File Cache User Guide

Amazon File Cache

Amazon File Cache: File Cache User Guide

Table of Contents

What is Amazon File Cache?

Amazon File Cache is a fully managed, high-speed cache on AWS that's used to process file data, regardless of where the data is stored. Amazon File Cache serves as a temporary, high-performance storage location for data that's stored in on-premises file systems, AWS file systems, and Amazon Simple Storage Service (Amazon S3) buckets. You can use this capability to make dispersed datasets available to file-based applications on AWS with a unified view, and at high speeds—sub-millisecond latencies and high throughput.

Amazon File Cache presents data from linked datasets as a unified set of files and directories. It serves data in the cache at consistent high speeds with sub-millisecond latency to applications running on AWS—up to hundreds of GBps of throughput, and up to millions of operations per second, speeding up workload completion times and optimizing compute resource consumption costs. Amazon File Cache automatically loads data into the cache when it’s accessed for the first time and releases data when it’s not used.

With a few clicks in the AWS console, CLI, or API, you can create a high-performance cache. With Amazon File Cache, you don't have to worry about managing file servers and storage volumes, updating hardware, configuring software, running out of capacity, or tuning performance—Amazon File Cache automates these time-consuming administration tasks.

Amazon File Cache is POSIX-compliant, so you can use your current Linux-based applications without having to make any changes. Amazon File Cache provides a native file system interface and works as any file system does with your Linux operating system. It also provides read-after-write consistency and supports file locking.

Topics

- Are you a first-time user of Amazon File Cache?

Amazon File Cache availability

Amazon File Cache is available in the following AWS Regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Stockholm)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Tokyo)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)

Amazon File Cache and data repositories

You can link your cache to data repositories on Amazon S3, or on file systems that support the NFSv3 protocol. The NFS data repository can be on-premises or in the AWS Cloud. You can link a maximum of 8 data repositories, but they must all be of the same repository type (either all Amazon S3 or all NFS). For more information about linking your cache to a data repository, see Linking your cache to a data repository.

When linked to a data repository, a cache transparently presents Amazon S3 or NFS objects as files and directories. By default, Amazon File Cache automatically loads data into the cache when it's accessed for the first time. You can optionally pre-load data into the cache before starting your workload. For more information about importing data repository files and directories, see Importing files from your data repository.

When the files in your cache are changed (either by users or by your workloads), you can write the cache data back to the data repository. You can use HSM commands to transfer the data and metadata between your cache and its linked data repositories. For more information, see Exporting changes to the data repository.

Deployment and storage type

Amazon File Cache supports the CACHE_1 deployment type. When you create a new cache on the AWS Management Console, this deployment type is automatically preset for your cache. For caches using the CACHE_1 deployment type, data is automatically replicated within the same Availability Zone in which the cache is located, and file servers are replaced if they fail.

Amazon File Cache is built on solid state drive (SSD) storage. SSD storage is suited for low-latency, IOPS-intensive workloads that typically have small, random file operations. For more information about cache performance, see Amazon File Cache performance.

Accessing Amazon File Cache

You can mix and match compute instance types and Linux Amazon Machine Images (AMIs) that are connected to a single cache.

Amazon File Cache is accessible from compute workloads running on Amazon Elastic Compute Cloud (Amazon EC2) instances, on Amazon Elastic Container Service (Amazon ECS) Docker containers, and on containers running on Amazon Elastic Kubernetes Service (Amazon EKS).

- **Amazon EC2** – You can access your cache from your Amazon EC2 compute instances using the open-source Lustre client. Amazon EC2 instances can access your cache from other Availability Zones within the same Amazon Virtual Private Cloud (Amazon VPC), provided that your networking configuration allows access across subnets within the VPC. After your cache is mounted, you can work with its files and directories as you do when using a local file system.
- **Amazon ECS** – You can access Amazon File Cache from Amazon ECS Docker containers on Amazon EC2 instances. For more information, see Mounting from Amazon Elastic Container Service.
- **Amazon EKS** – You access Amazon File Cache from containers running on Amazon EKS using the open-source Amazon File Cache CSI driver, as described in Amazon EKS User Guide. Your containers running on Amazon EKS can use high-performance persistent volumes (PVs) backed by Amazon File Cache.

Amazon File Cache is compatible with the most popular Linux-based AMIs, including Amazon Linux 2 and Amazon Linux, Red Hat Enterprise Linux (RHEL), CentOS, Rocky Linux, and Ubuntu. The Lustre client is included with Amazon Linux 2 and Amazon Linux. For RHEL, CentOS, Rocky Linux, and Ubuntu, an AWS Lustre client repository provides clients that are compatible with these operating systems.

For more information about the clients, compute instances, and environments from which you can access your cache, see Accessing caches.

Integrations with AWS services

Amazon File Cache integrates with AWS Batch using Amazon EC2 Launch Templates. You can use AWS Batch to run batch computing workloads on the AWS Cloud, including high performance computing (HPC), machine learning (ML), and other asynchronous workloads. AWS Batch automatically and dynamically sizes instances based on job resource requirements. For more information, see What Is AWS Batch? in the *AWS Batch User Guide*.

Amazon File Cache integrates with AWS Thinkbox Deadline. Deadline is an administration and compute management toolkit for Windows, Linux, and macOS based render farms. For more information about Deadline, see the Deadline User Guide.

Security and compliance

Amazon File Cache supports encryption at rest and in transit. Amazon File Cache automatically encrypts cache data at rest using keys managed in the AWS Key Management Service (AWS KMS). Data in transit is also automatically encrypted on caches when accessed from supported Amazon EC2 instances. For more information about data encryption in Amazon File Cache, see Data encryption in Amazon File Cache. For more information about security, see Security in Amazon File Cache.

Assumptions

In this guide, we make the following assumptions:

- If you use Amazon Elastic Compute Cloud (Amazon EC2), we assume that you're familiar with that service. For more information about how to use Amazon EC2, see the Amazon EC2 documentation.

- We assume that you're familiar with using Amazon Virtual Private Cloud (Amazon VPC). For more information about how to use Amazon VPC, see the Amazon VPC User Guide.
- We assume that you haven't changed the rules on the default security group for your VPC based on the Amazon VPC service. If you have, make sure that you add the necessary rules to allow network traffic from your Amazon EC2 instance to your cache. For more details, see Cache access control with Amazon VPC.

Pricing for Amazon File Cache

With Amazon File Cache, there are no up front hardware or software costs. You pay only for the resources used, with no minimum commitments, setup costs, or additional fees. For information about the pricing and fees associated with the service, see Amazon File Cache Pricing.

Are you a first-time user of Amazon File Cache?

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

1. If you're ready to create your first cache, try Getting started with Amazon File Cache.
2. For information about performance, see Amazon File Cache performance.
3. For information about linking your cache to an Amazon S3 bucket or NFS data repository, see Using data repositories with Amazon File Cache.
4. For Amazon File Cache security details, see Security in Amazon File Cache.
5. For information about the scalability limits of Amazon File Cache, see Quotas.
6. For information about the Amazon File Cache API, see the Amazon File Cache API Reference.

Setting up

Before you use Amazon File Cache for the first time, complete the tasks in the Sign up for Amazon Web Services section. To complete the Getting started tutorial, make sure the Amazon S3 bucket that you'll link to your cache has the permissions listed in Adding permissions to use data repositories in Amazon S3.

Topics

Sign up for Amazon Web Services

To set up for AWS, complete the following tasks:

1. Sign up for an AWS account
2. Create a user with administrative access

Sign up for an AWS account

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

To sign up for an AWS account

1. Open https://portal.aws.eu/billing/signup.
2. Follow the online instructions.

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

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

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Adding permissions to use data repositories in Amazon S3

Amazon File Cache is deeply integrated with Amazon Simple Storage Service (Amazon S3). This integration means that applications that access your cache can also seamlessly access the objects stored in your linked Amazon S3 bucket. For more information, see Using data repositories with Amazon File Cache.

To use data repositories, you must first allow Amazon File Cache certain IAM permissions in a role associated with the account for your administrator user.

To embed an inline policy for a role using the console

1. Sign in to the AWS Management Console and open the IAM console at https://eusc-de-east-1.console.amazonaws-eusc.eu/iam/.
2. In the navigation pane, choose **Roles**.
3. In the list, choose the name of the role to embed a policy in.
4. Choose the **Permissions** tab.
5. Scroll to the bottom of the page and choose **Add inline policy**.

> **Note**
>
> You can't embed an inline policy in a service-linked role in IAM. Because the linked service defines whether you can modify the permissions of the role, you might be able to add additional policies from the service console, API, or AWS Command Line

Interface (AWS CLI). To view the service-linked role documentation for a service, see **AWS Services That Work with IAM** and choose **Yes** in the **Service-Linked Role** column for your service.

6. Choose **Creating Policies with the Visual Editor**.
7. Add the following permissions policy statement.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceLinkedRole",
            "iam:AttachRolePolicy",
            "iam:PutRolePolicy"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/s3.data-
source.lustre.fsx.amazonaws.com/*"
    }
}
```

After you create an inline policy, it's automatically embedded in your role. For more information about service-linked roles, see Using service-linked roles for Amazon FSx.

How Amazon File Cache checks for access to linked S3 buckets

If the IAM role that you used to create the Amazon File Cache resource doesn't have the `iam:AttachRolePolicy` and `iam:PutRolePolicy` permissions, Amazon File Cache checks whether it can update your S3 bucket policy. Amazon File Cache can update your bucket policy if the `s3:PutBucketPolicy` permission is included in your IAM role to allow the Amazon File Cache resource to import or export data to your S3 bucket. If allowed to modify the bucket policy, Amazon File Cache adds the following permissions to the bucket policy:

- `s3:AbortMultipartUpload`
- `s3:DeleteObject`

- `s3:PutObject`
- `s3:Get*`
- `s3:List*`
- `s3:PutBucketNotification`
- `s3:PutBucketPolicy`
- `s3:DeleteBucketPolicy`

If Amazon File Cache can't modify the bucket policy, it then checks if the existing bucket policy grants Amazon File Cache access to the bucket.

If all of these options fail, then the request to create the DRA to the S3 bucket fails.

Next step

Getting started with Amazon File Cache

Getting started with Amazon File Cache

Learn how to start using Amazon File Cache. These steps walk you through creating an Amazon File Cache resource and accessing it from your compute instances. Amazon File Cache can link to an Amazon Simple Storage Service (Amazon S3) or Network File System (NFS) data repository (but not to both types at the same time). This exercise uses an Amazon S3 bucket as the data repository, and shows how to use your cache to process the data in your Amazon S3 bucket with your file-based applications.

This getting started exercise includes the following steps.

Topics

Prerequisites

To perform this getting started exercise, you'll need the following:

- An AWS account with the permissions necessary to create an Amazon File Cache and an Amazon Elastic Compute Cloud (Amazon EC2) instance. For more information, see Setting up.
- Each cache requires four IP addresses for the metadata servers (MDS) and one IP address for each storage server (OSS). Caches are provisioned with 2.4 TiB of storage per OSS.
- An Amazon EC2 instance running a supported Linux release in your virtual private cloud (VPC) based on the Amazon VPC service. You'll install the Lustre client on this Amazon EC2 instance, and then mount your cache on the Amazon EC2 instance. The Lustre client supports Amazon Linux, Amazon Linux 2, CentOS and Red Hat Enterprise Linux 7.9 and 8.4 through 8.6, Rocky Linux 8.4 through 8.6, and Ubuntu 18.04, 20.04, and 22.04. For this getting started exercise, we'll use Ubuntu 22.04.

  When creating your Amazon EC2 instance for this getting started exercise, keep the following in mind:

- We recommend that you create your instance in your default VPC.
- We recommend that you use the default security group when creating your Amazon EC2 instance.

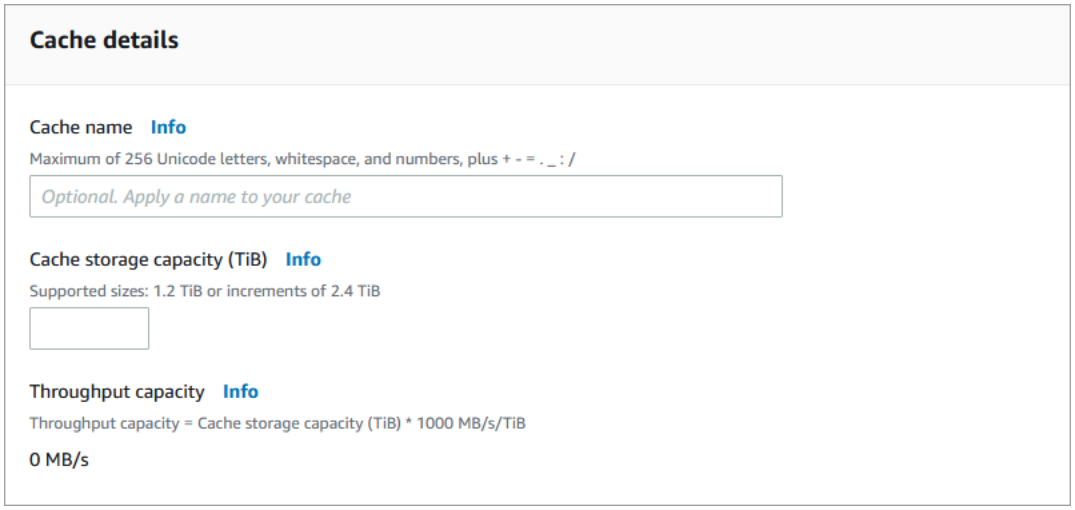
- An Amazon S3 bucket storing the data for your workload to process. The Amazon S3 bucket will be the linked data repository for your cache.

Step 1: Create your cache

Next, you create your cache. For this exercise, there are instructions about creating a data repository association to link to an Amazon S3 bucket when you create the File Cache.

1. Open the AWS Management Console at https://eusc-de-east-1.console.amazonaws-eusc.eu/fsx/.
2. Choose **Caches** in the navigation pane.
3. On the dashboard, choose **Create cache** to start the cache creation wizard.

   Begin your configuration with the **Cache details** section.



4. For **Cache name**, enter a name for your cache. We recommend using a name that helps you to identify and manage the cache in the future. You can use a maximum of 256 Unicode letters, white spaces, numbers, and these special characters: + - = . _ : /
5. For **Cache storage capacity**, set the amount of storage capacity for your cache, in TiB. Set this to a value of 1.2 TiB, 2.4 TiB, or increments of 2.4 TiB.

   Additionally, metadata storage capacity of 2.4 TiB is provisioned for all caches.

6. The amount of **Throughput capacity** is calculated by multiplying the cache storage capacity by the throughput tier. For example, for a 1.2 TiB cache, it's `1200` MBps; for a 9.6 TiB cache, it's `9600` MBps.

   Throughput capacity is the sustained speed at which the file server that hosts your cache can serve data.

7. In the **Network & security** section, provide networking and security group information:

   - For **Virtual Private Cloud (VPC)**, choose the Amazon VPC that you want to associate with your cache.
   - For **VPC Security Groups**, the ID for the default security group for your VPC should already be added.
   - For **Subnet**, choose any value from the list of available subnets.

8. In the **Encryption** section, for **Encryption key**, choose the AWS Key Management Service (AWS KMS) encryption key that protects your cache's data at rest.

9. For **Tags - *optional***, you can enter a key and value to add tags to your cache. A tag is a case-sensitive key-value pair that helps you to manage, filter, and search for your cache.

10. Choose **Next**.

11. In the **Data repository associations (DRAs)** section, there are no DRAs linking your cache to Amazon S3 or NFS data repositories. For detailed information about linking data repositories to Amazon File Cache, see To link an S3 bucket or NFS file system while creating a cache (console).

    The following instructions describe how to link your cache to an existing Amazon S3 bucket for this getting started exercise. In the **Data repository association** dialog box, provide information for the following fields.

    a. For **Repository type**, choose `S3`.

    b. For **Data repository path**, enter the path of the existing S3 bucket or prefix to associate with your cache (for example, `s3://my-bucket/my-prefix/`).

    c. For **Cache path**, enter the name of a high-level directory (such as `/ns1`) or subdirectory (such as `/ns1/subdir`) within Amazon File Cache to associate with the S3 data repository. The first forward slash in the path is required.

> **Note**
>
> **Cache path** can only be set to root (/) on NFS DRAs when **Subdirectories** is specified. If you specify root (/) as the **Cache path**, you can create only one DRA on the cache.
> **Cache path** can't be set to root (/) for an S3 DRA.

d. Choose **Add**.

12. Choose **Next**.
13. Review the cache configuration shown on the **Cache summary** page. For your reference, note which cache settings you can modify after the cache is created.
14. Choose **Create cache**.

Now that you've created your cache, note its fully qualified domain name and mount name for a later step. You can find the fully qualified domain name and mount name for a cache by choosing the name of the cache in the **Caches** dashboard, and then choosing **Attach**.

Step 2: Install and configure the Lustre client on your instance before mounting your cache

To mount your cache from your Amazon EC2 instance, first install the Lustre 2.12 client.

You can get Lustre packages from the Ubuntu 22.04 AWS Lustre client repository. To validate that the contents of the repository weren't tampered with before or during download, a GNU Privacy Guard (GPG) signature is applied to the metadata of the repository. Installing the repository fails unless you have the correct public GPG key installed on your system.

To download the Lustre client onto your Amazon EC2 instance

1. Open a terminal on your client.
2. Follow these steps to add the Lustre client Ubuntu repository:

   a. If you have not previously registered an AWS Lustre client Ubuntu repository on your client instance, download and install the required public key. Use the following command.

```
wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-
ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-
ubuntu-public-key.gpg >/dev/null
```

b. Add the AWS Lustre package repository to your local package manager using the following command.

```
sudo bash -c 'echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-
key.gpg] https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /
etc/apt/sources.list.d/fsxlustreclientrepo.list && apt-get update'
```

3. Determine which kernel is currently running on your client instance, and update as needed. The AWS Lustre client on Ubuntu 22.04 requires kernel `5.15.0.1020-aws` or later for both x86-based EC2 instances and Arm-based EC2 instances powered by AWS Graviton processors.

   a. Run the following command to determine which kernel is running.

   ```
   uname -r
   ```

   b. Run the following command to update to the latest Ubuntu kernel and Lustre version and then reboot.

   ```
   sudo apt install -y linux-aws lustre-client-modules-aws && sudo reboot
   ```

   If your kernel version is greater than `5.15.0.1020-aws` for both x86-based EC2 instances and Graviton-based instances, and you don't want to update to the latest kernel version, you can install the Lustre client for the current kernel with the following command.

   ```
   sudo apt install -y lustre-client-modules-$(uname -r)
   ```

   The two Lustre packages that are necessary for mounting and interacting with your cache are installed. You can optionally install additional related packages such as a package containing the source code and packages containing tests that are included in the repository.

   c. List all available packages in the repository by using the following command.

```
sudo apt-cache search ^lustre
```

d. (Optional) If you want your system upgrade to also always upgrade Lustre client modules, verify that the `lustre-client-modules-aws` package is installed using the following command.

```
sudo apt install -y lustre-client-modules-aws
```

For information about installing the Lustre client on other Linux distributions, see Installing the Lustre client.

To mount your cache

1. Make a directory for the mount point with the following command.

```
sudo mkdir -p /mnt
```

2. Mount the Amazon File Cache to the directory that you created. Use the following command and replace these items:

   - Replace *cache_dns_name* with the actual file cache's Domain Name System (DNS) name.
   - Replace *mountname* with the cache's mount name, which you can get by running the **describe-file-caches** AWS CLI command or the DescribeFileCaches API operation.

```
sudo mount -t lustre -o relatime,flock cache_dns_name@tcp:/mountname /mnt
```

   This command mounts your cache with these options:

   - `relatime` – Maintains `atime` (inode access times) data, but not for each time that a file is accessed. With this option enabled, `atime` data is written to disk only if the file was modified after the `atime` data was last updated (mtime), or if the file was last accessed more than a certain amount of time ago (one day by default). `relatime` is required for automatic cache eviction to work properly.
   - `flock` – Enables file locking for your cache. If you don't want file locking enabled, use the `mount` command without `flock`.

3. Verify that the mount command was successful by listing the contents of the directory to which you mounted the cache `/mnt`, by using the following command.

```
ls /mnt
import-path  lustre
$
```

You can also use the `df` command.

```
df
Filesystem                      1K-blocks    Used  Available Use% Mounted on
devtmpf                           1001808       0    1001808   0% /dev
tmpfs                             1019760       0    1019760   0% /dev/shm
tmpfs                             1019760     392    1019368   1% /run
tmpfs                             1019760       0    1019760   0% /sys/fs/cgroup
/dev/xvda1                        8376300 1263180    7113120  16% /
123.456.789.0@tcp:/mountname  3547698816   13824 3547678848   1% /mnt
tmpfs                              203956       0     203956   0% /run/user/1000
```

The results show the Amazon File Cache resource mounted on /mnt.

Step 3: Run your analysis

Now that your cache is created and mounted to a compute instance, you can use it to run your high-performance compute workload. The workload loads data from the Amazon S3 data repository as files are accessed by your workload.

After you run the workload, you can export the data that you write to your cache back to your Amazon S3 bucket at any time. From a terminal on one of your compute instances, run the following command to export a file to your Amazon S3 bucket.

```
sudo lfs hsm_archive file_name
```

For more information about how to run this command on a folder or large collection of files quickly, see Exporting files using HSM commands.

Step 4: Clean up resources

After you finish this exercise, we recommend that you follow these steps to clean up your resources and protect your AWS account.

To clean up resources

1. On the Amazon EC2 console, terminate your instance. For more information, see Terminate your instance in the *Amazon EC2 User Guide.*
2. On the AWS Management Console, delete your cache with the following procedure:

   a. In the navigation pane, choose **Caches**.

   b. Choose the cache that you want to delete from list of caches on the dashboard.

   c. For **Actions**, choose **Delete cache**.

   d. In the dialog box that appears, provide the cache ID to confirm the deletion. Choose **Delete cache**.
3. If you created an Amazon S3 bucket for this exercise, and don't want to preserve the data that you exported, you can now delete it. For more information, see Deleting a bucket in the *Amazon Simple Storage Service User Guide.*

Using data repositories with Amazon File Cache

Amazon File Cache is a fully managed, high-speed cache on AWS that makes it easier to process file data, regardless of where the data is stored. Your cache serves as a temporary, high-performance storage location for data stored in on-premises file systems, AWS file systems, and Amazon Simple Storage Service (Amazon S3) buckets. Using this capability, you can make dispersed data sets available to file-based applications on AWS with a unified view and at high speeds—sub-millisecond latencies and high throughput.

Topics

Overview of data repositories

Amazon File Cache is well integrated with data repositories in Amazon S3 or on Network File System (NFS) file systems that support the NFSv3 protocol. This integration means that you can seamlessly access the objects stored in your Amazon S3 buckets or NFS data repositories from applications that mount your cache. You can also run your compute-intensive workloads on Amazon EC2 instances in the AWS Cloud and export the results to your data repository after your workload is complete.

> **Note**
>
> You can link your cache to either S3 or NFS data repositories, but not to both types at the same time. You can't have a mix of linked S3 and NFS data repositories on a single cache.

When you use Amazon File Cache with multiple storage repositories, you can ingest and process large volumes of file data in a high-performance cache. At the same time, you can write results to your data repositories by using HSM commands. With these features, you can restart your workload at any time using the latest data stored in your data repository.

By default, Amazon File Cache automatically loads data into the cache when it's accessed for the first time (lazy load). You can optionally pre-load data into the cache before starting your workload. For more information, see Lazy load.

You can also export files and their associated metadata (including POSIX metadata) in your cache to your data repository using HSM commands. When you use HSM commands, file data and metadata that were created or modified since the last such export are exported to the data repository. For more information, see Exporting files using HSM commands.

> **Important**
>
> If you have linked one or more caches to a data repository on Amazon S3, don't delete the Amazon S3 bucket until you have deleted all linked caches.

POSIX metadata support for data repositories

Amazon File Cache automatically transfers Portable Operating System Interface (POSIX) metadata for files, directories, and symbolic links (symlinks) when importing and exporting data to and from a linked Amazon S3 or NFS data repository. When you export changes in your cache to a linked data repository, Amazon File Cache also exports POSIX metadata changes along with data changes. Because of this metadata export, you can implement and maintain access controls between your cache and its linked data repositories.

Amazon File Cache imports only objects that have POSIX-compliant object keys, such as the following.

```
test/mydir/
test/
```

Amazon File Cache stores directories and symlinks as separate objects in the linked data repository. On an S3 data repository for example, for directories, Amazon File Cache creates an S3 object with a key name that ends with a slash ("/"), as follows:

- The S3 object key `test/mydir/` maps to the cache directory `test/mydir`.
- The S3 object key `test/` maps to the cache directory `test`.

For symlinks, Amazon File Cache uses the following Amazon S3 schema for symlinks:

- **S3 object key** – The path to the link, relative to the Amazon File Cache mount directory
- **S3 object data** – The target path of the symlink
- **S3 object metadata** – The metadata for the symlink

Amazon File Cache stores POSIX metadata, including ownership, permissions, and timestamps for Amazon File Cache files, directories, and symbolic links, in S3 objects as follows:

- `Content-Type` – The HTTP entity header used to indicate the media type of the resource for web browsers.
- `x-amz-meta-file-permissions` – The file type and permissions in the format `<octal file type><octal permission mask>`, consistent with `st_mode` in the Linux stat(2) man page.

  > **Note**
  >
  > Amazon File Cache doesn't import or retain `setuid` information.

- `x-amz-meta-file-owner` – The owner user ID (UID) expressed as an integer.
- `x-amz-meta-file-group` – The group ID (GID) expressed as an integer.
- `x-amz-meta-file-atime` – The last-accessed time in nanoseconds. Terminate the time value with `ns`; otherwise Amazon File Cache interprets the value as milliseconds.
- `x-amz-meta-file-mtime` – The last-modified time in nanoseconds. Terminate the time value with `ns`; otherwise, Amazon File Cache interprets the value as milliseconds.
- `x-amz-meta-user-agent` – The user agent, ignored during Amazon File Cache import. During export, Amazon File Cache sets this value to `aws-fsx-lustre`.

The default POSIX permission that Amazon File Cache assigns to a file is 755. This permission allows read and execute access for all users and write access for the owner of the file.

> **Note**
>
> Amazon File Cache doesn't retain any user-defined custom metadata on S3 objects.

Walkthrough: Attaching POSIX permissions when uploading objects into an Amazon S3 bucket

The following procedure walks you through the process of uploading objects into Amazon S3 with POSIX permissions. By doing so, you can import the POSIX permissions when you create an Amazon File Cache that is linked to that S3 bucket.

To upload objects with POSIX permissions to Amazon S3

1. From your local computer or machine, use the following example commands to create a test directory (`s3cptestdir`) and file (`s3cptest.txt`) that will be uploaded to the S3 bucket.

   ```
   $ mkdir s3cptestdir
   $ echo "S3cp metadata import test" >> s3cptestdir/s3cptest.txt
   $ ls -ld s3cptestdir/ s3cptestdir/s3cptest.txt
   drwxr-xr-x 3 500 500 96 Jan 8 11:29 s3cptestdir/
   -rw-r--r-- 1 500 500 26 Jan 8 11:29 s3cptestdir/s3cptest.txt
   ```

   The newly created file and directory have a file owner user ID (UID) and group ID (GID) of 500, and permissions as shown in the preceding example.

2. Call the Amazon S3 API to create the directory `s3cptestdir` with metadata permissions. You must specify the directory name with a trailing slash (/). For information about supported POSIX metadata, see POSIX metadata support for data repositories.

   Replace *bucket_name* with the actual name of your S3 bucket.

   ```
   $ aws s3api put-object --bucket bucket_name --key s3cptestdir/ --metadata '{"user-
   agent":"aws-fsx-lustre" , \
         "file-atime":"1595002920000000000ns" , "file-owner":"500" , "file-
   permissions":"0100664","file-group":"500" , \
         "file-mtime":"1595002920000000000ns"}'
   ```

3. Verify that the POSIX permissions are tagged to S3 object metadata.

   ```
   $ aws s3api head-object --bucket bucket_name --key s3cptestdir/
   {
       "AcceptRanges": "bytes",
       "LastModified": "Fri, 08 Jan 2021 17:32:27 GMT",
       "ContentLength": 0,
       "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\"",
   ```

```
    "VersionId": "bAlhCoWq7aIEjc3R6Myc6UOb8sHHtJkR",
    "ContentType": "binary/octet-stream",
    "Metadata": {
        "user-agent": "aws-fsx-lustre",
        "file-atime": "1595002920000000000ns",
        "file-owner": "500",
        "file-permissions": "0100664",
        "file-group": "500",
        "file-mtime": "1595002920000000000ns"
    }
}
```

4. Upload the test file (created in step 1) from your computer to the S3 bucket with metadata permissions.

```
$ aws s3 cp s3cptestdir/s3cptest.txt s3://bucket_name/s3cptestdir/s3cptest.txt \
      --metadata '{"user-agent":"aws-fsx-lustre" , "file-
atime":"1595002920000000000ns" , \
      "file-owner":"500" , "file-permissions":"0100664","file-group":"500" , "file-
mtime":"1595002920000000000ns"}'
```

5. Verify that the POSIX permissions are tagged to S3 object metadata.

```
$ aws s3api head-object --bucket bucket_name --key s3cptestdir/s3cptest.txt
{
    "AcceptRanges": "bytes",
    "LastModified": "Fri, 08 Jan 2021 17:33:35 GMT",
    "ContentLength": 26,
    "ETag": "\"eb33f7e1f44a14a8e2f9475ae3fc45d3\"",
    "VersionId": "w9ztRoEhB832m8NC3a_JTlTyIx7Uzql6",
    "ContentType": "text/plain",
    "Metadata": {
        "user-agent": "aws-fsx-lustre",
        "file-atime": "1595002920000000000ns",
        "file-owner": "500",
        "file-permissions": "0100664",
        "file-group": "500",
        "file-mtime": "1595002920000000000ns"
    }
}
```

6. Verify permissions on the Amazon File Cache linked to the S3 bucket.

```
$ sudo lfs df -h /fsx
UUID                       bytes        Used   Available Use% Mounted on
3rnxfbmv-MDT0000_UUID       34.4G        6.1M       34.4G   0% /fsx[MDT:0]
3rnxfbmv-OST0000_UUID        1.1T        4.5M        1.1T   0% /fsx[OST:0]

filesystem_summary:          1.1T        4.5M        1.1T   0% /fsx

$ cd /fsx/s3cptestdir/
$ ls -ld s3cptestdir/
drw-rw-r-- 2 500 500 25600 Jan  8 17:33 s3cptestdir/

$ ls -ld s3cptestdir/s3cptest.txt
-rw-rw-r-- 1 500 500 26 Jan 8 17:33 s3cptestdir/s3cptest.txt
```

Both the `s3cptestdir` directory and the `s3cptest.txt` file have POSIX permissions imported.

Prerequisites for linking to on-premises NFS data repositories

Before you can link your cache to an on-premises NFS data store, verify that your resources and configurations meet the following requirements:

- Your on-premises NFS file system must support NFSv3.
- If you're using a domain name to link your NFS file system to Amazon File Cache, you must provide the IP address of a DNS server that Amazon File Cache can use to resolve the domain name of the on-premises NFSv3 file system. The DNS server can be located in the VPC where you plan to create the cache, or it can be on your on-premises network accessible from your VPC.
- The DNS server and on premises NFSv3 file system must use private IP addresses, as specified in RFC 1918:
  - 10.0.0.0-10.255.255.255 (10/8 prefix)
  - 172.16.0.0-172.31.255.255 (172.16/12 prefix)
  - 192.168.0.0-192.168.255.255 (192.168/16 prefix)
- You must establish an Direct Connect or VPN connection between your on-premises network and the Amazon VPC where your Amazon File Cache is located. For more information about Direct Connect, see the *Direct Connect User Guide*. For more information about setting up a VPC connection, see the *Amazon VPC User Guide*.

> **Important**
>
> Use an Site-to-Site VPN connection if you want to encrypt data as it transits between your Amazon VPC and your on-premises network. For more information, see What is AWS Site-to-Site VPN?

- Your on-premises firewall must allow traffic between IP addresses in your Amazon VPC subnet IP CIDR and the IP addresses of the DNS server and the on-premises NFSv3 file system. The following ports must be open for the daemons involved in sharing data via NFS:

Port	Protocol	Description
111	TCP/UDP	Port for the **portmapper** daemon. The port number is fixed.
2049	TCP/UDP	Port for the **nfsd** daemon. The port number is fixed.
635	TCP/UDP	Port for the **mountd** daemon. The port assignment is dynamic and could be another port number. You must verify the actual port and make sure it's open.
4045	TCP/UDP	Port for the **nlockmgr** daemon. The port assignment is dynamic and could be another port number. You must verify the actual port and make sure it's open.
4046	TCP/UDP	Port for the **status** daemon. The port assignment is dynamic and could be another port number. You

Port	Protocol	Description
		must verify the actual port and make sure it's open.

You can use the following command to look up dynamic ports for your on-premises NFS servers:

```
rpcinfo -p localhost
```

- Your on-premises NFSv3 file system is configured to allow access to IP addresses on the Amazon VPC where the cache is located.
- The Amazon VPC Security Group used for your cache must be configured to allow outbound traffic to the IP addresses of the DNS server and on-premises NFSv3 file system. Make sure to add outbound rules to allow port 53 for both UDP and TCP for DNS traffic, and to allow the TCP ports used by the on-premises NFSv3 file system for NFS. For more information, see Controlling access using inbound and outbound rules.
- While Amazon File Cache supports NFSv3 file systems with most NFSv3 export policies, you must not use the NFS export option `all_squash`. This configuration is required so that Amazon File Cache has the necessary permissions to read and write files owned by all users on your NFSv3 file system.

Linking your cache to a data repository

You can link your Amazon File Cache to data repositories in Amazon S3 or on NFS (Network File System) file systems that support the NFSv3 protocol. The NFS file systems can be on-premises or in-cloud file systems. You create the links when you create your cache.

A link between a directory on your cache and an Amazon S3 or NFS data repository is called a *data repository association (DRA)*. You can create a maximum of 8 data repository associations on a Amazon File Cache resource. Each DRA must have a unique Amazon File Cache directory and an S3 bucket or NFS file system associated with it.

> **Note**
>
> An Amazon File Cache resource can link to either S3 or NFS data repositories, but not to both types at the same time. All the DRAs on the cache must link to the same data repository type (S3 or NFS).

By default, Amazon File Cache automatically loads data into the cache when it's accessed for the first time (lazy load). You can optionally pre-load data into the cache before starting your workload.

> **Note**
>
> You shouldn't modify the same file on both the data repository and the cache at the same time, otherwise the behavior is undefined.

Creating a link to a data repository

The following procedure walks you through the process of creating a data repository association (DRA) while creating an Amazon File Cache resource, using the AWS Management Console. The DRA links the cache to an existing Amazon S3 bucket or NFS file system.

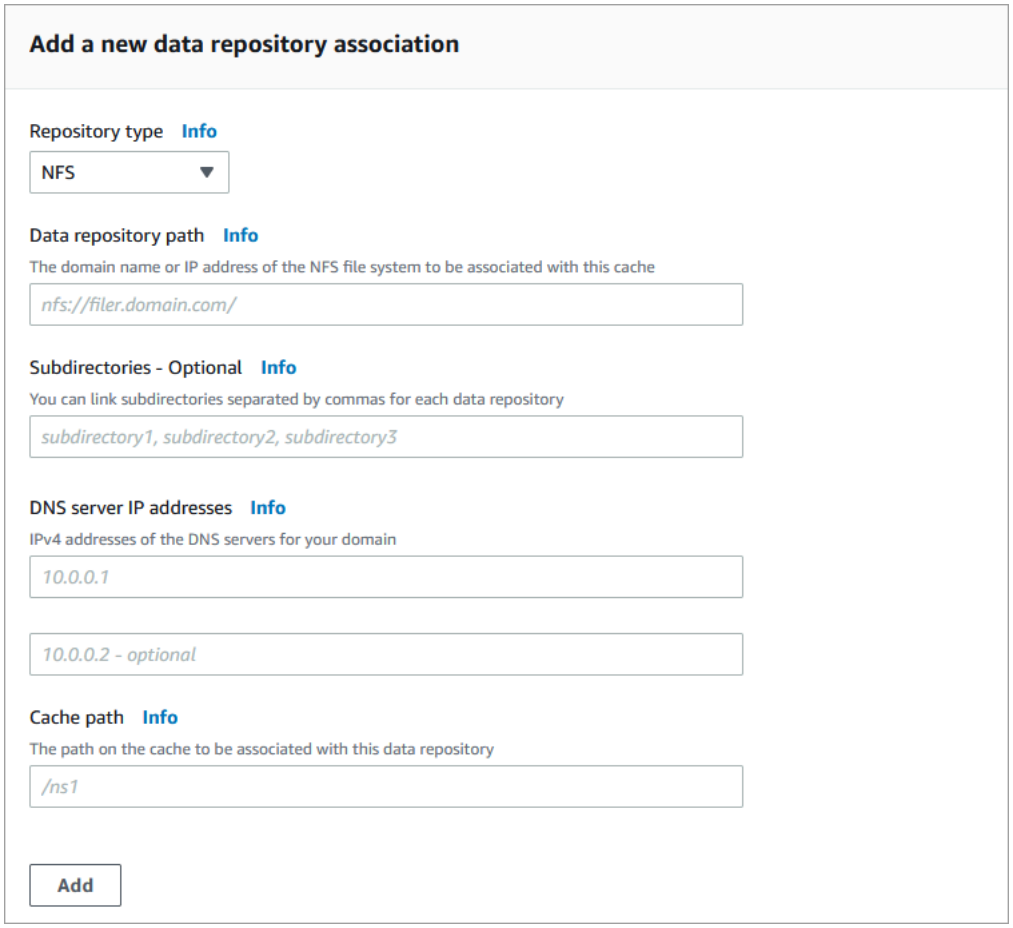
Keep the following in mind when working with DRAs.

- You can link to a data repository only when you create the cache.
- You can't update an existing DRA.
- You can't delete an existing DRA. To remove a link to a data repository, delete the cache and create it again.
- You can link your cache to either S3 data repositories or NFS data repositories, but not to both types in a single cache.

For information about using the AWS Command Line Interface (AWS CLI) to create a DRA while creating a cache, see To create a cache (CLI).

To link an S3 bucket or NFS file system while creating a cache (console)

1. Open the AWS Management Console at https://eusc-de-east-1.console.amazonaws-eusc.eu/fsx/.
2. Follow the procedure for creating a new Amazon File Cache described in Step 1: Create your cache.
3. In the **Data repository associations (DRAs)** section, the **Create a new data repository association** dialog box displays.



In the dialog box, provide information for the following fields.

- **Repository type** – Choose the type of data repository to link to:
  - NFS – NFS file system that supports the NFSv3 protocol.
  - S3 – Amazon S3 bucket
- **Data repository path** – Enter a path in either an S3 or NFS data repository to associate with your cache.

- For S3, the path can be an S3 bucket or prefix in the format `s3://myBucket/myPrefix/`. Amazon File Cache will append a trailing "/" to your data repository path if you don't provide one. For example, if you provide a data repository path of `s3://myBucket/myPrefix`, Amazon File Cache will interpret it as `s3://myBucket/myPrefix/`.
- For NFS, the path to the NFS data repository can be in one of two formats:
  - If you're not using **Subdirectories**, the path is to an NFS Export directory (or one of its subdirectories) in the format `nfs://nfs-domain-name/exportpath`.
  - If you're using **Subdirectories**, the path is the domain name of the NFS file system in the format `nfs://filer-domain-name`, which indicates the root of the NFS Export subdirectories specified with the `NFS Exports` field.

  Two data repository associations can't have overlapping data repository paths. For example, if a data repository with path `s3://myBucket/myPrefix/` is linked to the cache, you can't create another data repository association with data repository path `s3://myBucket/myPrefix/mySubPrefix`.

- **Subdirectories** – (NFS only) You can optionally provide a list of comma-delimited NFS export paths in the NFS data repository. When this field is provided, **Data repository path** can only contain the NFS domain name, indicating the root of the subdirectories.
- **DNS server IP addresses** – (NFS only) If you provided the domain name of the NFS file system for **Data repository path**, you can specify up to two IPv4 addresses of DNS servers used to resolve the NFS file system domain name. The provided IP addresses can either be the IP addresses of a DNS forwarder or resolver that the customer manages and runs inside the customer VPC, or the IP addresses of the on-premises DNS servers.
- **Cache path** – Enter the name of a high-level directory (such as `/ns1`) or subdirectory (such as `/ns1/subdir`) within the Amazon File Cache that will be associated with the data repository. The leading forward slash in the path is required. Two data repository associations cannot have overlapping cache paths. The **Cache path** setting must be unique across all the data repository associations for the cache.

  > **Note**
  >
  > **Cache path** can only be set to root (/) on NFS DRAs when **Subdirectories** is specified. If you specify root (/) as the **Cache path**, you can create only one DRA on the cache. **Cache path** cannot be set to root (/) for an S3 DRA.

4. When you finish configuring the DRA, choose **Add**.
5. You can add another data repository association using the same steps. You can create a maximum of 8 data repository associations, which must all be of the same repository type.
6. When you finish adding DRAs, choose **Next**.
7. Continue with the Amazon File Cache creation wizard.

Working with server-side encrypted Amazon S3 buckets

Amazon File Cache supports Amazon Simple Storage Service (Amazon S3) buckets that use server-side encryption with S3-managed keys (SSE-S3), and with AWS Key Management Service (AWS KMS) stored in AWS KMS (SSE-KMS).

If you want Amazon File Cache to encrypt data when writing to your S3 bucket, you must set the default encryption on your S3 bucket to either SSE-S3 or SSE-KMS. For more information, see Configuring default encryption in the *Amazon S3 User Guide*.

When writing files to your S3 bucket, Amazon File Cache follows the default encryption policy of your S3 bucket.

By default, Amazon File Cache supports S3 buckets encrypted using SSE-S3. If you want to link your Amazon File Cache to an S3 bucket encrypted using SSE-KMS encryption, you must add a statement to your customer managed key policy that allows Amazon File Cache to encrypt and decrypt objects in your S3 bucket using your AWS KMS key.

The following statement allows a specific Amazon File Cache to encrypt and decrypt objects for a specific S3 bucket, *bucket_name*.

```
{
    "Sid": "Allow access through S3 for the FSx SLR to use the KMS key on the objects
 in the given S3 bucket",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::aws_account_id:role/aws-service-role/s3.data-
source.lustre.fsx.amazonaws.com/AWSServiceRoleForFSxS3Access_file_cache_id"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
```

```
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:CallerAccount": "aws_account_id",
            "kms:ViaService": "s3.bucket-region.amazonaws.com"
        },
        "StringLike": {
            "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket_name/*"
        }
    }
}
```

Note

If you're using an AWS KMS with a CMK to encrypt your S3 bucket with S3 Bucket Keys enabled, set the `EncryptionContext` to the bucket ARN, not the object ARN, as in this example:

```
"StringLike": {
    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket_name"
}
```

The following policy statement allows every Amazon File Cache in your account to link to a specific S3 bucket.

```
{
    "Sid": "Allow access through S3 for the FSx SLR to use the KMS key on the objects
 in the given S3 bucket",
    "Effect": "Allow",
    "Principal": {
        "AWS": "*"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
```

```
            "kms:DescribeKey"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:CallerAccount": "aws_account_id",
                "kms:ViaService": "s3.bucket-region.amazonaws.com"
            },
            "StringLike": {
                "aws:userid": "*:FSx",
                "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket_name/*"
            }
        }
}
```

Accessing server-side encrypted Amazon S3 buckets in a different AWS account

After you create a cache linked to an encrypted Amazon S3 bucket, you must then grant the `AWSServiceRoleForFSxS3Access_fc-01234567890` service-linked role (SLR) access to the AWS KMS key used to encrypt the S3 bucket before reading or writing data from the linked S3 bucket. You can use an IAM role which already has permissions to the AWS KMS key.

> **Note**
>
> This IAM role must be in the account that the Amazon File Cache was created in (which is the same account as the S3 SLR), not the account that the AWS KMS key/S3 bucket belongs to.

You use the IAM role to call the following AWS KMS API to create a grant for the S3 SLR so that the SLR gains permission to the S3 objects. In order to find the ARN associated with your SLR, search your IAM roles using your cache ID as the search string.

```
$ aws kms create-grant --region cache_account_region \
      --key-id arn:aws:kms:s3_bucket_account_region:s3_bucket_account:key/key_id \
      --grantee-principal arn:aws:iam::cache_account_id:role/aws-service-role/s3.data-
source.lustre.fsx.amazonaws.com/AWSServiceRoleForFSxS3Access_file-cache-id \
      --operations "Decrypt" "Encrypt" "GenerateDataKey"
 "GenerateDataKeyWithoutPlaintext" "CreateGrant" "DescribeKey" "ReEncryptFrom"
 "ReEncryptTo"
```

For more information about service-linked roles, see Using service-linked roles for Amazon FSx.

Importing files from your data repository

When you create a Amazon File Cache resource, you can create a data repository association (DRA) to link your cache to an Amazon S3 or NFS data repository. Amazon File Cache transparently copies the content of a file from your repository and loads it into the cache, if it doesn't already exist, when your application accesses the file.

You can also preload your whole cache or an entire directory within your cache. For more information, see Preloading files into your cache.

This data movement is managed by Amazon File Cache and occurs transparently to your applications. Subsequent reads of these files are served directly out of Amazon File Cache with consistent sub-millisecond latencies. If you request the preloading of multiple files simultaneously, Amazon File Cache loads your files from your linked data repository in parallel. For more information, see Lazy load.

Amazon File Cache *only* imports objects that have POSIX-compliant object keys, such as:

```
test/mydir/
test/
```

Note

For a linked S3 bucket, Amazon File Cache doesn't support importing metadata for symbolic links (symlinks) from S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes. Metadata for S3 Glacier Flexible Retrieval objects that are not symlinks can be imported (that is, an inode is created on the cache with the correct metadata). However, to retrieve the data, you must restore the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive object first and then use an `hsm_restore` command to import the object. Importing file data directly from Amazon S3 objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class into Amazon File Cache is not supported.

Lazy load

When you access data on a linked Amazon S3 or NFS data repository using the cache, Amazon File Cache automatically loads the metadata (the name, ownership, timestamps, and permissions) and

file contents if they're not already present in the cache. The data in your data repositories appears as files and directories in the cache.

Lazy load is triggered when you're in a DRA directory and you read or write data or metadata to a file. Amazon File Cache loads data into the cache from the linked data repositories if it's not already available. For example, lazy load is triggered when you open a file, stat a file, or make metadata updates to the file.

You can also trigger lazy load by using the `ls` command to list the contents of a DRA directory. If you're at the root of a directory hierarchy that includes several DRA directories, the `ls` command will use lazy load on all the DRA directories in the hierarchy. For example, if you're at `/` in the directory tree, and your four DRAs are `/a`, `/b`, `/c`, and `/d`, then running a recursive `ls` command populates metadata for all DRAs. To run a recursive `ls` command, use the `-R` option shown in the examples below:

```
ls -R
ls -R /tmp/dir1
```

When you use the `ls` or `stat` commands, Amazon File Cache only loads file and directory metadata for requested files; no file content will be downloaded. The data from a file in the data repository is actually downloaded to your cache when the file is read.

> **Note**
>
> Amazon File Cache only loads a directory listing the first time `ls` is run on a directory. Subsequently, if new files are added or existing files are changed in the corresponding directory in the linked data repository, you can `stat` the file path to update the directory listing.

Preloading files into your cache

If the data you're accessing doesn't already exist in the cache, Amazon File Cache copies the data from your Amazon S3 or NFS data repository into the cache in line with file access. Because of this approach, the initial read or write to a file incurs a small amount of latency. If your application is sensitive to this latency, and you know which files or directories your application needs to access, you can optionally preload contents of individual files or directories. You do so using the `hsm_restore` command.

You can use the `hsm_action` command (issued with the `lfs` user utility) to verify that the file's contents have finished loading into the cache. A return value of `NOOP` indicates that the file has successfully been loaded. Run the following commands from a compute instance with the cache mounted. Replace *`path/to/file`* with the path of the file you're preloading into your cache.

```
sudo lfs hsm_restore path/to/file
sudo lfs hsm_action path/to/file
```

You can preload your whole cache or an entire directory within your cache by using the following commands. (The trailing ampersand makes a command run as a background process.) If you request the preloading of multiple files simultaneously, Amazon File Cache loads your files from your linked data repository in parallel.

```
nohup find local/directory -type f -print0 | xargs -0 -n 1 sudo lfs hsm_restore &
```

Note

If your linked data repository is larger than your cache, you can only load as much actual file data as will fit into the cache's remaining storage space. You'll receive an error if you attempt to access file data when there's no more storage remaining in the cache.

Exporting changes to the data repository

You can export data and metadata changes, including POSIX metadata, from Amazon File Cache to a linked Amazon S3 or NFS data repository. Associated POSIX metadata includes ownership, permissions, and timestamps. To export changes from the cache, use HSM commands. When you export a file or directory using HSM commands, your cache exports only data files and metadata that were created or modified since the last export. For more information, see Exporting files using HSM commands.

Important

For Amazon File Cache to export your data to your linked data repository, it must be stored in a UTF-8 compatible format.

Topics

Exporting files using HSM commands

To export an individual file to your data repository and verify the success of the export, run the following commands. A return value of `states: (0x00000009) exists archived` indicates that export was successful.

```
sudo lfs hsm_archive path/to/export/file
sudo lfs hsm_state path/to/export/file
```

> **Note**
>
> You must run the HSM commands (such as `hsm_archive`) as the root user or using `sudo`.

To export changes on an entire cache or an entire directory in your cache, run the following commands. If you export multiple files simultaneously, Amazon File Cache exports your files to your data repository in parallel.

```
nohup find local/directory -type f -print0 | xargs -0 -n 1 sudo lfs hsm_archive &
```

To determine whether the export is complete, run the following command.

```
find path/to/export/file -type f -print0 | xargs -0 -n 1 -P 8 sudo lfs hsm_state | awk
 '!/\<archived\>/ || /\<dirty\>/' | wc -l
```

If the command returns with zero files remaining, the export is complete.

Cache eviction

Files can be evicted (released) from the cache to free up space for new files. Releasing a file retains the file listing and metadata, but removes the local copy of that file's contents. You can't release a file if it's in use or if it hasn't been exported to a linked data repository. There are two methods to release files:

- Automatic cache eviction releases files automatically when the cache begins to fill up.

- Manual release using HSM commands to release files.

> **Important**
>
> Both methods only release files that are in the archived state. You must first export the files to your linked data repository using HSM commands, as described in Exporting files using HSM commands.

Automatic cache eviction

Amazon File Cache automatically manages the cache storage capacity by releasing the less recently used files on your cache when the cache begins to fill up. Automatic cache eviction is enabled by default when you create a cache using the File Cache console, AWS CLI, or the AWS API.

Releasing files using HSM commands

You can manually release individual files from your cache using the following commands:

- To release one or more files from your cache if you are the file owner:

```
lfs hsm_release file1 file2 ...
```

- To release one or more files from your cache if you're not the file owner:

```
sudo lfs hsm_release file1 file2 ...
```

The `hsm_release` command can only release regular files as defined by POSIX. You cannot release sockets, symbolic links, block devices, character devices, or named pipes. To identify whether a file is a regular file, use the `test -f` command on that file.

Amazon File Cache performance

Amazon File Cache is built on Lustre, the popular, high-performance file system, and provides scale-out performance that increases linearly with an Amazon File Cache's size. Lustre file systems scale horizontally across multiple file servers and disks. This scaling gives each client direct access to the data stored on each disk to remove many of the bottlenecks present in traditional file systems. Amazon File Cache builds on Lustre's scalable architecture to support high levels of performance across large numbers of clients.

Topics

How Amazon File Cache works

Each Amazon File Cache resource consists of the file servers that the clients communicate with, and a set of disks attached to each file server that stores your cache data. Each file server employs a fast, in-memory cache to enhance performance for the most frequently accessed data. When a client accesses data that's stored in the in-memorycache, the file server doesn't need to read it from disk, which reduces latency and increases the total amount of throughput you can drive.

When you read data that's stored in the file server's in-memory cache, your cache performance is determined by the network throughput. When you write data to your cache, or when you read data that isn't stored in the in-memory cache, your cache performance is determined by the lower of the network throughput and disk throughput.

Aggregate cache performance

The throughput that an Amazon File Cache supports is proportional to its storage capacity. Amazon File Cache scales to hundreds of GBps of throughput and millions of IOPS. Amazon File

Cache also supports concurrent access to the same file or directory from thousands of compute instances. This access enables rapid data checkpointing from application memory to storage, which is a common technique in high performance computing (HPC).

The following table shows performance that the Amazon File Cache deployment type is designed for.

File Cache performance for SSD storage

Deployment Type	Network throughput (MBps/TiB of storage provisioned)	Network IOPS (IOPS/TiB of storage provisioned)	Cache storage (GiB of RAM/ TiB of storage provisioned)	Disk latencies per file operation (milliseconds, P50)	Disk throughput (MBps/TiB of storage provisioned)
CACHE-1000	2600	Tens of thousands baseline Hundreds of thousands burst	27.3	Metadata: sub-ms Data: sub-ms	1000

Example: Aggregate baseline and burst throughput

The following example illustrates how storage capacity and disk throughput impact cache performance.

A cache with a storage capacity of 4.8 TiB and 1000 MBps per TiB of throughput per unit of storage provides an aggregate disk throughput of 4800 MBps.

Regardless of cache size, Amazon File Cache provides consistent, sub-millisecond latencies for file operations.

File storage layout

All file data in Lustre is stored on storage volumes called *object storage targets* (OSTs). All file metadata (including file names, timestamps, permissions, and more) is stored on storage volumes called *metadata targets* (MDTs). An Amazon File Cache is composed of multiple MDTs and OSTs. Each OST is approximately 1.2 TiB in size. Amazon File Cache spreads your file data across the OSTs that make up your cache to balance storage capacity with throughput and IOPS load.

To view the storage usage of the MDTs and OSTs that make up your cache, run the following command from a client that has the cache mounted.

```
lfs df -h mount/path
```

The output of this command looks like the following.

Example

```
UUID                            bytes       Used   Available Use% Mounted on
mountname-MDT0000_UUID          68.7G       5.4M       68.7G   0% /fsx[MDT:0]
mountname-OST0000_UUID           1.1T       4.5M        1.1T   0% /fsx[OST:0]
mountname-OST0001_UUID           1.1T       4.5M        1.1T   0% /fsx[OST:1]

filesystem_summary:              2.2T       9.0M        2.2T   0% /fsx
```

Striping data in your cache

You can optimize your cache's throughput performance with file striping. Amazon File Cache automatically spreads out files across OSTs in order to ensure that data is served from all storage

servers. You can apply the same concept at the file level by configuring how files are striped across multiple OSTs.

Striping means that files can be divided into multiple chunks that are then stored across different OSTs. When a file is striped across multiple OSTs, read or write requests to the file are spread across those OSTs, increasing the aggregate throughput or IOPS your applications can drive through it.

You can view the layout configuration of a file or directory using the `lfs getstripe` command.

```
lfs getstripe path/to/filename
```

This command reports a file's stripe count, stripe size, and stripe offset. The *stripe count* is how many OSTs the file is striped across. The *stripe size* is how much continuous data is stored on an OST. The *stripe offset* is the index of the first OST that the file is striped across.

Modifying your striping configuration

A file's layout parameters are set when the file is first created. Use the `lfs setstripe` command to create a new, empty file with a specified layout.

```
lfs setstripe filename --stripe-count number_of_OSTs
```

The `lfs setstripe` command affects only the layout of a new file. Use it to specify the layout of a file before you create it. You can also define a layout for a directory. Once set on a directory, that layout is applied to every new file added to that directory, but not to existing files. Any new subdirectory you create also inherits the new layout, which is then applied to any new file or directory you create within that subdirectory.

To modify the layout of an existing file, use the `lfs migrate` command. This command copies the file as needed to distribute its content according to the layout you specify in the command. For example, files that are appended to or are increased in size don't change the stripe count, so you have to migrate them to change the file layout. Alternatively, you can create a new file using the `lfs setstripe` command to specify its layout, copy the original content to the new file, and then rename the new file to replace the original file.

There can be cases where the default layout configuration is not optimal for your workload. For example, a cache with tens of OSTs and a large number of multi-gigabyte files may see higher performance by striping the files across more than the default stripe count value of five OSTs.

Creating large files with low stripe counts can cause I/O performance bottlenecks and can also cause OSTs to fill up. In this case, you can create a directory with a larger stripe count for these files.

Setting up a striped layout for large files (especially files larger than a gigabyte in size) is important for the following reasons:

- Improves throughput by allowing multiple OSTs and their associated servers to contribute IOPS, network bandwidth, and CPU resources when reading and writing large files.
- Reduces the likelihood that a small subset of OSTs become hot spots that limit overall workload performance.
- Prevents a single large file from filling an OST, possibly causing disk full errors.

There is no single optimal layout configuration for all use cases. For detailed guidance on file layouts, see Managing File Layout (Striping) and Free Space in the Lustre.org documentation. The following are general guidelines:

- Striped layout matters most for large files, especially for use cases where files are routinely hundreds of megabytes or more in size. For this reason, the default layout for a new cache assigns a striped count of five for files over 1 GiB in size.
- Stripe count is the layout parameter that you should adjust for systems supporting large files. The stripe count specifies the number of OST volumes that will hold chunks of a striped file. For example, with a stripe count of 2 and a stripe size of 1 MiB, Lustre writes alternate 1 MiB chunks of a file to each of two OSTs.
- The effective stripe count is the lesser of the actual number of OST volumes and the stripe count value you specify. You can use the special stripe count value of `-1` to indicate that stripes should be placed on all OST volumes.
- Setting a large stripe count for small files is sub-optimal because for certain operations Lustre requires a network round trip to every OST in the layout, even if the file is too small to consume space on all the OST volumes.
- You can set up a progressive file layout (PFL) that allows the layout of a file to change with size. A PFL configuration can simplify managing a cache that has a combination of large and small files without you having to explicitly set a configuration for each file. For more information, see Progressive file layouts.
- Stripe size by default is 1 MiB. Setting a stripe offset may be useful in special circumstances, but in general it's best to leave it unspecified and use the default.

Progressive file layouts

You can specify a progressive file layout (PFL) configuration for a directory to specify different stripe configurations for small and large files before populating it. For example, you can set a PFL on the top-level directory before any data is written to a new cache.

To specify a PFL configuration, use the `lfs setstripe` command with `-E` options to specify layout components for different sized files, such as the following command:

```
lfs setstripe -E 100M -c 1 -E 10G -c 8 -E 100G -c 16 -E -1 -c 32 /mountname/directory
```

This command sets four layout components:

- The first component (`-E 100M -c 1`) indicates a stripe count value of 1 for files up to 100MiB in size.
- The second component (`-E 10G -c 8`) indicates a stripe count of 8 for files up to 10GiB in size.
- The third component (`-E 100G -c 16`) indicates a stripe count of 16 for files up to 100GiB in size.
- The fourth component (`-E -1 -c 32`) indicates a stripe count of 32 for files larger than 100GiB.

Important

Appending data to a file created with a PFL layout will populate all of its layout components. For example, with the 3-component command shown above, if you create a 1 MiB file and then add data to the end of it, the layout of the file will expand to have a stripe count of -1, meaning all the OSTs in the system. This doesn't mean data will be written to every OST, but an operation such as reading the file length will send a request in parallel to every OST, adding significant network load to the cache.
Therefore, be careful to limit the stripe count for any small or medium length file that can subsequently have data appended to it. Because log files usually grow by having new records appended, Amazon File Cache assigns a default stripe count of 1 to any file created in append mode, regardless of the default stripe configuration specified by its parent directory.

The default PFL configuration on caches is set with this command:

```
lfs setstripe -E 100M -c 1 -E 10G -c 8 -E 100G -c 16 -E -1 -c 32 /mountname
```

Customers with workloads that have highly concurrent access on medium and large files are likely to benefit from a layout with more stripes at smaller sizes and striping across all OSTs for the largest files, as shown in the four-component example layout shown previously.

Monitoring performance and usage

Every minute, Amazon File Cache emits usage metrics for each disk (MDT and OST) to Amazon CloudWatch.

To view aggregate cache usage details, you can look at the Sum statistic of each metric. For example, the Sum of the `DataReadBytes` statistic reports the total read throughput seen by all the OSTs in a cache. Similarly, the Sum of the `FreeDataStorageCapacity` statistic reports the total available storage capacity for file data in the cache.

For more information about monitoring your cache's performance, see Monitoring Amazon File Cache.

Performance tips

When using Amazon File Cache, keep the following performance tips in mind. For service limits, see Quotas.

- **Average I/O size** – Because Amazon File Cache is a network file system, each file operation goes through a round trip between the client and Amazon File Cache, incurring a small latency overhead. 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.
- **Request model** – By enabling asynchronous writes to your cache, pending write operations are buffered on the Amazon EC2 instance before they're written to Amazon File Cache asynchronously. Asynchronous writes typically have lower latencies. When performing asynchronous writes, the kernel uses additional memory for caching. A cache that has enabled synchronous writes issues synchronous requests to Amazon File Cache. Every operation goes through a round trip between the client and Amazon File Cache.

> **Note**
>
> Your chosen request model has tradeoffs in consistency (if you're using multiple Amazon EC2 instances) and speed.

- **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 for your compute-intensive workload, choose instance types that have the amount of these resources that your application needs. The performance characteristics of Amazon File Cache don't depend on the use of Amazon EBS–optimized instances.
- **Recommended tuning for large client instance types**

  1. To tune large client instances for optimal performance:

     a. For client instance types with memory of more than 64 GiB, we recommend applying the following tuning:

     ```
     lctl set_param ldlm.namespaces.*.lru_max_age=600000
     ```

     b. For client instance types with more than 64 CPU cores, we recommend applying the following tuning:

     ```
     echo "options ptlrpc ptlrpcd_per_cpt_max=32" >> /etc/modprobe.d/modprobe.conf
     echo "options ksocklnd credits=2560" >> /etc/modprobe.d/modprobe.conf

     # reload all kernel modules to apply the above two settings
     sudo reboot
     ```

  2. After the client is mounted, the following tuning needs to be applied:

     ```
     sudo lctl set_param osc.*OST*.max_rpcs_in_flight=32
     sudo lctl set_param mdc.*.max_rpcs_in_flight=64
     sudo lctl set_param mdc.*.max_mod_rpcs_in_flight=50
     ```

     Note that `lctl set_param` is known to not persist over reboot. Since these parameters can't be set permanently from the client side, it's recommended to implement a boot cron job to set the configuration with the recommended tunings.

- **Workload balance across OSTs** – In some cases, your workload isn’t driving the aggregate throughput that your cache can provide (1000 MBps per TiB of storage). If so, you can use

CloudWatch metrics to troubleshoot if performance is affected by an imbalance in your workload's I/O patterns. To identify if this is the cause, look at the Maximum `DataReadBytes` and `DataWriteBytes` CloudWatch metrics for Amazon File Cache.

In some cases, this statistic shows a load at or above 1200 MBps of throughput (the throughput capacity of a single 1.2 TiB Amazon File Cache disk). In such cases, your workload isn't evenly spread out across your disks. If this is the case, you can use the `lfs setstripe` command to modify the striping of files your workload is most frequently accessing. For optimal performance, stripe files with high throughput requirements across all the OSTs comprising your cache.

Accessing caches

In the following topics, you can learn how to access your cache on a Linux instance. In addition, you can find how to use the file `fstab` to automatically remount your cache after any system restarts.

Before you can mount a cache, you must create, configure, and launch your related AWS resources. For detailed instructions, see Getting started with Amazon File Cache.

Next, you can install and configure the Lustre client on your compute instance.

Topics

Installing the Lustre client

To mount your cache from a Linux instance, first install the open-source Lustre client. Amazon File Cache version 2.12 supports access from the 2.12 versions of the Lustre client. Then, depending on your operating system version, use one of the following procedures.

If your compute instance isn't running the Linux kernel specified in the installation instructions, and you can't change the kernel, you can build your own Lustre client. For more information, see Compiling Lustre on the Lustre Wiki.

Amazon Linux 2 and Amazon Linux

To install the Lustre client on Amazon Linux 2

1. Open a terminal on your client.

2. Determine which kernel is currently running on your compute instance by running the following command.

```
uname -r
```

3. Do one of the following:

   - Kernel requirements:
     - 5.10 kernel minimum requirement - 5.10.155-138.670.amzn2
     - 5.4 kernel minimum requirement - 5.4.219-126.411.amzn2
     - 4.14 kernel minimum requirement - 4.14.299-223.520.amzn2

     Download and install the Lustre client with the following command.

     ```
     sudo amazon-linux-extras install -y lustre
     ```

   - If the command returns a result less than the kernel minimum requirement, update the kernel and reboot your Amazon EC2 instance by running the following command.

     ```
     sudo yum -y update kernel && sudo reboot
     ```

     Confirm that the kernel has been updated using the **uname -r** command. Then download and install the Lustre client as described previously.

To install the Lustre client on Amazon Linux

1. Open a terminal on your client.
2. Determine which kernel is currently running on your compute instance by running the following command.

```
uname -r
```

   The Lustre client requires Amazon Linux with minimum kernel version of 4.14.299-152.520.amzn1.

3. Do one of the following:

   - If the command returns a result equal to or greater than the kernel minimum requirement, download and install the Lustre client using the following command.

```
sudo yum install -y lustre-client
```

- If the command returns a result less than the kernel minimum requirement, update the kernel and reboot your Amazon EC2 instance by running the following command.

  ```
  sudo yum -y update kernel && sudo reboot
  ```

  Confirm that the kernel has been updated using the **uname -r** command. Then download and install the Lustre client as described previously.

CentOS, Rocky Linux, and Red Hat

To install the Lustre client on CentOS, Rocky Linux, and Red Hat 8.4 and newer

You can install and update Lustre client packages that are compatible with Red Hat Enterprise Linux (RHEL), Rocky Linux, and CentOS from the Lustre client yum package repository. These packages are signed to help verify that they haven't been tampered with before or during download. The repository installation fails if you don't install the corresponding public key on your system.

To add the Lustre client yum package repository

1. Open a terminal on your client.
2. Install the AWS Lustre rpm public key by using the following command.

   ```
   curl https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-rpm-public-key.asc -o /tmp/fsx-rpm-public-key.asc
   ```

3. Import the key by using the following command.

   ```
   sudo rpm --import /tmp/fsx-rpm-public-key.asc
   ```

4. Add the repository and update the package manager using the following command.

   ```
   sudo curl https://fsx-lustre-client-repo.s3.amazonaws.com/el/8/fsx-lustre-client.repo -o /etc/yum.repos.d/aws-fsx.repo
   ```

To configure the Lustre client yum repository

The Lustre client yum package repository is configured by default to install the Lustre client that's compatible with the kernel version that initially shipped with the latest supported CentOS, Rocky Linux, and RHEL 8 release. To install a Lustre client that's compatible with the kernel version you're using, edit the repository configuration file.

This section describes how to determine which kernel you're running, whether you need to edit the repository configuration, and how to edit the configuration file.

1. Determine which kernel is currently running on your compute instance by using the following command.

   ```
   uname -r
   ```

2. Do one of the following:

   - If the command returns `4.18.0-477*`, you don't need to modify the repository configuration. Continue to the **To install the Lustre client** procedure.
   - If the command returns `4.18.0-425*`, you must edit the repository configuration so that it points to the Lustre client for the CentOS, Rocky Linux, and RHEL 8.7 release.
   - If the command returns `4.18.0-372*`, you must edit the repository configuration so that it points to the Lustre client for the CentOS, Rocky Linux, and RHEL 8.6 release.
   - If the command returns `4.18.0-348*`, you must edit the repository configuration so that it points to the Lustre client for the CentOS, Rocky Linux, and RHEL 8.5 release.
   - If the command returns `4.18.0-305*`, you must edit the repository configuration so that it points to the Lustre client for the CentOS, Rocky Linux, and RHEL 8.4 release.

3. Edit the repository configuration file to point to a specific version of RHEL using the following command.

   ```
   sudo sed -i 's#8#specific_RHEL_version#' /etc/yum.repos.d/aws-fsx.repo
   ```

   For example, to point to release 8.7, substitute *`specific_RHEL_version`* with `8.7` in the command.

   ```
   sudo sed -i 's#8#8.7#' /etc/yum.repos.d/aws-fsx.repo
   ```

4. Use the following command to clear the yum cache.

```
sudo yum clean all
```

To install the Lustre client

- Install the packages from the repository using the following command.

```
sudo yum install -y kmod-lustre-client lustre-client
```

Additional information (CentOS, Rocky Linux, and Red Hat 8.4 and newer)

The preceding commands install the two packages that are necessary for mounting and interacting with your cache. The repository includes additional Lustre packages, such as a package containing the source code and packages containing tests, which you can optionally install. To list all available packages in the repository, use the following command.

```
yum --disablerepo="*" --enablerepo="aws-fsx" list available
```

To download the source rpm, containing a tarball of the upstream source code and the set of patches that we've applied, use the following command.

```
 sudo yumdownloader --source kmod-lustre-client
```

When you run yum update, a more recent version of the module is installed if available and the existing version is replaced. To prevent the currently installed version from being removed on update, add a line like the following to your `/etc/yum.conf` file.

```
installonlypkgs=kernel, kernel-PAE, installonlypkg(kernel), installonlypkg(kernel-
module),
              installonlypkg(vm), multiversion(kernel), kmod-lustre-client
```

This list includes the default install only packages, specified in the `yum.conf` man page, and the `kmod-lustre-client` package.

To install the Lustre client on CentOS and Red Hat 7.9 (x86_64 instances)

You can install and update Lustre client packages that are compatible with Red Hat Enterprise Linux (RHEL) and CentOS from the Lustre client yum package repository. These packages are

signed to help ensure that they haven't been tampered with before or during download. The repository installation fails if you don't install the corresponding public key on your system.

To add the AWS Lustre client yum package repository

1. Open a terminal on your client.
2. Install the AWS Lustre rpm public key using the following command.

```
curl https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-rpm-public-key.asc -o /tmp/fsx-rpm-public-key.asc
```

3. Import the key using the following command.

```
sudo rpm --import /tmp/fsx-rpm-public-key.asc
```

4. Add the repository and update the package manager using the following command.

```
sudo curl https://fsx-lustre-client-repo.s3.amazonaws.com/el/7/fsx-lustre-client.repo -o /etc/yum.repos.d/aws-fsx.repo
```

The AWS Lustre client yum package repository is configured by default to install the Lustre client that's compatible with the kernel version that initially shipped with the latest supported CentOS 7 release. If you use the `uname -r` command to determine which kernel you are running, the command should return `3.10.0-1160*`.

To install the Lustre client

- Install the Lustre client packages from the repository using the following command.

```
sudo yum install -y kmod-lustre-client lustre-client
```

Additional information (CentOS and Red Hat 7.9)

The preceding commands install the two packages that are necessary for mounting and interacting with your cache. The repository includes additional Lustre packages, such as a package containing the source code and packages containing tests, which you can optionally install. To list all available packages in the repository, use the following command.

```
yum --disablerepo="*" --enablerepo="aws-fsx" list available
```

To download the source rpm containing a tarball of the upstream source code and the set of patches that we've applied, use the following command.

```
 sudo yumdownloader --source kmod-lustre-client
```

When you run yum update, a more recent version of the module is installed if available, and the existing version is replaced. To prevent the currently installed version from being removed on update, add a line like the following to your `/etc/yum.conf` file.

```
installonlypkgs=kernel, kernel-big-mem, kernel-enterprise, kernel-smp,
              kernel-debug, kernel-unsupported, kernel-source, kernel-devel, kernel-
PAE,
              kernel-PAE-debug, kmod-lustre-client
```

This list includes the default install only packages, specified in the `yum.conf` man page, and the `kmod-lustre-client` package.

To install the Lustre client on CentOS 7.9 (Arm-based AWS Graviton-powered instances)

You can install Lustre client packages from the AWS Lustre client yum package repository that are compatible with CentOS 7 for Arm-based AWS Graviton-powered EC2 instances. These packages are signed to help verify they haven't been tampered with before or during download. The repository installation fails if you don't install the corresponding public key on your system.

To add the AWS Lustre client yum package repository

1. Open a terminal on your client.
2. Install the AWS Lustre rpm public key using the following command.

   ```
   curl https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-rpm-public-
   key.asc -o /tmp/fsx-rpm-public-key.asc
   ```

3. Import the key using the following command.

   ```
   sudo rpm --import /tmp/fsx-rpm-public-key.asc
   ```

4. Add the repository and update the package manager using the following command.

```
sudo curl https://fsx-lustre-client-repo.s3.amazonaws.com/centos/7/fsx-lustre-
client.repo -o /etc/yum.repos.d/aws-fsx.repo
```

The AWS Lustre client yum package repository is configured by default to install the Lustre client that is compatible with the kernel version that initially shipped with the latest supported CentOS 7 release. If you use the `uname -r` command to determine which kernel you are running, the command should return `4.18.0-193*`.

To install the Lustre client

- Install the packages from the repository using the following command.

```
sudo yum install -y kmod-lustre-client lustre-client
```

Additional information (CentOS 7.9 for Arm-based AWS Graviton-powered EC2 instances)

The preceding commands install the two packages that are necessary for mounting and interacting with your cache. The repository includes additional Lustre packages, such as a package containing the source code and packages containing tests, which you can optionally install. To list all available packages in the repository, use the following command.

```
yum --disablerepo="*" --enablerepo="aws-fsx" list available
```

To download the source rpm, containing a tarball of the upstream source code and the set of patches that we've applied, use the following command.

```
 sudo yumdownloader --source kmod-lustre-client
```

When you run yum update, a more recent version of the module is installed if available, and the existing version is replaced. To prevent the currently installed version from being removed on update, add a line like the following to your `/etc/yum.conf` file.

```
installonlypkgs=kernel, kernel-big-mem, kernel-enterprise, kernel-smp,
              kernel-debug, kernel-unsupported, kernel-source, kernel-devel, kernel-
PAE,
              kernel-PAE-debug, kmod-lustre-client
```

This list includes the default install only packages, specified in the `yum.conf` man page, and the `kmod-lustre-client` package.

Ubuntu

To install the Lustre client on Ubuntu 22.04

Starting with kernel `5.15.0.1020-aws`, Ubuntu 22.04.1 LTS is supported for Lustre 2.12 clients, for both x86 and Arm based instances.

> **Note**
>
> FSx-vended Lustre clients are not supported on kernel 5.19.

You can get Lustre packages from the Ubuntu 22.04 AWS Lustre repository. To validate that the contents of the repository haven't been tampered with before or during download, a GNU Privacy Guard (GPG) signature is applied to the metadata of the repository. Installing the repository fails unless you have the correct public GPG key installed on your system.

1. Open a terminal on your client.
2. Follow these steps to add the AWS Lustre client repository:

   a. If you haven't previously registered an AWS Lustre client Ubuntu repository on your client instance, download and install the required public key. Use the following command.

   ```
   wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-key.gpg >/dev/null
   ```

   b. Add the AWS Lustre package repository to your local package manager using the following command.

   ```
   sudo bash -c 'echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg] https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /etc/apt/sources.list.d/fsxlustreclientrepo.list && apt-get update'
   ```

3. Determine which kernel is currently running on your client instance, and update as needed. The Lustre client on Ubuntu 22.04 requires kernel `5.15.0.1020-aws` or later for both x86-based EC2 instances and Arm-based EC2 instances powered by AWS Graviton processors.

a. Run the following command to determine which kernel is running.

```
uname -r
```

b. Run the following command to update to the latest Ubuntu kernel and Lustre version and then reboot.

```
sudo apt install -y linux-aws lustre-client-modules-aws && sudo reboot
```

If your kernel version is greater than `5.15.0.1020-aws` for both x86-based EC2 instances and Graviton-based EC2 instances, and you don't want to update to the latest kernel version, you can install Lustre for the current kernel with the following command.

```
sudo apt install -y lustre-client-modules-$(uname -r)
```

The two AWS Lustre packages that are necessary for mounting and interacting with your cache are installed. You can optionally install additional related packages such as a package containing the source code and packages containing tests that are included in the repository.

c. List all available packages in the repository by using the following command.

```
sudo apt-cache search ^lustre
```

d. (Optional) If you want your system upgrade to also always upgrade Lustre client modules, verify that the `lustre-client-modules-aws` package is installed using the following command.

```
sudo apt install -y lustre-client-modules-aws
```

To install the Lustre client on Ubuntu 20.04

Starting with kernel `5.15.0.1020-aws`, Ubuntu 20.04.5 LTS is supported for Lustre 2.12 clients, for both x86 and Arm based instances.

You can get Lustre packages from the Ubuntu 20.04 AWS Lustre repository. To validate that the contents of the repository haven't been tampered with before or during download, a GNU Privacy

Guard (GPG) signature is applied to the metadata of the repository. Installing the repository fails unless you have the correct public GPG key installed on your system.

1. Open a terminal on your client.
2. Follow these steps to add the AWS Lustre client Ubuntu repository:

   a. If you haven't previously registered an AWS Lustre Ubuntu repository on your client instance, download and install the required public key. Use the following command.

   ```
   wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-key.gpg >/dev/null
   ```

   b. Add the AWS Lustre package repository to your local package manager using the following command.

   ```
   sudo bash -c 'echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg] https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu focal main" > /etc/apt/sources.list.d/fsxlustreclientrepo.list && apt-get update'
   ```

3. Determine which kernel is currently running on your client instance, and update as needed. The Lustre client on Ubuntu 20.04 requires kernel `5.15.0.1020-aws` or later for both x86-based EC2 instances and Arm-based EC2 instances powered by AWS Graviton processors.

   a. Run the following command to determine which kernel is running.

   ```
   uname -r
   ```

   b. Run the following command to update to the latest Ubuntu kernel and Lustre version and then reboot.

   ```
   sudo apt install -y linux-aws lustre-client-modules-aws && sudo reboot
   ```

   If your kernel version is greater than `5.15.0.1020-aws` for both x86-based EC2 instances and Graviton-based EC2 instances, and you don’t want to update to the latest kernel version, you can install Lustre for the current kernel with the following command.

   ```
   sudo apt install -y lustre-client-modules-$(uname -r)
   ```

The two AWS Lustre packages that are necessary for mounting and interacting with your cache are installed. You can optionally install additional related packages such as a package containing the source code and packages containing tests that are included in the repository.

c. List all available packages in the repository by using the following command.

```
sudo apt-cache search ^lustre
```

d. (Optional) If you want your system upgrade to also always upgrade Lustre client modules, verify that the `lustre-client-modules-aws` package is installed using the following command.

```
sudo apt install -y lustre-client-modules-aws
```

To install the Lustre client on Ubuntu 18.04

Starting with kernel `5.4.0.1085-aws`, Ubuntu 18.04.6 LTS is supported for Lustre 2.12 clients, for both x86 and Arm based instances.

You can get Lustre packages from the Ubuntu 18.04 AWS Lustre repository. To validate that the contents of the repository haven't been tampered with before or during download, a GNU Privacy Guard (GPG) signature is applied to the metadata of the repository. Installing the repository fails unless you have the correct public GPG key installed on your system.

1. Open a terminal on your client.
2. Follow these steps to add the AWS Lustre Ubuntu repository:

   a. If you haven't previously registered an AWS Lustre Ubuntu repository on your client instance, download and install the required public key. Use the following command.

```
wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-key.gpg >/dev/null
```

   b. Add the AWS Lustre package repository to your local package manager using the following command.

```
sudo bash -c 'echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-
key.gpg] https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu bionic main"
 > /etc/apt/sources.list.d/fsxlustreclientrepo.list && apt-get update'
```

3. Determine which kernel is currently running on your client instance, and update as needed. The Lustre client on Ubuntu 18.04 requires kernel `15.4.0.1085-aws` or later for both x86-based EC2 instances and Arm-based EC2 instances powered by AWS Graviton processors.

   a. Run the following command to determine which kernel is running.

   ```
   uname -r
   ```

   b. Run the following command to update to the latest Ubuntu kernel and Lustre version and then reboot.

   ```
   sudo apt install -y linux-aws lustre-client-modules-aws && sudo reboot
   ```

   If your kernel version is greater than `5.4.0.1085-aws` for both x86-based EC2 instances and Graviton-based EC2 instances, and you don't want to update to the latest kernel version, you can install Lustre for the current kernel with the following command.

   ```
   sudo apt install -y lustre-client-modules-$(uname -r)
   ```

   The two Lustre packages that are necessary for mounting and interacting with your cache are installed. You can optionally install additional related packages, such as a package containing the source code and packages containing tests that are included in the repository.

   c. List all available packages in the repository by using the following command.

   ```
   sudo apt-cache search ^lustre
   ```

   d. (Optional) If you want your system upgrade to also always upgrade Lustre client modules, make sure that the `lustre-client-modules-aws` package is installed using the following command.

   ```
   sudo apt install -y lustre-client-modules-aws
   ```

Mounting from an Amazon EC2 instance

You can mount your cache from an Amazon EC2 instance.

To mount your cache from Amazon EC2

1. Connect to your Amazon EC2 instance.
2. Make a directory on your cache for the mount point with the following command.

```
$ sudo mkdir -p /mnt
```

3. Mount the cache to the directory that you created. Use the following command and replace the following items:

   - Replace *cache_dns_name* with the actual file cache's DNS name.
   - Replace *mountname* with the cache's mount name. This mount name is returned in the `CreateFileCache` API operation response. It's also returned in the response of the **describe-file-caches** AWS CLI command, and the DescribeFileCaches API operation.

```
sudo mount -t lustre -o relatime,flock cache_dns_name@tcp:/mountname /mnt
```

   This command mounts your cache with these options:

   - `relatime` – Maintains `atime` (inode access times) data, but not for each time that a file is accessed. With this option enabled, `atime` data is written to disk only if the file has been modified since the `atime` data was last updated (mtime), or if the file was last accessed more than a certain amount of time ago (one day by default). `relatime` is required for automatic cache eviction to work properly.
   - `flock` –Enables file locking for your cache. If you don't want file locking enabled, use the `mount` command without `flock`.
4. Verify that the mount command was successful by listing the contents of the directory to which you mounted the cache, `/mnt` by using the following command.

```
$ ls /mnt
import-path  lustre
$
```

You can also use the `df` command.

```
$ df
Filesystem                     1K-blocks    Used  Available Use% Mounted on
devtmpfs                         1001808       0    1001808   0% /dev
tmpfs                            1019760       0    1019760   0% /dev/shm
tmpfs                            1019760     392    1019368   1% /run
tmpfs                            1019760       0    1019760   0% /sys/fs/cgroup
/dev/xvda1                       8376300 1263180    7113120  16% /
123.456.789.0@tcp:/mountname 3547698816   13824 3547678848   1% /mnt
tmpfs                             203956       0     203956   0% /run/user/1000
```

The results show Amazon File Cache mounted on `/mnt`.

Mounting from Amazon Elastic Container Service

You can access your cache from an Amazon Elastic Container Service (Amazon ECS) Docker container on an Amazon EC2 instance. You can do so by using either of the following options:

1. By mounting your cache from the Amazon EC2 instance that is hosting your Amazon ECS tasks, and exporting this mount point to your containers.
2. By mounting the cache directly inside your task container.

For more information about Amazon ECS, see What is Amazon Elastic Container Service? in the *Amazon Elastic Container Service Developer Guide*.

We recommend using option one (Mounting from an Amazon EC2 instance hosting Amazon ECS tasks) because it provides better resource use, especially if you start many containers (more than five) on the same EC2 instance, or if your tasks are short-lived (less than five minutes).

Use option two (Mounting from a Docker container), if you're unable to configure the EC2 instance, or if your application requires the container's flexibility.

> **Note**
>
> Mounting your cache on an AWS Fargate launch type isn't supported.

The following sections describe the procedures for each of the options for mounting your cache from an Amazon ECS container.

Topics

Mounting from an Amazon EC2 instance hosting Amazon ECS tasks

This procedure shows how you can configure an Amazon ECS on an EC2 instance to locally mount your cache. The procedure uses `volumes` and `mountPoints` container properties to share the resource and make this cache accessible to locally running tasks. For more information, see Launching an Amazon ECS container instance in the *Amazon Elastic Container Service Developer Guide*.

This procedure is for an Amazon ECS-Optimized Amazon Linux 2 AMI. If you're using another Linux distribution, see Installing the Lustre client.

To mount your cache from Amazon ECS on an EC2 instance

1. When launching Amazon ECS instances, either manually or using an Auto Scaling group, add the lines in the following code example to the end of the **User data** field. Replace the following items in the example:

   - Replace *cache_dns_name* with the actual cache's DNS name.
   - Replace *mountname* with the cache's mount name.
   - Replace *mountpoint* with the cache's mount point, which you need to create.

```
#!/bin/bash

...<existing user data>...

fsx_dnsname=cache_dns_name
fsx_mountname=mountname
fsx_mountpoint=mountpoint
amazon-linux-extras install -y lustre
mkdir -p "$fsx_mountpoint"
```

```
mount -t lustre ${fsx_dnsname}@tcp:/${fsx_mountname} ${fsx_mountpoint} -o
 relatime,flock
```

2. When creating your Amazon ECS tasks, add the following `volumes` and `mountPoints` container properties in the JSON definition. Replace *`mountpoint`* with the cache's mount point (such as `/mnt`).

```
{
    "volumes": [
           {
                 "host": {
                      "sourcePath": "mountpoint"
                 },
                 "name": "Lustre"
           }
    ],
    "mountPoints": [
           {
                 "containerPath": "mountpoint",
                 "sourceVolume": "Lustre"
           }
    ],
}
```

Mounting from a Docker container

The following procedure shows how you can configure an Amazon ECS task container to install the `lustre-client` package and mount your cache in it. The procedure uses an Amazon Linux (`amazonlinux`) Docker image, but a similar approach can work for other distributions.

To mount your cache from a Docker container

1. On your Docker container, install the `lustre-client` package and mount your cache with the `command` property. Replace the following items in the example:

   - Replace *`cache_dns_name`* with the actual file cache's DNS name.
   - Replace *`mountname`* with the cache's mount name.
   - Replace *`mountpoint`* with the cache's mount point.

```
"command": [
  "/bin/sh -c \"amazon-linux-extras install -y lustre; mount -t
 lustre cache_dns_name@tcp:/mountname mountpoint -o relatime,flock;\""
],
```

2. Add SYS_ADMIN capability to your container to authorize it to mount your cache, using the `linuxParameters` property.

```
"linuxParameters": {
  "capabilities": {
      "add": [
        "SYS_ADMIN"
      ]
   }
}
```

Mounting caches from on-premises or a peered Amazon VPC

You can access your cache in two ways. One is from Amazon EC2 instances located in an Amazon VPC that's peered to the cache's VPC. The other is from on-premises clients that are connected to your cache's VPC using Direct Connect or VPN.

Connect the client's VPC and your Amazon File Cache's VPC using either a VPC peering connection or a VPC transit gateway. When you use either option, Amazon EC2 instances that are in one VPC can access caches in another VPC, even if the VPCs belong to different accounts.

Before using the following the procedure, you must set up either a VPC peering connection or a VPC transit gateway.

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 RegionsAWS Regions. For more information about VPC peering, see What is VPC Peering? in the *Amazon VPC Peering Guide*.

You can mount your cache from outside its VPC using the IP address of its primary network interface. The primary network interface is the first network interface returned when you run the `aws fsx describe-file-caches` AWS Command Line Interface (AWS CLI) command. You can also get this IP address from the AWS Management Console.

To retrieve the IP address of the primary network interface for a cache

1. Open the AWS console at https://eusc-de-east-1.console.amazonaws-eusc.eu/fsx/.
2. In the navigation pane, choose **Caches**.
3. Choose your cache from the dashboard.
4. From the **Summary** details page, choose **Network & security**.
5. For **Network interface**, choose the ID for your primary elastic network interface. This takes you to the Amazon EC2 console.
6. On the **Details** tab, find the **Primary private IPv4 IP**. This is the IP address for your primary network interface.

> **Note**
>
> You can't use Domain Name System (DNS) name resolution when mounting an Amazon File Cache resource from outside the VPC it's associated with.

Mounting your cache automatically

You can update the `/etc/fstab` file in your Amazon EC2 instance after you connect to the instance for the first time so that it mounts your cache each time it reboots.

Using /etc/fstab to mount Amazon File Cache automatically

To automatically mount your cache directory when the Amazon EC2 instance reboots, you can use the `fstab` file. The `fstab` file contains information about the cache. The command `mount -a`, which runs during instance startup, mounts the caches listed in the `fstab` file.

> **Note**
>
> Before you can update the `/etc/fstab` file of your EC2 instance, verify that you've already created your cache. For more information, see Step 1: Create your cache in the Getting Started exercise.

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

1. Connect to your EC2 instance, and open the `/etc/fstab` file in an editor.
2. Add the following line to the `/etc/fstab` file.

   Mount Amazon File Cache to the directory that you created. Use these commands and replace the following:

   - Replace *`/mnt`* with the directory that you want to mount your cache to.
   - Replace *`cache_dns_name`* with the actual cache's DNS name.
   - Replace *`mountname`* with the cache's mount name. This mount name is returned in the `CreateFileCache` API operation response. It's also returned in the response of the **describe-file-caches** AWS CLI command, and the `DescribeFileCaches` API operation.

   ```
   cache_dns_name@tcp:/mountname /mnt lustre defaults,relatime,flock,_netdev,x-
   systemd.automount,x-systemd.requires=network.service 0 0
   ```

   > **Warning**
   >
   > Use the _netdev option that's used to identify network file systems, when mounting your cache automatically. If _netdev is missing, your EC2 instance might stop responding. This result is because network file systems must be initialized after the compute instance starts its networking.

3. Save the changes to the file.

Your EC2 instance is now configured to mount the cache whenever it restarts.

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

The fields in the line of code that you added to the `/etc/fstab` file do the following.

Field	Description
cache_dns_name `@tcp:/`	The DNS name for your cache, which identifies it. You can get this name from the console or programmatically from the AWS CLI or an AWS SDK.
mountname	The mount name for the cache. You can get this name from the console or programmatically from the AWS CLI using the **describe-file-caches** command or the AWS API or SDK using the `DescribeFileSystems` operation.
/mnt	The mount point for the cache on your EC2 instance.
`lustre`	The type of cache.
`mount options`	Mount options for the cache, presented as a comma-separated list of the following options: • `defaults` – This value tells the operating system to use the default mount options. You can list the default mount options after the cache has been mounted by viewing the output of the `mount` command. • `relatime` – Maintains `atime` (inode access times) data, but not for each time that a file is accessed. With this option enabled, `atime` data is written to disk only if the file has been modified since the `atime` data was last updated (mtime), or if the file was last accessed more than a certain amount of time ago (one day by default). `relatime` is required for automatic cache eviction to work properly.

Field	Description
	• `flock` – mounts your cache with file locking enabled. If you don't want file locking enabled, remove this mount option. • `_netdev` – The value tells the operating system that the cache resides on a device that requires network access. This option prevents the instance from mounting the cache until the network has been enabled on the client.
`x-systemd.automount,x-systemd.requires=network.service`	These options ensure that the auto mounter does not run until the network connectivity is online. **Note** For Ubuntu 22.04, use the `x-systemd.requires=systemd-networkd-wait-online.service` option instead of the `x-systemd.requires=network.service` option.
`0`	A value that indicates whether the cache should be backed up by dump. This value should be `0`.
`0`	A value that indicates the order in which `fsck` checks caches at boot. For caches, this value should be `0` to indicate that `fsck` should not run at startup.

Mounting specific filesets

By using the Lustre fileset feature, you can mount only a subset of the cache namespace, which is called a *fileset*. To mount a fileset of the cache on the client, specify the subdirectory path after the cache name. A fileset mount (also called a subdirectory mount) limits the cache namespace visibility on a specific client.

Example – Mount a Lustre fileset

1. Assume that you have a cache with the following directories:

```
team1/dataset1/
team2/dataset2/
```

2. You mount only the `team1/dataset1` fileset, making only this part of the cache locally visible on the client. Use these commands and replace the following items:

   - Replace *`cache_dns_name`* with the actual cache's DNS name.
   - Replace *`mountname`* with the cache's mount name. This mount name is returned in the `CreateFileCache` API operation response. It's also returned in the response of the **describe-file-caches** AWS CLI command, and the DescribeFileCaches API operation.

```
mount -t lustre -o relatime,flock cache_dns_name@tcp:/mountname/team1/dataset1 /mnt
```

When using the Lustre fileset feature, keep the following in mind:

- Before a cache directory can be mounted on a client, you must list the directory by running the `ls` command on a parent directory in the cache.
- There are no constraints preventing a client from remounting the cache using a different fileset, or no fileset at all.
- When using a fileset, some Lustre administrative commands requiring access to the `.lustre/` directory might not work, such as the `lfs fid2path` command.
- If you plan to mount several subdirectories from the same cache on the same host, keep in mind that this consumes more resources than a single mount point, and it could be more efficient to mount the cache root directory only once instead.

For more information about the Lustre fileset feature, see the *Lustre Operations Manual* on the Lustre documentation website.

Unmounting caches

Before you delete a cache, we recommend that you unmount it from every Amazon EC2 instance that it's connected to. You can unmount a cache on your Amazon EC2 instance by running the `umount` command on the instance itself. You can't unmount a cache through the AWS CLI, the

AWS Management Console, or through any of the AWS SDKs. To unmount a cache connected to an Amazon EC2 instance running Linux, use the `umount` command as follows:

```
umount /mnt
```

We recommend that you don't specify any other `umount` options and avoid setting any other `umount` options that are different from the defaults.

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

Example – Identify the mount status of an Amazon File Cache resource and unmount it

```
$ df -T
Filesystem Type 1K-blocks Used Available Use% Mounted on
cache_id.fsx.aws-region.amazonaws.com@tcp:/mountname /mnt 3547708416 61440 3547622400
 1% /mnt
      /dev/sda1 ext4 8123812 1138920 6884644 15% /
```

```
$ umount /mnt
```

```
$ df -T
```

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

Working with Amazon EC2 Spot Instances

Amazon File Cache can be used with EC2 Spot Instances to significantly lower your Amazon EC2 costs. A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Amazon EC2 can interrupt your Spot Instance when the Spot price exceeds your maximum price, when the demand for Spot Instances rises, or when the supply of Spot Instances decreases.

When Amazon EC2 interrupts a Spot Instance, it provides a Spot Instance interruption notice, which gives the instance a two-minute warning before Amazon EC2 interrupts it. For more information, see Spot Instances in the *Amazon EC2 User Guide*.

To verify that Amazon File Cache resources are unaffected by EC2 Spot Instance Interruptions, we recommend unmounting caches prior to terminating or hibernating EC2 Spot Instances. For more information, see Unmounting caches.

Handling Amazon EC2 Spot Instance interruptions

Amazon File Cache is built on the Lustre distributed file system where server and client instances cooperate to provide a performant and reliable file system. They maintain a distributed and coherent state across both client and server instances. Lustre servers delegate temporary access permissions to clients while they are actively doing I/O and caching file system data. Clients are expected to reply in a short period of time when servers request them to revoke their temporary access permissions. To protect the cache against misbehaving clients, servers can evict Lustre clients that do not respond after a few minutes. To avoid having to wait multiple minutes for a non-responding client to reply to the server request, it's important to cleanly unmount Lustre clients, especially before terminating EC2 Spot Instances.

EC2 Spot sends termination notices two minutes in advance before shutting down an instance. We recommend that you automate the process of cleanly unmounting Lustre clients before terminating EC2 Spot Instances.

Example – Script to cleanly unmount terminating EC2 Spot Instances

This example script cleanly unmounts terminating EC2 Spot Instances by doing the following:

- Watches for Spot termination notices.
- When it receives a termination notice:
  - Stops applications that are accessing the cache.
  - Unmounts the cache before the instance is terminated.

You can adapt the script as needed, especially for gracefully shutting down your application. For more information about best practices for handling Spot Instance interruptions, see Best practices for handling EC2 Spot Instance interruptions.

```
#!/bin/bash

# TODO: Specify below the FSx mount point you are using
*FSXPATH=/mnt*

cd /
```

```
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")
if [ "$?" -ne 0 ]; then
    echo "Error running 'curl' command" >&2
    exit 1
fi

# Periodically check for termination
while sleep 5
do

    HTTP_CODE=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s -w %{http_code} -o /dev/
null http://169.254.169.254/latest/meta-data/spot/instance-action)

    if [[ "$HTTP_CODE" -eq 401 ]] ; then
        # Refreshing Authentication Token
        TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 30")
        continue
    elif [[ "$HTTP_CODE" -ne 200 ]] ; then
        # If the return code is not 200, the instance is not going to be interrupted
        continue
    fi

    echo "Instance is getting terminated. Clean and unmount '$FSXPATH' ..."
    curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-
data/spot/instance-action
    echo

    # Gracefully stop applications accessing the filesystem
    #
    # TODO*: Replace with the proper command to stop your application if possible*

    # Kill every process still accessing Lustre filesystem
    echo "Kill every process still accessing Lustre filesystem..."
    fuser -kMm -TERM "${FSXPATH}"; sleep 2
    fuser -kMm -KILL "${FSXPATH}"; sleep 2

    # Unmount the cache
    if ! umount -c "${FSXPATH}"; then
        echo "Error unmouting '$FSXPATH'. Processes accessing it:" >&2
        lsof "${FSXPATH}"
```

```
        echo "Retrying..."
        continue
    fi

    # Start a graceful shutdown of the host
    shutdown now

done
```

Managing Amazon File Cache resources

Amazon File Cache provides a set of features that simplify the performance of administrative tasks. These include the ability to manage caches (such as creating and deleting caches), to manage storage quotas, and to set maintenance windows for performing routine software patching of the system.

You can administer your Amazon File Cache resources using the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS API, or AWS SDKs.

Topics

Managing caches

A cache is the primary Amazon File Cache resource. You can create, list, update, and delete an Amazon File Cache using the AWS Management Console, AWS Command Line Interface (AWS CLI), and API.

Creating caches

You can create an Amazon File Cache using the AWS Management Console, AWS CLI, or the AWS API.

> **Note**
>
> You can link your cache to data repositories only when you create the cache. You can link up to 8 data repositories of the same type (all Amazon S3 or all NFS0.

To create a cache (console)

- For information about how to create a cache from the console, see Step 1: Create your cache.

To create a cache (CLI)

- To create an Amazon File Cache, use the create-file-cache CLI command (or the equivalent CreateFileCache API operation). The following example creates an Amazon File Cache. Use the `--data-repository-associations` option to specify links to up to 8 data repositories. This example creates links to three Amazon S3 buckets.

```
fsx create-file-cache \
--storage-capacity 1200 \
--subnet-ids subnet-0156f2909ff21e78b \
--security-group-ids sg-0123456789abcdef3 \
--file-cache-type LUSTRE \
--file-cache-type-version 2.12 \
--lustre-configuration
 "DeploymentType=CACHE_1,PerUnitStorageThroughput=1000,MetadataConfiguration={StorageCapaci
 \
--data-repository-associations FileCachePath=/ns2/,DataRepositoryPath=s3://
s3cache2/fsx/ \
  FileCachePath=/ns3/,DataRepositoryPath=s3://s3cache3/fsx/ \
  FileCachePath=/ns4/,DataRepositoryPath=s3://s3cache4/fsx/
```

After successfully creating the cache, Amazon File Cache returns the cache description in JSON format, as shown in the following example.

```
{
    "FileCache": {
        "OwnerId": "1234567890120",
        "CreationTime": 1661841150.136,
        "FileCacheId": "fc-0123456789abcdef0",
        "FileCacheType": "LUSTRE",
        "FileCacheTypeVersion": "2.12",
        "Lifecycle": "CREATING",
        "StorageCapacity": 1200,
        "VpcId": "vpc-0222d95ad3328a351",
        "SubnetIds": [
            "sg-0123456789abcdef3"
        ],
        "DNSName": "fc-0123456789abcdef0.fsx.us-east-1.amazonaws.com",
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/71726ts-
r4sd-98gs-9ist-35w7tbs",
```

```
        "ResourceARN": "arn:aws:fsx:us-east-1:123456789012:file-cache/
fc-0123456789abcdef0",
        "Tags": [],
        "CopyTagsToDataRepositoryAssociations": false,
        "LustreConfiguration": {
            "PerUnitStorageThroughput": 1000,
            "DeploymentType": "CACHE_1",
            "MountName": "fmvszbmv",
            "WeeklyMaintenanceStartTime": "5:06:00",
            "MetadataConfiguration": {
                "StorageCapacity": 2400
            },
            "LogConfiguration": {
                "Level": "WARN_ERROR",
                "Destination": "arn:aws:logs:us-east-1:123456789012:log-group:/aws/fsx/
filecache/lustre:log-stream:datarepo_fc-0123456789abcdef0"
            }
        },
        "DataRepositoryAssociationIds": [
            "dra-04d15aa012bdcf0f8",
            "dra-036182e206928928c",
            "dra-07a25a1250582261e"
        ]
    }
}
```


Updating caches

You can update the **Weekly maintenance window** configuration that sets the day of the week and time that Amazon File Cache performs cache maintenance and updates. You can update the configuration of an Amazon File Cache resource using the AWS Management Console, the AWS CLI, and the AWS API.

To update a cache (console)

- For information about how to change the configuration of the weekly maintenance window, see Amazon File Cache maintenance windows.

To update a cache (CLI)

- To update the configuration of an Amazon File Cache, use the update-file-cache CLI command (or the equivalent UpdateFileCache API operation), as shown in the following example.

```
aws fsx update-file-cache \
    --file-cache-id fc-0123456789abcdef0 \
    --lustre-configuration WeeklyMaintenanceStartTime=1:01:30
```

Deleting caches

You can delete an Amazon File Cache using the AWS Management Console, the AWS CLI, and the AWS API and SDKs.

To delete a cache:

- **Using the console** – Follow the procedure described in Step 4: Clean up resources.
- **Using the CLI or API** – Use the delete-file-cache CLI command or the DeleteFileCache API operation. The following example shows how to use the CLI to delete an Amazon File Cache resource.

```
aws fsx delete-file-cache --file-cache-id fc-0123456789abcdef0
```

Viewing caches

You can view the details of your cache using the AWS Management Console, the AWS CLI, and the AWS API and SDKs.

To view a cache:

- **Using the console** – Choose a cache to view the cache detail page. The **Summary** panel shows the cache ID, lifecycle state, cache type, deployment type, storage type, storage capacity, metadata storage capacity, throughput per unit of storage, total throughput, Lustre version, Availability Zones, creation time, and mount name.

  The tabs provide detailed information about the cache's features, such as your linked data repositories.

- **Using the CLI or API** – Use the describe-file-caches CLI command or the DescribeFileCaches API operation. The following example shows how to use the CLI to return the description of a specific Amazon File Cache.

```
aws fsx describe-file-caches \
    --file-cache-ids fc-0123456789abcdef0
```

  To return descriptions of all existing caches, omit the `--file-cache-ids` option.

Amazon File Cache status

You can view the status of a cache by using the AWS Management Console, the AWS CLI command describe-caches, or the DescribeFileCaches API operation.

Cache status	Description
AVAILABLE	The cache has been successfully created and is available for use.
CREATING	A new cache is being created.
DELETING	An existing cache is being deleted.
UPDATING	The cache is undergoing a customer-initiated update.
FAILED	This status can mean either of the following: • The cache has failed and cannot be recovered. • The cache couldn't be created.

Storage quotas

You can create storage quotas for users and groups on Amazon File Cache resources. With storage quotas, you can limit the amount of disk space and the number of files that a user or a group

can consume. Storage quotas automatically track user and group-level usage so you can monitor consumption whether or not you choose to set storage limits.

Amazon File Cache enforces quotas and prevents users who have exceeded them from writing to the storage space. When users exceed their quotas, they must delete enough files to get under the quota limits so that they can write to the cache again.

Storage quotas are enforced only when your users write data to the cache directly. Amazon File Cache acts as a user with root access when loading data into the cache from a data repository, and thus bypasses quota enforcement. After data is imported, it counts toward the user or group based on the ownership set in the data repository, which can cause users or groups to exceed their block limits. If this happens, those users or groups need to free file space to be able to write to the cache again.

Topics

Quota enforcement

User and group quota enforcement is automatically enabled on all caches. You cannot disable quota enforcement.

Types of quotas

System administrators with AWS account root user credentials can create the following types of quotas:

- A *user quota* applies to an individual user. A user quota for a specific user can be different from the quotas of other users.
- A *group quota* applies to all users who are members of a specific group.
- A *block quota* limits the amount of disk space that a user or group can consume. You configure the storage size in kilobytes.

- An *inode quota* limits the number of files or directories that a user or group can create. You configure the maximum number of inodes as an integer.

> **Note**
>
> Default quotas and project quotas aren't supported.

If you set quotas for a particular user and a group, and the user is a member of that group, the user's data usage applies to both quotas. It's also limited by both quotas. If either quota limit is reached, the user is blocked from writing to the cache.

> **Note**
>
> Quotas that are set for the root user aren't enforced. Similarly, writing data as the root user using the `sudo` command bypasses enforcement of the quota.

Quota limits and grace periods

Amazon File Cache enforces user and group quotas as a hard limit or as a soft limit with a configurable grace period.

The hard limit is the absolute limit. If users exceed their hard limit, a block or inode allocation fails with a Disk quota exceeded message. Users who reach their quota hard limit must delete enough files or directories to get under the quota limit before they can write to the cache again. When a grace period is set, users can exceed the soft limit within the grace period if they're under the hard limit.

For soft limits, you configure a grace period in seconds. The soft limit must be smaller than the hard limit.

You can set different grace periods for inode and block quotas. You can also set different grace periods for a user quota and a group quota. When user and group quotas have different grace periods, the soft limit transforms to a hard limit after the grace period of either user or group quota elapses.

When users exceed a soft limit, Amazon File Cache allows them to continue exceeding their quota until the grace period has elapsed or until the hard limit is reached. After the grace period ends,

the soft limit converts to a hard limit, and users are blocked from any further write operations until their storage usage returns below the defined block quota or inode quota limits. Users don't receive a notification or warning when the grace period begins.

Setting and viewing quotas

You set storage quotas using Lustre file system `lfs` commands in your Linux terminal. The `lfs setquota` command sets quota limits, and the `lfs quota` command displays quota information.

For more information about Lustre quota commands, see the *Lustre Operations Manual* on the Lustre documentation website.

Setting user and group quotas

The syntax of the `setquota` command for setting user or group quotas is as follows.

```
lfs setquota {-u|--user|-g|--group} username|groupname
             [-b block_softlimit] [-B block_hardlimit]
             [-i inode_softlimit] [-I inode_hardlimit]
             /mount_point
```

Where:

- `-u` or `--user` specifies a user to set a quota for.
- `-g` or `--group` specifies a group to set a quota for.
- `-b` sets a block quota with a soft limit. `-B` sets a block quota with a hard limit. Both *`block_softlimit`* and *`block_hardlimit`* are expressed in kilobytes, and the minimum value is 1024 KB.
- `-i` sets an inode quota with a soft limit. `-I` sets an inode quota with a hard limit. Both *`inode_softlimit`* and *`inode_hardlimit`* are expressed in number of inodes, and the minimum value is 1024 inodes.
- *`mount_point`* is the directory that the cache was mounted on.

The following command sets a 5,000 KB soft block limit, an 8,000 KB hard block limit, a 2,000 soft inode limit, and a 3,000 hard inode limit quota for `user1` on the cache mounted to `/mnt`.

```
sudo lfs setquota -u user1 -b 5000 -B 8000 -i 2000 -I 3000 /mnt
```

The following command sets a 100,000 KB hard block limit for the group named `group1` on the cache mounted to `/mnt`.

```
sudo lfs setquota -g group1 -B 100000 /mnt
```

Setting grace periods

The default grace period is one week. You can adjust the default grace period for users and groups, using the following syntax.

```
lfs setquota -t {-u|-g}
             [-b block_grace]
             [-i inode_grace]
             /mount_point
```

Where:

- `-t` indicates that a grace time period will be set.
- `-u` sets a grace period for all users.
- `-g` sets a grace period for all groups.
- `-b` sets a grace period for block quotas. `-i` sets a grace period for inode quotas. Both *`block_grace`* and *`inode_grace`* are expressed in integer seconds or in the `XXwXXdXXhXXmXXs` format.
- *`mount_point`* is the directory that the cache was mounted on.

The following command sets grace periods of 1,000 seconds for user block quotas and 1 week and 4 days for user inode quotas.

```
sudo lfs setquota -t -u -b 1000 -i 1w4d /mnt
```

Viewing quotas

The `quota` command displays information about user quotas, group quotas, and grace periods.

View quota command	Quota information displayed
`lfs quota /`*`mount_point`*	General quota information (disk usage and limits) for the

View quota command	Quota information displayed
	user running the command and the user's primary group.
`lfs quota -u username /mount_point`	General quota informati on for a specific user. Users with AWS account root user credentials can run this command for any user, but non-root users can't run this command to get quota information about other users.
`lfs quota -u username -v /mount_point`	General quota information for a specific user and detailed quota statistics for each object storage target (OST) and metadata target (MDT). Users with AWS account root user credentials can run this command for any user, but non-root users can't run this command to get quota information about other users.
`lfs quota -g groupname  /mount_point`	General quota information for a specific group.
`lfs quota -t -u /mount_point`	Block and inode grace times for user quotas.
`lfs quota -t -g /mount_point`	Block and inode grace times for group quotas.

Quotas and linked data repositories

You can link your cache to an Amazon S3 or NFS data repository. For more information, see Linking your cache to a data repository.

You can optionally choose a specific folder or prefix within a linked data repository as an import path to your cache. Only data loaded into the cache from the linked data repository is applied towards the quota. The data of the entire repository is not counted against the quota limits.

File metadata in a linked data repository are imported into a folder with a structure matching the imported folder from the repository. These files count towards the inode quotas of the users and groups who own the files.

When a user performs an `hsm_restore` or lazy loads a file, the file's full size counts toward the block quota that's associated with the owner of the file. For example, if user A lazy loads a file that is owned by user B, the amount of storage and inode usage counts towards user B's quota. Similarly, when a user uses the `hsm_release` command on a file, the data is freed up from the block quotas of the user or group who owns the file.

Amazon File Cache acts as a user with root access when loading data into the cache from a data repository, and thus bypasses quota enforcement.

Tag your Amazon File Cache resources

To help you manage your cache resources such as caches and data repository associations (DRAs), you can assign your own metadata to each resource in the form of tags. Tags enable you to categorize your AWS resources in different ways, for example, by purpose, owner, or environment. This is useful when you have many resources of the same type—you can quickly identify a specific resource based on the tags that you've assigned to it. This topic describes tags and shows you how to create them.

Topics

Tag basics

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

Tags enable you to categorize your AWS resources in different ways, such as by purpose, owner, or environment. For example, you could define a set of tags for your account's Amazon File Cache resources which helps you to track each cache's owner and stack level.

We recommend that you devise a set of tag keys that meets your needs for each resource type. You can manage your resources better by using a consistent set of tag keys. You can search and filter the resources based on the tags you add.

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

If you're using the Amazon File Cache API, the AWS Command Line Interface (AWS CLI), or an AWS SDK, you can use the `TagResource` API action to apply tags to existing resources. Additionally, some resource-creating actions enable you to specify tags for a resource when the resource is created. If tags can't be applied during resource creation, we roll back the resource creation process. This ensures that resources are either created with tags or not created at all, and that no resources are kept untagged at any time. By tagging resources at the time of creation, you can eliminate the need to run custom tagging scripts after resource creation. For more information about enabling users to tag resources on creation, see Grant permission to tag resources during creation.

Tagging your resources

You can tag Amazon File Cache resources that exist in your account. If you're using the AWS Management Console, you can apply tags to resources by using the **Tags** tab on the relevant resource screen. When you create resources, you can apply the **Name** key with a value, and you can apply tags of your choice when creating a new cache. The console may organize resources according to the **Name** tag, but this tag doesn't have any semantic meaning to the Amazon File Cache service.

You can apply tag-based, resource-level permissions in your IAM policies to the Amazon File Cache API actions that support tagging on creation to implement granular control over the users and groups that can tag resources on creation. Your resources are properly secured from creation—tags are applied immediately to your resources, therefore any tag-based, resource-level permissions controlling the use of resources are immediately effective. Your resources can be tracked and reported on more accurately. You can enforce the use of tagging on new resources, and control which tag keys and values are set on your resources.

You can also apply resource-level permissions to the `TagResource` and `UntagResource` Amazon File Cache API actions in your IAM policies to control which tag keys and values are set on your existing resources.

For more information about tagging your resources for billing, see Using Cost Allocation Tags in the *AWS Billing User Guide*.

Tag restrictions

The following basic restrictions apply to tags:

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

You can't delete a resource based solely on its tags; you must specify the resource identifier. For example, to delete a cache that you tagged with a tag key called `DeleteMe`, you must use the `DeleteFileCache` action with the cache resource identifier, such as fc-1234567890abcdef0.

When you tag public or shared resources, the tags you assign are available only to your AWS account; no other AWS account will have access to those tags. For tag-based access control to

shared resources, each AWS account must assign its own set of tags to control access to the resource.

Permissions and tag

For more information about the permissions required to tag Amazon File Cache resources at creation, see Grant permission to tag resources during creation. For more information about using tags to restrict access to Amazon File Cache resources in IAM policies, see Using tags to control access to your Amazon File Cache resources.

Amazon File Cache maintenance windows

Amazon File Cache performs routine software patching for the Lustre software that it manages. The maintenance window is your opportunity to control what day and time of the week this software patching occurs.

Patching occurs infrequently, typically once every several weeks. Patching should require only a fraction of your 30-minute maintenance window. During these few minutes of time, your cache will be temporarily unavailable.

You choose the maintenance window during cache creation. If you have no time preference, then a 30-minute default window is assigned.

You can use the AWS Management Console, AWS Command Line Interface (AWS CLI), or one of the AWS SDKs to change the maintenance window for your caches.

To change the maintenance window using the console

1. Open the AWS Management Console at https://eusc-de-east-1.console.amazonaws-eusc.eu/fsx/.
2. Choose **Caches** in the navigation pane.
3. Choose the cache that you want to change the maintenance window for. The **Summary** details page appears.
4. Choose the **Maintenance** tab. The maintenance window **Settings** panel appears.
5. Choose **Update** and enter the new day and time that you want the maintenance window to start.
6. Choose **Save** to save your changes. The new maintenance start time is displayed in the **Settings** panel.

You can use the AWS CLI or one of the AWS SDKs to change the maintenance window for your cache using the UpdateFileCache operation.

Run the following command, replacing the `--file-cache-id` value with the ID for your cache, and the date and time with when you want to begin the window.

```
aws fsx update-file-cache --file-cache-id fc-01234567890123456 --lustre-configuration
 WeeklyMaintenanceStartTime=1:01:30
```

Monitoring Amazon File Cache

You can use the following automated monitoring tools to watch Amazon File Cache and report when something is wrong:

- **Monitoring using Amazon CloudWatch** – CloudWatch collects and processes raw data from Amazon File Cache into readable, near real-time metrics. You can create a CloudWatch alarm that sends an Amazon SNS message when the alarm changes state.
- **Log monitoring using AWS CloudTrail** – You can 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.

Topics

Monitoring with CloudWatch

You can monitor caches using Amazon CloudWatch, which collects and processes raw data from Amazon File Cache into readable, near real-time metrics. These statistics are retained 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 File Cache metric data is automatically sent to CloudWatch at 1-minute periods. For more information about CloudWatch, see What Is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*.

CloudWatch metrics are reported as raw *Bytes*. Bytes are not rounded to either a decimal or binary multiple of the unit.

CloudWatch metrics for an Amazon File Cache resource are organized into three categories:

All CloudWatch metrics for Amazon File Cache are published to the AWS/FSx namespace in CloudWatch. For each metric, Amazon File Cache emits a data point per disk per minute. To view aggregate cache details, you can use the Sum statistic. Note that the file servers behind your caches are spread across multiple disks.

Front-end I/O metrics

The following metrics report cache-level information on system read and write operations. All these metrics take one dimension (FileCacheId) and are published into the AWS/FSx namespace in CloudWatch.

Metric	Description
DataReadBytes	The number of bytes for cache read operations. The Sum statistic is the total number of bytes associated with read operations during the period. The Minimum statistic is the minimum number of bytes associated with read operations on a single disk. The Maximum statistic is the maximum number of bytes associate d with read operations on the disk. The Average statistic is the average number of bytes associated with read operations per disk. The SampleCount statistic is the number of disks. 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 Sum, Minimum, Maximum, and Average. • Count for SampleCount . Valid statistics: Sum, Minimum, Maximum, Average, SampleCount .
DataWriteBytes	The number of bytes for cache write operations. The Sum statistic is the total number of bytes associated with write operations. The Minimum statistic is the minimum number of bytes associated with write operations on a single disk. The Maximum

Metric	Description
	statistic is the maximum number of bytes associated with write operations on the disk. The Average statistic is the average number of bytes associated with write operations per disk. The SampleCount statistic is the number of disks. 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 Sum, Minimum, Maximum, and Average. • Count for SampleCount . Valid statistics: Sum, Minimum, Maximum, Average, SampleCount .
DataRead0 perations	The number of read operations. The Sum statistic is the total number of read operations. The Minimum statistic is the minimum number of read operations on a single disk. The Maximum statistic is the maximum number of read operation s on the disk. The Average statistic is the average number of read operations per disk. The SampleCount statistic is the number of disks. To calculate the average number of read operations (operations per second) for a period, divide the Sum statistic by the number of seconds in the period. Units: • Bytes for Sum, Minimum, Maximum, and Average. • Valid statistics: Sum, Minimum, Maximum, Average, SampleCount .

Metric	Description
DataWrite Operations	The number of write operations. The Sum statistic is the total number of write operations. The Minimum statistic is the minimum number of write operations on a single disk. The Maximum statistic is the maximum number write operations on the disk. The Average statistic is the average number of write operations per disk. The SampleCount statistic is the number of disks. To calculate the average number of write operations (operations per second) for a period, divide the Sum statistic by the number of seconds in the period. Units: • Bytes for Sum, Minimum, Maximum, and Average. • Count for SampleCount . Valid statistics: Sum, Minimum, Maximum, Average, SampleCount .

Metric	Description
MetadataOperations	The number of metadata operations. The Sum statistic is the count of metadata operations. The Minimum statistic is the minimum number of metadata operations per disk. The Maximum statistic is the maximum number of metadata operations per disk. The Average statistic is the average number of metadata operations per disk. The SampleCount statistic is the number of disks. To calculate the average number of metadata operations (operations per second) for a period, divide the Sum statistic by the number of seconds in the period. Units: • Count for Sum, Minimum, Maximum, Average, and SampleCount . Valid statistics: Sum, Minimum, Maximum, Average, SampleCount .

Backend I/O metrics

The following metrics report information on read and write operations between the cache and its linked data repository. The metric takes one dimension (FileCacheId) and is published into the AWS/FSx namespace in CloudWatch.

Metric	Description
RepositoryReadSuccess	The rates of files being read into the cache from its linked data repository.
RepositoryReadFailure	The rate of files failing to be read into the cache from its linked data repositories. If you observe a high rate of read failures, check if any data repository association (DRA) is unreachable or misconfigured, and if the link

Metric	Description
	between your VPC and an on-premises data store is saturated. If the link is not saturated, you can create a larger cache.
RepositoryWriteSuccess	The rate of files being written from the cache to its linked data repositories.
RepositoryMetadataReadSuccess	Understand the rate at which the workload is triggering the loading of file and directory metadata into the cache.
RepositoryMetadataReadFail	The rate of file and directory metadata failing to be read into the cache from its linked data repositories. If you have high metadata read failures, check the connectivity to the linked data repository and share the workload.
RepositoryMetadataReadNotFound	The rate at which cache is looking up the data repository for paths that don't exist.
RepositoryOperationsInProgress	The number of read and write operations (including metadata operations) on data repositories that are in progress.

Cache utilization metrics

The following metrics report cache-level storage information. These metrics take one dimension (`FileCacheId`) and are published into the `AWS/FSx` namespace in CloudWatch.

Metric	Description
FreeDataStorageCapacity	The amount of available data storage capacity. The `Sum` statistic is the total number of bytes available in the cache The `Minimum` statistic is the total number bytes available in the fullest disk. The `Maximum` statistic is the total number of bytes available in

Metric	Description
	the disk with the most remaining available storage. The `Average` statistic is the average number of bytes available per disk. The `SampleCount` statistic is the number of disks. Units: • Bytes for Sum, `Minimum`, `Maximum`. • Count for `SampleCount` . Valid statistics: Sum, `Minimum`, `Maximum`, `Average`, `SampleCount` .
`FreeMetadataStorageCapacity`	The amount of available metadata storage capacity. The Sum statistic is the total number of bytes of metadata storage available in the cache. The `Minimum` statistic is the total number of bytes available in the fullest metadata target (MDT) disk. The `Maximum` statistic is the total number of bytes available in the MDT disk with the most remaining storage. The `Average` statistic is the average number of bytes available per MDT disk. The `SampleCount` statistic is the number of MDT disks. Units: • Bytes for Sum, `Minimum`, `Maximum`, `Average`. • Count for `SampleCount` . Valid statistics: Sum, `Minimum`, `Maximum`, `Average`, `SampleCount` .

How to use Amazon File Cache metrics

The metrics reported by Amazon File Cache provide information that you can analyze in different ways. The following list shows some common uses for the metrics. These are suggestions to get you started, not a comprehensive list.

How Do I Determine ...	Relevant Metrics
My cache's throughput?	SUM(DataReadBytes + DataWriteBytes)/Period (in seconds)
My cache's IOPS?	Total IOPS = SUM(DataReadOperations + DataWriteOperations + MetadataOperations)/Period (in seconds)

Accessing CloudWatch metrics

You can see Amazon File Cache metrics for Amazon CloudWatch Logs 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.
2. In the navigation pane, choose **Metrics**.
3. Select the **FSx** namespace.
4. (Optional) To view a metric, type its name in the search field.
5. (Optional) To filter by dimension, select **FileCacheId**.

To access metrics from the AWS CLI

- Use the `list-metrics` command with the `--namespace "AWS/FSx"` 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 File Cache

You can create an Amazon CloudWatch Logs alarm that sends an Amazon SNS message when the alarm's state changes. An alarm watches a single metric over a time period that 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 because they are in a particular state; the state must have changed and been maintained for a specified number of periods.

The following procedures outline how to create alarms for an Amazon File Cache resource.

To set alarms using the CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at https://eusc-de-east-1.console.amazonaws-eusc.eu/cloudwatch/.
2. Choose **Create Alarm**. Doing this launches the Create Alarm Wizard.
3. Choose **FSx Metrics** and scroll through the Amazon File Cache metrics to locate the metric that you want to place an alarm on. To display only the Amazon File Cache metrics in this dialog box, search the cache ID of your cache. Choose the metric to create an alarm on, and choose **Next**.
4. In the **Conditions** section, choose the conditions that you want for the alarm, then choose **Next**.

   > **Note**
   >
   > Metrics may not be published during system maintenance. To prevent unnecessary and misleading alarm condition changes and to configure your alarms so that they are resilient to missing data points, see Configuring how CloudWatch alarms treat missing data in the *Amazon CloudWatch User Guide*.

5. If you want CloudWatch to send you an email when the alarm state is reached, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, choose an existing SNS topic. If you choose **Create topic**, you can set the name and email addresses for a new email subscription list. This list is saved and appears in this box for future alarms.

> **Note**
>
> If you use **Create topic** to create a new Amazon SNS topic, verify the email addresses before sending them 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 don't receive a notification.

6. Preview the alarm you're about to create in the **Alarm Preview** area. If it appears as expected, 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*.

Logging Amazon File Cache API calls with CloudTrail

Amazon File Cache is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon File Cache. CloudTrail captures all API calls for Amazon File Cache as events. Captured calls include calls from the Amazon File Cache console and from code calls to Amazon File Cache API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon File Cache. 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 File Cache. You can also determine 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 File Cache information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When API activity occurs in Amazon File Cache, 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 File Cache, 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 File Cache API calls are logged by CloudTrail. For example, calls to the `CreateFileCache` and `TagResource` operations generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps to 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 File Cache 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.

Security in Amazon File Cache

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between you and AWS. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You're also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.
- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the Amazon Web Services Cloud. AWS also provides you with services that you can securely use. 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 File Cache, see AWS Services in Scope by Compliance Program.

This documentation explains how to apply the shared responsibility model when using Amazon File Cache. The following topics show you how to configure File Cache to meet your security and compliance objectives. You'll also learn how to use other Amazon services that help you to monitor and secure your Amazon File Cache resources.

You can find a description of security considerations for working with Amazon File Cache in the following topics.

Topics

Data protection in Amazon File Cache

The AWS applies to data protection in Amazon File Cache. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Amazon File Cache or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Topics

- Data encryption in Amazon File Cache

Data encryption in Amazon File Cache

Amazon File Cache supports two forms of data encryption for caches, encryption of data at rest and encryption in transit. Encryption of data at rest is automatically enabled when creating an Amazon File Cache cache. Encryption of data in transit is automatically enabled when you access an Amazon File Cache cache from Amazon EC2 instances that support this feature.

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 cache and mounting your cache using encryption of data in transit.

Topics

Encrypting data at rest

Encryption of data at rest is automatically enabled when you create an Amazon File Cache resource through the AWS Management Console, the AWS CLI, or programmatically through the AWS 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 specify the AWS Key Management Service (AWS KMS) key to use for encrypting the data when you create an Amazon File Cache resource. For more information about creating a cache encrypted at rest using the console, see Step 1: Create your cache.

> **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.

For more information, see How Amazon File Cache uses AWS KMS.

How encryption at rest works

Data and metadata are automatically encrypted before being written to the cache. When you attach the encrypted volume to an instance, Amazon EC2 sends a `CreateGrant` request to AWS KMS, so that it can decrypt the data key. Amazon EC2 uses the plaintext data key in hypervisor memory to encrypt disk I/O to the volume. The plaintext data key persists in memory as long as the volume is attached to the instance.Similarly, as data and metadata are read, they are automatically decrypted before being presented to the application. These processes are handled transparently by Amazon File Cache, so you don't have to modify your applications.

Amazon File Cache uses industry-standard AES-256 encryption algorithm to encrypt cache data at rest. For more information, see Cryptography Basics in the *AWS Key Management Service Developer Guide*.

Encrypting data in transit

Encryption of data in transit is automatically enabled when you access an Amazon File Cache resource from compute instances that support encryption in transit. To learn which EC2 instances support encryption in transit, see Encryption in Transit in the *Amazon EC2 User Guide*.

Amazon File Cache encrypts traffic between the cache and your S3 data repositories using HTTPS (TLS). Amazon File Cache does not encrypt traffic between the cache and your NFSv3 data repositories, as the NFSv3 protocol does not encrypt data at the protocol level. For encryption in transit between your on-premises file systems and Amazon File Cache, you can use a virtual private network (VPN) to ensure encrypted data transfers between your VPC and your on-premises network. If you're using Direct Connect, you can use Site-to-Site VPN to combine one or more Direct Connect dedicated network connections with Site-to-Site VPN. Additionally, you can use MAC Security (MACsec) to encrypt your data from your corporate data center to the Direct Connect location. For more information, see Encryption in Direct Connect.

You can explicitly prohibit some or all users in your organization from creating Amazon File Cache resources linked to NFSv3 file systems using the `fsx:NfsDataRepositoryEncryptionInTransitEnabled` and `fsx:NfsDataRepositoryAuthenticationEnabled` context keys to control access to the `CreateFileCache` API action. The following is an example of an AWS Organizations Service control policy (SCP) that prohibits creation of a Amazon File Cache resource linked to an NFSv3 data repository:

For more information about IAM condition keys, see Policy condition keys for File Cache.

How Amazon File Cache uses AWS KMS

Amazon File Cache integrates with AWS Key Management Service (AWS KMS) for key management for encrypting data at rest. File Cache uses AWS KMS keys to encrypt your cache in the following way:

- **Encrypting data at rest** – Amazon File Cache encrypts data automatically before it is written to the cache, and automatically decrypts data as it is read. Data is encrypted using an XTS-AES-256 block cipher. You choose the KMS key that's used to encrypt and decrypt data, either the AWS managed key for Amazon File Cache, `aws/fsx`, or a customer managed key. You can enable, disable, or revoke grants on this KMS key. This KMS key can be one of the two following types:
  - **AWS managed key for Amazon File Cache** – This is the default KMS key, `aws/fsx`. You're not charged to create and store a KMS key, but there are usage charges. For more information, see AWS Key Management Service pricing.
  - **Customer managed key** – This is the most flexible KMS key to use, because you can configure its key policies and grants for multiple users or services. For more information on creating customer managed keys, see Creating keys in the *AWS Key Management Service Developer Guide.*

    If you use a customer managed key as your KMS key for file data encryption and decryption, you can enable key rotation. When you enable key rotation, AWS KMS automatically rotates your key once per year. Additionally, with a customer managed key, you can choose when to disable, re-enable, delete, or revoke access to your customer managed key at any time.

> **Important**
>
> Amazon File Cache accepts only symmetric encryption KMS keys. You can't use asymmetric KMS keys with Amazon File Cache.

Amazon File Cache key policies for AWS KMS

Key policies are the primary way to control access to KMS keys. 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 File Cache for encrypted at rest caches:

- **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 KMS key, without exposing the plaintext of the data on the client side. The data is first decrypted and then re-encrypted. This permission is included in the default key policy.
- **kms:GenerateDataKeyWithoutPlaintext** – (Required) Returns a data encryption key encrypted under a KMS key. This permission is included in the default key policy under **kms:GenerateDataKey***.
- **kms:CreateGrant** – (Required) Adds a grant to a key to specify who can use the key and under what conditions. Grants are alternate permission mechanisms to key policies. For more information on grants, see 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 KMS 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 cache, this permission populates the list to select the KMS key. We recommend using this permission to provide the best user experience. This permission is included in the default key policy.

Internetwork traffic privacy

This topic describes how Amazon File Cache secures connections from the service to other locations.

Traffic between Amazon File Cache and on-premises clients

You have two connectivity options between your private network and AWS:

- An AWS Site-to-Site VPN connection. For more information, see What is AWS Site-to-Site VPN?
- An AWS Direct Connect connection. For more information, see What is AWS Direct Connect?

You can access Amazon File Cache over the network to reach AWS-published API operations for performing administrative tasks and Lustre ports to interact with the cache.

Access to Amazon File Cache by using the network is through AWS-published APIs. Clients must support Transport Layer Security (TLS) 1.2 and later. We require TLS 1.2 and recommend TLS 1.3. Clients must also support cipher suites with Perfect Forward Secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Most modern systems such as Java 7 and later support these modes. Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (STS) to generate temporary security credentials to sign requests.

API traffic between AWS resources in the same Region

An Amazon Virtual Private Cloud (Amazon VPC) endpoint for Amazon File Cache is a logical entity within a VPC that allows connectivity only to Amazon File Cache. The Amazon VPC routes API requests to Amazon File Cache and routes responses back to the VPC. For more information, see VPC Endpoints in the *Amazon VPC User Guide*.

Identity and Access Management for Amazon File Cache

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use File Cache resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Amazon File Cache identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Amazon File Cache works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for Amazon File Cache)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon File Cache works with IAM

Before you use IAM to manage access to File Cache, learn what IAM features are available to use with File Cache.

IAM features you can use with Amazon File Cache

IAM feature	File Cache support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Forward access sessions (FAS)	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how Amazon File Cache and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for File Cache

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for File Cache

To view examples of File Cache identity-based policies, see Identity-based policy examples for Amazon File Cache.

Resource-based policies within File Cache

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for File Cache

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of File Cache actions, see Actions defined by Amazon File Cache in the *Service Authorization Reference*.

Policy actions in File Cache use the following prefix before the action:

```
fsx
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "fsx:action1",
      "fsx:action2"
         ]
```

To view examples of File Cache identity-based policies, see Identity-based policy examples for Amazon File Cache.

Policy resources for File Cache

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of File Cache resource types and their ARNs, see Resources defined by Amazon File Cache in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon File Cache.

To view examples of File Cache identity-based policies, see Identity-based policy examples for Amazon File Cache.

Policy condition keys for File Cache

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of File Cache condition keys, see Condition keys for Amazon File Cache in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon File Cache.

To view examples of File Cache identity-based policies, see Identity-based policy examples for Amazon File Cache.

ACLs in File Cache

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with File Cache

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with File Cache

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Forward access sessions for File Cache

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for File Cache

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon File Cache functionality. Edit service roles only when Amazon File Cache provides guidance to do so.

Service-linked roles for File Cache

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon File Cache

By default, users and roles don't have permission to create or modify File Cache resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by File Cache, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon File Cache in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the File Cache console
- Allow users to view their own permissions

Policy best practices

Identity-based policies determine whether someone can create, access, or delete File Cache resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies

that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the File Cache console

To access the Amazon File Cache console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the File Cache resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the Amazon File Cache console, also attach the Amazon File Cache `AmazonFSxConsoleFullAccess` or `AmazonFSxConsoleReadOnlyAccess` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

AWS managed policies for Amazon File Cache

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AmazonFSxServiceRolePolicy

Allows Amazon FSx to manage AWS resources on your behalf. See Using service-linked roles for Amazon FSx to learn more.

AWS managed policy: AmazonFSxDeleteServiceLinkedRoleAccess

You can't attach `AmazonFSxDeleteServiceLinkedRoleAccess` to your IAM entities. This policy is linked to a service and used only with the service-linked role for that service. You cannot attach, detach, modify, or delete this policy. For more information, see Using service-linked roles for Amazon FSx.

This policy grants administrative permissions that allow Amazon FSx to delete its Service Linked Role for Amazon S3 access, used only by Amazon FSx for Lustre.

Permissions details

This policy includes permissions in `iam` to allow Amazon FSx to view, delete, and view the deletion status for the FSx Service Linked Roles for Amazon S3 access.

To view the permissions for this policy, see AmazonFSxDeleteServiceLinkedRoleAccess in the AWS Managed Policy Reference Guide.

AWS managed policy: AmazonFSxFullAccess

You can attach AmazonFSxFullAccess to your IAM entities. Amazon FSx also attaches this policy to a service role that allows Amazon FSx to perform actions on your behalf.

Provides full access to Amazon FSx and access to related AWS services.

Permissions details

This policy includes the following permissions.

- `fsx` – Allows principals full access to perform all Amazon FSx actions, except for `BypassSnaplockEnterpriseRetention`.
- `ds` – Allows principals to view information about the Directory Service directories.
- `ec2`
  - Allows principals to create tags under the specified conditions.
  - To provide enhanced security group validation of all security groups that can be used with a VPC.
- `iam` – Allows principles to create an Amazon FSx service linked role on the user's behalf. This is required so that Amazon FSx can manage AWS resources on the user's behalf.
- `firehose` – Allows principals to write records to a Amazon Data Firehose. This is required so that users can monitor FSx for Windows File Server file system access by sending audit access logs to Firehose.
- `logs` – Allows principals to create log groups, log streams, and write events to log streams. This is required so that users can monitor FSx for Windows File Server file system access by sending audit access logs to CloudWatch Logs.

To view the permissions for this policy, see AmazonFSxFullAccess in the AWS Managed Policy Reference Guide.

AWS managed policy: AmazonFSxConsoleFullAccess

You can attach the `AmazonFSxConsoleFullAccess` policy to your IAM identities.

This policy grants administrative permissions that allow full access to Amazon FSx and access to related AWS services via the AWS Management Console.

Permissions details

This policy includes the following permissions.

- `fsx` – Allows principals to perform all actions in the Amazon FSx management console, except for `BypassSnaplockEnterpriseRetention`.
- `cloudwatch` – Allows principals to view CloudWatch Alarms and metrics in the Amazon FSx management console.
- `ds` – Allows principals to list information about an Directory Service directory.
- `ec2`
  - Allows principals to create tags on route tables, list network interfaces, route tables, security groups, subnets and the VPC associated with an Amazon FSx file system.
  - Allows principals to provide enhanced security group validation of all security groups that can be used with a VPC.
  - Allows principals to view the elastic network interfaces associated with an Amazon FSx file system.
- `kms` – Allows principals to list aliases for AWS Key Management Service keys.
- `s3` – Allows principals to list some or all of the objects in an Amazon S3 bucket (up to 1000).
- `iam` – Grants permission to create a service linked role that allows Amazon FSx to perform actions on the user's behalf.

To view the permissions for this policy, see AmazonFSxConsoleFullAccess in the AWS Managed Policy Reference Guide.

AWS managed policy: AmazonFSxConsoleReadOnlyAccess

You can attach the `AmazonFSxConsoleReadOnlyAccess` policy to your IAM identities.

This policy grants read-only permissions to Amazon FSx and related AWS services so that users can view information about these services in the AWS Management Console.

Permissions details

This policy includes the following permissions.

- `fsx` – Allows principals to view information about Amazon FSx file systems, including all tags, in the Amazon FSx Management Console.
- `cloudwatch` – Allows principals to view CloudWatch Alarms and metrics in the Amazon FSx Management Console.
- `ds` – Allows principals to view information about an Directory Service directory in the Amazon FSx Management Console.
- `ec2`
  - Allows principals to view network interfaces, security groups, subnets and the VPC associated with an Amazon FSx file system in the Amazon FSx Management Console.
  - Allows principals to provide enhanced security group validation of all security groups that can be used with a VPC.
  - Allows principals to view the elastic network interfaces associated with an Amazon FSx file system.
- `kms` – Allows principals to view aliases for AWS Key Management Service keys in the Amazon FSx Management Console.
- `log` – Allows principals to describe the Amazon CloudWatch Logs log groups associated with the account making the request. This is required so that principals can view the existing file access auditing configuration for an FSx for Windows File Server file system.
- `firehose` – Allows principals to describe the Amazon Data Firehose delivery streams associated with the account making the request. This is required so that principals can view the existing file access auditing configuration for an FSx for Windows File Server file system.

To view the permissions for this policy, see AmazonFSxConsoleReadOnlyAccess in the AWS Managed Policy Reference Guide.

AWS managed policy: AmazonFSxReadOnlyAccess

You can attach the `AmazonFSxReadOnlyAccess` policy to your IAM identities.

- `fsx` – Allows principals to view information about Amazon FSx file systems, including all tags, in the Amazon FSx Management Console.
- `ec2` – To provide enhanced security group validation of all security groups that can be used with a VPC.

To view the permissions for this policy, see AmazonFSxReadOnlyAccess in the AWS Managed Policy Reference Guide.

Amazon FSx updates to AWS managed policies

View details about updates to AWS managed policies for Amazon FSx since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon FSx Document history page.

Change	Description	Date
AmazonFSxServiceRolePolicy – Update to an existing policy	Amazon FSx added a new permission, `ec2:AssignIpv6Addresses` that allows principals to assign IPv6 addresses to customer network interfaces that have an `AmazonFSx.FileSystemId` tag.	July 22, 2025
AmazonFSxServiceRolePolicy – Update to an existing policy	Amazon FSx added a new permission, `ec2:UnassignIpv6Addresses` that allows principals to unassign IPv6 addresses from customer network interfaces that have an `AmazonFSx.FileSystemId` tag.	July 22, 2025
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added a new permission, `fsx:CreateAndAttachS3AccessPoint` that allows principals to create an S3 access point and attach it to an FSx volume.	June 25, 2025

Change	Description	Date
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added a new permission, `fsx:DescribeS3AccessPointAttachments` that allows principals to list all S3 access points in an AWS account in an AWS Region.	June 25, 2025
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added a new permission, `fsx:DetachAndDeleteS3AccessPoint` that allows principals to delete an S3 access point.	June 25, 2025
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added a new permission, `fsx:CreateAndAttachS3AccessPoint` that allows principals to create an S3 access point and attach it to an FSx volume.	June 25, 2025
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added a new permission, `fsx:DescribeS3AccessPointAttachments` that allows principals to list all S3 access points in an AWS account in an AWS Region.	June 25, 2025
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added a new permission, `fsx:DetachAndDeleteS3AccessPoint` that allows principals to delete an S3 access point.	June 25, 2025

Change	Description	Date
AmazonFSxConsoleReadOnlyAccess – Update to an existing policy	Amazon FSx added new permission, `ec2:DescribeNetworkInterfaces` that allows principals to view the elastic network interfaces associated with their file system.	February 25, 2025
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permission, `ec2:DescribeNetworkInterfaces` that allows principals to view the elastic network interfaces associated with their file system.	February 07, 2025
AmazonFSxServiceRolePolicy – Update to an existing policy	Amazon FSx added new permission, `ec2:GetSecurityGroupsForVpc` that allows principals to provide enhanced security group validation of all security groups that can be used with a VPC.	January 9, 2024
AmazonFSxReadOnlyAccess – Update to an existing policy	Amazon FSx added new permission, `ec2:GetSecurityGroupsForVpc` that allows principals to provide enhanced security group validation of all security groups that can be used with a VPC.	January 9, 2024

Change	Description	Date
AmazonFSxConsoleReadOnlyAccess – Update to an existing policy	Amazon FSx added new permission, `ec2:GetSecurityGroupsForVpc` that allows principals to provide enhanced security group validation of all security groups that can be used with a VPC.	January 9, 2024
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permission, `ec2:GetSecurityGroupsForVpc` that allows principals to provide enhanced security group validation of all security groups that can be used with a VPC.	January 9, 2024
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permission, `ec2:GetSecurityGroupsForVpc` that allows principals to provide enhanced security group validation of all security groups that can be used with a VPC.	January 9, 2024
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permission to enable users to perform cross-region and cross-account data replication for FSx for OpenZFS file systems.	December 20, 2023

Change	Description	Date
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permission to enable users to perform cross-region and cross-account data replicati on for FSx for OpenZFS file systems.	December 20, 2023
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permission to enable users to perform on-demand replicati on of volumes for FSx for OpenZFS file systems.	November 26, 2023
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permission to enable users to perform on-demand replicati on of volumes for FSx for OpenZFS file systems.	November 26, 2023
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permissions to enable users to view, enable, and disable shared VPC support for FSx for ONTAP Multi-AZ file systems.	November 14, 2023
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permissions to enable users to view, enable, and disable shared VPC support for FSx for ONTAP Multi-AZ file systems.	November 14, 2023

Change	Description	Date
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow Amazon FSx to manage network configurations for FSx for OpenZFS Multi-AZ file systems.	August 9, 2023
AWS managed policy: AmazonFSxServiceRolePolicy – Update to an existing policy	Amazon FSx modified the existing `cloudwatch:PutMetricData` permission so that Amazon FSx publishes CloudWatch metrics to the `AWS/FSx` namespace.	July 24, 2023
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx updated the policy to remove the `fsx:*` permission and add specific `fsx` actions.	July 13, 2023
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx updated the policy to remove the `fsx:*` permission and add specific `fsx` actions.	July 13, 2023
AmazonFSxConsoleReadOnlyAccess – Update to an existing policy	Amazon FSx added new permissions to enable users to view enhanced performance metrics and recommended actions for FSx for Windows File Server file systems in the Amazon FSx console.	September 21, 2022

Change	Description	Date
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permissions to enable users to view enhanced performance metrics and recommended actions for FSx for Windows File Server file systems in the Amazon FSx console.	September 21, 2022
AmazonFSxReadOnlyAccess – Started tracking policy	This policy grants read-only access to all Amazon FSx resources and any tags associated with them.	February 4, 2022
AmazonFSxDeleteServiceLinkedRoleAccess – Started tracking policy	This policy grants administrative permissions that allow Amazon FSx to delete its Service Linked Role for Amazon S3 access.	January 7, 2022
AmazonFSxServiceRolePolicy – Update to an existing policy	Amazon FSx added new permissions to allow Amazon FSx to manage network configurations for Amazon FSx for NetApp ONTAP file systems.	September 2, 2021
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow Amazon FSx to create tags on EC2 route tables for scoped down calls.	September 2, 2021

Change	Description	Date
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow Amazon FSx to create Amazon FSx for NetApp ONTAP Multi-AZ file systems.	September 2, 2021
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow Amazon FSx to create tags on EC2 route tables for scoped down calls.	September 2, 2021
AmazonFSxServiceRolePolicy – Update to an existing policy	Amazon FSx added new permissions to allow Amazon FSx to describe and write to CloudWatch Logs log streams. This is required so that users can view file access audit logs for FSx for Windows File Server file systems using CloudWatch Logs.	June 8, 2021
AmazonFSxServiceRolePolicy – Update to an existing policy	Amazon FSx added new permissions to allow Amazon FSx to describe and write to Amazon Data Firehose delivery streams. This is required so that users can view file access audit logs for an FSx for Windows File Server file system using Amazon Data Firehose.	June 8, 2021

Change	Description	Date
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow principals to describe and create CloudWatch Logs log groups, log streams, and write events to log streams. This is required so that principals can view file access audit logs for FSx for Windows File Server file systems using CloudWatch Logs.	June 8, 2021
AmazonFSxFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow principals to describe and write records to a Amazon Data Firehose. This is required so that users can view file access audit logs for an FSx for Windows File Server file system using Amazon Data Firehose.	June 8, 2021

Change	Description	Date
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow principals to describe the Amazon CloudWatch Logs log groups associated with the account making the request. This is required so that principals can choose an existing CloudWatch Logs log group when configuring file access auditing for an FSx for Windows File Server file system.	June 8, 2021
AmazonFSxConsoleFullAccess – Update to an existing policy	Amazon FSx added new permissions to allow principals to describe the Amazon Data Firehose delivery streams associated with the account making the request. This is required so that principals can choose an existing Firehose delivery stream when configuring file access auditing for an FSx for Windows File Server file system.	June 8, 2021

Change	Description	Date
AmazonFSxConsoleReadOnlyAccess – Update to an existing policy	Amazon FSx added new permissions to allow principals to describe the Amazon CloudWatch Logs log groups associated with the account making the request. This is required so that principals can view the existing file access auditing configuration for an FSx for Windows File Server file system.	June 8, 2021
AmazonFSxConsoleReadOnlyAccess – Update to an existing policy	Amazon FSx added new permissions to allow principals to describe the Amazon Data Firehose delivery streams associated with the account making the request. This is required so that principals can view the existing file access auditing configuration for an FSx for Windows File Server file system.	June 8, 2021
Amazon FSx started tracking changes	Amazon FSx started tracking changes for its AWS managed policies.	June 8, 2021

Troubleshooting Amazon File Cache identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with File Cache and IAM.

Topics

I am not authorized to perform an action in File Cache

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `fsx:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 fsx:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `fsx:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to File Cache.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in File Cache. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my File Cache resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether File Cache supports these features, see How Amazon File Cache works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Using tags with Amazon File Cache

You can use tags to control access to Amazon File Cache resources and to implement attribute-based access control (ABAC). Users need to have permission to apply tags to Amazon File Cache resources during creation.

Grant permission to tag resources during creation

Some resource-creating Amazon File Cache API actions enable you to specify tags when you create the resource. You can use resource tags to implement attribute-based access control (ABAC). For more information, see What is ABAC for AWS in the *IAM User Guide*.

To enable users to tag resources on creation, they must have permissions to use the action that creates the resource, such as `fsx:CreateFileCache`. If tags are specified in the resource-creating action, Amazon performs additional authorization on the `fsx:TagResource` action to verify if users have permissions to create tags. Therefore, users must also have explicit permissions to use the `fsx:TagResource` action.

The following example demonstrates a policy that allows users to create caches and apply tags to them during creation in a specific AWS account.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "fsx:CreateFileCache",
         "fsx:TagResource"
      ],
      "Resource": [
         "arn:aws:fsx:region:account-id:file-cache/*"
      ]
    }
  ]
}
```

The `fsx:TagResource` action is only evaluated if tags are applied during the resource-creating action. Therefore, a user that has permissions to create a resource (assuming there are no tagging conditions) does not require permissions to use the `fsx:TagResource` action if no tags are specified in the request. However, if the user attempts to create a resource with tags, the request fails if the user does not have permissions to use the `fsx:TagResource` action.

For more information about tagging Amazon FSx resources, see Tag your Amazon File Cache resources. For more information about using tags to control access to FSx resources, see Using tags to control access to your Amazon File Cache resources.

Using tags to control access to your Amazon File Cache resources

To control access to Amazon FSx resources and actions, you can use AWS Identity and Access Management (IAM) policies based on tags. You can provide the control in two ways:

1. Control access to Amazon FSx resources based on the tags on those resources.
2. Control what tags can be passed in an IAM request condition.

For information about how to use tags to control access to AWS resources, see Controlling access using tags in the *IAM User Guide*. For more information about tagging Amazon File Cache resources at creation, see Grant permission to tag resources during creation. For more information about tagging resources, see Tag your Amazon File Cache resources.

Controlling access based on tags on a resource

To control what actions a user or role can perform on an Amazon FSx resource, you can use tags on the resource. For example, you might want to allow or deny specific API operations on a cache resource based on the key-value pair of the tag on the resource.

Example policy – Create a cache on when providing a specific tag

This policy allows the user to create a cache only when they tag it with a specific tag key value pair, in this example, `key=Department, value=Finance`.

```
{
    "Effect": "Allow",
    "Action": [
        "fsx:CreateFileCache",
        "fsx:TagResource"
    ],
    "Resource": "arn:aws:fsx:region:account-id:file-system/*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Department": "Finance"
        }
    }
}
```

Example policy – Delete caches with specific tags

This policy allows a user to delete only caches that are tagged with `Department=Finance`. If they create a final backup, then it must be tagged with `Department=Finance`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "fsx:DeleteFileCache"
            ],
            "Resource": "arn:aws:fsx:us-east-1:111122223333:file-system/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Department": "Finance"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "fsx:TagResource"
            ],
            "Resource": "arn:aws:fsx:us-east-1:111122223333:backup/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Department": "Finance"
                }
            }
        }
    ]
}
```

Using service-linked roles for Amazon FSx

Amazon FSx uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Amazon FSx. Service-linked roles are

predefined by Amazon FSx and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon FSx easier because you don't have to manually add the necessary permissions. Amazon FSx defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon FSx can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your Amazon FSx resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Amazon FSx

Amazon FSx uses the service-linked role named **AWSServiceRoleForAmazonFSx** – which performs certain actions in your account, like creating Elastic Network Interfaces for your caches in your VPC.

For updates to this policy, see AmazonFSxServiceRolePolicy.

The AmazonFSxServiceRolePolicy permissions policy allows Amazon FSx to complete the following actions on all applicable AWS resources:

> **Note**
>
> The AWSServiceRoleForAmazonFSx is used by all Amazon FSx file system types; some of the listed permissions are not applicable to Amazon File Cache.

- `ds` – Allows Amazon FSx to view, authorize, and unauthorize applications in your Directory Service directory.
- `ec2` – Allows Amazon FSx to do the following:
  - View, create, and disassociate network interfaces associated with an Amazon FSx file system.
  - View one or more Elastic IP addresses associated with an Amazon FSx file system.
  - View Amazon VPCs, security groups, and subnets associated with an Amazon FSx file system.

- Assign IPv6 addresses to customer network interfaces that have an `AmazonFSx.FileSystemId` tag.
- Unassign IPv6 addresses from customer network interfaces that have an `AmazonFSx.FileSystemId` tag.
- To provide enhanced security group validation of all security groups that can be used with a VPC.
- Create a permission for an AWS-authorized user to perform certain operations on a network interface.

- `cloudwatch` – Allows Amazon FSx to publish metric data points to CloudWatch under the AWS/FSx namespace.
- `route53` – Allows Amazon FSx to associate an Amazon VPC with a private hosted zone.
- `logs` – Allows Amazon FSx to describe and write to CloudWatch Logs log streams. This is so that users can send file access audit logs for an FSx for Windows File Server file system to a CloudWatch Logs stream.
- `firehose` – Allows Amazon FSx to describe and write to Amazon Data Firehose delivery streams. This is so that users can publish the file access audit logs for an FSx for Windows File Server file system to an Amazon Data Firehose delivery stream.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CreateFileSystem",
            "Effect": "Allow",
            "Action": [
                "ds:AuthorizeApplication",
                "ds:GetAuthorizedApplicationDetails",
                "ds:UnauthorizeApplication",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeAddresses",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
```

```
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVPCs",
                "ec2:DisassociateAddress",
                "ec2:GetSecurityGroupsForVpc",
                "route53:AssociateVPCWithHostedZone"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PutMetrics",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "AWS/FSx"
                }
            }
        },

        {
            "Sid": "TagResourceNetworkInterface",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "AmazonFSx.FileSystemId"
                }
            }
        },
        {
```

```
            "Sid": "ManageNetworkInterface",
            "Effect": "Allow",
            "Action": [
                "ec2:AssignPrivateIpAddresses",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:UnassignPrivateIpAddresses"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AmazonFSx.FileSystemId": "false"
                }
            }
        },
        {
            "Sid": "ManageRouteTable",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:ReplaceRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:route-table/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonFSx": "ManagedByAmazonFSx"
                }
            }
        },
        {
            "Sid": "PutCloudWatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/fsx/*"
        },
        {
```

```
            "Sid": "ManageAuditLogs",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/aws-fsx-*"
        }
    ]
}
```

Any updates to this policy are described in Amazon FSx updates to AWS managed policies.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

Creating a service-linked role for Amazon FSx

You don't need to manually create a service-linked role. When you create a cache in the AWS Management Console, the IAM CLI, or the IAM API, Amazon FSx creates the service-linked role for you.

> **Important**
>
> This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role. To learn more, see A New Role Appeared in My IAM Account.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create a cache, Amazon FSx creates the service-linked role for you again.

Editing a service-linked role for Amazon FSx

Amazon FSx does not allow you to edit the AWSServiceRoleForAmazonFSx 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 FSx

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 delete all of your file caches before you can manually delete the service-linked role.

> **Note**
>
> If the Amazon FSx 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 manually delete the service-linked role using IAM

Use the IAM console, the IAM CLI, or the IAM API to delete the AWSServiceRoleForAmazonFSx service-linked role. For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

Supported regions for Amazon FSx service-linked roles

Amazon FSx supports using service-linked roles in all of the regions where the service is available. For more information, see AWS Regions and Endpoints.

Cache access control with Amazon VPC

A cache is accessible through an elastic network interface that resides in the virtual private cloud (VPC) based on the Amazon VPC service that you associate with your cache. You access your cache through its DNS name, which maps to the cache's network interface. Only resources within the associated VPC, or a peered VPC, can access your cache's network interface. For more information, see What is Amazon VPC? in the *Amazon VPC User Guide.*

> **Warning**
>
> You must not modify or delete the Amazon File Cache elastic network interface. Modifying or deleting the network interface can cause a permanent loss of connection between your VPC and your cache.

Amazon VPC security groups

To further control network traffic going through your cache's network interface within your VPC, use security groups to limit access. A *security group* acts as a virtual firewall to control the traffic for its associated resources. In this case, the associated resource is your cache's network interface. You also use VPC security groups to control network traffic for your clients.

Controlling access using inbound and outbound rules

To use a security group to control access to your cache and clients, add the inbound rules to control incoming traffic and outbound rules to control the outgoing traffic from your cache and clients. Make sure to have the right network traffic rules in your security group to map your cache to a folder on your supported compute instance.

For more information about security group rules, see Security Group Rules in the *Amazon EC2 User Guide.*

To create a security group for your cache

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2.
2. In the navigation pane, choose **Security Groups**.
3. Choose **Create security group**.
4. Specify a name and description for the security group.
5. For **VPC**, choose the VPC associated with your cache to create the security group within that VPC.
6. Choose **Create** to create the security group.

Next, add inbound rules to the security group that you just created to enable traffic between your Amazon File Cache file servers.

To add inbound rules to your security group

1. Select the security group you just created if it's not already selected. For **Actions**, choose **Edit inbound rules**.
2. Add the following inbound rules.

Type	Protocol	Port Range	Source	Description
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group ID of the security group that you just created	Allows traffic between Amazon File Cache file servers
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group IDs of the security groups associated with your Lustre clients	Allows traffic between Amazon File Cache file servers and clients
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group ID of the security group that you just created	Allows traffic between Amazon File Cache file servers
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group IDs of the security groups associated with your File Cache file servers	Allows traffic between Amazon File Cache file servers

3. Choose **Save** to save and apply the new inbound rules.

By default, security group rules allow all outbound traffic (All, 0.0.0.0/0). If your security group doesn't allow all outbound traffic, add the following outbound rules to your security group. These rules allow traffic between Amazon File Cache file servers and clients, and between Amazon File Cache file servers.

To add outbound rules to your security group

1. Choose the same security group to which you just added the inbound rules. For **Actions**, choose **Edit outbound rules**.
2. Add the following outbound rules.

Type	Protocol	Port Range	Source	Description
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group ID of the security group that you just created	Allow Lustre traffic between Amazon File Cache file servers
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group IDs of the security group associated with your Lustre clients	Allow Lustre traffic between Amazon File Cache file servers and Lustre clients
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group ID of the security group that you just created	Allows Lustre traffic between Amazon File Cache file servers

Type	Protocol	Port Range	Source	Description
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group IDs of the security groups associated with your Lustre clients	Allows traffic between Amazon File Cache file servers and Lustre clients

3. Choose **Save** to save and apply the new outbound rules.

To associate a security group with your Amazon File Cache

1. Open the AWS Management Console at https://console.aws.amazon.com/fsx/#fc/file-caches.
2. On the console dashboard, chose your cache to view its details.
3. On the **Network & Security** tab, choose your cache's network interface IDs (for example, `ENI-0123456789012345`6). Doing this redirects you to the Amazon EC2 console.
4. Choose each network interface ID. Each action opens a new instance of the Amazon EC2 console in your browser. For each security group, choose **Change Security Groups** for **Actions**.
5. In the **Change Security Groups** dialog box, choose the security groups to use, and choose **Save**.

Lustre client VPC security group rules

You use VPC security groups to control access to your Lustre clients by adding inbound rules to control incoming traffic and outbound rules to control the outgoing traffic from your Lustre clients. Make sure to have the right network traffic rules in your security group to ensure that Lustre traffic can flow between your Lustre clients and your Amazon File Caches.

Add the following inbound rules to the security groups applied to your clients.

Type	Protocol	Port Range	Source	Description
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group IDs of the security groups that are applied to your Lustre clients	Allows traffic between Amazon File Cache file servers
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group IDs of the security groups associated with your Amazon File Cache file servers	Allow Lustre traffic between Amazon File Cache file servers and clients
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group IDs of the security groups that are applied to your clients	Allows traffic between Amazon File Cache file servers
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group IDs of the security groups associated with your Amazon	Allows traffic between Amazon File Cache file servers and clients

Type	Protocol	Port Range	Source	Description
			File Cache file servers	

Add the following outbound rules to the security groups applied to your clients.

Type	Protocol	Port Range	Source	Description
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group IDs of the security groups that are applied to your Lustre clients	Allows traffic between Lustre clients
Custom TCP rule	TCP	988	Choose **Custom** and enter the security group IDs of the security groups associated with your Amazon File Cache file servers	Allow Lustre traffic between Amazon File Cache file servers and Lustre clients
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group IDs of the security groups that are applied to your Lustre clients	Allows Lustre traffic between Lustre clients

Type	Protocol	Port Range	Source	Description
Custom TCP rule	TCP	1018-1023	Choose **Custom** and enter the security group IDs of the security groups associated with your Amazon File Cache file servers	Allows Lustre traffic between Amazon File Cache file servers and Lustre clients

Amazon VPC Network ACLs

Another option for securing access to the cache within your VPC is to establish network access control lists (network ACLs). Network ACLs are separate from security groups, but have similar functionality to add an additional layer of security to the resources in your VPC. For more information about implementing access control using network ACLs, see Control traffic to subnets using Network ACLs in the *Amazon VPC User Guide*.

Compliance Validation for Amazon File Cache

To learn whether an AWS service is within the scope of specific compliance programs, see and choose the compliance program that you are interested in. For general information, see .

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Amazon File Cache and interface VPC endpoints (AWS PrivateLink)

You can improve the security posture of your VPC by configuring Amazon File Cache to use an interface VPC endpoint. Interface VPC endpoints are powered by AWS PrivateLink, a technology that enables you to privately access AWS APIs without an internet gateway, NAT device, VPN

connection, or Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with AWS APIs. Traffic between your VPC and Amazon FSx does not leave the AWS network.

Each interface VPC endpoint is represented by one or more elastic network interfaces in your subnets. A network interface provides a private IP address that serves as an entry point for traffic to the AWS API.

Considerations for Amazon File Cache interface VPC endpoints

Before you set up an interface VPC endpoint for Amazon File Cache, be sure to review Interface VPC endpoint properties and limitations in the *Amazon VPC User Guide*.

You can call any of the Amazon File Cache API operations from your VPC. For example, you can create a cache by calling the `CreateFileCache` API from within your VPC. For the full list of Amazon File Cache APIs, see Actions in the Amazon FSx API Reference.

VPC peering considerations

You can connect other VPCs to the VPC with interface VPC endpoints using VPC peering. VPC peering is a networking connection between two VPCs. You can establish a VPC peering connection between your own two VPCs, or with a VPC in another AWS account. The VPCs can also be in two different AWS Regions.

Traffic between peered VPCs stays on the AWS network and does not traverse the public internet. Once VPCs are peered, resources like Amazon Elastic Compute Cloud (Amazon EC2) instances in both VPCs can access the Amazon FSx API through interface VPC endpoints created in the one of the VPCs.

Creating an interface VPC endpoint

You can create a VPC endpoint for the Amazon File Cache API using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface VPC endpoint in the *Amazon VPC User Guide*.

For a complete list of endpoints for Amazon FSx services, see Amazon FSx endpoints and quotas in the *Amazon Web Services General Reference*.

To create an interface VPC endpoint for Amazon File Cache, use one of the following:

- **`com.amazonaws.region.fsx`** – Creates an endpoint for Amazon File Cache API operations.
- **`com.amazonaws.region.fsx-fips`** – Creates an endpoint for the Amazon File Cache API that complies with Federal Information Processing Standard (FIPS) 140-2.

To use the private DNS option, you must set the `enableDnsHostnames` and `enableDnsSupport` attributes of your VPC. For more information, see Viewing and updating DNS support for your VPC in the *Amazon VPC User Guide*.

If you enable private DNS for the endpoint, you can make API requests to Amazon FSx with the VPC endpoint using its default DNS name for the AWS Region, for example `fsx.us-east-1.amazonaws.com`.

For more information, see Accessing a service through an interface VPC endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy

To further control access to the Amazon File Cache API, you can optionally 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*.

Quotas

Learn about quotas when working with Amazon File Cache.

Topics

Quotas that you can increase

The following are quotas for Amazon File Cache per AWS account, per AWS Region, which you can increase.

Resource	Default	Description
Lustre Cache_1 caches	100	The maximum number of Amazon File Cache caches with cache type `Lustre` and deployment type `Cache_1` that you can create in this account.
Lustre Cache_1 storage capacity	100800	The maximum amount of storage capacity (in GiB) that you can configure in this account for all Amazon File Cache caches with cache type `Lustre` and deployment type `Cache_1`.

Resource quotas for each cache

The following are limits on Amazon File Cache resources for each cache in an AWS Region.

Resource	Limit per cache
Maximum number of tags	50
Number of file updates from linked S3 bucket per caches	10 million / month
Minimum storage capacity	1.2 TiB
Maximum throughput per unit of storage	1000 MBps

Additional considerations

In addition, note the following:

- You can use each AWS Key Management Service (AWS KMS) key on up to 125 Amazon File Cache caches.
- For a list of AWS Regions where you can create caches, see Amazon File Cache availability.

Troubleshooting

Use the following information to help you resolve issues that you might encounter when working with Amazon File Cache.

Topics

Troubleshooting cache mount issues

There are a number of potential causes when a cache mount command fails, as described in the following topics.

Cache mount fails right away

The cache mount command fails right away. The following code shows an example.

```
mount.lustre: mount fc-0123456789abcdef0.fsx.us-east-1.aws@tcp:/fsx at /mnt
failed: No such file or directory

Is the MGS specification correct?
Is the filesystem name correct?
```

This error can occur if you aren't using the correct `mountname` value when mounting a cache by using the **mount** command. You can get the `mountname` value from the response of the **describe-file-caches** AWS CLI command or the **DescribeFileCaches** API operation, and also from the **Mount name** field on the cache console's **Summary** panel.

Cache mount hangs and then fails with timeout error

The cache mount command hangs for a minute or two, and then fails with a timeout error.

The following code shows an example.

```
sudo mount -t lustre -o relatime,flock cache_dns_name@tcp:/mountname /mnt
```

```
[2+ minute wait here]
Connection timed out
```

This error can occur because the security groups for the Amazon EC2 instance or the cache aren't configured properly.

Action to take

Make sure that your security groups for the cache have the inbound rules specified in Amazon VPC security groups.

Automatic mounting fails and the instance is unresponsive

In some cases, automatic mounting might fail for a cache and your Amazon EC2 instance might stop responding.

This issue can occur if the `_netdev` option wasn't declared. If `_netdev` is missing, your Amazon EC2 instance can stop responding. This result is because network caches need to be initialized after the compute instance starts its networking.

Action to take

If this issue occurs, contact AWS Support.

Cache mount fails during system boot

The cache mount fails during the system boot. The mounting is automated using `/etc/fstab`. When the cache is not mounted, the following error is seen in the syslog for the instance booting time frame.

```
LNetError: 3135:0:(lib-socket.c:583:lnet_sock_listen()) Can't create socket: port 988
 already in use
LNetError: 122-1: Can't start acceptor on port 988: port already in use
```

This error can occur when port 988 is not available. When the instance is configured to mount NFS caches, it is possible that the NFS mounts will bind its client port to port 988

Action to take

You can work around this problem by tuning the NFS client's `noresvport` and `noauto` mount options where possible.

Cache mount using DNS name fails

Misconfigured Domain Name Service (DNS) names can cause cache mount failures, as shown in the following scenarios.

Scenario 1: A cache mount that is using a Domain Name Service (DNS) name fails. The following code shows an example.

```
sudo mount -t lustre cache_dns_name@tcp:/mountname /mnt
mount.lustre: Can't parse NID
'cache_dns_name@tcp:/mountname'
```

Action to take

Check your virtual private cloud (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, do the following:

- Ensure that the Amazon EC2 instance is in the same VPC as your Amazon File Cache.
- Connect your Amazon EC2 instance inside a VPC configured to use the DNS server provided by Amazon. For more information, see DHCP option sets in Amazon VPC in the *Amazon VPC User Guide*.
- Ensure that the Amazon VPC of the connecting Amazon EC2 instance has DNS host names enabled. For more information, see View and update DNS attributes for your VPC in the *Amazon VPC User Guide*.

Scenario 2: A cache mount that is using a Domain Name Service (DNS) name fails. The following code shows an example.

```
mount -t lustre cache_dns_name@tcp:/mountname /mnt
mount.lustre: mount cache_dns_name@tcp:/mountname at /mnt failed: Input/output error Is
 the MGS running?
```

Action to take

Make sure that the client's VPC security groups have the correct outbound traffic rules applied. This recommendation holds true especially if you aren't using the default security group, or if you have

modified the default security group. For more information, see Cache access control with Amazon VPC.

Troubleshooting file access issues

There are a number of potential causes for being unable to access files.

Cannot see files on the cache

You mounted your cache, but you don't see any files.

Action to take

Ensure that you're running a supported Lustre client and that the Linux instance you're using to access the cache has a Linux kernel version that meets the minimum requirement for your client operating system. For more information, see Installing the Lustre client.

Cannot read files in linked NFS file system

You can successfully create a cache linked to your NFS file system, but you get an error when you try to read a file on it.

Action to take

If this issue occurs, do the following:

1. Ensure that your setup complies with the Prerequisites for linking to on-premises NFS data repositories.

   > **Note**
   >
   > While Amazon File Cache supports NFSv3 file systems with most NFSv3 export policies, you must not use the NFS export option `all_squash`.

2. Are all of the file servers and metadata servers in the cache able to reach your linked NFS file system?

   In order for Amazon File Cache to be able to access data in your NFS file system, all of the File Cache network interfaces must be able to communicate with your NFS file system. If you have a network firewall or IP allow-list, ensure that it includes all of the cache IP addresses and

allows traffic on the ports that File Cache requires to be open. For more information, see Cache access control with Amazon VPC.

You can see all of your cache IP addresses by going to the Amazon EC2 console, go to **Network Interfaces** in the left-side menu under **Network & Security**, and filter by `Cache ID`.

Alternatively, you can use the AWS CLI and type the following command:

```
aws ec2 describe-network-interfaces --filters
 Name=description,Values=*fc-0123456789abcdef0
```

For `Values` in the `--filters` option, be sure to replace the sample cache ID in the command with your cache ID.

Troubleshooting File Cache CSI driver issues

If you're experiencing issues with the Amazon File Cache CSI driver for containers running on Amazon EKS, see Troubleshooting CSI Driver (Common Issues) which is available on GitHub.

Document history

- **API version:** 2018-03-01
- **Latest documentation update:** July 22, 2025

The following table describes important changes to the *Amazon File Cache User Guide*. For notifications about documentation updates, you can subscribe to the RSS feed.

Change	Description	Date
Amazon FSx updated the AmazonFSxServiceRolePolicy AWS managed policy	Amazon FSx added the `ec2:AssignIpv6Addresses` and `ec2:UnassignIpv6Addresses` permissions to the AmazonFSxServiceRolePolicy. For more information, see Amazon FSx updates to AWS managed policies.	July 22, 2025
Amazon FSx updated the AmazonFSxFullAccess AWS managed policy	The AmazonFSxFullAccess managed policy was updated to add the `fsx:CreateAndAttachS3AccessPoint` , `fsx:DescribeS3AccessPointAttachments` , and `fsx:DetachAndDeleteS3AccessPoint` permissions.	June 25, 2025
Amazon FSx updated the AmazonFSxConsoleFullAccess AWS managed policy	The AmazonFSxConsoleFullAccess managed policy was updated to add the `fsx:CreateAndAttachS3AccessPoint` ,	June 25, 2025

	`fsx:DescribeS3AccessPointAttachments`, and `fsx:DetachAndDeleteS3AccessPoint` permissions.	
Amazon FSx updated the AmazonFSxConsoleReadOnlyAccess AWS managed policy	Amazon FSx updated the AmazonFSxConsoleReadOnlyAccess policy to add the `ec2:DescribeNetworkInterfaces` permission. For more information, see the AmazonFSxConsoleReadOnlyAccess policy.	February 25, 2025
Amazon FSx updated the AmazonFSxConsoleFullAccess AWS managed policy	Amazon FSx updated the AmazonFSxConsoleFullAccess policy to add the `ec2:DescribeNetworkInterfaces` permission. For more information, see the AmazonFSxConsoleFullAccess policy.	February 7, 2025
Amazon FSx updated the AmazonFSxFullAccess, AmazonFSxConsoleFullAccess, AmazonFSxReadOnlyAccess, AmazonFSxConsoleReadOnlyAccess, and AmazonFSxServiceRolePolicy AWS managed policies	Amazon FSx updated the AmazonFSxFullAccess, AmazonFSxConsoleFullAccess, AmazonFSxReadOnlyAccess, AmazonFSxConsoleReadOnlyAccess, and AmazonFSxServiceRolePolicy policies to add the `ec2:GetSecurityGroupsForVpc` permission. For more information, see Amazon FSx updates to AWS managed policies.	January 9, 2024

Amazon FSx for Lustre updated the AmazonFSx FullAccess and the AmazonFSxConsoleFullAccess AWS managed policies	Amazon FSx updated the AmazonFSxFullAccess and AmazonFSxConsoleFullAccess policies to add the `ManageCrossAccountDataReplication` action. For more information, see Amazon FSx updates to AWS managed policies.	December 20, 2023
Amazon FSx updated the AmazonFSxFullAccess and the AmazonFSxConsoleFullAccess AWS managed policies	Amazon FSx updated the AmazonFSxFullAccess and AmazonFSxConsoleFullAccess policies to add the `fsx:CopySnapshotAndUpdateVolume` permission. For more information, see Amazon FSx updates to AWS managed policies.	November 26, 2023
Amazon FSx updated the AmazonFSxFullAccess and the AmazonFSxConsoleFullAccess AWS managed policies	Amazon FSx updated the AmazonFSxFullAccess and AmazonFSxConsoleFullAccess policies to add the `fsx:DescribeSharedVPCConfiguration` and `fsx:UpdateSharedVPCConfiguration` permissions. For more information, see Amazon FSx updates to AWS managed policies.	November 14, 2023

Amazon FSx updated the AmazonFSxServiceRolePolicy AWS managed policy	Amazon FSx updated the `cloudwatch:PutMetricData` permission in the AmazonFSxServiceRolePolicy. For more information, see Amazon FSx updates to AWS managed policies.	July 24, 2023
Amazon FSx updated the AmazonFSxFullAccess AWS managed policy	Amazon FSx updated the AmazonFSxFullAccess policy to remove the `fsx:*` permission and add specific `fsx` actions. For more information, see AmazonFSxFullAccess policy.	July 13, 2023
Amazon FSx updated the AmazonFSxConsoleFullAccess AWS managed policy	Amazon FSx updated the AmazonFSxConsoleFullAccess policy to remove the `fsx:*` permission and add specific `fsx` actions. For more information, see AmazonFSxConsoleFullAccess policy.	July 13, 2023
Lustre client support for Centos, Rocky Linux, and Red Hat Enterprise Linux (RHEL) 8.8 added	The FSx for Lustre client now supports Amazon EC2 instances running Centos, Rocky Linux, and Red Hat Enterprise Linux (RHEL) 8.8. For more information, see Installing the Lustre client.	May 25, 2023

Support for Amazon EKS added	You can now access your cache from containers running on Amazon EKS using the open-source Amazon File Cache CSI driver. For more information, see Accessing Amazon File Cache.	March 3, 2023
Lustre client support for Amazon Linux and Amazon Linux 2 added	The Lustre client now supports Amazon EC2 instances running Amazon Linux and Amazon Linux 2. For more information, see Installing the Lustre client.	December 13, 2022
Additional AWS Region support added for caches	Amazon File Cache is now available in the Europe (Stockholm), Asia Pacific (Hong Kong), Asia Pacific (Mumbai), and Asia Pacific (Seoul) AWS Regions. For more information, see Amazon File Cache availability.	November 10, 2022
Amazon File Cache is now generally available	Amazon File Cache is a high-speed cache on AWS that makes it easier to process file data, regardless of where the data is stored.	September 29, 2022