

---

# Amazon Elastic File System

## User Guide



## **Amazon Elastic File System: User Guide**

Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

## Table of Contents

What Is Amazon Elastic File System? .....	1
Are You a First-Time User of Amazon EFS? .....	2
How It Works .....	3
Overview .....	3
How Amazon EFS Works with Amazon EC2 .....	4
How Amazon EFS Works with AWS Direct Connect and AWS Managed VPN .....	4
How Amazon EFS Works with AWS Backup .....	5
Implementation Summary .....	6
Authentication and Access Control .....	7
Data Consistency in Amazon EFS .....	7
Storage Classes and Lifecycle Management .....	7
Setting Up .....	9
Sign up for AWS .....	9
Create an IAM User .....	9
Getting Started .....	11
Assumptions .....	11
Related Topics .....	11
Step 1: Create Your Amazon EFS File System .....	12
Step 2: Create Your EC2 Resources and Launch Your EC2 Instance .....	15
Step 3: Transfer Files Using DataSync .....	16
Before You Begin .....	16
Step 4: Clean Up Resources and Protect Your AWS Account .....	16
Creating Resources for Amazon EFS .....	18
Creating a File System .....	19
Requirements .....	19
Permissions Required .....	20
Creating a File System .....	20
Creating Mount Targets .....	22
Creating a Mount Target Using the Amazon EFS console .....	23
Creating a Mount Target using the AWS CLI .....	27
Creating Security Groups .....	27
Creating Security Groups Using the AWS Management Console .....	28
Creating Security Groups Using the AWS CLI .....	29
Creating File System Policies .....	29
Creating Access Points .....	31
Using File Systems .....	34
Related Topics .....	34
Using amazon-efs-utils .....	35
Overview .....	35
Installing the amazon-efs-utils Package on Amazon Linux .....	36
Installing the amazon-efs-utils Package on Other Linux Distributions .....	36
Upgrading Stunnel .....	37
Disabling Certificate Hostname Checking .....	38
Enabling Online Certificate Status Protocol .....	39
EFS Mount Helper .....	39
How It Works .....	39
Using the EFS Mount Helper .....	40
Getting Support Logs .....	40
Using amazon-efs-utils with AWS Direct Connect and VPN .....	40
Related Topics .....	40
Managing File Systems .....	41
Managing Network Accessibility .....	41
Creating or Deleting Mount Targets in a VPC .....	43
Changing the VPC for Your Mount Target .....	45

Updating the Mount Target Configuration .....	47
Managing Tags .....	48
Using the Console .....	48
Using the CLI .....	48
Transferring Data into Amazon EFS .....	50
Storage Classes .....	50
Using Storage Classes .....	51
Lifecycle Management .....	51
Using a Lifecycle Policy .....	51
File System Operations for Lifecycle Management .....	51
Pricing and Billing .....	52
Enabling Lifecycle Management .....	52
Stopping Lifecycle Management .....	54
Metering File System and Object Sizes .....	55
Metering Amazon EFS File System Objects .....	55
Metering an Amazon EFS File System .....	56
Managing File System Costs with Budgets .....	57
Prerequisites .....	57
Creating a Monthly Cost Budget for an EFS File System .....	57
Deleting a File System .....	58
Using the Console .....	58
Using the CLI .....	58
Related Topics .....	59
Managing Access to Encrypted File Systems .....	59
Performing Administrative Actions on Amazon EFS Customer Master Keys .....	59
Related Topics .....	60
Mounting EFS File Systems .....	61
Troubleshooting AMI and Kernel Versions .....	61
Installing amazon-efs-utils .....	61
Mounting with the EFS Mount Helper .....	61
Mounting on EC2 with the EFS Mount Helper .....	62
Mounting with IAM Authorization .....	62
Mounting with EFS Access Points .....	63
Mounting Automatically with EFS Mount Helper .....	63
Mounting on Your On-Premises Linux Client with the EFS Mount Helper over AWS Direct Connect and VPN .....	64
Mounting EFS Automatically .....	64
Configuring EC2 Instances to Mount an EFS File System at Instance Launch .....	65
Using /etc/fstab to Mount Automatically .....	66
Mounting from Another Account or VPC .....	68
Mounting Using IAM or Access Points from Another VPC .....	68
Mounting from Another Account in the Same VPC .....	70
Additional Mounting Considerations .....	70
Unmounting File Systems .....	71
Monitoring EFS .....	73
Monitoring Tools .....	73
Automated Tools .....	73
Manual Monitoring Tools .....	74
Monitoring with CloudWatch .....	74
Amazon CloudWatch Metrics for Amazon EFS .....	74
Bytes Reported in CloudWatch .....	77
Amazon EFS Dimensions .....	77
How Do I Use Amazon EFS Metrics? .....	77
Accessing CloudWatch Metrics .....	77
Creating Alarms .....	78
Using Metric Math with Amazon EFS .....	79
Logging Amazon EFS API Calls with AWS CloudTrail .....	82

Amazon EFS Information in CloudTrail .....	82
Understanding Amazon EFS Log File Entries .....	83
Amazon EFS Log File Entries for Encrypted-at-Rest File Systems .....	88
Performance .....	90
Performance Overview .....	90
Amazon EFS Use Cases .....	91
Big Data and Analytics .....	91
Media Processing Workflows .....	91
Content Management and Web Serving .....	91
Home Directories .....	91
Performance Modes .....	91
General Purpose Performance Mode .....	91
Max I/O Performance Mode .....	92
Using the Right Performance Mode .....	92
Throughput Modes .....	92
Bursting .....	93
Provisioned Throughput .....	95
Using the Right Throughput Mode .....	95
On-Premises Performance Considerations .....	96
Architecting for High Availability .....	96
Amazon EFS Performance Tips .....	96
Related Topics .....	97
Data Protection for Amazon EFS .....	98
AWS Backup and EFS .....	98
How AWS Backup Works with EFS File Systems .....	98
Walkthroughs .....	101
Walkthrough: Create and Mount a File System Using the AWS CLI .....	101
Before You Begin .....	102
Setting Up Tools .....	102
Step 1: Create Amazon EC2 Resources .....	103
Step 2: Create Amazon EFS Resources .....	107
Step 3: Mount and Test the File System .....	109
Step 4: Clean Up .....	111
Walkthrough: Set Up an Apache Web Server and Serve Files .....	113
Single EC2 Instance Serving Files .....	113
Multiple EC2 Instances Serving Files .....	115
Walkthrough: Create Writable Per-User Subdirectories .....	118
Automatic Remounting on Reboot .....	119
Walkthrough: Mount EFS on an On-Premises Client .....	119
Before You Begin .....	120
Step 1: Create Your Amazon Elastic File System Resources .....	120
Step 2: Install the NFS Client .....	121
Step 3: Mount the Amazon EFS File System on Your On-Premises Client .....	121
Step 4: Clean Up Resources and Protect Your AWS Account .....	122
Optional: Encrypting Data in Transit .....	123
Walkthrough: Mount a File System from a Different VPC .....	125
Before You Begin .....	126
Step 1: Determine the Availability Zone ID of the EFS Mount Target .....	126
Step 2: Determine the Mount Target IP Address .....	127
Step 3: Add a Host Entry for the Mount Target .....	127
Step 4: Mount Your File System Using the EFS Mount Helper .....	128
Step 5: Clean Up Resources and Protect Your AWS Account .....	129
Walkthrough: Enforcing Encryption on an Amazon EFS File System at Rest .....	130
Enforcing Encryption at Rest .....	130
Walkthrough: Enable Root Squashing Using IAM for NFS .....	132
Security .....	135
Data Encryption in EFS .....	135

When to Use Encryption .....	135
Encrypting Data at Rest .....	136
Encrypting Data in Transit .....	138
Identity and Access Management .....	139
Authentication .....	139
Access Control .....	141
Overview of Managing Access .....	141
Controlling API Access .....	144
Using IAM to Control NFS Access .....	150
Using EFS Service-Linked Role .....	152
Controlling Network Access .....	154
Using Security Groups for Amazon EC2 Instances and Mount Targets .....	154
Source Ports .....	156
Security Considerations for Network Access .....	156
Working with VPC Endpoints .....	156
NFS-Level Users, Groups, and Permissions .....	158
File and Directory Permissions .....	158
Example Amazon EFS File System Use Cases and Permissions .....	159
User and Group ID Permissions for Files and Directories Within a File System .....	159
No Root Squashing .....	160
Permissions Caching .....	161
Changing File System Object Ownership .....	161
EFS Access Points .....	161
Working with Access Points .....	161
Creating an Access Point .....	161
Mounting a File System Using an Access Point .....	162
Enforcing a User Identity Using an Access Point .....	162
Enforcing a Root Directory with an Access Point .....	162
Using Access Points in IAM Policies .....	163
Logging and Monitoring .....	164
Compliance Validation .....	165
Resilience .....	165
Network Isolation .....	166
EFS Quotas .....	167
Amazon EFS Quotas That You Can Increase .....	167
Requesting a Quota Increase .....	168
Resource Quotas .....	168
Quotas for NFS Clients .....	169
Quotas for Amazon EFS File Systems .....	169
Unsupported NFSv4 Features .....	170
Additional Considerations .....	170
Troubleshooting Amazon EFS .....	172
Troubleshooting General Issues .....	172
Amazon EC2 Instance Hangs .....	172
Application Writing Large Amounts of Data Hangs .....	173
Poor Performance When Opening Many Files in Parallel .....	173
Custom NFS Settings Causing Write Delays .....	173
Creating Backups with Oracle Recovery Manager Is Slow .....	174
Troubleshooting File Operation Errors .....	174
Command Fails with "Disk quota exceeded" Error .....	174
Command Fails with "I/O error" .....	175
Command Fails with "File name is too long" Error .....	175
Command Fails with "File not found" Error .....	175
Command Fails with "Too many links" Error .....	176
Command Fails with "File too large" Error .....	176
Troubleshooting AMI and Kernel Issues .....	176
Unable to chown .....	176

File System Keeps Performing Operations Repeatedly Due to Client Bug .....	176
Deadlocked Client .....	177
Listing Files in a Large Directory Takes a Long Time .....	177
Troubleshooting Mount Issues .....	177
File System Mount on Windows Instance Fails .....	178
Access Denied by Server .....	178
Automatic Mounting Fails and the Instance Is Unresponsive .....	178
Mounting Multiple Amazon EFS File Systems in /etc/fstab Fails .....	178
Mount Command Fails with "wrong fs type" Error Message .....	179
Mount Command Fails with "incorrect mount option" Error Message .....	179
File System Mount Fails Immediately After File System Creation .....	180
File System Mount Hangs and Then Fails with Timeout Error .....	180
File System Mount Using DNS Name Fails .....	180
File System Mount Fails with "nfs not responding" .....	181
Mount Target Lifecycle State Is Stuck .....	181
Mount Does Not Respond .....	181
Operations on Newly Mounted File System Return "bad file handle" Error .....	182
Unmounting a File System Fails .....	182
Troubleshooting Encryption .....	182
Mounting with Encryption of Data in Transit Fails .....	182
Mounting with Encryption of Data in Transit is Interrupted .....	183
Encrypted-at-Rest File System Can't Be Created .....	183
Unusable Encrypted File System .....	183
Amazon EFS API .....	185
API Endpoint .....	185
API Version .....	185
Related Topics .....	186
Working with the Query API Request Rate for Amazon EFS .....	186
Polling .....	186
Retries or Batch Processing .....	186
Calculating the Sleep Interval .....	186
Actions .....	186
CreateAccessPoint .....	188
CreateFileSystem .....	192
CreateMountTarget .....	200
CreateTags .....	207
DeleteAccessPoint .....	210
DeleteFileSystem .....	212
DeleteFileSystemPolicy .....	214
DeleteMountTarget .....	216
DeleteTags .....	219
DescribeAccessPoints .....	221
DescribeFileSystemPolicy .....	224
DescribeFileSystems .....	227
DescribeLifecycleConfiguration .....	231
DescribeMountTargets .....	234
DescribeMountTargetSecurityGroups .....	238
DescribeTags .....	241
ListTagsForResource .....	244
ModifyMountTargetSecurityGroups .....	246
PutFileSystemPolicy .....	249
PutLifecycleConfiguration .....	252
TagResource .....	256
UntagResource .....	258
UpdateFileSystem .....	260
Data Types .....	264
AccessPointDescription .....	265

CreationInfo .....	267
FileSystemDescription .....	268
FileSystemSize .....	271
LifecyclePolicy .....	272
MountTargetDescription .....	273
PosixUser .....	275
RootDirectory .....	276
Tag .....	277
Additional Information .....	278
Backup with AWS Data Pipeline .....	278
Recommended EFS Backup Solutions .....	278
Legacy EFS Backup Solution Using AWS Data Pipeline .....	278
Performance for Amazon EFS Backups Using AWS Data Pipeline .....	279
Considerations for Amazon EFS Backup by Using AWS Data Pipeline .....	280
Assumptions for Amazon EFS Backup with AWS Data Pipeline .....	280
How to Back Up an Amazon EFS File System with AWS Data Pipeline .....	281
Additional Backup Resources .....	286
Mounting File Systems Without the EFS Mount Helper .....	290
NFS Support .....	291
Installing the NFS Client .....	291
Recommended NFS Mount Options .....	293
Mounting on Amazon EC2 with a DNS Name .....	294
Mounting with an IP Address .....	295
Document History .....	297



# What Is Amazon Elastic File System?

Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth. Amazon EFS has a simple web services interface that allows you to 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. Multiple Amazon EC2 instances can access an Amazon EFS file system at the same time, providing a common data source for workloads and applications running on more than one instance or server.

With Amazon EFS, you pay only for the storage used by your file system and there is no minimum fee or setup cost. Amazon EFS offers two storage classes, Standard and Infrequent Access. The Standard storage class is used to store frequently accessed files. The Infrequent Access (IA) storage class is a lower-cost storage class that's designed for storing long-lived, infrequently accessed files cost-effectively. For more information, see [EFS Storage Classes \(p. 50\)](#). Costs related to Provisioned Throughput are determined by the throughput values you specify. For more information, see [Amazon EFS Pricing](#).

The service is designed to be highly scalable, highly available, and highly durable. Amazon EFS file systems store data and metadata across multiple Availability Zones in an AWS Region. EFS file systems can grow to petabyte scale, drive high levels of throughput, and allow massively parallel access from Amazon EC2 instances to your data.

Amazon EFS provides file system access semantics, such as strong data consistency and file locking. For more information, see [Data Consistency in Amazon EFS \(p. 7\)](#). Amazon EFS also enables you to control access to your file systems through Portable Operating System Interface (POSIX) permissions. For more information, see [Security in Amazon EFS \(p. 135\)](#).

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 Amazon EFS file system. If you do, all your data and metadata is encrypted. You can enable encryption in transit when you mount the file system. NFS client access to EFS is controlled by both AWS Identity and Access Management (IAM) policies and network security policies like security groups. For more information, see [Data Encryption in EFS \(p. 135\)](#), [Identity and Access Management for Amazon EFS \(p. 139\)](#), and [Controlling Network Access to Amazon EFS File Systems for NFS Clients \(p. 154\)](#).

Amazon EFS is designed to provide the throughput, IOPS, and low latency needed for a broad range of workloads. With Amazon EFS, you can choose from two performance modes and two throughput modes:

- The default General Purpose performance mode is ideal for latency-sensitive use cases, like web serving environments, content management systems, home directories, and general file serving. File systems in the Max I/O mode can scale to higher levels of aggregate throughput and operations per second with a tradeoff of slightly higher latencies for file metadata operations. For more information, see [Performance Modes \(p. 91\)](#).
- Using the default Bursting Throughput mode, throughput scales as your file system grows. Using Provisioned Throughput mode, you can specify the throughput of your file system independent of the amount of data stored. For more information, see [Amazon EFS Performance \(p. 90\)](#).

## 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 [Amazon EFS: How It Works \(p. 3\)](#).
3. Try the introductory exercises:
  - [Getting Started \(p. 11\)](#)
  - [Walkthroughs \(p. 101\)](#)

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

- [Creating Resources for Amazon EFS \(p. 18\)](#)
- [Managing Amazon EFS File Systems \(p. 41\)](#)
- [Amazon EFS API \(p. 185\)](#)

# Amazon EFS: How It Works

Following, you can find a description about how Amazon EFS works, its implementation details, and security considerations.

## Topics

- [Overview \(p. 3\)](#)
- [How Amazon EFS Works with Amazon EC2 \(p. 4\)](#)
- [How Amazon EFS Works with AWS Direct Connect and AWS Managed VPN \(p. 4\)](#)
- [How Amazon EFS Works with AWS Backup \(p. 5\)](#)
- [Implementation Summary \(p. 6\)](#)
- [Authentication and Access Control \(p. 7\)](#)
- [Data Consistency in Amazon EFS \(p. 7\)](#)
- [Storage Classes and Lifecycle Management \(p. 7\)](#)

## Overview

Amazon EFS provides file storage in the AWS Cloud. 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 Amazon EFS file system in your 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, Redhat, and Ubuntu AMIs, in conjunction with the Amazon EFS Mount Helper. For instructions, see [Using the amazon-efs-utils Tools \(p. 35\)](#).

For a list of Amazon EC2 Linux Amazon Machine Images (AMIs) that support this protocol, see [NFS Support \(p. 291\)](#). For some AMIs, you'll need to install an NFS client to mount your file system on your Amazon EC2 instance. For instructions, see [Installing the NFS Client \(p. 291\)](#).

You can access your Amazon EFS file system concurrently from multiple NFS clients, so applications that scale beyond a single connection can access a file system. Amazon EC2 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 Amazon EFS file system, see the [Amazon Web Services General Reference](#).

To access your Amazon 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 Amazon 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 Amazon EFS file system can only have mount targets in one VPC at a time.

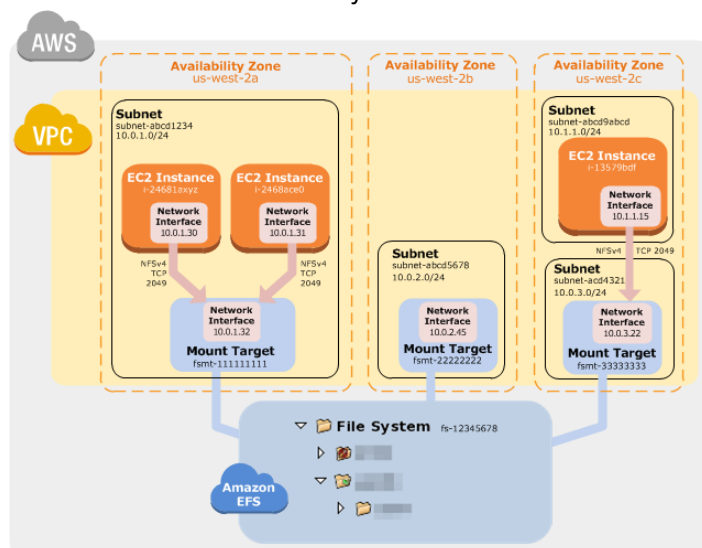
Mount targets themselves are designed to be highly available. As you design for high availability and failover to other Availability Zones (AZs), keep in mind that while the IP addresses and DNS for your mount targets in each AZ are static, they are redundant components backed by multiple resources.

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 [Working with Users, Groups, and Permissions at the Network File System \(NFS\) Level](#) (p. 158).

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 AWS VPN. You can mount your EFS file systems on on-premises servers to migrate datasets to EFS, enable cloud bursting scenarios, or backup your on-premises data to EFS.

## How Amazon EFS Works with Amazon EC2

The following illustration shows an example VPC accessing an Amazon EFS file system. Here, EC2 instances in the VPC have file systems mounted.



In this illustration, the VPC has three Availability Zones, and each has one mount target created in it. We recommend that you access the file system from a mount target within the same Availability Zone. One of the Availability Zones has two subnets. However, a mount target is created in only one of the subnets. Creating this setup works as follows:

1. Create your Amazon EC2 resources and launch your Amazon EC2 instance. For more information on Amazon EC2, see [Amazon EC2 - Virtual Server Hosting](#).
2. Create your Amazon EFS file system.
3. Connect to your Amazon EC2 instance, and mount the Amazon EFS file system.

For detailed steps, see [Getting Started with Amazon Elastic File System](#) (p. 11).

## 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

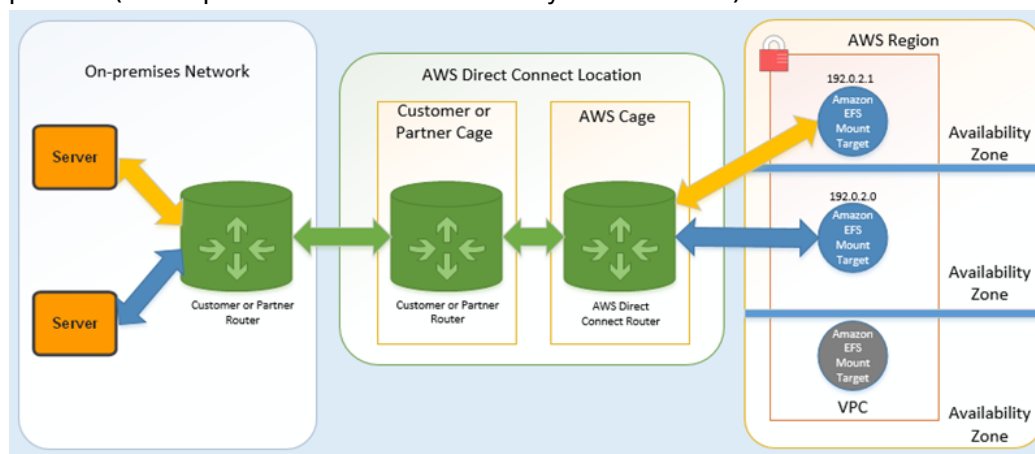
fleet of Amazon EC2 instances in your Amazon VPC. You can then store the results permanently in your file system or move the results back to your on-premises server.

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.

To create a setup like this, you do the following:

1. Establish an AWS Direct Connect connection between your on-premises data center and your Amazon VPC. For more information on AWS Direct Connect, see [AWS Direct Connect](#).
2. Create your Amazon EFS file system.
3. Mount the Amazon EFS file system on your on-premises server.

For detailed steps, see [Walkthrough: Create and Mount a File System On-Premises with AWS Direct Connect and VPN](#) (p. 119).

## 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 [Using AWS Backup with Amazon EFS \(p. 98\)](#).

## 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 state. For more information, see [CreateFileSystem \(p. 192\)](#).

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,wsiz=1048576,hard,timeo=600,retrans=2,noresvport file-system-  
DNS-name:/ ~/efs-mount-point
```

For more information, see [Creating Mount Targets \(p. 22\)](#). First, you need to install the NFS client on your EC2 instance. The [Getting Started \(p. 11\)](#) exercise provides step-by-step instructions.

- **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 Amazon EFS Access Points \(p. 161\)](#).

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 also 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 \(p. 11\)](#).
- Use the Amazon EFS command line interface (CLI) – For an example, see [Walkthrough: Create an Amazon EFS File System and Mount It on an Amazon EC2 Instance Using the AWS CLI \(p. 101\)](#).
- 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 \(p. 185\)](#).

## Authentication and Access Control

You must have valid credentials to make Amazon EFS API requests, such as create a file system. In addition, you must also have permissions to create or access resources. By default, when you use the root account credentials of your AWS account you can create and access resources owned by that account. However, we don't recommend using root account credentials. In addition, any AWS Identity and Access Management (IAM) users and roles that you create in your account must be granted permissions to create or access resources. For more information about permissions, see [Identity and Access Management for Amazon EFS \(p. 139\)](#).

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 NFS Access to Amazon EFS \(p. 150\)](#).

## Data Consistency in Amazon EFS

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

In Amazon EFS, write operations 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 nonappending writes have read-after-write consistency for data access.

## Storage Classes and Lifecycle Management

With Amazon EFS, you can use two storage classes for your file systems:

- **Infrequent Access** – The Infrequent Access (IA) storage class is a lower-cost storage class that's designed for storing long-lived, infrequently accessed files cost-effectively.
- **Standard** – The Standard storage class is used to store frequently accessed files.

The EFS IA storage class reduces storage costs for files that aren't accessed every day. It does this without sacrificing the high availability, high durability, elasticity, and POSIX file system access that EFS provides. We recommend EFS IA storage if you need your full dataset to be readily accessible and want to automatically save on storage costs for files that are less frequently accessed. Examples include

keeping files accessible to satisfy audit requirements, perform historical analysis, or perform backup and recovery. To learn more about EFS storage classes, see [EFS Storage Classes \(p. 50\)](#).

Amazon EFS lifecycle management automatically manages cost-effective file storage for your file systems. When enabled, lifecycle management migrates files that haven't been accessed for a set period of time to the Infrequent Access (IA) storage class. You define that period of time by using a *lifecycle policy*. To learn more about lifecycle management, see [EFS Lifecycle Management \(p. 51\)](#).



# Setting Up

Before you use Amazon EFS for the first time, complete the following tasks:

1. [Sign up for AWS](#) (p. 9)
2. [Create an IAM User](#) (p. 9)

## Sign up for AWS

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

With Amazon EFS, you pay only for the storage you use. For more information about Amazon EFS usage rates, see the [Amazon Elastic File System Pricing](#). If you are a new AWS customer, you can get started with Amazon EFS for free. For more information, see [AWS Free Usage Tier](#).

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

### To create an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

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

Note your AWS account number, because you'll need it for the next task.

## Create an IAM User

Services in AWS, such as Amazon EFS, require that you provide credentials when you access them, so that the service can determine whether you have permissions to access its resources. AWS recommends that you do not use the root credentials of your AWS account to make requests. Instead, create an IAM user, and grant that user full access. We refer to these users as administrator users. You can use the administrator user credentials, instead of root credentials of your account, to interact with AWS and perform tasks, such as create a bucket, create users, and grant them permissions. For more information, see [Root Account Credentials vs. IAM User Credentials](#) in the *AWS General Reference* and [IAM Best Practices](#) in the *IAM User Guide*.

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console.

### To create an administrator user for yourself and add the user to an administrators group (console)

1. Use your AWS account email address and password to sign in as the *AWS account root user* to the IAM console at <https://console.aws.amazon.com/iam/>.

**Note**

We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user below and securely lock away the root user credentials. Sign in as the root user only to perform a few [account and service management tasks](#).

2. In the navigation pane, choose **Users** and then choose **Add user**.
3. For **User name**, enter **Administrator**.
4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.
5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Under **Set permissions**, choose **Add user to group**.
8. Choose **Create group**.
9. In the **Create group** dialog box, for **Group name** enter **Administrators**.
10. Choose **Filter policies**, and then select **AWS managed -job function** to filter the table contents.
11. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

**Note**

You must activate IAM user and role access to Billing before you can use the **AdministratorAccess** permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in [step 1 of the tutorial about delegating access to the billing console](#).

12. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
13. Choose **Next: Tags**.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see [Tagging IAM Entities](#) in the *IAM User Guide*.
15. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see [Access Management](#) and [Example Policies](#).

To sign in as this new IAM user, sign out of the AWS Management Console, and then use the following URL, where *your\_aws\_account\_id* is your AWS account number without the hyphens (for example, if your AWS account number is 1234-5678-9012, your AWS account ID is 123456789012):

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

Enter the IAM user name and password that you just created. When you're signed in, the navigation bar displays ***your\_user\_name@your\_aws\_account\_id***.

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM dashboard, locate **IAM users sign-in link:** and choose **Customize**. Enter an alias, such as your company name. To sign in after you create an account alias, use the following URL:

```
https://your_account_alias.signin.aws.amazon.com/console/
```

To verify the sign-in link for IAM users for your account, open the IAM console and check under **AWS Account Alias** on the dashboard.

# Getting Started with Amazon Elastic File System

In this Getting Started exercise, you can learn how to quickly create an Amazon Elastic File System (Amazon EFS) file system. As part of this process, you mount your file system on an Amazon Elastic Compute Cloud (Amazon EC2) instance in your virtual private cloud (VPC). You also test the end-to-end setup.

There are four steps that you need to perform to create and use your first Amazon EFS file system:

- Create your Amazon EFS file system.
- Create your Amazon EC2 resources, launch your instance, and mount the file system.
- Transfer files to your EFS file system using AWS DataSync.
- Clean up your resources and protect your AWS account.

## Topics

- [Assumptions \(p. 11\)](#)
- [Related Topics \(p. 11\)](#)
- [Step 1: Create Your Amazon EFS File System \(p. 12\)](#)
- [Step 2: Create Your EC2 Resources and Launch Your EC2 Instance \(p. 15\)](#)
- [Step 3: Transfer Files to Amazon EFS Using AWS DataSync \(p. 16\)](#)
- [Step 4: Clean Up Resources and Protect Your AWS Account \(p. 16\)](#)

## Assumptions

For this exercise, we assume the following:

- You're already familiar with using the Amazon EC2 console to launch instances.
- Your Amazon VPC, Amazon EC2, and Amazon EFS resources are all in the same AWS Region. This guide uses the US West (Oregon) Region (us-west-2).
- You have a default VPC in the AWS Region that you're using for this Getting Started exercise. 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, you can still use this Getting Started exercise. To do so, configure [Using Security Groups for Amazon EC2 Instances and Mount Targets \(p. 154\)](#).
- You haven't changed the default inbound access rule for the default security group.

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. For more information, see [Setting Up \(p. 9\)](#).

## Related Topics

This guide also provides a walkthrough to perform a similar Getting Started exercise using AWS Command Line Interface (AWS CLI) commands to make the Amazon EFS API calls. For more information,

see [Walkthrough: Create an Amazon EFS File System and Mount It on an Amazon EC2 Instance Using the AWS CLI](#) (p. 101).

## Step 1: Create Your Amazon EFS File System

In this step, you create your Amazon EFS file system.

### To create your Amazon EFS file system

1. Open the Amazon EFS Management Console at <https://console.aws.amazon.com/efs/>.
2. Choose **Create File System**. The **Create file system** page is displayed.
3. In the **Configure network access** section, for **VPC**, choose your default VPC. It looks something like `vpc-xxxxxxx` (`172.31.0.0/16`) (`default`). Note the VPC ID; you need it in a later step.
4. Select the check boxes 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 [Creating Mount Targets](#) (p. 22).

## Create file system

### Step 1: Configure network access

Step 2: Configure file system settings

Step 3: Review and create

### Configure network access

An Amazon EFS file system is accessed by EC2 mount targets. Each mount target has an IP address, which we assign to a subnet in your VPC.

**VPC** vpc- (default) ▼

### Create mount targets

Instances connect to a file system by using mount targets. You can create mount targets in any subnet in your VPC that can access the file system.

	Availability Zone	Subnet
<input checked="" type="checkbox"/>	us-east-1a	subnet-
<input checked="" type="checkbox"/>	us-east-1b	subnet-
<input checked="" type="checkbox"/>	us-east-1c	subnet-
<input checked="" type="checkbox"/>	us-east-1d	subnet-
<input checked="" type="checkbox"/>	us-east-1e	subnet-
<input checked="" type="checkbox"/>	us-east-1f	subnet-

5. Choose **Next Step**.
6. In **Add tags**, name your file system, and add any other tags to help describe and manage your file system.
7. (Optional) In **Enable Lifecycle Management**, select a **Lifecycle policy** so that your file system uses the lower-cost Infrequent Access storage class. For more information about storage classes, see [EFS Storage Classes \(p. 50\)](#).
8. Keep **Bursting** and **General Purpose** selected as your default performance and throughput modes.
9. (Optional) For **Enable encryption**, leave **Enable encryption of data at rest** unchecked for this exercise. For more information on encrypting the data on your file system, see [Data Encryption in EFS \(p. 135\)](#). Choose **Next Step**. The **Configure client access** page is displayed.
10. For this exercise, leave the default settings for **File system policy** and **Access Points**. For more information about EFS file system policies and how to use them, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#). For more information about EFS access points, see [Working with Amazon EFS Access Points \(p. 161\)](#). Choose **Next Step**. The **Review and create** page is displayed.

### Review and create

Review the configuration below before proceeding to create your file system.

#### File system access

VPC	Availability Zone	Subnet	IP address	Security group
vpc-6296a00a - VPC-2 (default)	us-east-2a	subnet-a3520bcb (default)	Automatic	sg-9a01b14d
	us-east-2b	subnet-95bb2cef (default)	Automatic	sg-9a01b14d
	us-east-2c	subnet-c24eb18e (default)	Automatic	sg-9a01b14d

#### Optional settings

Tags

Name: Example\_EFS\_file\_system

Performance mode

General Purpose

Throughput mode

Bursting

Encrypted

No

Lifecycle policy

14 days since last access

Number of access points

0

File system policy

None

[Cancel](#)[Previous](#)

11. Review the file system configuration, and then choose **Create File System**.
12. The **File systems** page is displayed, with your new file system selected. Note the **File system ID** value. You need this value for the next step.

## Step 2: Create Your EC2 Resources and Launch Your EC2 Instance

### Note

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

Before you can launch and connect to an Amazon EC2 instance, you need to create a key pair, unless you already have one. You can create a key pair using the Amazon EC2 console, and then you can launch your EC2 instance.

### To create a key pair

- Follow the steps in [Setting Up with Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances* to create a key pair. If you already have a key pair, you don't need to create a new one. You can use your existing key pair for this exercise.

### To launch the EC2 instance and mount an EFS file system

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 you noted when you created your EFS file system in [Step 1: Create Your Amazon EFS File System \(p. 12\)](#).
  - For **Subnet**, choose a default subnet in any Availability Zone.
  - For **File systems**, make sure that the EFS file system that you created in [Step 1: Create Your Amazon EFS File System \(p. 12\)](#) is selected. The path shown next to the file system ID is the mount point that the EC2 instance will use, which you can change. Choose **Add to user data** to mount the file system when the EC2 is launched.
  - Under **Advanced Details**, confirm that the user data is present in **User data**.
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. SSH access isn't required for this exercise. To add access by SSH later, you can edit the default security and add a rule to allow SSH. Or you can create a new security group that allows SSH. You can use the following settings to add SSH access:

- **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**.

## Step 3: Transfer Files to Amazon EFS Using AWS DataSync

Now that you have created a functioning Amazon EFS file system, you can use AWS DataSync to transfer files from an existing file system to Amazon EFS. AWS 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. AWS DataSync can transfer your file data, and also file system metadata such as ownership, time stamps, and access permissions.

### Before You Begin

In this step, we assume that 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 Amazon EFS file systems.
- A destination Amazon EFS file system to transfer files to. If you don't have an Amazon EFS file system, create one. For more information, see [Getting Started with Amazon Elastic File System \(p. 11\)](#).
- Your server and network meet the AWS DataSync requirements. To learn more, see the [AWS DataSync requirements](#).

To transfer files from a source location to a destination location using AWS DataSync, you do the following:

- Download and deploy an agent in your environment and activate it.
- Create and configure a source and destination location.
- Create and configure a task.
- Run the task to transfer files from the source to the destination.

To learn how to transfer files from an existing on-premises file system to your EFS file system, see [Getting Started with AWS DataSync](#) in the *AWS DataSync User Guide*. To learn how to transfer files from an existing in-cloud file system to your EFS file system, see [Deploying the AWS DataSync Agent as an Amazon EC2 Instance](#) in the *AWS DataSync User Guide*, and the [Amazon EFS AWS DataSync In-Cloud Transfer Quick Start and Scheduler](#).

## Step 4: Clean Up Resources and Protect Your AWS Account

This guide includes walkthroughs that you can use to further explore Amazon EFS. Before you perform this clean-up step, you can use the resources you've created and connected to in this Getting Started exercise in those walkthroughs. For more information, see [Walkthroughs \(p. 101\)](#). After you finish the walkthroughs, or if you don't want to explore the walkthroughs, take the following steps to clean up your resources and protect your AWS account.

### To clean up resources and protect your AWS account

1. Connect to your Amazon EC2 instance.
2. Unmount the Amazon EFS file system with the following command.



```
$ sudo umount efs
```

3. Open the Amazon EFS console at <https://console.aws.amazon.com/efs/>.
4. Choose the Amazon EFS file system that you want to delete from the list of file systems.
5. For **Actions**, choose **Delete file system**.
6. 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**.
7. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
8. Choose the Amazon EC2 instance that you want to terminate from the list of instances.
9. For **Actions**, choose **Instance State** and then choose **Terminate**.
10. In **Terminate Instances**, choose **Yes, Terminate** to terminate the instance that you created for this Getting Started exercise.
11. In the navigation pane, choose **Security Groups**.
12. Select the name of the security group that you created for this Getting Started exercise in [Step 2: Create Your EC2 Resources and Launch Your EC2 Instance \(p. 15\)](#) as a part of the Amazon EC2 instance launch wizard.

**Warning**

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

13. For **Actions**, choose **Delete Security Group**.
14. In **Delete Security Group**, choose **Yes, Delete** to delete the security group you created for this Getting Started exercise.

# Creating Resources for Amazon EFS

Amazon EFS provides elastic, shared file storage that is POSIX-compliant. The file system you create supports concurrent read and write access from multiple Amazon EC2 instances and is 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 using the Network File System versions 4.0 and 4.1 protocol (NFSv4). For more information, see [Amazon EFS: How It Works \(p. 3\)](#).

## Topics

- [Creating an Amazon Elastic File System \(p. 19\)](#)
- [Creating Mount Targets \(p. 22\)](#)
- [Creating Security Groups \(p. 27\)](#)
- [Creating File System Policies \(p. 29\)](#)
- [Creating Access Points \(p. 31\)](#)

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 you need to perform to use Amazon EFS file systems in the VPC:

- **Create an Amazon EFS file system** – When creating a file system, we recommend that you consider using the **Name** tag because 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 need to have associated security groups. These security groups act as a virtual firewall that controls the traffic between them. You can use the security group you associated with the mount target to control inbound traffic to your file system by adding 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.

If you are new to Amazon EFS, we recommend that you try the following exercises that provide a first-hand, end-to-end experience of using an Amazon EFS file system:

- [Getting Started \(p. 11\)](#) – The Getting Started exercise provides a console-based end-to-end setup in which you create a file system, mount it on an EC2 instance, and test the setup. The console takes care of many things for you and helps you set up the end-to-end experience quickly.
- [Walkthrough: Create an Amazon EFS File System and Mount It on an Amazon EC2 Instance Using the AWS CLI \(p. 101\)](#) – The walkthrough is similar to the Getting Started exercise, but it uses the AWS Command Line Interface (AWS CLI) to perform most of the tasks. Because the AWS CLI commands closely map to the Amazon EFS API, the walkthrough can help you familiarize yourself with the Amazon EFS API operations.

For more information about creating and accessing a file system, see the following topics.

## Topics

- [Creating an Amazon Elastic File System \(p. 19\)](#)
- [Creating Mount Targets \(p. 22\)](#)

- [Creating Security Groups \(p. 27\)](#)
- [Creating File System Policies \(p. 29\)](#)
- [Creating Access Points \(p. 31\)](#)

## Creating an Amazon Elastic File System

Following, you can find an explanation about how to create an Amazon EFS file system and optional tags for the file system. This section explains how to create these resources using both the console and the AWS CLI.

### Note

If you are new to Amazon EFS, we recommend you go through the Getting Started exercise, which provides console-based end-to-end instructions to create and access a file system in your VPC. For more information, see [Getting Started \(p. 11\)](#).

### Topics

- [Requirements \(p. 19\)](#)
- [Permissions Required \(p. 20\)](#)
- [Creating a File System \(p. 20\)](#)

## Requirements

To create a file system, the only requirement is that you create a token to ensure idempotent operation. If you use the console, it generates the token for you. For more information, see [CreateFileSystem \(p. 192\)](#). After you create a file system, Amazon EFS returns the file system description as JSON. Following is an example.

```
{
  "OwnerId": "231243201240",
  "CreationToken": "console-d7f56c5f-e433-41ca-8307-9d9c0example",
  "FileSystemId": "fs-c7a0456e",
  "CreationTime": 1422823614.0,
  "LifecycleState": "creating",
  "Name": "MyFileSystem",
  "NumberOfMountTargets": 0,
  "SizeInBytes": {
    "Value": 6144,
    "ValueInIA": 0
    "ValueInStandard": 6144
  },
  "PerformanceMode": "generalPurpose",
  "Encrypted": false,
  "ThroughputMode": "bursting",
  "Tags": [
    {
      "Key": "Name",
      "Value": "MyFileSystem"
    }
  ]
}
```

If you use the console, the console displays this information in the user interface.

You can create additional tags and edit existing tags later using the Amazon EFS [CreateTags \(p. 207\)](#) operation. Each tag is simply a key-value pair.

## Permissions Required

For all operations, such as creating a file system and creating tags, a user must have AWS Identity and Access Management permissions for the corresponding API action and resource.

You can perform any Amazon EFS operations using the root credentials of your AWS account, but using root credentials is not recommended. If you create IAM users in your account, you can grant them permissions for Amazon EFS actions with user policies. You can also use roles to grant cross-account permissions. For more information about managing permissions for the API actions, see [Identity and Access Management for Amazon EFS \(p. 139\)](#).

## Creating a File System

You can create a file system using the Amazon EFS console or using the AWS CLI. You can also create file systems programmatically using AWS SDKs or the EFS API directly.

To create the file system mount targets in your VPC, you must specify VPC subnets. The console prepopulates the list of VPCs in your account that are in the selected AWS Region. First, you select your VPC, and then the console lists the Availability Zones in the VPC. For each Availability Zone, you can select a subnet from the list. After you select a subnet, you can either specify an available IP address in the subnet or let Amazon EFS choose an address.

When creating a file system, you also choose a performance mode. There are two performance modes to choose from—General Purpose and Max I/O. For the majority of use cases, we recommend that you use the general purpose performance mode for your file system. For more information about the different performance modes, see [Performance Modes \(p. 91\)](#).

In addition to performance modes, you can also choose your throughput mode. There are two throughput modes to choose from—Bursting and Provisioned. The default Bursting Throughput mode is simple to work with and is suitable for a majority of applications and a wide range of performance requirements. Provisioned mode is for applications that require a greater ratio of throughput to storage capacity than allowed by Bursting Throughput mode. For more information, see [Specifying Throughput with Provisioned Mode \(p. 95\)](#).

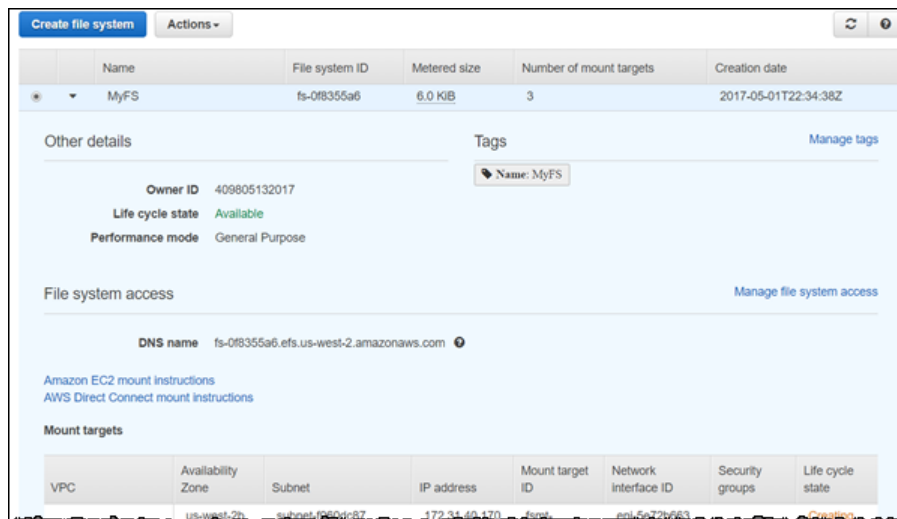
### Note

There are additional charges associated with using Provisioned Throughput mode. For more information, see <https://aws.amazon.com/efs/pricing>.

You can enable encryption at rest when creating a file system. If you enable encryption at rest for your file system, all data and metadata stored on it is encrypted. You can enable encryption in transit later, when you mount the file system. For more information about Amazon EFS encryption, see [Security in Amazon EFS \(p. 135\)](#).

## Creating a File System Using the Amazon EFS Console

When you choose **Create File System**, the console sends a series of API requests to create the file system. The console then sends API requests to create tags and mount targets for the file system. The following example console shows the **MyFS** file system. It has the **Name** tag and three mount targets that are being created. The mount target lifecycle state must be **Available** before you can use it to mount the file system on an EC2 instance.



For instructions on how to create an Amazon EFS file system using the console, see [Step 2: Create Your EC2 Resources and Launch Your EC2 Instance](#) (p. 15).

## Creating a File System Using the AWS CLI

When 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 using corresponding AWS CLI commands.

The following examples use the `adminuser` as the `profile` parameter value. You need to use an appropriate user profile to provide your credentials. For information about the AWS CLI, see [Getting Set Up with the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

- To create a file system, use the Amazon EFS `create-file-system` CLI command (the corresponding operation is [CreateFileSystem](#) (p. 192)), as shown following.

```
$ aws efs create-file-system \
--creation-token creation-token \
--performance-mode generalPurpose \
--throughput-mode bursting \
--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 in the **us-west-2** AWS Region. The command specifies **MyFirstFS** as the creation token. For a list of AWS Regions where you can create an Amazon EFS file system, see the [Amazon Web Services General Reference](#).

```
$ aws efs create-file-system \
--creation-token MyFirstFS \
--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.

```
{
  "OwnerID": "123456789abcd",
  "CreationToken": "MyFirstFS",
  "FileSystemId": "fs-c7a0456e",
  "CreationTime": 1422823614.0,
  "LifecycleState": "creating",
  "Name": "Test File System",
  "NumberOfMountTargets": 0,
  "SizeInBytes": {
    "Value": 6144,
    "ValueInIA": 0,
    "ValueInStandard": 6144
  },
  "PerformanceMode": "generalPurpose",
  "ThroughputMode": "bursting",
  "Tags": [
    {
      "Key": "Name",
      "Value": "Test File System"
    },
    {
      "Key": "developer",
      "Value": "rhoward"
    }
  ]
}
```

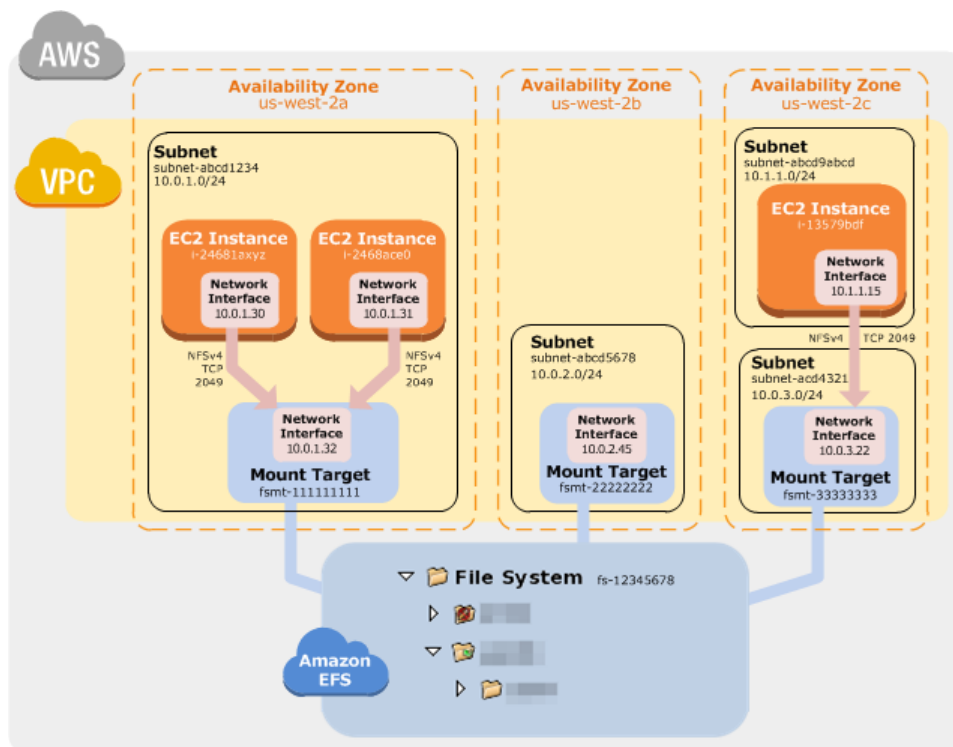
Amazon EFS also provides the `describe-file-systems` CLI command (the corresponding API operation is [DescribeFileSystems](#) (p. 227)) that you can use to retrieve a list of file systems in your account, as shown following:

```
$ 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.

## Creating Mount Targets

After you create a file system, you can create mount targets and then you can mount the file system on EC2 instances in your VPC, as shown in the following illustration.



For more information about creating a file system, see [Creating an Amazon Elastic File System \(p. 19\)](#).

The mount target security group acts as a virtual firewall that controls the traffic. For example, it determines which Amazon EC2 instances can access the file system. This section explains the following:

- Mount target security groups and how to enable traffic.
- How to mount the file system on your Amazon EC2 instance.
- 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 [Working with Users, Groups, and Permissions at the Network File System \(NFS\) Level \(p. 158\)](#).

You can create mount targets for a file system using the console, using AWS CLI, or programmatically using the AWS SDKs. When using the console, you can create mount targets when you first create a file system or after the file system is created.

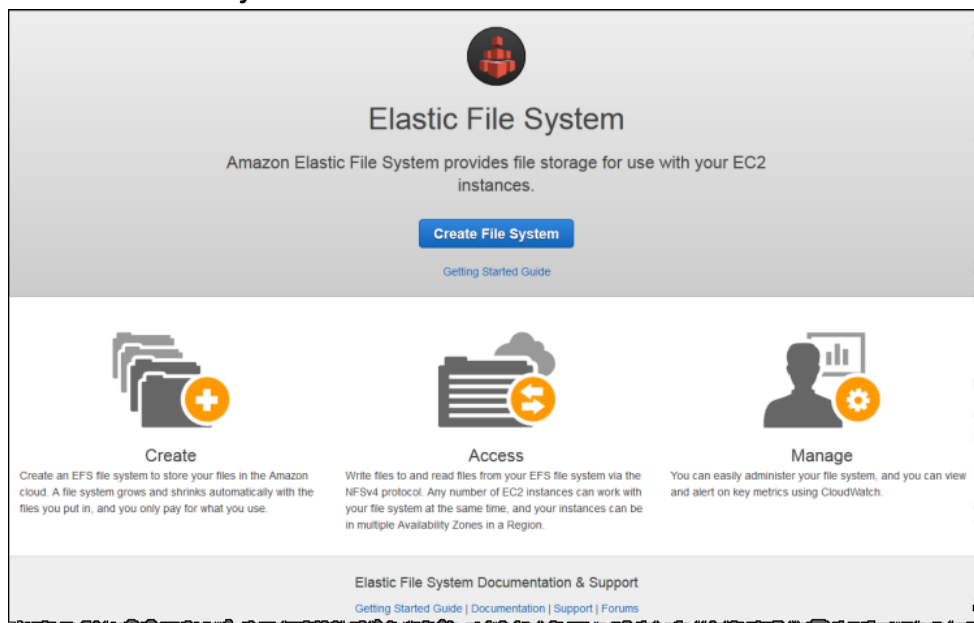
## Creating a Mount Target Using the Amazon EFS console

Perform the steps in the following procedure to create a mount target using the console. As you follow the console steps, you can also create one or more mount targets. You can create one mount target for each Availability Zone in your VPC.

### To create an Amazon EFS file system (console)

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**.



**Note**

The console shows the preceding page only if you don't already have any Amazon EFS file systems. If you have created file systems, the console shows a list of your file systems. On the list page, choose **Create File System**.

3. On the **Step 1: Configure File System Access** page, select the VPC and the Availability Zone in the VPC where you want the console to create one or more mount targets for the file system that you are creating. This VPC should be the same Amazon VPC in which you created your Amazon EC2 instance in the preceding section.

- a. Select a Amazon VPC from the **VPC** list.

**Warning**

If the Amazon VPC you want is not listed, verify the AWS Region in the global navigation in the Amazon EFS console.

- b. In the **Create Mount Targets** section, select all of the Availability Zones listed.

We recommend that you create mount targets in all Availability Zones. You can then mount your file system on Amazon EC2 instances created in any of the Amazon VPC subnets.

**Note**

You can access a file system on an Amazon EC2 instance in one Availability Zone by using a mount target created in another Availability Zone, but there are costs associated with cross-Availability Zone access.

For each Availability Zone, do the following:

- Choose a **Subnet** from the list where you want to create the mount target.

You can create one mount target in each Availability Zone. If you have multiple subnets in an Availability Zone where you launched your Amazon EC2 instance, you don't have to create mount target in the same subnet, it can be any subnet in the Availability Zone.

- Leave **IP Address** select to **Automatic**. Amazon EFS will select one of the available IP addresses for the mount target.



- Specify the **Security Group** you created specifically for the mount target, or the default security group for the default VPC. Both security groups will have the necessary inbound rule that allows inbound access from the EC2 instance security group.

Click in the **Security Group** box and the console will show you the available security groups. Here you can select a specific security group and remove the **Default** security group, or leave the default in place, depending on how you configured your Amazon EC2 instance.

The screenshot shows the 'Create File System' console page, specifically Step 1: Configure File System Access. The page is divided into two main sections: 'Configure File System Access' and 'Create Mount Targets'.

**Configure File System Access:** This section includes a VPC dropdown menu set to 'vpc-460fba23'. Below this, the 'Create Mount Targets' section explains that instances connect to a file system via mount targets and recommends creating a mount target in each of the VPC's availability zones.

**Create Mount Targets:** This section contains a table with columns for Availability Zone, Subnet, IP Address, and Security Group. Three availability zones are listed: eu-west-1a, eu-west-1b, and eu-west-1c. Each zone has a checkbox checked, a subnet dropdown menu, an IP Address dropdown menu set to 'Automatic', and a Security Group dropdown menu.

Availability Zone	Subnet	IP Address	Security Group
<input checked="" type="checkbox"/> eu-west-1a	subnet-85c2a39d	Automatic	x sg-2291ce47 - efs-getting-started-ml-sg
<input checked="" type="checkbox"/> eu-west-1b	subnet-8c41c8fb	Automatic	x sg-2291ce47 - efs-getting-started-ml-sg
<input checked="" type="checkbox"/> eu-west-1c	subnet-0dc36154	Automatic	x sg-2291ce47 - efs-getting-started-ml-sg

At the bottom right of the table, there are 'Cancel' and 'Next Step' buttons.

4. On the **Step 2: Configure optional settings** page, specify a value for the **Name** tag (**MyExampleFileSystem**). Also, add any additional tags that help describe and manage your file system, and choose your performance mode.

The console prepopulates the **Name** tag because Amazon EFS uses its value as the file system display name.

**Create file system**

Step 1: Configure file system access  
**Step 2: Configure optional settings**  
Step 3: Review and create

### Configure optional settings

#### Add tags

You can add tags to describe your file system. A tag consists of a case-sensitive key-value pair. (For example, you can define a tag with key-value pair with key = Corporate Department and value = Sales and Marketing.) At a minimum, we recommend a tag with key = Name.

Key	Value	Remove
Name	MyExampleFileSystem	
<input type="text" value="Add New Key"/>		

#### Choose performance mode

We recommend **General Purpose** performance mode for most file systems. **Max I/O** performance mode is optimized for applications where tens, hundreds, or thousands of EC2 instances are accessing the file system — it scales to higher levels of aggregate throughput and operations per second with a tradeoff of slightly higher latencies for file operations.

☒ General Purpose (default)  
☐ Max I/O

[Cancel](#) [Previous](#) [Next Step](#)

5. On the **Step 3: Review and Create** page, choose **Create File System**.

**Create file system**

Step 1: Configure file system access  
Step 2: Configure optional settings  
**Step 3: Review and create**

### Review and create

Review the configuration below before proceeding to create your file system.

#### File system access

VPC	Availability Zone	Subnet	IP address	Security groups
vpc- (default)	us-west-2a		Automatic	
	us-west-2b		Automatic	
	us-west-2c		Automatic	

#### Optional settings

Tags Name: MyExampleFileSystem

Performance mode: General Purpose (default)

[Cancel](#) [Previous](#) [Create File System](#)

6. The console shows the newly created file system on the **File Systems** page. Verify that all mount targets show the **Life Cycle State** as **Available**. It might take a few moments before the mount

targets become available (you can expand/collapse the file system in the EFS console to force it to refresh).

7. Under **File system access**, you'll see the file system's **DNS name**. Make a note of this DNS name. In the next section, you use the DNS name to mount the file system on the Amazon EC2 instance through the mount target. The Amazon EC2 instance on which you mount the file system can resolve the file system's DNS name to the mount target's IP address.

Now you are ready to mount the Amazon EFS file system on an Amazon EC2 instance.

## Creating a Mount Target using the AWS CLI

To create a mount target using AWS CLI, use the `create-mount-target` CLI command (corresponding operation is [CreateMountTarget](#) (p. 200)), as shown following.

```
$ 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 \
--profile adminuser
```

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

```
{
  "MountTargetId": "fsmt-f9a14450",
  "NetworkInterfaceId": "eni-3851ec4e",
  "FileSystemId": "fs-b6a0451f",
  "LifeCycleState": "available",
  "SubnetId": "subnet-b3983dc4",
  "OwnerId": "23124example",
  "IpAddress": "10.0.1.24"
}
```

You can also retrieve a list of mount targets created for a file system using the `describe-mount-targets` command (corresponding operation is [DescribeMountTargets](#) (p. 234)), as shown following.

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

For an example, see [Walkthrough: Create an Amazon EFS File System and Mount It on an Amazon EC2 Instance Using the AWS CLI](#) (p. 101).

## Creating Security Groups

### Note

The following section is specific to Amazon EC2 and discusses how to create security groups so you can use Secure Shell (SSH) to connect to any instances that have mounted Amazon EFS file systems. If you're not using SSH to connect to your Amazon EC2 instances, you can skip this section.

Both an Amazon EC2 instance and a mount target have associated security groups. These security groups act as a virtual firewall that controls the traffic between them. If you don't provide a security group when creating a mount target, Amazon EFS associates the default security group of the VPC with it.

Regardless, to enable traffic between an EC2 instance and a mount target (and thus the file system), you must configure the following rules in these security groups:

- The security groups you associate with a mount target must allow inbound access for the TCP protocol on the NFS port from all EC2 instances on which you want to mount the file system.
- Each EC2 instance that mounts the file system must have a security group that allows outbound access to the mount target on the NFS port.

For more information about security groups, see [Amazon EC2 Security Groups](#) in the *Amazon EC2 User Guide for Linux Instances*.

## Creating Security Groups Using the AWS Management Console

You can use the AWS Management Console to create security groups in your VPC. To connect your Amazon EFS file system to your Amazon EC2 instance, you'll need to create two security groups: one for your Amazon EC2 instance and another for your Amazon EFS mount target.

1. Create two security groups in your VPC. For instructions, see [Creating a Security Group](#) in the *Amazon VPC User Guide*.
2. In the VPC console, verify the default rules for these security groups. Both security groups should have only an outbound rule that allows traffic to leave.
3. You need to authorize additional access to the security groups as follows:
  - a. Add a rule to the EC2 security group to allow SSH access to the instance on port 22 as shown following. This is useful if you're planning on using an SSH client like PuTTY to connect to and administer your EC2 instance through a terminal interface. Optionally, you can restrict the **Source** address.

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere	Inbound SSH access, anywhere

[Add Rule](#)

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

[Cancel](#) [Save](#)

For instructions, see [Adding and Removing Rules](#) in the *Amazon VPC User Guide*.

- b. Add a rule to the mount target security group to allow inbound access from the EC2 security group, as shown following (where the EC2 security group is identified as the source):

Type	Protocol	Port Range	Source	Description
NFS	TCP	2049	Custom	e.g. SSH for Admin Desktop

[Add Rule](#)

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

[Cancel](#) [Save](#)

**Note**

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

4. Verify that both security groups now authorize inbound and outbound access as described in this section.

## Creating Security Groups Using the AWS CLI

For an example that shows how to create security groups using the AWS CLI, see [Step 1: Create Amazon EC2 Resources](#) (p. 103).

## Creating File System Policies

You can create a file system policy using the Amazon EFS console or using the AWS CLI. You can also create a file system policy programmatically using AWS SDKs or the EFS API directly. To learn more about file system policies and see examples, see [Using IAM to Control NFS Access to Amazon EFS](#) (p. 150).

### Creating a File System Policy (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 create a file system policy for.
4. For **Actions**, choose **Manage client access**. The **File system policy** page appears.
5. On the **Policy settings** tab, choose one or more of the preset policy statements available.

Or choose the **JSON** tab to configure your own policy using the JSON editor.

File system policy

A file system policy is an IAM resource policy that applies to all NFS clients connecting to this file system. You can use the JSON editor to create a more advanced policy, such as one that grants permissions to different users.

Policy settings JSON

Select a combination of policy statements and set your policy.

- ☐ Disable root access by default\*
- ☐ Enforce read-only access by default\*
- ☐ Enforce in-transit encryption for all clients

\* Identity-based policies can override these default permissions.

Set policy Save policy

- After you complete creating your policy, choose **Set policy**. If you chose one of the policy presets, the **JSON** tab displays the JSON formatted policy.

Elastic File System > File systems > Manage file system permissions

File system policy

A file system policy is an IAM resource policy that applies to all NFS clients connecting to this file system. You can use the JSON editor to create a more advanced policy, such as one that grants permissions to different users.

Policy settings JSON

```
1 {
2   "Version": "2012-10-17",
3   "Id": "efs-policy-wizard-8bdb0673-752a-4db7-bec1-61bc0377a9c7",
4   "Statement": [
5     {
6       "Sid": "efs-statement-5413d891-5d49-4a45-af4c-97fe863a432b",
7       "Effect": "Allow",
8       "Principal": {
9         "AWS": "*"
10      },
11      "Action": [
12        "elasticfilesystem:ClientMount",
13        "elasticfilesystem:ClientWrite"
14      ]
15    }
16  ]
17 }
```

The policy shown is the JSON that is generated if you chose the **Disable root access by default** policy.

7. Choose **Save policy** to save the file system policy.

## Creating a File System Policy (CLI)

In the following example, the `put-file-system-policy` CLI command creates a file system policy that allows all IAM principals read-only access to the EFS file system. The equivalent API command is [PutFileSystemPolicy](#) (p. 249).

```
aws efs put-file-system-policy --file-system-id fs-01234567 --policy '{
  "Version": "2012-10-17",
  "Id": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:ClientMount"
      ],
      "Principal": {
        "AWS": "*"
      }
    }
  ]
}'
```

```
{
  "FileSystemId": "fs-01234567",
  "Policy": "{
    \"Version\" : \"2012-10-17\",
    \"Id\" : \"1\",
    \"Statement\" : [
      {
        \"Sid\" : \"efs-statement-7c8d8687-1c94-4fdc-98b7-111122223333\",
        \"Effect\" : \"Allow\",
        \"Principal\" : {
          \"AWS\" : \"*\"
        },
        \"Action\" : [
          \"elasticfilesystem:ClientMount\"
        ],
        \"Resource\" : \"arn:aws:elasticfilesystem:us-east-2:111122223333:file-system/
fs-01234567\"
      }
    ]
  }
}
```

## Creating Access Points

You can create an EFS access point using the Amazon EFS console or the AWS CLI. You can also create access points programmatically using the AWS SDKs or the EFS API directly. For more information about EFS access points, see [Working with Amazon EFS Access Points](#) (p. 161).

The following procedures describe how to create an access point for an existing file system using the console and CLI. To create an access point while configuring a new file system, see [Step 1: Create Your Amazon EFS File System](#) (p. 12) in the *Getting Started* section.

## Creating an Access Point (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 create an access point for.
4. For **Actions**, choose **Manage client access**. The **Manage file system permissions** page appears.
5. For **Access points**, choose **+Add access point**. The **New access points** panel expands, displaying boxes where you can enter the access point configuration information.

Access points

You can create access points to provide applications access to your file system. Optionally, you can configure the POSIX identity and root directory for all connections to this access point. If you specify the owner for root directory, EFS will automatically create it with the ownership and permissions that you specify once a client connects to the access point. Once you create your file system, you can update its policy to apply to access points. [Learn more](#)

Access point ID	Name	User ID	Group ID	Directory	Lifecycle state
No access points found					

New access points

Name	Posix User	Directory	Owner
Name	User ID	Path	Owner User ID
	Group ID		Owner Group ID
	Secondary Group IDs		Permissions

+ Add access point

Save access points

6. Enter the following information:
  - (Optional) For **Name**, enter a name for the access point.
  - (Optional) Under **Posix User**, enter the numeric POSIX values for **User ID** and **Group ID** if you want to assign a user to the access point and override the identity information provided by the NFS client. A group ID is required if you specify a user ID. Optionally, also enter any secondary group IDs.
  - (Optional) For **Path**, enter a path for the access point root directory. If you don't enter one, the file system root directory is used. The path can have a maximum of four levels.
  - If you provide a value for **Path** that doesn't already exist on the file system, make sure to provide directory owner information under **Owner**, enter the numeric POSIX values for **Owner User ID**, **Owner Group ID**, and **Permissions**. EFS creates the directory using this information. For more information about root directories for access points, see [Enforcing a Root Directory with an Access Point](#) (p. 162).
7. Choose **Save access points** to save and create the access point. The new access point is added to the list. Choose the expand icon next to the access point ID to see the detail information.

Access points

You can create access points to provide applications access to your file system. Optionally, you can configure the POSIX identity and root directory for all connections to this access point. If you specify the owner for root directory, EFS will automatically create it with the ownership and permissions that you specify once a client connects to the access point. Once you create your file system, you can update its policy to apply to access points. [Learn more](#)

Access point ID	Name	User ID	Group ID	Directory	Lifecycle state
fsap-06344a875cf8d6fc9	access_point_example	1000	1000	/access_point_path1	Available

Posix User

User ID 1000

Group ID 1000

Secondary Group IDs

Root Directory

Path /access\_point\_path1

Owner User ID 1000

Owner Group ID 1000

Permissions 777



## Creating an Access Point (CLI)

In the following example, the `create-access-point` CLI command creates an access point for the file system. The equivalent API command is [CreateAccessPoint](#) (p. 188).

```
aws efs create-access-point --file-system-id fs-01234567 --client-token 010102020-3
{
  "ClientToken": "010102020-3",
  "Tags": [],
  "AccessPointId": "fsap-092e9f80b3fb5e6f3",
  "AccessPointArn": "arn:aws:elasticfilesystem:us-east-2:111122223333:access-point/
fsap-092e9f80b3fb5e6f3",
  "FileSystemId": "fs-01234567",
  "RootDirectory": {
    "Path": "/"
  },
  "OwnerId": "111122223333",
  "LifecycleState": "creating"
}
```

# Using File Systems in Amazon EFS

Amazon Elastic File System presents a standard file-system interface that supports full file-system access semantics. Using Network File System (NFS) version 4.1 (NFSv4.1), you can mount your Amazon EFS file system on any Amazon Elastic Compute Cloud (Amazon EC2) Linux-based instance. After your system is mounted, you can work with the files and directories just as you do with a local file system. For more information on mounting, see [Mounting EFS File Systems \(p. 61\)](#).

After you create a file system and mount it on your EC2 instance, to use your file system effectively you need to know about managing NFS-level permissions for users, groups, and related resources. When you first create your file system, there is only one root directory at /. By default, only the root user (UID 0) has read-write-execute permissions. For other users to modify the file system, the root user must explicitly grant them access. You use EFS access points to provision directories that are writable from a specific application. For more information, see [Working with Users, Groups, and Permissions at the Network File System \(NFS\) Level \(p. 158\)](#) and [Working with Amazon EFS Access Points \(p. 161\)](#).

## Related Topics

[Amazon EFS: How It Works \(p. 3\)](#)

[Getting Started \(p. 11\)](#)

[Walkthroughs \(p. 101\)](#)

# Using the amazon-efs-utils Tools

Following, you can find a description of amazon-efs-utils, an open-source collection of Amazon EFS tools.

## Topics

- [Overview \(p. 35\)](#)
- [Installing the amazon-efs-utils Package on Amazon Linux \(p. 36\)](#)
- [Installing the amazon-efs-utils Package on Other Linux Distributions \(p. 36\)](#)
- [Upgrading Stunnel \(p. 37\)](#)
- [EFS Mount Helper \(p. 39\)](#)

## Overview

The amazon-efs-utils package is an open-source collection of Amazon EFS tools. There's no additional cost to use amazon-efs-utils, and you can download these tools from GitHub here: <https://github.com/aws/efs-utils>. The amazon-efs-utils package is available in the Amazon Linux package repositories, and you can build and install the package on other Linux distributions.

The amazon-efs-utils package comes with a mount helper and tooling that makes it easier to perform encryption of data in transit for Amazon EFS. A *mount helper* is a program that you use when you mount a specific type of file system. We recommend that you use the mount helper included in amazon-efs-utils to mount your Amazon EFS file systems.

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

- NFS client (the nfs-utils package)
- Network relay (stunnel package, version 4.56 or later)
- Python (version 2.7 or later)
- OpenSSL 1.0.2 or newer

### Note

By default, when using the Amazon EFS mount helper with Transport Layer Security (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 Amazon EFS file system using TLS fails.

When you've installed the amazon-efs-utils package, to upgrade your system's version of stunnel, see [Upgrading Stunnel \(p. 37\)](#).

For issues with encryption, see [Troubleshooting Encryption \(p. 182\)](#).

The following Linux distributions support amazon-efs-utils:

- Amazon Linux 2
- Amazon Linux
- Red Hat Enterprise Linux (and derivatives such as CentOS) version 7 and newer

- Ubuntu 16.04 LTS and newer

In the following sections, you can find out how to install amazon-efs-utils on your Linux instances.

## Installing the amazon-efs-utils Package on Amazon Linux

The amazon-efs-utils package is available for installation in Amazon Linux and the Amazon Machine Images (AMIs) for Amazon Linux 2.

### Note

If you're using AWS Direct Connect, you can find installation instructions in [Walkthrough: Create and Mount a File System On-Premises with AWS Direct Connect and VPN](#) (p. 119).

### To install the amazon-efs-utils package

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

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

## Installing the amazon-efs-utils Package on Other Linux Distributions

If you don't want to get the amazon-efs-utils package from Amazon Linux or Amazon Linux 2 AMIs, the amazon-efs-utils package 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 Amazon Linux, Red Hat Linux, CentOS, and similar.
- **DEB** – This package type is supported by Ubuntu, Debian, and similar.

### To clone amazon-efs-utils from GitHub

1. Make sure that you've created an Amazon EC2 instance of the supported AMI type. For more information on how to do this, see [Step 1: Launch an Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. Access the terminal for your instance through Secure Shell (SSH), and log in with the appropriate user name. For more information on how to do this, see [Connecting to Your Linux Instance Using SSH](#) in the *Amazon EC2 User Guide for Linux Instances*.
3. If you haven't done so already, install git with the following commands.

```
sudo yum -y install git
```

4. 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
```

Because you need the bash command `make`, you can install it with the following command if your operating system doesn't already have it.

```
sudo yum -y install make
```

### 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 (for example `"/home/centos/efs-utils"`).
2. If you haven't done so already, install the rpm-builder package with the following command.

```
sudo yum -y install rpm-build
```

3. Build the package with the following command.

```
sudo make rpm
```

4. Install the amazon-efs-utils package with the following command.

```
sudo yum -y install ./build/amazon-efs-utils*rpm
```

### To build and install amazon-efs-utils as a 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. Install the binutils package, a dependency for building DEB packages.

```
sudo apt-get -y install binutils
```

3. Build the package with the following command.

```
./build-deb.sh
```

4. Install the package with the following command.

```
sudo apt-get -y install ./build/amazon-efs-utils*deb
```

## Upgrading Stunnel

Using encryption of data in transit with the Amazon EFS mount helper requires OpenSSL version 1.0.2 or newer, and a version of stunnel that supports both OCSP and certificate hostname checking. The Amazon 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 versions, mounting an Amazon EFS file system using TLS fails.

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

### To upgrade stunnel

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.

```
4. $ sudo yum install -y gcc openssl-devel tcp_wrappers-devel
```

5. 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://stunnel.org/downloads/latest-stunnel-version.tar.gz
```

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

```
7. $ cd latest-stunnel-version/
```

```
8. $ sudo ./configure
```

```
9. $ sudo make
```

10. The current amazon-efs-utils 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
```

```
11. $ sudo make install
```

#### 12. **Note**

The default CentOS shell is `csh`, which has different syntax than the `bash` shell. The following code first invokes `bash`, then runs.

```
bash
```

```
if [[ -f /bin/stunnel ]]; then  
  sudo mv /bin/stunnel /root  
fi
```

```
13. $ sudo ln -s /usr/local/bin/stunnel /bin/stunnel
```

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

## 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 \(p. 61\)](#).

## 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, 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 within your VPC. In the event that an EFS TLS certificate is revoked, Amazon will publish 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
```

## EFS Mount Helper

The Amazon EFS mount helper simplifies mounting your file systems. It includes the Amazon EFS recommended mount options by default. Additionally, the mount helper has built-in logging for troubleshooting purposes. If you encounter an issue with your Amazon EFS file system, you can share these logs with AWS Support.

## How It Works

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 Amazon EFS file system at instance boot time automatically by using entries in the `/etc/fstab` configuration file.

### 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](#) (p. 178).

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, and a supervisor process called `amazon-efs-mount-watchdog`. Stunnel is a multipurpose network relay that is open-source. 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 [Data Encryption in EFS](#) (p. 135).

## Using the EFS Mount Helper

The mount helper helps you mount your EFS file systems on your Linux EC2 instances. For more information, see [Mounting EFS File Systems](#) (p. 61).

## Getting Support Logs

The mount helper has built-in logging for your Amazon 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` for systems with the mount helper installed. These logs are for the mount helper, the stunnel process itself, and for the `amazon-efs-mount-watchdog` process that monitors the stunnel process.

### Note

The watchdog process ensures that each mount's stunnel process is running, and stops the stunnel when the Amazon EFS file system is unmounted. If for some reason a stunnel process is terminated unexpectedly, the watchdog process restarts it.

You can change the configuration of your logs in `/etc/amazon/efs/efs-utils.conf`. However, doing so requires unmounting and then remounting the file system with the mount helper for the changes to take effect. 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.

## Using amazon-efs-utils with AWS Direct Connect and VPN

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. Using `amazon-efs-utils` also makes mounting simpler with the mount helper and allows you to enable encryption of data in transit. To see how to use `amazon-efs-utils` with AWS Direct Connect to mount Amazon EFS file systems onto on-premises Linux clients, see [Walkthrough: Create and Mount a File System On-Premises with AWS Direct Connect and VPN](#) (p. 119).

## Related Topics

For more information on the Amazon EFS mount helper, see these related topics:

- [Data Encryption in EFS](#) (p. 135)
- [Mounting EFS File Systems](#) (p. 61)



# Managing Amazon EFS File Systems

File system management tasks refer to creating and deleting file systems, managing tags, and managing network accessibility of an existing file system. Managing network accessibility is about creating and managing mount targets.

You can perform these file system management tasks using the Amazon EFS console, AWS Command Line Interface (AWS CLI), or programmatically, as discussed in the following sections.

## Topics

- [Managing File System Network Accessibility \(p. 41\)](#)
- [Managing File System Tags \(p. 48\)](#)
- [EFS Storage Classes \(p. 50\)](#)
- [EFS Lifecycle Management \(p. 51\)](#)
- [Metering: How Amazon EFS Reports File System and Object Sizes \(p. 55\)](#)
- [Managing Amazon EFS File System Costs Using AWS Budgets \(p. 57\)](#)
- [Deleting an Amazon EFS File System \(p. 58\)](#)
- [Managing Access to Encrypted File Systems \(p. 59\)](#)

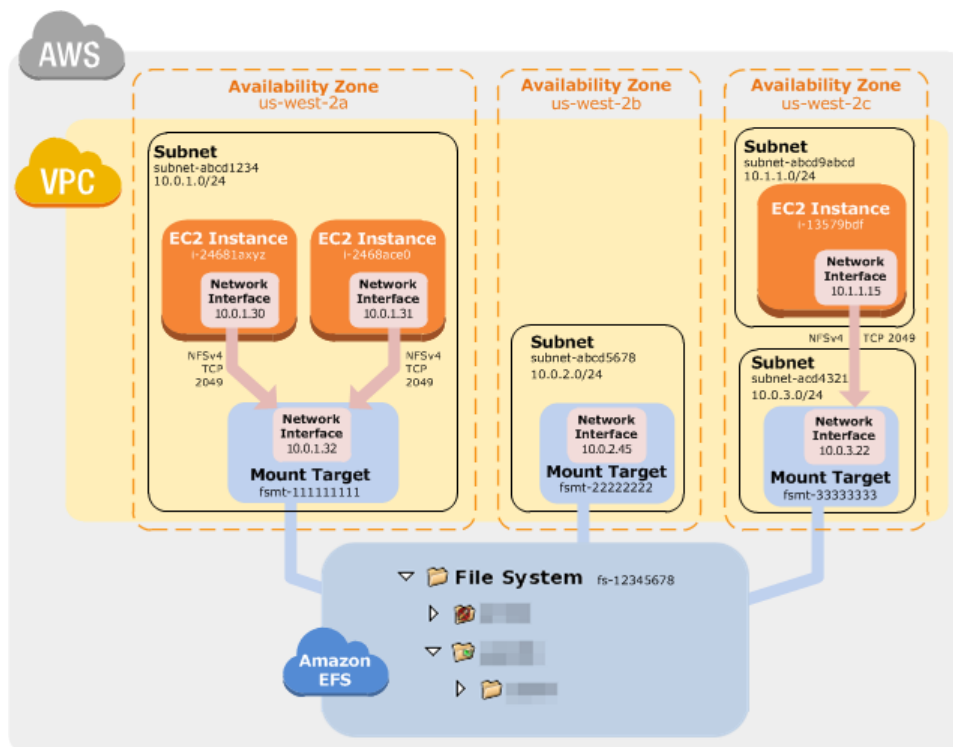
If you are new to Amazon EFS, we recommend that you try the following exercises that provide you with first-hand end-to-end experience using an Amazon EFS file system:

- [Getting Started \(p. 11\)](#) – This exercise provides a console-based, end-to-end setup in which you create a file system, mount it on an Amazon EC2 instance, and test the setup. The console takes care of many things for you and thus helps you quickly set up the end-to-end experience.
- [Walkthrough: Create an Amazon EFS File System and Mount It on an Amazon EC2 Instance Using the AWS CLI \(p. 101\)](#) – This walkthrough is similar to the Getting Started exercise, but it uses the AWS CLI to perform most of the tasks. Because the CLI commands closely map to the Amazon EFS API, the walkthrough can help you familiarize yourself with the Amazon EFS API.

## Managing File System Network Accessibility

You mount your file system on an EC2 instance in your virtual private cloud (VPC) using a mount target that you create for the file system. Managing file system network accessibility refers to managing the mount targets.

The following illustration shows how EC2 instances in a VPC access an Amazon EFS file system using a mount target.



The illustration shows three EC2 instances launched in different VPC subnets accessing an Amazon 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.

Managing mount targets refers to these activities:

- **Creating and deleting mount targets in a VPC** – At a minimum, you should create a mount target in each Availability Zone from which you want to access the file system.

#### Note

We recommend that you create mount targets in all the Availability Zones. If you do, you can easily mount the file system on EC2 instances that you might launch in any of the Availability Zones.

If 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.

You can use a file system only in one VPC at a time. That is, 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, first delete the mount targets from the current VPC. Then create new mount targets in another VPC.

- **Updating the mount target configuration** – When you create a mount target, you associate security groups with the mount target. A security group acts as a virtual firewall that controls the traffic to and from the mount target. You can add inbound rules to control access to the mount target, and thus the file system. After creating a mount target, you might want to modify the security groups assigned to them.

Each mount target also has an IP address. When you create a mount target, you can choose an IP address from the subnet where you are placing the mount target. If you omit a value, Amazon EFS selects an unused IP address from that subnet.

There is no Amazon EFS operation to change the IP address after creating a mount target. Thus, you can't change the IP address programmatically or by using the AWS CLI. But the console enables you to change the IP address. Behind the scenes, the console deletes the mount target and creates the mount target again.

**Warning**

If you change the IP address of a mount target, you break any existing file system mounts and you need to remount the file system.

None of the configuration changes to file system network accessibility affects the file system itself. Your file system and data remain.

The following sections provide information about managing network accessibility of your file system.

**Topics**

- [Creating or Deleting Mount Targets in a VPC \(p. 43\)](#)
- [Changing the VPC for Your Mount Target \(p. 45\)](#)
- [Updating the Mount Target Configuration \(p. 47\)](#)

## Creating or Deleting Mount Targets in a VPC

To access an Amazon EFS file system in a VPC, you need mount targets. For an Amazon EFS file system, the following is true:

- You can create one mount target in each Availability Zone.
- 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.

**Note**

We recommend that you create a mount target in each of the Availability Zones. 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](#). In addition, by always using a mount target local to the instance's Availability Zone, you eliminate a partial failure scenario. If the mount target's zone goes down, you can't access your file system through that mount target.

For more information about the operation, see [CreateMountTarget \(p. 200\)](#).

You can delete mount targets. A mount target deletion forcibly breaks any mounts of the file system via that mount target, which might disrupt instances or applications using those mounts. For more information, see [DeleteMountTarget \(p. 216\)](#).

## Using the Console

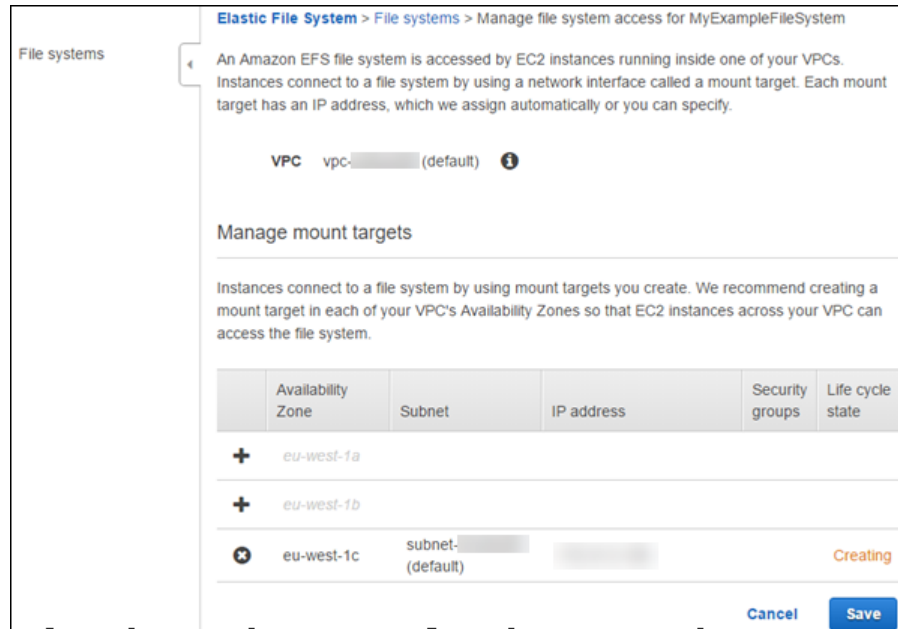
Use the following procedure to create new mount targets or delete or update existing mount targets using the AWS Management Console.

**To create new mount targets or update or delete existing ones (console)**

1. In the Amazon EFS console, choose the file system, and for **Actions** choose **Manage File System Access**.

The console displays the **Manage File System Access** page with a list of file system mount targets you have created in the selected VPC. The console shows a list of Availability Zones and mount target information, if there is a mount target in that Availability Zone.

The console shows that the file system has one mount target in the **eu-west-2c** Availability Zone, as shown following:



2. To create new mount targets
  - a. Click on the left side in the specific **Availability Zone** row.
  - b. If the Availability Zone has multiple subnets, select a subnet from the **Subnet** list.
  - c. Amazon EFS automatically selects an available IP address, or you can provide another IP address explicitly.
  - d. Choose a **Security Group** from the list.

For more information about security groups, see [Amazon EC2 Security Groups](#) in the *Amazon EC2 User Guide for Linux Instances*.
3. To delete a mount target, choose the **X** next to the Availability Zone from which you want to remove a mount target.

## Using the AWS CLI

To create a mount target, use the `create-mount-target` AWS CLI command (corresponding operation is `CreateMountTarget` (p. 200)), as shown following:

```
$ aws efs create-mount-target \
--file-system-id file-system-ID (for which to create the mount target) \
--subnet-id vpc-subnet-ID (in which to create mount target) \
--security-group security-group IDs (to associate with the mount target) \
--region aws-region (for example, us-west-2) \
--profile adminuser
```

The AWS Region (the `region` parameter) must be the VPC region.

You can get a list of mount targets created for a file system using the `describe-mount-target` AWS CLI command (corresponding operation is [DescribeMountTargets](#) (p. 234)), as shown following:

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

Here's a sample response:

```
{
  "MountTargets": [
    {
      "MountTargetId": "fsmt-52a643fb",
      "NetworkInterfaceId": "eni-f11e8395",
      "FileSystemId": "fs-6fa144c6",
      "LifeCycleState": "available",
      "SubnetId": "subnet-15d45170",
      "OwnerId": "23124example",
      "IpAddress": "10.0.2.99"
    },
    {
      "MountTargetId": "fsmt-55a643fc",
      "NetworkInterfaceId": "eni-14a6ae4d",
      "FileSystemId": "fs-6fa144c6",
      "LifeCycleState": "available",
      "SubnetId": "subnet-0b05fc52",
      "OwnerId": "23124example",
      "IpAddress": "10.0.19.174"
    }
  ]
}
```

To delete an existing mount target, use the `delete-mount-target` AWS CLI command (corresponding operation is [DeleteMountTarget](#) (p. 216)), as shown following:

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

## Changing the VPC for Your Mount Target

You can use an Amazon 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 Amazon 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 Internet Protocol

version 6 (IPv6) addresses. For more information on how Amazon EFS works with VPC peering, see [Mounting EFS File Systems from Another Account or VPC \(p. 68\)](#).

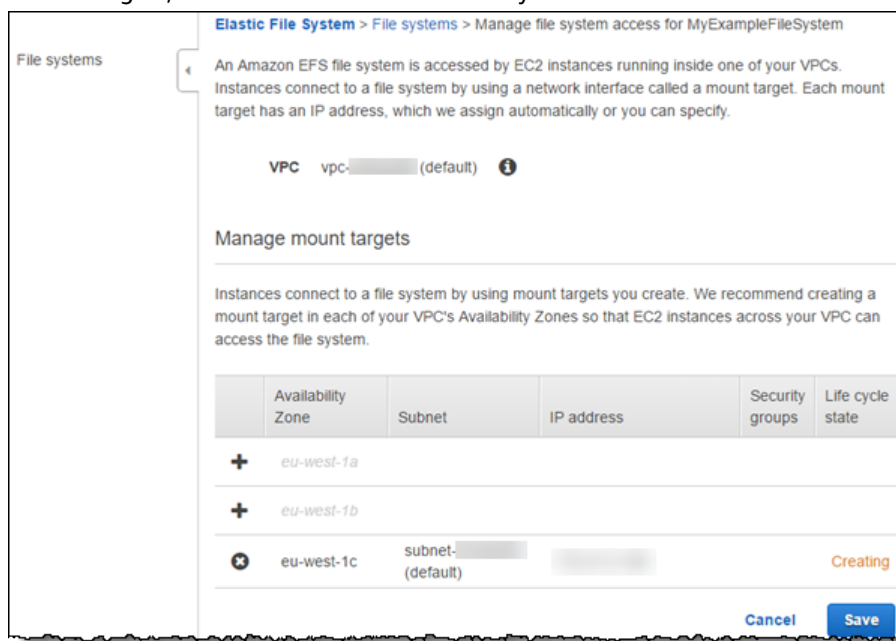
To access the file system from EC2 instances in another VPC, you must first delete the current mount targets and then create new mount targets, as described following.

## Using the Console

### To delete mount targets and create new ones (console)

1. In the Amazon EFS console, choose the file system, and for **Actions**, choose **Manage File System Access**.

The console displays the **Manage File System Access** page with a list of mount targets that you created for the file system in a VPC. The following illustration shows a file system that has three mount targets, one in each of three Availability Zones.



2. Choose another VPC for **VPC** to choose the VPC.

The console clears all of the mount target information and lists only the Availability Zone.

3. Create mount targets in one or more Availability Zones as follows:
  - a. If the Availability Zone has multiple subnets, choose a subnet for **Subnet**.
  - b. Amazon EFS automatically selects an available IP address, or you can provide another IP address explicitly.
  - c. Choose the security groups that you want to associate. For inter-region VPC peering, the security groups that you choose need to have a rule that allows inbound traffic over NFS (port 2049) from your other VPC or VPCs.

For information about security groups, see [Amazon EC2 Security Groups](#) in the *Amazon EC2 User Guide for Linux Instances*.

4. Choose **Save**.

The console first deletes the mount targets from the previous VPC and then creates new mount targets in the new VPC that you selected.

## Using the CLI

To use a file system in another VPC, first delete any mount targets that you previously created in a VPC. Then create new mount targets in another VPC. For example AWS CLI commands, see [Creating or Deleting Mount Targets in a VPC](#).

## Updating the Mount Target Configuration

After you create a mount target for your file system, you might want to update security groups that are in effect. You can't change the IP address of an existing mount target. To change an IP address, delete the mount target and create a new one with the new address. Deleting a mount target breaks any existing file system mounts.

## Modifying a Security Group

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 [Amazon EC2 Security Groups](#) in the *Amazon EC2 User Guide for Linux Instances*.

## Using the Console

### To modify an EFS security group

1. In the Amazon EFS console, choose the file system and for **Actions**, choose **Manage File System Access**.

The console displays the **Manage File System Access** page with a list of Availability Zones and mount target information, if there is a mount target in the Availability Zone.

Elastic File System > File systems > Manage file system access for MyExampleFileSystem

An Amazon EFS file system is accessed by EC2 instances running inside one of your VPCs. Instances connect to a file system by using a network interface called a mount target. Each mount target has an IP address, which we assign automatically or you can specify.

VPC vpc-... (default) ⓘ

### Manage mount targets

Instances connect to a file system by using mount targets you create. We recommend creating a mount target in each of your VPC's Availability Zones so that EC2 instances across your VPC can access the file system.

	Availability Zone	Subnet	IP address	Security groups	Life cycle state
+	eu-west-1a				
+	eu-west-1b				
*	eu-west-1c	subnet-... (default)			Creating

Cancel Save

2. In the **Security Group** column, you can add or remove security groups. Choose **X** to remove an existing security group. Choose the **Security Group** box to choose from other available security groups.

If you remove all security groups, Amazon EFS assigns the VPC's default security group.

## Using the CLI

To modify security groups that are in effect for a mount target, use the `modify-mount-target-security-group` AWS CLI command (corresponding operation is [ModifyMountTargetSecurityGroups](#) (p. 246)) 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
```

# Managing File System Tags

You can include file system tags when you create your Amazon EFS file system. You can also create new tags, update values of existing tags, or delete tags associated with a file system anytime after it is created.

## Using the Console

The console lists existing tags associated with a file system. You can add new tags, change values of existing tags, or delete existing tags.

### To add tags, change tag values, or delete tags (console)

1. Open the Amazon Elastic File System console at <https://console.aws.amazon.com/efs/>.
2. Choose the file system that you want to manage tags for.
3. Choose **Action** and then choose **Manage Tags**.
4. On the **Manage Tags** page, add new tags, and update or delete existing tags. For each new tag, provide a **Key** and its **Value**.
5. Choose **Save**.

## Using the CLI

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

CLI Command	Description	Equivalent API Operation
<code>create-tags</code>	Add new tags or update existing tags	<a href="#">CreateTags</a> (p. 207)
<code>describe-tags</code>	Retrieve existing tags	<a href="#">DescribeTags</a> (p. 241)
<code>delete-tags</code>	Delete existing tags	<a href="#">DeleteTags</a> (p. 219)



### To create file system tags (AWS CLI)

- Create tags by using the Amazon EFS `create-tags` CLI command (the corresponding API operation is [CreateTags](#) (p. 207)). The following example command creates a new Department tag that has a value of Business Intelligence to the file system.

```
$ aws efs create-tags \  
--file-system-id File-System-ID \  
--tags Key=Department,Value="Business Intelligence" \  
--region aws-region \  
--profile adminuser
```

If the command is successful, the AWS CLI doesn't provide a response.

### To retrieve all tags associated with a file system

- Retrieve a list of tags associated with a file system by using the `describe-tags` CLI command (the corresponding API operation is [DescribeTags](#) (p. 241)), as shown following.

```
$ aws efs describe-tags \  
--file-system-id File-System-ID \  
--region aws-region \  
--profile adminuser
```

Amazon EFS returns these descriptions as JSON. The following is an example of tags returned by the `DescribeTags` operation. It shows a file system as having three tags.

```
{  
  "Tags": [  
    {  
      "Key": "Name",  
      "Value": "Test File System"  
    },  
    {  
      "Key": "developer",  
      "Value": "rhoward"  
    },  
    {  
      "Key": "Department",  
      "Value": "Business Intelligence"  
    }  
  ]  
}
```

### To delete one or more tags associated with a file system

- Delete a file system's tags using the `delete-tags` CLI command (the corresponding API operation is [DeleteTags](#) (p. 219)). The following command removes the tag keys `test1` and `test2` from the tag list of the specified file system. You specify the key of each tag that you want to delete in the request.

```
$ aws efs \  
delete-tags \  
--file-system-id fs-c5a1446c \  
--tag-keys "test1" "test2" \  
--region us-west-2 \  
--profile adminuser
```

If the command is successful, the AWS CLI doesn't provide a response.

## Transferring Data into Amazon EFS

We recommend using AWS DataSync to transfer data into Amazon EFS. 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 system data and metadata, such as ownership, time stamps, and access permissions.

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.

To simplify transferring files between two EFS file systems using DataSync, you can use the [AWS DataSync In-Cloud QuickStart and Scheduler](#).

For more information, see [Getting Started with Amazon Elastic File System \(p. 11\)](#) and the [AWS DataSync User Guide](#).

## EFS Storage Classes

Amazon EFS file systems have two storage classes available:

- **Infrequent Access** – The Infrequent Access (IA) storage class is a lower-cost storage class that's designed for storing long-lived, infrequently accessed files cost-effectively.
- **Standard** – The Standard storage class is used to store frequently accessed files.

The EFS IA storage class reduces storage costs for files that are not accessed every day. It does this without sacrificing the high availability, high durability, elasticity, and POSIX file system access that EFS provides. We recommend EFS IA storage if you need your full dataset to be readily accessible and want to automatically save on storage costs for files that are less frequently accessed. Examples include keeping files accessible to satisfy audit requirements, performing historical analysis, or performing backup and recovery.

EFS IA storage is compatible with all EFS features, and is available in all AWS Regions where Amazon EFS is available.

EFS IA billing is based on the amount of data stored in IA and for data access used to read files that are stored in IA. You incur access charges when files in IA storage are read, and when files are transitioned to IA storage from Standard storage. To view how much data you have in each storage class, see [Pricing and Billing \(p. 52\)](#). For storage pricing information, see [Amazon EFS Pricing](#).

First-byte latency when reading from or writing to the IA storage class is higher than that for the Standard storage class. For file systems using Bursting Throughput, the allowed throughput is determined based on the amount of the data stored in the Standard storage class only. For file systems using Provisioned Throughput mode, you're billed for the throughput provisioned above what you are provided based on the amount of data that is in the Standard storage class. For more information on EFS performance, see [Throughput Modes \(p. 92\)](#).

## Using Storage Classes

To use the IA storage class, enable the EFS lifecycle management feature. When enabled, lifecycle management automates moving files from Standard storage to IA storage. To learn more, see [EFS Lifecycle Management \(p. 51\)](#).

## EFS Lifecycle Management

Amazon EFS lifecycle management automatically manages cost-effective file storage for your file systems. When enabled, lifecycle management migrates files that have not been accessed for a set period of time to the Infrequent Access (IA) storage class. You define that period of time by using a *lifecycle policy*.

After lifecycle management moves a file into the IA storage class, the file remains there indefinitely. Amazon EFS lifecycle management uses an internal timer to track when a file was last accessed. It doesn't use the POSIX file system attributes that are publicly viewable. Whenever a file in Standard storage is written to or read from, the lifecycle management timer is reset.

Metadata operations, such as listing the contents of a directory, don't count as file access. During the process of transitioning a file's content to IA storage, the file is stored in the Standard storage class and billed at the Standard storage rate.

Lifecycle management applies to all files in the file system.

## Using a Lifecycle Policy

You define when EFS transitions files to the IA storage class by setting a lifecycle policy. A file system has one lifecycle policy that applies to the entire file system. If a file is not accessed for the period of time defined by the lifecycle policy that you choose, Amazon EFS transitions the file to the IA storage class. You can specify one of four lifecycle policies for your Amazon EFS file system, as follows:

- AFTER\_7\_DAYS
- AFTER\_14\_DAYS
- AFTER\_30\_DAYS
- AFTER\_60\_DAYS
- AFTER\_90\_DAYS

To learn more about enabling lifecycle management on your Amazon EFS file system and setting a lifecycle policy, see [Enabling Lifecycle Management \(p. 52\)](#).

## File System Operations for Lifecycle Management

File system operations for lifecycle management, which transition files to IA storage, have a lower priority than operations for EFS file system workloads. The time required to transition files to the IA storage class varies depending on the file size and file system workload.

File metadata, including file names, ownership information, and file system directory structure, is always stored in Standard storage to ensure consistent metadata performance. All write operations to files in IA storage are first written to Standard storage, then transitioned to IA storage. Files smaller than 128 KB aren't eligible for lifecycle management and are always stored in the Standard class.

## Pricing and Billing

You are billed for the amount of data in each storage class. You are also billed for data access when files in IA storage are read and when files are transitioned to IA storage from Standard storage. The AWS bill displays the capacity for each storage class and the metered access against the IA storage class. To learn more, see [Amazon EFS Pricing](#).

### 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 and EFS lifecycle management, see [EFS Storage Classes \(p. 99\)](#)

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.

```
$ aws efs describe-file-systems \
--region us-west-2 \
--profile adminuser
```

In the response, `ValueInIA` displays the last metered size in IA storage. `ValueInStandard` displays the last metered size in Standard storage. Added together, they equal the size of the entire file system, displayed by `Value`.

```
{
  "FileSystems":[
    {
      "OwnerId":"251839141158",
      "CreationToken":"MyFileSystem1",
      "FileSystemId":"fs-47a2c22e",
      "PerformanceMode" : "generalPurpose",
      "CreationTime": 1403301078,
      "LifecycleState":"created",
      "NumberOfMountTargets":1,
      "SizeInBytes":{"
        "Value": 29313417216,
        "ValueInIA": 675432,
        "ValueInStandard": 29312741784
      },
      "ThroughputMode": "bursting"
    }
  ]
}
```

For additional ways to view and measure disk usage, see [Metering Amazon EFS File System Objects \(p. 55\)](#).

## Enabling Lifecycle Management

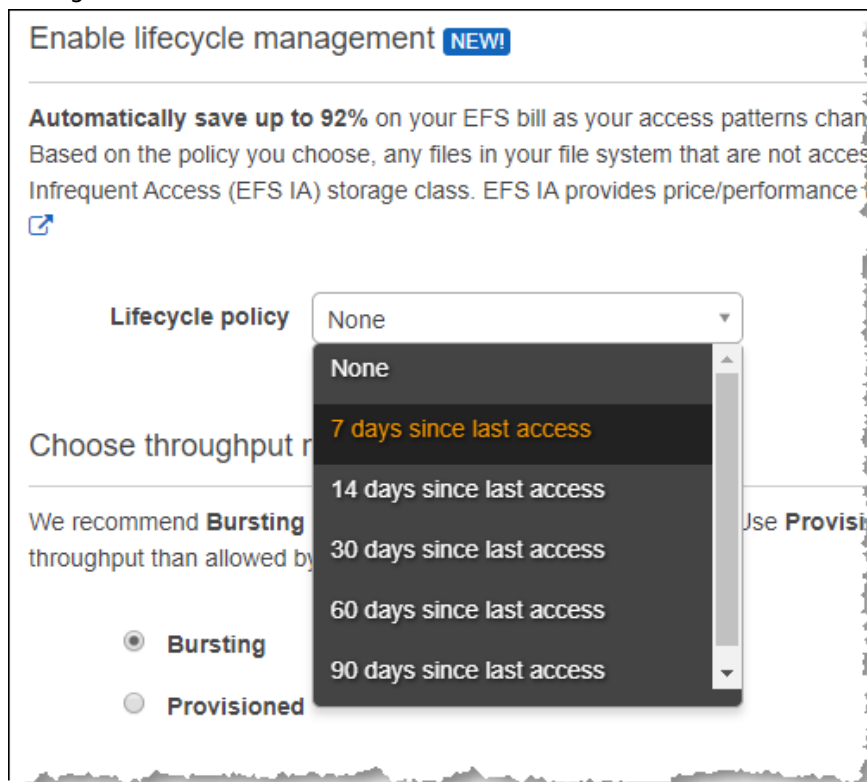
You can enable lifecycle management when creating a new Amazon EFS file system by using the Amazon EFS console. You can enable lifecycle management for an existing file system by using the EFS console, CLI, or API. Following, find procedures for enabling and stopping EFS lifecycle management.

### Enable Lifecycle Management When Creating a New File System (Console)

You can use the AWS Management Console to enable lifecycle management when creating a new Amazon EFS file system as follows.

### To enable lifecycle management for a new EFS file system

1. Open the Amazon EFS Management Console at <https://console.aws.amazon.com/efs/>.
2. Choose **Create file system**.
3. Complete **Step 1: Configure file system access** by choosing a VPC and creating mount targets. For more information, see [Step 1: Create Your Amazon EFS File System \(p. 12\)](#).
4. Choose **Next Step**.
5. On the **Configure optional settings** page, add any tags that you want.
6. Choose a value for **Lifecycle policy** in the **Enable lifecycle management** section to enable lifecycle management.



7. Configure any additional optional settings, and choose **Next Step**.
8. On the **Review and create** page, review the **Lifecycle policy** setting that you chose in the **Optional settings** section.
9. To make any changes, choose **Previous**. Otherwise, choose **Create File System** to create your file system with lifecycle management enabled.

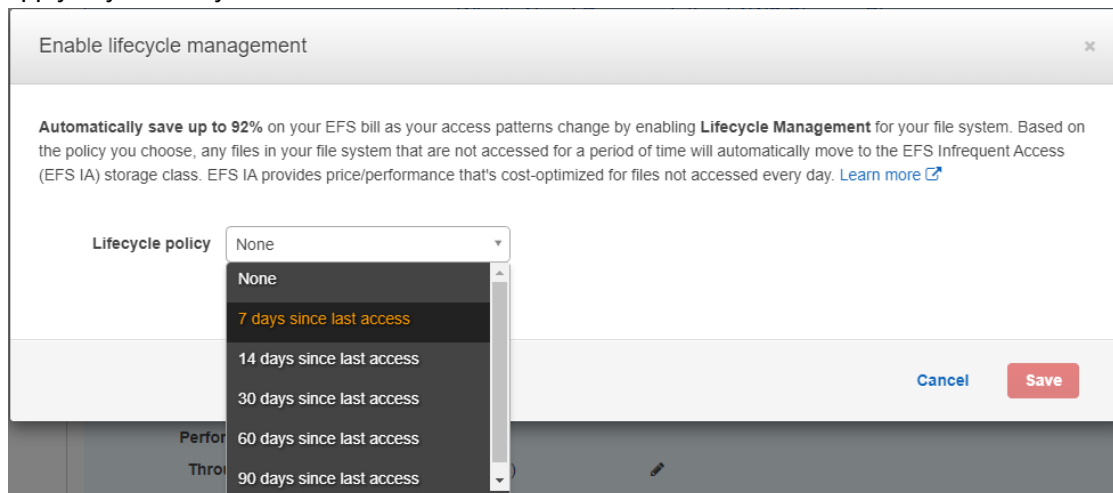
### Enable Lifecycle Management for an Existing File System (Console)

You can use the AWS Management Console to enable lifecycle management for an existing file system.

#### To enable lifecycle management on an existing file system (console)

1. Open the Amazon EFS Management Console at <https://console.aws.amazon.com/efs/>.
2. Choose **File Systems**.
3. Choose the file system for which you want to enable lifecycle management.
4. Locate **Lifecycle policy** in **Other details**. Choose **edit** to change the setting.

5. In the **Enable lifecycle management** panel, choose the policy for **Lifecycle policy** that you want to apply to your file system.



Enable lifecycle management

Automatically save up to 92% on your EFS bill as your access patterns change by enabling **Lifecycle Management** for your file system. Based on the policy you choose, any files in your file system that are not accessed for a period of time will automatically move to the EFS Infrequent Access (EFS IA) storage class. EFS IA provides price/performance that's cost-optimized for files not accessed every day. [Learn more](#)

Lifecycle policy: None

- None
- 7 days since last access
- 14 days since last access
- 30 days since last access
- 60 days since last access
- 90 days since last access

Cancel Save

6. Choose **Save** to save the setting.

After the setting is saved, **Lifecycle policy** is enabled in the **Other details** section and shows the policy that you chose.

## Enable Lifecycle Management for an Existing File System (AWS CLI)

You can use the AWS CLI to enable lifecycle management on an existing file system, as follows. You can't use the CLI or API to enable lifecycle management when creating a new file system.

### To enable lifecycle management on an existing file system (CLI)

- Run the `put-lifecycle-configuration` command specifying the file system ID of the file system for which you are enabling lifecycle management.

```
$ aws efs put-lifecycle-configuration \
--file-system-id File-System-ID \
--lifecycle-policies TransitionToIA=AFTER_60_DAYS \
--region us-west-2 \
--profile adminuser
```

You get the following response.

```
{
  "LifecyclePolicies": [
    {
      "TransitionToIA": "AFTER_60_DAYS"
    }
  ]
}
```

## Stopping EFS Lifecycle Management

You can stop EFS lifecycle management at any time using the EFS Management Console, the CLI, or the EFS API. When you stop lifecycle management, your files are no longer moved into the IA storage class.

Any files currently in IA storage remain there. You can move files from IA to Standard storage by copying them to another location on your file system.

## Stopping Lifecycle Management for an Existing File System (Console)

You can stop EFS lifecycle management by using the console as follows.

### To stop lifecycle management for an existing file system (console)

1. Open the Amazon EFS Management Console at <https://console.aws.amazon.com/efs/>.
2. Choose **File Systems**.
3. Choose the file system for which to stop lifecycle management.
4. Locate **Lifecycle policy** in **Other details**. Choose **edit** to change the setting.
5. In the **Enable lifecycle management** panel, choose **None** for **Lifecycle policy**.
6. Choose **Save** to save the setting.

In **Other details**, **Lifecycle policy** is now set to **None**.

## Stopping Lifecycle Management for an Existing File System (AWS CLI)

You can stop EFS lifecycle management by using the AWS CLI as follows.

### 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.

```
$ 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": []
}
```

# Metering: How Amazon EFS Reports File System and Object Sizes

In the following section, find how Amazon EFS reports file system sizes and sizes of objects within a file system.

## Metering Amazon EFS File System Objects

Objects that you can view in an Amazon EFS 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, with the `-k` option to return the output in kilobytes.

```
$ du -k file
4      file
```

Following is an example of 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
```

## Metering an Amazon EFS File System

The metered size of an Amazon EFS file system includes the sum of the sizes of all current objects in the standard and IA storage classes. The size of each object is calculated from a representative sampling that represents the size of the object during the metered hour. An example is the hour from 8 AM to 9 AM.

For example, an empty file contributes 6 KiB (2 KiB metadata + 4 KiB data) to the metered size of its 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 this metered size for an Amazon EFS file system in the following ways:

- Call the [DescribeFileSystems operation \(p. 227\)](#) using one the SDKs, HTTP, or the AWS CLI.



- View the **File Systems** table, for each file system listed in the AWS Management Console.
- Run the `df` command in Linux at the terminal prompt of an EC2 instance.

Use the `df` command and not the `du` command. Don't use the `du` command on the root of the file system for storage metering purposes. The results don't provide full data.

**Note**

The metered size of the Standard storage class is also used to determine your I/O throughput baseline and burst rates. For more information, see [Throughput Scaling with Bursting Mode](#) (p. 93).

## Metering for Infrequent Access

Infrequent Access (IA) storage is metered in 4 KiB increments. IA file metadata (2 KiB per file) is always stored and metered in the Standard storage class. Data access for IA storage is metered in 1 MiB increments.

# Managing Amazon EFS File System Costs Using AWS Budgets

You can plan and manage your Amazon EFS file system costs by using AWS Budgets.

You can work with AWS Budgets from the AWS Billing and Cost Management console. To use AWS Budgets, you create a monthly cost budget for your EFS file systems. You can set up your budget to notify you if your costs are forecast to exceed your budgeted amount, and then make adjustments to maintain your budget as needed.

There are costs associated with using AWS Budgets. For regular AWS accounts, your first two budgets are free. For more information about AWS Budgets, including costs, see [Managing Your Costs with Budgets](#) in the *AWS Billing and Cost Management User Guide*.

You can set custom budgets for your EFS costs and usage at the account, AWS Region, service, or tag level by using budget parameters. In the following section, you can find a high-level description of how to set up a cost budget on an EFS file system with AWS Budgets. You do so by using cost allocation tags.

## Prerequisites

To perform the procedures referenced in the following sections, make sure that you have the following:

- An EFS file system
- An AWS Identity and Access Management (IAM) policy with the following permissions:
  - Access to the Billing and Cost Management console.
  - Ability to perform the `elasticfilesystem:CreateTags` and `elasticfilesystem:DescribeTags` actions.

## Creating a Monthly Cost Budget for an EFS File System

Creating a monthly cost budget for your Amazon EFS file system using tags is a three-step process.

### To create a monthly cost budget for your EFS file system using tags

1. Create a tag to use to identify the file system that you want to track costs for. To learn how, see [Managing File System Tags \(p. 48\)](#).
2. In the Billing and Cost Management console, activate the tag as a cost allocation tag. For a detailed procedure, see [Activating User-Defined Cost Allocation Tags](#) in the *AWS Billing and Cost Management User Guide*.
3. In the Billing and Cost Management console, under **Budgets**, create a monthly cost budget in AWS Budgets. For a detailed procedure, see [Creating a Cost Budget](#) in the *AWS Billing and Cost Management User Guide*.

After you create your EFS monthly cost budget, you can view it in the **Budgets** dashboard, which displays the following budget data:

- Your current costs and usage incurred for a budget during the budget period.
- Your budgeted costs for the budget period.
- Your forecast costs for the budget period.
- A percentage that shows your costs compared to your budgeted amount.
- A percentage that shows your forecast costs compared to your budgeted amount.

For more information about viewing your EFS cost budget, see [Viewing Your Budgets](#) in the *AWS Billing and Cost Management User Guide*.

## Deleting an Amazon EFS File System

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.

### 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.
3. Choose **Action** and then choose **Delete File System**.
4. In **Permanently Delete File System** confirmation box, type the file system ID and then choose **Delete File System**.

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

## Using the CLI

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

For example AWS CLI commands, see [Step 4: Clean Up \(p. 111\)](#).

## Related Topics

[Managing Amazon EFS File Systems \(p. 41\)](#)

# 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 you need to perform is only related 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 a customer master key (CMK). The CMK can be `aws/elasticfilesystem` (the AWS-managed CMK for Amazon EFS), or it can be a CMK that you manage.

File data—the contents of your files—is encrypted at rest using the CMK that you specified when you created your file system. Metadata—file names, directory names, and directory contents—is encrypted by a key that Amazon EFS manages.

The AWS-managed CMK for your file system is used as the master key for the metadata in your file system, for example file names, directory names, and directory contents. You own the CMK used to encrypt file data (the contents of your files) at rest.

You manage who has access to your CMKs 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 CMK you specified when the file system was created. For more information, see the following:

- [IAM Users](#) in the *IAM User Guide*
- [Using Key Policies in AWS KMS](#) in the *AWS Key Management Service Developer Guide*
- [Using Grants](#) 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 CMK that you specified (when you created the file system with encryption at rest) affects access to its contents. The CMK must be in the `enabled` state for users to have access to the contents of an encrypted-at-rest file system.

## Performing Administrative Actions on Amazon EFS Customer Master Keys

Following, you can find how to enable, disable, or delete the CMKs associated with your Amazon EFS file system. You can also learn about the behavior to expect from your file system when you perform these actions.

### Disabling, Deleting, or Revoking Access to the CMK for a File System

You can disable or delete your customer-managed CMKs, or you can revoke Amazon EFS access to your CMKs. Disabling and revoking access for Amazon EFS to your keys are reversible actions. Exercise significant caution when deleting CMKs. Deleting a CMK is an irreversible action.

If you disable or delete the CMK used for your mounted file system, the following is true:

- That CMK can't be used as the master key for new encrypted-at-rest file systems.
- Existing encrypted-at-rest file systems that use that CMK 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 CMK. 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 CMK 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 CMK, but you can schedule it for deletion in 7-30 days. While a CMK is scheduled for deletion, you can't use it for cryptographic operations. You can also cancel a CMK's scheduled deletion.

To learn how to disable and re-enable customer-managed CMKs, see [Enabling and Disabling Keys](#) in the AWS Key Management Service Developer Guide. To learn how to schedule deletion of customer-managed CMKs, see [Deleting Customer Master Keys](#) in the AWS Key Management Service Developer Guide.

## Related Topics

- For more information on encrypted data and metadata at rest in Amazon EFS, see [Data Encryption in EFS \(p. 135\)](#).
- For example key policies, see [Amazon EFS Key Policies for AWS KMS \(p. 137\)](#).
- For a list of AWS CloudTrail log entries associated with an encrypted file system, see [Amazon EFS Log File Entries for Encrypted-at-Rest File Systems \(p. 88\)](#).
- For more information on determining what accounts and services have access to your CMKs, see [Determining Access to an AWS KMS Customer Master Key](#) in the *AWS Key Management Service Developer Guide*.

# Mounting EFS File Systems

In the following section, you can learn how to mount your Amazon EFS file system on a Linux instance using the Amazon EFS mount helper. In addition, you can find how to use the file `fstab` to automatically remount your file system after any system restarts. To learn more about the Amazon EFS mount helper, see [EFS Mount Helper](#) (p. 39).

Before the Amazon EFS mount helper was available, we recommended mounting your Amazon EFS file systems using the standard Linux NFS client. For more information on those changes, see [Mounting File Systems Without the EFS Mount Helper](#) (p. 290).

## Note

You can configure a new Amazon EC2 Linux instance to mount an Amazon EFS file system when it launches. However, before you mount a file system, you must create, configure, and launch your related AWS resources. For detailed instructions, see [Getting Started with Amazon Elastic File System](#) (p. 11).

## Topics

- [Troubleshooting AMI and Kernel Versions](#) (p. 61)
- [Installing the amazon-efs-utils Package](#) (p. 61)
- [Mounting with the EFS Mount Helper](#) (p. 61)
- [Mounting Your Amazon EFS File System Automatically](#) (p. 64)
- [Mounting EFS File Systems from Another Account or VPC](#) (p. 68)
- [Additional Mounting Considerations](#) (p. 70)

## Troubleshooting AMI and Kernel Versions

To troubleshoot issues related to certain Amazon Machine Image (AMI) or kernel versions when using Amazon EFS from an Amazon EC2 instance, see [Troubleshooting AMI and Kernel Issues](#) (p. 176).

## Installing the amazon-efs-utils Package

To mount your Amazon EFS file system on your Amazon EC2 instance, we recommend that you use the mount helper in the amazon-efs-utils package. The amazon-efs-utils package is an open-source collection of Amazon EFS tools. For more information, see [Installing the amazon-efs-utils Package on Amazon Linux](#) (p. 36).

## Mounting with the EFS Mount Helper

You can mount an Amazon EFS file system on a number of clients using the Amazon EFS mount helper. The following sections, you can find the mount helper process for the different types of clients.

## Topics

- [Mounting on Amazon EC2 with the EFS Mount Helper](#) (p. 62)
- [Mounting with IAM Authorization](#) (p. 62)
- [Mounting with EFS Access Points](#) (p. 63)

- [Mounting Automatically with EFS Mount Helper \(p. 63\)](#)
- [Mounting on Your On-Premises Linux Client with the EFS Mount Helper over AWS Direct Connect and VPN \(p. 64\)](#)

## Mounting on Amazon EC2 with the EFS Mount Helper

You can mount an Amazon EFS file system on an Amazon EC2 instance using the Amazon EFS mount helper. For more information on the mount helper, see [EFS Mount Helper \(p. 39\)](#). To use the mount helper, you need the following:

- **The file system ID of the EFS file system that you want to mount** – After you create an Amazon EFS file system, you can get that file system's ID from the console or programmatically through the Amazon EFS API. The ID is in this format: `fs-12345678`.
- **An Amazon EFS mount target** – You create mount targets in your virtual private cloud (VPC). If you create your file system in the console, you create your mount targets at the same time. For more information, see [Creating a Mount Target Using the Amazon EFS console \(p. 23\)](#).
- **An Amazon EC2 instance running a supported distribution of Linux** – The supported Linux distributions for mounting your file system with the mount helper are the following:
  - Amazon Linux 2
  - Amazon Linux 2017.09 and newer
  - Red Hat Enterprise Linux (and derivatives such as CentOS) version 7 and newer
  - and Ubuntu 16.04 LTS and newer
- **The Amazon EFS mount helper installed** – The mount helper is a tool in `amazon-efs-utils`. For information on how to install `amazon-efs-utils`, see [Installing the amazon-efs-utils Package on Amazon Linux \(p. 36\)](#).

### To mount your Amazon EFS file system with the mount helper

1. Access the terminal for your instance through Secure Shell (SSH), and log in with the appropriate user name. For more information on how to do this, see [Connecting to Your Linux Instance Using SSH](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. Run the following command to mount your file system.

```
sudo mount -t efs fs-12345678:/ /mnt/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-12345678:/ /mnt/efs
```

## Mounting with IAM Authorization

To mount your Amazon EFS file system on Linux instances using AWS Identity and Access Management (IAM) authorization, you use the EFS mount helper. For more information about IAM authorization for NFS clients, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#).

### 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 Amazon 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. To do this, use the EFS mount helper.

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 Amazon EFS Access Points \(p. 161\)](#).

## Mounting Automatically with EFS Mount Helper

You also have the option of mounting automatically by adding an entry to your `/etc/fstab` file. When you mount automatically using `/etc/fstab`, you must add the `_netdev` mount option. For more information, see [Using /etc/fstab to Mount Automatically \(p. 66\)](#).

### Note

Mounting with the mount helper automatically uses the following mount options that are optimized for Amazon EFS:

- `nfsvers=4.1`
- `rsiz=1048576`

- `wsz=1048576`
- `hard`
- `timeo=600`
- `retrans=2`
- `noresvport`

To use the `mount` command, the following must be true:

- 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 Amazon. For information about the Amazon DNS server, see [DHCP Options Sets](#) in the *Amazon VPC User Guide*.
- The VPC of the connecting EC2 instance must have DNS hostnames enabled. For more information, see [Viewing DNS Hostnames for Your EC2 Instance](#) in the *Amazon VPC User Guide*.

**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 on Your On-Premises Linux Client with the EFS Mount Helper over AWS Direct Connect and VPN

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. Mounting your Amazon EFS file systems with `amazon-efs-utils` also makes mounting simpler with the mount helper and allows you to enable encryption of data in transit.

To see how to use `amazon-efs-utils` with AWS Direct Connect and VPN to mount Amazon EFS file systems onto on-premises Linux clients, see [Walkthrough: Create and Mount a File System On-Premises with AWS Direct Connect and VPN](#) (p. 119).

## Mounting Your Amazon EFS File System Automatically

You can configure an Amazon EC2 instance to automatically mount an EFS file system when it reboots in two ways:

- When you create a new EC2 instance using the Launch Instance Wizard.
- Update the EC2 `/etc/fstab` file with an entry for the EFS file system.

Both of these methods use 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 [Using the amazon-efs-utils Tools](#) (p. 35). 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-utils Package on Other Linux Distributions](#) (p. 36).



## Configuring EC2 Instances to Mount an EFS File System at Instance Launch

When you create a new Amazon EC2 Linux instance using the 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.

Before you perform this procedure, make sure that you have created your Amazon EFS file system. For more information, see [Step 1: Create Your Amazon EFS File System \(p. 12\)](#) in the Amazon EFS Getting Started exercise.

### Note

You can't use Amazon EFS with Microsoft Windows-based Amazon EC2 instances.

Before you can launch and connect to an Amazon EC2 instance, you need to create a key pair, unless you already have one. Follow the steps in [Setting Up with Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances* to create a key pair. If you already have a key pair, you can use it for this exercise.

### 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.

## Using /etc/fstab to Mount Automatically

To automatically remount your Amazon EFS file system directory when the Amazon EC2 instance reboots, use the file `/etc/fstab`. The `/etc/fstab` file contains information about file systems. The command `mount -a`, which runs during instance startup, mounts the file systems listed in `/etc/fstab`. This procedure uses the EFS mount helper to mount the file system and needs to be installed on the EC2 instance.

The mount helper is part of the `amazon-efs-utils` set of tools, which is available for installation on Amazon Linux and Amazon Linux 2 Amazon Machine Images (AMIs). For more information about installing `amazon-efs-utils` on an Amazon Linux or Amazon Linux 2 AMI, see [Installing the amazon-efs-utils Package on Amazon Linux \(p. 36\)](#). 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-utils Package on Other Linux Distributions \(p. 36\)](#).

### Note

Before you can update the `/etc/fstab` file of your EC2 instance, make sure that you already created your Amazon EFS file system. For more information, see [Step 1: Create Your Amazon EFS File System \(p. 12\)](#) in the Amazon EFS Getting Started exercise.

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

1. Connect to your EC2 instance:
  - To connect to your instance from a computer running macOS or Linux, specify the `.pem` file for your SSH command. To do this, use the `-i` option and the path to your private key.
  - To connect to your instance from a computer running Windows, you can use either MindTerm or PuTTY. To use PuTTY, install it and convert the `.pem` file to a `.ppk` file.

For more information, see the following topics in the *Amazon EC2 User Guide for Linux Instances*:

- [Connecting to Your Linux Instance Using SSH](#)
  - [Connecting to Your Linux Instance from Windows Using PuTTY](#)
2. Open the `/etc/fstab` file in an editor.
  3. Automatically mount your EFS file system using either IAM authorization or an EFS access point:
    - To automatically mount with IAM authorization to an Amazon EC2 instance that has an instance profile, add the following line to the `/etc/fstab` file.

```
file-system-id:/ efs-mount-point efs _netdev,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,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,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](#) (p. 178).

For more information, see [Mounting with IAM Authorization](#) (p. 62) and [Mounting with EFS Access Points](#) (p. 63).

4. Save the changes to the file.
5. Test the `fstab` entry by using the `mount` command with the `'fake'` option along with the `'all'` and `'verbose'` options.

```
$ sudo mount -fav  
home/ec2-user/efs      : successfully mounted
```

Your EC2 instance is now configured to mount the EFS file system whenever it restarts.

**Note**

In some cases, your Amazon EC2 instance might need to start regardless of the status of your mounted Amazon 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
<code>file-system-id:/</code>	The ID for your Amazon EFS file system. You can get this ID from the console or programmatically from the CLI or an AWS SDK.
<code>efs-mount-point</code>	The mount point for the EFS file system on your EC2 instance.
<code>efs</code>	The type of file system. When you're using the mount helper, this type is always <code>efs</code> .
<code>mount options</code>	Mount options for the file system. This is a comma-separated list of the following options: <ul style="list-style-type: none"><li>• <code>_netdev</code> – 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.</li><li>• <code>tls</code> – enables encryption of data in transit.</li><li>• <code>iam</code> – Use this option to mount with IAM authorization to an Amazon EC2 that has an instance profile. Using the <code>iam</code> mount option requires also using the <code>tls</code> option. For more information, see <a href="#">Using IAM to Control NFS Access to Amazon EFS</a> (p. 150).</li><li>• <code>awsprofile=namedprofile</code> – Use this option with the <code>iam</code> and <code>tls</code> options to mount with IAM authorization to a Linux instance using a credentials file. For more information about EFS access points, see <a href="#">Using IAM to Control NFS Access to Amazon EFS</a> (p. 150).</li><li>• <code>accesspoint=access-point-id</code> – Use this option with the <code>tls</code> option to mount using an EFS access point. For more information about EFS access points, see <a href="#">Working with Amazon EFS Access Points</a> (p. 161).</li></ul>
<code>0</code>	A nonzero value indicates that the file system should be backed up by <code>dump</code> . For EFS, this value should be 0.

Field	Description
0	The order in which <code>fsck</code> checks file systems at boot. For EFS file systems, this value should be 0 to indicate that <code>fsck</code> should not run at startup.

## Mounting EFS File Systems from Another Account or VPC

You can mount your Amazon 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.

## Mounting Using IAM or Access Points 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.

### Prerequisites

Before using the following the procedure, take these steps:

- Install the `amazon-efs-utils` set of tools on the client. You use the EFS mount helper to mount the file system, and the mount helper is part of `amazon-efs-utils`. For instructions on installing `amazon-efs-utils`, see [Using the amazon-efs-utils Tools \(p. 35\)](#).
- 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 (AZ) 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 AZ ID. This requirement applies because AZ names can differ between accounts.

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

1. Connect to your EC2 instance:

- To connect to your instance from a computer running Mac OS or Linux, specify the .pem file for your SSH command. To do this, use the `-i` option and the path to your private key.
- To connect to your instance from a computer running Windows, you can use either MindTerm or PuTTY. To use PuTTY, install it and convert the .pem file to a .ppk file.

For more information, see the following topics in the *Amazon EC2 User Guide for Linux Instances*:

- [Connecting to Your Linux Instance from Windows Using PuTTY](#)
- [Connecting to Your Linux Instance Using SSH](#)

2. Determine the AZ 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 `curl -s
http://169.254.169.254/latest/meta-data/placement/availability-zone`
{
  "AvailabilityZones": [
    {
      "State": "available",
      "ZoneName": "us-east-2b",
      "Messages": [],
      "ZoneId": "use2-az2",
      "RegionName": "us-east-2"
    }
  ]
}
```

The AZ ID is returned in the `ZoneId` property, `use2-az2`.

3. 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 AZ ID has an IP address of 10.0.2.153.

4. Add a line for the mount target IP address to the client's `/etc/hosts` file, in the format of `mount-target-IP-Address file-system-ID.efs.region.amazonaws.com`, as follows.

```
echo "10.0.2.153 fs-01234567.efs.us-east-2.amazonaws.com" | sudo tee -a /etc/hosts
```

5. Create a directory for mounting the file system using the following command.

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

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

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

For more information about using IAM authorization with EFS, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#).

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-id /mnt/efs/
```

For more information about EFS access points, see [Working with Amazon EFS Access Points \(p. 161\)](#).

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](#) and [Working with Records](#) in the *Amazon Route 53 Developer Guide*.

## Mounting from Another Account in the Same VPC

Using shared VPCs, you can mount an Amazon EFS file system that is owned by one account from Amazon EC2 instances that are owned by a different account. For more information about setting up a shared VPC, see [Working with Shared VPCs](#) 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.

## Additional Mounting Considerations

We recommend the following default values for mount options on Linux:

- `rsz=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.

- `wsiz=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 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.

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 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 [man fstab](#) and [man nfs](#) pages in 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.

## 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 Amazon EFS file system through the AWS CLI, the AWS Management Console, or through any of the AWS SDKs. To unmount an Amazon EFS file system connected to an Amazon 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 Amazon 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 Amazon 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 Example: Identify the Mount Status of an Amazon 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% /
```



# Monitoring Amazon EFS

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon EFS and your AWS solutions. You should 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, you should 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, you should 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/or metadata operations, client connections, and burst credit balances for your file systems. When performance falls outside your established baseline, you might need 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.

## 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, while 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 simply 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 EFS with Amazon CloudWatch \(p. 74\)](#).

- **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see [Monitoring Log Files](#) 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 console dashboards provide an at-a-glance view of the state of your AWS environment. We recommend that you also check the log files on file system.

- 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 life cycle 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 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

## Monitoring EFS 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 access historical information and 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. For more information about CloudWatch, see [What Are Amazon CloudWatch, Amazon CloudWatch Events, and Amazon CloudWatch Logs?](#) in the *Amazon CloudWatch User Guide*.

## Amazon CloudWatch Metrics for Amazon EFS

The AWS/EFs namespace includes the following metrics.

### **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. For more information, see [Throughput Scaling with Bursting Mode \(p. 93\)](#).

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`

### **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`

### **DataReadIOBytes**

The 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 number of bytes for each file 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 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`

#### **PercentIOLimit**

Shows how close a file system is to reaching the I/O limit of the General Purpose performance mode. If this metric is at 100% more often than not, consider moving your application to a file system using the Max I/O performance mode.

##### **Note**

This metric is only submitted for file systems using the General Purpose performance mode.

Units:

- Percent

#### **PermittedThroughput**

The maximum amount of throughput a file system is allowed. For file systems in the Provisioned Throughput mode, if the amount of storage allows your file system to drive a higher amount of throughput than you provisioned, this metric will reflect the higher throughput instead of the provisioned amount. For file systems in the Bursting Throughput mode, this value is a function of the file system size and `BurstCreditBalance`. For more information, see [Amazon EFS Performance \(p. 90\)](#).

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.

Units: Bytes per second

Valid statistics: `Minimum`, `Maximum`, `Average`

#### **TotalIOBytes**

The number of bytes for each file system operation, including data read, data write, and metadata operations.

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`

## Bytes Reported in CloudWatch

As with Amazon S3 and Amazon EBS, Amazon EFS CloudWatch metrics are reported as raw *Bytes*. Bytes are not rounded to either a decimal or binary multiple of the unit. Keep this in mind when calculating your burst rate using the data you get from the metrics. For more information on bursting, see [Throughput Scaling with Bursting Mode \(p. 93\)](#).

## Amazon EFS Dimensions

Amazon EFS metrics use the `EFS` namespace and provides metrics for a single dimension, `FileSystemId`. A file system's ID can be found in the Amazon EFS management console, and it takes the form of `fs-XXXXXXX`.

## How Do I Use Amazon EFS Metrics?

The metrics reported by Amazon EFS provide information that you can analyze in different ways. The list below 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 <code>Sum</code> statistic of the <code>TotalIOBytes</code> metric to see your throughput.
How can I track the number of Amazon EC2 instances that are connected to a file system?	You can monitor the <code>Sum</code> statistic of the <code>ClientConnections</code> metric. To calculate the average <code>ClientConnections</code> 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 <code>BurstCreditBalance</code> metric for your file system. For more information on bursting and burst credits, see <a href="#">Throughput Scaling with Bursting Mode (p. 93)</a> .

## Accessing CloudWatch Metrics

You can see Amazon EFS metrics for CloudWatch in many ways. You can view them through the CloudWatch console, or you can access them using the CloudWatch CLI or the CloudWatch API. The following procedures show you how to access the metrics using these various tools.

### To view metrics using the 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, type 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 [Amazon CloudWatch API Reference](#).

## 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 you specify, and 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 simply 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. For more information, see [Walkthrough: Enforcing Encryption on an Amazon EFS File System at Rest \(p. 130\)](#).

### Note

Currently, you can't enforce encryption in transit.

The following procedures outline how to create alarms for Amazon EFS.

### 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 just the Amazon EFS metrics in this dialog box, search on 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**.

### To set an alarm using the AWS CLI

- Call `put-metric-alarm`. For more information, see [AWS CLI Command Reference](#).

### To set an alarm using the CloudWatch API

- Call `PutMetricAlarm`. For more information, see [Amazon CloudWatch API Reference](#)

## Using Metric Math with Amazon EFS

Using metric math, you can query multiple 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 [Use Metric Math](#) in the *Amazon CloudWatch User Guide*.

Following, find some useful metric math expressions for Amazon EFS.

### Topics

- [Metric Math: Throughput in MiB/Second \(p. 79\)](#)
- [Metric Math: Percent Throughput \(p. 79\)](#)
- [Metric Math: Percentage of Permitted Throughput Utilization \(p. 80\)](#)
- [Metric Math: Throughput IOPS \(p. 80\)](#)
- [Metric Math: Percentage of IOPS \(p. 81\)](#)
- [Metric Math: Average I/O Size in KiB \(p. 81\)](#)
- [Using Metric Math Through an AWS CloudFormation Template for Amazon EFS \(p. 82\)](#)

## Metric Math: Throughput in MiB/Second

To calculate the average throughput (in MiB/second) 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	<ul style="list-style-type: none"><li>• <code>DataReadIOBytes</code></li><li>• <code>DataWriteIOBytes</code></li><li>• <code>MetadataIOBytes</code></li><li>• <code>TotalIOBytes</code></li></ul>	sum	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	<code>(m1/1048576)/PERIOD(m1)</code>

## Metric Math: Percent Throughput

To calculate the percent throughput of the different 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	<ul style="list-style-type: none"> <li><code>TotalIOBytes</code></li> </ul>	sum	1 minute
m2	<ul style="list-style-type: none"> <li><code>DataReadIOBytes</code></li> <li><code>DataWriteIOBytes</code></li> <li><code>MetadataIOBytes</code></li> </ul>	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 (`TotalIOBytes`) for a time period, first multiply the throughput in MiB/second by 100. Then divide the result by the sum statistic of `PermittedThroughput` converted to MiB for the same period.

Suppose that your example logic is this: (metric math expression for throughput in MiB/second 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
e1	<ul style="list-style-type: none"> <li>Throughput in MiB/Second</li> </ul>		
m1	<ul style="list-style-type: none"> <li><code>PermittedThroughput</code></li> </ul>	sum	1 minute

Your metric math ID and expression are the following.

ID	Expression
e2	$(e1 * 100) / (m1 / 1048576)$

## 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	<ul style="list-style-type: none"> <li><code>DataReadIOBytes</code></li> <li><code>DataWriteIOBytes</code></li> <li><code>MetadataIOBytes</code></li> <li><code>TotalIOBytes</code></li> </ul>	sample count	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	<code>m1 / PERIOD(m1)</code>

## 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	<ul style="list-style-type: none"> <li><code>TotalIOBytes</code></li> </ul>	sample count	1 minute
m2	<ul style="list-style-type: none"> <li><code>DataReadIOBytes</code></li> <li><code>DataWriteIOBytes</code></li> <li><code>MetadataIOBytes</code></li> </ul>	sample count	1 minute

Your metric math ID and expression are the following.

ID	Expression
e1	<code>(m2*100)/m1</code>

## 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:  $(\text{sum of DataReadIOBytes} \div 1,024 \text{ (to convert to KiB)}) \div \text{sample count of DataReadIOBytes}$

Then your CloudWatch metric information is the following.

ID	Usable Metrics	Statistic	Period
m1	<ul style="list-style-type: none"><li>DataReadIOBytes</li><li>DataWriteIOBytes</li><li>MetadataIOBytes</li></ul>	sum	1 minute
m2	<ul style="list-style-type: none"><li>DataReadIOBytes</li><li>DataWriteIOBytes</li><li>MetadataIOBytes</li></ul>	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*.

# 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.

To learn more about CloudTrail, see 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 [Viewing Events 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*:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

All Amazon EFS [API calls](#) (p. 185) 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 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 the [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 the Amazon EFS Service-Linked Role \(p. 152\)](#).

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": {
        "111122223333-10-111122223333Statement111122223333Action111122223333AssumeRole111122223333Effect%22%3A%20%22Allow%22%2C%20%22Principal%22%3A%20%7B%22Service%22%3A%20%5B%22elasticfilesystem.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": "2019-10-23T22:50:05Z",
"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 IAM Authentication

Amazon EFS IAM 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, It's also emitted when the new list of permitted actions doesn't include `ClientMount`. For more information about EFS IAM authorization, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#).

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"  
    },  
    "type": "AssumedRole"  
  },  
  "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-ae3-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 [Data Encryption in EFS \(p. 135\)](#).

If you're using an encrypted-at-rest file system, the calls that Amazon EFS makes on your behalf appear in your AWS CloudTrail logs as coming from an AWS-owned account. If you see one of the following account IDs in your CloudTrail logs, depending on the AWS Region that your file system is created in, this ID is one owned by the Amazon EFS service.

AWS Region	Account ID
US East (Ohio)	771736226457
US East (N. Virginia)	055650462987
US West (N. California)	208867197265
US West (Oregon)	736298361104
Africa (Cape Town)	855919597013
Asia Pacific (Hong Kong)	832882505983
Asia Pacific (Mumbai)	063610658798
Asia Pacific (Seoul)	518632624599
Asia Pacific (Singapore)	448676862907
Asia Pacific (Sydney)	288718191711
Asia Pacific (Tokyo)	620757817088
Canada (Central)	838331228873
China (Beijing)	365623262523
China (Ningxia)	361049930386
Europe (Frankfurt)	992038834663
Europe (Ireland)	805538244694
Europe (London)	838331228873



AWS Region	Account ID
Europe (Milan)	393586363892
Europe (Paris)	063566772258
Europe (Stockholm)	030059730498
Middle East (Bahrain)	545970776878
South America (São Paulo)	041656882570
AWS GovCloud (US-East)	167972735943
AWS GovCloud (US-West)	174619389399

## Amazon EFS Encryption Context for Encryption at Rest

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"
```

# Amazon EFS Performance

Following, you can find an overview of Amazon EFS performance, with a discussion of the available performance and throughput modes and some useful performance tips.

## Performance Overview

Amazon EFS file systems are distributed across an unconstrained number of storage servers. This distributed data storage design enables file systems to grow elastically to petabyte scale and enables massively parallel access from Amazon EC2 instances to your data. The distributed design of Amazon EFS avoids the bottlenecks and constraints inherent to traditional file servers.

This distributed data storage design means that multithreaded applications and applications that concurrently access data from multiple Amazon EC2 instances can drive substantial levels of aggregate throughput and IOPS. Big data and analytics workloads, media processing workflows, content management, and web serving are examples of these applications.

In addition, Amazon EFS data is distributed across multiple Availability Zones (AZs), providing a high level of durability and availability. The following tables compare high-level performance and storage characteristics for Amazon's file and block cloud storage services.

### Performance Comparison, Amazon EFS and Amazon EBS

	Amazon EFS	Amazon EBS Provisioned IOPS
<b>Per-operation latency</b>	Low, consistent latency.	Lowest, consistent latency.
<b>Throughput scale</b>	10+ GB per second.	Up to 2 GB per second.

### Storage Characteristics Comparison, Amazon EFS and Amazon EBS

	Amazon EFS	Amazon EBS Provisioned IOPS
<b>Availability and durability</b>	Data is stored redundantly across multiple AZs.	Data is stored redundantly in a single AZ.
<b>Access</b>	Up to thousands of Amazon EC2 instances, from multiple AZs, can connect concurrently to a file system.	A single Amazon EC2 instance in a single AZ can connect to a file system.
<b>Use cases</b>	Big data and analytics, media processing workflows, content management, web serving, and home directories.	Boot volumes, transactional and NoSQL databases, data warehousing, and ETL.

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. Due to 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. Amazon EFS supports highly parallelized workloads (for example, using concurrent operations from multiple threads and multiple Amazon EC2 instances), which enables high levels of aggregate throughput and operations per second.

## Amazon EFS Use Cases

Amazon EFS is designed to meet the performance needs of the following use cases.

### Big Data and Analytics

Amazon EFS provides the scale and performance required for big data applications that require high throughput to compute nodes coupled with read-after-write consistency and low-latency file operations.

### Media Processing Workflows

Media workflows like video editing, studio production, broadcast processing, sound design, and rendering often depend on shared storage to manipulate large files. A strong data consistency model with high throughput and shared file access can cut the time it takes to perform these jobs and consolidate multiple local file repositories into a single location for all users.

### Content Management and Web Serving

Amazon EFS provides a durable, high throughput file system for content management systems that store and serve information for a range of applications like websites, online publications, and archives.

### Home Directories

Amazon EFS can provide storage for organizations that have many users that need to access and share common datasets. An administrator can use Amazon EFS to create a file system accessible to people across an organization and establish permissions for users and groups at the file or directory level.

## Performance Modes

To support a wide variety of cloud storage workloads, Amazon EFS offers two performance modes. You select a file system's performance mode when you create it.

The two performance modes have no additional costs, so your Amazon EFS file system is billed and metered the same, regardless of your performance mode. For information about file system limits, see [Quotas for Amazon EFS File Systems \(p. 169\)](#).

**Note**

An Amazon EFS file system's performance mode can't be changed after the file system has been created.

### General Purpose Performance Mode

We recommend the General Purpose performance mode for the majority of your Amazon EFS file systems. General Purpose is ideal for latency-sensitive use cases, like web serving environments, content management systems, home directories, and general file serving. If you don't choose a performance

mode when you create your file system, Amazon EFS selects the General Purpose mode for you by default.

## Max I/O Performance Mode

File systems in the Max I/O mode can scale to higher levels of aggregate throughput and operations per second. This scaling is done with a tradeoff of slightly higher latencies for file metadata operations. Highly parallelized applications and workloads, such as big data analysis, media processing, and genomics analysis, can benefit from this mode.

## Using the Right Performance Mode

Our recommendation for determining which performance mode to use is as follows:

1. [Create a new file system \(p. 12\)](#) using the default General Purpose performance mode.
2. Run your application (or a use case similar to your application) for a period of time to test its performance.
3. Monitor the [PercentIOLimit \(p. 74\)](#) Amazon CloudWatch metric for Amazon EFS during the performance test. For more information about accessing this and other metrics, see [Amazon CloudWatch Metrics \(p. 73\)](#).

If the `PercentIOLimit` percentage returned was at or near 100 percent for a significant amount of time during the test, your application should use the Max I/O performance mode. Otherwise, it should use the default General Purpose mode.

To move to a different performance mode, migrate the data to a different file system that was created in the other performance mode. You can use DataSync to transfer files between two EFS file systems. To learn more, see [Transferring Data into Amazon EFS \(p. 50\)](#).

Some latency-sensitive workloads require the higher I/O levels provided by Max I/O performance mode and the lower latency provided by General Purpose performance mode. For this type of workload, we recommend creating multiple General Purpose performance mode file systems. In this case, we recommend then spreading the application workload across all these file systems, as long as the workload and applications can support it.

By taking this approach, you can create a logical file system and shard data across multiple EFS file systems. Each file system is mounted as a subdirectory, and your application can access these subdirectories in parallel. This approach allows latency-sensitive workloads to scale to higher levels of file system operations per second, aggregated across multiple file systems. At the same time, these workloads can take advantage of the lower latencies offered by General Purpose performance mode file systems.

## Throughput Modes

There are two throughput modes to choose from for your file system, Bursting Throughput and Provisioned Throughput. With *Bursting Throughput* mode, throughput on Amazon EFS scales as the size of your file system in the standard storage class grows. For more information about EFS storage classes, see [EFS Storage Classes \(p. 50\)](#). With *Provisioned Throughput* mode, you can instantly provision the throughput of your file system (in MiB/s) independent of the amount of data stored.

### Note

You can decrease your file system throughput in Provisioned Throughput mode as long as it's been more than 24 hours since the last decrease. Additionally, you can change between

Provisioned Throughput mode and the default Bursting Throughput mode as long as it's been more than 24 hours since the last throughput mode change.

## Throughput Scaling with Bursting Mode

With Bursting Throughput mode, throughput on Amazon EFS scales as a file system stored in the standard storage class grows. File-based workloads are typically spiky, driving high levels of throughput for short periods of time, and low levels of throughput the rest of the time. To accommodate this, Amazon EFS is designed to burst to high throughput levels for periods of time.

All file systems, regardless of size, can burst to 100 MiB/s of throughput. Those over 1 TiB in the standard storage class can burst to 100 MiB/s per TiB of data stored in the file system. For example, a 10-TiB file system can burst to 1,000 MiB/s of throughput ( $10 \text{ TiB} \times 100 \text{ MiB/s/TiB}$ ). The portion of time a file system can burst is determined by its size. The bursting model is designed so that typical file system workloads can burst virtually any time they need to. For file systems using Bursting Throughput mode, the allowed throughput is determined based on the amount of the data stored in the Standard storage class only. For more information about EFS storage classes, see [EFS Storage Classes \(p. 50\)](#).

Amazon EFS uses a credit system to determine when file systems can burst. Each file system earns credits over time at a baseline rate that is determined by the size of the file system that is stored in the standard storage class. A file system uses credits whenever it reads or writes data. The baseline rate is 50 MiB/s per TiB of storage (equivalently, 50 KiB/s per GiB of storage).

Accumulated burst credits give the file system the ability to drive throughput above its baseline rate. A file system can drive throughput continuously at its baseline rate, and whenever it's inactive or driving throughput below its baseline rate, the file system accumulates burst credits.

For example, a 100-GiB file system can burst (at 100 MiB/s) for 5 percent of the time if it's inactive for the remaining 95 percent. Over a 24-hour period, the file system earns 432,000 MiBs worth of credit, which can be used to burst at 100 MiB/s for 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.

The following table provides examples of bursting behavior.

File System Size	Aggregate Read/Write Throughput
A 100-GiB file system can...	<ul style="list-style-type: none"><li>Burst to 100 MiB/s for up to 72 minutes each day, or</li><li>Drive up to 5 MiB/s continuously</li></ul>
A 1-TiB file system can...	<ul style="list-style-type: none"><li>Burst to 100 MiB/s for 12 hours each day, or</li><li>Drive 50 MiB/s continuously</li></ul>
A 10-TiB file system can...	<ul style="list-style-type: none"><li>Burst to 1 GiB/s for 12 hours each day, or</li><li>Drive 500 MiB/s continuously</li></ul>
Generally, a larger file system can...	<ul style="list-style-type: none"><li>Burst to 100MiB/s per TiB of storage for 12 hours each day, or</li><li>Drive 50 MiB/s per TiB of storage continuously</li></ul>

### Note

The minimum file system size used when calculating the baseline rate is 1 GiB, so all file systems have a baseline rate of at least 50 KiB/s.

The file system size used when determining the baseline rate and burst rate is the same as the metered size available through the `DescribeFileSystems` 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 approach implies that file systems can accumulate enough credits to burst for up to 12 hours continuously.

The following table provides more detailed examples of bursting behavior for file systems of different sizes.

File System Size (GiB)	Baseline Aggregate Throughput (MiB/s)	Burst Aggregate Throughput (MiB/s)	Maximum Burst Duration (Min/Day)	% of Time File System Can Burst (Per Day)
10	0.5	100	7.2	0.5%
256	12.5	100	180	12.5%
512	25.0	100	360	25.0%
1024	50.0	100	720	50.0%
1536	75.0	150	720	50.0%
2048	100.0	200	720	50.0%
3072	150.0	300	720	50.0%
4096	200.0	400	720	50.0%

#### Note

As previously mentioned, new file systems have an initial burst credit balance of 2.1 TiB. With this starting balance, you can burst at 100 MB/s for 6.12 hours without spending any credits that you're earning from your storage. This starting formula is calculated as  $2.1 \times 1024 \times (1024 / 100 / 3600)$  to get 6.116 hours, rounded up to 6.12.

## Managing Burst Credits

When a file system has a positive burst credit balance, it can burst. You can see the burst credit balance for a file system by viewing the `BurstCreditBalance` Amazon CloudWatch metric for Amazon EFS. For more information about accessing this and other metrics, see [Monitoring Amazon EFS \(p. 73\)](#).

The bursting capability (both in terms of length of time and burst rate) of a file system is directly related to its size. Larger file systems can burst at larger rates for longer periods of time. In some cases, your application might need to burst more (that is, you might find that your file system is running out of burst credits. In these cases, you should increase the size of your file system, or switch to Provisioned Throughput mode.

Use your historical throughput patterns to calculate the file system size you need to sustain the level of activity that you want. The following steps outline how to do this.

#### To calculate the file system size that you need to sustain the level activity that you want

1. Identify your throughput needs by looking at your historical usage. From the [Amazon CloudWatch console](#), check the sum statistic of the `TotalIOBytes` metric with daily aggregation, for the past 14 days. Identify the day with the largest value for `TotalIOBytes`.
2. Divide this number by 24 hours, 60 minutes, 60 seconds, and 1024 bytes to get the average KiB/second your application required for that day.

3. Calculate the file system size (in GiB) required to sustain this average throughput by dividing the average throughput number (in KiB/s) by the baseline throughput number (50 KiB/s/GiB) that EFS provides.

## Specifying Throughput with Provisioned Mode

Provisioned Throughput mode is available for applications with high throughput to storage (MiB/s per TiB) ratios, or with requirements greater than those allowed by the Bursting Throughput mode. For example, say you're using Amazon EFS for development tools, web serving, or content management applications where the amount of data in your file system is low relative to throughput demands. Your file system can now get the high levels of throughput your applications require without having to pad your file system.

Additional charges are associated with using Provisioned Throughput mode. Using Provisioned Throughput mode, you're billed for the storage that you use and for the throughput that you provision above what you're provided. The amount of throughput that you are provided is based on the amount of data stored in the Standard storage class. For more information about EFS storage classes, see [EFS Storage Classes \(p. 50\)](#). For more information on pricing, see [Amazon EFS Pricing](#).

Throughput limits remain the same, regardless of the throughput mode you choose. For more information on these limits, see [Amazon EFS Quotas That You Can Increase \(p. 167\)](#).

If your file system is in the Provisioned Throughput mode, you can increase the Provisioned Throughput of your file system as often as you want. You can decrease your file system throughput in Provisioned Throughput mode as long as it's been more than 24 hours since the last decrease. Additionally, you can change between Provisioned Throughput mode and the default Bursting Throughput mode as long as it's been more than 24 hours since the last throughput mode change.

If your file system's metered size provides a higher baseline rate than the amount of throughput you provisioned, your file system follows the default Amazon EFS Bursting Throughput model. You don't incur charges for Provisioned Throughput below your file system's entitlement in Bursting Throughput mode. For more information, see [Throughput Scaling with Bursting Mode \(p. 93\)](#).

## Using the Right Throughput Mode

By default, we recommend that you run your application in the Bursting Throughput mode. If you experience performance issues, check the `BurstCreditBalance` CloudWatch metric. If the value of the `BurstCreditBalance` metric is either zero or steadily decreasing, Provisioned Throughput is right for your application.

In some cases, your file system might run in Provisioned Throughput mode with no performance issues. However, at the same time, your `BurstCreditBalance` continuously increases for long periods of normal operations. In such a case, consider decreasing the amount of provisioned throughput to reduce costs.

If you're planning on migrating large amounts of data into your file system, consider switching to Provisioned Throughput mode. In this case, you can provision a higher throughput beyond your allotted burst capability to accelerate loading data. Following the migration, consider lowering the amount of provisioned throughput or switch to Bursting Throughput mode for normal operations.

Compare the average throughput to which you're driving the file system to the `PermittedThroughput` metric. If the calculated average throughput that you're driving the file system to is less than permitted, consider changing throughput to lower costs.

In some cases, your calculated average throughput during normal operations might be at or below the ratio of baseline throughput to storage capacity ratio for Bursting Throughput mode. That ratio is 50 MiB/s per TiB of data stored. In such cases, consider switching to Bursting Throughput mode. In

other cases, the calculated average throughput during normal operations might be above this ratio. In these cases, consider lowering the provisioned throughput to a point between your current provisioned throughput and the calculated average throughput during normal operations.

You can change the throughput mode of your file system using the AWS Management Console, the AWS CLI, or the EFS API. With the CLI, use the `update-file-system` action. With the EFS API, use the [UpdateFileSystem \(p. 260\)](#) operation.

**Note**

As previously mentioned, new file systems have an initial burst credit balance of 2.1 TB. With this starting balance, you can burst at 100 MB/s for 6.12 hours without spending any credits that you're earning from your storage. This starting formula is calculated as  $2.1 \times 1024 \times (1024/100/3600)$  to get 6.116 hours, rounded up to 6.12.

## On-Premises Performance Considerations

The Bursting Throughput model for Amazon EFS file systems remains the same whether accessed from your on-premises servers or your Amazon EC2 instances. However, when accessing Amazon EFS file data from your on-premises servers, the maximum throughput is also constrained by the bandwidth of the AWS Direct Connect connection.

Because of the propagation delay tied to data traveling over long distances, the network latency of an AWS Direct Connect connection between your on-premises data center and your Amazon VPC can be tens of milliseconds. If your file operations are serialized, the latency of the AWS Direct Connect connection directly impacts your read and write throughput. In essence, the volume of data you can read or write during a period of time is bounded by the amount of time it takes for each read and write operation to complete. To maximize your throughput, parallelize your file operations so that multiple reads and writes are processed by Amazon EFS concurrently. Standard tools like [GNU parallel](#) enable you to parallelize the copying of file data.

## Architecting for High Availability

To ensure continuous availability between your on-premises data center and your Amazon VPC, we recommend configuring two AWS Direct Connect connections. For more information, see [Step 4: Configure Redundant Connections with AWS Direct Connect](#) in the AWS Direct Connect User Guide.

To ensure continuous availability between your application and Amazon EFS, we recommend that your application be designed to recover from potential connection interruptions. In general, there are two scenarios for on-premises applications connected to an Amazon EFS file system; highly available and not highly available.

In the first, your application is highly available (HA) and uses multiple on-premises servers in its HA cluster. In this case, ensure that each on-premises server in the HA cluster connects to a mount target in a different Availability Zone (AZ) in your Amazon VPC. If your on-premises server can't access the mount target because the AZ in which the mount target exists becomes unavailable, your application should fail over to a server with an available mount target.

In the second, your application is not highly available, and your on-premises server can't access the mount target because the AZ in which the mount target exists becomes unavailable. In this case, your application should implement restart logic and connect to a mount target in a different AZ.

## 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. Due to 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.
- **Simultaneous Connections** – Amazon EFS file systems can be mounted on up to thousands of Amazon EC2 instances concurrently. If you can parallelize your application across more instances, you can drive higher throughput levels on your file system in aggregate across instances.
- **Request Model** – By enabling asynchronous writes to your file system, pending write operations are buffered on the Amazon EC2 instance before they are 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.

- **NFS Client Mount Settings** – Verify that you're using the recommended mount options as outlined in [Mounting EFS File Systems \(p. 61\)](#) and in [Additional Mounting Considerations \(p. 70\)](#). Amazon EFS supports the Network File System versions 4.0 and 4.1 (NFSv4) protocols when mounting your file systems on Amazon EC2 instances. NFSv4.1 provides better performance.

**Note**

You might want to increase the size of the read and write buffers for your NFS client to 1 MB when you mount your file system.

- **Amazon EC2 Instances** – Applications that perform a large number of read and write operations likely need more memory or computing capacity than applications that don't. When launching your Amazon EC2 instances, choose instance types that have the amount of these resources that your application needs. The performance characteristics of Amazon EFS file systems don't depend on the use of EBS-optimized instances.
- **Encryption** – Amazon EFS supports two forms of encryption, encryption in transit and encryption at rest. This option is for encryption at rest. Choosing to enable either or both types of encryption for your file system has a minimal effect on I/O latency and throughput.

For information about the Amazon EFS limits for total file system throughput, per-instance throughput, and operations per second in General Purpose performance mode, see [Amazon EFS Quotas and Limits \(p. 167\)](#).

## Related Topics

- [On-Premises Performance Considerations \(p. 96\)](#)
- [Amazon EFS Performance Tips \(p. 96\)](#)
- [Metering: How Amazon EFS Reports File System and Object Sizes \(p. 55\)](#)
- [Troubleshooting Amazon EFS \(p. 172\)](#)

# Data Protection for Amazon EFS

There are two options available for protecting your data by backing up your EFS file systems.

- AWS Backup service
- The EFS-to-EFS backup solution

AWS Backup is a simple and cost-effective way to back up your Amazon EFS file systems. AWS Backup is a unified backup service designed to simplify the creation, migration, restoration, and deletion of backups, while providing improved reporting and auditing. For more information, see [Using AWS Backup with Amazon EFS \(p. 98\)](#).

The EFS-to-EFS backup solution is suitable for all Amazon EFS file systems in all AWS Regions. It includes an AWS CloudFormation template that launches, configures, and runs the AWS services required to deploy this solution. This solution follows AWS best practices for security and availability. For more information, see [EFS-to-EFS Backup Solution](#) in AWS Answers.

## Using AWS Backup with Amazon EFS

AWS Backup is a simple and cost-effective way to protect your data by backing up your Amazon EFS file systems. AWS Backup is a unified backup service designed to simplify the creation, migration, restoration, and deletion of backups, while providing improved reporting and auditing. AWS Backup makes it easier to develop a centralized backup strategy for legal, regulatory, and professional compliance. AWS Backup also makes protecting your AWS storage volumes, databases, and file systems simpler by providing a central place where you can do the following:

- Configure and audit the AWS resources that you want to back up
- Automate backup scheduling
- Set retention policies
- Monitor all recent backup and restore activity

Amazon EFS is integrated with AWS Backup. You can use AWS Backup to set backup plans where you specify your backup frequency, when to back up, how long to retain backups, and a lifecycle policy for backups. You can then assign Amazon EFS file systems, or other AWS resources, to that backup plan.

## How AWS Backup Works with EFS File Systems

With AWS Backup, first you create a backup plan. The backup plan defines backup schedule, backup window, retention policy, lifecycle policy, and tags. You can create a backup plan using the [AWS Backup Management Console](#), the AWS CLI, or the AWS Backup API. As part of a backup plan, you can define the following:

- Schedule – When the backup occurs
- Backup window – The window of time in which the backup needs to start
- Lifecycle – When to move a recovery point to cold storage and when to delete it
- Backup vault – Used to organize recovery points created by the Backup rule.

After your backup plan is created, you assign the specific Amazon EFS file systems to the backup plan by using either tags or the Amazon EFS file system ID. After a plan is assigned, AWS Backup begins

automatically backing up the Amazon EFS file system on your behalf according to the backup plan that you defined. You can use the AWS Backup console to manage backup configurations or monitor backup activity. For more information, see the [AWS Backup Developer Guide](#).

**Note**

Sockets and named pipes are not supported, and are omitted from backups.

## 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. 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 Amazon 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.

## Performance

In general, you can expect the following backup rates with AWS Backup:

- 100 MB/s for file systems composed of mostly large files
- 500 files/s for file systems composed of mostly small files
- The maximum duration for a backup or a restore operation in AWS Backup is seven days.

Complete restore operations generally take longer than the corresponding backup.

Using AWS Backup doesn't consume accumulated burst credits, and it doesn't count against the General Purpose mode file operation limits. For more information, see [Quotas for Amazon EFS File Systems](#) (p. 169).

## 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 complete. 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 complete during the window.

Backups that don't complete 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](#).

## EFS Storage Classes

You can use AWS Backup to back up all data in an EFS file system, 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) storage class.

When you restore a recovery point, all files are restored to the Standard storage class. For more information on storage classes, see [EFS Storage Classes](#) (p. 50) and [EFS Lifecycle Management](#) (p. 51).

## On-Demand Backups

Using either the [AWS Backup Management Console](#) or the CLI, 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.

## Concurrent Backups

AWS Backup limits backups to one concurrent backup per resource. Therefore, scheduled or on-demand backups may fail if a backup job is already in progress. For more information about AWS Backup limits, see [AWS Backup Limits](#) in the *AWS Backup Developer Guide*.

## Restore a Recovery Point

Using either the [AWS Backup Management Console](#) or the CLI, you can restore a recovery point to a new EFS file system or to the source file system. You can perform a Complete restore, which restores the entire file system. Or, you can restore specific files and directories using a Partial restore. To restore a specific file or directory, you must specify the relative path related to the mount point. For example, if the file system is mounted to `/user/home/myname/efs` and the file path is `user/home/myname/efs/file1`, enter `/file1`. Paths are case sensitive and cannot contain special characters, wildcards and regex strings.

When you perform either a Complete or a Partial restore, your recovery point is restored to the restore directory, `aws-backup-restore_`*timestamp-of-restore*, which you can see at the root of the file system when the restore is complete. If you attempt multiple restores for the same path, several directories containing the restored items might exist. If the restore fails to complete, you can see the directory `aws-backup-failed-restore_`*timestamp-of-restore*. You need to manually delete the restore and failed\_restore directories when you are through using them.

### Note

For Partial restores to an existing EFS file system, AWS Backup restores the files and directories to a new directory under the file system root directory. The full hierarchy of the specified items is preserved in the recovery directory. For example, if directory A contains subdirectories B, C, and D, AWS Backup retains the hierarchical structure when A, B, C, and D are recovered.

After restoring a recovery point, data fragments that can't be restored to the appropriate directory are placed in the `aws-backup-lost+found` directory. Fragments might be moved to this directory if modifications are made to the file system while the backup is occurring.

# Amazon Elastic File System Walkthroughs

This section provides walkthroughs that you can use to explore Amazon EFS and test the end-to-end setup.

## Topics

- [Walkthrough: Create an Amazon EFS File System and Mount It on an Amazon EC2 Instance Using the AWS CLI \(p. 101\)](#)
- [Walkthrough: Set Up an Apache Web Server and Serve Amazon EFS Files \(p. 113\)](#)
- [Walkthrough: Create Writable Per-User Subdirectories and Configure Automatic Remounting on Reboot \(p. 118\)](#)
- [Walkthrough: Create and Mount a File System On-Premises with AWS Direct Connect and VPN \(p. 119\)](#)
- [Walkthrough: Mount a File System from a Different VPC \(p. 125\)](#)
- [Walkthrough: Enforcing Encryption on an Amazon EFS File System at Rest \(p. 130\)](#)
- [Walkthrough: Enable Root Squashing Using IAM Authorization for NFS Clients \(p. 132\)](#)

## Walkthrough: Create an Amazon EFS File System and Mount It on an Amazon EC2 Instance Using the AWS CLI

This walkthrough uses the AWS CLI to explore the Amazon EFS API. In this walkthrough, you create an Amazon EFS file system, mount it on an Amazon EC2 instance in your VPC, and test the setup.

### Note

This walkthrough is similar to the Getting Started exercise. In the [Getting Started \(p. 11\)](#) exercise, you use the console to create EC2 and Amazon EFS resources. In this walkthrough, you use the AWS CLI to do the same—primarily to familiarize yourself with the Amazon EFS API.

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

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

You add rules to these security groups to authorize appropriate inbound/outbound access. Doing this allows your EC2 instance to connect to the file system through the mount target by using a standard NFSv4.1 TCP port.

- An Amazon 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 [Amazon EFS: How It Works \(p. 3\)](#).

Then, you test the file system on your EC2 instance. The cleanup 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, EC2 resources, and Amazon EFS resources—must be in the same AWS Region.

## Before You Begin

- 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. For more information, see [Setting Up \(p. 9\)](#).
- 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 [Updating DNS Support for Your VPC](#) in the *Amazon VPC User Guide*.
  - The Internet gateway is attached to your VPC. For more information, see [Internet Gateways](#) 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 in Your VPC](#) 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 the following topics in the *AWS Command Line Interface User Guide*.

[Getting Set Up with the AWS Command Line Interface](#)

[Installing the AWS Command Line Interface](#)

[Configuring the AWS Command Line Interface](#)

2. Set profiles.

You store user credentials in the AWS CLI `config` file. The example CLI commands in this walkthrough 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
```

### Next Step

[Step 1: Create Amazon EC2 Resources \(p. 103\)](#)

## Step 1: Create Amazon 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 Amazon EFS file system on this instance in the next step.

### Topics

- [Step 1.1: Create Two Security Groups \(p. 103\)](#)
- [Step 1.2: Add Rules to the Security Groups to Authorize Inbound/Outbound Access \(p. 104\)](#)
- [Step 1.3: Launch an EC2 Instance \(p. 105\)](#)

## Step 1.1: Create Two Security Groups

In this section, you create security groups in your VPC for your EC2 instance and Amazon EFS mount target. Later in the walkthrough, you assign these security groups to an EC2 instance and an Amazon EFS mount target. For information about security groups, see [Security Groups for EC2-VPC](#) in the *Amazon EC2 User Guide 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 Amazon 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 Amazon EFS mount target (with which you associate these security groups later in this walkthrough).

## 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 Amazon 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 introductory information, see [Setting Up with Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.
    - For instructions to create a .pem file, see [Create a Key Pair](#) in the *Amazon EC2 User Guide for Linux Instances*.
  - 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 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 [Before You Begin \(p. 102\)](#).

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
```

### Next Step

[Step 2: Create Amazon EFS Resources \(p. 107\)](#)

## Step 2: Create Amazon EFS Resources

In this step, you do the following:

- Create an Amazon EFS file system.
- Enable lifecycle management.
- Create a mount target in the Availability Zone where you have your EC2 instance launched.

### Topics

- [Step 2.1: Create Amazon EFS File System \(p. 107\)](#)
- [Step 2.2: Enable Lifecycle Management \(p. 108\)](#)
- [Step 2.3: Create a Mount Target \(p. 108\)](#)

## Step 2.1: Create Amazon EFS File System

In this step, you create an Amazon 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 \  
--creation-token FileSystemForWalkthrough1 \  
--tags Key=Name,Value=SomeExampleNameValue \  
--region us-west-2 \  
--profile adminuser
```

You get the following response.

```
{  
  "OwnerId": "123456789abcd",  
  "CreationToken": "FileSystemForWalkthrough1",  
  "FileSystemId": "fs-c657c8bf",  
  "CreationTime": 1548950706.0,  
  "LifecycleState": "creating",  
  "NumberOfMountTargets": 0,  
  "SizeInBytes": {  
    "Value": 0,  
    "ValueInIA": 0,  
    "ValueInStandard": 0  
  },  
  "PerformanceMode": "generalPurpose",  
  "Encrypted": false,  
  "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 \(p. 108\)](#).

## Step 2.2: Enable Lifecycle Management

In this step, you enable lifecycle management on your file system in order to use the Infrequent Access storage class. To learn more, see [EFS Lifecycle Management \(p. 51\)](#) and [EFS Storage Classes \(p. 50\)](#).

### 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](#).

For this walkthrough, 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
```

## Next Step

[Step 3: Mount the Amazon EFS File System on the EC2 Instance and Test \(p. 109\)](#)

# Step 3: Mount the Amazon EFS File System on the EC2 Instance and Test

In this step, you do the following:

## Topics

- [Step 3.1: Gather Information \(p. 109\)](#)
- [Step 3.2: Install the NFS Client on Your EC2 Instance \(p. 110\)](#)
- [Step 3.3: Mount File System on Your EC2 Instance and Test \(p. 110\)](#)

## 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 Amazon 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:
  - To connect to your instance from a computer running macOS or Linux, specify the .pem file for your SSH command with the `-i` option and the path to your private key.
  - To connect to your instance from a computer running Windows, you can use either MindTerm or PuTTY. If you plan to use PuTTY, you need to install it and use the following procedure to convert the .pem file to a .ppk file.

For more information, see the following topics in the *Amazon EC2 User Guide for Linux Instances*:

- [Connecting to Your Linux Instance from Windows Using PuTTY](#)
  - [Connecting to Your Linux Instance Using SSH](#)
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 Amazon EC2 instance, you don't need to install `nfs-utils` because it is already included in the AMI by default.

## Step 3.3: Mount 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 Amazon 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,wsiz=1048576,hard,timeo=600,retrans=2,noresvport mount-  
target-ip:/ ~/efs-mount-point
```

3. Now that you have the Amazon 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 Amazon 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. For more information, see [Automatic Remounting on Reboot \(p. 119\)](#). If you are using an Auto Scaling group to launch EC2 instances, you can also set scripts in a launch configuration. For an example, see [Walkthrough: Set Up an Apache Web Server and Serve Amazon EFS Files \(p. 113\)](#).

### Next Step

[Step 4: Clean Up \(p. 111\)](#)

## 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 Amazon EFS resources (file system, mount target).

### To delete AWS resources created in this walkthrough

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

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

You can also delete EC2 resources using the console. For instructions, see [Terminating an Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

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 [Deleting a Security Group](#) in the *Amazon EC2 User Guide for Linux Instances*.

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
```



## Walkthrough: Set Up an Apache Web Server and Serve Amazon EFS Files

You can have EC2 instances running the Apache web server serving files stored on your Amazon EFS file system. It can be one EC2 instance, or if your application needs, you can have multiple EC2 instances serving files from your Amazon EFS file system. The following procedures are described.

- [Set up an Apache web server on an EC2 instance \(p. 113\).](#)
- [Set up an Apache web server on multiple EC2 instances by creating an Auto Scaling group \(p. 115\).](#) You can create multiple EC2 instances using Amazon EC2 Auto Scaling, an AWS service that allows you to increase or decrease the number of EC2 instances in a group according to your application needs. When you have multiple web servers, you also need a load balancer to distribute request traffic among them.

### Note

For both procedures, you create all resources in the US West (Oregon) Region (us-west-2).

## Single EC2 Instance Serving Files

Follow the steps to set up an Apache web server on one EC2 instance to serve files you create in your Amazon EFS file system.

1. Follow the steps in the Getting Started exercise so that you have a working configuration consisting of the following:
  - Amazon EFS file system
  - EC2 instance
  - File system mounted on the EC2 instance

For instructions, see [Getting Started with Amazon Elastic File System \(p. 11\)](#). As you follow the steps, write down the following:

- Public DNS name of the EC2 instance.
  - Public DNS name of the mount target created in the same Availability Zone where you launched the EC2 instance.
2. (Optional) You may choose to unmount the file system from the mount point you created in the Getting Started exercise.

```
$ sudo umount ~/efs-mount-point
```

In this walkthrough, you create another mount point for the file system.

3. On your EC2 instance, install the Apache web server and configure it as follows:
  - a. Connect to your EC2 instance and install the Apache web server.

```
$ sudo yum -y install httpd
```

- b. Start the service.

```
$ sudo service httpd start
```

- c. Create a mount point.

First note that the DocumentRoot in the `/etc/httpd/conf/httpd.conf` file points to `/var/www/html` (DocumentRoot `" /var/www/html"`).

You will mount your Amazon EFS file system on a subdirectory under the document root.

- i. Create a subdirectory `efs-mount-point` under `/var/www/html`.

```
$ sudo mkdir /var/www/html/efs-mount-point
```

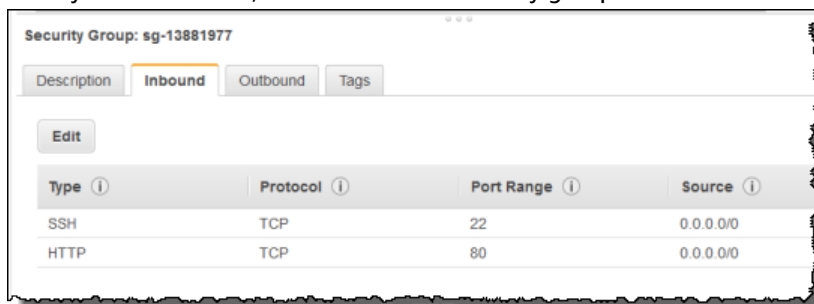
- ii. Mount your Amazon EFS file system. You need to update the following mount command using the EFS mount helper utility by providing your file system ID.

```
$ sudo mount -t efs fs-12345678:/ /var/www/html/efs-mount-point
```

4. Test the setup.

- a. Add a rule in the EC2 instance security group, which you created in the Getting Started exercise, to allow HTTP traffic on TCP port 80 from anywhere.

After you add the rule, the EC2 instance security group will have the following inbound rules.



Security Group: sg-13881977			
Description	Inbound	Outbound	Tags
Edit			
Type	Protocol	Port Range	Source
SSH	TCP	22	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0

For instructions, see [Creating Security Groups Using the AWS Management Console \(p. 28\)](#).

- b. Create a sample html file.

- i. Change directory.

```
$ cd /var/www/html/efs-mount-point
```

- ii. Make a subdirectory for `sampledir` and change the ownership. And change directory so you can create files in the `sampledir` subdirectory.

```
$ sudo mkdir sampledir
$ sudo chown ec2-user sampledir
$ sudo chmod -R o+r sampledir
$ cd sampledir
```

- iii. Create a sample `hello.html` file.

```
$ echo "<html><h1>Hello from Amazon EFS</h1></html>" > hello.html
```

- c. Open a browser window and enter the URL to access the file (it is the public DNS name of the EC2 instance followed by the file name). For example:

```
http://EC2-instance-public-DNS/efs-mount-point/sampledir/hello.html
```

Now you are serving web pages stored on an Amazon EFS file system.

**Note**

This setup does not configure the EC2 instance to automatically start `httpd` (web server) on boot, and also does not mount the file system on boot. In the next walkthrough, you create a launch configuration to set this up.

## Multiple EC2 Instances Serving Files

Follow the steps to serve the same content in your Amazon EFS file system from multiple EC2 instances for improved scalability or availability.

1. Follow the steps in the [Getting Started \(p. 11\)](#) exercise so that you have an Amazon EFS file system created and tested.

**Important**

For this walkthrough, you don't use the EC2 instance that you created in the Getting Started exercise. Instead, you launch new EC2 instances.

2. Create a load balancer in your VPC using the following steps.

1. Define a load balancer

In the **Basic Configuration** section, select your VPC where you also create the EC2 instances on which you mount the file system.

In the **Select Subnets** section, select all of the available subnets. For details, see the `cloud-config` script in the next section.

2. Assign security groups

Create a new security group for the load balancer to allow HTTP access from port 80 from anywhere, as shown following:

- Type: HTTP
- Protocol: TCP
- Port Range: 80
- Source: Anywhere (0.0.0.0/0)

**Note**

When everything works, you can also update the EC2 instance security group inbound rule access to allow HTTP traffic only from the load balancer.

3. Configure a health check

Set the **Ping Path** value to `/efs-mount-point/test.html`. The `efs-mount-point` is the subdirectory where you have the file system mounted. You add `test.html` page in it later in this procedure.

**Note**

Don't add any EC2 instances. Later, you create an Auto Scaling Group in which you launch EC2 instance and specify this load balancer.

For instructions to create a load balancer, see [Getting Started with Elastic Load Balancing](#) in the *Elastic Load Balancing User Guide*.

3. Create an Auto Scaling group with two EC2 instances. First, you create a launch configuration describing the instances. Then, you create an Auto Scaling group by specifying the launch configuration. The following steps provide configuration information that you specify to create an Auto Scaling group from the Amazon EC2 console.
  - a. Choose **Launch Configurations** under **AUTO SCALING** from the left hand navigation.
  - b. Choose **Create Auto Scaling group** to launch the wizard.
  - c. Choose **Create launch configuration**.
  - d. From **Quick Start**, select the latest version of the **Amazon Linux (HVM)** AMI. This is same AMI you used in [Step 2: Create Your EC2 Resources and Launch Your EC2 Instance \(p. 15\)](#) of the Getting Started exercise.
  - e. In the **Advanced** section, do the following:
    - For **IP Address Type**, choose **Assign a public IP address to every instance**.
    - Copy/paste the following script in the **User data** box.

You must update the script by providing values for the *file-system-id* and *aws-region* (if you followed the Getting Started exercise, you created the file system in the us-west-2 region).

In the script, note the following:

- The script installs the NFS client and the Apache web server.
- The echo command writes the following entry in the `/etc/fstab` file identifying the file system's DNS name and subdirectory on which to mount it. This entry ensures that the file gets mounted after each system reboot. Note that the file system's DNS name is dynamically constructed. For more information, see [Mounting on Amazon EC2 with a DNS Name \(p. 294\)](#).

```
file-system-ID.efs.aws-region.amazonaws.com:/ /var/www/html/efs-mount-point  
nfs4 defaults
```

- Creates `efs-mount-point` subdirectory and mounts the file system on it.
- Creates a `test.html` page so ELB health check can find the file (when creating a load balancer you specified this file as the ping point).

For more information about user data scripts, see [Adding User Data](#) in the *Amazon EC2 User Guide for Linux Instances*.

```
#cloud-config  
package_upgrade: true  
packages:  
- nfs-utils  
- httpd  
runcmd:  
- echo "$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-  
zone).file-system-id.efs.aws-region.amazonaws.com:/ /var/www/html/efs-mount-  
point nfs4 defaults" >> /etc/fstab  
- mkdir /var/www/html/efs-mount-point  
- mount -a  
- touch /var/www/html/efs-mount-point/test.html  
- service httpd start  
- chkconfig httpd on
```

- f. For **Assign a security group**, choose **Select an existing security group**, and then choose the security group you created for the EC2 instance.

---

When configuring the Auto Scaling group details, use the following information:

1. For **Group size**, choose **Start with 2 instances**. You will create two EC2 instances.
2. Select your VPC from the **Network** list.
3. Select a subnet in the same Availability Zone that you used when specifying the mount target ID in the User Data script when creating the launch configuration in the preceding step.
4. In the Advanced Details section
  - a. For **Load Balancing**, choose **Receive traffic from Elastic Load Balancer(s)**, and then select the load balancer you created for this exercise.
  - b. For **Health Check Type**, choose **ELB**.

Follow the instructions to create an Auto Scaling group at [Set Up a Scaled and Load-Balanced Application](#) in the *Amazon EC2 Auto Scaling User Guide*. Use the information in the preceding tables where applicable.

4. Upon successful creation of the Auto Scaling group, you have two EC2 instances with `nfs-utils` and the Apache web server installed. On each instance, verify that you have the `/var/www/html/efs-mount-point` subdirectory with your Amazon EFS file system mounted on it. For instructions to connect to an EC2 instance, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

**Note**

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

5. Create a sample page (`index.html`).
  - a. Change directory.

```
$ cd /var/www/html/efs-mount-point
```

- b. Make a subdirectory for `sampledir` and change the ownership. And change directory so you can create files in the `sampledir` subdirectory. If you followed the preceding [Single EC2 Instance Serving Files \(p. 113\)](#), you already created the `sampledir` subdirectory, so you can skip this step.

```
$ sudo mkdir sampledir
$ sudo chown ec2-user sampledir
$ sudo chmod -R o+r sampledir
$ cd sampledir
```

- c. Create a sample `index.html` file.

```
$ echo "<html><h1>Hello from Amazon EFS</h1></html>" > index.html
```

6. Now you can test the setup. Using the load balancer's public DNS name, access the `index.html` page.

```
http://load balancer public DNS Name/efs-mount-point/sampledir/index.html
```

The load balancer sends a request to one of the EC2 instances running the Apache web server. Then, the web server serves the file that is stored in your Amazon EFS file system.

# Walkthrough: Create Writable Per-User Subdirectories and Configure Automatic Remounting on Reboot

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

In this walkthrough, you first create a user "mike" on your EC2 instance. You then mount an Amazon EFS subdirectory onto user mike's home directory. The walkthrough also explains how to configure automatic remounting of subdirectories if the system reboots.

Suppose you have an Amazon EFS file system created and mounted on a local directory on your EC2 instance. Let's call it *EFSroot*.

## Note

You can follow the [Getting Started \(p. 11\)](#) exercise to create and mount an Amazon EFS file system on your EC2 instance.

In the following steps, you create a user (mike), create a subdirectory for the user (*EFSroot*/mike), make user mike the owner of the subdirectory, granting him full permissions, and finally mount the Amazon EFS subdirectory on the user's home directory (/home/mike).

### 1. Create user mike:

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

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

This also creates a home directory, /home/mike, for the user.

### 2. Create a subdirectory under *EFSroot* for user mike:

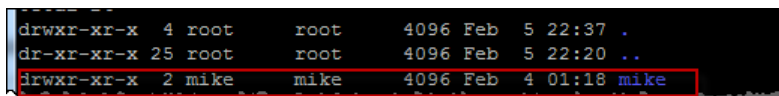
- Create subdirectory mike under *EFSroot*.

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

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

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

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



```
drwxr-xr-x  4 root    root    4096 Feb  5 22:37 .  
dr-xr-xr-x 25 root    root    4096 Feb  5 22:20 ..  
drwxr-xr-x  2 mike    mike    4096 Feb  4 01:18 mike
```

### 3. Use the `mount` command to mount the *EFSroot*/mike subdirectory onto mike's home directory.

```
$ 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 Amazon EFS file system root.

Now user mike's home directory is a subdirectory, writable by mike, in the Amazon EFS file system. If you unmount this mount target, the user can't access their EFS directory without remounting, which requires root permissions.

## Automatic Remounting on Reboot

You can use the file `fstab` to automatically remount your file system after any system reboots. For more information, see [Mounting Your Amazon EFS File System Automatically \(p. 64\)](#).

# Walkthrough: Create and Mount a File System On-Premises with AWS Direct Connect and VPN

This walkthrough uses the AWS Management Console to create and mount a file system on an on-premises client. You do so using either an AWS Direct Connect connection or a connection on an AWS virtual private network (VPN).

### Note

Using Amazon EFS with Microsoft Windows-based clients isn't supported.

### Topics

- [Before You Begin \(p. 120\)](#)
- [Step 1: Create Your Amazon Elastic File System Resources \(p. 120\)](#)
- [Step 2: Install the NFS Client \(p. 121\)](#)
- [Step 3: Mount the Amazon EFS File System on Your On-Premises Client \(p. 121\)](#)
- [Step 4: Clean Up Resources and Protect Your AWS Account \(p. 122\)](#)
- [Optional: Encrypting Data in Transit \(p. 123\)](#)

In this walkthrough, we assume that you already have an AWS Direct Connect or VPN connection. If you don't have one, you can begin the connection process now and come back to this walkthrough when your connection is established. For more information on AWS Direct Connect, see the [AWS Direct Connect User Guide](#). For more information on setting up a VPN connection, see [VPN Connections](#) in the *Amazon VPC User Guide*.

When you have an AWS Direct Connect or VPN connection, you create an Amazon 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.

### 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.

## Before You Begin

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 [Setting Up \(p. 9\)](#).

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 [Internet Gateways](#) 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 Amazon Elastic File System Resources

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

### To create your Amazon EFS file system

1. Open the Amazon EFS 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 check boxes 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 [Creating Mount Targets \(p. 22\)](#).
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 Amazon Elastic File System Resources \(p. 120\)](#).
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,wsiz=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
```

You can also mount your file system automatically by adding an entry to the `/etc/fstab` file. For more information, see [Mounting Your Amazon EFS File System Automatically \(p. 64\)](#).

#### 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 \(p. 178\)](#).

## 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 <https://console.aws.amazon.com/efs/>.
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 [Using the amazon-efs-utils Tools \(p. 35\)](#). 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
```

## Walkthrough: Mount a File System from a Different VPC

In this walkthrough, you set up an Amazon EC2 instance to mount an Amazon 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 [Using the amazon-efs-utils Tools \(p. 35\)](#).

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, Amazon 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

- [Before You Begin \(p. 126\)](#)
- [Step 1: Determine the Availability Zone ID of the EFS Mount Target \(p. 126\)](#)

- [Step 2: Determine the Mount Target IP Address \(p. 127\)](#)
- [Step 3: Add a Host Entry for the Mount Target \(p. 127\)](#)
- [Step 4: Mount Your File System Using the EFS Mount Helper \(p. 128\)](#)
- [Step 5: Clean Up Resources and Protect Your AWS Account \(p. 129\)](#)

## Before You Begin

In this walkthrough, we assume that you already 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 [Using the amazon-efs-utils Tools \(p. 35\)](#).
- 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 Transit Gateways](#) in the *Amazon VPC Transit Gateways Guide*.

## Step 1: Determine the Availability Zone ID of the EFS Mount Target

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 (AZ) 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 AZ ID. This requirement applies because AZ names can differ between accounts.

### To determine the AZ ID of the EC2 instance

1. Connect to your EC2 instance:
  - To connect to your instance from a computer running macOS or Linux, specify the `.pem` file for your SSH command. To do this, use the `-i` option and the path to your private key.
  - To connect to your instance from a computer running Windows, you can use either MindTerm or PuTTY. To use PuTTY, install it and convert the `.pem` file to a `.ppk` file.

For more information, see the following topics in the *Amazon EC2 User Guide for Linux Instances*:

- [Connecting to Your Linux Instance Using SSH](#)
  - [Connecting to Your Linux Instance from Windows Using PuTTY](#)
2. Determine the AZ 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 `curl -s
http://169.254.169.254/latest/meta-data/placement/availability-zone`
{
  "AvailabilityZones": [
```

```
{
  "State": "available",
  "ZoneName": "us-east-2b",
  "Messages": [],
  "ZoneId": "use2-az2",
  "RegionName": "us-east-2"
}
```

The AZ ID is returned in the `ZoneId` property, `use2-az2`.

## Step 2: Determine the Mount Target IP Address

Now that you know the AZ ID of the EC2 instance, you can now retrieve the IP address of the mount target that is in the same AZ ID.

### To determine the mount target IP address in the same AZ 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` AZ 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

- 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
```

## 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 IAM authorization or an EFS access point. For more information, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#) and [Working with Amazon EFS Access Points \(p. 161\)](#).

### 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 Amazon EFS file system, you can test it 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 ~/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 [Using /etc/fstab to Mount Automatically \(p. 66\)](#).

### 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 \(p. 178\)](#).

## Step 5: Clean Up Resources and Protect Your AWS Account

After you have finished this walkthrough, or if you don't want to explore the walkthroughs, make sure to take the following steps. These 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 <https://console.aws.amazon.com/efs/>.
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**.

# Walkthrough: Enforcing Encryption on an Amazon EFS File System at Rest

Following, you can find details about how to enforce encryption at rest using Amazon CloudWatch and AWS CloudTrail. This walkthrough is based upon the AWS whitepaper [Encrypt Data at Rest with Amazon EFS Encrypted File Systems](#).

## Note

Currently, you can't enforce encryption in transit.

## Enforcing Encryption at Rest

Your organization might require the encryption at rest of all data that meets a specific classification or that is associated with a particular application, workload, or environment. You can enforce policies for data encryption at rest for Amazon EFS file systems by using detective controls. These controls detect the creation of a file system and verify that encryption at rest is enabled.

If a file system that doesn't have encryption at rest is detected, you can respond in a number of ways. These range from deleting the file system and mount targets to notifying an administrator.

If you want to delete an unencrypted-at-rest file system but want to retain the data, first create a new encrypted-at-rest file system. Next, copy the data over to the new encrypted-at-rest file system. After the data is copied over, you can delete the unencrypted-at-rest file system.

## Detecting Files Systems That Are Unencrypted at Rest

You can create a CloudWatch alarm to monitor CloudTrail logs for the `CreateFileSystem` event. You can then trigger the alarm to notify an administrator if the file system that was created was unencrypted at rest.

## Create a Metric Filter

To create a CloudWatch alarm that is triggered when an unencrypted Amazon EFS file system is created, use the following procedure.

Before you begin, you must have an existing trail created that is sending CloudTrail logs to a CloudWatch Logs log group. For more information, see [Sending Events to CloudWatch Logs](#) in the *AWS CloudTrail User Guide*.

### To create a metric filter

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Logs**.
3. In the list of log groups, choose the log group that you created for CloudTrail log events.
4. Choose **Create Metric Filter**.
5. On the **Define Logs Metric Filter** page, choose **Filter Pattern** and then type the following:

```
{ ($.eventName = CreateFileSystem) && ($.responseElements.encrypted IS FALSE) }
```

6. Choose **Assign Metric**.
7. For **Filter Name**, type **UnencryptedFileSystemCreated**.
8. For **Metric Namespace**, type **CloudTrailMetrics**.
9. For **Metric Name**, type **UnencryptedFileSystemCreatedEventCount**.

10. Choose **Show advanced metric settings**.
11. For **Metric Value**, type 1.
12. Choose **Create Filter**.

## Create an Alarm

After you create the metric filter, use the following procedure to create an alarm.

### To create an alarm

1. On the **Filters** for the **Log\_Group\_Name** page, next to the **UnencryptedFileSystemCreated** filter name, choose **Create Alarm**.
2. On the **Create Alarm** page, set the following parameters:
  - For **Name**, type **Unencrypted File System Created**
  - For **Whenever**, do the following:
    - Set **is to** to **> = 1**
    - Set **for:** to **1** consecutive period(s).
  - For **Treat missing data as**, choose **good (not breaching threshold)**.
  - For **Actions**, do the following:
    - For **Whenever this alarm**, choose **State is ALARM**.
    - For **Send notification to**, choose **NotifyMe**, choose **New list**, and then type a unique topic name for this list.
    - For **Email list**, type in the email address where you want notifications sent. You should receive an email at this address to confirm that you created this alarm.
  - For **Alarm Preview**, do the following:
    - For **Period**, choose **1 Minute**.
    - For **Statistic**, choose **Standard** and **Sum**.
3. Choose **Create Alarm**.

## Test the Alarm for the Creation of Unencrypted File Systems

You can test the alarm by creating an unencrypted-at-rest file system, as follows.

### To test the alarm by creating an unencrypted-at-rest file system

1. Open the Amazon EFS console at <https://console.aws.amazon.com/efs>.
2. Choose **Create File System**.
3. From the **VPC** list, choose your default VPC.
4. Choose all the Availability Zones. Ensure that the default subnets, automatic IP addresses, and the default security groups are chosen. These are your mount targets.
5. Choose **Next Step**.
6. Name your file system and keep **Enable encryption** unchecked to create an unencrypted file system.
7. Choose **Next Step**.
8. Choose **Create File System**.

Your trail logs the `CreateFileSystem` operation and delivers the event to your CloudWatch Logs log group. The event triggers your metric alarm and CloudWatch Logs sends you a notification about the change.

# Walkthrough: Enable Root Squashing Using IAM Authorization for NFS Clients

In this walkthrough, you configure Amazon EFS to deny root access to your Amazon 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. For more information about IAM authorization for NFS clients in EFS, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#).

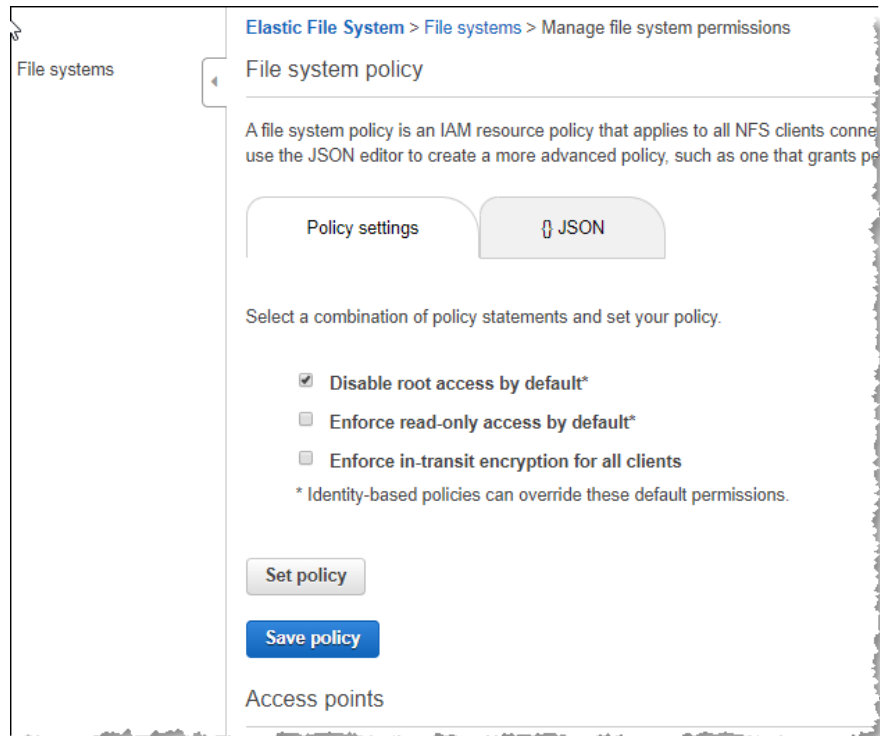
To do this requires configuring two IAM permissions policies, as follows:

- 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 Amazon EC2 instance profile. For more information about Amazon EC2 instance profiles, see [Using 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 EFS, see [AWS Managed \(Predefined\) Policies for Amazon EFS \(p. 146\)](#).

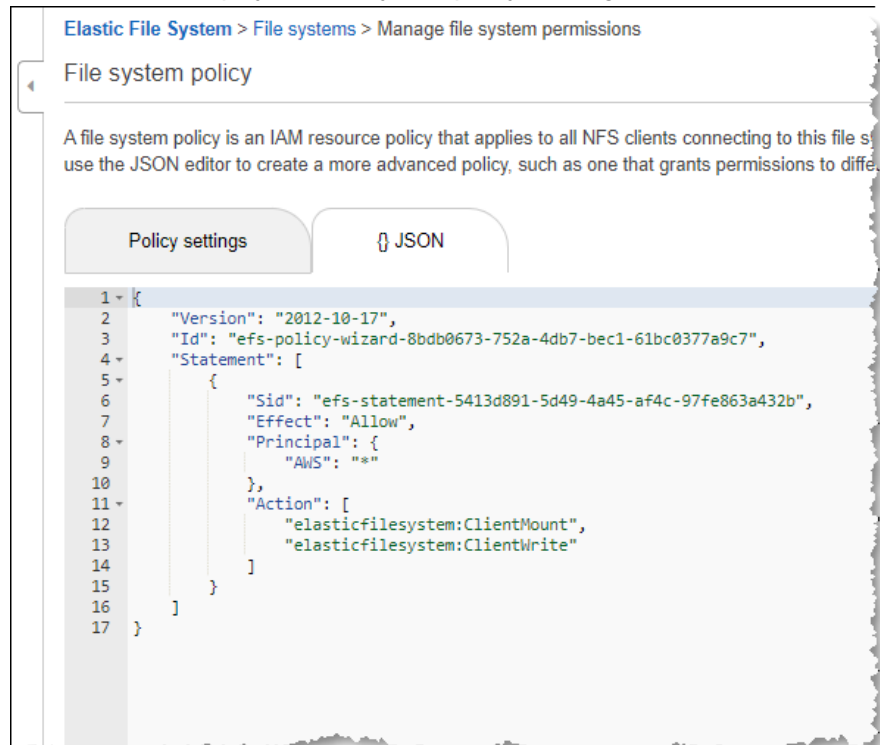
To enable root squashing using IAM authorization for NFS clients, use the following procedures.

## To disable root access to the file system

1. Open the Amazon Elastic File System console at <https://console.aws.amazon.com/efs/>.
2. On the **File systems** page, choose the file system that you want to enable root squashing on.
3. For **Actions**, choose **Manage client access**. The **Manage file system permission** page appears.
4. Choose **Disable root access**, and then choose **Set policy**.



The **JSON** tab displays the file system policy that is generated.



This policy implicitly denies root access for all AWS principals. With this file system policy in place, you grant root access by using an IAM identity-based policy. Anonymous NFS clients are also denied root access because identity-based policies are not available for anonymous NFS clients.

5. 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Resource": "*",
      "Effect": "Allow",
      "Action": "elasticfilesystem:Client*"
    }
  ],
  "Condition": {}
}
```

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 and the `amazon-efs-utils` package, see [Using the amazon-efs-utils Tools \(p. 35\)](#).
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 \(p. 62\)](#).

# Security in Amazon EFS

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to Amazon Elastic File System, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon EFS. The following topics show you how to configure Amazon EFS to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon EFS resources.

Following, you can find a description of security considerations for working with Amazon EFS. There are four levels of access control to consider for Amazon EFS file systems, with different mechanisms used for each.

## Topics

- [Data Encryption in EFS \(p. 135\)](#)
- [Identity and Access Management for Amazon EFS \(p. 139\)](#)
- [Controlling Network Access to Amazon EFS File Systems for NFS Clients \(p. 154\)](#)
- [Working with Users, Groups, and Permissions at the Network File System \(NFS\) Level \(p. 158\)](#)
- [Working with Amazon EFS Access Points \(p. 161\)](#)
- [Logging and Monitoring in Amazon EFS \(p. 164\)](#)
- [Compliance Validation for Amazon Elastic File System \(p. 165\)](#)
- [Resilience in Amazon Elastic File System \(p. 165\)](#)
- [Amazon Elastic File System Network Isolation \(p. 166\)](#)

## Data Encryption in 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.

## When to Use Encryption

If your organization is subject to corporate or regulatory policies that require encryption of data and metadata at rest, we recommend creating an encrypted file system mounting your file system using encryption of data in transit.

## Related Topics

For more information on encryption with Amazon EFS, see these related topics:

- [Creating Resources for Amazon EFS \(p. 18\)](#)
- [Managing Access to Encrypted File Systems \(p. 59\)](#)
- [Amazon EFS Performance Tips \(p. 96\)](#)
- [Amazon EFS API Permissions: Actions, Resources, and Conditions Reference \(p. 148\)](#)
- [Amazon EFS Log File Entries for Encrypted-at-Rest File Systems \(p. 88\)](#)
- [Troubleshooting Encryption \(p. 182\)](#)

### Topics

- [Encrypting Data at Rest \(p. 136\)](#)
- [Encrypting Data in Transit \(p. 138\)](#)

## Encrypting Data at Rest

As with unencrypted file systems, you can create encrypted file systems through 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.

You can monitor whether encryption at rest is being used for Amazon EFS file systems by using Amazon CloudWatch and AWS CloudTrail to detect the creation of a file system and verify that encryption is enabled. For more information, see [Walkthrough: Enforcing Encryption on an Amazon EFS File System at Rest \(p. 130\)](#). You can enforce encryption in transit on file systems using a file system policy. For more information, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#).

### 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.

## Encrypting a File System at Rest Using the Console

You can choose to enable encryption at rest for a file system when you create it. The following procedure describes how to enable encryption for a new file system when you create it from the console.

### To encrypt a new file system on the console

1. Open the Amazon Elastic File System console at <https://console.aws.amazon.com/efs/>.
2. Choose **Create file system** to open the file system creation wizard.
3. For **Step 1: Configure network access**, choose your VPC, create your mount targets, and then choose **Next Step**.
4. For **Step 2: Configure file system settings**, add any tags, enable Lifecycle management, choose your throughput and performance modes, and Enable encryption to encrypt your file system, and then choose **Next Step**.
5. For **Step 3. Configure client access**, you can choose a preconfigured **File system policy**, or use the JSON editor in the **JSON** tab to create a more advanced policy. For more information, see [Using IAM to Control NFS Access to Amazon EFS \(p. 150\)](#). Choose **Next Step**.
6. For **Step 4: Review and create**, review your settings, and choose **Create File System**.



You now have a new encrypted-at-rest file system.

## 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 [Cryptography Basics](#) in the *AWS Key Management Service Developer Guide*.

## How Amazon EFS Uses AWS KMS

Amazon EFS integrates with AWS Key Management Service (AWS KMS) for key management. Amazon EFS uses customer master keys (CMKs) to encrypt your file system in the following way:

- **Encrypting metadata at rest** – Amazon EFS uses the AWS managed CMK 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 CMK used to encrypt and decrypt file data (that is, the contents of your files). You can enable, disable, or revoke grants on this CMK. This CMK can be one of the two following types:
  - **AWS managed CMK for Amazon EFS** – This is the default CMK, `aws/elasticfilesystem`. You're not charged to create and store a CMK, but there are usage charges. To learn more, see [AWS Key Management Service pricing](#).
  - **Customer-managed CMK** – This is the most flexible master key to use, because you can configure its key policies and grants for multiple users or services. For more information on creating CMKs, see [Creating Keys](#) in the *AWS Key Management Service Developer Guide*.

If you use a customer-managed CMK as your master 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 CMK, you can choose when to disable, re-enable, delete, or revoke access to your CMK at any time. For more information, see [Disabling, Deleting, or Revoking Access to the CMK for a File System](#) (p. 59).

### Important

Amazon EFS accepts only symmetric CMKs. You cannot use asymmetric CMKs 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](#) (p. 88).

## Amazon EFS Key Policies for AWS KMS

Key policies are the primary way to control access to CMKs. For more information on key policies, see [Using Key Policies in AWS KMS](#) in the *AWS Key Management Service Developer Guide*. The following list describes all the AWS KMS-related permissions 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 master key (CMK), 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 CMK. 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 [Using Grants](#) 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 master 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 master key** list. We recommend using this permission to provide the best user experience. This permission is included in the default key policy.

## Encrypting Data in Transit

You can encrypt data in transit using an Amazon EFS file system, without needing to modify your applications.

### Encrypting Data in Transit with TLS

Enabling encryption of data in transit for your Amazon 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 with the EFS Mount Helper \(p. 61\)](#).

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.

#### To mount your Amazon EFS file system with the mount helper with encryption of data in transit enabled

1. Access the terminal for your instance through Secure Shell (SSH), and log in with the appropriate user name. For more information on how to do this, see [Connecting to Your Linux Instance Using SSH](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. Run the following command to mount your file system.

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

### How Encrypting in Transit Works

To enable encryption of data in transit, you connect to Amazon EFS using TLS. We recommend using the mount helper because it's the simplest option.

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 the mount helper

1. Download and install stunnel, and note the port that the application is listening on.
2. Run stunnel to connect to your Amazon 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 mount helper listens on local ports 20049 and 20449, 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 Amazon 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 \(p. 37\)](#).

For issues with encryption, see [Troubleshooting Encryption \(p. 182\)](#).

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,wsiz=1048576,namlen=255,hard,proto=tcp,port=20127,timeo=600,retra
```

When mounting with TLS and the Amazon EFS mount helper, you are reconfiguring your NFS client to mount to a local port. The mount helper starts a client stunnel process that is listening on this local port, and stunnel is opening an encrypted connection to EFS using TLS. The EFS mount helper is responsible for setting up and maintaining this encrypted connection and associated configuration.

To determine which Amazon 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 Amazon EFS file system is unmounted. If for some reason a stunnel process is terminated unexpectedly, the watchdog process restarts it.

## Identity and Access Management for Amazon EFS

Access to Amazon EFS requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an Amazon EFS file system or an Amazon EC2 instance. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and Amazon EFS to help secure your resources by controlling who can access them.

- [Authentication \(p. 139\)](#)
- [Access Control \(p. 141\)](#)

### Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. This is your *AWS account root user*. Its credentials provide complete access to all of your AWS resources.

**Important**

For security reasons, we recommend that you use the root user only to create an *administrator*, which is an *IAM user* with full permissions to your AWS account. You can then use this administrator user to create other IAM users and roles with limited permissions. For more information, see [IAM Best Practices](#) and [Creating an Admin User and Group](#) in the *IAM User Guide*.

- **IAM user** – An [IAM user](#) is simply an identity within your AWS account that has specific custom permissions (for example, permissions to create a file system in Amazon EFS). You can use an IAM user name and password to sign in to secure AWS web pages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. Amazon EFS supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

- **IAM role** – An [IAM role](#) is another IAM identity that you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
  - **Federated user access** – Instead of creating an IAM user, you can use preexisting user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.
  - **Cross-account administration** – You can use an IAM role in your account to grant another AWS account permissions to administer your account's Amazon EFS resources. For an example, see [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles](#) in the *IAM User Guide*. You can't mount Amazon EFS file systems from across VPCs or accounts. For more information, see [Managing File System Network Accessibility](#) (p. 41).
  - **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.
  - **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications running on an EC2 instance and making 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 running on the EC2 instance to get temporary credentials. For more information, see [Using Roles for Applications on Amazon EC2](#) in the *IAM User Guide*.

## Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you can't create or access Amazon Elastic File System resources. For example, you must have permissions to create an Amazon EFS file system.

The following sections describe how to manage permissions for Amazon EFS. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Your Amazon EFS Resources](#) (p. 141)
- [Controlling Access to the EFS API](#) (p. 144)

## Overview of Managing Access Permissions to Your Amazon EFS Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles). Some services, including Amazon EFS, also support attaching permissions policies to resources.

### Note

An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

### Topics

- [Amazon EFS Resources and Operations](#) (p. 141)
- [Understanding Resource Ownership](#) (p. 142)
- [Managing Access to Resources](#) (p. 142)
- [Specifying Policy Elements: Actions, Effects, and Principals](#) (p. 143)
- [Specifying Conditions in a Policy](#) (p. 144)

## Amazon EFS Resources and Operations

In Amazon EFS, the primary resource is a *file system*. Amazon EFS also supports additional resource types, the *mount target* and *access point*. However, for Amazon EFS, you can create mount targets and access points only in the context of an existing file system. Mount targets and access points are referred to as *subresources*.

These resources and subresources have unique Amazon Resource Names (ARNs) associated with them as shown in the following table.

Amazon EFS provides a set of operations to work with Amazon EFS resources. For a list of available operations, see [Amazon EFS Actions](#) (p. 186) and [EFS Actions for NFS Clients](#) (p. 150).

## Understanding Resource Ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the [principal entity](#) (that is, the root account, an IAM user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a file system, your AWS account is the owner of the resource (in Amazon EFS, the resource is the file system).
- If you create an IAM user in your AWS account and grant permissions to create a file system to that user, the user can create a file system. However, your AWS account, to which the user belongs, owns the file system resource.
- If you create an IAM role in your AWS account with permissions to create a file system, anyone who can assume the role can create a file system. Your AWS account, to which the role belongs, owns the file system resource.

## Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

### Note

This section discusses using IAM in the context of Amazon EFS. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [IAM Policy Reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies) and policies attached to a resource are referred to as *resource-based* policies. Amazon EFS supports both identity-based policies and resource-based policies.

### Topics

- [Identity-Based Policies \(IAM Policies\)](#) (p. 142)
- [Resource-Based Policies](#) (p. 143)

## Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities to control access to the EFS API or to control NFS client access. For example, to grant a user permissions to create an Amazon EFS resource, such as a file system, you can attach a permissions policy to a user or group that the user belongs to.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

The following is an example policy that allows a user to perform the `CreateFileSystem` action for your AWS account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1EFSpermissions",
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:CreateFileSystem",
        "elasticfilesystem:CreateMountTarget"
      ]
    }
  ]
}
```

```
    "Resource": "arn:aws:elasticfilesystem:us-west-2:account-id:file-system/*"
  },
  {
    "Sid": "Stmnt2EC2permissions",
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeSubnets",
      "ec2:CreateNetworkInterface",
      "ec2:DescribeNetworkInterfaces"
    ],
    "Resource": "*"
  }
]
```

For more information about using identity-based policies with Amazon EFS, see [Controlling Access to the EFS API \(p. 144\)](#). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

## Resource-Based Policies

You can use file system policies to control API access and NFS client access to the file system. Amazon EFS supports a resource-based policy for file systems, called a `FileSystemPolicy`. Using an EFS `FileSystemPolicy` you can specify who has access to the file system and what actions they can perform on it. Using file system policies provides you an easy way to control access to your file systems, and lets you grant usage permission to other accounts on a per-file system basis. The following file system policy grants `ClientMount`, or read-only, permissions to all AWS IAM principals.

```
{
  "Version": "2012-10-17",
  "Id": "read-only-example-policy02",
  "Statement": [
    {
      "Sid": "efs-statement-example02",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "elasticfilesystem:ClientMount"
      ]
    }
  ]
}
```

## Specifying Policy Elements: Actions, Effects, and Principals

For each Amazon EFS resource (see [Amazon EFS Resources and Operations \(p. 141\)](#)), the service defines a set of actions API operations (see [Actions \(p. 186\)](#) and [EFS Actions for NFS Clients \(p. 150\)](#)) that you can grant permissions for. For the Amazon EFS file system resource, example actions are: `CreateFileSystem`, `DeleteFileSystem`, and `DescribeFileSystems`. Performing an API operation can require permissions for more than one action.

The following are the most basic policy elements:

- **Resource** – In resource-based policies (file system policies), the resource that the policy is attached to is the implicit resource. For identity-based policies, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For more information, see [Amazon EFS Resources and Operations \(p. 141\)](#).



- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, depending on the specified Effect, `elasticfilesystem:CreateFileSystem` either allows or denies the user permissions to perform the Amazon EFS `CreateFileSystem` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only).

To learn more about IAM policy syntax and descriptions, see [IAM Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the Amazon EFS API actions, see [Amazon EFS API Permissions: Actions, Resources, and Conditions Reference](#) (p. 148).

For a table showing all of the Amazon EFS NFS client actions, see [Using IAM to Control NFS Access to Amazon EFS](#) (p. 150).

## Specifying Conditions in a Policy

When you grant permissions, you can use the IAM policy language to specify the conditions when a policy should take effect. For more information about specifying conditions in a policy, see [IAM JSON Policy Elements: Condition](#) in the *IAM User Guide*.

There are both EFS-specific and AWS-wide condition keys that you can use as appropriate. For a complete list of AWS-wide keys, see [Available Keys for Conditions](#) in the *IAM User Guide*. For a complete list of EFS-specific condition keys, see [EFS Condition Keys for NFS Clients](#) (p. 151).

### Note

Do not use the `aws:SourceIp` AWS-wide condition for the `CreateMountTarget`, `DeleteMountTarget`, `DescribeMountTargetSecurityGroups`, or `ModifyMountTargetSecurityGroup` actions. Amazon EFS provisions mount targets by using its own IP address, not the IP address of the originating request.

## Controlling Access to the EFS API

You can use both IAM identity policies and resource policies to grant permissions to perform API operations on Amazon EFS resources. This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) because this is the most common way of controlling access to the Amazon EFS API.

### Important

We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon EFS resources. For more information, see [Overview of Managing Access Permissions to Your Amazon EFS Resources](#) (p. 141).

The sections in this topic cover the following:

- [Permissions Required to Use the Amazon EFS Console](#) (p. 145)
- [AWS Managed \(Predefined\) Policies for Amazon EFS](#) (p. 146)
- [Customer Managed Policy Examples](#) (p. 146)

The following shows an example of a permissions policy.



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid" : "AllowFileSystemPermissions",
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:CreateFileSystem",
        "elasticfilesystem:CreateMountTarget"
      ],
      "Resource": "arn:aws:elasticfilesystem:us-west-2:account-id:file-system/*"
    },
    {
      "Sid" : "AllowEC2Permissions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSubnets",
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces"
      ],
      "Resource": "*"
    }
  ]
}
```

The policy has two statements:

- The first statement grants permissions for two Amazon EFS actions (`elasticfilesystem:CreateFileSystem` and `elasticfilesystem:CreateMountTarget`) on a resource using the Amazon Resource Name (ARN) for the file system. The ARN specifies a wild card character (\*) because you don't know the file system ID until after you create a file system.
- The second statement grants permissions for some of the Amazon EC2 actions because the `elasticfilesystem:CreateMountTarget` action in the first statement requires permissions for specific Amazon EC2 actions. Because these Amazon EC2 actions don't support resource-level permissions, the policy specifies the wild card character (\*) as the `Resource` value instead of specifying a file system ARN.

The policy doesn't specify the `Principal` element because in an identity-based policy you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permissions policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon EFS API actions and the resources that they apply to, see [Amazon EFS API Permissions: Actions, Resources, and Conditions Reference](#) (p. 148).

## Permissions Required to Use the Amazon EFS Console

The permissions reference table lists the Amazon EFS API operations and shows the required permissions for each operation. For more information about Amazon EFS API operations, see [Amazon EFS API Permissions: Actions, Resources, and Conditions Reference](#) (p. 148).

To use the Amazon EFS console, you need to grant permissions for additional actions as shown in the following permissions policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid" : "Stmt1AdditionalEC2PermissionsForConsole",
```

```
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcAttribute"
    ],
    "Resource": "*"
  },
  {
    "Sid" : "Stmnt2AdditionalKMSPermissionsForConsole",
    "Effect": "Allow",
    "Action": [
        "kms:ListAliases",
        "kms:DescribeKey"
    ],
    "Resource": "*"
  }
]
```

The Amazon EFS console needs these additional permissions for the following reasons:

- Permissions for the Amazon EFS actions enable the console to display Amazon EFS resources in the account.
- The console needs permissions for the `ec2` actions to query Amazon EC2 so it can display Availability Zones, VPCs, security groups, and account attributes.
- The console needs permissions for the `kms` actions to create an encrypted file system. For more information on encrypted file systems, see [Data Encryption in EFS \(p. 135\)](#).

## AWS Managed (Predefined) Policies for Amazon EFS

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon EFS:

- **AmazonElasticFileSystemReadOnlyAccess** – Grants read-only access to Amazon EFS resources.
- **AmazonElasticFileSystemFullAccess** – Grants full access to Amazon EFS resources.

### Note

You can review these permissions policies by signing in to the IAM console and searching for specific policies there.

You can also create your own custom IAM policies to allow permissions for Amazon EFS API actions. You can attach these custom policies to the IAM users or groups that require those permissions.

## Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various Amazon EFS actions. These policies work when you are using AWS SDKs or the AWS CLI. When you are using the console, you need to grant additional permissions specific to the console, which is discussed in [Permissions Required to Use the Amazon EFS Console \(p. 145\)](#).

### Note

All examples use the us-west-2 Region and contain fictitious account IDs.

### Examples

- [Example 1: Allow a User to Create a Mount Target and Tags on an Existing File System \(p. 147\)](#)
- [Example 2: Allow a User to Perform All Amazon EFS Actions \(p. 147\)](#)

## Example 1: Allow a User to Create a Mount Target and Tags on an Existing File System

The following permissions policy grants the user permissions to create mount targets and tags on a particular file system in the us-west-2 region. To create mount targets, permissions for specific Amazon EC2 actions are also required and are included in the permissions policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid" : "Stmt1CreateMountTargetAndTag",
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:CreateMountTarget",
        "elasticfilesystem:DescribeMountTargets",
        "elasticfilesystem:CreateTags",
        "elasticfilesystem:DescribeTags"
      ],
      "Resource": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-system/file-system-ID"
    },
    {
      "Sid" : "Stmt2AdditionalEC2PermissionsToCreateMountTarget",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSubnets",
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces"
      ],
      "Resource": "*"
    }
  ]
}
```

## Example 2: Allow a User to Perform All Amazon EFS Actions

The following permissions policy uses a wild card character ("elasticfilesystem:\*") to allow all Amazon EFS actions in the us-west-2 region. Because some of the Amazon EFS actions also require permissions for Amazon EC2 actions, the policy also grants permissions for all those actions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid" : "Stmt1PermissionForAllEFSActions",
      "Effect": "Allow",
      "Action": "elasticfilesystem:*",
      "Resource": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-system/*"
    },
    {
      "Sid" : "Stmt2RequiredEC2PermissionsForAllEFSActions",
      "Effect": "Allow",

```

```

    "Action": [
      "ec2:DescribeSubnets",
      "ec2:CreateNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2>DeleteNetworkInterface",
      "ec2:ModifyNetworkInterfaceAttribute",
      "ec2:DescribeNetworkInterfaceAttribute"
    ],
    "Resource": "*"
  }
]
}
```

## Amazon EFS API Permissions: Actions, Resources, and Conditions Reference

When you are setting up [Access Control \(p. 141\)](#) and writing a permissions policy that you can attach to an IAM identity (identity-based policies), you can use the following list as a reference. The list includes each Amazon EFS API operation, the corresponding actions for which you can grant permissions to perform the action, and the AWS resource for which you can grant the permissions. You specify the actions in the policy's `Action` field, and you specify the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your Amazon EFS policies to express conditions. For a complete list of AWS-wide keys, see [Available Keys](#) in the *IAM User Guide*.

### Note

To specify an action, use the `elasticfilesystem:` prefix followed by the API operation name (for example, `elasticfilesystem:CreateFileSystem`).

### Amazon EFS API and Required Permissions for Actions

#### CreateFileSystem (p. 192)

**Action or Actions:** `elasticfilesystem:CreateFileSystem`

**Resource:** `arn:aws:elasticfilesystem:region:account-id:file-system/*`

#### CreateMountTarget (p. 200)

**Action or Actions:** `elasticfilesystem:CreateMountTarget`, `ec2:DescribeSubnets`, `ec2:DescribeNetworkInterfaces`, `ec2:CreateNetworkInterface`

**Resource:** `arn:aws:elasticfilesystem:region:account-id:file-system/file-system-id`

#### CreateTags (p. 207)

**Action or Actions:** `elasticfilesystem:CreateTags`

**Resource:** `arn:aws:elasticfilesystem:region:account-id:file-system/file-system-id`

#### DeleteFileSystem (p. 212)

**Action or Actions:** `elasticfilesystem>DeleteFileSystem`

**Resource:** `arn:aws:elasticfilesystem:region:account-id:file-system/file-system-id`

#### DeleteMountTarget (p. 216)

**Action or Actions:** `elasticfilesystem>DeleteMountTarget`, `ec2>DeleteNetworkInterface`

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[DeleteTags \(p. 219\)](#)

**Action or Actions:** elasticfilesystem:DeleteTags

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[DescribeFileSystems \(p. 227\)](#)

**Action or Actions:** elasticfilesystem:DescribeFileSystems

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*, arn:aws:elasticfilesystem:*region*:*account-id*:file-system/\*

[DescribeLifecycleConfiguration \(p. 231\)](#)

**Action or Actions:** elasticfilesystem:DescribeLifecycleConfiguration

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[DescribeMountTargetSecurityGroups \(p. 238\)](#)

**Action or Actions:** elasticfilesystem:DescribeMountTargetSecurityGroups, ec2:DescribeNetworkInterfaceAttribute

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[DescribeMountTargets \(p. 234\)](#)

**Action or Actions:** elasticfilesystem:DescribeMountTargets

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[DescribeTags \(p. 241\)](#)

**Action or Actions:** elasticfilesystem:DescribeTags

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[ModifyMountTargetSecurityGroups \(p. 246\)](#)

**Action or Actions:** elasticfilesystem:ModifyMountTargetSecurityGroups, ec2:ModifyNetworkInterfaceAttribute

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[PutFileSystemPolicy \(p. 249\)](#)

**Action or Actions:** elasticfilesystem:PutFileSystemPolicy

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[PutLifecycleConfiguration \(p. 252\)](#)

**Action or Actions:** elasticfilesystem:PutLifecycleConfiguration

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[TagResource \(p. 256\)](#)

**Action or Actions:** elasticfilesystem:TagResource

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[UntagResource \(p. 258\)](#)

**Action or Actions:** elasticfilesystem:UntagResource

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

[UpdateFileSystem \(p. 260\)](#)

**Action or Actions:** elasticfilesystem:UpdateFileSystem,  
elasticfilesystem:UpdateFileSystem

**Resource:** arn:aws:elasticfilesystem:*region*:*account-id*:file-system/*file-system-id*

## Using IAM to Control NFS Access to Amazon EFS

You can use both IAM identity policies and resource policies to control NFS 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.

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 \(p. 82\)](#).

### Important

You must use the EFS mount helper to mount your Amazon EFS file systems in order to use IAM authorization to control access by NFS clients. For more information, see [Mounting with IAM Authorization \(p. 62\)](#).

## Default EFS File System Policy

The default EFS file system policy grants full access to any NFS client that can connect to the file system. The default policy is in effect whenever a user-configured file system policy doesn't exist, including at file system creation. Whenever the default file system policy is in effect, a [DescribeFileSystemPolicy \(p. 224\)](#) API operation returns a PolicyNotFound response.

## EFS Actions for NFS Clients

You can specify the following actions for NFS clients on a file system using a file system policy.

Action	Description
elasticfilesystem:ClientMount	Provides an NFS client read-only access to a file system.
elasticfilesystem:ClientWrite	Provides an NFS client with write permissions on a file system.
elasticfilesystem:ClientRootAccess	Provides an NFS client the ability to use the root user when accessing a file system.

## EFS Condition Keys for NFS Clients

To express conditions, you use predefined condition keys. Amazon EFS has the following predefined condition keys for NFS clients.

EFS Condition Key	Description	Operator
aws:SecureTransport	Use this key to require NFS clients to use TLS when connecting to an EFS file system.	Boolean
elasticfilesystem:AccessPointArn	Use this key to require NFS clients to connect to a specific EFS access point that the client is connecting to.	

## File System Policy Examples

In this section, you can find example file system policies that grant or deny permissions for various Amazon EFS actions. For information about the elements of a resource-based policy, see [Specifying Policy Elements: Actions, Effects, and Principals \(p. 143\)](#).

### 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 still 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*.

### Example 1: Grant Read and Write Access to all IAM Principals

This example EFS file system policy has the following characteristics:

- The effect is `Allow`.
- The principal is set to `"*"`, all IAM entities.
- The action is set to `ClientMount`, `ClientWrite`, and `ClientRootAccess`.
- The condition for granting permissions is set to `SecureTransport`—only NFS clients using TLS to connect to the file system are granted access.

```
{
  "Version": "2012-10-17",
  "Id": "ExamplePolicy01",
  "Statement": [
    {
      "Sid": "ExampleStatement01",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "elasticfilesystem:ClientMount",
        "elasticfilesystem:ClientWrite"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

## Example 2: Grant Read-Only Access

The following file system policy only grants `ClientMount`, or read-only, permissions to all IAM principals.

```
{
  "Version": "2012-10-17",
  "Id": "read-only-example-policy02",
  "Statement": [
    {
      "Sid": "efs-statement-example02",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "elasticfilesystem:ClientMount"
      ]
    }
  ]
}
```

To learn how to set additional file system policies, including denying root access to all IAM principals, except for a specific management workstation, see [Walkthrough: Enable Root Squashing Using IAM Authorization for NFS Clients](#) (p. 132).

## Example 3: Grant Access to a 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 EFS access points, see [Working with Amazon EFS Access Points](#) (p. 161).

```
{
  "Version": "2012-10-17",
  "Id": "access-point-example03",
  "Statement": [
    {
      "Sid": "access-point-statement-example03",
      "Effect": "Allow",
      "Principal": {"arn:aws::account_id:role/myapp"},
      "Action": "elasticfilesystem:Client*",
      "Condition": {
        "StringEquals": {
          "elasticfilesystem:AccessPointArn": "arn:aws::account_id/access-
point/access_point_id"
        }
      }
    }
  ]
}
```

## Using the Amazon EFS Service-Linked Role

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 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 \(p. 84\)](#).

## 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 following services to assume the role:

- `elasticfilesystem.amazonaws.com`

The role permissions policy allows Amazon EFS to complete the following actions:

- `ec2:CreateNetworkInterface`
- `ec2:DeleteNetworkInterface`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSubnets`
- `ec2:DescribeNetworkInterfaceAttribute`
- `ec2:ModifyNetworkInterfaceAttribute`

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*.

### Note

You must manually configure IAM permissions for AWS KMS when creating a new Amazon EFS file system that is encrypted at rest. To learn more, see [Encrypting Data at Rest \(p. 136\)](#).

## Creating a Service-Linked Role for Amazon EFS

You don't need to manually create a service-linked role. When you create mount targets 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.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create mount targets for your EFS file system, Amazon EFS creates the service-linked role for you again.

## 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 [Editing 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.

### 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.

### To delete Amazon EFS resources used by the `AWSServiceRoleForAmazonElasticFileSystem`

Complete the following steps to delete Amazon EFS resources used by the `AWSServiceRoleForAmazonElasticFileSystem`. For the detailed procedure, see [Step 4: Clean Up Resources and Protect Your AWS Account](#) (p. 16)

1. On your Amazon EC2 instance, unmount the Amazon EFS file system.
2. Delete the Amazon EFS file system.
3. Delete the custom security group for the file system.

### Warning

If you used the default security group for your VPC, DO NOT delete it.

### To manually delete the service-linked role using IAM

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*.

## Controlling Network Access to Amazon 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 Security Groups for Amazon EC2 Instances and Mount Targets](#) (p. 154)
- [Source Ports for Working with EFS](#) (p. 156)
- [Security Considerations for Network Access](#) (p. 156)
- [Working with Interface VPC Endpoints in Amazon EFS](#) (p. 156)

## Using Security Groups for Amazon EC2 Instances and Mount Targets

When using Amazon EFS, you specify Amazon EC2 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 EC2 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 EFS.

You can authorize inbound and outbound access to your EFS file system. To do so, you add rules that allow your EC2 instance to connect to your Amazon EFS file system through the mount target using the Network File System (NFS) port. Take the following steps to create and update your security groups.

### To create security groups for EC2 instances and mount targets

1. Create two security groups in your VPC.

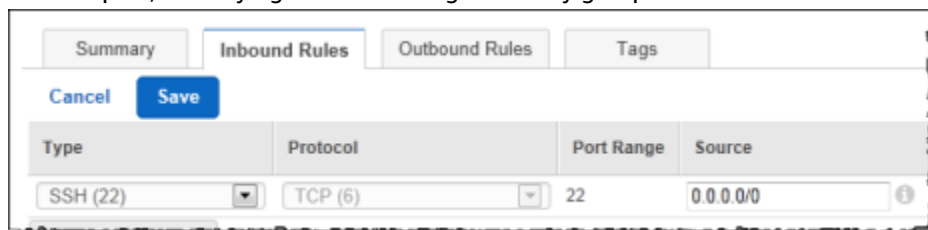
For instructions, see the procedure "To create a security group" in [Creating a Security Group](#) in the *Amazon VPC User Guide*.

2. Open the Amazon VPC Management Console at <https://console.aws.amazon.com/vpc/>, and verify the default rules for these security groups. Both security groups should have only an outbound rule that allows traffic to leave.

### To update the necessary access for your security groups

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Add a rule for your EC2 security group to allow inbound access using Secure Shell (SSH) from any host. Optionally, restrict the **Source** address.

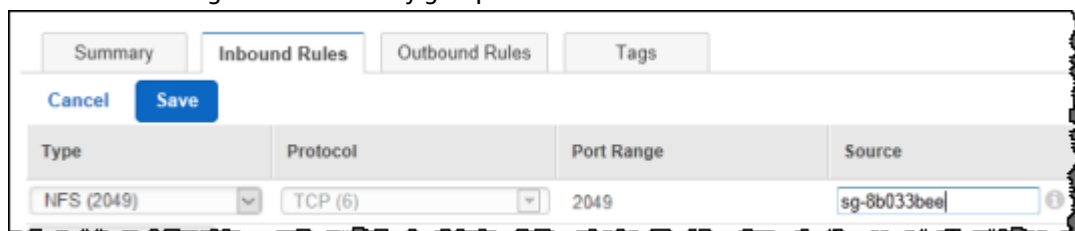
You don't need to add an outbound rule, because the default outbound rule allows all traffic to leave. If this were not the case, you'd need to add an outbound rule to open the TCP connection on the NFS port, identifying the mount target security group as the destination.



Type	Protocol	Port Range	Source
SSH (22)	TCP (6)	22	0.0.0.0/0

For instructions, see [Adding and Removing Rules](#) in the *Amazon VPC User Guide*.

3. Add a rule for the mount target security group to allow inbound access from the EC2 security group as shown following. The EC2 security group is identified as the source.



Type	Protocol	Port Range	Source
NFS (2049)	TCP (6)	2049	sg-8b033bee

4. Verify that both security groups now authorize inbound and outbound access.

For more information about security groups, see [Security Groups for EC2-VPC](#) in the *Amazon EC2 User Guide for Linux Instances*.

## Source Ports for Working with 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.

```
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 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 [Creating Mount Targets \(p. 22\)](#).

## 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. You can use this connection to call the Amazon EFS API from your VPC without sending traffic over the internet. 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 [Interface VPC Endpoints](#) 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 [Accessing Services Through 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](#) in the *Amazon Web Services General Reference*.

For more information about how to create an interface endpoint, see [Creating an Interface 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 [Controlling Access to Services with VPC Endpoints](#) 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": "*"
    }
  ]
}
```

For more information, see [Using VPC Endpoint Policies](#) in the *Amazon VPC User Guide*.

# Working with Users, Groups, and Permissions at the Network File System (NFS) Level

## Topics

- [File and Directory Permissions \(p. 158\)](#)
- [Example Amazon EFS File System Use Cases and Permissions \(p. 159\)](#)
- [User and Group ID Permissions for Files and Directories Within a File System \(p. 159\)](#)
- [No Root Squashing \(p. 160\)](#)
- [Permissions Caching \(p. 161\)](#)
- [Changing File System Object Ownership \(p. 161\)](#)
- [EFS Access Points \(p. 161\)](#)

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 Amazon EFS Access Points \(p. 161\)](#).

Amazon 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.

Following, you can find examples of permissions and a discussion about NFS permissions considerations for Amazon EFS.

## 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 [Working with Users, Groups, and Permissions at the Network File System \(NFS\) Level \(p. 158\)](#).

### 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 NFS Access to Amazon EFS \(p. 150\)](#) and [Creating Access Points \(p. 31\)](#).

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`.

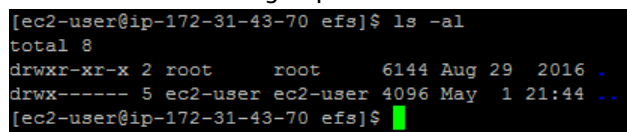
## Example Amazon EFS File System Use Cases and Permissions

After you create an Amazon 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 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,rsz=1048576,wsz=1048576,hard,timeo=600,retrans=2,noresvport file-system-  
id.efs.aws-region.amazonaws.com:/ efs-mount-point
```

The root user and root group own the mounted directory.



```
[ec2-user@ip-172-31-43-70 efs]$ ls -al  
total 8  
drwxr-xr-x 2 root    root      6144 Aug 29  2016 .  
drwx----- 5 ec2-user ec2-user 4096 May  1 21:44 ..  
[ec2-user@ip-172-31-43-70 efs]$
```

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 [Walkthrough: Create Writable Per-User Subdirectories and Configure Automatic Remounting on Reboot \(p. 118\)](#).
- Allow users to write to the Amazon EFS file system root. A user with root privileges can grant other users access to the file system.
  - To change the Amazon 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 Amazon 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 Amazon 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 Amazon 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, as shown following.

```
$ 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.



## Permissions Caching

Amazon EFS caches file permissions for a small time period. As a result, there might be a brief window where a user who had access to a file system object but the 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 Amazon EFS Access Points \(p. 161\)](#).

## Working with Amazon EFS Access Points

Amazon 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.

For more information on creating an access point, see [Creating Access Points \(p. 31\)](#).

### Topics

- [Creating an Access Point \(p. 161\)](#)
- [Mounting a File System Using an Access Point \(p. 162\)](#)
- [Enforcing a User Identity Using an Access Point \(p. 162\)](#)
- [Enforcing a Root Directory with an Access Point \(p. 162\)](#)
- [Using Access Points in IAM Policies \(p. 163\)](#)

## Creating an Access Point

You can create access points for an existing Amazon EFS file system using the AWS Management Console, the AWS Command Line Interface (AWS CLI), and the EFS API.

For directions about how to create an access point, see [Creating Access Points \(p. 31\)](#).

## Mounting a File System Using an Access Point

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,accesspoint=fsap-12345678 fs-12345678: /localmountpoint
```

For more information on mounting file systems using an access point, see [Mounting with EFS Access Points](#) (p. 63).

## 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.

## 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 configurable ownership and permissions. 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. You can configure the root directory ownership and permission by using the following attributes when creating an access point:

- **OwnerId** – 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. 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 [Working with Users, Groups, and Permissions at the Network File System \(NFS\) Level \(p. 158\)](#).

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. If the root directory configured on the access point 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.

## 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, 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`.

```
{
  "Version": "2012-10-17",
  "Id": "MyFileSystemPolicy",
  "Statement": [
    {
      "Sid": "App1Access",
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/app1" },
      "Action": [
        "elasticfilesystem:ClientMount",
        "elasticfilesystem:ClientWrite"
      ]
    }
  ]
}
```

```
    ],
    "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"
    ],
    "Condition": {
      "StringEquals": {
        "elasticfilesystem:AccessPointArn" : "arn:aws:elasticfilesystem:us-
east-1:222233334444:access-point/fsap-89abcdef"
      }
    }
  }
]
}
```

## Logging and Monitoring in Amazon EFS

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon EFS and your AWS solutions. You should 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. AWS provides several tools for monitoring your Amazon EFS resources and responding to potential incidents:

### Amazon CloudWatch Alarms

Using Amazon CloudWatch events, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch events do not invoke actions simply 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 EFS with Amazon CloudWatch \(p. 74\)](#).

### Amazon CloudWatch Logs

You can use Amazon CloudWatch Logs to monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see [Monitoring Log Files](#) 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

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon EFS. 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 [Logging Amazon EFS API Calls with AWS CloudTrail \(p. 82\)](#).

## Compliance Validation for Amazon Elastic File System

Third-party auditors assess the security and compliance of Amazon Elastic File System as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, DoD CC SRG, HITRUST CSF, OSPAR, CS, and ENS High.

For a list of AWS services in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using EFS is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [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 that helps you check your compliance with security industry standards and best practices.

## Resilience in Amazon Elastic File System

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

## Amazon Elastic File System Network Isolation

As a managed service, Amazon Elastic File System is protected by the AWS global network security procedures described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

Access to Amazon EFS by using the network is through AWS published APIs. Clients must support Transport Layer Security (TLS) 1.0. We recommend TLS 1.2. Clients must also support cipher suites with Perfect Forward Secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes. Additionally, requests must be signed by using either an access key ID and a secret access key that is associated with an IAM principal or the [AWS Security Token Service \(STS\)](#) 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 and Limits

Following, you can find out about quotas when working with Amazon EFS.

## Topics

- [Amazon EFS Quotas That You Can Increase \(p. 167\)](#)
- [Resource Quotas \(p. 168\)](#)
- [Quotas for NFS Clients \(p. 169\)](#)
- [Quotas for Amazon EFS File Systems \(p. 169\)](#)
- [Unsupported NFSv4 Features \(p. 170\)](#)
- [Additional Considerations \(p. 170\)](#)

## Amazon EFS Quotas That You Can Increase

Service Quotas is an AWS service that helps you manage your quotas, or limits, from one location. You can view all Amazon EFS limit values in the [Service Quotas console](#). You can also request a quota increase for the number of EFS file systems in an AWS Region using the Service Quotas console.

You can also request an increase to the following Amazon EFS quotas by contacting AWS Support. To learn more, see [Requesting a Quota Increase \(p. 168\)](#). The Amazon EFS service team reviews each request individually.

Resource	Default Quota
Number of file systems for each customer account in an AWS Region	1,000

The default throughput quotas apply to both throughput modes, Bursting and Provisioned. For more information about these different modes, see [Amazon EFS Performance \(p. 90\)](#).

Resource	Default Quota
Total bursting throughput for all connected clients	US East (Ohio) Region – 3 GB/s US East (N. Virginia) Region – 3 GB/s US West (N. California) Region – 1 GB/s US West (Oregon) Region – 3 GB/s Africa (Cape Town) Region – 1 GB/s Asia Pacific (Hong Kong) Region – 1 GB/s Asia Pacific (Mumbai) Region – 1 GB/s Asia Pacific (Seoul) Region – 1 GB/s Asia Pacific (Singapore) Region – 1 GB/s

Resource	Default Quota
	Asia Pacific (Sydney) Region – 3 GB/s
	Asia Pacific (Tokyo) Region – 1 GB/s
	Canada (Central) Region – 1 GB/s
	China (Beijing) Region – 1 GB/s
	China (Ningxia) Region – 1 GB/s
	Europe (Frankfurt) Region – 1 GB/s
	Europe (Ireland) Region – 3 GB/s
	Europe (London) Region – 1 GB/s
	Europe (Milan) Region – 1 GB/s
	Europe (Paris) Region – 1 GB/s
	Europe (Stockholm) Region – 1 GB/s
	Middle East (Bahrain) Region – 1 GB/s
	South America (São Paulo) Region – 1 GB/s
Total provisioned throughput for all connected clients	All AWS Regions – 1 GB/s

## 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.

## Resource Quotas

Following are the quotas on Amazon EFS resources for each customer account in an AWS Region.

Resource	Quota
Number of access points for each file system	120
Number of mount targets for each file system in an Availability Zone	1
Number of mount targets for each VPC	400



Resource	Quota
Number of security groups for each mount target	5
Number of tags for each file system	50
Number of VPCs for each file system	1

**Note**

Clients can also connect to mount targets in another account or VPC. For more information, see [Mounting EFS File Systems from Another Account or VPC \(p. 68\)](#).

## Quotas for NFS Clients

The following quotas for NFS clients apply, assuming a Linux NFSv4.1 client:

- The maximum throughput you can drive for each NFS client is 250 MB/s. NFS client throughput is calculated as the total number of bytes that are sent and received, with a minimum NFS request size of 4KB.
- Up to 128 active user accounts for each client can have files open at the same time. Each user account represents one local user logged in to the instance. A user account that is logged in multiple times counts as one active user.
- Up to 32,768 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 8,192 locks across a maximum of 256 unique file-process pairs. For example, a single process can acquire one or more locks on 256 separate files. As another example, eight processes can each acquire one or more locks on 32 files.
- 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. For more information, see [On-Premises Performance Considerations \(p. 96\)](#).
- 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:

- Maximum name length: 255 bytes.
- Maximum symbolic link (symlink) length: 4,080 bytes.
- Maximum number of hard links to a file: 177.
- Maximum size of a single file: 52,673,613,135,872 bytes (47.9 TiB).
- Maximum directory depth: 1,000 levels deep.
- Any one particular file can have up to 512 locks across all instances connected and users accessing the file.
- In General Purpose mode, there is a limit of 35,000 file operations per second. Operations that read data or metadata consume one file operation, operations that write data or update metadata consume five file operations. This means that a file system can support 35,000 read operations per second, or

7,000 write operations, or some combination of the two. For example, 20,000 read operations and 3,000 write operations (20,000 reads x 1 file operation per read + 3,000 writes x 5 file operations per write = 35,000 file operations). File operations are counted from all connecting clients.

## Unsupported NFSv4 Features

Although Amazon Elastic File System doesn't support NFSv2, or NFSv3, Amazon EFS supports both NFSv4.1 and NFSv4.0, except for the following features:

- pNFS
- 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](#).
- You mount your file system from EC2 instances in your VPC by using the mount targets you create in the VPC. You can also mount your file system on your EC2-Classic instances, which are not in the VPC. However, you must first link them to your VPC by using ClassicLink. For more information about using ClassicLink, see [ClassicLink](#) in the *Amazon EC2 User Guide for Linux Instances*.
- You can mount an Amazon EFS file system from on-premises data center servers using AWS Direct Connect and VPN.

# Troubleshooting Amazon EFS

You can find information about how to troubleshoot the following issues for Amazon Elastic File System (Amazon EFS).

## Topics

- [Troubleshooting Amazon EFS: General Issues \(p. 172\)](#)
- [Troubleshooting File Operation Errors \(p. 174\)](#)
- [Troubleshooting AMI and Kernel Issues \(p. 176\)](#)
- [Troubleshooting Mount Issues \(p. 177\)](#)
- [Troubleshooting Encryption \(p. 182\)](#)

## Troubleshooting Amazon EFS: General Issues

Use this information to troubleshoot general Amazon EFS issues. For information about performance, see [Amazon EFS Performance \(p. 90\)](#).

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
- 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 \(p. 173\)](#).

## Topics

- [Amazon EC2 Instance Hangs \(p. 172\)](#)
- [Application Writing Large Amounts of Data Hangs \(p. 173\)](#)
- [Poor Performance When Opening Many Files in Parallel \(p. 173\)](#)
- [Custom NFS Settings Causing Write Delays \(p. 173\)](#)
- [Creating Backups with Oracle Recovery Manager Is Slow \(p. 174\)](#)

## 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 \(p. 71\)](#).

## 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 \(p. 172\)](#) 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 File Operation Errors

When you access Amazon EFS file systems, certain limits on the files in the file system apply. Exceeding these limits causes file operation errors. For more information on client and file-based limits in Amazon EFS, see [Quotas for NFS Clients](#) (p. 169). Following, you can find some common file operation errors and the limits associated with each error.

**Topics**

- [Command Fails with "Disk quota exceeded" Error](#) (p. 174)
- [Command Fails with "I/O error" Error](#) (p. 175)
- [Command Fails with "File name is too long" Error](#) (p. 175)
- [Command Fails with "File not found" Error](#) (p. 175)
- [Command Fails with "Too many links" Error](#) (p. 176)
- [Command Fails with "File too large" Error](#) (p. 176)

### 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 128 active user accounts can have files open at once for an instance.

- Up to 32,768 files can be open at once for an instance.
- Each unique mount on the instance can acquire up to a total of 8,192 locks across 256 unique file-process pairs. For example, a single process can acquire one or more locks on 256 separate files, or eight processes can each acquire one or more locks on 32 files.

#### 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.

## Command Fails with "I/O error"

This error occurs when you encounter one of the following issues:

- More than 128 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.

## 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 \(p. 176\)](#)
- [File System Keeps Performing Operations Repeatedly Due to Client Bug \(p. 176\)](#)
- [Deadlocked Client \(p. 177\)](#)
- [Listing Files in a Large Directory Takes a Long Time \(p. 177\)](#)

## 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 workflow, 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.

## Troubleshooting Mount Issues

Following, you can find information about troubleshooting file-system mounting issues for Amazon EFS.

- [File System Mount on Windows Instance Fails \(p. 178\)](#)
- [Access Denied by Server \(p. 178\)](#)
- [Automatic Mounting Fails and the Instance Is Unresponsive \(p. 178\)](#)
- [Mounting Multiple Amazon EFS File Systems in /etc/fstab Fails \(p. 178\)](#)
- [Mount Command Fails with "wrong fs type" Error Message \(p. 179\)](#)
- [Mount Command Fails with "incorrect mount option" Error Message \(p. 179\)](#)
- [File System Mount Fails Immediately After File System Creation \(p. 180\)](#)
- [File System Mount Hangs and Then Fails with Timeout Error \(p. 180\)](#)
- [File System Mount Using DNS Name Fails \(p. 180\)](#)
- [File System Mount Fails with "nfs not responding" \(p. 181\)](#)
- [Mount Target Lifecycle State Is Stuck \(p. 181\)](#)

- [Mount Does Not Respond \(p. 181\)](#)
- [Operations on Newly Mounted File System Return "bad file handle" Error \(p. 182\)](#)
- [Unmounting a File System Fails \(p. 182\)](#)

## 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` 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.

## 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` with the following contents.

```
[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
```

After you do so, 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 \(p. 291\)](#).

## 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 \(p. 291\)](#).

## 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,rsz=1048576,wsz=1048576,hard,timeo=600,retrans=2,noresvport mount-target-  
ip:/ mnt  
  
[2+ minute wait here]  
mount.nfs: Connection timed out  
$
```

### 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.

Type	Protocol	Port Range	Source	Description
NFS	TCP	2049	Custom	e.g. SSH for Admin Desktop

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel Save

For more information, see [Creating Security Groups \(p. 27\)](#).

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 Using DNS Name Fails

A file system mount that is using a DNS name fails. The following code shows an example.

```
$ sudo mount -t nfs -o  
nfsvers=4.1,rsz=1048576,wsz=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 [Using DNS with Your VPC](#) in the *Amazon VPC User Guide*.

To specify a 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 Account or VPC \(p. 68\)](#).
- Connect your Amazon EC2 instance inside an Amazon VPC configured to use the DNS server provided by Amazon. For more information, see [DHCP Options Sets](#) 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 [Updating DNS Support for 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 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.

## 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 \(p. 71\)](#).

## 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:

- Wait for all read and write operations to finish, and then attempt the `umount` command again.
- Force the `umount` command to finish with the `-f` option.

#### Warning

Forcing an unmount interrupts any data read or write operations that are currently in process for the file system.

## Troubleshooting Encryption

Following, you can find information about troubleshooting encryption issues for Amazon EFS.

- [Mounting with Encryption of Data in Transit Fails \(p. 182\)](#)
- [Mounting with Encryption of Data in Transit is Interrupted \(p. 183\)](#)
- [Encrypted-at-Rest File System Can't Be Created \(p. 183\)](#)
- [Unusable Encrypted File System \(p. 183\)](#)

## 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 \(p. 37\)](#).

## 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 \(p. 40\)](#).
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.



# 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 \(p. 139\)](#).

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 Endpoint

The API endpoint is the DNS name used as a host in the HTTP URI for the API calls. These API endpoints are specific to AWS Regions and take the following form.

```
elasticfilesystem.aws-region.amazonaws.com
```

For example, the Amazon EFS API endpoint for the US West (Oregon) Region is the following.

```
elasticfilesystem.us-west-2.amazonaws.com
```

For a list of AWS Regions that Amazon EFS supports (where you can create and manage file systems), see [Amazon Elastic File System](#) in the *AWS General Reference*.

The region-specific API endpoint defines the scope of the Amazon EFS resources that are accessible when you make an API call. For example, when you call the `DescribeFileSystems` operation using the preceding endpoint, you get a list of file systems in the US West (Oregon) Region that have been created in your account.

## 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 \(p. 192\)](#).

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 \(p. 139\)](#)
- [Actions \(p. 186\)](#)
- [Data Types \(p. 264\)](#)

## 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 by AWS Identity and Access Management (IAM) users 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 [Error Retries and Exponential Backoff in AWS](#) in the *Amazon Web Services General Reference*.

## Actions

The following actions are supported:

- [CreateAccessPoint](#) (p. 188)
- [CreateFileSystem](#) (p. 192)
- [CreateMountTarget](#) (p. 200)
- [CreateTags](#) (p. 207)
- [DeleteAccessPoint](#) (p. 210)
- [DeleteFileSystem](#) (p. 212)
- [DeleteFileSystemPolicy](#) (p. 214)
- [DeleteMountTarget](#) (p. 216)
- [DeleteTags](#) (p. 219)
- [DescribeAccessPoints](#) (p. 221)
- [DescribeFileSystemPolicy](#) (p. 224)
- [DescribeFileSystems](#) (p. 227)
- [DescribeLifecycleConfiguration](#) (p. 231)
- [DescribeMountTargets](#) (p. 234)
- [DescribeMountTargetSecurityGroups](#) (p. 238)
- [DescribeTags](#) (p. 241)
- [ListTagsForResource](#) (p. 244)
- [ModifyMountTargetSecurityGroups](#) (p. 246)
- [PutFileSystemPolicy](#) (p. 249)
- [PutLifecycleConfiguration](#) (p. 252)
- [TagResource](#) (p. 256)
- [UntagResource](#) (p. 258)
- [UpdateFileSystem](#) (p. 260)

## 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 its own directory and below. To learn more, see [Mounting a File System Using EFS Access Points](#).

This operation requires permissions for the `elasticfilesystem:CreateAccessPoint` action.

### 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 (p. 188)

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.

Required: Yes

#### FileSystemId (p. 188)

The ID of the EFS file system that the access point provides access to.

Type: String

Required: Yes

#### PosixUser (p. 188)

The operating system user and group applied to all file system requests made using the access point.

Type: [PosixUser \(p. 275\)](#) object

Required: No

#### RootDirectory (p. 188)

Specifies the directory on the Amazon 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, EFS creates it and applies the `CreationInfo` settings when a client connects to an access point. When specifying a `RootDirectory`, you need to provide the `Path`, and the `CreationInfo` is optional.

Type: [RootDirectory \(p. 276\)](#) object

Required: No

#### Tags (p. 188)

Creates tags associated with the access point. Each tag is a key-value pair.

Type: Array of [Tag \(p. 277\)](#) 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** (p. 189)

The unique Amazon Resource Name (ARN) associated with the access point.

Type: String

### **AccessPointId** (p. 189)

The ID of the access point, assigned by Amazon EFS.

Type: String

### **ClientToken** (p. 189)

The opaque string specified in the request to ensure idempotent creation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

### **FileSystemId** (p. 189)

The ID of the EFS file system that the access point applies to.

Type: String

### **LifecycleState** (p. 189)

Identifies the lifecycle phase of the access point.

Type: String

Valid Values: `creating` | `available` | `updating` | `deleting` | `deleted`

### **Name** (p. 189)

The name of the access point. This is the value of the `Name` tag.

Type: String

### **OwnerId** (p. 189)

Identified the AWS account that owns the access point resource.

Type: String

### **PosixUser** (p. 189)

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](#) (p. 275) object

### **RootDirectory** (p. 189)

The directory on the Amazon EFS file system that the access point exposes as the root directory to NFS clients using the access point.

Type: [RootDirectory](#) (p. 276) object

**Tags (p. 189)**

The tags associated with the access point, presented as an array of Tag objects.

Type: Array of [Tag](#) (p. 277) objects

## Errors

### **AccessPointAlreadyExists**

Returned if the access point 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.

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

### **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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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.

### 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 \(p. 227\)](#) operation, which among other things returns the file system state.

This operation also takes an optional `PerformanceMode` parameter that you choose for your file system. We recommend `generalPurpose` performance mode for most 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. For more information, see [Amazon EFS: Performance Modes](#).

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 \(p. 200\)](#). 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.

## Request Syntax

```
POST /2015-02-01/file-systems HTTP/1.1
Content-type: application/json

{
  "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.

### CreationToken (p. 192)

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.

Required: Yes

### Encrypted (p. 192)

A Boolean value that, if true, creates an encrypted file system. When creating an encrypted file system, you have the option of specifying [CreateFileSystem:KmsKeyId \(p. 193\)](#) for an existing AWS Key Management Service (AWS KMS) customer master key (CMK). If you don't specify a CMK, then the default CMK for Amazon EFS, `/aws/elasticfilesystem`, is used to protect the encrypted file system.

Type: Boolean

Required: No

### KmsKeyId (p. 192)

The ID of the AWS KMS CMK to be used to protect the encrypted file system. This parameter is only required if you want to use a nondefault CMK. If this parameter is not specified, the default CMK for Amazon EFS is used. This ID can be in one of 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 `KmsKeyId` is specified, the [CreateFileSystem:Encrypted \(p. 193\)](#) parameter must be set to true.

#### Important

EFS accepts only symmetric CMKs. You cannot use asymmetric CMKs with EFS file systems.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

### PerformanceMode (p. 192)

The performance mode of the file system. We recommend `generalPurpose` performance mode for most 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.

Type: String

Valid Values: `generalPurpose` | `maxIO`

Required: No

#### ProvisionedThroughputInMibps (p. 192)

The throughput, measured in MiB/s, that you want to provision for a file system that you're creating. Valid values are 1-1024. Required if `ThroughputMode` is set to `provisioned`. The upper limit for throughput is 1024 MiB/s. You can get this limit increased by contacting AWS Support. For more information, see [Amazon EFS Limits That You Can Increase](#) in the *Amazon EFS User Guide*.

Type: Double

Valid Range: Minimum value of 1.0.

Required: No

#### Tags (p. 192)

A value that specifies 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.

Type: Array of [Tag \(p. 277\)](#) objects

Required: No

#### ThroughputMode (p. 192)

The throughput mode for the file system to be created. There are two throughput modes to choose from for your file system: `bursting` and `provisioned`. If you set `ThroughputMode` to `provisioned`, you must also set a value for `ProvisionedThroughPutInMibps`. You can decrease your file system's throughput in `Provisioned Throughput` mode or change between the throughput modes as long as it's been more than 24 hours since the last decrease or throughput mode change. For more, see [Specifying Throughput with Provisioned Mode](#) in the *Amazon EFS User Guide*.

Type: String

Valid Values: `bursting` | `provisioned`

Required: No

## Response Syntax

```
HTTP/1.1 201
Content-type: application/json

{
  "CreationTime": number,
  "CreationToken": "string",
  "Encrypted": boolean,
  "FileSystemId": "string",
  "KmsKeyId": "string",
  "LifeCycleState": "string",
  "Name": "string",
  "NumberOfMountTargets": number,
```

```
{
  "OwnerId": "string",
  "PerformanceMode": "string",
  "ProvisionedThroughputInMibps": number,
  "SizeInBytes": {
    "Timestamp": number,
    "Value": 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.

### CreationTime (p. 194)

The time that the file system was created, in seconds (since 1970-01-01T00:00:00Z).

Type: Timestamp

### CreationToken (p. 194)

The opaque string specified in the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

### Encrypted (p. 194)

A Boolean value that, if true, indicates that the file system is encrypted.

Type: Boolean

### FileSystemId (p. 194)

The ID of the file system, assigned by Amazon EFS.

Type: String

### KmsKeyId (p. 194)

The ID of an AWS Key Management Service (AWS KMS) customer master key (CMK) that was used to protect the encrypted file system.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

### LifeCycleState (p. 194)

The lifecycle phase of the file system.

Type: String

Valid Values: `creating` | `available` | `updating` | `deleting` | `deleted`

#### **Name (p. 194)**

You can add tags to a file system, including a `Name` tag. For more information, see [CreateFileSystem \(p. 192\)](#). If the file system has a `Name` tag, Amazon EFS returns the value in this field.

Type: String

Length Constraints: Maximum length of 256.

#### **NumberOfMountTargets (p. 194)**

The current number of mount targets that the file system has. For more information, see [CreateMountTarget \(p. 200\)](#).

Type: Integer

Valid Range: Minimum value of 0.

#### **OwnerId (p. 194)**

The AWS account that created the file system. If the file system was created by an IAM user, the parent account to which the user belongs is the owner.

Type: String

#### **PerformanceMode (p. 194)**

The performance mode of the file system.

Type: String

Valid Values: `generalPurpose` | `maxIO`

#### **ProvisionedThroughputInMibps (p. 194)**

The throughput, measured in MiB/s, that you want to provision for a file system. Valid values are 1-1024. Required if `ThroughputMode` is set to `provisioned`. The limit on throughput is 1024 MiB/s. You can get these limits increased by contacting AWS Support. For more information, see [Amazon EFS Limits That You Can Increase](#) in the *Amazon EFS User Guide*.

Type: Double

Valid Range: Minimum value of 1.0.

#### **SizeInBytes (p. 194)**

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 \(p. 271\)](#) object

#### **Tags (p. 194)**

The tags associated with the file system, presented as an array of `Tag` objects.

Type: Array of [Tag \(p. 277\)](#) objects

#### **ThroughputMode (p. 194)**

The throughput mode for a file system. There are two throughput modes to choose from for your file system: `bursting` and `provisioned`. If you set `ThroughputMode` to `provisioned`, you

must also set a value for `ProvisionedThroughputInMibps`. You can decrease your file system's throughput in Provisioned Throughput mode or change between the throughput modes as long as it's been more than 24 hours since the last decrease or throughput mode change.

Type: String

Valid Values: `bursting` | `provisioned`

## 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 to provisioned throughput mode.

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

## Example

### Create a File System

The following example sends a POST request to create a file system in the `us-west-2` region. The request specifies `myFileSystem1` as the creation token.

## 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",
  "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",
  "PerformanceMode" : "generalPurpose",
  "fileSystemId": "fs-01234567",
  "CreationTime": "1403301078",
  "LifecycleState": "creating",
  "numberOfMountTargets": 0,
  "SizeInBytes": {
    "Timestamp": 1403301078,
    "Value": 29313417216,
    "ValueInIA": 675432,
    "ValueInStandard": 29312741784
  },
  "Tags": [
    {
      "Key": "Name",
      "Value": "Test Group1"
    }
  ]
}
```

## 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)

- [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.

You can create one mount target in each Availability Zone in your VPC. All EC2 instances in a VPC within a given Availability Zone share a single mount target for a given file system. If you have multiple subnets in an Availability Zone, you create a mount target in one of the subnets. EC2 instances do not need to be in the same subnet as the mount target in order to access their file system. For more information, see [Amazon EFS: How it Works](#).

In the request, you also specify a file system ID for which you are creating the mount target and the file system's lifecycle state must be `available`. For more information, see [DescribeFileSystems \(p. 227\)](#).

In the request, you also provide a subnet ID, which determines the following:

- VPC in which Amazon EFS creates the mount target
- Availability Zone in which Amazon EFS creates the mount target
- IP address range from which Amazon EFS selects the IP address of the mount target (if you don't specify an IP address in the request)

After creating the mount target, Amazon EFS returns a response that includes, a `MountTargetId` and an `IpAddress`. 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 it Works: Implementation Overview](#).

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. That is, if the file system already has one or more mount targets created for it, the subnet specified in the request to add another mount target must meet the following requirements:

- Must belong to the same VPC as the subnets of the existing mount targets
- Must not be in the same Availability Zone as any of the subnets of the existing mount targets

If the request satisfies the requirements, Amazon EFS does the following:

- Creates a new mount target in the specified subnet.
- Also creates a new network interface in the subnet as follows:
  - If the request provides an `IpAddress`, Amazon EFS assigns that IP address to the network interface. Otherwise, Amazon EFS assigns a free address in the subnet (in the same way that the Amazon EC2 `CreateNetworkInterface` call does when a request does not specify a primary private IP address).
  - If the request provides `SecurityGroups`, this network interface is associated with those security groups. Otherwise, it belongs to the default security group for the subnet's VPC.
  - Assigns 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`.
  - Sets the `requesterManaged` property of the network interface to `true`, and the `requesterId` value to `EFS`.

Each Amazon EFS 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 `IpAddress` field 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 \(p. 234\)](#) operation, which among other things returns the mount target state.

We recommend that you create a mount target in each of the Availability Zones. There are cost considerations for using a file system in an Availability Zone through a mount target created in another Availability Zone. For more information, see [Amazon EFS](#). In addition, by always using a mount target local to the instance's Availability Zone, you eliminate a partial failure scenario. If the Availability Zone in which your mount target is created goes down, then you can't access your file system through that mount target.

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",
  "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 \(p. 201\)](#)

The ID of the file system for which to create the mount target.

Type: String

Required: Yes

### [IpAddress \(p. 201\)](#)

Valid IPv4 address within the address range of the specified subnet.

Type: String

Required: No

### SecurityGroups (p. 201)

Up to five VPC security group IDs, of the form `sg-xxxxxxx`. These must be for the same VPC as subnet specified.

Type: Array of strings

Array Members: Maximum number of 5 items.

Required: No

### SubnetId (p. 201)

The ID of the subnet to add the mount target in.

Type: String

Required: Yes

## Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "AvailabilityZoneId": "string",
  "AvailabilityZoneName": "string",
  "FileSystemId": "string",
  "IpAddress": "string",
  "LifecycleState": "string",
  "MountTargetId": "string",
  "NetworkInterfaceId": "string",
  "OwnerId": "string",
  "SubnetId": "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 (p. 202)

The unique and consistent identifier of the Availability Zone (AZ) that the mount target resides in. For example, `us-east-1a` is an AZ ID for the `us-east-1` Region and it has the same location in every AWS account.

Type: String

### AvailabilityZoneName (p. 202)

The name of the Availability Zone (AZ) that the mount target resides in. AZs 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

### FileSystemId (p. 202)

The ID of the file system for which the mount target is intended.

Type: String

**IpAddress (p. 202)**

Address at which the file system can be mounted by using the mount target.

Type: String

**LifeCycleState (p. 202)**

Lifecycle state of the mount target.

Type: String

Valid Values: `creating` | `available` | `updating` | `deleting` | `deleted`

**MountTargetId (p. 202)**

System-assigned mount target ID.

Type: String

**NetworkInterfaceId (p. 202)**

The ID of the network interface that Amazon EFS created when it created the mount target.

Type: String

**OwnerId (p. 202)**

AWS account ID that owns the resource.

Type: String

**SubnetId (p. 202)**

The ID of the mount target's subnet.

Type: String

## 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

**IpAddressInUse**

Returned if the request specified an `IpAddress` 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. The client should try to delete some elastic network interfaces or get the account limit raised. For more information, see [Amazon VPC Limits](#) in the *Amazon VPC User Guide* (see the Network interfaces per VPC entry in the table).

HTTP Status Code: 409

**NoFreeAddressesInSubnet**

Returned if `IpAddress` was not specified in the request and there are no free IP addresses in the subnet.

HTTP Status Code: 409

**SecurityGroupLimitExceeded**

Returned if the size of `SecurityGroups` specified in the request is greater than five.

HTTP Status Code: 400

**SecurityGroupNotFound**

Returned if one of the specified security groups doesn't exist in the subnet's VPC.

HTTP Status Code: 400

**SubnetNotFound**

Returned if there is no subnet with ID `SubnetId` provided in the request.

HTTP Status Code: 400

**UnsupportedAvailabilityZone**

HTTP Status Code: 400

## Examples

### Add a Mount Target to a File System

The following request creates a mount target for a file system. The request specifies values for only the required `FileSystemId` and `SubnetId` parameters. The request does not provide the optional `IpAddress` and `SecurityGroups` parameters. For `IpAddress`, the operation uses one of the available IP addresses in the specified subnet. And, the operation uses the default security group associated with the VPC for the `SecurityGroups`.

#### 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

{
  "MountTargetId": "fsmt-55a4413c",
  "NetworkInterfaceId": "eni-01234567",
  "FileSystemId": "fs-01234567",
  "LifeCycleState": "available",
  "SubnetId": "subnet-01234567",
  "OwnerId": "231243201240",
  "IpAddress": "172.31.22.183"
}
```

## Add a Mount Target to a File System

The following request specifies all the request parameters to create a 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"  
  }
```

## 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## CreateTags

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](#) (p. 227) 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 requires the following URI parameters.

### [FileSystemId](#) (p. 207)

The ID of the file system whose tags you want to modify (String). This operation modifies the tags only, not the file system.

## Request Body

The request accepts the following data in JSON format.

### [Tags](#) (p. 207)

An array of `Tag` objects to add. Each `Tag` object is a key-value pair.

Type: Array of [Tag](#) (p. 277) 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

## Example

### 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/create-tags/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": [
    {
      "Value": "value1",
      "Key": "key1"
    },
    {
      "Value": "value2",
      "Key": "key2"
    },
    {
      "Value": "value3",
      "Key": "key3"
    }
  ]
}
```

#### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 requires the following URI parameters.

#### **AccessPointId** (p. 210)

The ID of the access point that you want to delete.

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 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 \(p. 234\)](#) and [DeleteMountTarget \(p. 216\)](#).

### 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 \(p. 227\)](#) 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 \(p. 227\)](#) 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 requires the following URI parameters.

### **FileSystemId** (p. 212)

The ID of the file system you want to delete.

## 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

## Example

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 requires the following URI parameters.

#### `FileSystemId` (p. 214)

Specifies the EFS file system for which to delete the `FileSystemPolicy`.

### 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

#### **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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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](#) (p. 234) 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 requires the following URI parameters.

### **MountTargetId** (p. 216)

The ID of the mount target to delete (String).

## 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 account.

HTTP Status Code: 404

## Example

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DeleteTags

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 requires the following URI parameters.

### [FileSystemId \(p. 219\)](#)

The ID of the file system whose tags you want to delete (String).

## Request Body

The request accepts the following data in JSON format.

### [TagKeys \(p. 219\)](#)

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.

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

## Example

### Delete Tags from a File System

The following request deletes the tag `key2` from the tag set associated with the file system.

#### Sample Request

```
POST /2015-02-01/delete-tags/fs-01234567 HTTP/1.1
Host: elasticfilesystem.us-west-2.amazonaws.com
x-amz-date: 20140620T215123Z
Authorization: <...>
Content-Type: application/json
Content-Length: 223

{
  "TagKeys": [
    "key2"
  ]
}
```

#### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 requires the following URI parameters.

### AccessPointId (p. 221)

(Optional) Specifies an EFS access point to describe in the response; mutually exclusive with `FileSystemId`.

### FileSystemId (p. 221)

(Optional) If you provide a `FileSystemId`, EFS returns all access points for that file system; mutually exclusive with `AccessPointId`.

### MaxResults (p. 221)

(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 (p. 221)

`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.

## 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 (p. 221)

An array of access point descriptions.

Type: Array of [AccessPointDescription](#) (p. 265) objects

### NextToken (p. 221)

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

## 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 requires the following URI parameters.

#### [FileSystemId \(p. 224\)](#)

Specifies which EFS file system to retrieve the `FileSystemPolicy` for.

### 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 \(p. 224\)](#)

Specifies the EFS file system to which the `FileSystemPolicy` applies.

Type: String

#### [Policy \(p. 224\)](#)

The JSON formatted `FileSystemPolicy` for the EFS file system.

Type: String

### Errors

#### **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 the default file system policy is in effect for the EFS file system specified.

HTTP Status Code: 404

## Example

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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. Currently, this number is automatically set to 10. 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 requires the following URI parameters.

### **CreationToken** (p. 227)

(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.

### **FileSystemId** (p. 227)

(Optional) ID of the file system whose description you want to retrieve (String).

### **Marker** (p. 227)

(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.

### **MaxItems** (p. 227)

(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": [
    {
      "CreationTime": number,
      "CreationToken": "string",
      "Encrypted": boolean,
      "FileSystemId": "string",
      "KmsKeyId": "string",
      "LifeCycleState": "string",
      "Name": "string",
      "NumberOfMountTargets": number,
      "OwnerId": "string",
      "PerformanceMode": "string",
      "ProvisionedThroughputInMibps": number,
      "SizeInBytes": {
        "Timestamp": number,
        "Value": 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 (p. 228)

An array of file system descriptions.

Type: Array of [FileSystemDescription \(p. 268\)](#) objects

### Marker (p. 228)

Present if provided by caller in the request (String).

Type: String

### NextMarker (p. 228)

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

## 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

## Example

### 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": 29313417216,
        "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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## DescribeLifecycleConfiguration

Returns the current `LifecycleConfiguration` object for the specified Amazon EFS file system. EFS lifecycle management uses the `LifecycleConfiguration` object to identify which files to move to the EFS Infrequent Access (IA) storage class. 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 requires the following URI parameters.

#### `FileSystemId` (p. 231)

The ID of the file system whose `LifecycleConfiguration` object you want to retrieve (String).

### Request Body

The request does not have a request body.

### Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "LifecyclePolicies": [
    {
      "TransitionToIA": "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` (p. 231)

An array of lifecycle management policies. Currently, EFS supports a maximum of one policy per file system.

Type: Array of `LifecyclePolicy` (p. 272) 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

## Example

### Retrieve a 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": [
    {
      "TransitionToIA": "AFTER_14_DAYS"
    }
  ]
}
```

## 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](#)



- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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`.

## Request Syntax

```
GET /2015-02-01/mount-targets?
AccessPointId=AccessPointId&FileSystemId=FileSystemId&Marker=Marker&MaxItems=MaxItems&MountTargetId=MountTargetId
HTTP/1.1
```

## URI Request Parameters

The request requires the following URI parameters.

### **AccessPointId** (p. 234)

(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.

### **FileSystemId** (p. 234)

(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.

### **Marker** (p. 234)

(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.

### **MaxItems** (p. 234)

(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** (p. 234)

(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.

## 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",
      "LifecycleState": "string",
      "MountTargetId": "string",
      "NetworkInterfaceId": "string",
      "OwnerId": "string",
      "SubnetId": "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 (p. 234)

If the request included the `Marker`, the response returns that value in this field.

Type: String

### MountTargets (p. 234)

Returns the file system's mount targets as an array of `MountTargetDescription` objects.

Type: Array of [MountTargetDescription \(p. 273\)](#) objects

### NextMarker (p. 234)

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

## 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 account.

HTTP Status Code: 404

## Example

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)

- [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 requires the following URI parameters.

### **MountTargetId** (p. 238)

The ID of the mount target whose security groups you want to retrieve.

## 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** (p. 238)

An array of security groups.

Type: Array of strings

Array Members: Maximum number of 5 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

### **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 account.

HTTP Status Code: 404

## Example

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)



## DescribeTags

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 requires the following URI parameters.

### `FileSystemId` (p. 241)

The ID of the file system whose tag set you want to retrieve.

### `Marker` (p. 241)

(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.

### `MaxItems` (p. 241)

(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 (p. 241)

If the request included a `Marker`, the response returns that value in this field.

Type: String

#### NextMarker (p. 241)

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

#### Tags (p. 241)

Returns tags associated with the file system as an array of `Tag` objects.

Type: Array of [Tag \(p. 277\)](#) 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

## Example

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 requires the following URI parameters.

#### MaxResults (p. 244)

(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 (p. 244)

You can use `NextToken` in a subsequent request to fetch the next page of access point descriptions if the response payload was paginated.

#### ResourceId (p. 244)

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.

### 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 (p. 244)**

`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

### **Tags (p. 244)**

An array of the tags for the specified EFS resource.

Type: Array of [Tag \(p. 277\)](#) 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 \(p. 200\)](#). 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 requires the following URI parameters.

### **MountTargetId (p. 246)**

The ID of the mount target whose security groups you want to modify.

## Request Body

The request accepts the following data in JSON format.

### **SecurityGroups (p. 246)**

An array of up to five VPC security group IDs.

Type: Array of strings

Array Members: Maximum number of 5 items.

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 account.

HTTP Status Code: 404

### **SecurityGroupLimitExceeded**

Returned if the size of `SecurityGroups` specified in the request is greater than five.

HTTP Status Code: 400

### **SecurityGroupNotFound**

Returned if one of the specified security groups doesn't exist in the subnet's VPC.

HTTP Status Code: 400

## Example

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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. 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](#).

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 requires the following URI parameters.

#### **FileSystemId** (p. 249)

The ID of the EFS file system that you want to create or update the `FileSystemPolicy` for.

### Request Body

The request accepts the following data in JSON format.

#### **BypassPolicyLockoutSafetyCheck** (p. 249)

(Optional) A flag to indicate whether to bypass the `FileSystemPolicy` lockout safety check. The policy lockout safety check determines whether the policy in the request will prevent the principal making the request will be locked out from making future `PutFileSystemPolicy` requests on the file system. Set `BypassPolicyLockoutSafetyCheck` to `True` only when you intend to prevent the principal that is making the request from making a subsequent `PutFileSystemPolicy` request on the file system. The default value is `False`.

Type: Boolean

Required: No

#### **Policy** (p. 249)

The `FileSystemPolicy` that you're creating. Accepts a JSON formatted policy definition. To find out more about the elements that make up a file system policy, see [EFS Resource-based Policies](#).

Type: String

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 (p. 249)

Specifies the EFS file system to which the `FileSystemPolicy` applies.

Type: String

### Policy (p. 249)

The JSON formatted `FileSystemPolicy` for the EFS file system.

Type: String

## Errors

### 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 an invalid parameter value or a missing required parameter. Returned in the case of a policy lockout safety check error.

HTTP Status Code: 400

## Example

### 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-1:0123456789abc: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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## PutLifecycleConfiguration

Enables lifecycle management by creating a new `LifecycleConfiguration` object. A `LifecycleConfiguration` object defines when files in an Amazon EFS file system are automatically transitioned to the lower-cost EFS Infrequent Access (IA) storage class. A `LifecycleConfiguration` applies to all files in a file system.

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` and disables lifecycle management.

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 files are moved to the IA storage class. The array can contain only one `LifecyclePolicy` item.

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 (AWS KMS) 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": [
    {
      "TransitionToIA": "string"
    }
  ]
}
```

## URI Request Parameters

The request requires the following URI parameters.

### `FileSystemId` (p. 252)

The ID of the file system for which you are creating the `LifecycleConfiguration` object (String).

## Request Body

The request accepts the following data in JSON format.

### `LifecyclePolicies` (p. 252)

An array of `LifecyclePolicy` objects that define the file system's `LifecycleConfiguration` object. A `LifecycleConfiguration` object tells lifecycle management when to transition files from the Standard storage class to the Infrequent Access storage class.

Type: Array of [LifecyclePolicy \(p. 272\)](#) objects

Required: Yes

## Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
  "LifecyclePolicies": [
    {
      "TransitionToIA": "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 \(p. 253\)](#)

An array of lifecycle management policies. Currently, EFS supports a maximum of one policy per file system.

Type: Array of [LifecyclePolicy \(p. 272\)](#) 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

### **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` operation. This object tells EFS lifecycle management to move all files in the file system that haven't been accessed in the last 14 days to the IA storage class. This is the only lifecycle policy that is currently supported. To learn more, see [EFS Lifecycle Management](#).

#### 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": [
    {
      "TransitionToIA": "AFTER_14_DAYS"
    }
  ]
}
```

#### Sample Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-type: application/json
Content-Length: 86

{
  "LifecyclePolicies": [
    {
      "TransitionToIA": "AFTER_14_DAYS"
    }
  ]
}
```

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 requires the following URI parameters.

### **ResourceId** (p. 256)

The ID specifying the EFS resource that you want to create a tag for.

## Request Body

The request accepts the following data in JSON format.

### **Tags** (p. 256)

Type: Array of [Tag](#) (p. 277) 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

## 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 HTTP/1.1
Content-type: application/json

{
  "TagKeys": [ "string" ]
}
```

### URI Request Parameters

The request requires the following URI parameters.

#### **ResourceId** (p. 258)

Specifies the EFS resource that you want to remove tags from.

### Request Body

The request accepts the following data in JSON format.

#### **TagKeys** (p. 258)

The keys of the key:value tag pairs that you want to remove from the specified EFS resource.

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.

Required: No

### 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 requires the following URI parameters.

### **FileSystemId** (p. 260)

The ID of the file system that you want to update.

## Request Body

The request accepts the following data in JSON format.

### **ProvisionedThroughputInMibps** (p. 260)

(Optional) The amount of throughput, in MiB/s, that you want to provision for your file system. Valid values are 1-1024. Required if `ThroughputMode` is changed to `provisioned` on update. If you're not updating the amount of provisioned throughput for your file system, you don't need to provide this value in your request.

Type: Double

Valid Range: Minimum value of 1.0.

Required: No

### **ThroughputMode** (p. 260)

(Optional) The throughput mode that you want your file system to use. 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`

Required: No

## Response Syntax

```
HTTP/1.1 202
Content-type: application/json
```

```
{
  "CreationTime": number,
  "CreationToken": "string",
  "Encrypted": boolean,
  "FileSystemId": "string",
  "KmsKeyId": "string",
  "LifecycleState": "string",
  "Name": "string",
  "NumberOfMountTargets": number,
  "OwnerId": "string",
  "PerformanceMode": "string",
  "ProvisionedThroughputInMibps": number,
  "SizeInBytes": {
    "Timestamp": number,
    "Value": 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.

### CreationTime (p. 260)

The time that the file system was created, in seconds (since 1970-01-01T00:00:00Z).

Type: Timestamp

### CreationToken (p. 260)

The opaque string specified in the request.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

### Encrypted (p. 260)

A Boolean value that, if true, indicates that the file system is encrypted.

Type: Boolean

### FileSystemId (p. 260)

The ID of the file system, assigned by Amazon EFS.

Type: String

### KmsKeyId (p. 260)

The ID of an AWS Key Management Service (AWS KMS) customer master key (CMK) that was used to protect the encrypted file system.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

#### **LifeCycleState (p. 260)**

The lifecycle phase of the file system.

Type: String

Valid Values: `creating` | `available` | `updating` | `deleting` | `deleted`

#### **Name (p. 260)**

You can add tags to a file system, including a Name tag. For more information, see [CreateFileSystem \(p. 192\)](#). If the file system has a Name tag, Amazon EFS returns the value in this field.

Type: String

Length Constraints: Maximum length of 256.

#### **NumberOfMountTargets (p. 260)**

The current number of mount targets that the file system has. For more information, see [CreateMountTarget \(p. 200\)](#).

Type: Integer

Valid Range: Minimum value of 0.

#### **OwnerId (p. 260)**

The AWS account that created the file system. If the file system was created by an IAM user, the parent account to which the user belongs is the owner.

Type: String

#### **PerformanceMode (p. 260)**

The performance mode of the file system.

Type: String

Valid Values: `generalPurpose` | `maxIO`

#### **ProvisionedThroughputInMibps (p. 260)**

The throughput, measured in MiB/s, that you want to provision for a file system. Valid values are 1-1024. Required if `ThroughputMode` is set to `provisioned`. The limit on throughput is 1024 MiB/s. You can get these limits increased by contacting AWS Support. For more information, see [Amazon EFS Limits That You Can Increase](#) in the *Amazon EFS User Guide*.

Type: Double

Valid Range: Minimum value of 1.0.

#### **SizeInBytes (p. 260)**

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 \(p. 271\)](#) object

### Tags (p. 260)

The tags associated with the file system, presented as an array of `Tag` objects.

Type: Array of [Tag \(p. 277\)](#) objects

### ThroughputMode (p. 260)

The throughput mode for a file system. There are two throughput modes to choose from for your file system: `bursting` and `provisioned`. If you set `ThroughputMode` to `provisioned`, you must also set a value for `ProvisionedThroughPutInMibps`. You can decrease your file system's throughput in `Provisioned Throughput` mode or change between the throughput modes as long as it's been more than 24 hours since the last decrease or throughput mode change.

Type: String

Valid Values: `bursting` | `provisioned`

## 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` to `provisioned throughput` mode.

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 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](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

## Data Types

The following data types are supported:

- [AccessPointDescription](#) (p. 265)
- [CreationInfo](#) (p. 267)
- [FileSystemDescription](#) (p. 268)
- [FileSystemSize](#) (p. 271)
- [LifecyclePolicy](#) (p. 272)
- [MountTargetDescription](#) (p. 273)
- [PosixUser](#) (p. 275)
- [RootDirectory](#) (p. 276)
- [Tag](#) (p. 277)



## 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

Required: No

#### **AccessPointId**

The ID of the access point, assigned by Amazon EFS.

Type: String

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.

Required: No

#### **FileSystemId**

The ID of the EFS file system that the access point applies to.

Type: String

Required: No

#### **LifeCycleState**

Identifies the lifecycle phase of the access point.

Type: String

Valid Values: `creating` | `available` | `updating` | `deleting` | `deleted`

Required: No

#### **Name**

The name of the access point. This is the value of the `Name` tag.

Type: String

Required: No

#### **OwnerId**

Identified the AWS account that owns the access point resource.

Type: String

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 \(p. 275\)](#) object

Required: No

**RootDirectory**

The directory on the Amazon EFS file system that the access point exposes as the root directory to NFS clients using the access point.

Type: [RootDirectory \(p. 276\)](#) object

Required: No

**Tags**

The tags associated with the access point, presented as an array of Tag objects.

Type: Array of [Tag \(p. 277\)](#) 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 Go](#)
- [AWS SDK for Java](#)
- [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.

### 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

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 Go](#)
- [AWS SDK for Java](#)
- [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.

Required: Yes

### **Encrypted**

A Boolean value that, if true, indicates that the file system is encrypted.

Type: Boolean

Required: No

### **FileSystemId**

The ID of the file system, assigned by Amazon EFS.

Type: String

Required: Yes

### **KmsKeyId**

The ID of an AWS Key Management Service (AWS KMS) customer master key (CMK) that was used to protect the encrypted file system.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Required: No

### **LifeCycleState**

The lifecycle phase of the file system.

Type: String

Valid Values: `creating` | `available` | `updating` | `deleting` | `deleted`

Required: Yes

### **Name**

You can add tags to a file system, including a `Name` tag. For more information, see [CreateFileSystem \(p. 192\)](#). If the file system has a `Name` tag, Amazon EFS returns the value in this field.

Type: String

Length Constraints: Maximum length of 256.

Required: No

#### **NumberOfMountTargets**

The current number of mount targets that the file system has. For more information, see [CreateMountTarget \(p. 200\)](#).

Type: Integer

Valid Range: Minimum value of 0.

Required: Yes

#### **OwnerId**

The AWS account that created the file system. If the file system was created by an IAM user, the parent account to which the user belongs is the owner.

Type: String

Required: Yes

#### **PerformanceMode**

The performance mode of the file system.

Type: String

Valid Values: `generalPurpose` | `maxIO`

Required: Yes

#### **ProvisionedThroughputInMibps**

The throughput, measured in MiB/s, that you want to provision for a file system. Valid values are 1-1024. Required if `ThroughputMode` is set to `provisioned`. The limit on throughput is 1024 MiB/s. You can get these limits increased by contacting AWS Support. For more information, see [Amazon EFS Limits That You Can Increase](#) in the *Amazon EFS User Guide*.

Type: Double

Valid Range: Minimum value of 1.0.

Required: No

#### **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 \(p. 271\)](#) object

Required: Yes

#### **Tags**

The tags associated with the file system, presented as an array of `Tag` objects.

Type: Array of [Tag \(p. 277\)](#) objects

Required: Yes

### **ThroughputMode**

The throughput mode for a file system. There are two throughput modes to choose from for your file system: `bursting` and `provisioned`. If you set `ThroughputMode` to `provisioned`, you must also set a value for `ProvisionedThroughPutInMibps`. You can decrease your file system's throughput in Provisioned Throughput mode or change between the throughput modes as long as it's been more than 24 hours since the last decrease or throughput mode change.

Type: String

Valid Values: `bursting` | `provisioned`

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 Go](#)
- [AWS SDK for Java](#)
- [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

### 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

### 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

### 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 Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V3](#)

## LifecyclePolicy

Describes a policy used by EFS lifecycle management to transition files to the Infrequent Access (IA) storage class.

### Contents

#### TransitionToIA

A value that describes the period of time that a file is not accessed, after which it transitions to the IA storage class. 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`

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 Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V3](#)



# MountTargetDescription

Provides a description of a mount target.

## Contents

### **AvailabilityZoneId**

The unique and consistent identifier of the Availability Zone (AZ) 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 (AZ) that the mount target resides in. AZs 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

Required: No

### **FileSystemId**

The ID of the file system for which the mount target is intended.

Type: String

Required: Yes

### **IpAddress**

Address at which the file system can be mounted by using the mount target.

Type: String

Required: No

### **LifeCycleState**

Lifecycle state of the mount target.

Type: String

Valid Values: `creating` | `available` | `updating` | `deleting` | `deleted`

Required: Yes

### **MountTargetId**

System-assigned mount target ID.

Type: String

Required: Yes

### **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

Required: No

**SubnetId**

The ID of the mount target's subnet.

Type: String

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 Go](#)
- [AWS SDK for Java](#)
- [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

#### **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

#### **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

### 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 Go](#)
- [AWS SDK for Java](#)
- [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 \(p. 267\)](#) 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.

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 Go](#)
- [AWS SDK for Java](#)
- [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.

Required: Yes

### Value

The value of the tag key.

Type: String

Length Constraints: Maximum length of 256.

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 Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V3](#)

# Additional Information for Amazon EFS

Following, you can find some additional information about Amazon EFS, including features that are still supported but not necessarily recommended.

## Topics

- [Backing Up Amazon EFS File Systems Using AWS Data Pipeline \(p. 278\)](#)
- [Mounting File Systems Without the EFS Mount Helper \(p. 290\)](#)

## Backing Up Amazon EFS File Systems Using AWS Data Pipeline

### Note

Using the AWS Data Pipeline to back up your EFS file systems is a legacy solution.

## Recommended EFS Backup Solutions

There are two recommended solutions available for backing up your EFS file systems.

- AWS Backup service
- The EFS-to-EFS backup solution

AWS Backup is a simple and cost-effective way to back up your Amazon EFS file systems. AWS Backup is a unified backup service designed to simplify the creation, migration, restoration, and deletion of backups, while providing improved reporting and auditing. For more information, see [Using AWS Backup with Amazon EFS \(p. 98\)](#).

The EFS-to-EFS backup solution is suitable for all Amazon EFS file systems in all AWS Regions. It includes an AWS CloudFormation template that launches, configures, and runs the AWS services required to deploy this solution. This solution follows AWS best practices for security and availability. For more information, see [EFS-to-EFS Backup Solution](#) in AWS Answers.

## Legacy EFS Backup Solution Using AWS Data Pipeline

Using AWS Data Pipeline to backup EFS file system is a legacy backup solution. In this backup solution, you create a data pipeline by using the AWS Data Pipeline service. This pipeline copies data from your Amazon EFS file system (called the *production file system*) to another Amazon EFS file system (called the *backup file system*).

This solution consists of AWS Data Pipeline templates that implement the following:

- Automated EFS backups based on a schedule that you define (for example, hourly, daily, weekly, or monthly).

- Automated rotation of the backups, where the oldest backup is replaced with the newest backup based on the number of backups that you want to retain.
- Quicker backups using rsync, which only back up the changes made between one backup to the next.
- Efficient storage of backups using hard links. A *hard link* is a directory entry that associates a name with a file in a file system. By setting up a hard link, you can perform a full restoration of data from any backup while only storing what changed from backup to backup.

After you set up the backup solution, this walkthrough shows you how to access your backups to restore your data. This backup solution depends on running scripts that are hosted on GitHub, and is therefore subject to GitHub availability. If you'd prefer to eliminate this reliance and host the scripts in an Amazon S3 bucket instead, see [Hosting the rsync Scripts in an Amazon S3 Bucket \(p. 289\)](#).

### Important

This solution requires using AWS Data Pipeline in the same AWS Region as your file system. Because AWS Data Pipeline is not supported in US East (Ohio), this solution doesn't work in that AWS Region. We recommend that if you want to back up your file system using this solution, you use your file system in one of the other supported AWS Regions.

### Topics

- [Performance for Amazon EFS Backups Using AWS Data Pipeline \(p. 279\)](#)
- [Considerations for Amazon EFS Backup by Using AWS Data Pipeline \(p. 280\)](#)
- [Assumptions for Amazon EFS Backup with AWS Data Pipeline \(p. 280\)](#)
- [How to Back Up an Amazon EFS File System with AWS Data Pipeline \(p. 281\)](#)
- [Additional Backup Resources \(p. 286\)](#)

## Performance for Amazon EFS Backups Using AWS Data Pipeline

When performing data backups and restorations, your file system performance is subject to [Amazon EFS Performance \(p. 90\)](#), including baseline and burst throughput capacity. The throughput used by your backup solution counts toward your total file system throughput. The following table provides some recommendations for the Amazon EFS file system and Amazon EC2 instance sizes that work for this solution, assuming that your backup window is 15 minutes long.

EFS Size (30 MB Average File Size)	Daily Change Volume	Remaining Burst Hours	Minimum Number of Backup Agents
256 GB	Less than 25 GB	6.75	1 - m3.medium
512 GB	Less than 50 GB	7.75	1 - m3.large
1.0 TB	Less than 75 GB	11.75	2 - m3.large*
1.5 TB	Less than 125 GB	11.75	2 - m3.xlarge*
2.0 TB	Less than 175 GB	11.75	3 - m3.large*
3.0 TB	Less than 250 GB	11.75	4 - m3.xlarge*

\* These estimates are based on the assumption that data stored in an EFS file system that is 1 TB or larger is organized so that the backup can be spread across multiple backup nodes. The multiple-node

example scripts divide the backup load across nodes based on the contents of the first-level directory of your EFS file system.

For example, if there are two backup nodes, one node backs up all of the even files and directories located in the first-level directory. The odd node does the same for the odd files and directories. In another example, with six directories in the Amazon EFS file system and four backup nodes, the first node backs up the first and the fifth directories. The second node backs up the second and the sixth directories, and the third and fourth nodes back up the third and the fourth directories respectively.

## Considerations for Amazon EFS Backup by Using AWS Data Pipeline

Consider the following when you're deciding whether to implement an Amazon EFS backup solution using AWS Data Pipeline:

- This approach to EFS backup involves a number of AWS resources. For this solution, you need to create the following:
  - One production file system and one backup file system that contains a full copy of the production file system. The system also contains any incremental changes to your data over the backup rotation period.
  - Amazon EC2 instances, whose lifecycles are managed by AWS Data Pipeline, that perform restorations and scheduled backups.
  - One regularly scheduled AWS Data Pipeline for backing up data.
  - An AWS Data Pipeline for restoring backups.

When this solution is implemented, it results in billing to your account for these services. For more information, see the pricing pages for [Amazon EFS](#), [Amazon EC2](#), and [AWS Data Pipeline](#).

- This solution isn't an offline backup solution. To ensure a fully consistent and complete backup, pause any file writes to the file system or unmount the file system while the backup occurs. We recommend that you perform all backups during scheduled downtime or off hours.

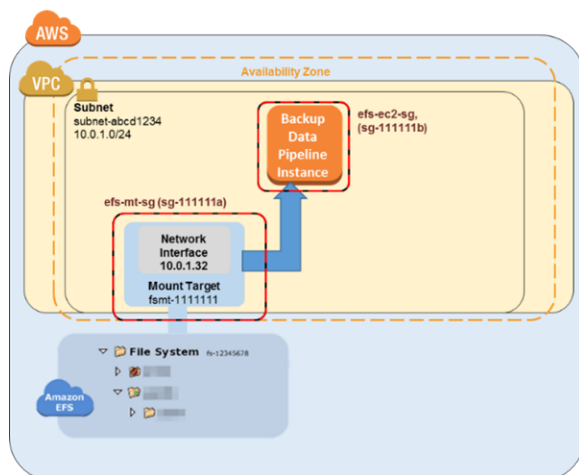
## Assumptions for Amazon EFS Backup with AWS Data Pipeline

This walkthrough makes several assumptions and declares example values as follows:

- Before you get started, this walkthrough assumes that you already completed [Getting Started \(p. 11\)](#).
- After you've completed the Getting Started exercise, you have two security groups, a VPC subnet, and a file system mount target for the file system that you want to back up. For the rest of this walkthrough, you use the following example values:
  - The ID of the file system that you back up in this walkthrough is `fs-12345678`.
  - The security group for the file system that is associated with the mount target is called `efs-mt-sg (sg-1111111a)`.
  - The security group that grants Amazon EC2 instances the ability to connect to the production EFS mount point is called `efs-ec2-sg (sg-1111111b)`.
  - The VPC subnet has the ID value of `subnet-abcd1234`.
  - The source file system mount target IP address for the file system that you want to back up is `10.0.1.32:/`.
  - The example assumes that the production file system is a content management system serving media files with an average size of 30 MB.



The preceding assumptions and examples are reflected in the following initial setup diagram.



## How to Back Up an Amazon EFS File System with AWS Data Pipeline

Follow the steps in this section to back up or restore your Amazon EFS file system with AWS Data Pipeline.

### Topics

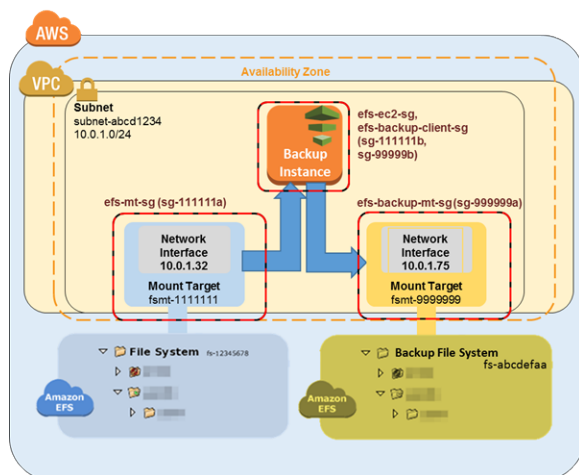
- [Step 1: Create Your Backup Amazon EFS File System \(p. 281\)](#)
- [Step 2: Download the AWS Data Pipeline Template for Backups \(p. 282\)](#)
- [Step 3: Create a Data Pipeline for Backup \(p. 282\)](#)
- [Step 4: Access Your Amazon EFS Backups \(p. 283\)](#)

## Step 1: Create Your Backup Amazon EFS File System

In this walkthrough, you create separate security groups, file systems, and mount points to separate your backups from your data source. In this first step, you create those resources:

1. First, create two new security groups. The example security group for the backup mount target is `efs-backup-mt-sg (sg-9999999a)`. The example security group for the EC2 instance to access the mount target is `efs-backup-ec2-sg (sg-9999999b)`. Remember to create these security groups in the same VPC as the EFS volume that you want to back up. In this example, the VPC associated with the `subnet-abcd1234` subnet. For more information about creating security groups, see [Creating Security Groups \(p. 27\)](#).
2. Next, create a backup Amazon EFS file system. In this example, the file system ID is `fs-abcdefaa`. For more information about creating file systems, see [Creating an Amazon Elastic File System \(p. 19\)](#).
3. Finally, create a mount point for the EFS backup file system and assume that it has the value of `10.0.1.75 : /`. For more information about creating mount targets, see [Creating Mount Targets \(p. 22\)](#).

After you've completed this first step, your setup should look similar to the following example diagram.



## Step 2: Download the AWS Data Pipeline Template for Backups

AWS Data Pipeline helps you reliably process and move data between different AWS compute and storage services at specified intervals. By using the AWS Data Pipeline console, you can create preconfigured pipeline definitions, known as templates. You can use these templates to get started with AWS Data Pipeline quickly. For this walkthrough, a template is provided to make the process of setting up your backup pipeline easier.

When implemented, this template creates a data pipeline that launches a single Amazon EC2 instance on the schedule that you specify to back up data from the production file system to the backup file system. This template has a number of placeholder values. You provide the matching values for those placeholders in the **Parameters** section of the AWS Data Pipeline console. Download the AWS Data Pipeline template for backups at [1-Node-EFSBackupDataPipeline.json](#) from GitHub.

### Note

This template also references and runs a script to perform the backup commands. You can download the script before creating the pipeline to review what it does. To review the script, download [efs-backup.sh](#) from GitHub. This backup solution depends on running scripts that are hosted on GitHub and is subject to GitHub availability. If you'd prefer to eliminate this reliance and host the scripts in an Amazon S3 bucket instead, see [Hosting the rsync Scripts in an Amazon S3 Bucket](#) (p. 289).

## Step 3: Create a Data Pipeline for Backup

Use the following procedure to create your data pipeline.

### To create a data pipeline for Amazon EFS backups

1. Open the AWS Data Pipeline console at <https://console.aws.amazon.com/datapipeline/>.

#### Important

Make sure that you're working in the same AWS Region as your Amazon EFS file systems.

2. Choose **Create new pipeline**.
3. Add values for **Name** and optionally for **Description**.
4. For **Source**, choose **Import a definition**, and then choose **Load local file**.
5. In the file explorer, navigate to the template that you saved in [Step 2: Download the AWS Data Pipeline Template for Backups](#) (p. 282), and then choose **Open**.
6. In **Parameters**, provide the details for both your backup and production EFS file systems.

**Parameters**

Production EFS mount target IP address.	10.0.1.32/
Security group that can connect to the Production EFS mount point.	sg-1111111b
Interval for backups.	daily
Security group that can connect to the Backup EFS mount point.	sg-9999999b
Number of backups to retain.	7
Backup EFS mount target IP address.	10.0.1.75/
VPC subnet for your backup EC2 instance (ideally the same subnet used for the production EFS mount point).	subnet-1234abcd
Instance type for creating backups.	m3.medium
Name for the directory that will contain your backups.	backup-fs-12345678
Shell command to run.	wget https://raw.githubusercontent.com/aws-labs/data-pipeline-

- Configure the options in **Schedule** to define your Amazon EFS backup schedule. The backup in the example runs once every day, and the backups are kept for a week. When a backup is seven days old, it is replaced with next oldest backup.

**Schedule**

**You can run your pipeline once or specify a schedule. [More](#)**

**Run** ☐ once on pipeline activation ☒ on a schedule

**Run every**  day(s)

**Starting** ☒ on pipeline activation ☐ 2016-05-28 02:46 UTC (Current time is 02:48 UTC)  
YYYY-MM-DD HH:MM

**Ending** ☒ never ☐ after 1 occurrence(s) ☐ 2016-05-29 02:46 UTC (Current time is 02:48 UTC)  
YYYY-MM-DD HH:MM

#### Note

We recommend that you specify a run time that occurs during your off-peak hours.

- (Optional) Specify an Amazon S3 location for storing pipeline logs, configure a custom IAM role, or add tags to describe your pipeline.
- When your pipeline is configured, choose **Activate**.

You've now configured and activated your Amazon EFS backup data pipeline. For more information about AWS Data Pipeline, see the [AWS Data Pipeline Developer Guide](#). At this stage, you can perform the backup now as a test, or you can wait until the backup is performed at the scheduled time.

## Step 4: Access Your Amazon EFS Backups

Your Amazon EFS backup has now been created, activated, and is running on the schedule you defined. This step outlines how you can access your EFS backups. Your backups are stored in the EFS backup file system that you created in the following format.

```
backup-efs-mount-target:/efs-backup-id/[backup interval].[0-backup retention]-->
```

Using the values from the example scenario, the backup of the file system is located in `10.1.0.75:/fs-12345678/daily.[0-6]`, where `daily.0` is the most recent backup and `daily.6` is the oldest of the seven rotating backups.

Accessing your backups gives you the ability to restore data to your production file system. You can choose to restore an entire file system, or you can choose to restore individual files.

## Step 4.1: Restore an Entire Amazon EFS Backup

Restoring a backup copy of an Amazon EFS file system requires another AWS Data Pipeline, similar to the one you configured in [Step 3: Create a Data Pipeline for Backup \(p. 282\)](#). However, this restoration pipeline works in the reverse of the backup pipeline. Typically, these restorations aren't scheduled to begin automatically.

As with backups, restores can be done in parallel to meet your recovery time objective. Keep in mind that when you create a data pipeline, you need to schedule when you want it run. If you choose to run on activation, you start the restoration process immediately. We recommend that you only create a restoration pipeline when you need to do a restoration, or when you already have a specific window of time in mind.

Burst capacity is consumed by both the backup EFS and restoration EFS. For more information about performance, see [Amazon EFS Performance \(p. 90\)](#). The following procedure shows you how to create and implement your restoration pipeline.

### To create a data pipeline for EFS data restoration

1. Download the data pipeline template for restoring data from your backup EFS file system. This template launches a single Amazon EC2 instance based on the specified size. It launches only when you specify it to launch. Download the AWS Data Pipeline template for backups at [1-Node-EFSRestoreDataPipeline.json](#) from GitHub.

#### Note

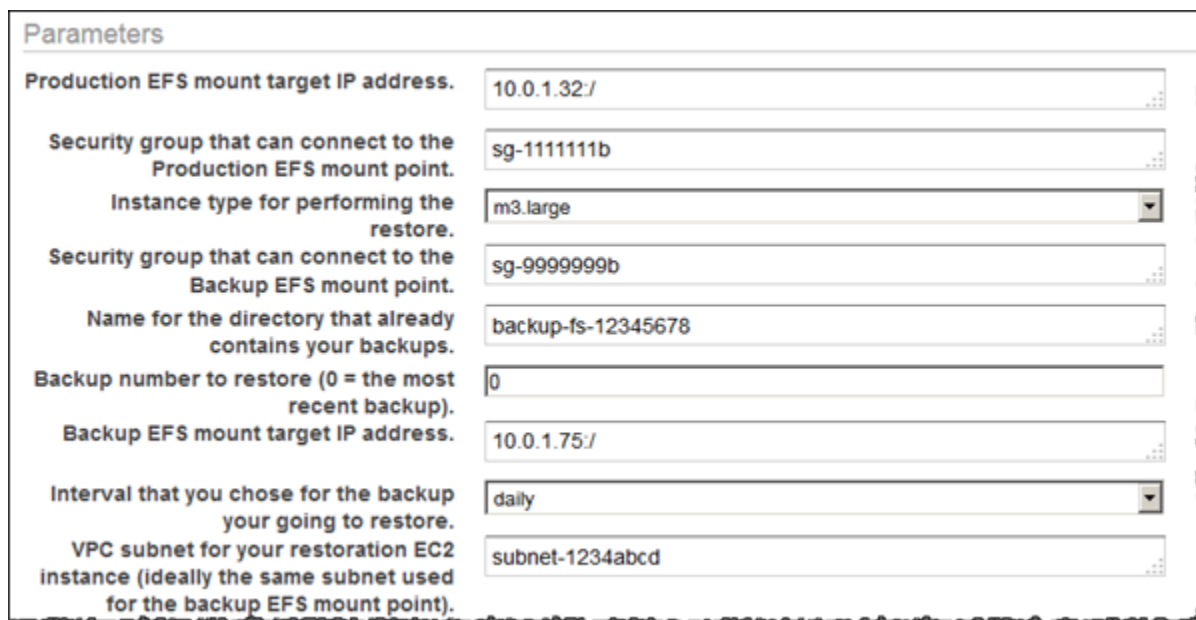
This template also references and runs a script to perform the restoration commands. You can download the script before creating the pipeline to review what it does. To review the script, download [efs-restore.sh](#) from GitHub.

2. Open the AWS Data Pipeline console at <https://console.aws.amazon.com/datapipeline/>.

#### Important

Make sure that you're working in the same AWS Region as your Amazon EFS file systems and Amazon EC2.

3. Choose **Create new pipeline**.
4. Add values for **Name** and optionally for **Description**.
5. For **Source**, choose **Import a definition**, and then choose **Load local file**.
6. In the file explorer, navigate to the template that you saved in [Step 1: Create Your Backup Amazon EFS File System \(p. 281\)](#), and then choose **Open**.
7. In **Parameters**, provide the details for both your backup and production EFS file systems.



Parameters	
Production EFS mount target IP address.	10.0.1.32/
Security group that can connect to the Production EFS mount point.	sg-1111111b
Instance type for performing the restore.	m3.large
Security group that can connect to the Backup EFS mount point.	sg-9999999b
Name for the directory that already contains your backups.	backup-fs-12345678
Backup number to restore (0 = the most recent backup).	0
Backup EFS mount target IP address.	10.0.1.75/
Interval that you chose for the backup your going to restore.	daily
VPC subnet for your restoration EC2 instance (ideally the same subnet used for the backup EFS mount point).	subnet-1234abcd

- Because you typically perform restorations only when you need them, you can schedule the restoration to run **once on pipeline activation**. Or schedule a one-time restoration at a future time of your choosing, like during an off-peak window of time.
- (Optional) Specify an Amazon S3 location for storing pipeline logs, configure a custom IAM role, or add tags to describe your pipeline.
- When your pipeline is configured, choose **Activate**.

You've now configured and activated your Amazon EFS restoration data pipeline. Now when you need to restore a backup to your production EFS file system, you just activate it from the AWS Data Pipeline console. For more information, see the [AWS Data Pipeline Developer Guide](#).

## Step 4.2: Restore Individual Files from Your Amazon EFS Backups

You can restore files from your Amazon EFS file system backups by launching an Amazon EC2 instance to temporarily mount both the production and backup EFS file systems. The EC2 instance must be a member of both of the EFS client security groups (in this example, **efs-ec2-sg** and **efs-backup-clients-sg**). Both EFS mount targets can be mounted by this restoration instance. For example, a recovery EC2 instance can create the following mount points. Here, the `-o ro` option is used to mount the Backup EFS as read-only to prevent accidentally modifying the backup when attempting to restore from a backup.

```
mount -t nfs source-efs-mount-target:/ /mnt/data
```

```
mount -t nfs -o ro backup-efs-mount-target:/fs-12345678/daily.0 /mnt/backup>
```

After you've mounted the targets, you can copy files from `/mnt/backup` to the appropriate location in `/mnt/data` in the terminal using the `cp -p` command. For example, an entire home directory (with its file system permissions) can be recursively copied with the following command.

```
sudo cp -rp /mnt/backup/users/my_home /mnt/data/users/my_home
```

You can restore a single file by running the following command.

```
sudo cp -p /mnt/backup/user/my_home/.profile /mnt/data/users/my_home/.profile
```

### Warning

When you are manually restoring individual data files, be careful that you don't accidentally modify the backup itself. Otherwise, you might corrupt it.

## Additional Backup Resources

The backup solution presented in this walkthrough uses templates for AWS Data Pipeline. The templates used in [Step 2: Download the AWS Data Pipeline Template for Backups \(p. 282\)](#) and [Step 4.1: Restore an Entire Amazon EFS Backup \(p. 284\)](#) both use a single Amazon EC2 instance to perform their work. However, there's no real limit to the number of parallel instances that you can run for backing up or restoring your data in Amazon EFS file systems. In this topic, you can find links to other AWS Data Pipeline templates configured for multiple EC2 instances that you can download and use for your backup solution. You can also find instructions for how to modify the templates to include additional instances.

### Topics

- [Using Additional Templates \(p. 286\)](#)
- [Adding Additional Backup Instances \(p. 286\)](#)
- [Adding Additional Restoration Instances \(p. 288\)](#)
- [Hosting the rsync Scripts in an Amazon S3 Bucket \(p. 289\)](#)

## Using Additional Templates

You can download the following additional templates from GitHub:

- [2-Node-EFSBackupPipeline.json](#) – This template starts two parallel Amazon EC2 instances to back up your production Amazon EFS file system.
- [2-Node-EFSRestorePipeline.json](#) – This template starts two parallel Amazon EC2 instances to restore a backup of your production Amazon EFS file system.

## Adding Additional Backup Instances

You can add additional nodes to the backup templates used in this walkthrough. To add a node, modify the following sections of the `2-Node-EFSBackupDataPipeline.json` template.

### Important

If you're using additional nodes, you can't use spaces in file names and directories stored in the top-level directory. If you do, those files and directories aren't backed up or restored. All files and subdirectories that are at least one level below the top level are backed up and restored as expected.

- Create an additional EC2Resource for each additional node you want to create (in this example, a fourth EC2 instance).

```
{
  "id": "EC2Resource4",
  "terminateAfter": "70 Minutes",
  "instanceType": "#{myInstanceType}",
  "name": "EC2Resource4",
  "type": "Ec2Resource",
  "securityGroupIds": [ "#{mySrcSecGroupID}", "#{myBackupSecGroupID}" ],
  "subnetId": "#{mySubnetID}",
  "associatePublicIpAddress": "true"
},
```

- Create an additional data pipeline activity for each additional node (in this case, activity BackupPart4), make sure to configure the following sections:
- Update the runsOn reference to point to the EC2Resource created previously (EC2Resource4 in the following example).
- Increment the last two scriptArgument values to equal the backup part that each node is responsible for and the total number of nodes. For "2" and "3" in the example following, the backup part is "3" for the fourth node because in this example our modulus logic needs to count starting with 0.

```
{
  "id": "BackupPart4",
  "name": "BackupPart4",
  "runsOn": {
    "ref": "EC2Resource4"
  },
  "command": "wget https://raw.githubusercontent.com/awslabs/data-pipeline-samples/master/samples/EFSBackup/efs-backup-rsync.sh\nchmod a+x efs-backup-rsync.sh\n./efs-backup-rsync.sh $1 $2 $3 $4 $5 $6 $7",
  "scriptArgument": [ "#{myEfsSource}", "#{myEfsBackup}", "#{myInterval}",
    "#{myRetainedBackups}", "#{myEfsID}", "3", "4" ],
  "type": "ShellCommandActivity",
  "dependsOn": {
    "ref": "InitBackup"
  },
  "stage": "true"
},
```

- Increment the last value in all existing scriptArgument values to the number of nodes (in this example, "4").

```
{
  "id": "BackupPart1",
  ...
  "scriptArgument": [ "#{myEfsSource}", "#{myEfsBackup}", "#{myInterval}",
    "#{myRetainedBackups}", "#{myEfsID}", "1", "4" ],
  ...
},
{
  "id": "BackupPart2",
  ...
  "scriptArgument": [ "#{myEfsSource}", "#{myEfsBackup}", "#{myInterval}",
    "#{myRetainedBackups}", "#{myEfsID}", "2", "4" ],
  ...
},
{
  "id": "BackupPart3",
  ...
  "scriptArgument": [ "#{myEfsSource}", "#{myEfsBackup}", "#{myInterval}",
    "#{myRetainedBackups}", "#{myEfsID}", "0", "4" ],
  ...
},
```

- Update FinalizeBackup activity and add the new backup activity to the dependsOn list (BackupPart4 in this case).

```
{
  "id": "FinalizeBackup", "name": "FinalizeBackup", "runsOn": { "ref":
    "EC2Resource1" }, "command": "wget
    https://raw.githubusercontent.com/awslabs/data-pipeline-samples/master/samples/EFSBackup/
    efs-backup-end.sh\nchmod a+x
    efs-backup-end.sh\n./efs-backup-end.sh $1 $2", "scriptArgument": [ "#{myInterval}",
    "#{myEfsID}" ], "type": "ShellCommandActivity", "dependsOn": [ { "ref": "BackupPart1" },
```

```
{ "ref": "BackupPart2" }, { "ref": "BackupPart3" }, { "ref": "BackupPart4" } ], "stage": "true"
```

## Adding Additional Restoration Instances

You can add nodes to the restoration templates used in this walkthrough. To add a node, modify the following sections of the `2-Node-EFSRestorePipeline.json` template.

- Create an additional `EC2Resource` for each additional node you want to create (in this case, a third EC2 instance called `EC2Resource3`).

```
{
  "id": "EC2Resource3",
  "terminateAfter": "70 Minutes",
  "instanceType": "#{myInstanceType}",
  "name": "EC2Resource3",
  "type": "EC2Resource",
  "securityGroupIds": [ "#{mySrcSecGroupID}", "#{myBackupSecGroupID}" ],
  "subnetId": "#{mySubnetID}",
  "associatePublicIpAddress": "true"
},
```

- Create an additional data pipeline activity for each additional node (in this case, Activity `RestorePart3`). Make sure to configure the following sections:
  - Update the `runsOn` reference to point to the `EC2Resource` created previously (in this example, `EC2Resource3`).
  - Increment the last two `scriptArgument` values to equal the backup part that each node is responsible for and the total number of nodes. For "2" and "3" in the example following, the backup part is "3" for the fourth node because in this example our modulus logic needs to count starting with 0.

```
{
  "id": "RestorePart3",
  "name": "RestorePart3",
  "runsOn": {
    "ref": "EC2Resource3"
  },
  "command": "wget https://raw.githubusercontent.com/awslabs/data-pipeline-samples/master/samples/EFSBackup/efs-restore-rsync.sh\nchmod a+x efs-restore-rsync.sh\n./efs-backup-rsync.sh $1 $2 $3 $4 $5 $6 $7",
  "scriptArgument": [ "#{myEfsSource}", "#{myEfsBackup}", "#{myInterval}",
    "#{myBackup}", "#{myEfsID}", "2", "3" ],
  "type": "ShellCommandActivity",
  "dependsOn": {
    "ref": "InitBackup"
  },
  "stage": "true"
},
```

- Increment the last value in all existing `scriptArgument` values to the number of nodes (in this example, "3").

```
{
  "id": "RestorePart1",
  ...
  "scriptArgument": [ "#{myEfsSource}", "#{myEfsBackup}", "#{myInterval}",
    "#{myBackup}", "#{myEfsID}", "1", "3" ],
  ...
},
```



```
{
  "id": "RestorePart2",
  ...
  "scriptArgument": [ "#{myEfsSource}", "#{myEfsBackup}", "#{myInterval}",
    "#{myBackup}", "#{myEfsID}", "0", "3" ],
  ...
},
```

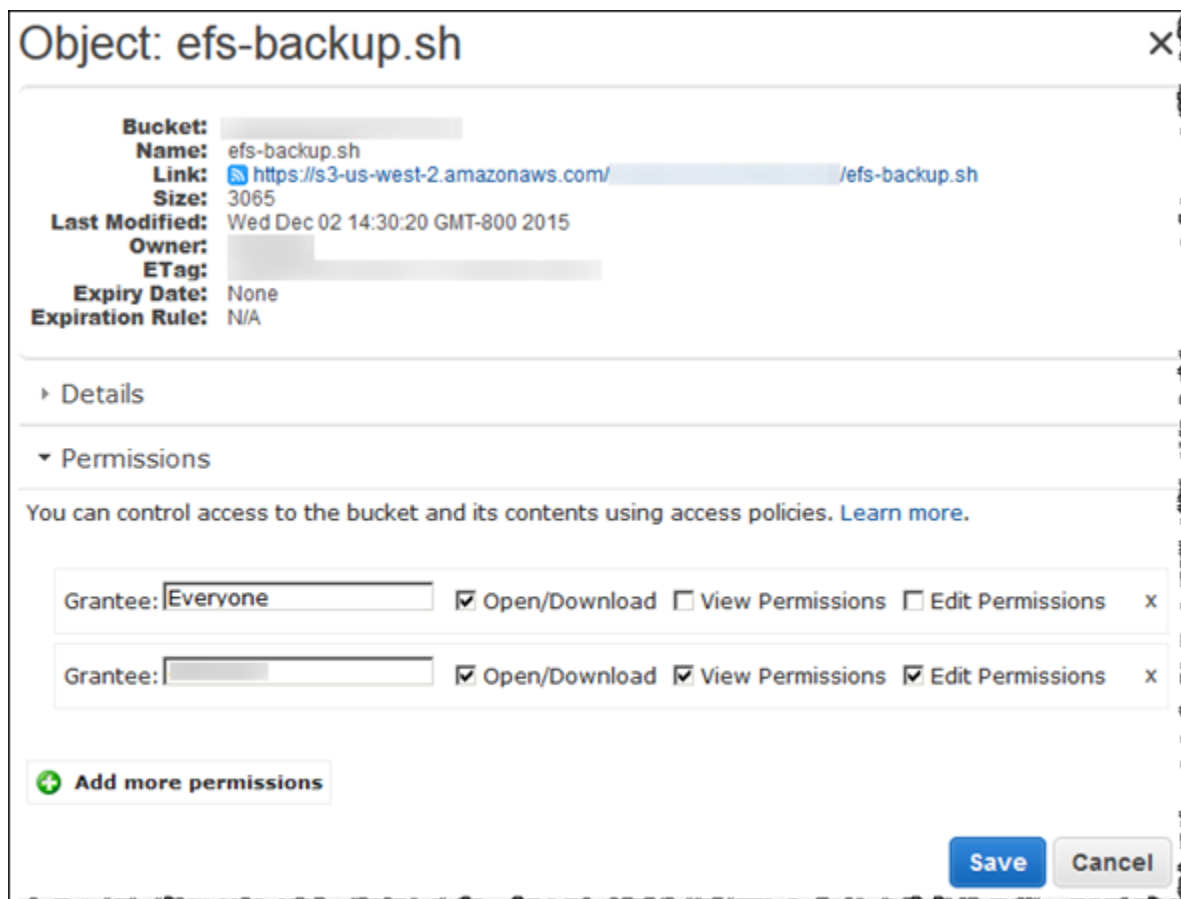
## Hosting the rsync Scripts in an Amazon S3 Bucket

This backup solution is dependent on running rsync scripts that are hosted in a GitHub repository on the internet. Therefore, this backup solution is subject to the GitHub repository being available. This requirement means that if the GitHub repository removes these scripts, or if the GitHub website goes offline, the backup solution as implemented preceding doesn't function.

If you'd prefer to eliminate this GitHub dependency, you can choose to host the scripts in an Amazon S3 bucket that you own instead. Following, you can find the steps necessary to host the scripts yourself.

### To host the rsync scripts in your own Amazon S3 bucket

1. **Sign Up for AWS** – If you already have an AWS account, go ahead and skip to the next step. Otherwise, see [Sign up for AWS \(p. 9\)](#).
2. **Create an AWS Identity and Access Management User** – If you already have an IAM user, go ahead and skip to the next step. Otherwise, see [Create an IAM User \(p. 9\)](#).
3. **Create an Amazon S3 bucket** – If you already have a bucket that you want to host the rsync scripts in, go ahead and skip to the next step. Otherwise, see [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.
4. **Download the rsync scripts and templates** – Download all of the rsync scripts and templates in the [EFSBackup folder](#) from GitHub. Make a note of the location on your computer where you downloaded these files.
5. **Upload the rsync scripts to your S3 bucket** – For instructions on how to upload objects into your S3 bucket, see [Add an Object to a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.
6. Change the permissions on the uploaded rsync scripts to allow **Everyone to Open/Download** them. For instructions on how to change the permissions on an object in your S3 bucket, see [Editing Object Permissions](#) in the *Amazon Simple Storage Service Console User Guide*.



7. **Update your templates** – Modify the `wget` statement in the `shellCmd` parameter to point to the Amazon S3 bucket where you put the startup script. Save the updated template, and use that template when you're following the procedure in [Step 3: Create a Data Pipeline for Backup](#) (p. 282).

**Note**

We recommend that you limit access to your Amazon S3 bucket to include the IAM account that activates the AWS Data Pipeline for this backup solution. For more information, see [Editing Bucket Permissions](#) in the *Amazon Simple Storage Service Console User Guide*.

You are now hosting the rsync scripts for this backup solution, and your backups are no longer dependent on GitHub availability.

## Mounting File Systems Without the EFS Mount Helper

**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 [Using the amazon-efs-utils Tools](#) (p. 35)

Following, you can 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**

Before you can mount a file system, you must create, configure, and launch your related AWS resources. For detailed instructions, see [Getting Started with Amazon Elastic File System \(p. 11\)](#).

**Topics**

- [NFS Support \(p. 291\)](#)
- [Installing the NFS Client \(p. 291\)](#)
- [Recommended NFS Mount Options \(p. 293\)](#)
- [Mounting on Amazon EC2 with a DNS Name \(p. 294\)](#)
- [Mounting with an IP Address \(p. 295\)](#)

## 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.

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 2015.09 or newer
- RHEL 7.3 or newer
- RHEL 6.9 with kernel 2.6.32-696 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 \(p. 173\)](#).

**Note**

Using Amazon EFS with Amazon EC2 instances based on Microsoft Windows is not supported.

## Troubleshooting AMI and Kernel Versions

To troubleshoot issues related to certain AMI or kernel versions when using Amazon EFS from an EC2 instance, see [Troubleshooting AMI and Kernel Issues \(p. 176\)](#).

## Installing the NFS Client

To mount your Amazon 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. Note the following about connecting to the instance:
  - To connect to your instance from a computer running macOS or Linux, specify the .pem file to your Secure Shell (SSH) client with the `-i` option and the path to your private key.
  - To connect to your instance from a computer running Windows, you can use either MindTerm or PuTTY. If you plan to use PuTTY, you need to install it and use the following procedure to convert the .pem file to a .ppk file.

For more information, see the following topics in the *Amazon EC2 User Guide for Linux Instances*:

- [Connecting to Your Linux Instance from Windows Using PuTTY](#)
- [Connecting to Your Linux Instance Using SSH](#)

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](#).

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 command.

```
$ sudo service nfs 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 \(p. 294\)](#)
- [Mounting with an IP Address \(p. 295\)](#)
- [Mounting Your Amazon EFS File System Automatically \(p. 64\)](#)

## Recommended NFS Mount Options

We recommend the following default 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 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.

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 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 [man fstab](#) and [man nfs](#) pages in 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 on Amazon EC2 with a DNS Name

- **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 this 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 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 instance with the following command.

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

- **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
```

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.

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 Options Sets](#) 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 [Viewing DNS Hostnames for Your EC2 Instance](#) 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 [Walkthrough: Create and Mount a File System On-Premises with AWS Direct Connect and VPN](#) (p. 119) and [Walkthrough: Mount a File System from a Different VPC](#) (p. 125).

**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

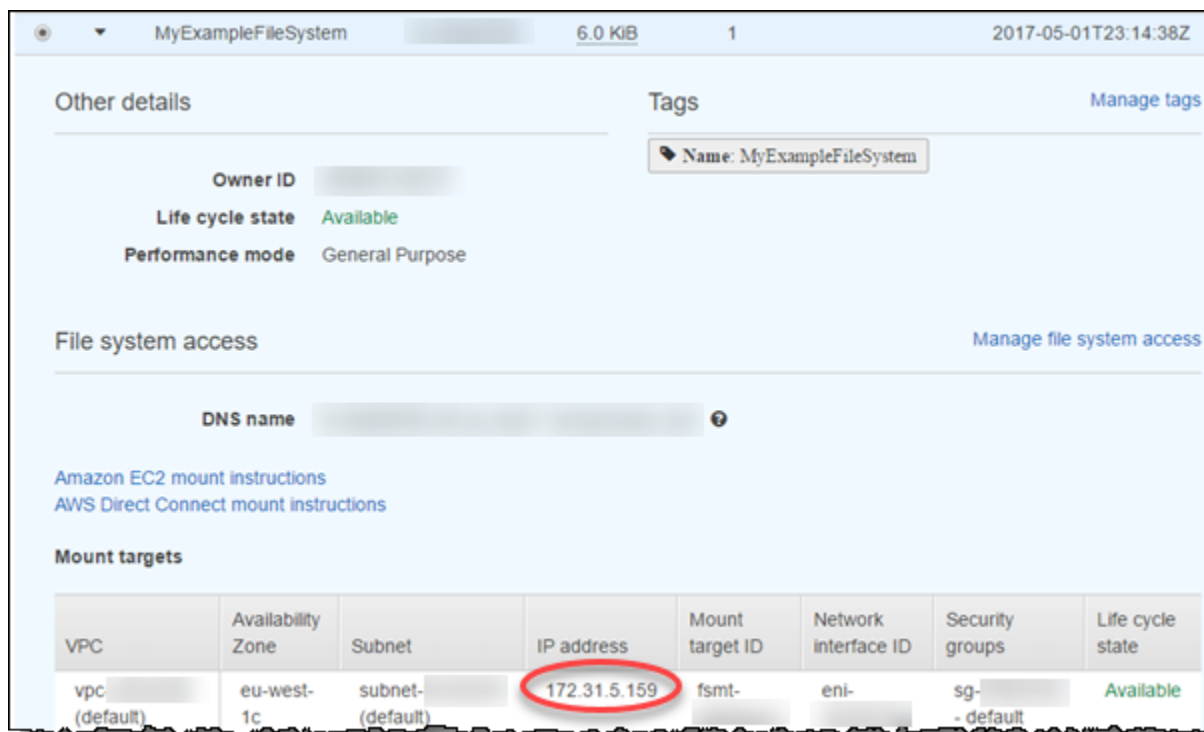
As an alternative to mounting your Amazon EFS file system with the DNS name, Amazon EC2 instances can mount a file system using a mount target's IP address. Mounting by IP address works in environments where DNS is disabled, such as VPCs with DNS hostnames disabled, and EC2-Classic instances mounting using ClassicLink. For more information on ClassicLink, see [ClassicLink](#) in the *Amazon EC2 User Guide for Linux Instances*.

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 get the mount target IP address for your EFS file system through the console using the following procedure.

**To obtain the mount target IP address for your EFS file system**

1. Open the Amazon Elastic File System console at <https://console.aws.amazon.com/efs/>.
2. Choose the **Name** value of your EFS file system for **File systems**.
3. In the **Mount targets** table, identify the **Availability Zone** that you want to use to mount your EFS file system to your EC2 instance.
4. Make a note of the **IP address** associated with your chosen **Availability Zone**.



You can specify the IP address of a mount target in the mount command, as shown following.

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

## Mounting with an IP Address in AWS CloudFormation

You can also mount your file system using an IP address in a 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.



# Document History

- **API version:** 2015-02-01
- **Latest documentation update:** April 1, 2020

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.

update-history-change	update-history-description	update-history-date
<a href="#">Performance of General Purpose Mode file systems increased (p. 297)</a>	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 <a href="#">Quotas for Amazon EFS File Systems</a> .	April 1, 2020
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Beijing and Ningxia AWS Regions.	January 22, 2020
<a href="#">Support added for IAM authorization for NFS clients (p. 297)</a>	You can now use AWS Identity and Access Management (IAM) to manage NFS access to an Amazon EFS file system. For more information, see <a href="#">Using AWS IAM to Control NFS Access to Amazon EFS</a> .	January 13, 2020
<a href="#">Support added for EFS Access Points (p. 297)</a>	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 <a href="#">Working with Amazon EFS Access Points</a> .	January 13, 2020
<a href="#">Support added for AWS Backup partial restore. (p. 297)</a>	You can now restore specific files and directories using a partial restore, in addition to restoring a complete recovery point. For more information, see <a href="#">Using AWS Backup with Amazon EFS</a> .	January 13, 2020
<a href="#">Support added for IAM service-linked roles (p. 297)</a>	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 <a href="#">Using Service-Linked Roles for Amazon EFS</a> .	December 10, 2019

<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Europe (Stockholm) AWS Region.	November 20, 2019
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Asia Pacific (Hong Kong) AWS Region.	November 20, 2019
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the South America (São Paulo) AWS Region.	November 20, 2019
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Middle East (Bahrain) AWS Region.	November 20, 2019
<a href="#">New 7 day Lifecycle management policy added (p. 297)</a>	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 <a href="#">EFS Lifecycle Management</a> .	November 6, 2019
<a href="#">Support added for Interface VPC Endpoints (p. 297)</a>	You can establish a private connection between your virtual private cloud and Amazon EFS to call the EFS API. For more information, see <a href="#">Working with VPC Endpoints</a> .	October 22, 2019
<a href="#">Mount an EFS file system when launching a new EC2 instance. (p. 297)</a>	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 <a href="#">Step 2. Create Your EC2 Resources and Launch Your EC2 Instance</a> .	October 17, 2019
<a href="#">Support for Service Quotas added (p. 297)</a>	You can now view all Amazon EFS limits in the Service Quotas console. For more information, see <a href="#">Amazon EFS Limits</a> .	September 10, 2019
<a href="#">New lifecycle management policies added (p. 297)</a>	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 <a href="#">EFS Lifecycle Management</a> .	July 9, 2019

<a href="#">EFS Lifecycle Management now available on all EFS file systems. (p. 297)</a>	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 <a href="#">EFS Lifecycle Management</a> .	July 9, 2019
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Europe (Paris) AWS Region.	June 12, 2019
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Asia Pacific (Mumbai) AWS Region.	June 5, 2019
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Canada (Central) AWS Region.	May 1, 2019
<a href="#">API Update: Tags are now part of the CreateFileSystem operation payload (p. 297)</a>	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 <a href="#">CreateFileSystem</a> and <a href="#">Creating a File System Using the AWS CLI</a> .	February 19, 2019
<a href="#">New features: EFS Infrequent Access storage class and EFS lifecycle management (p. 297)</a>	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 <a href="#">EFS Storage Classes</a> .	February 13, 2019
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Europe (London) AWS Region.	January 23, 2019
<a href="#">AWS Backup Service integration with Amazon EFS (p. 297)</a>	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 <a href="#">AWS Backup and EFS</a> .	January 16, 2019

<a href="#">Transit Gateway connection support to on-premises storage systems added. (p. 297)</a>	Amazon EFS file systems are now accessible using Transit Gateway connections to on-premises storage systems. For more information, see <a href="#">Mounting from Another Account or VPC</a> and <a href="#">Walkthrough: Mount a File System from a Different VPC</a> .	December 6, 2018
<a href="#">EFS File Sync is now part of the new AWS DataSync service. (p. 297)</a>	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 <a href="#">Transfer Files from On-Premises File Systems to Amazon EFS Using AWS DataSync</a> .	November 26, 2018
<a href="#">VPN and inter-region VPC peering connection support added (p. 297)</a>	Amazon EFS are now accessible over VPN connections and inter-region VPC peering connections. For more information, see <a href="#">Transfer Files from On-Premises File Systems to Amazon EFS Using AWS DataSync</a> .	October 23, 2018
<a href="#">VPN and Inter-region VPC Peering connection support added (p. 297)</a>	Amazon EFS file systems are now accessible over VPN connections and inter-region VPC peering connections. For more information, see <a href="#">Mounting from Another Account or VPC</a> and <a href="#">How Amazon EFS Works with Direct Connect and VPNs</a> .	October 23, 2018
<a href="#">Additional AWS Region support added (p. 297)</a>	Amazon EFS is now available to all users in the Asia Pacific (Singapore) AWS Region.	July 13, 2018
<a href="#">Introducing Provisioned Throughput mode (p. 297)</a>	You can now provision throughput for new or existing file systems with the new Provisioned Throughput mode. For more information, see <a href="#">Throughput Modes</a> .	July 12, 2018
<a href="#">Additional AWS Region support added (p. 297)</a>	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. For more information, see <a href="#">Using Metric Math with Amazon EFS (p. 79)</a> .	April 4, 2018
Added the amazon-efs-utils set of open-source tools, and added encryption in transit	<p>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 <a href="#">Using the amazon-efs-utils Tools (p. 35)</a>.</p> <p>Also in this release, Amazon EFS now supports encryption in transit through Transport Layer Security (TLS) tunneling. For more information, see <a href="#">Data Encryption in EFS (p. 135)</a>.</p>	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 <a href="#">Resource Quotas (p. 168)</a> .	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 <a href="#">Data Encryption in EFS (p. 135)</a> .	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 automatically resolves to a mount target's IP address in the Availability Zone for the connecting Amazon EC2 instance. For more information, see <a href="#">Mounting on Amazon EC2 with a DNS Name (p. 294)</a> .	December 20, 2016
Increased tag support for file systems	Amazon EFS now supports 50 tags per file system. For more information on tags in Amazon EFS, see <a href="#">Managing File System Tags (p. 48)</a> .	August 29, 2016
General availability	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

Change	Description	Date Changed
New guide	This is the first release of the <i>Amazon Elastic File System User Guide</i> .	May 26, 2015