AWS Whitepaper

Encrypting File Data with Amazon Elastic File System

Encrypting File Data with Amazon Elastic File System: AWS Whitepaper

Table of Contents

Encrypting File Data with Amazon Elastic File System

Publication date: **February 22,2021** (*Document History and Contributors*)

Abstract

Security is the top priority at AWS and we give our customers the tools to prioritize security in their enterprise. Government regulations and industry or company compliance policies may require data of different classifications to be secured by using encryption policies, cryptographic algorithms, and proper key management. This paper outlines best practices for encrypting Amazon Elastic File System (Amazon EFS).

Introduction

Amazon Elastic File System (Amazon EFS) provides simple, scalable, highly available, and highly durable shared file systems in the cloud. The file systems you create using Amazon EFS are elastic, allowing them to grow and shrink automatically as you add and remove data. They can grow in size to petabytes, distributing data across an unconstrained number of storage servers in multiple Availability Zones (AZs).

Data stored in these file systems can be encrypted at rest and in transit using Amazon EFS. For encryption of data at rest, you can create encrypted file systems through the AWS Management Console or the AWS Command Line Interface (AWS CLI). Or you can create encrypted file systems programmatically through the Amazon EFS API or one of the AWS SDKs.

For encryption of data at rest, Amazon EFS integrates with AWS Key Management Service (AWS KMS) for key management. You can also enable encryption of data in transit by mounting the file system and transferring all NFS traffic over a Transport Layer Security (TLS).

This paper outlines encryption best practices for Amazon EFS. It describes how to enable encryption of data in transit at the client connection layer, and how to create an encrypted file system in the AWS Management Console and in the AWS CLI.

Note

Using the APIs and SDKs to create an encrypted file system is outside the scope of this paper. For more information about how this is done, see Amazon EFS API in the *Amazon EFS User Guide* or the SDK documentation.

Basic Concepts and Terminology

This section defines concepts and terminology referenced in this whitepaper.

- **Amazon Elastic File System (Amazon EFS)** – A highly available and highly durable service that provides simple, scalable, shared file storage in the AWS Cloud. Amazon EFS provides a standard file system interface and file system semantics. You can store virtually an unlimited amount of data across an unconstrained number of storage servers in multiple Availability Zones.
- **AWS Identity and Access Management (IAM) –** A service that enables you to securely control fine-grained access to AWS service APIs. Policies are created and used to limit access to individual users, groups, and roles. You can manage your AWS KMS keys through the IAM console.
- **AWS KMS –** A managed service that makes it easy for you to create and control KMS keys (formerly CMKs), the encryption keys used to encrypt your data. KMS keys are protected by hardware security modules (HSMs) that are validated by the FIPS 140-2 Cryptographic Module Validation Program except in the China (Beijing) and China (Ningxia) Regions. AWS KMS is integrated with other AWS services that encrypt your data. It is also fully integrated with AWS CloudTrail to provide logs of API calls made by AWS KMS on your behalf, which can be helpful for meeting compliance or regulatory requirements applicable to your organization.
- **KMS key (formerly CMK) –** Represents the top of your key hierarchy. It contains key material to encrypt and decrypt data. AWS KMS can generate this key material, or you can generate it and then import it into AWS KMS. KMS keys are specific to an AWS account and AWS Region and can be customer-managed or AWS-managed.
- **AWS KMS key (formerly AWS-managed CMK)** – A KMS key that is generated by AWS on your behalf. An AWS KMS key is created when you enable encryption for a resource of an integrated AWS service. AWS KMS key policies are managed by AWS and you cannot change them. There is no charge for the creation or storage of AWS KMS keys.
- **Customer-managed KMS key** – A KMS key you create by using the AWS Management Console or API, AWS CLI, or SDKs. You can use a customer-managed KMS key when you need more granular control over the CMK.
- **KMS Key Policy –** A resource policy that controls access to a customer managed KMS key. Customers define these permissions using the key policy or a combination of IAM policies and the key policy. For more information, see Overview of Managing Access in the *AWS KMS Developer Guide*.

- **Data keys –** Cryptographic keys generated by AWS KMS to encrypt data outside of AWS KMS. AWS KMS allows authorized entities (users or services) to obtain data keys protected by a KMS key.
- **Transport Layer Security (TLS) –** The successor to Secure Sockets Layer (SSL), TLS is a cryptographic protocol essential for encrypting information that is exchanged over a network.
- **EFS mount helper –** A Linux client agent (`amazon-efs-utils`) used to simplify the mounting of EFS file systems. It can be used to setup, maintain, and route all NFS traffic over a TLS tunnel.

For more information about basic concepts and terminology, see AWS Key Management Service Concepts in the *AWS KMS Developer Guide*.

Encryption of Data at Rest

AWS provides the tools for you to create an encrypted file system that encrypts all of your data and metadata at rest using an industry standard AES-256 encryption algorithm . An encrypted file system is designed to handle encryption and decryption automatically and transparently, so you don't have to modify your applications. If your organization is subject to corporate or regulatory policies that require encryption of data and metadata at rest, we recommend that you create an encrypted file system.

Topics

Managing Keys

Amazon EFS is integrated with AWS KMS, which manages the encryption keys for encrypted file systems. AWS KMS also supports encryption by other AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), Amazon Relational Database Service (Amazon RDS), Amazon Aurora, Amazon Redshift, Amazon WorkMail, WorkSpaces, etc. To encrypt file system contents, Amazon EFS uses the Advanced Encryption Standard algorithm with XTS Mode and a 256-bit key (XTS-AES-256).

There are three important questions to answer when considering how to secure data at rest by adopting any encryption policy. These questions are equally valid for data stored in managed and unmanaged services such as Amazon EBS.

Where are keys stored?

AWS KMS stores your KMS keys in highly durable storage in an encrypted format to help ensure that they can be retrieved when needed.

Where are keys used?

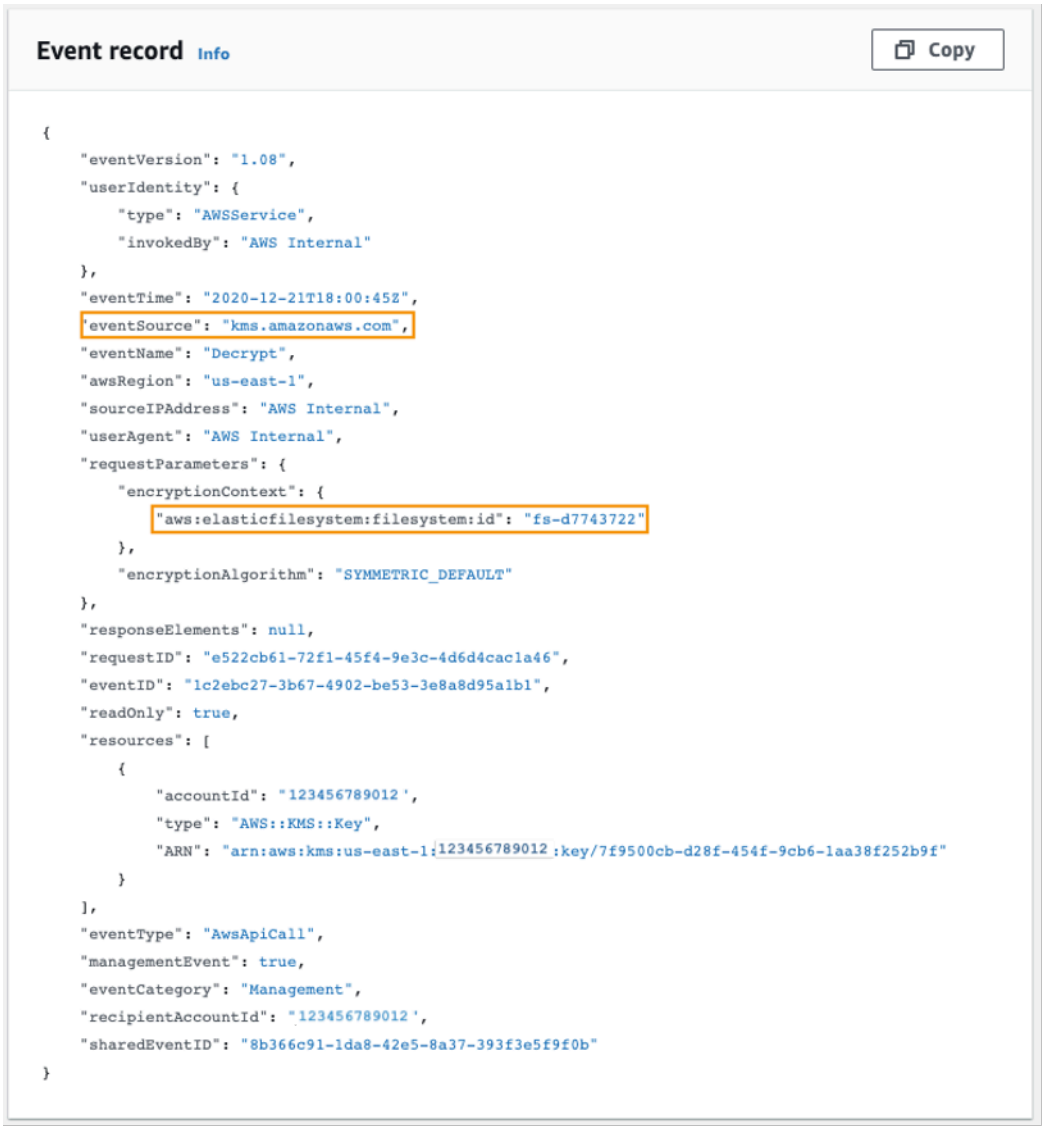
Using an encrypted Amazon EFS file system is transparent to clients mounting the file system. All cryptographic operations occur within the EFS service, as data is encrypted before it is written to disk and decrypted after a client issues a read request.

Who can use the keys?

AWS KMS key policies control access to encryption keys.

We recommend you combine them with IAM policies to provide another layer of control. Each key has a key policy. If the key is an AWS managed CMK, AWS manages the key policy. If the key is a customer managed CMK, you manage the key policy. These key policies are the primary way to control access to CMKs. They define the permissions that govern the use and management of keys.

When you create an encrypted file system using Amazon EFS, you grant the Amazon EFS access to use the CMK on your behalf. The calls that Amazon EFS makes to AWS KMS on your behalf appear in your CloudTrail logs as though they originated from your AWS account. The following screenshot shows the sample CloudTrail event for a KMS Decrypt call made by Amazon EFS.



CloudTrail log for KMS Decrypt

For more information about AWS KMS and how to manage access to encryption keys, see Managing Access to AWS KMS CMKs in the *AWS KMS Developer Guide*.

For more information about how AWS KMS manages cryptography, see the AWS KMS Cryptographic Details whitepaper.

For more information about how to create an administrative user and group, see Create an administrative user in the *IAM User Guide*.

Creating an Encrypted File System

You can create an encrypted file system using the AWS Management Console, AWS CLI, Amazon EFS API, or AWS SDKs. You can only enable encryption for a file system when you create it.

Amazon EFS integrates with AWS KMS for key management and uses a CMK to encrypt the file system. File system metadata, such as file names, directory names, and directory contents, are encrypted and decrypted using an AWS managed CMK.

The contents of your files, or file data, is encrypted and decrypted using a CMK that you choose. The CMK can be one of three types:

- An AWS managed CMK for Amazon EFS
- A customer managed CMK from your AWS account
- A customer managed CMK from a different AWS account

Your organization might be subject to corporate or regulatory policies that require complete control in terms of creation, rotation, deletion as well as the access control and usage policy for the CMKs. If so, we recommend that you use a customer managed CMK. In other scenarios, you can use an AWS managed CMK.

All users have an AWS-managed CMK for Amazon EFS, whose alias is `aws/elasticfilesystem`. AWS manages this CMK's key policy and you cannot change it. There is no cost for creating and storing AWS managed CMKs.

If you decide to use a customer managed CMK to encrypt your file system, select the key alias of the customer managed CMK that you own. Alternatively, you can enter the Amazon Resource Name (ARN) of a customer managed CMK that is owned by a different account. With a customer managed CMK that you own, you control which users and services can use the key through key policies and key grants.

You also control the life span and rotation of these keys by choosing when to disable, re-enable, delete, or revoke access to them. For information about managing access to keys in other AWS accounts, see Changing a key policy in the *AWS KMS Developer Guide*.

For more information about how to manage customer managed CMKs, see KMS keys (formerly CMKs) in the *AWS KMS Developer Guide*.

The following sections discuss how to create an encrypted file system using the AWS Management Console and using the AWS CLI.

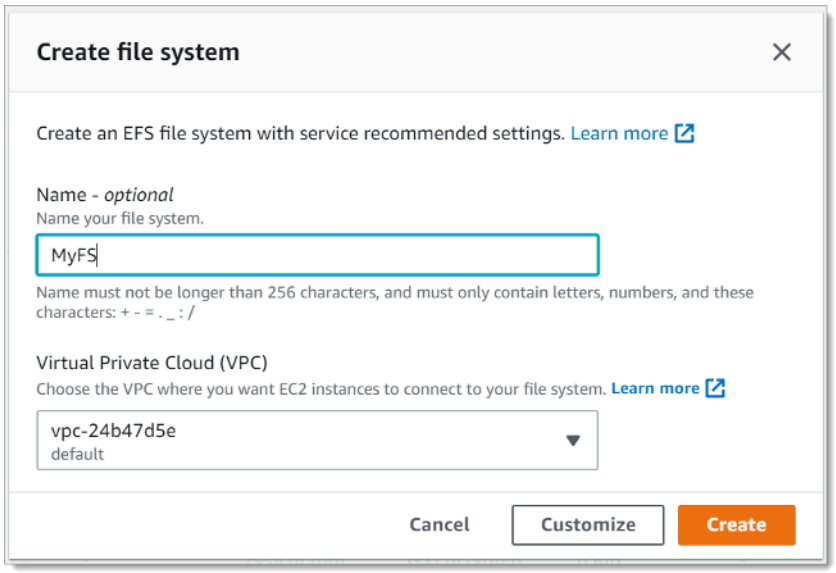
Creating an Encrypted File System Using the AWS Management Console

Use the following procedure to create an encrypted Amazon EFS file system using the AWS Management Console.

Step 1. Configure File System Settings

In this step, you configure general file system settings, including Lifecycle management, Performance and Throughput modes, and encryption of data at rest.

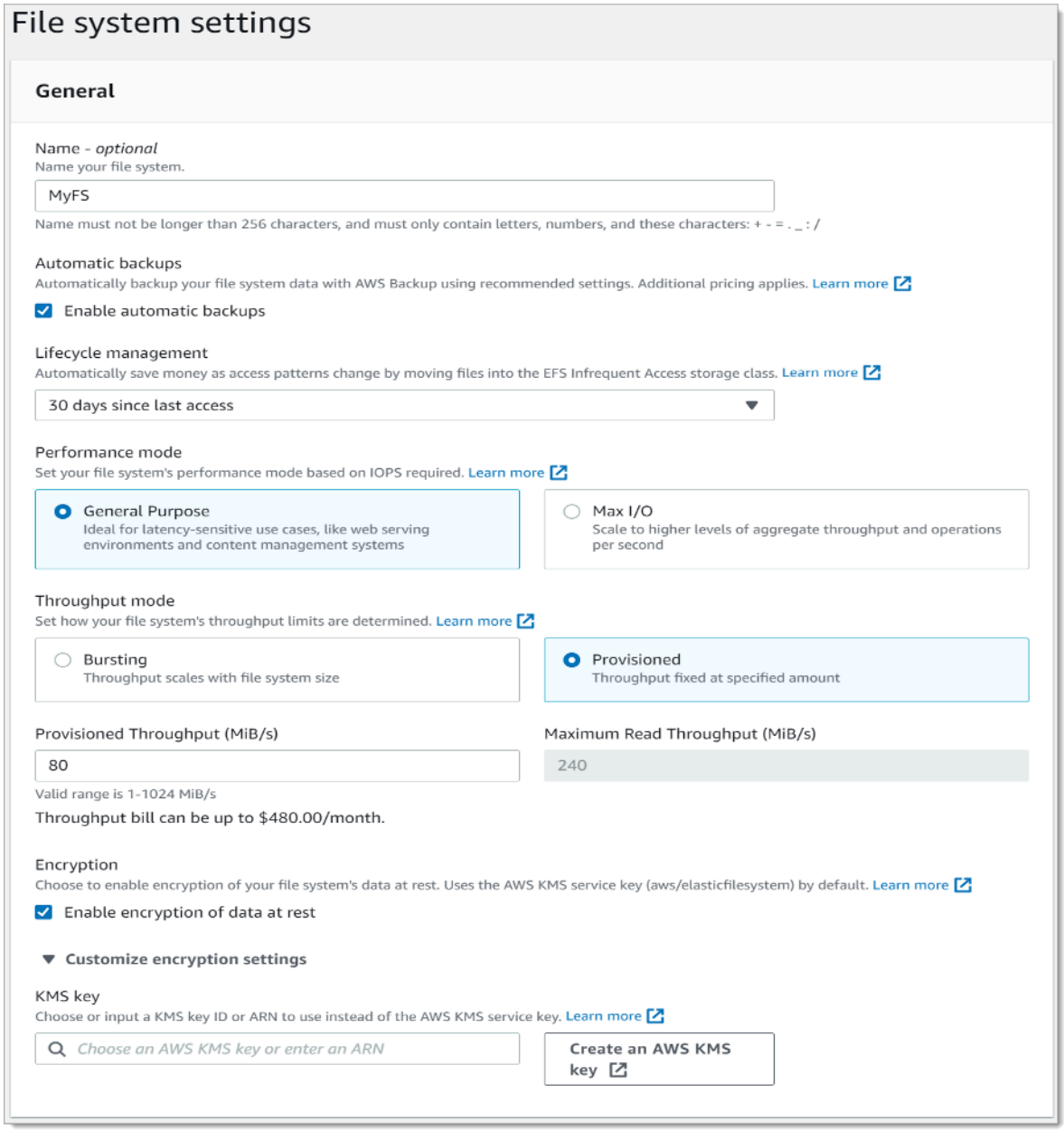
1. Sign in to the AWS Management Console and open the Amazon EFS console.
2. Choose **Create file system** to open the **Create file system** dialog box. For more information about creating a file system using the recommended settings that include enabling encryption by default, see Create Your Amazon EFS File System.



Create EFS File System

3. (Optional) Select **Customize** to create a customized file system instead of creating a file system using the service recommended settings.

   The File system settings page appears.



Create EFS file system: general settings

4. For **General** settings, enter the following details.
   - (Optional) Enter a **Name** for the file system.
   - **Automatic backups** are turned on by default. You can turn off automatic backups by clearing the check box. For more information, see Using AWS Backup with Amazon EFS.
   - Choose a **Lifecycle management** policy. Amazon EFS lifecycle management automatically manages cost-effective file storage for your file systems. When enabled, lifecycle management migrates files that have not been accessed for a set period to the Infrequent

Access (IA) storage class. You define that period by using a lifecycle policy. If you don't want lifecycle management enabled, choose **None**. For more information, see EFS lifecycle management in the *Amazon EFS User Guide*.

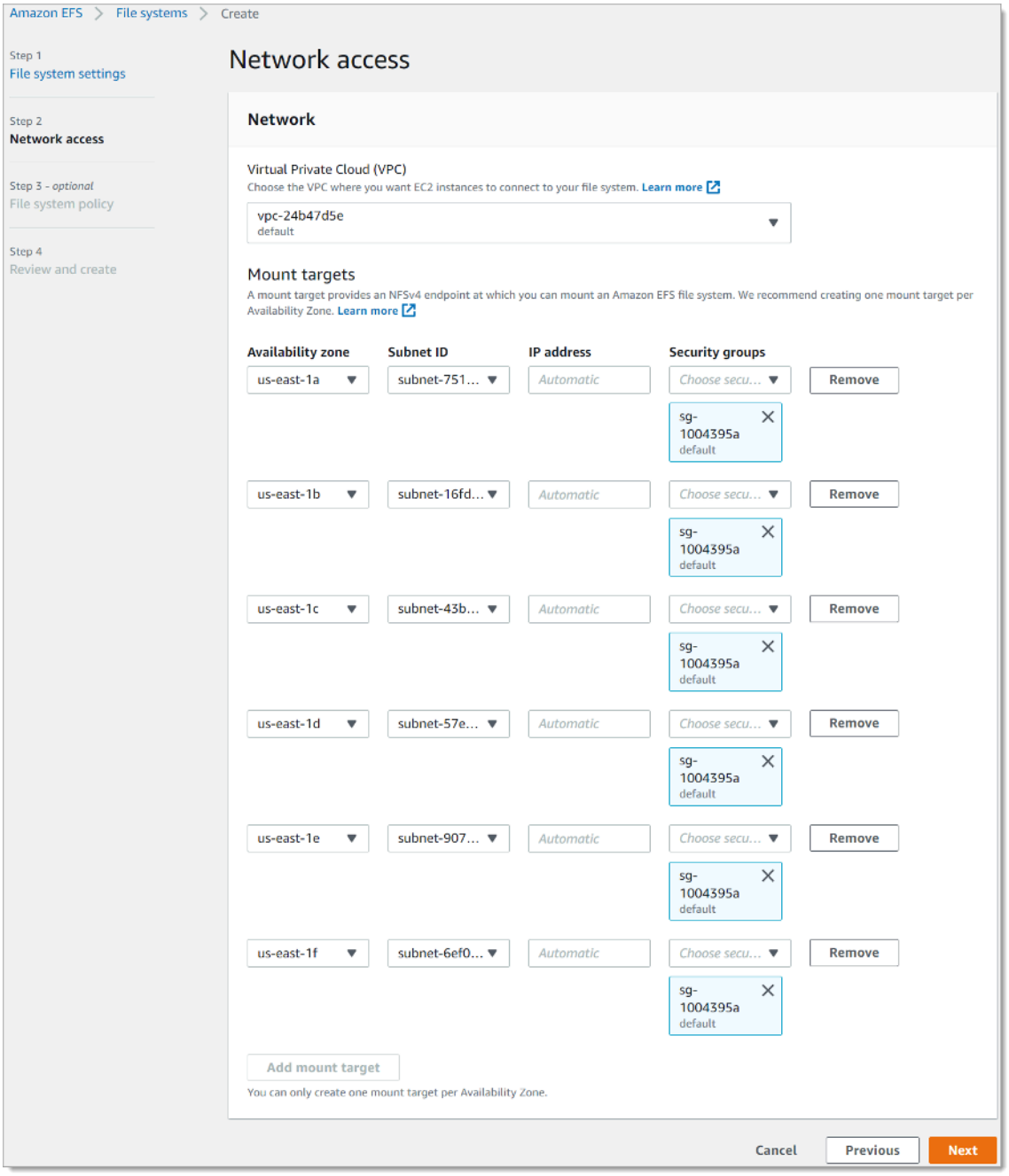
- Choose a **Performance mode** either the default **General Purpose mode** or **Max I/O**. For more information, see Performance Modes in the *Amazon EFS User Guide*.
- Choose a **Throughput mode** either the default **Bursting mode** or **Provisioned mode**.
- If you selected **Provisioned** the **Provisioned Throughput (MiB/s)** field displays. Enter the amount of throughput to provision for the file system. After you enter the throughput, the console displays an estimate of the monthly cost next to the field. For more information, see Throughput Modes in the *Amazon EFS User Guide*.
- For **Encryption**, encryption of data at rest is enabled by default. It uses your AWS Key Management Service (AWS KMS) EFS service key (`aws/elasticfilesystem`) by default. To choose a different KMS key to use for encryption, expand Customize encryption settings and choose a key from the list. Or, enter a KMS key ID or Amazon Resource Name (ARN) for the KMS key you want to use.

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

5. (Optional) Choose **Add tag** to add key-value pairs to your file system.
6. Choose **Next** to continue to the **Network Access** step in the configuration process.

Step 2. Configure Network Access

In this step, you configure the file system's network settings, including the virtual private cloud (VPC) and mount targets. For each mount target, set the Availability Zone, subnet, IP address, and security groups.



Create EFS file system: Network access

1. Choose the **Virtual Private Cloud** (VPC) where you want EC2 instances to connect to your file system. For more information, see Managing file system network accessibility in the *Amazon EFS User Guide*.
   - **Availability zone** – By default, a mount target is configured in each Availability Zone in an AWS Region. If you don't want a mount target in a particular Availability Zone, choose

Remove to delete the mount target for that zone. Create a mount target in every Availability Zone that you plan to access your file system from. There is no cost to do so.

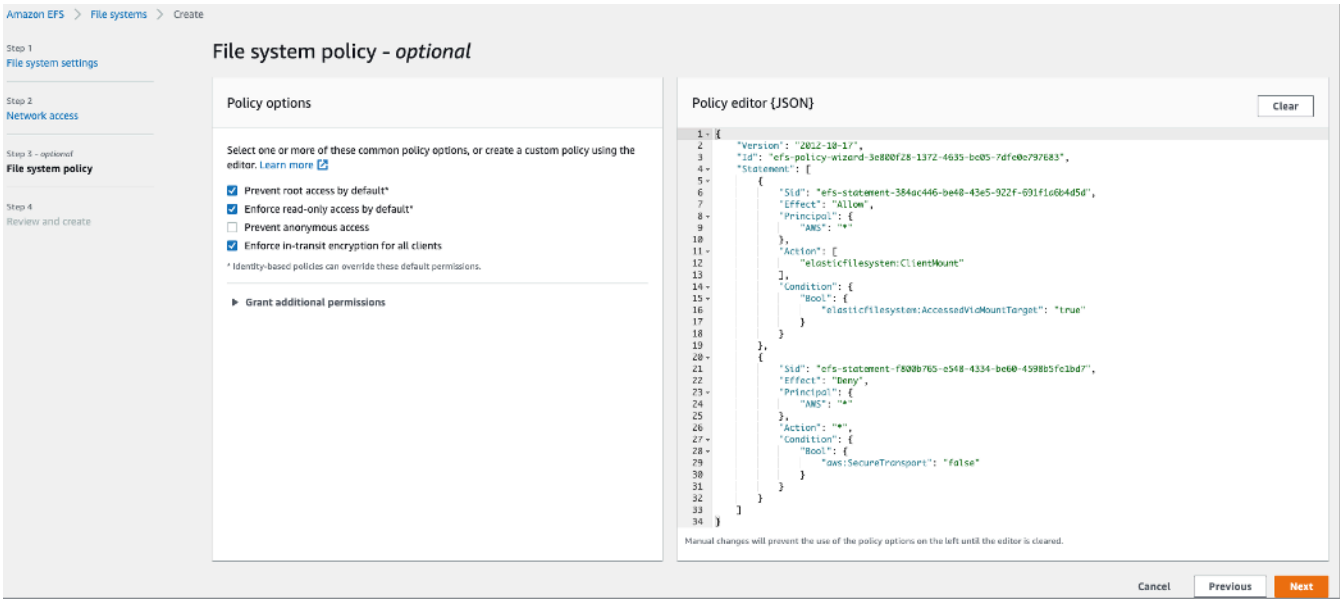
- **Subnet ID** – Choose from the available subnets in an Availability Zone. The default subnet is preselected. As a best practice, ensure that the chosen subnet is public or private based on your security requirements.
- **IP Address** – By default, Amazon EFS chooses the IP address automatically from the available addresses in the subnet. Or, you can enter a specific IP address that's in the subnet. Although mount targets have a single IP address, they are redundant, highly available network resources.
- **Security groups** – You can specify one or more security groups for the mount target. As a best practice, ensure the security group is only used for EFS mount purposes (NFS Port 2049) and inbound rules allow only port 2049 from other VPC CIDR block range or use Security Group as the source for resources that needs to access EFS. For more information, see Using Security Groups for Amazon EC2 Instances and Mount Targets in the *Amazon EFS User Guide*.

  To add another security group, or to change the security group, select **Choose security groups** and add another security group from the list. If you don't want to use the default security group, you can delete it. For more information, see Creating security groups in the *Amazon EFS User Guide*.

2. Choose **Add mount target** to create a mount target for an Availability Zone that doesn't have one. If a mount target is configured for each Availability Zone, this choice is not available.
3. Choose **Next** to continue. The **File system policy** page is displayed.

Step 3. Create a File System Policy

In this step, you create a file system policy to control NFS client access to the file system. An EFS file system policy is an IAM resource policy that you use to control NFS client access to the file system. For more information, see Using IAM to Control NFS Access to Amazon EFS in the *Amazon EFS User Guide*.

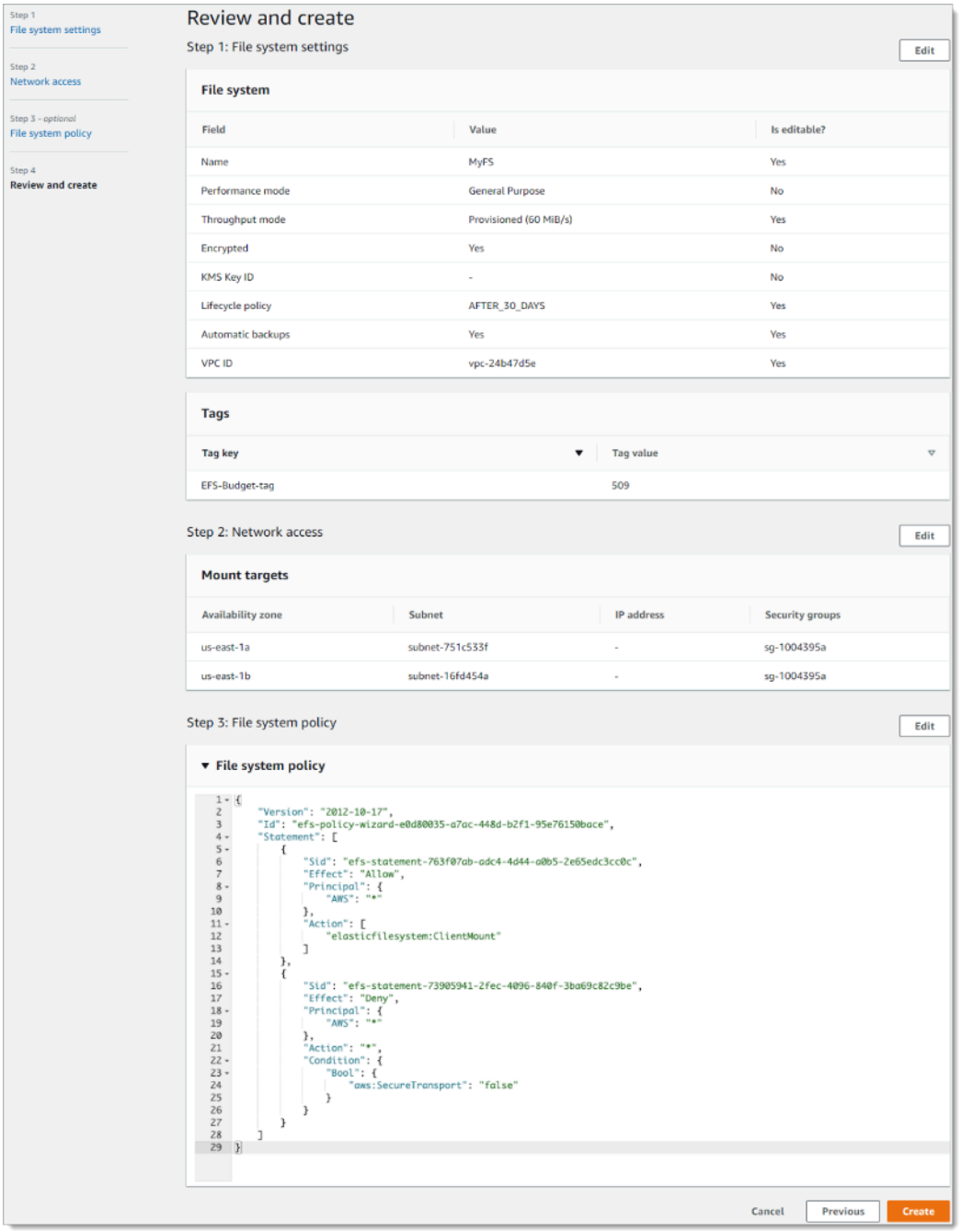


Create EFS file system: File system policy

1. In **Policy options**, we recommend that you choose following available preconfigured policy options:
   - Prevent root access by default
   - Enforce read-only access by default
   - Enforce in-transit encryption for all clients
2. Use **Grant additional permissions** to grant file system permissions to additional IAM principals, including another AWS account. Choose **Add**, then enter the Principal ARN of the entity to which you are granting permissions to, then choose the **Permissions** to grant.
3. Use the **Policy editor** to customize a preconfigured policy or to create your own policy based on your requirements. When you choose one of the preconfigured policies, the JSON policy definition appears in the policy editor.
4. Choose **Next** to continue. The **Review and create** page will appear.

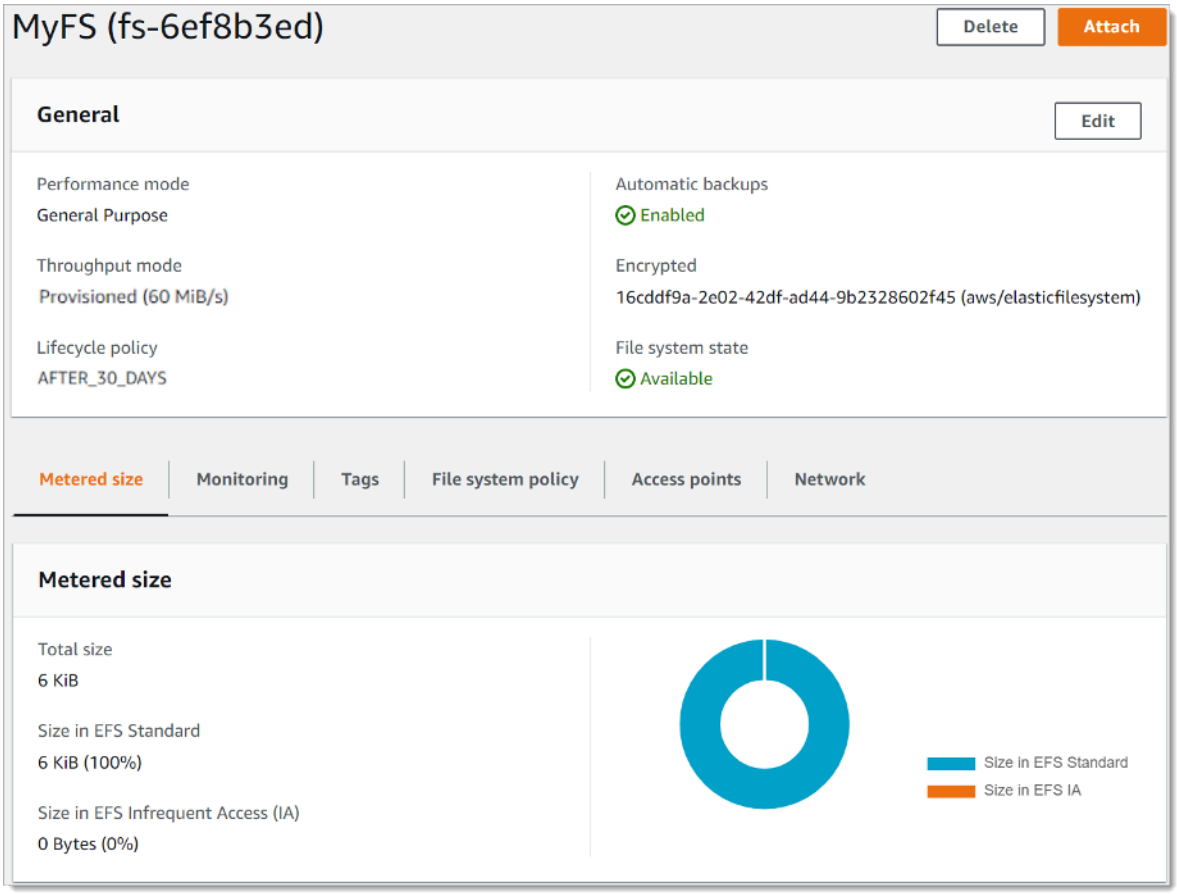
Step 4. Review and Create

In this step, you review the file system settings, make any modifications, then create the file system.



Create EFS file system: Review and create

1. Review each of the file system configuration groups. You can make changes to each group at this time by choosing Edit.
2. Choose Create to create your file system and return to the File systems page.
3. The File systems page displays the file system and its configuration details, as shown in the following image.



File Systems

Creating an Encrypted File System Using the AWS CLI

When you use the AWS CLI to create an encrypted file system, you can use additional parameters to set the encryption status and customer managed CMK. Be sure you are using the latest version of the AWS CLI. For information about how to upgrade your AWS CLI, see Installing, Updating, and Uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*.

In the `CreateFileSystem` operation, the `--encrypted` parameter is a Boolean and is required for creating encrypted file systems. The `--kms-key-id` is required only when you use a customer managed CMK and you include the key's alias or ARN. Do not include this parameter if you're using the AWS managed CMK.

```
$ aws efs create-file-system \
--creation-token $(uuidgen) \
--performance-mode generalPurpose \
```

```
        --encrypted \
        --kms-key-id user/customer-managedCMKalias
```

For more information about creating Amazon EFS file systems using the AWS Management Console, AWS CLI, AWS SDKs, or Amazon EFS API, see What is Amazon Elastic File System *Amazon EFS User Guide*.

Enforcing Encryption of Data at Rest

Encryption has minimal effect on I/O latency and throughput. Encryption and decryption are transparent to users, applications, and services. All data and metadata is encrypted by Amazon EFS on your behalf before it is written to disk and is decrypted before it is read by clients. You don't need to change client tools, applications, or services to access an encrypted file system.

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 can use AWS Identity and Access Management (IAM) identity based policies enforce encryption of data at rest for your Amazon EFS file system resources. Using an IAM condition key, you can prevent users from creating EFS file systems that aren't encrypted.

For example, an IAM policy that explicitly allows users to create only encrypted EFS file systems uses the following combination of effect, action, and condition:

- The `Effect` is `Allow`.
- The `Action` is `elasticfilesystem:CreateFileSystem`.
- The `Condition elasticfilesystem:Encrypted` is `true`.

The following example illustrates an IAM identity-based policy that authorizes principals to create only encrypted file systems.

```
{
  “Version”: “2012-10-17”,
  “Statement”: [
    {
      “Sid”: “VisualEditior0”,
      “Effect”: “Allow”,
      “Action”: “elasticfilesystem:CreateFileSystem”,
```

```
      "Condition": {
        "Bool": {
          "elasticfilesystem:Encrypted": "true"
        }
      },
      "Resource": "*"
    }
}
```

The `Resource` attribute set to `*` means that the IAM policy applies to all the EFS resources created. You can add additional conditional attributes based on tags to enforce it only for a subset of EFS resources with data classification needs.

You can also enforce the creation of encrypted Amazon EFS file systems at the AWS Organizations level by using service control policies for all AWS Accounts or OUs in your organization. For more information about service control policies in AWS Organizations, see Service control policies in the *AWS Organizations User Guide*.

Creating an IAM Policy Requiring that all EFS File Systems be Encrypted

You can create an IAM identity-based policy that authorizes users to create only encrypted Amazon EFS file systems using the console, the AWS CLI, or the API. The following procedure describes how to create such a policy using the IAM console, and then apply the policy to a user in your account.

To create an IAM policy to enforce encrypted EFS file systems:

1. Sign in to the AWS Management Console and open the IAM console.
2. In the navigation pane, under **Access Management**choose **Policies**.
3. Choose **Create policy** to display the Create policy page.
4. In the **Visual Editor** tab, enter the following information.

   - For **Service**, choose **EFS**.
   - For **Actions**, enter `create` in the search field, and then choose **CreateFileSystem**.
   - For **Request conditions**, click on the **Add condition** link, search for `elasticfilesystem:Encrypted` for **Condition Key**, `Bool` for **Operator** and `true` for **Value**.

5. Provide a **Name** and a **Description** for the policy. Verify the policy summary, including the **Encrypted** request condition.
6. Choose **Create policy** to create the policy.

To apply the policy to a user in your account:

1. In the IAM console, under **Access management**, choose **Users**.
2. Select the user that you want to apply the policy to.
3. Choose **Add permissions** to display the Add permissions page.
4. Choose **Attach existing policies directly**.
5. Enter the name of the EFS policy that you created in the previous procedure.
6. Select and expand the policy. Then choose **{}JSON** to verify the policy content. It should look like the following JSON policy.

```
{
  “Version”: “2012-10-17”,
  “Statement”: [
    {
      “Sid”: “VisualEditior0”,
      “Effect”: “Allow”,
      “Action”: “elasticfilesystem:CreateFileSystem”,
      “Condition”: {
        “Bool”: {
          “elasticfilesystem:Encrypted”: “true”
        }
      },
      “Resource”: “*”
    }
}
```

Detecting Unencrypted File Systems

Your organization may have a requirement to identify Amazon EFS resources that are not encrypted. You can detect unencrypted file systems by using AWS Config Managed Rules. AWS Config provides AWS Managed Rules, which are predefined, customizable rules that AWS Config uses to evaluate whether your AWS resources comply with common best practices and flag the resources failing the rules as NON_COMPLIANT.

You can use the AWS Managed Config rule `efs-encrypted-check` to check whether Amazon Elastic File System (Amazon EFS) is configured to encrypt the file data using AWS Key Management Service (AWS KMS). For more information about setting up and activating the AWS Managed Rules, see Working with AWS Config Managed Rules.

Encryption of Data in Transit

You can mount a file system so that all NFS traffic is encrypted in transit using Transport Layer Security 1.2 (TLS) with an industry-standard AES-256 cipher. TLS is a set of industry-standard cryptographic protocols used for encrypting information that is exchanged over the network. AES-256 is a 256-bit encryption cipher used for data transmission in TLS. We recommend setting up encryption in transit on every client accessing the file system.

You can use IAM policies to enforce encryption in transit for NFS client access to Amazon EFS. When a client connects to a file system, Amazon EFS evaluates the file system's IAM resource policy, which is called a file system policy, along with any identity-based IAM policies to determine the appropriate file system access permissions to grant. You can use the `aws:SecureTransport` Condition Key in the file system resource policy to enforce NFS clients to use TLS when connecting to an EFS file system.

> **Note**
>
> You must use the EFS mount helper to mount your Amazon EFS file systems in order to use IAM authorization to control access by NFS clients. For more information, see Mounting with IAM authorization in the *Amazon EFS User Guide*.

The following example EFS file system policy enforces encryption in transit and has the following characteristics:

- The `effect` is `allow`.
- The principal is set to `*` for all IAM entities.
- The action is set to `ClientMount`, `ClientWrite`, `ClientRootAccess`.
- The condition for granting permissions is set to `SecureTransport`. Only NFS clients using TLS to connect to the file system are granted access.

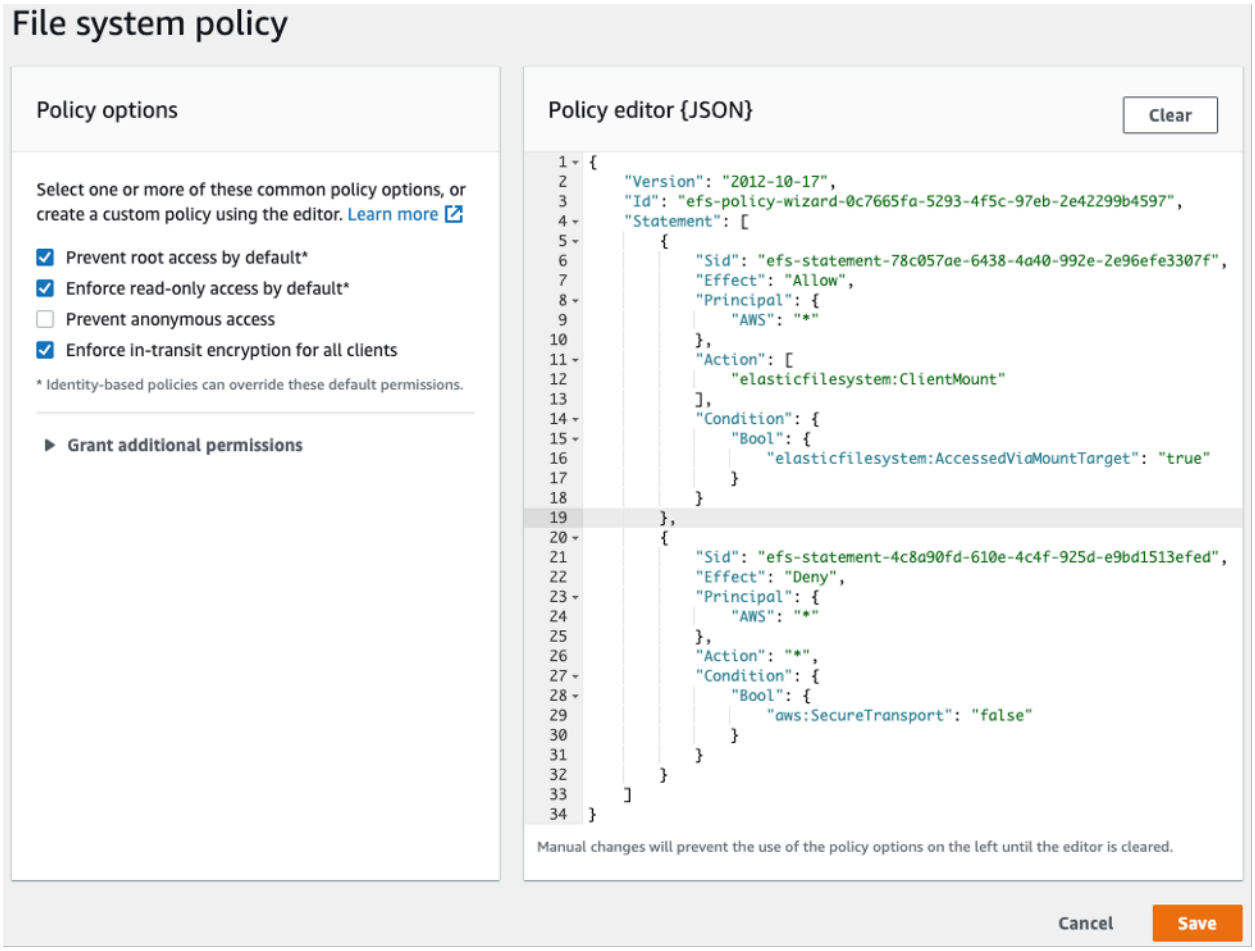
```
{
  "Version": "2012-10-17",
  "Id": "ExamplePolicy01",
  "Statement": [
    {
      "Sid": "VisualEditior0",
```

```
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
"elasticfilesystem:ClientRootAccess",
"elasticfilesystem:ClientMount",
"elasticfilesystem:ClientWrite"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

You can create a file system policy using the Amazon EFS console or using the AWS CLI.

To create a file system policy using the EFS console:

1. Open the Amazon EFS console.
2. Choose **File Systems**.
3. On the File systems page, choose the file system that you want to edit or create a file system policy for. The details page for that file system is displayed.
4. Choose **File system policy**, then choose **Edit.** The File system policy page appears.



Create file system policy

5. In **Policy options**, we recommend that you choose the following available preconfigured policy options:

   - Prevent root access by default
   - Enforce read-only access by default
   - Enforce in-transit encryption for all clients

   If you choose a preconfigured policy, the policy JSON object is displayed in the **Policy editor** panel.

6. Use **Grant additional permissions** to grant file system permissions to additional IAM principals, including another AWS account. Choose **Add**, then enter the Principal ARN of the entity to which you are granting permissions to, then choose the **Permissions** to grant.

7. Use the **Policy editor** to customize a preconfigured policy or to create your own policy based on your requirements. When you use the editor, the preconfigured policy options become unavailable. To undo your policy changes, choose **Clear**.

   When you clear the editor, the preconfigured policies become available once again.
8. After you complete editing or creating the policy, choose **Save**.

   The details page for the file system is displayed, showing the policy in **File system policy**.

   You can also create a file system policy programmatically using AWS CloudFormation, AWS SDKs, or the Amazon EFS API directly. For more information about creating file system policies, see Creating file system policies in the *Amazon EFS User Guide*.

Setting up Encryption of Data in Transit

To set up encryption of data in transit, we recommend that you download the EFS mount helper on each client. The EFS mount helper is an open-source utility that AWS provides to simplify using EFS, including setting up encryption of data in transit. The mount helper uses the EFS recommended mount options by default.

The EFS mount helper is supported on the following Linux distributions:

- Amazon Linux 2017.09+
- Amazon Linux 2+
- Debian 9+
- Fedora 28+
- Red Hat Enterprise Linux / CentOS 7+
- Ubuntu 16.04+

To set up encryption of data in transit:

1. Install the EFS mount helper:
   - For Amazon Linux, use this command:

     ```
     sudo yum install -y amazon-efs-utils
     ```
   - For other Linux distributions, download from GitHub and install.

The amazon-efs-utils package automatically installs the following dependencies: NFS client (nfs-utils), Network relay (stunnel), OpenSSL, and Python.

2. Mount the file system:

```
sudo mount -t efs -o tls file-system-id
efs-mount-point
```

- `mount -t efs` invokes the EFS mount helper.
- Using the DNS name of the file system or the IP address of a mount target is not supported when mounting using the EFS mount helper, use the file system id instead.
- The EFS mount helper uses the AWS recommended mount options by default. Overriding these default mount options is not recommended but we provide the flexibility to do so when the occasion arises. We recommend thoroughly testing any mount option overrides so you understand how these changes impact file system access and performance.
- The following table represents the default mount options used by the EFS mount helper.

Option	Description
nfsvers=4.1	The version of NFS protocol
rsize=1048576	TThe maximum number of bytes of data that the NFS client can receive for each network READ request)
wsize=1048576	The maximum number of bytes of data that the NFS

Option	Description
	client can send for each network WRITE request
hard	The recovery behavior of the NFS client after an NFS request times out, so that NFS requests are retried indefinitely until the server replies
timeo=600	The timeout value that the NFS client uses to wait for a response before it retries an NFS request in Deci seconds
retrans=2	The number of times the NFS client retries a request before it attempts further recovery action

Option	Description
noresvport	Tells the NFS client to use a new TCP source port when a network connection is reestablished

- Add the following line to `/etc/fstab` to automatically remount your file system after any system restart.

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

Using Encryption of Data in Transit

If your organization is subject to corporate or regulatory policies that require encryption of data in transit, we recommend using encryption of data in transit on every client accessing the file system. Encryption and decryption are configured at the connection level and add another layer of security.

Mounting the file system using the EFS mount helper sets up and maintains a TLS 1.2 tunnel between the client and Amazon EFS, and routes all NFS traffic over this encrypted tunnel. The certificate used to establish the encrypted TLS connection is signed by the Amazon Certificate Authority (CA) and trusted by most modern Linux distributions. The EFS mount helper also spawns a watchdog process to monitor all secure tunnels to each file system and ensures they are running.

After using the EFS mount helper to establish encrypted connections to Amazon EFS, no other user input or configuration is required. Encryption is transparent to user connections and applications accessing the file system.

After successfully mounting and establishing an encrypted connection to an EFS file system using the EFS mount helper, the output of a mount command shows the file system is mounted and an encrypted tunnel has been established using the localhost (127.0.0.1) as the network relay. See the following sample output.

```
      127.0.0.1:/ on efs-mount-point type nfs4

(rw,relatime,vers=4.1,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,port=20059,timeo=6
```

To map an `efs-mount-point` to an EFS file system, query the mount.log file in `/var/log/amazon/efs` and find the last successful mount operation. This can be done using the following simple grep command.

```
grep -E "Successfully
mounted.*efs-mount-point"
/var/log/amazon/efs/mount.log | tail -1
```

The output of this grep command will return the DNS name of the mounted EFS file system. See sample output below.

```
2018-03-15 07:03:42,363 - INFO - Successfully mounted
file-system-id.efs.region.amazonaws.com
at efs-mount-point
```

Conclusion

Amazon EFS file system data can be encrypted at rest and in transit. You can encrypt data at rest by using CMKs that you can control and manage using AWS KMS. Creating an encrypted file system is as simple as selecting a check box in the Amazon EFS file system creation wizard in the AWS Management Console, or adding a single parameter to the `CreateFileSystem` operation in the AWS CLI, AWS SDKs, or Amazon EFS API.

You can enforce encryption at rest and transit using AWS IAM identity-based policies and file system policies to further strengthen your security requirements and help meet your compliance needs. Using an encrypted file system is also transparent to services, applications, and users, with minimal effect on the file system's performance. You can encrypt data in transit by using the EFS mount helper to establish an encrypted TLS tunnel on each client, encrypting all NFS traffic between the client and the mounted EFS file system. Enforcing encryption of Amazon EFS data at rest using IAM identity policies and in transit using EFS file system policies is available to you at no additional cost.

Resources

- AWS KMS Cryptographic Details Whitepaper
- Amazon EFS User Guide

Document History and Contributors

Document History

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Minor update	Fix non-inclusive language.	April 6, 2022
Minor update	Bug fixes and numerous minor changes throughout.	April 1, 2022
Minor updates	Adjusted page layout.	April 30, 2021
Whitepaper updated	Added enforcement of encryption at-rest and in transit using IAM.	February 22, 2021
Whitepaper updated	Added encryption of data in transit.	April 1, 2018
Initial publication	Encrypt Data at Rest with Amazon EFS Encrypted File Systems published.	September 1, 2017

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser you are using.

Contributors

Contributors to this document include:

- Darryl S. Osborne, Storage Specialist Solutions Architect, AWS

- Joseph Travaglini, Senior Product Manager, Amazon EFS
- Peter Buonora, Principal Solutions Architect, AWS
- Siva Rajamani, Senior Solutions Architect, AWS