"creating",
   "numberOfMountTargets":0,
   "SizeInBytes":{
       "Timestamp": 1403301078,
       "Value": 29313618372,
       "ValueInArchive": 201156,
       "ValueInIA": 675432,
       "ValueInStandard": 29312741784
   },
   "Tags":[
      {
        "Key": "Name",
        "Value": "Test Group1"
      }
   ],
   "ThroughputMode": "elastic"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateMountTarget

Creates a mount target for a file system. You can then mount the file system on EC2 instances by using the mount target. For more information about mount targets, see Managing mount targets.

To create a mount target for an EFS file system, the file system's lifecycle state must be `available`. For more information, see DescribeFileSystems.

After creating the mount target, Amazon EFS returns a response that includes a `MountTargetId` and an IP address (`IpAddress` or `Ipv6Address`). You use this IP address when mounting the file system in an EC2 instance. You can also use the mount target's DNS name when mounting the file system. The EC2 instance on which you mount the file system by using the mount target can resolve the mount target's DNS name to its IP address. For more information, see How Amazon EFS works.

Note that you can create mount targets for a file system in only one VPC, and there can be only one mount target per Availability Zone. For more information, see Creating mount targets.

If the request satisfies the requirements, Amazon EFS does the following:

- Creates a new mount target in the specified subnet.
- Creates a new network interface in the subnet with the folowing:
  - The description `Mount target fsmt-id for file system fs-id` where *`fsmt-id`* is the mount target ID, and *`fs-id`* is the `FileSystemId`.
  - The `requesterManaged` property of the network interface set to `true`, and the `requesterId` value set to `EFS`.

  Each mount target has one corresponding requester-managed EC2 network interface. After the network interface is created, Amazon EFS sets the `NetworkInterfaceId` field in the mount target's description to the network interface ID, and the IP address to its address. If network interface creation fails, the entire `CreateMountTarget` operation fails.

> **Note**
>
> The `CreateMountTarget` call returns only after creating the network interface, but while the mount target state is still `creating`, you can check the mount target creation status by

calling the DescribeMountTargets operation, which among other things returns the mount target state.

This operation requires permissions for the following action on the file system:

- `elasticfilesystem:CreateMountTarget`

This operation also requires permissions for the following Amazon EC2 actions:

- `ec2:DescribeSubnets`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`

Request Syntax

```
POST /2015-02-01/mount-targets HTTP/1.1
Content-type: application/json

{
   "FileSystemId": "string",
   "IpAddress": "string",
   "IpAddressType": "string",
   "Ipv6Address": "string",
   "SecurityGroups": [ "string" ],
   "SubnetId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

FileSystemId

The ID of the file system for which to create the mount target.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

IpAddress

If the `IpAddressType` for the mount target is IPv4 (`IPV4_ONLY` or `DUAL_STACK`), then specify the IPv4 address to use. If you do not specify an `IpAddress`, then Amazon EFS selects an unused IP address from the subnet specified for `SubnetId`.

Type: String

Length Constraints: Minimum length of 7. Maximum length of 15.

Pattern: `^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$`

Required: No

IpAddressType

The IP address type for the mount target. The possible values are `IPV4_ONLY` (only IPv4 addresses), `IPV6_ONLY` (only IPv6 addresses), and `DUAL_STACK` (dual-stack, both IPv4 and IPv6 addresses). If you don't specify an `IpAddressType`, then `IPV4_ONLY` is used.

> **Note**
>
> The `IPAddressType` must match the IP type of the subnet. Additionally, the `IPAddressType` parameter overrides the value set as the default IP address for the subnet in the VPC. For example, if the `IPAddressType` is `IPV4_ONLY` and `AssignIpv6AddressOnCreation` is `true`, then IPv4 is used for the mount target. For more information, see [Modify the IP addressing attributes of your subnet](#).

Type: String

Valid Values: `IPV4_ONLY | IPV6_ONLY | DUAL_STACK`

Required: No

Ipv6Address

If the `IPAddressType` for the mount target is IPv6 (`IPV6_ONLY` or `DUAL_STACK`), then specify the IPv6 address to use. If you do not specify an `Ipv6Address`, then Amazon EFS selects an unused IP address from the subnet specified for `SubnetId`.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 39.

Required: No

SecurityGroups

VPC security group IDs, of the form `sg-xxxxxxxx`. These must be for the same VPC as the subnet specified. The maximum number of security groups depends on account quota. For more information, see Amazon VPC Quotas in the *Amazon VPC User Guide* (see the **Security Groups** table). If you don't specify a security group, then Amazon EFS uses the default security group for the subnet's VPC.

Type: Array of strings

Array Members: Maximum number of 100 items.

Length Constraints: Minimum length of 11. Maximum length of 43.

Pattern: `^sg-[0-9a-f]{8,40}`

Required: No

SubnetId

The ID of the subnet to add the mount target in. For One Zone file systems, use the subnet that is associated with the file system's Availability Zone. The subnet type must be the same type as the `IpAddressType`.

Type: String

Length Constraints: Minimum length of 15. Maximum length of 47.

Pattern: `^subnet-[0-9a-f]{8,40}$`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "AvailabilityZoneId": "string",
   "AvailabilityZoneName": "string",
   "FileSystemId": "string",
   "IpAddress": "string",
   "Ipv6Address": "string",
   "LifeCycleState": "string",
   "MountTargetId": "string",
   "NetworkInterfaceId": "string",
   "OwnerId": "string",
   "SubnetId": "string",
   "VpcId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

AvailabilityZoneId

The unique and consistent identifier of the Availability Zone that the mount target resides in. For example, `use1-az1` is an AZ ID for the us-east-1 Region and it has the same location in every AWS account.

Type: String

AvailabilityZoneName

The name of the Availability Zone in which the mount target is located. Availability Zones are independently mapped to names for each AWS account. For example, the Availability Zone `us-east-1a` for your AWS account might not be the same location as `us-east-1a` for another AWS account.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

FileSystemId

The ID of the file system for which the mount target is intended.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

IpAddress

The IPv4 address for the mount target.

Type: String

Length Constraints: Minimum length of 7. Maximum length of 15.

Pattern: `^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$`

Ipv6Address

The IPv6 address for the mount target.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 39.

LifeCycleState

Lifecycle state of the mount target.

Type: String

Valid Values: `creating | available | updating | deleting | deleted | error`

MountTargetId

System-assigned mount target ID.

Type: String

Length Constraints: Minimum length of 13. Maximum length of 45.

Pattern: `^fsmt-[0-9a-f]{8,40}$`

NetworkInterfaceId

The ID of the network interface that Amazon EFS created when it created the mount target.

Type: String

OwnerId

AWS account ID that owns the resource.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

SubnetId

The ID of the mount target's subnet.

Type: String

Length Constraints: Minimum length of 15. Maximum length of 47.

Pattern: `^subnet-[0-9a-f]{8,40}$`

VpcId

The virtual private cloud (VPC) ID that the mount target is configured in.

Type: String

Errors

AvailabilityZonesMismatch

Returned if the Availability Zone that was specified for the mount target is different from the file system's Availability Zone. For more information, see Regional and One Zone storage redundancy.

HTTP Status Code: 400

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

IpAddressInUse

Returned if the request specified an IP address (`IpAddress` or `Ipv6Address`) that is already in use in the subnet.

HTTP Status Code: 409

MountTargetConflict

Returned if the mount target would violate one of the specified restrictions based on the file system's existing mount targets.

HTTP Status Code: 409

NetworkInterfaceLimitExceeded

The calling account has reached the limit for elastic network interfaces for the specific AWS Region. Either delete some network interfaces or request that the account quota be raised. For more information, see Amazon VPC Quotas in the *Amazon VPC User Guide* (see the **Network interfaces per Region** entry in the **Network interfaces** table).

HTTP Status Code: 409

NoFreeAddressesInSubnet

Returned no `IpAddress` or `Ipv6Address` was provided in the request and there are no free IP addresses in the specified subnet.

HTTP Status Code: 409

SecurityGroupLimitExceeded

Returned if the number of `SecurityGroups` specified in the request is greater than the limit, which is based on account quota. Either delete some security groups or request that the account quota be raised. For more information, see [Amazon VPC Quotas](#) in the *Amazon VPC User Guide* (see the **Security Groups** table).

HTTP Status Code: 400

SecurityGroupNotFound

Returned if one of the specified security groups doesn't exist in the subnet's virtual private cloud (VPC).

HTTP Status Code: 400

SubnetNotFound

Returned if there is no subnet with ID `SubnetId` provided in the request.

HTTP Status Code: 400

UnsupportedAvailabilityZone

Returned if the requested Amazon EFS functionality is not available in the specified Availability Zone.

HTTP Status Code: 400

Examples

Create a mount target at an available IPv4 address on a subnet

The following request specifies only the file system and subnet for the mount target. The target is created at an available IPv4 address on the specified subnet, with the default security group associated with the VPC.

Sample Request

```
POST /2015-02-01/mount-targets HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T221118Z
```

```
Authorization: <...>
Content-Type: application/json
Content-Length: 160

{"SubnetId": "subnet-748c5d03", "FileSystemId": "fs-01234567"}
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 252

{
    "OwnerId": "231243201240",
    "MountTargetId": "fsmt-55a4413c",
    "FileSystemId": "fs-01234567",
    "SubnetId": "subnet-01234567",
    "LifeCycleState": "available",
    "IpAddress": "172.31.22.183"
    "NetworkInterfaceId":"eni-1bcb7772"
    "AvailabilityZoneId": "eus1-az2",
    "AvailabilityZoneName": "eu-south-1b",
    "VpcId": "vpc-08d45b31fa009a15e"
}
```

Create a mount target at a specific IPv4 address

The following request specifies the file system, subnet, security group, and IPv4 address to use for the mount target.

Sample Request

```
POST /2015-02-01/mount-targets HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T221118Z
Authorization: <...>
Content-Type: application/json
Content-Length: 160

{
   "FileSystemId":"fs-01234567",
   "SubnetId":"subnet-01234567",
```

```
    "IpAddress":"10.0.2.42",
    "SecurityGroups":[
       "sg-01234567"
    ]
}
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 252

{
    "OwnerId":"251839141158",
    "MountTargetId":"fsmt-9a13661e",
    "FileSystemId":"fs-01234567",
    "SubnetId":"subnet-fd04ff94",
    "LifeCycleState":"available",
    "IpAddress":"10.0.2.42",
    "NetworkInterfaceId":"eni-1bcb7772"
    "AvailabilityZoneId": "eus1-az2",
    "AvailabilityZoneName": "eu-south-1b",
    "VpcId": "vpc-08d45b31fa009a15e"
}
```

Create a mount target at a specific IPv6 address

The following request specifies the file system, subnet, security group, and IPv6 address to use for the mount target.

Sample Request

```
POST /2015-02-01/mount-targets HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T221118Z
Authorization: <...>
Content-Type: application/json
Content-Length: 160

{
    "FileSystemId":"fs-01234567",
    "SubnetId":"subnet-01234567",
```

```
    "Ipv6Address":"2001:0db8:85a3:0000:0000:8a2e:0370:7334",
    "IpAddressType": "IPV6_ONLY",
    "SecurityGroups":[
       "sg-01234567"
    ]
}
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 252

{
    "OwnerId":"251839141158",
    "MountTargetId":"fsmt-9a13661e",
    "FileSystemId":"fs-01234567",
    "SubnetId":"subnet-fd04ff94",
    "LifeCycleState":"available",
    "Ipv6Address":"2001:0db8:85a3:0000:0000:8a2e:0370:7334",
    "NetworkInterfaceId":"eni-1bcb7772"
    "AvailabilityZoneId": "eus1-az2",
    "AvailabilityZoneName": "eu-south-1b",
    "VpcId": "vpc-08d45b31fa009a15e"
}
```

Create a mount target at an available IPv4 address on dual-stack subnet

The following request specifies the file system, subnet, and address type for the mount target. The target is created at an available IPv4 address on the specified dual-stack subnet, with the default security group associated with the VPC.

Sample Request

```
POST /2015-02-01/mount-targets HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T221118Z
Authorization: <...>
Content-Type: application/json
Content-Length: 160

```

```
{"SubnetId": "subnet-748c5d03", "FileSystemId": "fs-01234567"}
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 252

{
    "OwnerId": "251839141158",
    "MountTargetId": "fsmt-55a4413c",
    "FileSystemId": "fs-01234567",
    "SubnetId":"subnet-fd04ff94",
    "LifeCycleState": "available",
    "IpAddress": "172.31.22.183"
    "Ipv6Address": "2a05:d01a:419:8611:3e57:75ab:5719:b238",
    "NetworkInterfaceId": "eni-01234567",
    "AvailabilityZoneId": "eus1-az2",
    "AvailabilityZoneName": "eu-south-1b",
    "VpcId": "vpc-08d45b31fa009a15e"
}
```

Create mount target at specific Ipv4 and IPv6 addresses on dual-stack subnet

The following request specifies the file system, subnet, security group, IPv4 address, IPv6 address for the mount target. The target is created at the specified IPv4 and IPv6 addresses on a dual-stack subnet.

Sample Request

```
POST /2015-02-01/mount-targets HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T221118Z
Authorization: <...>
Content-Type: application/json
Content-Length: 160

{
   "FileSystemId":"fs-01234567",
   "SubnetId":"subnet-01234567",
   "IpAddressType": "DUAL_STACK",
```

```
    "IpAddress": "10.0.1.25",
    "Ipv6Address":"2001:0db8:85a3:0000:0000:8a2e:0370:7334",
    "SecurityGroups":[
       "sg-01234567"
    ]
}
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 252

{
    "OwnerId": "231243201240",
    "MountTargetId": "fsmt-55a4413c",
    "FileSystemId": "fs-01234567",
    "SubnetId": "subnet-01234567",
    "LifeCycleState": "available",
    "IpAddress": "10.0.1.25",
    "Ipv6Address":"2001:0db8:85a3:0000:0000:8a2e:0370:7334",
    "NetworkInterfaceId": "eni-01234567",
    "AvailabilityZoneId": "eus1-az2",
    "AvailabilityZoneName": "eu-south-1b",
    "VpcId": "vpc-08d45b31fa009a15e"
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateReplicationConfiguration

Creates a replication configuration to either a new or existing EFS file system. For more information, see [Amazon EFS replication](#) in the *Amazon EFS User Guide*. The replication configuration specifies the following:

- **Source file system** – The EFS file system that you want to replicate.
- **Destination file system** – The destination file system to which the source file system is replicated. There can only be one destination file system in a replication configuration.

> **Note**
>
> A file system can be part of only one replication configuration.

The destination parameters for the replication configuration depend on whether you are replicating to a new file system or to an existing file system, and if you are replicating across AWS accounts. See DestinationToCreate for more information.

This operation requires permissions for the `elasticfilesystem:CreateReplicationConfiguration` action. Additionally, other permissions are required depending on how you are replicating file systems. For more information, see Required permissions for replication in the *Amazon EFS User Guide*.

Request Syntax

```
POST /2015-02-01/file-systems/SourceFileSystemId/replication-configuration HTTP/1.1
Content-type: application/json

{
   "Destinations": [
      {
         "AvailabilityZoneName": "string",
         "FileSystemId": "string",
         "KmsKeyId": "string",
         "Region": "string",
         "RoleArn": "string"
      }
   ]
```

```
}
```

URI Request Parameters

The request uses the following URI parameters.

SourceFileSystemId

Specifies the Amazon EFS file system that you want to replicate. This file system cannot already be a source or destination file system in another replication configuration.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

Destinations

An array of destination configuration objects. Only one destination configuration object is supported.

Type: Array of DestinationToCreate objects

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "CreationTime": number,
   "Destinations": [
      {
         "FileSystemId": "string",
```

```
            "LastReplicatedTimestamp": number,
            "OwnerId": "string",
            "Region": "string",
            "RoleArn": "string",
            "Status": "string",
            "StatusMessage": "string"
         }
      ],
      "OriginalSourceFileSystemArn": "string",
      "SourceFileSystemArn": "string",
      "SourceFileSystemId": "string",
      "SourceFileSystemOwnerId": "string",
      "SourceFileSystemRegion": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CreationTime

Describes when the replication configuration was created.

Type: Timestamp

Destinations

An array of destination objects. Only one destination object is supported.

Type: Array of Destination objects

OriginalSourceFileSystemArn

The Amazon Resource Name (ARN) of the original source EFS file system in the replication configuration.

Type: String

SourceFileSystemArn

The Amazon Resource Name (ARN) of the current source file system in the replication configuration.

Type: String

SourceFileSystemId

The ID of the source Amazon EFS file system that is being replicated.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

SourceFileSystemOwnerId

ID of the AWS account in which the source file system resides.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

SourceFileSystemRegion

The AWS Region in which the source EFS file system is located.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-z]{2}-((iso[a-z]{0,1}-)|(gov-)){0,1}[a-z]+-{0,1}[0-9]{0,1}$`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

ConflictException

Returned if the source file system in a replication is encrypted but the destination file system is unencrypted.

HTTP Status Code: 409

FileSystemLimitExceeded

Returned if the AWS account has already created the maximum number of file systems allowed per account.

HTTP Status Code: 403

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InsufficientThroughputCapacity

Returned if there's not enough capacity to provision additional throughput. This value might be returned when you try to create a file system in provisioned throughput mode, when you attempt to increase the provisioned throughput of an existing file system, or when you attempt to change an existing file system from Bursting Throughput to Provisioned Throughput mode. Try again later.

HTTP Status Code: 503

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ReplicationNotFound

Returned if the specified file system does not have a replication configuration.

HTTP Status Code: 404

ThroughputLimitExceeded

Returned if the throughput mode or amount of provisioned throughput can't be changed because the throughput limit of 1024 MiB/s has been reached.

HTTP Status Code: 400

UnsupportedAvailabilityZone

Returned if the requested Amazon EFS functionality is not available in the specified Availability Zone.

HTTP Status Code: 400

ValidationException

Returned if the AWS Backup service is not available in the AWS Region in which the request was made.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateTags

> **Note**
>
> DEPRECATED - `CreateTags` is deprecated and not maintained. To create tags for EFS resources, use the TagResource API action.

Creates or overwrites tags associated with a file system. Each tag is a key-value pair. If a tag key specified in the request already exists on the file system, this operation overwrites its value with the value provided in the request. If you add the `Name` tag to your file system, Amazon EFS returns it in the response to the DescribeFileSystems operation.

This operation requires permission for the `elasticfilesystem:CreateTags` action.

Request Syntax

```
POST /2015-02-01/create-tags/FileSystemId HTTP/1.1
Content-type: application/json

{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system whose tags you want to modify (String). This operation modifies the tags only, not the file system.

Length Constraints: Maximum length of 128.

Pattern: ^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$

Required: Yes

Request Body

The request accepts the following data in JSON format.

Tags

An array of `Tag` objects to add. Each `Tag` object is a key-value pair.

Type: Array of Tag objects

Required: Yes

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteAccessPoint

Deletes the specified access point. After deletion is complete, new clients can no longer connect to the access points. Clients connected to the access point at the time of deletion will continue to function until they terminate their connection.

This operation requires permissions for the `elasticfilesystem:DeleteAccessPoint` action.

Request Syntax

```
DELETE /2015-02-01/access-points/AccessPointId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

AccessPointId

The ID of the access point that you want to delete.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:access-point/fsap-[0-9a-f]{8,40}|fsap-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

AccessPointNotFound

Returned if the specified `AccessPointId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteFileSystem

Deletes a file system, permanently severing access to its contents. Upon return, the file system no longer exists and you can't access any contents of the deleted file system.

You need to manually delete mount targets attached to a file system before you can delete an EFS file system. This step is performed for you when you use the AWS console to delete a file system.

> **Note**
>
> You cannot delete a file system that is part of an EFS replication configuration. You need to delete the replication configuration first.

You can't delete a file system that is in use. That is, if the file system has any mount targets, you must first delete them. For more information, see DescribeMountTargets and DeleteMountTarget.

> **Note**
>
> The `DeleteFileSystem` call returns while the file system state is still `deleting`. You can check the file system deletion status by calling the DescribeFileSystems operation, which returns a list of file systems in your account. If you pass file system ID or creation token for the deleted file system, the DescribeFileSystems returns a `404 FileSystemNotFound` error.

This operation requires permissions for the `elasticfilesystem:DeleteFileSystem` action.

Request Syntax

```
DELETE /2015-02-01/file-systems/FileSystemId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system you want to delete.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemInUse

Returned if a file system has mount targets.

HTTP Status Code: 409

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

Examples

Delete a file system

The following example sends a DELETE request to the `file-systems` endpoint (`elasticfilesystem.us-west-2.amazonaws.com/2015-02-01/file-systems/fs-01234567`) to delete a file system whose ID is `fs-01234567`.

Sample Request

```
DELETE /2015-02-01/file-systems/fs-01234567 HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140622T233021Z
Authorization: <...>
```

Sample Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: a2d125b3-7ebd-4d6a-ab3d-5548630bff33
Content-Length: 0
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby V3

DeleteFileSystemPolicy

Deletes the `FileSystemPolicy` for the specified file system. The default `FileSystemPolicy` goes into effect once the existing policy is deleted. For more information about the default file system policy, see Using Resource-based Policies with EFS.

This operation requires permissions for the `elasticfilesystem:DeleteFileSystemPolicy` action.

Request Syntax

```
DELETE /2015-02-01/file-systems/FileSystemId/policy HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

Specifies the EFS file system for which to delete the `FileSystemPolicy`.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python

- [AWS SDK for Ruby V3]

DeleteMountTarget

Deletes the specified mount target.

This operation forcibly breaks any mounts of the file system by using the mount target that is being deleted, which might disrupt instances or applications using those mounts. To avoid applications getting cut off abruptly, you might consider unmounting any mounts of the mount target, if feasible. The operation also deletes the associated network interface. Uncommitted writes might be lost, but breaking a mount target using this operation does not corrupt the file system itself. The file system you created remains. You can mount an EC2 instance in your VPC by using another mount target.

This operation requires permissions for the following action on the file system:

- `elasticfilesystem:DeleteMountTarget`

> **Note**
>
> The `DeleteMountTarget` call returns while the mount target state is still `deleting`. You can check the mount target deletion by calling the DescribeMountTargets operation, which returns a list of mount target descriptions for the given file system.

The operation also requires permissions for the following Amazon EC2 action on the mount target's network interface:

- `ec2:DeleteNetworkInterface`

Request Syntax

```
DELETE /2015-02-01/mount-targets/MountTargetId HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

MountTargetId

The ID of the mount target to delete (String).

Length Constraints: Minimum length of 13. Maximum length of 45.

Pattern: `^fsmt-[0-9a-f]{8,40}$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

DependencyTimeout

The service timed out trying to fulfill the request, and the client should try the call again.

HTTP Status Code: 504

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

MountTargetNotFound

Returned if there is no mount target with the specified ID found in the caller's AWS account.

HTTP Status Code: 404

Examples

Remove a file system's mount target

The following example sends a DELETE request to delete a specific mount target.

Sample Request

```
DELETE /2015-02-01/mount-targets/fsmt-9a13661e HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140622T232908Z
Authorization: <...>
```

Sample Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteReplicationConfiguration

Deletes a replication configuration. Deleting a replication configuration ends the replication process. After a replication configuration is deleted, the destination file system becomes `Writeable` and its replication overwrite protection is re-enabled. For more information, see Delete a replication configuration.

This operation requires permissions for the `elasticfilesystem:DeleteReplicationConfiguration` action.

Request Syntax

```
DELETE /2015-02-01/file-systems/SourceFileSystemId/replication-configuration?
deletionMode=DeletionMode HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

DeletionMode

When replicating across AWS accounts or across AWS Regions, Amazon EFS deletes the replication configuration from both the source and destination account or Region (`ALL_CONFIGURATIONS`) by default. If there's a configuration or permissions issue that prevents Amazon EFS from deleting the replication configuration from both sides, you can use the `LOCAL_CONFIGURATION_ONLY` mode to delete the replication configuration from only the local side (the account or Region from which the delete is performed).

> **Note**
>
> Only use the `LOCAL_CONFIGURATION_ONLY` mode in the case that Amazon EFS is unable to delete the replication configuration in both the source and destination account or Region. Deleting the local configuration leaves the configuration in the other account or Region unrecoverable.
> Additionally, do not use this mode for same-account, same-region replication as doing so results in a BadRequest exception error.

Valid Values: `ALL_CONFIGURATIONS | LOCAL_CONFIGURATION_ONLY`

SourceFileSystemId

The ID of the source file system in the replication configuration.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ReplicationNotFound

Returned if the specified file system does not have a replication configuration.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteTags

Note

DEPRECATED - `DeleteTags` is deprecated and not maintained. To remove tags from EFS resources, use the UntagResource API action.

Deletes the specified tags from a file system. If the `DeleteTags` request includes a tag key that doesn't exist, Amazon EFS ignores it and doesn't cause an error. For more information about tags and related restrictions, see Tag restrictions in the *AWS Billing and Cost Management User Guide*.

This operation requires permissions for the `elasticfilesystem:DeleteTags` action.

Request Syntax

```
POST /2015-02-01/delete-tags/FileSystemId HTTP/1.1
Content-type: application/json

{
   "TagKeys": [ "string" ]
}
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system whose tags you want to delete (String).

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

TagKeys

A list of tag keys to delete.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^(?![aA]{1}[wW]{1}[sS]{1}:)([\p{L}\p{Z}\p{N}_.:/=+\-@]+)$`

Required: Yes

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeAccessPoints

Returns the description of a specific Amazon EFS access point if the AccessPointId is provided. If you provide an EFS FileSystemId, it returns descriptions of all access points for that file system. You can provide either an AccessPointId or a FileSystemId in the request, but not both.

This operation requires permissions for the elasticfilesystem:DescribeAccessPoints action.

Request Syntax

```
GET /2015-02-01/access-points?
AccessPointId=AccessPointId&FileSystemId=FileSystemId&MaxResults=MaxResults&NextToken=NextToken
 HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

AccessPointId

(Optional) Specifies an EFS access point to describe in the response; mutually exclusive with FileSystemId.

Length Constraints: Maximum length of 128.

Pattern: ^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:access-point/fsap-[0-9a-f]{8,40}|fsap-[0-9a-f]{8,40})$

FileSystemId

(Optional) If you provide a FileSystemId, EFS returns all access points for that file system; mutually exclusive with AccessPointId.

Length Constraints: Maximum length of 128.

Pattern: ^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$

MaxResults

(Optional) When retrieving all access points for a file system, you can optionally specify the `MaxItems` parameter to limit the number of objects returned in a response. The default value is 100.

Valid Range: Minimum value of 1.

NextToken

`NextToken` is present if the response is paginated. You can use `NextMarker` in the subsequent request to fetch the next page of access point descriptions.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "AccessPoints": [
      {
         "AccessPointArn": "string",
         "AccessPointId": "string",
         "ClientToken": "string",
         "FileSystemId": "string",
         "LifeCycleState": "string",
         "Name": "string",
         "OwnerId": "string",
         "PosixUser": {
            "Gid": number,
            "SecondaryGids": [ number ],
            "Uid": number
         },
         "RootDirectory": {
```

```
            "CreationInfo": {
               "OwnerGid": number,
               "OwnerUid": number,
               "Permissions": "string"
            },
            "Path": "string"
         },
         "Tags": [
            {
               "Key": "string",
               "Value": "string"
            }
         ]
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

AccessPoints

An array of access point descriptions.

Type: Array of AccessPointDescription objects

NextToken

Present if there are more access points than returned in the response. You can use the NextMarker in the subsequent request to fetch the additional descriptions.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Errors

AccessPointNotFound

Returned if the specified `AccessPointId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby V3

DescribeAccountPreferences

Returns the account preferences settings for the AWS account associated with the user making the request, in the current AWS Region.

Request Syntax

```
GET /2015-02-01/account-preferences HTTP/1.1
Content-type: application/json

{
   "MaxResults": number,
   "NextToken": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

MaxResults

(Optional) When retrieving account preferences, you can optionally specify the `MaxItems` parameter to limit the number of objects returned in a response. The default value is 100.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

NextToken

(Optional) You can use `NextToken` in a subsequent request to fetch the next page of AWS account preferences if the response payload was paginated.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "ResourceIdPreference": {
      "ResourceIdType": "string",
      "Resources": [ "string" ]
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

Present if there are more records than returned in the response. You can use the `NextToken` in the subsequent request to fetch the additional descriptions.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

ResourceIdPreference

Describes the resource ID preference setting for the AWS account associated with the user making the request, in the current AWS Region.

Type: ResourceIdPreference object

Errors

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeBackupPolicy

Returns the backup policy for the specified EFS file system.

Request Syntax

```
GET /2015-02-01/file-systems/FileSystemId/backup-policy HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

Specifies which EFS file system for which to retrieve the `BackupPolicy`.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "BackupPolicy": {
      "Status": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

BackupPolicy

Describes the file system's backup policy, indicating whether automatic backups are turned on or off.

Type: BackupPolicy object

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

PolicyNotFound

Returned if `no backup` is specified for a One Zone EFS file system.

HTTP Status Code: 404

ValidationException

Returned if the AWS Backup service is not available in the AWS Region in which the request was made.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeFileSystemPolicy

Returns the `FileSystemPolicy` for the specified EFS file system.

This operation requires permissions for the `elasticfilesystem:DescribeFileSystemPolicy` action.

Request Syntax

```
GET /2015-02-01/file-systems/FileSystemId/policy HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

Specifies which EFS file system to retrieve the `FileSystemPolicy` for.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "FileSystemId": "string",
   "Policy": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

FileSystemId

Specifies the EFS file system to which the `FileSystemPolicy` applies.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Policy

The JSON formatted `FileSystemPolicy` for the EFS file system.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 20000.

Pattern: `[\s\S]+`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

PolicyNotFound

Returned if `no backup` is specified for a One Zone EFS file system.

HTTP Status Code: 404

Examples

Example

This example illustrates one usage of DescribeFileSystemPolicy.

Sample Request

```
GET /2015-02-01/file-systems/fs-01234567/policy HTTP/1.1
```

Sample Response

```
{
    "FileSystemId": "fs-01234567",
    "Policy": "{
        "Version": "2012-10-17",
        "Id": "efs-policy-wizard-cdef0123-aaaa-6666-5555-444455556666",
        "Statement": [
            {
                "Sid": "efs-statement-abcdef01-1111-bbbb-2222-111122224444",
                "Effect" : "Deny",
                "Principal": {
                "AWS": "*"
                },
                "Action": "*",
                "Resource": "arn:aws:elasticfilesystem:us-east-2:111122223333:file-
system/fs-01234567",
                "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                    }
                }
            },
            {
                "Sid": "efs-statement-01234567-aaaa-3333-4444-111122223333",
```

```
                "Effect": "Allow",
                "Principal": {
                    "AWS": "*"
                },
                "Action": [
                    "elasticfilesystem:ClientMount",
                    "elasticfilesystem:ClientWrite"
                ],
                "Resource" : "arn:aws:elasticfilesystem:us-east-2:111122223333:file-
system/fs-01234567"
            }
        ]
    }
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeFileSystems

Returns the description of a specific Amazon EFS file system if either the file system `CreationToken` or the `FileSystemId` is provided. Otherwise, it returns descriptions of all file systems owned by the caller's AWS account in the AWS Region of the endpoint that you're calling.

When retrieving all file system descriptions, you can optionally specify the `MaxItems` parameter to limit the number of descriptions in a response. This number is automatically set to 100. If more file system descriptions remain, Amazon EFS returns a `NextMarker`, an opaque token, in the response. In this case, you should send a subsequent request with the `Marker` request parameter set to the value of `NextMarker`.

To retrieve a list of your file system descriptions, this operation is used in an iterative process, where `DescribeFileSystems` is called first without the `Marker` and then the operation continues to call it with the `Marker` parameter set to the value of the `NextMarker` from the previous response until the response has no `NextMarker`.

The order of file systems returned in the response of one `DescribeFileSystems` call and the order of file systems returned across the responses of a multi-call iteration is unspecified.

This operation requires permissions for the `elasticfilesystem:DescribeFileSystems` action.

Request Syntax

```
GET /2015-02-01/file-systems?
CreationToken=CreationToken&FileSystemId=FileSystemId&Marker=Marker&MaxItems=MaxItems
 HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

CreationToken

(Optional) Restricts the list to the file system with this creation token (String). You specify a creation token when you create an Amazon EFS file system.

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

FileSystemId

(Optional) ID of the file system whose description you want to retrieve (String).

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Marker

(Optional) Opaque pagination token returned from a previous `DescribeFileSystems` operation (String). If present, specifies to continue the list from where the returning call had left off.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

MaxItems

(Optional) Specifies the maximum number of file systems to return in the response (integer). This number is automatically set to 100. The response is paginated at 100 per page if you have more than 100 file systems.

Valid Range: Minimum value of 1.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "FileSystems": [
      {
         "AvailabilityZoneId": "string",
         "AvailabilityZoneName": "string",
```

```
            "CreationTime": number,
            "CreationToken": "string",
            "Encrypted": boolean,
            "FileSystemArn": "string",
            "FileSystemId": "string",
            "FileSystemProtection": {
               "ReplicationOverwriteProtection": "string"
            },
            "KmsKeyId": "string",
            "LifeCycleState": "string",
            "Name": "string",
            "NumberOfMountTargets": number,
            "OwnerId": "string",
            "PerformanceMode": "string",
            "ProvisionedThroughputInMibps": number,
            "SizeInBytes": {
               "Timestamp": number,
               "Value": number,
               "ValueInArchive": number,
               "ValueInIA": number,
               "ValueInStandard": number
            },
            "Tags": [
               {
                  "Key": "string",
                  "Value": "string"
               }
            ],
            "ThroughputMode": "string"
         }
      ],
      "Marker": "string",
      "NextMarker": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

FileSystems

An array of file system descriptions.

Type: Array of [FileSystemDescription] objects

Marker

Present if provided by caller in the request (String).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

NextMarker

Present if there are more file systems than returned in the response (String). You can use the `NextMarker` in the subsequent request to fetch the descriptions.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

Examples

Retrieve a list of 10 file systems

The following example sends a GET request to the `file-systems` endpoint (`elasticfilesystem.us-west-2.amazonaws.com/2015-02-01/file-systems`). The request specifies a `MaxItems` query parameter to limit the number of file system descriptions to 10.

Sample Request

```
GET /2015-02-01/file-systems?MaxItems=10 HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140622T191208Z
Authorization: <...>
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 499
{
   "FileSystems":[
      {
         "OwnerId":"251839141158",
         "CreationToken":"MyFileSystem1",
         "FileSystemId":"fs-01234567",
         "PerformanceMode" : "generalPurpose",
         "CreationTime":"1403301078",
         "LifeCycleState":"created",
         "Name":"my first file system",
         "NumberOfMountTargets":1,
         "SizeInBytes":{
            "Timestamp": 1403301078,
            "Value": 29313618372,
            "ValueInArchive": 201156,
            "ValueInIA": 675432,
            "ValueInStandard": 29312741784
         }
      }
   ]
```

```
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeLifecycleConfiguration

Returns the current `LifecycleConfiguration` object for the specified EFS file system. Lifecycle management uses the `LifecycleConfiguration` object to identify when to move files between storage classes. For a file system without a `LifecycleConfiguration` object, the call returns an empty array in the response.

This operation requires permissions for the `elasticfilesystem:DescribeLifecycleConfiguration` operation.

Request Syntax

```
GET /2015-02-01/file-systems/FileSystemId/lifecycle-configuration HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system whose `LifecycleConfiguration` object you want to retrieve (String).

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "LifecyclePolicies": [
      {
         "TransitionToArchive": "string",
         "TransitionToIA": "string",
         "TransitionToPrimaryStorageClass": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

LifecyclePolicies

An array of lifecycle management policies. EFS supports a maximum of one policy per file system.

Type: Array of LifecyclePolicy objects

Array Members: Maximum number of 3 items.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

Examples

Retrieve the lifecycle configuration for a file system

The following request retrieves the `LifecycleConfiguration` object for the specified file system.

Sample Request

```
GET /2015-02-01/file-systems/fs-01234567/lifecycle-configuration HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20181120T221118Z
Authorization: <...>
```

Sample Response

```
HTTP/1.1 200 OK
        x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
        Content-Type: application/json
        Content-Length: 86
{
  "LifecyclePolicies": [
    {
        "TransitionToArchive": "AFTER_270_DAYS"
    },
    {
        "TransitionToIA": "AFTER_14_DAYS"
    },
    {
        "TransitionToPrimaryStorageClass": "AFTER_1_ACCESS"
    }
  ]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeMountTargets

Returns the descriptions of all the current mount targets, or a specific mount target, for a file system. When requesting all of the current mount targets, the order of mount targets returned in the response is unspecified.

This operation requires permissions for the `elasticfilesystem:DescribeMountTargets` action, on either the file system ID that you specify in `FileSystemId`, or on the file system of the mount target that you specify in `MountTargetId`.

The corresponding CLI command is `describe-mount-targets`.

Request Syntax

```
GET /2015-02-01/mount-targets?
AccessPointId=AccessPointId&FileSystemId=FileSystemId&Marker=Marker&MaxItems=MaxItems&MountTarg
 HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

AccessPointId

(Optional) The ID of the access point whose mount targets that you want to list. It must be included in your request if a `FileSystemId` or `MountTargetId` is not included in your request. Accepts either an access point ID or ARN as input.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:access-point/fsap-[0-9a-f]{8,40}|fsap-[0-9a-f]{8,40})$`

FileSystemId

(Optional) ID of the file system whose mount targets you want to list (String). It must be included in your request if an `AccessPointId` or `MountTargetId` is not included. Accepts either a file system ID or ARN as input.

Length Constraints: Maximum length of 128.

Pattern: ^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$

Marker

(Optional) Opaque pagination token returned from a previous DescribeMountTargets operation (String). If present, it specifies to continue the list from where the previous returning call left off.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: .+

MaxItems

(Optional) Maximum number of mount targets to return in the response. Currently, this number is automatically set to 10, and other values are ignored. The response is paginated at 100 per page if you have more than 100 mount targets.

Valid Range: Minimum value of 1.

MountTargetId

(Optional) ID of the mount target that you want to have described (String). It must be included in your request if FileSystemId is not included. Accepts either a mount target ID or ARN as input.

Length Constraints: Minimum length of 13. Maximum length of 45.

Pattern: ^fsmt-[0-9a-f]{8,40}$

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Marker": "string",
```

```
   "MountTargets": [
      {
         "AvailabilityZoneId": "string",
         "AvailabilityZoneName": "string",
         "FileSystemId": "string",
         "IpAddress": "string",
         "Ipv6Address": "string",
         "LifeCycleState": "string",
         "MountTargetId": "string",
         "NetworkInterfaceId": "string",
         "OwnerId": "string",
         "SubnetId": "string",
         "VpcId": "string"
      }
   ],
   "NextMarker": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Marker

If the request included the `Marker`, the response returns that value in this field.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

MountTargets

Returns the file system's mount targets as an array of `MountTargetDescription` objects.

Type: Array of MountTargetDescription objects

NextMarker

If a value is present, there are more mount targets to return. In a subsequent request, you can provide `Marker` in your request with this value to retrieve the next set of mount targets.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Errors

AccessPointNotFound

Returned if the specified `AccessPointId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

MountTargetNotFound

Returned if there is no mount target with the specified ID found in the caller's AWS account.

HTTP Status Code: 404

Examples

Retrieve descriptions of mount targets created for a file system

The following request retrieves descriptions of mount targets created for the specified file system.

Sample Request

```
GET /2015-02-01/mount-targets?FileSystemId=fs-01234567 HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140622T191252Z
Authorization: <...>
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 357

{
   "MountTargets":[
      {
         "OwnerId":"251839141158",
         "MountTargetId":"fsmt-01234567",
         "FileSystemId":"fs-01234567",
         "SubnetId":"subnet-01234567",
         "LifeCycleState":"added",
         "IpAddress":"10.0.2.42",
         "NetworkInterfaceId":"eni-1bcb7772"
      }
   ]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeMountTargetSecurityGroups

Returns the security groups currently in effect for a mount target. This operation requires that the network interface of the mount target has been created and the lifecycle state of the mount target is not `deleted`.

This operation requires permissions for the following actions:

- `elasticfilesystem:DescribeMountTargetSecurityGroups` action on the mount target's file system.
- `ec2:DescribeNetworkInterfaceAttribute` action on the mount target's network interface.

Request Syntax

```
GET /2015-02-01/mount-targets/MountTargetId/security-groups HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

MountTargetId

The ID of the mount target whose security groups you want to retrieve.

Length Constraints: Minimum length of 13. Maximum length of 45.

Pattern: `^fsmt-[0-9a-f]{8,40}$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
```

```
   "SecurityGroups": [ "string" ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

SecurityGroups

An array of security groups.

Type: Array of strings

Array Members: Maximum number of 100 items.

Length Constraints: Minimum length of 11. Maximum length of 43.

Pattern: `^sg-[0-9a-f]{8,40}`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

IncorrectMountTargetState

Returned if the mount target is not in the correct state for the operation.

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

MountTargetNotFound

Returned if there is no mount target with the specified ID found in the caller's AWS account.

HTTP Status Code: 404

Examples

Retrieve security groups in effect for a file system

The following example retrieves the security groups that are in effect for the network interface associated with a mount target.

Sample Request

```
GET /2015-02-01/mount-targets/fsmt-9a13661e/security-groups HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T223513Z
Authorization: <...>
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Length: 57

{
"SecurityGroups" : [
"sg-188d9f74"
]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3

- [AWS SDK for Kotlin]
- [AWS SDK for PHP V3]
- [AWS SDK for Python]
- [AWS SDK for Ruby V3]

DescribeReplicationConfigurations

Retrieves the replication configuration for a specific file system. If a file system is not specified, all of the replication configurations for the AWS account in an AWS Region are retrieved.

Request Syntax

```
GET /2015-02-01/file-systems/replication-configurations?
FileSystemId=FileSystemId&MaxResults=MaxResults&NextToken=NextToken HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

You can retrieve the replication configuration for a specific file system by providing its file system ID. For cross-account,cross-region replication, an account can only describe the replication configuration for a file system in its own Region.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

MaxResults

(Optional) To limit the number of objects returned in a response, you can specify the `MaxItems` parameter. The default value is 100.

Valid Range: Minimum value of 1.

NextToken

`NextToken` is present if the response is paginated. You can use `NextToken` in a subsequent request to fetch the next page of output.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Replications": [
      {
         "CreationTime": number,
         "Destinations": [
            {
               "FileSystemId": "string",
               "LastReplicatedTimestamp": number,
               "OwnerId": "string",
               "Region": "string",
               "RoleArn": "string",
               "Status": "string",
               "StatusMessage": "string"
            }
         ],
         "OriginalSourceFileSystemArn": "string",
         "SourceFileSystemArn": "string",
         "SourceFileSystemId": "string",
         "SourceFileSystemOwnerId": "string",
         "SourceFileSystemRegion": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

You can use the `NextToken` from the previous response in a subsequent request to fetch the additional descriptions.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Replications

The collection of replication configurations that is returned.

Type: Array of ReplicationConfigurationDescription objects

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ReplicationNotFound

Returned if the specified file system does not have a replication configuration.

HTTP Status Code: 404

ValidationException

Returned if the AWS Backup service is not available in the AWS Region in which the request was made.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeTags

Note

DEPRECATED - The `DescribeTags` action is deprecated and not maintained. To view tags associated with EFS resources, use the `ListTagsForResource` API action.

Returns the tags associated with a file system. The order of tags returned in the response of one `DescribeTags` call and the order of tags returned across the responses of a multiple-call iteration (when using pagination) is unspecified.

This operation requires permissions for the `elasticfilesystem:DescribeTags` action.

Request Syntax

```
GET /2015-02-01/tags/FileSystemId/?Marker=Marker&MaxItems=MaxItems HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system whose tag set you want to retrieve.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Marker

(Optional) An opaque pagination token returned from a previous `DescribeTags` operation (String). If present, it specifies to continue the list from where the previous call left off.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

MaxItems

(Optional) The maximum number of file system tags to return in the response. Currently, this number is automatically set to 100, and other values are ignored. The response is paginated at 100 per page if you have more than 100 tags.

Valid Range: Minimum value of 1.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Marker": "string",
   "NextMarker": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Marker

If the request included a `Marker`, the response returns that value in this field.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

NextMarker

If a value is present, there are more tags to return. In a subsequent request, you can provide the value of `NextMarker` as the value of the `Marker` parameter in your next request to retrieve the next set of tags.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Tags

Returns tags associated with the file system as an array of `Tag` objects.

Type: Array of Tag objects

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

Examples

Retrieve tags associated with a file system

The following request retrieves tags (key-value pairs) associated with the specified file system.

Sample Request

```
GET /2015-02-01/tags/fs-01234567/ HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T215404Z
Authorization: <...>
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: application/json
Content-Length: 288

{
    "Tags":[
        {
            "Key":"Name",
            "Value":"my first file system"
        },
        {
            "Key":"Fleet",
            "Value":"Development"
        },
        {
            "Key":"Developer",
            "Value":"Alice"
        }
    ]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListTagsForResource

Lists all tags for a top-level EFS resource. You must provide the ID of the resource that you want to retrieve the tags for.

This operation requires permissions for the `elasticfilesystem:DescribeAccessPoints` action.

Request Syntax

```
GET /2015-02-01/resource-tags/ResourceId?MaxResults=MaxResults&NextToken=NextToken
 HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

MaxResults

(Optional) Specifies the maximum number of tag objects to return in the response. The default value is 100.

Valid Range: Minimum value of 1.

NextToken

(Optional) You can use `NextToken` in a subsequent request to fetch the next page of access point descriptions if the response payload was paginated.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

ResourceId

Specifies the EFS resource you want to retrieve tags for. You can retrieve tags for EFS file systems and access points using this API endpoint.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:(access-point/fsap|file-system/fs)-[0-9a-f]{8,40}|fs(ap)?-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken

`NextToken` is present if the response payload is paginated. You can use `NextToken` in a subsequent request to fetch the next page of access point descriptions.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `.+`

Tags

An array of the tags for the specified EFS resource.

Type: Array of Tag objects

Errors

AccessPointNotFound

Returned if the specified `AccessPointId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python

- [AWS SDK for Ruby V3]

ModifyMountTargetSecurityGroups

Modifies the set of security groups in effect for a mount target.

When you create a mount target, Amazon EFS also creates a new network interface. For more information, see CreateMountTarget. This operation replaces the security groups in effect for the network interface associated with a mount target, with the `SecurityGroups` provided in the request. This operation requires that the network interface of the mount target has been created and the lifecycle state of the mount target is not `deleted`.

The operation requires permissions for the following actions:

- `elasticfilesystem:ModifyMountTargetSecurityGroups` action on the mount target's file system.
- `ec2:ModifyNetworkInterfaceAttribute` action on the mount target's network interface.

Request Syntax

```
PUT /2015-02-01/mount-targets/MountTargetId/security-groups HTTP/1.1
Content-type: application/json

{
   "SecurityGroups": [ "string" ]
}
```

URI Request Parameters

The request uses the following URI parameters.

MountTargetId

The ID of the mount target whose security groups you want to modify.

Length Constraints: Minimum length of 13. Maximum length of 45.

Pattern: `^fsmt-[0-9a-f]{8,40}$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

SecurityGroups

An array of VPC security group IDs.

Type: Array of strings

Array Members: Maximum number of 100 items.

Length Constraints: Minimum length of 11. Maximum length of 43.

Pattern: `^sg-[0-9a-f]{8,40}`

Required: No

Response Syntax

```
HTTP/1.1 204
```

Response Elements

If the action is successful, the service sends back an HTTP 204 response with an empty HTTP body.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

IncorrectMountTargetState

Returned if the mount target is not in the correct state for the operation.

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

MountTargetNotFound

Returned if there is no mount target with the specified ID found in the caller's AWS account.

HTTP Status Code: 404

SecurityGroupLimitExceeded

Returned if the number of `SecurityGroups` specified in the request is greater than the limit, which is based on account quota. Either delete some security groups or request that the account quota be raised. For more information, see [Amazon VPC Quotas](https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html) in the *Amazon VPC User Guide* (see the **Security Groups** table).

HTTP Status Code: 400

SecurityGroupNotFound

Returned if one of the specified security groups doesn't exist in the subnet's virtual private cloud (VPC).

HTTP Status Code: 400

Examples

Replace a mount target's security groups

The following example replaces security groups in effect for the network interface associated with a mount target.

Sample Request

```
PUT /2015-02-01/mount-targets/fsmt-9a13661e/security-groups HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T223446Z
Authorization: <...>
Content-Type: application/json
Content-Length: 57

{
  "SecurityGroups" : [
  "sg-188d9f74"
  ]
```

```
}
```

Sample Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutAccountPreferences

Use this operation to set the account preference in the current AWS Region to use long 17 character (63 bit) or short 8 character (32 bit) resource IDs for new EFS file system and mount target resources. All existing resource IDs are not affected by any changes you make. You can set the ID preference during the opt-in period as EFS transitions to long resource IDs. For more information, see [Managing Amazon EFS resource IDs](#).

> **Note**
>
> Starting in October, 2021, you will receive an error if you try to set the account preference to use the short 8 character format resource ID. Contact AWS support if you receive an error and must use short IDs for file system and mount target resources.

Request Syntax

```
PUT /2015-02-01/account-preferences HTTP/1.1
Content-type: application/json

{
   "ResourceIdType": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

ResourceIdType

Specifies the EFS resource ID preference to set for the user's AWS account, in the current AWS Region, either `LONG_ID` (17 characters), or `SHORT_ID` (8 characters).

> **Note**
>
> Starting in October, 2021, you will receive an error when setting the account preference to SHORT_ID. Contact AWS support if you receive an error and must use short IDs for file system and mount target resources.

Type: String

Valid Values: LONG_ID | SHORT_ID

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ResourceIdPreference": {
      "ResourceIdType": "string",
      "Resources": [ "string" ]
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ResourceIdPreference

Describes the resource type and its ID preference for the user's AWS account, in the current AWS Region.

Type: ResourceIdPreference object

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutBackupPolicy

Updates the file system's backup policy. Use this action to start or stop automatic backups of the file system.

Request Syntax

```
PUT /2015-02-01/file-systems/FileSystemId/backup-policy HTTP/1.1
Content-type: application/json

{
   "BackupPolicy": {
      "Status": "string"
   }
}
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

Specifies which EFS file system to update the backup policy for.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

BackupPolicy

The backup policy included in the `PutBackupPolicy` request.

Type: BackupPolicy object

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "BackupPolicy": {
      "Status": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

BackupPolicy

Describes the file system's backup policy, indicating whether automatic backups are turned on or off.

Type: BackupPolicy object

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ValidationException

Returned if the AWS Backup service is not available in the AWS Region in which the request was made.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutFileSystemPolicy

Applies an Amazon EFS `FileSystemPolicy` to an Amazon EFS file system. A file system policy is an IAM resource-based policy and can contain multiple policy statements. A file system always has exactly one file system policy, which can be the default policy or an explicit policy set or updated using this API operation. EFS file system policies have a 20,000 character limit. When an explicit policy is set, it overrides the default policy. For more information about the default file system policy, see [Default EFS file system policy](#).

> **Note**
>
> EFS file system policies have a 20,000 character limit.

This operation requires permissions for the `elasticfilesystem:PutFileSystemPolicy` action.

Request Syntax

```
PUT /2015-02-01/file-systems/FileSystemId/policy HTTP/1.1
Content-type: application/json

{
   "BypassPolicyLockoutSafetyCheck": boolean,
   "Policy": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the EFS file system that you want to create or update the `FileSystemPolicy` for.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

BypassPolicyLockoutSafetyCheck

(Optional) A boolean that specifies whether or not to bypass the `FileSystemPolicy` lockout safety check. The lockout safety check determines whether the policy in the request will lock out, or prevent, the IAM principal that is making the request from making future `PutFileSystemPolicy` requests on this file system. Set `BypassPolicyLockoutSafetyCheck` to `True` only when you intend to prevent the IAM principal that is making the request from making subsequent `PutFileSystemPolicy` requests on this file system. The default value is `False`.

Type: Boolean

Required: No

Policy

The `FileSystemPolicy` that you're creating. Accepts a JSON formatted policy definition. EFS file system policies have a 20,000 character limit. To find out more about the elements that make up a file system policy, see Resource-based policies within Amazon EFS.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 20000.

Pattern: `[\s\S]+`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "FileSystemId": "string",
```

```
    "Policy": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

FileSystemId

Specifies the EFS file system to which the `FileSystemPolicy` applies.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Policy

The JSON formatted `FileSystemPolicy` for the EFS file system.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 20000.

Pattern: `[\s\S]+`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

InvalidPolicyException

Returned if the `FileSystemPolicy` is malformed or contains an error such as a parameter value that is not valid or a missing required parameter. Returned in the case of a policy lockout safety check error.

HTTP Status Code: 400

Examples

Create an EFS FileSystemPolicy

The following request creates a `FileSystemPolicy` that allows all AWS principals to mount the specified EFS file system with read and write permissions.

Sample Request

```
PUT /2015-02-01/file-systems/fs-01234567/file-system-policy HTTP/1.1
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite"
            ],
            "Principal": {
                "AWS": ["*"]
            },
        }
```

```
    ]
}
```

Sample Response

```
{
    "Version": "2012-10-17",
    "Id": "1",
    "Statement": [
        {
            "Sid": "efs-statement-abcdef01-1111-bbbb-2222-111122224444",
            "Effect": "Allow",
            "Action": [
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite"
            ],
            "Principal": {
                "AWS": ["*"]
            },
            "Resource":"arn:aws:elasticfilesystem:us-east-1:111122223333444:file-
system/fs-01234567"
        }
    ]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby V3

PutLifecycleConfiguration

Use this action to manage storage for your file system. A `LifecycleConfiguration` consists of one or more `LifecyclePolicy` objects that define the following:

- **`TransitionToIA`** – When to move files in the file system from primary storage (Standard storage class) into the Infrequent Access (IA) storage.
- **`TransitionToArchive`** – When to move files in the file system from their current storage class (either IA or Standard storage) into the Archive storage.

  File systems cannot transition into Archive storage before transitioning into IA storage. Therefore, TransitionToArchive must either not be set or must be later than TransitionToIA.

  > **Note**
  >
  > The Archive storage class is available only for file systems that use the Elastic throughput mode and the General Purpose performance mode.

- **`TransitionToPrimaryStorageClass`** – Whether to move files in the file system back to primary storage (Standard storage class) after they are accessed in IA or Archive storage.

For more information, see [Managing file system storage](#).

Each Amazon EFS file system supports one lifecycle configuration, which applies to all files in the file system. If a `LifecycleConfiguration` object already exists for the specified file system, a `PutLifecycleConfiguration` call modifies the existing configuration. A `PutLifecycleConfiguration` call with an empty `LifecyclePolicies` array in the request body deletes any existing `LifecycleConfiguration`. In the request, specify the following:

- The ID for the file system for which you are enabling, disabling, or modifying lifecycle management.
- A `LifecyclePolicies` array of `LifecyclePolicy` objects that define when to move files to IA storage, to Archive storage, and back to primary storage.

> **Note**
>
> Amazon EFS requires that each `LifecyclePolicy` object have only have a single transition, so the `LifecyclePolicies` array needs to be structured with separate `LifecyclePolicy` objects. See the example requests in the following section for more information.

This operation requires permissions for the `elasticfilesystem:PutLifecycleConfiguration` operation.

To apply a `LifecycleConfiguration` object to an encrypted file system, you need the same AWS Key Management Service permissions as when you created the encrypted file system.

Request Syntax

```
PUT /2015-02-01/file-systems/FileSystemId/lifecycle-configuration HTTP/1.1
Content-type: application/json

{
   "LifecyclePolicies": [
      {
         "TransitionToArchive": "string",
         "TransitionToIA": "string",
         "TransitionToPrimaryStorageClass": "string"
      }
   ]
}
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system for which you are creating the `LifecycleConfiguration` object (String).

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

LifecyclePolicies

An array of `LifecyclePolicy` objects that define the file system's `LifecycleConfiguration` object. A `LifecycleConfiguration` object informs lifecycle management of the following:

- **`TransitionToIA`** – When to move files in the file system from primary storage (Standard storage class) into the Infrequent Access (IA) storage.
- **`TransitionToArchive`** – When to move files in the file system from their current storage class (either IA or Standard storage) into the Archive storage.

  File systems cannot transition into Archive storage before transitioning into IA storage. Therefore, TransitionToArchive must either not be set or must be later than TransitionToIA.

  > **Note**
  >
  > The Archive storage class is available only for file systems that use the Elastic throughput mode and the General Purpose performance mode.

- **`TransitionToPrimaryStorageClass`** – Whether to move files in the file system back to primary storage (Standard storage class) after they are accessed in IA or Archive storage.

> **Note**
>
> When using the `put-lifecycle-configuration` CLI command or the `PutLifecycleConfiguration` API action, Amazon EFS requires that each `LifecyclePolicy` object have only a single transition. This means that in a request body, `LifecyclePolicies` must be structured as an array of `LifecyclePolicy` objects, one object for each storage transition. See the example requests in the following section for more information.

Type: Array of LifecyclePolicy objects

Array Members: Maximum number of 3 items.

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "LifecyclePolicies": [
      {
         "TransitionToArchive": "string",
         "TransitionToIA": "string",
         "TransitionToPrimaryStorageClass": "string"
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

LifecyclePolicies

An array of lifecycle management policies. EFS supports a maximum of one policy per file system.

Type: Array of LifecyclePolicy objects

Array Members: Maximum number of 3 items.

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

Examples

Create a lifecycle configuration

The following example creates a `LifecyclePolicy` object using the `PutLifecycleConfiguration` action. This example creates a lifecycle policy that instructs EFS to do the following:

- Move all files in the file system that haven't been accessed in Standard storage within the last 30 days to IA storage.
- Move all files in the file system that haven't been accessed in Standard storage within the last 90 days to Archive storage.
- Move files back to Standard storage after they are accessed in IA or Archive storage. The Archive storage class is available only for file systems that use the Elastic throughput mode and the General Purpose performance mode.

For more information, see EFS storage classes and Managing file system storage.

Sample Request

```
PUT /2015-02-01/file-systems/fs-0123456789abcdefb/lifecycle-configuration HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
```

```
x-amz-date: 20181122T232908Z
Authorization: <...>
Content-type: application/json
Content-Length: 86

{
   "LifecyclePolicies": [
      {
         "TransitionToArchive": "AFTER_90_DAYS"
      },
      {
         "TransitionToIA": "AFTER_30_DAYS"
      },
      {
         "TransitionToPrimaryStorage": "AFTER_1_ACCESS"
      }
   ]
}
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-type: application/json
Content-Length: 86

{
    "LifecyclePolicies": [
      {
         "TransitionToArchive": "AFTER_90_DAYS"
      },
      {
         "TransitionToIA": "AFTER_30_DAYS"
      },
      {
         "TransitionToPrimaryStorage": "AFTER_1_ACCESS"
      }
    ]
}
```

Example put-lifecycle-configuration CLI request

This example illustrates one usage of PutLifecycleConfiguration.

Sample Request

```
aws efs put-lifecycle-configuration \
   --file-system-id fs-0123456789abcdefb \
   --lifecycle-policies "[{"TransitionToArchive":"AFTER_90_DAYS"},
     {"TransitionToIA":"AFTER_30_DAYS"},
     {"TransitionToPrimaryStorageClass":"AFTER_1_ACCESS"}]
   --region us-west-2 \
   --profile adminuser
```

Sample Response

```
{
   "LifecyclePolicies": [
       {
           "TransitionToArchive": "AFTER_90_DAYS"
       },
       {
           "TransitionToIA": "AFTER_30_DAYS"
       },
       {
           "TransitionToPrimaryStorageClass": "AFTER_1_ACCESS"
       }
   ]
}
```

Disable lifecycle management

The following example disables lifecycle management for the specified file system.

Sample Request

```

PUT /2015-02-01/file-systems/fs-01234567/lifecycle-configuration HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20181122T232908Z
Authorization: <...>
Content-type: application/json
Content-Length: 86

{
```

```
    "LifecyclePolicies": [ ]
}
```

Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-type: application/json
Content-Length: 86

{
    "LifecyclePolicies": [ ]
}
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

TagResource

Creates a tag for an EFS resource. You can create tags for EFS file systems and access points using this API operation.

This operation requires permissions for the `elasticfilesystem:TagResource` action.

Request Syntax

```
POST /2015-02-01/resource-tags/ResourceId HTTP/1.1
Content-type: application/json

{
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ]
}
```

URI Request Parameters

The request uses the following URI parameters.

ResourceId

The ID specifying the EFS resource that you want to create a tag for.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:(access-point/fsap|file-system/fs)-[0-9a-f]{8,40}|fs(ap)?-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

Tags

An array of `Tag` objects to add. Each `Tag` object is a key-value pair.

Type: Array of Tag objects

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

AccessPointNotFound

Returned if the specified `AccessPointId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

Examples

Create Tags on a File System

The following request creates three tags ("key1", "key2", and "key3") on the specified file system.

Sample Request

```
POST /2015-02-01/tag-resource/fs-01234567 HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T221118Z
Authorization: <...>
Content-Type: application/json
Content-Length: 160

{
    "Tags": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": "value2"
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
    ]
}
```

Sample Response

```
HTTP/1.1 204 no content
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UntagResource

Removes tags from an EFS resource. You can remove tags from EFS file systems and access points using this API operation.

This operation requires permissions for the `elasticfilesystem:UntagResource` action.

Request Syntax

```
DELETE /2015-02-01/resource-tags/ResourceId?tagKeys=TagKeys HTTP/1.1
```

URI Request Parameters

The request uses the following URI parameters.

ResourceId

Specifies the EFS resource that you want to remove tags from.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:(access-point/fsap|file-system/fs)-[0-9a-f]{8,40}|fs(ap)?-[0-9a-f]{8,40})$`

Required: Yes

TagKeys

The keys of the key-value tag pairs that you want to remove from the specified EFS resource.

Array Members: Minimum number of 1 item. Maximum number of 50 items.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^(?![aA]{1}[wW]{1}[sS]{1}:)([\p{L}\p{Z}\p{N}_.:/=+\-@]+)$`

Required: Yes

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

AccessPointNotFound

Returned if the specified `AccessPointId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateFileSystem

Updates the throughput mode or the amount of provisioned throughput of an existing file system.

Request Syntax

```
PUT /2015-02-01/file-systems/FileSystemId HTTP/1.1
Content-type: application/json

{
   "ProvisionedThroughputInMibps": number,
   "ThroughputMode": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system that you want to update.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

ProvisionedThroughputInMibps

(Optional) The throughput, measured in mebibytes per second (MiBps), that you want to provision for a file system that you're creating. Required if `ThroughputMode` is set to `provisioned`. Valid values are 1-3414 MiBps, with the upper limit depending on Region. To increase this limit, contact Support. For more information, see Amazon EFS quotas that you can increase in the *Amazon EFS User Guide*.

Type: Double

Valid Range: Minimum value of 1.0.

Required: No

ThroughputMode

(Optional) Updates the file system's throughput mode. If you're not updating your throughput mode, you don't need to provide this value in your request. If you are changing the `ThroughputMode` to `provisioned`, you must also set a value for `ProvisionedThroughputInMibps`.

Type: String

Valid Values: `bursting | provisioned | elastic`

Required: No

Response Syntax

```
HTTP/1.1 202
Content-type: application/json

{
   "AvailabilityZoneId": "string",
   "AvailabilityZoneName": "string",
   "CreationTime": number,
   "CreationToken": "string",
   "Encrypted": boolean,
   "FileSystemArn": "string",
   "FileSystemId": "string",
   "FileSystemProtection": {
      "ReplicationOverwriteProtection": "string"
   },
   "KmsKeyId": "string",
   "LifeCycleState": "string",
   "Name": "string",
   "NumberOfMountTargets": number,
   "OwnerId": "string",
   "PerformanceMode": "string",
   "ProvisionedThroughputInMibps": number,
   "SizeInBytes": {
```

```
      "Timestamp": number,
      "Value": number,
      "ValueInArchive": number,
      "ValueInIA": number,
      "ValueInStandard": number
   },
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ],
   "ThroughputMode": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 202 response.

The following data is returned in JSON format by the service.

AvailabilityZoneId

The unique and consistent identifier of the Availability Zone in which the file system is located, and is valid only for One Zone file systems. For example, `use1-az1` is an Availability Zone ID for the us-east-1 AWS Region, and it has the same location in every AWS account.

Type: String

AvailabilityZoneName

Describes the AWS Availability Zone in which the file system is located, and is valid only for One Zone file systems. For more information, see Using EFS storage classes in the *Amazon EFS User Guide*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

CreationTime

The time that the file system was created, in seconds (since 1970-01-01T00:00:00Z).

Type: Timestamp

CreationToken

The opaque string specified in the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Encrypted

A Boolean value that, if true, indicates that the file system is encrypted.

Type: Boolean

FileSystemArn

The Amazon Resource Name (ARN) for the EFS file system, in the format `arn:aws:elasticfilesystem:`*`region`*`:`*`account-id`*`:file-system/`*`file-system-id`* . Example with sample data: `arn:aws:elasticfilesystem:us-west-2:1111333322228888:file-system/fs-01234567`

Type: String

FileSystemId

The ID of the file system, assigned by Amazon EFS.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

FileSystemProtection

Describes the protection on the file system.

Type: FileSystemProtectionDescription object

KmsKeyId

The ID of an AWS KMS key used to protect the encrypted file system.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `^([0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|mrk-[0-9a-f]{32}|alias/[a-zA-Z0-9/_-]+|(arn:aws[-a-z]*:kms:[a-z0-9-]+:\d{12}:((key/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})|(key/mrk-[0-9a-f]{32})|(alias/[a-zA-Z0-9/_-]+))))$`

LifeCycleState

The lifecycle phase of the file system.

Type: String

Valid Values: `creating | available | updating | deleting | deleted | error`

Name

You can add tags to a file system, including a `Name` tag. For more information, see CreateFileSystem. If the file system has a `Name` tag, Amazon EFS returns the value in this field.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

NumberOfMountTargets

The current number of mount targets that the file system has. For more information, see CreateMountTarget.

Type: Integer

Valid Range: Minimum value of 0.

OwnerId

The AWS account that created the file system.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

PerformanceMode

The performance mode of the file system.

Type: String

Valid Values: `generalPurpose | maxIO`

ProvisionedThroughputInMibps

The amount of provisioned throughput, measured in MiBps, for the file system. Valid for file systems using `ThroughputMode` set to `provisioned`.

Type: Double

Valid Range: Minimum value of 1.0.

SizeInBytes

The latest known metered size (in bytes) of data stored in the file system, in its `Value` field, and the time at which that size was determined in its `Timestamp` field. The `Timestamp` value is the integer number of seconds since 1970-01-01T00:00:00Z. The `SizeInBytes` value doesn't represent the size of a consistent snapshot of the file system, but it is eventually consistent when there are no writes to the file system. That is, `SizeInBytes` represents actual size only if the file system is not modified for a period longer than a couple of hours. Otherwise, the value is not the exact size that the file system was at any point in time.

Type: FileSystemSize object

Tags

The tags associated with the file system, presented as an array of `Tag` objects.

Type: Array of Tag objects

ThroughputMode

Displays the file system's throughput mode. For more information, see Throughput modes in the *Amazon EFS User Guide*.

Type: String

Valid Values: `bursting | provisioned | elastic`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InsufficientThroughputCapacity

Returned if there's not enough capacity to provision additional throughput. This value might be returned when you try to create a file system in provisioned throughput mode, when you attempt to increase the provisioned throughput of an existing file system, or when you attempt to change an existing file system from Bursting Throughput to Provisioned Throughput mode. Try again later.

HTTP Status Code: 503

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ThroughputLimitExceeded

Returned if the throughput mode or amount of provisioned throughput can't be changed because the throughput limit of 1024 MiB/s has been reached.

HTTP Status Code: 400

TooManyRequests

Returned if you don’t wait at least 24 hours before either changing the throughput mode, or decreasing the Provisioned Throughput value.

HTTP Status Code: 429

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateFileSystemProtection

Updates protection on the file system.

This operation requires permissions for the `elasticfilesystem:UpdateFileSystemProtection` action.

Request Syntax

```
PUT /2015-02-01/file-systems/FileSystemId/protection HTTP/1.1
Content-type: application/json

{
   "ReplicationOverwriteProtection": "string"
}
```

URI Request Parameters

The request uses the following URI parameters.

FileSystemId

The ID of the file system to update.

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Request Body

The request accepts the following data in JSON format.

ReplicationOverwriteProtection

The status of the file system's replication overwrite protection.

- `ENABLED` – The file system cannot be used as the destination file system in a replication configuration. The file system is writeable. Replication overwrite protection is `ENABLED` by default.

- DISABLED – The file system can be used as the destination file system in a replication configuration. The file system is read-only and can only be modified by EFS replication.
- REPLICATING – The file system is being used as the destination file system in a replication configuration. The file system is read-only and is only modified only by EFS replication.

If the replication configuration is deleted, the file system's replication overwrite protection is re-enabled and the file system becomes writeable.

Type: String

Valid Values: ENABLED | DISABLED | REPLICATING

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "ReplicationOverwriteProtection": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

ReplicationOverwriteProtection

The status of the file system's replication overwrite protection.

- ENABLED – The file system cannot be used as the destination file system in a replication configuration. The file system is writeable. Replication overwrite protection is ENABLED by default.
- DISABLED – The file system can be used as the destination file system in a replication configuration. The file system is read-only and can only be modified by EFS replication.
- REPLICATING – The file system is being used as the destination file system in a replication configuration. The file system is read-only and is modified only by EFS replication.

If the replication configuration is deleted, the file system's replication overwrite protection is re-enabled, the file system becomes writeable.

Type: String

Valid Values: `ENABLED | DISABLED | REPLICATING`

Errors

BadRequest

Returned if the request is malformed or contains an error such as an invalid parameter value or a missing required parameter.

HTTP Status Code: 400

FileSystemNotFound

Returned if the specified `FileSystemId` value doesn't exist in the requester's AWS account.

HTTP Status Code: 404

IncorrectFileSystemLifeCycleState

Returned if the file system's lifecycle state is not "available".

HTTP Status Code: 409

InsufficientThroughputCapacity

Returned if there's not enough capacity to provision additional throughput. This value might be returned when you try to create a file system in provisioned throughput mode, when you attempt to increase the provisioned throughput of an existing file system, or when you attempt to change an existing file system from Bursting Throughput to Provisioned Throughput mode. Try again later.

HTTP Status Code: 503

InternalServerError

Returned if an error occurred on the server side.

HTTP Status Code: 500

ReplicationAlreadyExists

Returned if the file system is already included in a replication configuration.>

HTTP Status Code: 409

ThroughputLimitExceeded

Returned if the throughput mode or amount of provisioned throughput can't be changed because the throughput limit of 1024 MiB/s has been reached.

HTTP Status Code: 400

TooManyRequests

Returned if you don’t wait at least 24 hours before either changing the throughput mode, or decreasing the Provisioned Throughput value.

HTTP Status Code: 429

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for Kotlin
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Data Types

The following data types are supported:

- AccessPointDescription
- BackupPolicy
- CreationInfo
- Destination
- DestinationToCreate
- FileSystemDescription
- FileSystemProtectionDescription
- FileSystemSize
- LifecyclePolicy
- MountTargetDescription
- PosixUser
- ReplicationConfigurationDescription
- ResourceIdPreference
- RootDirectory
- Tag

AccessPointDescription

Provides a description of an EFS file system access point.

Contents

AccessPointArn

The unique Amazon Resource Name (ARN) associated with the access point.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:access-point/fsap-[0-9a-f]{8,40}$`

Required: No

AccessPointId

The ID of the access point, assigned by Amazon EFS.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:access-point/fsap-[0-9a-f]{8,40}|fsap-[0-9a-f]{8,40})$`

Required: No

ClientToken

The opaque string specified in the request to ensure idempotent creation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: No

FileSystemId

The ID of the EFS file system that the access point applies to.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: No

LifeCycleState

Identifies the lifecycle phase of the access point.

Type: String

Valid Values: `creating | available | updating | deleting | deleted | error`

Required: No

Name

The name of the access point. This is the value of the `Name` tag.

Type: String

Required: No

OwnerId

Identifies the AWS account that owns the access point resource.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

Required: No

PosixUser

The full POSIX identity, including the user ID, group ID, and secondary group IDs on the access point that is used for all file operations by NFS clients using the access point.

Type: PosixUser object

Required: No

RootDirectory

The directory on the EFS file system that the access point exposes as the root directory to NFS clients using the access point.

Type: RootDirectory object

Required: No

Tags

The tags associated with the access point, presented as an array of Tag objects.

Type: Array of Tag objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

BackupPolicy

The backup policy for the file system used to create automatic daily backups. If status has a value of `ENABLED`, the file system is being automatically backed up. For more information, see [Automatic backups](#).

Contents

Status

Describes the status of the file system's backup policy.

- **`ENABLED`** – EFS is automatically backing up the file system.
- **`ENABLING`** – EFS is turning on automatic backups for the file system.
- **`DISABLED`** – Automatic back ups are turned off for the file system.
- **`DISABLING`** – EFS is turning off automatic backups for the file system.

Type: String

Valid Values: `ENABLED | ENABLING | DISABLED | DISABLING`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CreationInfo

Required if the `RootDirectory` > `Path` specified does not exist. Specifies the POSIX IDs and permissions to apply to the access point's `RootDirectory` > `Path`. If the access point root directory does not exist, EFS creates it with these settings when a client connects to the access point. When specifying `CreationInfo`, you must include values for all properties.

Amazon EFS creates a root directory only if you have provided the CreationInfo: OwnUid, OwnGID, and permissions for the directory. If you do not provide this information, Amazon EFS does not create the root directory. If the root directory does not exist, attempts to mount using the access point will fail.

> **Important**
>
> If you do not provide `CreationInfo` and the specified `RootDirectory` does not exist, attempts to mount the file system using the access point will fail.

Contents

OwnerGid

Specifies the POSIX group ID to apply to the `RootDirectory`. Accepts values from 0 to 2^32 (4294967295).

Type: Long

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: Yes

OwnerUid

Specifies the POSIX user ID to apply to the `RootDirectory`. Accepts values from 0 to 2^32 (4294967295).

Type: Long

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: Yes

Permissions

Specifies the POSIX permissions to apply to the `RootDirectory`, in the format of an octal number representing the file's mode bits.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 4.

Pattern: `^[0-7]{3,4}$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Destination

Describes the destination file system in the replication configuration.

Contents

FileSystemId

The ID of the destination Amazon EFS file system.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

Region

The AWS Region in which the destination file system is located.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-z]{2}-((iso[a-z]{0,1}-)|(gov-)){0,1}[a-z]+-{0,1}[0-9]{0,1}$`

Required: Yes

Status

Describes the status of the replication configuration. For more information about replication status, see Viewing replication details in the *Amazon EFS User Guide*.

Type: String

Valid Values: `ENABLED | ENABLING | DELETING | ERROR | PAUSED | PAUSING`

Required: Yes

LastReplicatedTimestamp

The time when the most recent sync was successfully completed on the destination file system. Any changes to data on the source file system that occurred before this time have been successfully replicated to the destination file system. Any changes that occurred after this time might not be fully replicated.

Type: Timestamp

Required: No

OwnerId

ID of the AWS account in which the destination file system resides.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

Required: No

RoleArn

Amazon Resource Name (ARN) of the IAM role in the source account that allows Amazon EFS to perform replication on its behalf. This is optional for same-account replication and required for cross-account replication.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `arn:(aws[a-zA-Z-]*)?:iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: No

StatusMessage

Message that provides details about the `PAUSED` or `ERRROR` state of the replication destination configuration. For more information about replication status messages, see Viewing replication details in the *Amazon EFS User Guide*.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

DestinationToCreate

Describes the new or existing destination file system for the replication configuration.

- If you want to replicate to a new file system, do not specify the File System ID for the destination file system. Amazon EFS creates a new, empty file system. For One Zone storage, specify the Availability Zone to create the file system in. To use an AWS Key Management Service key other than the default KMS key, then specify it. For more information, see Configuring replication to new Amazon EFS file system in the *Amazon EFS User Guide*.

> **Note**
>
> After the file system is created, you cannot change the KMS key or the performance mode.

- If you want to replicate to an existing file system that's in the same account as the source file system, then you need to provide the ID or Amazon Resource Name (ARN) of the file system to which to replicate. The file system's replication overwrite protection must be disabled. For more information, see Replicating to an existing file system in the *Amazon EFS User Guide*.
- If you are replicating the file system to a file system that's in a different account than the source file system (cross-account replication), you need to provide the ARN for the file system and the IAM role that allows Amazon EFS to perform replication on the destination account. The file system's replication overwrite protection must be disabled. For more information, see Replicating across AWS accounts in the *Amazon EFS User Guide*.

Contents

AvailabilityZoneName

To create a file system that uses One Zone storage, specify the name of the Availability Zone in which to create the destination file system.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: No

FileSystemId

The ID or ARN of the file system to use for the destination. For cross-account replication, this must be an ARN. The file system's replication overwrite replication must be disabled. If no ID or ARN is specified, then a new file system is created.

> **Note**
>
> When you initially configure replication to an existing file system, Amazon EFS writes data to or removes existing data from the destination file system to match data in the source file system. If you don't want to change data in the destination file system, then you should replicate to a new file system instead. For more information, see https://docs.aws.amazon.com/efs/latest/ug/create-replication.html.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: No

KmsKeyId

Specify the AWS Key Management Service (AWS KMS) key that you want to use to encrypt the destination file system. If you do not specify a KMS key, Amazon EFS uses your default KMS key for Amazon EFS, `/aws/elasticfilesystem`. This ID can be in one of the following formats:

- Key ID - The unique identifier of the key, for example `1234abcd-12ab-34cd-56ef-1234567890ab`.
- ARN - The ARN for the key, for example `arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`.
- Key alias - A previously created display name for a key, for example `alias/projectKey1`.
- Key alias ARN - The ARN for a key alias, for example `arn:aws:kms:us-west-2:444455556666:alias/projectKey1`.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `^([0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|mrk-[0-9a-f]{32}|alias/[a-zA-Z0-9/_-]+|(arn:aws[-a-z]*:kms:[a-z0-9-]+:\d{12}:((key/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})|(key/mrk-[0-9a-f]{32})|(alias/[a-zA-Z0-9/_-]+))))$`

Required: No

Region

To create a file system that uses Regional storage, specify the AWS Region in which to create the destination file system. The Region must be enabled for the AWS account that owns the source file system. For more information, see Managing AWS Regions in the *AWS General Reference Reference Guide*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-z]{2}-((iso[a-z]{0,1}-)|(gov-)){0,1}[a-z]+-{0,1}[0-9]{0,1}$`

Required: No

RoleArn

Amazon Resource Name (ARN) of the IAM role in the source account that allows Amazon EFS to perform replication on its behalf. This is optional for same-account replication and required for cross-account replication.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `arn:(aws[a-zA-Z-]*)?:iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

FileSystemDescription

A description of the file system.

Contents

CreationTime

The time that the file system was created, in seconds (since 1970-01-01T00:00:00Z).

Type: Timestamp

Required: Yes

CreationToken

The opaque string specified in the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: Yes

FileSystemId

The ID of the file system, assigned by Amazon EFS.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

LifeCycleState

The lifecycle phase of the file system.

Type: String

Valid Values: `creating | available | updating | deleting | deleted | error`

Required: Yes

NumberOfMountTargets

The current number of mount targets that the file system has. For more information, see CreateMountTarget.

Type: Integer

Valid Range: Minimum value of 0.

Required: Yes

OwnerId

The AWS account that created the file system.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

Required: Yes

PerformanceMode

The performance mode of the file system.

Type: String

Valid Values: `generalPurpose | maxIO`

Required: Yes

SizeInBytes

The latest known metered size (in bytes) of data stored in the file system, in its `Value` field, and the time at which that size was determined in its `Timestamp` field. The `Timestamp` value is the integer number of seconds since 1970-01-01T00:00:00Z. The `SizeInBytes` value doesn't represent the size of a consistent snapshot of the file system, but it is eventually consistent when there are no writes to the file system. That is, `SizeInBytes` represents actual size only if the file system is not modified for a period longer than a couple of hours. Otherwise, the value is not the exact size that the file system was at any point in time.

Type: FileSystemSize object

Required: Yes

Tags

The tags associated with the file system, presented as an array of `Tag` objects.

Type: Array of Tag objects

Required: Yes

AvailabilityZoneId

The unique and consistent identifier of the Availability Zone in which the file system is located, and is valid only for One Zone file systems. For example, `use1-az1` is an Availability Zone ID for the us-east-1 AWS Region, and it has the same location in every AWS account.

Type: String

Required: No

AvailabilityZoneName

Describes the AWS Availability Zone in which the file system is located, and is valid only for One Zone file systems. For more information, see Using EFS storage classes in the *Amazon EFS User Guide*.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: No

Encrypted

A Boolean value that, if true, indicates that the file system is encrypted.

Type: Boolean

Required: No

FileSystemArn

The Amazon Resource Name (ARN) for the EFS file system, in the format `arn:aws:elasticfilesystem:`*`region`*`:`*`account-id`*`:file-system/`*`file-system-id`* . Example with sample data: `arn:aws:elasticfilesystem:us-west-2:1111333322228888:file-system/fs-01234567`

Type: String

Required: No

FileSystemProtection

Describes the protection on the file system.

Type: FileSystemProtectionDescription object

Required: No

KmsKeyId

The ID of an AWS KMS key used to protect the encrypted file system.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `^([0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|mrk-[0-9a-f]{32}|alias/[a-zA-Z0-9/_-]+|(arn:aws[-a-z]*:kms:[a-z0-9-]+:\d{12}:((key/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})|(key/mrk-[0-9a-f]{32})|(alias/[a-zA-Z0-9/_-]+))))$`

Required: No

Name

You can add tags to a file system, including a `Name` tag. For more information, see CreateFileSystem. If the file system has a `Name` tag, Amazon EFS returns the value in this field.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Required: No

ProvisionedThroughputInMibps

The amount of provisioned throughput, measured in MiBps, for the file system. Valid for file systems using `ThroughputMode` set to `provisioned`.

Type: Double

Valid Range: Minimum value of 1.0.

Required: No

ThroughputMode

Displays the file system's throughput mode. For more information, see [Throughput modes](#) in the *Amazon EFS User Guide*.

Type: String

Valid Values: `bursting | provisioned | elastic`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

FileSystemProtectionDescription

Describes the protection on a file system.

Contents

ReplicationOverwriteProtection

The status of the file system's replication overwrite protection.

- `ENABLED` – The file system cannot be used as the destination file system in a replication configuration. The file system is writeable. Replication overwrite protection is `ENABLED` by default.
- `DISABLED` – The file system can be used as the destination file system in a replication configuration. The file system is read-only and can only be modified by EFS replication.
- `REPLICATING` – The file system is being used as the destination file system in a replication configuration. The file system is read-only and is modified only by EFS replication.

If the replication configuration is deleted, the file system's replication overwrite protection is re-enabled, the file system becomes writeable.

Type: String

Valid Values: `ENABLED | DISABLED | REPLICATING`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

FileSystemSize

The latest known metered size (in bytes) of data stored in the file system, in its `Value` field, and the time at which that size was determined in its `Timestamp` field. The value doesn't represent the size of a consistent snapshot of the file system, but it is eventually consistent when there are no writes to the file system. That is, the value represents the actual size only if the file system is not modified for a period longer than a couple of hours. Otherwise, the value is not necessarily the exact size the file system was at any instant in time.

Contents

Value

The latest known metered size (in bytes) of data stored in the file system.

Type: Long

Valid Range: Minimum value of 0.

Required: Yes

Timestamp

The time at which the size of data, returned in the `Value` field, was determined. The value is the integer number of seconds since 1970-01-01T00:00:00Z.

Type: Timestamp

Required: No

ValueInArchive

The latest known metered size (in bytes) of data stored in the Archive storage class.

Type: Long

Valid Range: Minimum value of 0.

Required: No

ValueInIA

The latest known metered size (in bytes) of data stored in the Infrequent Access storage class.

Type: Long

Valid Range: Minimum value of 0.

Required: No

ValueInStandard

The latest known metered size (in bytes) of data stored in the Standard storage class.

Type: Long

Valid Range: Minimum value of 0.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

LifecyclePolicy

Describes a policy used by lifecycle management that specifies when to transition files into and out of storage classes. For more information, see Managing file system storage.

> **Note**
>
> When using the `put-lifecycle-configuration` CLI command or the `PutLifecycleConfiguration` API action, Amazon EFS requires that each `LifecyclePolicy` object have only a single transition. This means that in a request body, `LifecyclePolicies` must be structured as an array of `LifecyclePolicy` objects, one object for each transition. For more information, see the request examples in PutLifecycleConfiguration.

Contents

TransitionToArchive

The number of days after files were last accessed in primary storage (the Standard storage class) at which to move them to Archive storage. Metadata operations such as listing the contents of a directory don't count as file access events.

Type: String

Valid Values: `AFTER_1_DAY | AFTER_7_DAYS | AFTER_14_DAYS | AFTER_30_DAYS | AFTER_60_DAYS | AFTER_90_DAYS | AFTER_180_DAYS | AFTER_270_DAYS | AFTER_365_DAYS`

Required: No

TransitionToIA

The number of days after files were last accessed in primary storage (the Standard storage class) at which to move them to Infrequent Access (IA) storage. Metadata operations such as listing the contents of a directory don't count as file access events.

Type: String

Valid Values: AFTER_7_DAYS | AFTER_14_DAYS | AFTER_30_DAYS | AFTER_60_DAYS | AFTER_90_DAYS | AFTER_1_DAY | AFTER_180_DAYS | AFTER_270_DAYS | AFTER_365_DAYS

Required: No

TransitionToPrimaryStorageClass

Whether to move files back to primary (Standard) storage after they are accessed in IA or Archive storage. Metadata operations such as listing the contents of a directory don't count as file access events.

Type: String

Valid Values: AFTER_1_ACCESS

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MountTargetDescription

Provides a description of a mount target.

Contents

FileSystemId

The ID of the file system for which the mount target is intended.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

LifeCycleState

Lifecycle state of the mount target.

Type: String

Valid Values: `creating | available | updating | deleting | deleted | error`

Required: Yes

MountTargetId

System-assigned mount target ID.

Type: String

Length Constraints: Minimum length of 13. Maximum length of 45.

Pattern: `^fsmt-[0-9a-f]{8,40}$`

Required: Yes

SubnetId

The ID of the mount target's subnet.

Type: String

Length Constraints: Minimum length of 15. Maximum length of 47.

Pattern: `^subnet-[0-9a-f]{8,40}$`

Required: Yes

AvailabilityZoneId

The unique and consistent identifier of the Availability Zone that the mount target resides in. For example, `use1-az1` is an AZ ID for the us-east-1 Region and it has the same location in every AWS account.

Type: String

Required: No

AvailabilityZoneName

The name of the Availability Zone in which the mount target is located. Availability Zones are independently mapped to names for each AWS account. For example, the Availability Zone `us-east-1a` for your AWS account might not be the same location as `us-east-1a` for another AWS account.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `.+`

Required: No

IpAddress

The IPv4 address for the mount target.

Type: String

Length Constraints: Minimum length of 7. Maximum length of 15.

Pattern: `^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$`

Required: No

Ipv6Address

The IPv6 address for the mount target.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 39.

Required: No

NetworkInterfaceId

The ID of the network interface that Amazon EFS created when it created the mount target.

Type: String

Required: No

OwnerId

AWS account ID that owns the resource.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

Required: No

VpcId

The virtual private cloud (VPC) ID that the mount target is configured in.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++

- AWS SDK for Java V2
- AWS SDK for Ruby V3

PosixUser

The full POSIX identity, including the user ID, group ID, and any secondary group IDs, on the access point that is used for all file system operations performed by NFS clients using the access point.

Contents

Gid

The POSIX group ID used for all file system operations using this access point.

Type: Long

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: Yes

Uid

The POSIX user ID used for all file system operations using this access point.

Type: Long

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: Yes

SecondaryGids

Secondary POSIX group IDs used for all file system operations using this access point.

Type: Array of longs

Array Members: Minimum number of 0 items. Maximum number of 16 items.

Valid Range: Minimum value of 0. Maximum value of 4294967295.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ReplicationConfigurationDescription

Describes the replication configuration for a specific file system.

Contents

CreationTime

Describes when the replication configuration was created.

Type: Timestamp

Required: Yes

Destinations

An array of destination objects. Only one destination object is supported.

Type: Array of Destination objects

Required: Yes

OriginalSourceFileSystemArn

The Amazon Resource Name (ARN) of the original source EFS file system in the replication configuration.

Type: String

Required: Yes

SourceFileSystemArn

The Amazon Resource Name (ARN) of the current source file system in the replication configuration.

Type: String

Required: Yes

SourceFileSystemId

The ID of the source Amazon EFS file system that is being replicated.

Type: String

Length Constraints: Maximum length of 128.

Pattern: `^(arn:aws[-a-z]*:elasticfilesystem:[0-9a-z-:]+:file-system/fs-[0-9a-f]{8,40}|fs-[0-9a-f]{8,40})$`

Required: Yes

SourceFileSystemRegion

The AWS Region in which the source EFS file system is located.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-z]{2}-((iso[a-z]{0,1}-)|(gov-)){0,1}[a-z]+-{0,1}[0-9]{0,1}$`

Required: Yes

SourceFileSystemOwnerId

ID of the AWS account in which the source file system resides.

Type: String

Length Constraints: Maximum length of 14.

Pattern: `^(\d{12})|(\d{4}-\d{4}-\d{4})$`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

ResourceIdPreference

Describes the resource type and its ID preference for the user's AWS account, in the current AWS Region.

Contents

ResourceIdType

Identifies the EFS resource ID preference, either `LONG_ID` (17 characters) or `SHORT_ID` (8 characters).

Type: String

Valid Values: `LONG_ID | SHORT_ID`

Required: No

Resources

Identifies the Amazon EFS resources to which the ID preference setting applies, `FILE_SYSTEM` and `MOUNT_TARGET`.

Type: Array of strings

Valid Values: `FILE_SYSTEM | MOUNT_TARGET`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

RootDirectory

Specifies the directory on the Amazon EFS file system that the access point provides access to. The access point exposes the specified file system path as the root directory of your file system to applications using the access point. NFS clients using the access point can only access data in the access point's `RootDirectory` and its subdirectories.

Contents

CreationInfo

(Optional) Specifies the POSIX IDs and permissions to apply to the access point's `RootDirectory`. If the `RootDirectory` > `Path` specified does not exist, EFS creates the root directory using the `CreationInfo` settings when a client connects to an access point. When specifying the `CreationInfo`, you must provide values for all properties.

> **Important**
>
> If you do not provide `CreationInfo` and the specified `RootDirectory` > `Path` does not exist, attempts to mount the file system using the access point will fail.

Type: CreationInfo object

Required: No

Path

Specifies the path on the EFS file system to expose as the root directory to NFS clients using the access point to access the EFS file system. A path can have up to four subdirectories. If the specified path does not exist, you are required to provide the `CreationInfo`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^(\/|(\/(?!\.)+[^$#<>;`|&?{}^*/\n]+){1,4})$`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Tag

A tag is a key-value pair. Allowed characters are letters, white space, and numbers that can be represented in UTF-8, and the following characters: `+ - = . _ : /`.

Contents

Key

The tag key (String). The key can't start with `aws:`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^(?![aA]{1}[wW]{1}[sS]{1}:)([\p{L}\p{Z}\p{N}_.:/=+\-@]+)$`

Required: Yes

Value

The value of the tag key.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Document history

- **API version**: 2015-02-01
- **Latest documentation update**: June 9, 2025

The following table describes important changes to the *Amazon Elastic File System User Guide* after July 2018. For notifications about documentation updates, you can subscribe to the RSS feed.

Change	Description	Date
Support for IPv6	Internet Protocol Version 6 (IPv6) is supported on Amazon EFS Service APIs and mount targets. For more information about the Amazon EFS API, see Amazon EFS API. For more informati on about mount targets, see Managing mount targets.	June 9, 2025
Elastic throughput increase in certain AWS Regions	The maximum read throughput for file systems using Elastic throughput in the following Regions has increased from 20 GiBps to 60 GiBps: Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Sydney), and Europe (London). For more informati on, see Amazon EFS quotas that you can increase.	May 2, 2025
Elastic throughput increase in certain AWS Regions	The maximum read throughput for file systems using Elastic throughpu t in specific Regions is 60	April 10, 2025

	GiBps. In all other Regions, the read throughput limit for file systems using Elastic throughput has been increased from 10 to 20 GiBps. For more information, see Amazon EFS quotas that you can increase.	
Increased access point quota	The maximum number of access points a single file system can have has increased from 1,000 to 10,000. You can also request an increase for this limit. For more information, see Resource quotas that you can increase.	February 10, 2025
Improved backup and restore rates	The speed of performing backup and restores has improved. For more informati on, see Backup performance.	January 8, 2025
Increased IOPS quota by request	You can now request increased IOPS for EFS file systems using Elastic throughput mode. For more information, see Amazon EFS quotas that you can increase.	November 26, 2024
Support cross-account replication	Amazon EFS supports cross-account replication. For more information, see Replicating EFS file systems across AWS accounts.	November 19, 2024

Updated existing AWS managed policy	`ReplicationRead` , `ReplicationWrite` , `iam:PassRole` are added to the `AmazonElasticFileSystemFullAccess` policy. For more information, see AmazonElasticFileSystemFullAccess.	November 7, 2024
Updated existing AWS managed policy	`ReplicationRead` and `ReplicationWrite` added to `AmazonElasticFileSystemServiceRolePolicy` .For more information, see AmazonElasticFileSystemServiceRolePolicy.	November 7, 2024
Updated existing AWS managed policy	Permission `ReplicationRead` added to existing `AmazonElasticFileSystemReadOnlyAccess` policy. For more information, see AmazonElasticFileSystemReadOnlyAccess.	November 7, 2024
Elastic throughput limit increased	Elastic throughput limit has increased to 60 gibibytes per second (GiBps) for specific AWS Regions and to 10 GiBps for all other Regions. For more information, see Total default Elastic throughput for all connected clients in each AWS Region.	October 14, 2024

Updated existing AWS managed policy	The optional `Sid` (statement ID) element is now included in the `AmazonElasticFileSystemReadOnlyAccess` policy statement. The `Sid` value is `ElasticFileSystemReadOnlyAccess`. For more information about the `Sid` policy element, see IAM JSON policy elements: Sid.	August 7, 2024
Elastic throughput limit increased	Elastic throughput limit has increased for specific AWS Regions. For more information, see Total default Elastic throughput for all connected clients in each AWS Region.	July 31, 2024
Increased quota for mount targets	The maximum number of mount targets for each virtual private cloud (VPC)increased from 400 to 1,400. For more information, see Amazon EFS resource quotas that you cannot change.	May 15, 2024

Increased combined throughput limit for Elastic file systems	Maximum combined read and write throughput is 1,500 MiBps for file systems using Elastic throughput and mounted using version 2.0 or later of the Amazon EFS client (amazon-efs-utils version) or the Amazon EFS CSI Driver (aws-efs-csi-driver). For more information, see the Performance summary table in Amazon EFS performance.	April 30, 2024
Elastic throughput limit increased	Elastic throughput limit has increased for specific AWS Regions. For more informati on, see Total default Elastic throughput for all connected clients in each AWS Region.	March 13, 2024
Increased IOPS	File systems that use Elastic throughput can drive a maximum of 90,000 read for infrequently accessed data. For more information, see Performance summary.	January 22, 2024
Updated existing AWS managed policy	Permission `elasticfilesystem:UpdateFileSystemProtection` added to existing AmazonElasticFileSystemFullAccess policy to allow principals to update protection on a file system. For more information, see Amazon EFS updates to AWS managed policies.	November 27, 2023

Replicate to existing file system	File systems can now be replicated to existing file systems, making it easier to synchronize changes between file systems for failback purposes. For more information, see Destination file systems.	November 27, 2023
File system protection added	Replication overwrite protection has been added to file systems and is enabled by default. The protection prevents file systems from being used as the destination in a replication configuration. For more information, see File system protection.	November 27, 2023
New storage class, file system types, and lifecycle policy	Amazon EFS now offers the EFS Archive storage class, file system types, and the Transition into Archive lifecycle policy. For more information, see File system types and storage classes.	November 26, 2023
Increased IOPS	Elastic throughput file systems now support a maximum of 65,000 read and 50,000 write operation s IOPS for infrequently accessed data, and now support 250,000 read IOPS for frequently accessed data. For more information, see Performance summary.	November 26, 2023

Delete replication configura tion from source file system	Replication configurations can now be deleted from the source file system. For more information, see Deleting a replication configuration.	September 19, 2023
Additional AWS Region support added	Amazon EFS is now available to all users in the Israel (Tel Aviv) Region.	August 7, 2023
Performance increase for General Purpose mode file systems	Amazon EFS General Purpose mode file systems now support up to 55,000 read operations per second and 25,000 write operations. For more information, see Quotas for Amazon EFS File Systems.	August 3, 2023
Provisioned throughput limit increased	Provisioned throughput limit has increased for specific AWS Regions. For more information, see Total default Provisioned throughput for all connected clients in each AWS Region.	June 21, 2023
Expanded Region support for EFS replication	EFS replication is now available in all AWS Regions in which EFS is available. For more information, see Amazon EFS replication.	April 28, 2023

Elastic throughput limit increase	Elastic throughput limit has increased for specific AWS Regions. For more informati on, see the table Total default Elastic throughput for all connected clients in each AWS Region.	April 17, 2023
Elastic replaces Bursting as the default throughput mode	The default (and recommend ed) throughput mode for file systems is now Elastic instead of Bursting. For more information, see Throughput modes.	April 13, 2023
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Melbourne) Region.	April 12, 2023
Support added for macOS Ventura	Amazon EFS can now be installed on EC2 Mac instances running on macOS Ventura. For more informati on, see Supported distribut ions.	April 10, 2023
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Hyderabad) Region.	February 16, 2023
Additional AWS Region support added	Amazon EFS is now available to all users in the Europe (Spain) AWS Region.	January 19, 2023

The access point limit for file systems has increased	The maximum number of access points a single file system can have has increased from 120 to 1,000. For more information, see Resource quotas.	January 17, 2023
Additional AWS Region support added	Amazon EFS is now available to all users in the Europe (Zurich) AWS Region.	December 15, 2022
Support added for one day lifecycle policies	You can now select one day for the Transition into IA lifecycle policy. For more information, see Using Lifecycle policies.	November 27, 2022
Reduced read and write latencies	Latencies for file data reads and writes have reduced for both One Zone storage and Standard storage file systems. For more information, see Performance summary.	November 27, 2022
Additional throughput mode added	Elastic throughput mode is added as a throughput option for Amazon EFS file systems. For more information, see Elastic throughput.	November 27, 2022
Additional AWS Region support added	Amazon EFS is now available to all users in the Middle East (UAE) Region.	October 17, 2022

Support added for EFS Replication	Amazon EFS has removed a previous limit where EFS replication does not support sockets and named pipes, or FIFOs.	September 15, 2022
The limit for the number of file locks per connection has increased	The number of file locks per connection has increased from 8192 to 65,536. For more information, see Quotas for NFS clients.	May 4, 2022
Limit for processes using file locks removed	Amazon EFS has removed a previous limit where a maximum of 256 processes on a single instance could use file locks at the same time. For more information, see Quotas for NFS clients.	May 4, 2022
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Jakarta) AWS Region.	January 27, 2022
Support added for EFS Replication	Use EFS Replication to replicate the data and metadata on an EFS file system to another EFS file system in the AWS Region of your choice. For more information, see Amazon EFS replication.	January 25, 2022

File system and mount target resources use 17-character resource ID format	New Amazon EFS file system and mount target resources are now assigned 17-charac ter IDs. For more information, see Working with Amazon EFS resources.	October 22, 2021
Support added for EFS Intelligent-Tiering	EFS Intelligent-Tiering uses EFS Lifecycle Management to monitor file access patterns and is designed to automatic ally transition files to and from your corresponding Infrequent Access (IA) storage classes. For more informati on, see EFS Intelligent-Tiering and Lifecycle Management.	September 2, 2021
Support added for testing 17-character resource ID format	Amazon EFS is transitioning from using 8-character IDs to 17-character IDs for file systems and mount targets on October 1, 2021. During this transition, you can opt in and start using 17-charac ters resource IDs on a per AWS Region basis. For more information, see Resource IDs.	May 5, 2021

Support added for mounting One Zone file systems from a different Availability Zone using Amazon EFS mount helper	You can now use the EFS mount helper to mount an Amazon EFS file system that uses One Zone storage classes to an EC2 instance that is in a different Availability Zone. You can use the new `az` option to specify the Availability Zone of the Amazon EFS file system. For more information, see Mounting file systems with One Zone storage classes.	April 6, 2021
Support added for EFS One Zone storage classes	Amazon EFS One Zone storage classes store data redundantly within a single Availability Zone in an AWS Region. The EFS One Zone and One Zone-Infrequent Access (One Zone-IA) storage classes are a cost-effective option for storing data that doesn't require the Multi-AZ resilience of the EFS Standard and Standard-IA storage classes. For more information, see Using EFS storage classes.	March 9, 2021
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Osaka) AWS Region.	March 3, 2021

Support added for Amazon EC2 macOS instances running macOS Big Sur	You can now mount your Amazon EFS file system from EC2 macOS instances that are running macOS Big Sur by using the EFS mount helper or by using the NFS mount command. For more information, see Mounting with the EFS mount helper or Mounting file systems without the EFS mount helper.	February 23, 2021
New Amazon EFS console is available in AWS GovCloud (US) Region	The new Amazon EFS console is now available in the AWS GovCloud (US) AWS Region.	February 10, 2021
Support added for new Amazon EFS CloudWatch metric `MeteredIOBytes`	You can use `MeteredIOBytes` to measure the number of metered bytes for each file system operation , including data read, data write, and metadata operations. Read operation s are metered at one-third the rate of other operation s. For more information, see Amazon CloudWatch metrics for Amazon EFS.	January 28, 2021
Amazon EFS increases file system read throughput by 300%	Amazon EFS file systems now meter read requests at one-third the rate of other requests.	January 28, 2021

Support added for new Amazon EFS CloudWatch metric `StorageBytes`	You can use `StorageBytes` to measure and monitor the size of the file system in bytes, including the amount of data stored in the Standard and Infrequent Access storage classes. For more information, see Amazon CloudWatch metrics for Amazon EFS.	January 11, 2021
Use AWS Transfer Family to access Amazon EFS file systems	You can use AWS Transfer Family to transfer files into and out of your Amazon EFS file systems. For more information, see Using AWS Transfer Family to access files in your EFS file system.	January 6, 2021
Use AWS Systems Manager to manage Amazon EFS client (`amazon-efs-utils`)	You can use AWS Systems Manager to automatically install or update the Amazon EFS clients (`amazon-efs-utils`) on your EC2 instances. For more information, see Using AWS Systems Manager to automatically install or update Amazon EFS clients.	September 29, 2020

Enforce the creation of encrypted EFS file systems	You can use the `elasticfilesystem:Encrypted` AWS Identity and Access Management (IAM) condition key to enforce that users create Amazon EFS file systems that are encrypted at rest. For more information, see Enforcing the Creation of Amazon EFS File Systems Encrypted at Rest.	September 16, 2020
Amazon EFS per-client throughput increased 100%	EFS now supports up to 500 MB/s of per-client throughput, a 100% increase from the previous limit of 250 MB/s. For more information, see Quotas for Amazon EFS file systems.	July 23, 2020
Support added for automatic daily backups of Amazon EFS file systems	Automatic daily backups are now enabled by default when creating a file system using the EFS console. For more information, see Using AWS Backup with Amazon EFS.	July 16, 2020
New Quick Create workflow simplifies creating Amazon EFS file systems	Using the Quick Create option in the EFS console, you can create an EFS file system using service recommended settings with a single button. For more information, see Create your EFS file system.	July 16, 2020

New Amazon EFS console is now available	The new EFS console makes it easier for you to use Amazon EFS and simplifies the management of your EFS file systems.	July 16, 2020
Amazon EFS increases file system minimum throughput	Amazon EFS file systems using Bursting throughput now have a minimum throughput of 1 MiB/s. For more information, see Throughput modes.	June 30, 2020
Performance of General Purpose mode file systems increased	Amazon EFS General Purpose mode file systems now support up to 35,000 read operations per second, a 400% increase from the previous limit of 7,000. For more information, see Quotas for Amazon EFS File Systems.	April 1, 2020
Additional AWS Region support added	Amazon EFS is now available to all users in the Beijing and Ningxia AWS Regions.	January 22, 2020
Support added for IAM authorization for NFS clients	You can now use AWS Identity and Access Management (IAM) to manage NFS access to an Amazon EFS file system. For more information, see Using AWS IAM to Control NFS Access to Amazon EFS.	January 13, 2020

Support added for EFS Access Points	Amazon EFS access points are application-specific entry points into an Amazon EFS file system that make it easy to manage application access to shared datasets. For more information, see Working with Amazon EFS Access Points.	January 13, 2020
Support added for AWS Backup partial restore.	You can now restore specific files and directories using a partial restore, in addition to restoring a complete recovery point. For more information, see Using AWS Backup with Amazon EFS.	January 13, 2020
Support added for IAM service-linked roles	Amazon EFS now uses a service-linked role based on IAM, making it easier to set up EFS by automatically adding the necessary permissions. For more information, see Using Service-Linked Roles for Amazon EFS.	December 10, 2019
Additional AWS Region support added	Amazon EFS is now available to all users in the Europe (Stockholm) AWS Region.	November 20, 2019
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Hong Kong) AWS Region.	November 20, 2019

Additional AWS Region support added	Amazon EFS is now available to all users in the South America (São Paulo) AWS Region.	November 20, 2019
Additional AWS Region support added	Amazon EFS is now available to all users in the Middle East (Bahrain) AWS Region.	November 20, 2019
New 7 day Lifecycle management policy added	Lifecycle management now has an additional policy to move data to the cost-effective Infrequent Access storage class after 7 days. For more information, see EFS Lifecycle Management.	November 6, 2019
Support added for Interface VPC Endpoints	You can establish a private connection between your virtual private cloud and Amazon EFS to call the EFS API. For more informati on, see Working with VPC Endpoints.	October 22, 2019
Mount an EFS file system when launching a new EC2 instance.	You can now configure new Amazon EC2 instances to mount your EFS file systems at launch in the EC2 Launch Instance Wizard. For more information, see Step 2. Create Your EC2 Resources and Launch Your EC2 Instance.	October 17, 2019

Support for Service Quotas added	You can now view all Amazon EFS limits in the Service Quotas console. For more information, see Amazon EFS Limits.	September 10, 2019
New lifecycle management policies added	When using Lifecycle Management, you can now choose from one of four lifecycle policies to define when files are transitioned into the cost-effective Infrequent Access storage class. For more information, see EFS Lifecycle Management.	July 9, 2019
EFS Lifecycle Management now available on all EFS file systems.	The EFS Lifecycle Management feature is now available on all EFS file systems. A previous restriction based on when a file system was created is now removed. For more information, see EFS Lifecycle Management.	July 9, 2019
Additional AWS Region support added	Amazon EFS is now available to all users in the Europe (Paris) AWS Region.	June 12, 2019
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Mumbai) AWS Region.	June 5, 2019
Additional AWS Region support added	Amazon EFS is now available to all users in the Canada (Central) AWS Region.	May 1, 2019

API Update: Tags are now part of the CreateFileSystem operation payload	You can now include tags when using the AWS API and CLI CreateFileSystem operation to create an Amazon EFS file system. For more information, see CreateFileSystem and Creating a File System Using the AWS CLI.	February 19, 2019
New features: EFS Infrequent Access storage class and EFS lifecycle management	Amazon EFS Infrequent Access is a cost-optimized storage class for infrequently accessed files. EFS lifecycle management automatically transitions files from Standard to Infrequent Access storage. For more information, see EFS Storage Classes.	February 13, 2019
Additional AWS Region support added	Amazon EFS is now available to all users in the Europe (London) AWS Region.	January 23, 2019
AWS Backup Service integration with Amazon EFS	Amazon EFS file systems can be backed up using AWS Backup, a fully managed, centralized, automated backup service for backing up data across AWS services in the cloud and on premises. For more information, see AWS Backup and Amazon EFS.	January 16, 2019

Transit Gateway connection support to on-premises storage systems added.	Amazon EFS file systems are now accessible using Transit Gateway connections to on-premises storage systems. For more information, see Mounting from Another Account or VPC and Walkthrough: Mount a File System from a Different VPC.	December 6, 2018
EFS File Sync is now part of the new AWS DataSync service.	AWS DataSync is a managed data transfer service that simplifies synchronizing large amounts of data between on-premises storage systems and AWS storage services. For more information, see Transfer Files from On-Premises File Systems to Amazon EFS Using AWS DataSync.	November 26, 2018
VPN and inter-Region VPC peering connection support added	Amazon EFS are now accessible over VPN connections and inter-Region VPC peering connections. For more information, see Transfer Files from On-Premises File Systems to Amazon EFS Using AWS DataSync.	October 23, 2018

VPN and inter-Region VPC peering connection support added	Amazon EFS file systems are now accessible over VPN connections and inter-Region VPC peering connections. For more information, see Mounting from Another Account or VPC and How Amazon EFS Works with Direct Connect and VPNs.	October 23, 2018
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Singapore) AWS Region.	July 13, 2018
Introducing Provisioned Throughput mode	You can now provision throughput for new or existing file systems with the new Provisioned Throughput mode. For more information, see Throughput modes.	July 12, 2018
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Tokyo) AWS Region.	July 11, 2018

The following table describes important changes to the *Amazon Elastic File System User Guide* before July 2018.

Change	Description	Date Changed
Additional AWS Region support added	Amazon EFS is now available to all users in the Asia Pacific (Seoul) AWS Region.	May 30, 2018
Added CloudWatch metric math support	Metric math enables you to query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics.	April 4, 2018

Change	Description	Date Changed
	For more information, see Using metric math with CloudWatch metrics.	
Added the `amazon-efs-utils` set of open-source tools, and added encryption in transit	The `amazon-efs-utils` tools are a set of open-source executable files that simplifies aspects of using Amazon EFS, like mounting. There's no additional cost to use `amazon-efs-utils`, and you can download these tools from GitHub. For more information, see Installing the Amazon EFS client. Also in this release, Amazon EFS now supports encryption in transit through Transport Layer Security (TLS) tunneling. For more information, see Encrypting data in Amazon EFS.	April 4, 2018
Updated file system limits per AWS Region	Amazon EFS has increased the limit on the number of file systems for all accounts in all AWS Regions. For more information, see Amazon EFS resource quotas that you cannot change.	March 15, 2018
Additional AWS Region support added	Amazon EFS is now available to all users in the US West (N. California) AWS Region.	March 14, 2018
Data encryption at rest	Amazon EFS now supports data encryption at rest. For more information, see Encrypting data in Amazon EFS.	August 14, 2017
Additional Region support added	Amazon EFS is now available to all users in the Europe (Frankfurt) Region.	July 20, 2017
File system names using Domain Name System (DNS)	Amazon EFS now supports DNS names for file systems. A file system's DNS name automatic ally resolves to a mount target's IP address in the Availability Zone for the connecting Amazon EC2 instance. For more information, see Mounting on Amazon EC2 with a DNS name.	December 20, 2016

Change	Description	Date Changed
Increased tag support for file systems	Amazon EFS now supports 50 tags per file system. For more information on tags in Amazon EFS, see Tagging EFS resources.	August 29, 2016
General availabil ity	Amazon EFS is now generally available to all users in the US East (N. Virginia), US West (Oregon), and Europe (Ireland) Regions.	June 28, 2016
File system limit increase	The number of Amazon EFS file systems that can be created per account for each AWS Region increased from 5 to 10.	August 21, 2015
Updated Getting Started exercise	The Getting Started exercise has been updated to simplify the getting started process.	August 17, 2015
New guide	This is the first release of the *Amazon Elastic File System User Guide*.	May 26, 2015