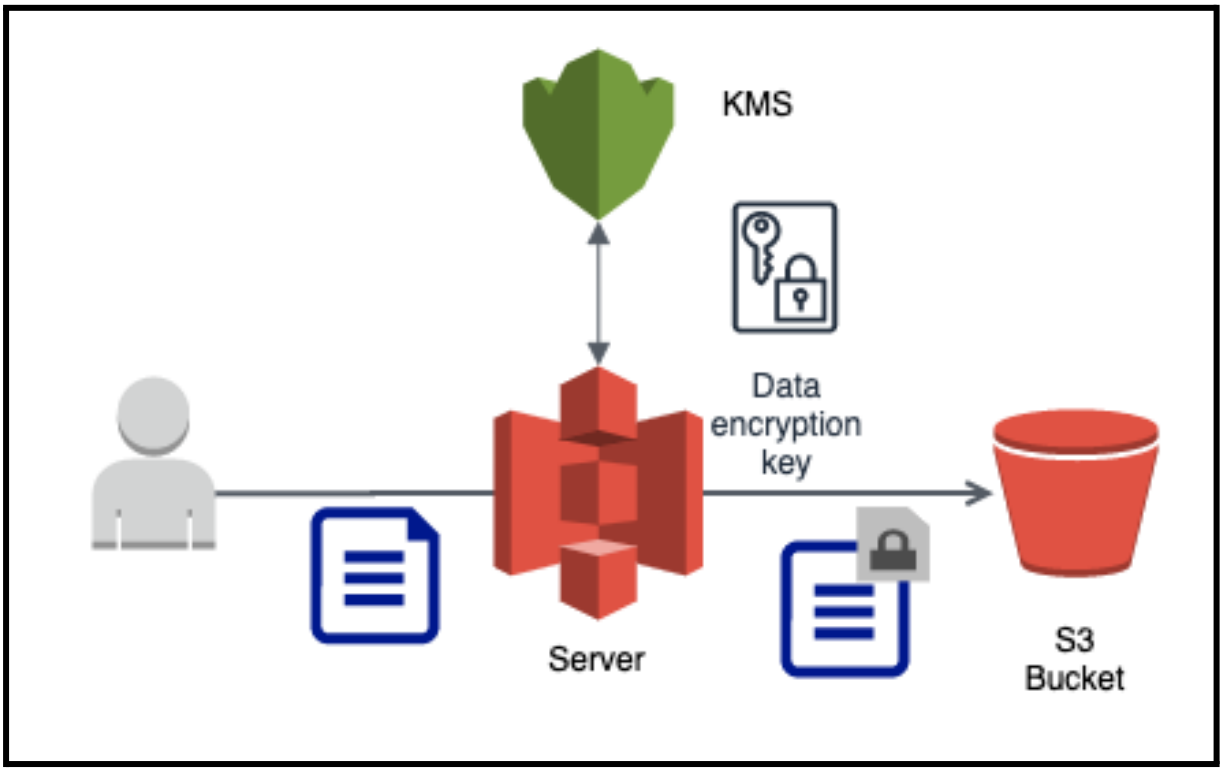
AIM : Configuring AWS S3 and KMS for Restricted Data Access



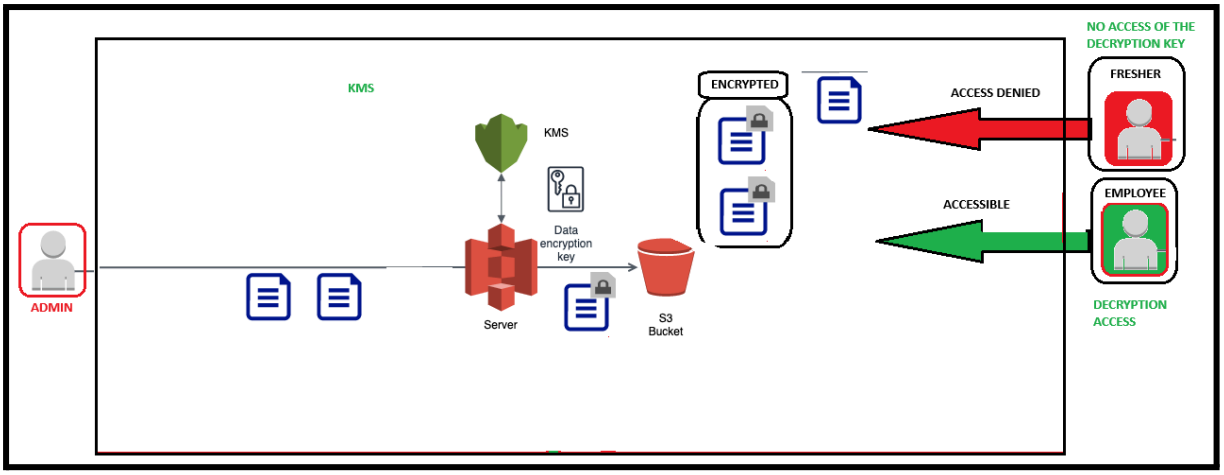
Introduction

This documentation provides a comprehensive guide to setting up a secure AWS environment using Amazon S3 for storage and AWS Key Management Service (KMS) for encryption. The scenario outlines creating two IAM users with S3 read-only access and configuring selective encryption on S3 objects using a symmetric KMS key.

Objective

The primary goal is to securely store data in an S3 bucket where:

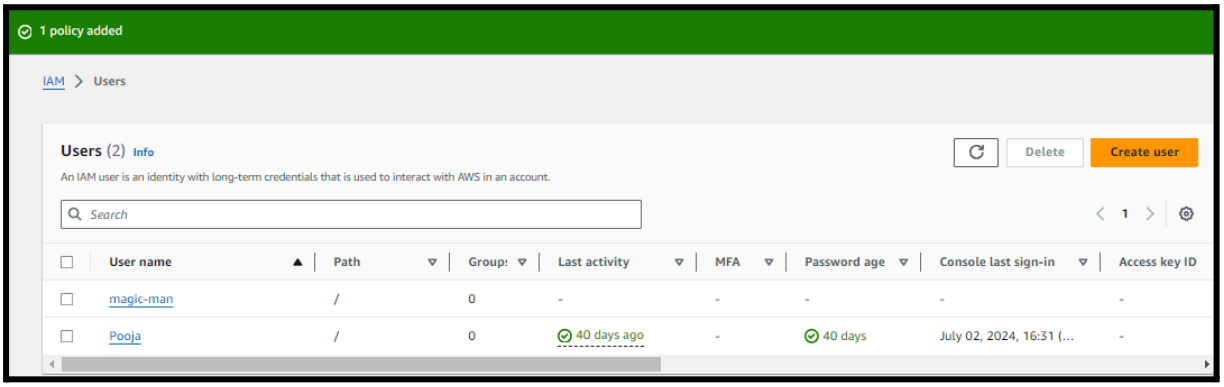
- Two IAM users (USER1 and USER2) have read-only access to all objects in the bucket.
- Some objects are encrypted using a symmetric KMS key, which both Admin and User1 can decrypt.
- The setup ensures compliance with security policies while enabling effective data access and protection.



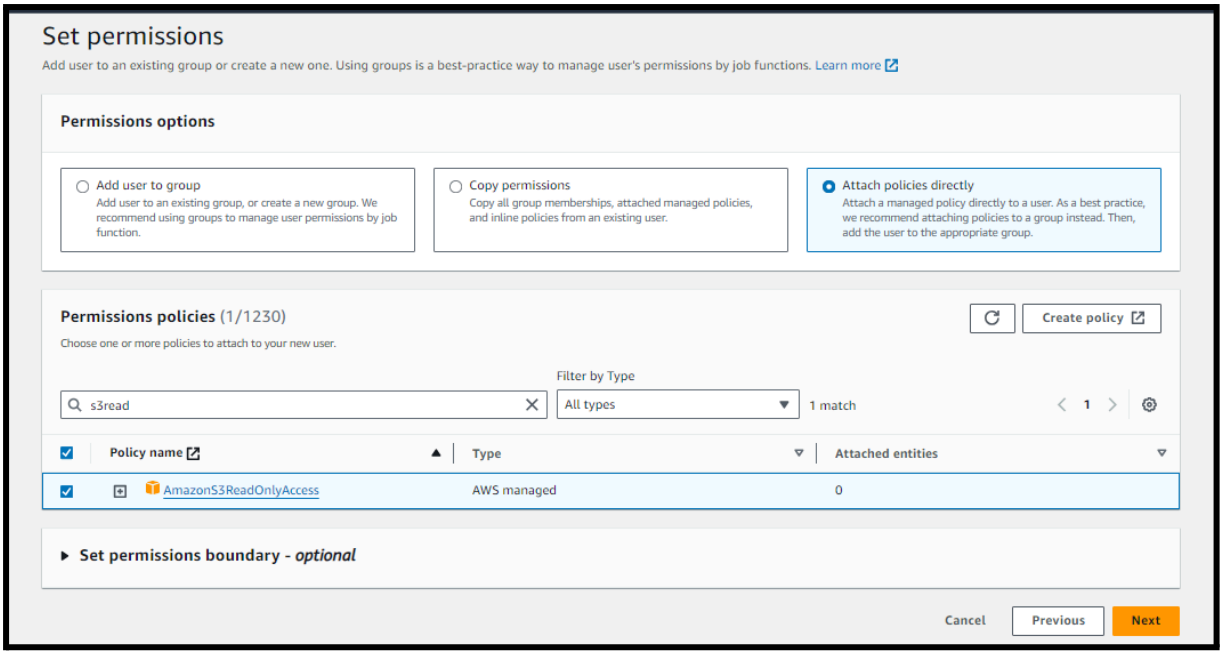
Setup Configuration

Step 1: Creating IAM Users

1. **Login to AWS Management Console** and navigate to the IAM dashboard.
2. **Create two users (Admin and User1)**:
    - Click on “Add user”.

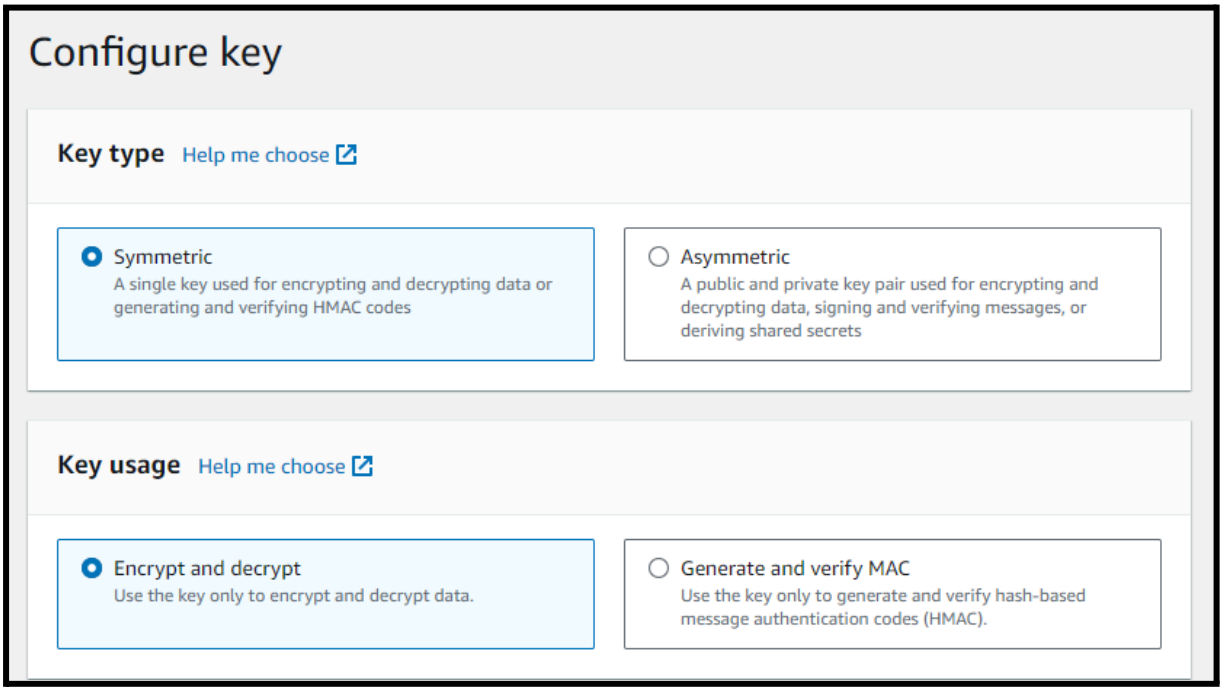


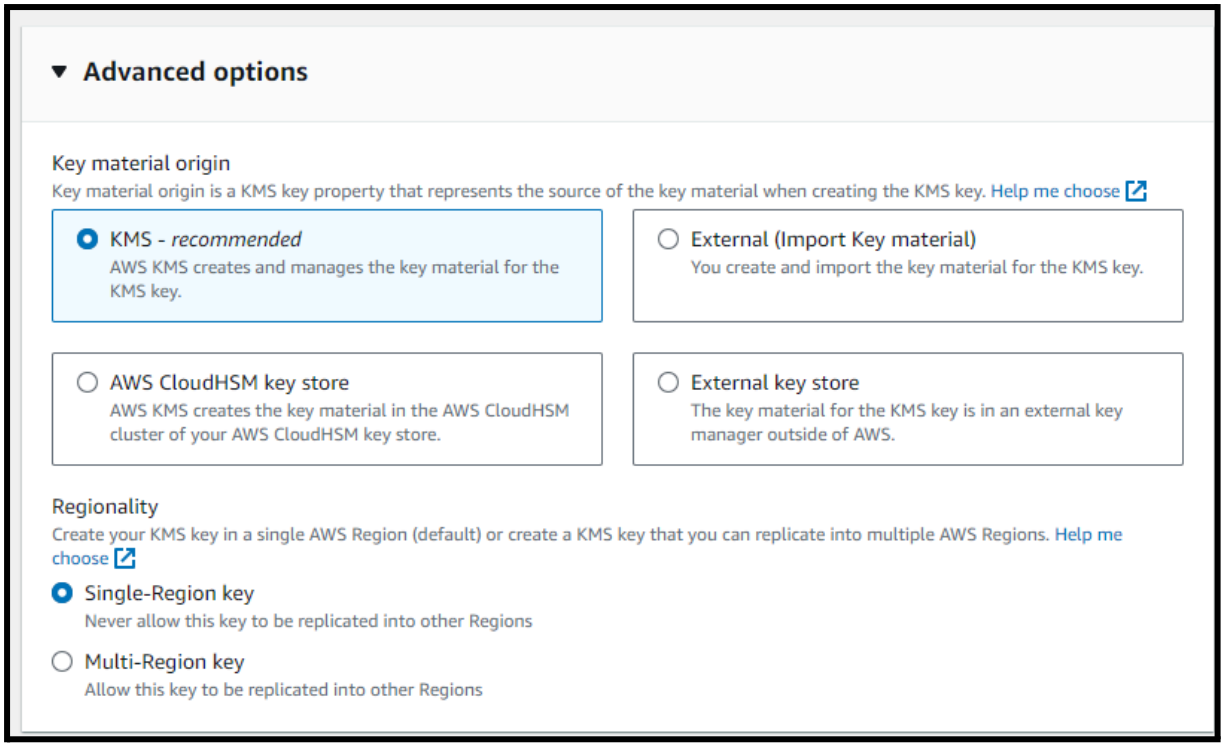
    - Enter user names and select both Programmatic and AWS Management Console access types.
    - Assign AmazonS3ReadOnlyAccess to ensure they have read-only access to S3 buckets.



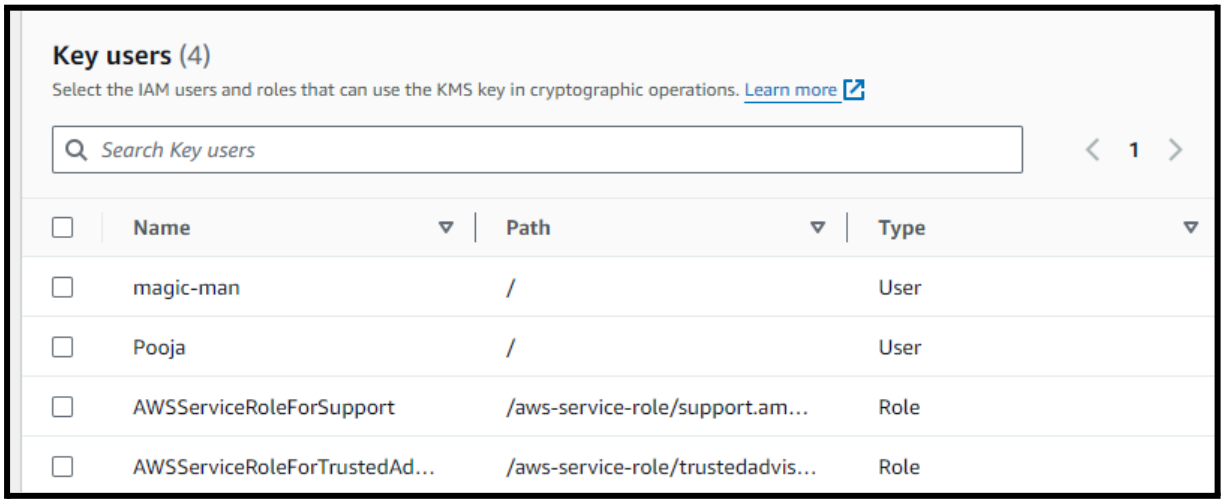
Step 2: Setting Up AWS KMS

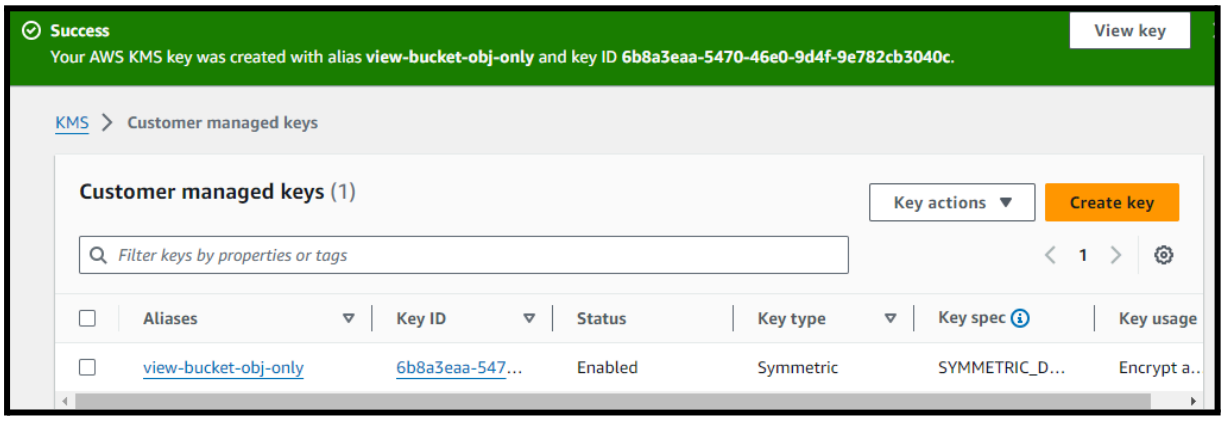
1. **Navigate to the KMS section** in the IAM dashboard.
2. **Create a new symmetric key**:
    - Choose “Create a key” and select “Symmetric”.
    - Set an alias and description for the key.
    - Configure permissions to allow only Admin and User1 to use this key for decryption purposes.





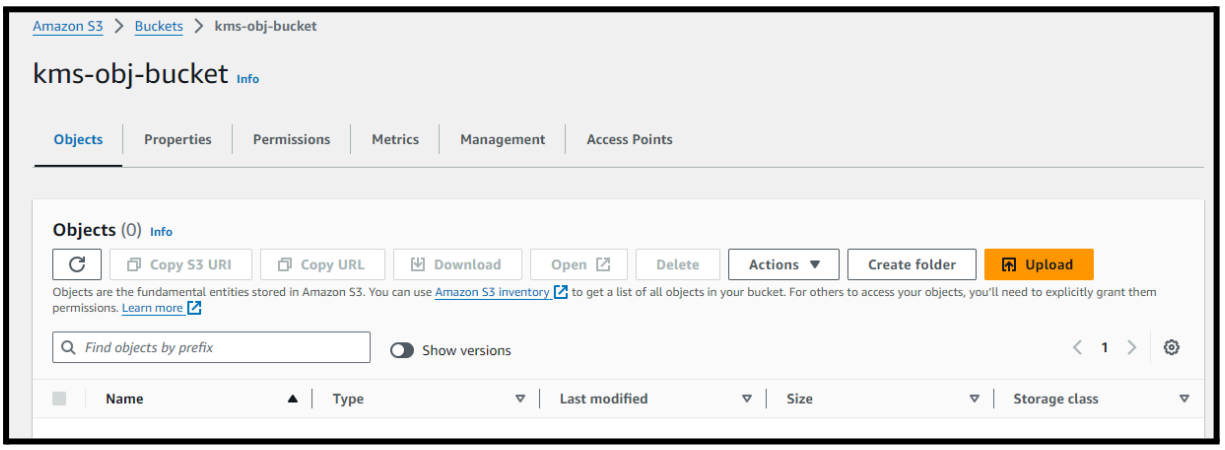
Keep this field empty for now

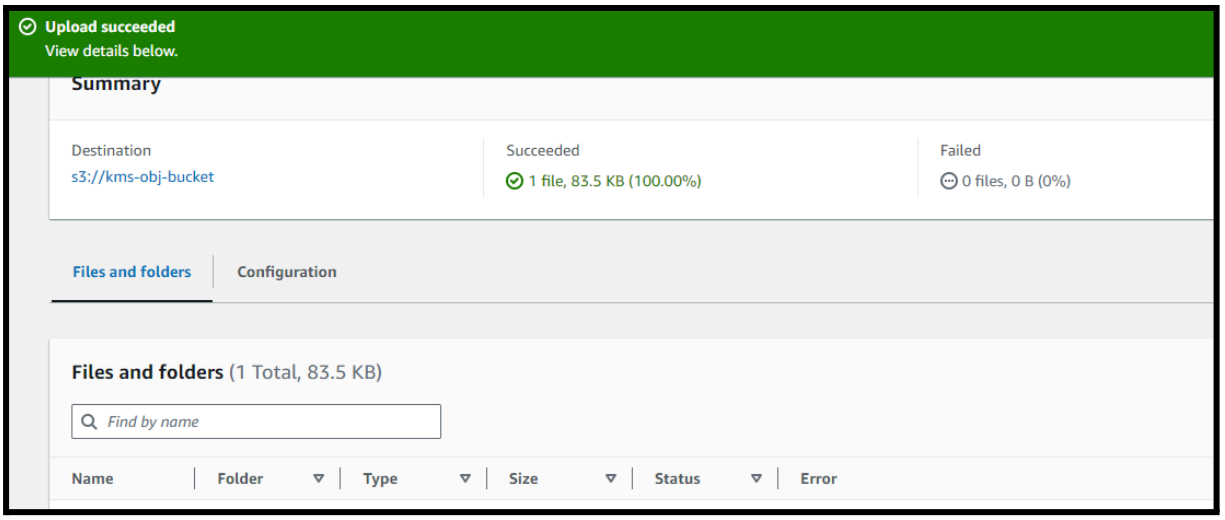




Step 3: Creating and Configuring an S3 Bucket

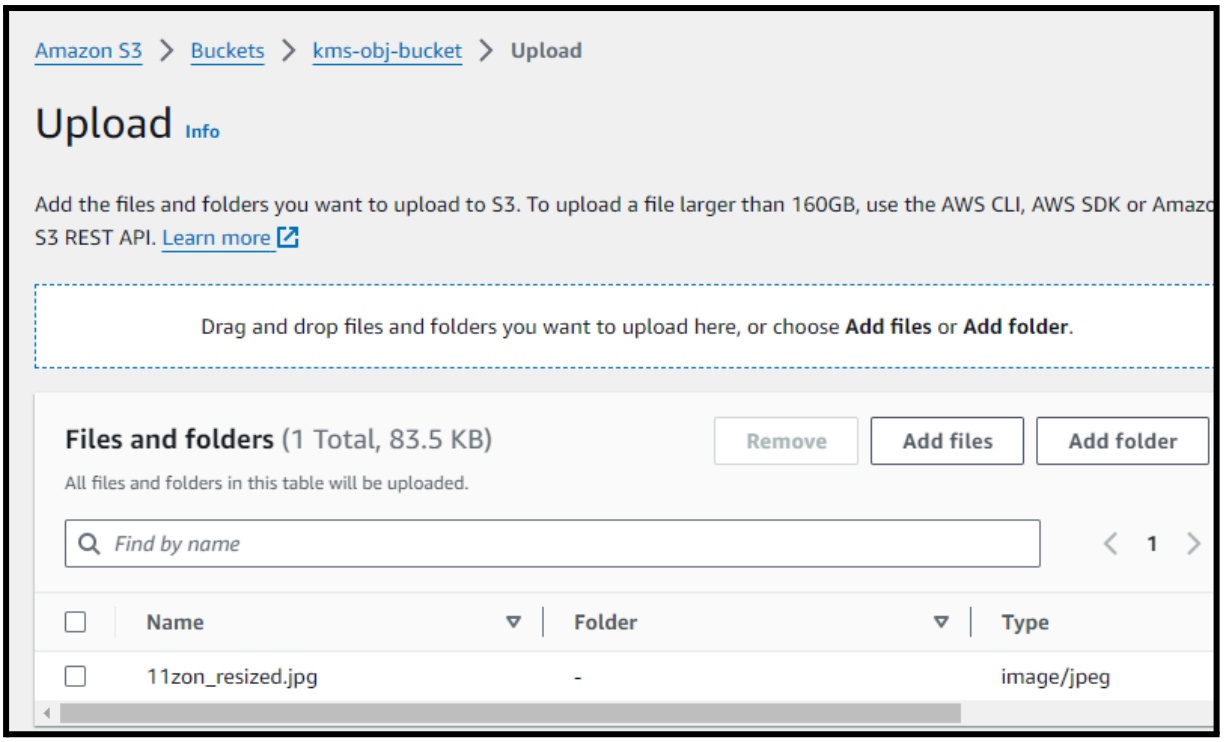
1. **Go to the S3 service** in the AWS Management Console.
2. **Create a new bucket**:
    - Click “Create bucket”.
    - Follow the setup to name the bucket and select the appropriate region.
    - Ensure all public access is blocked to enhance security.

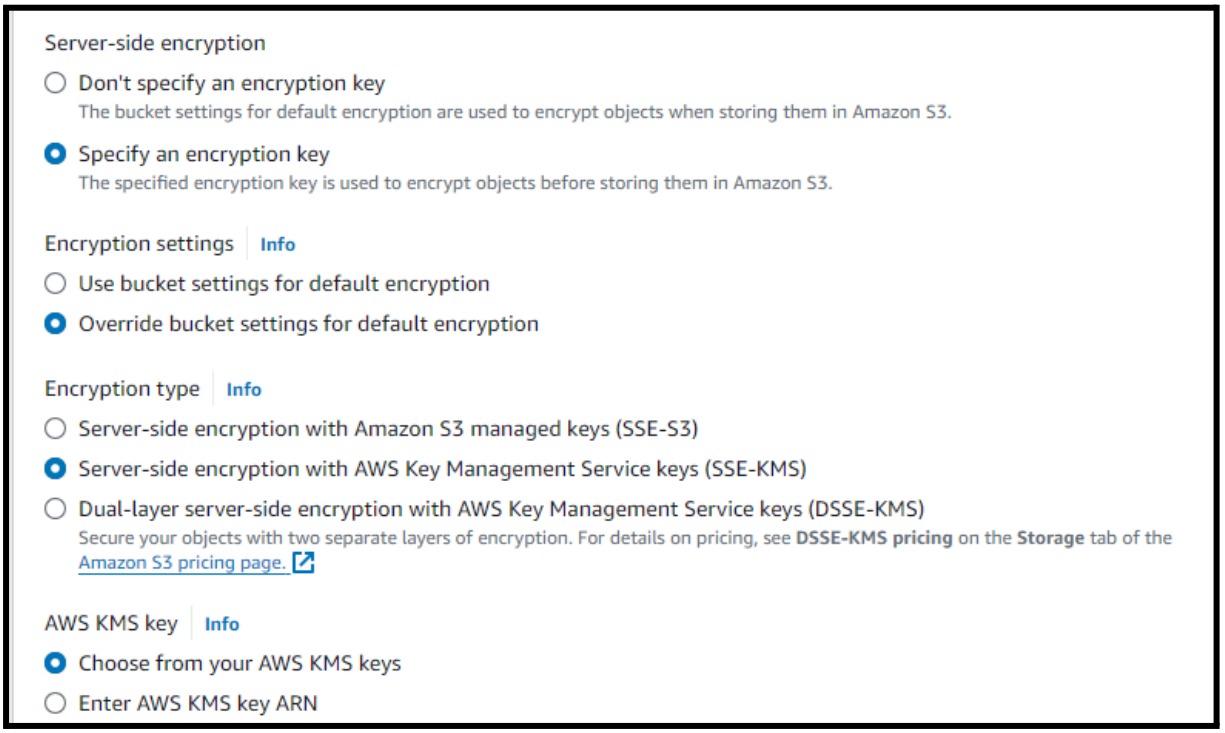


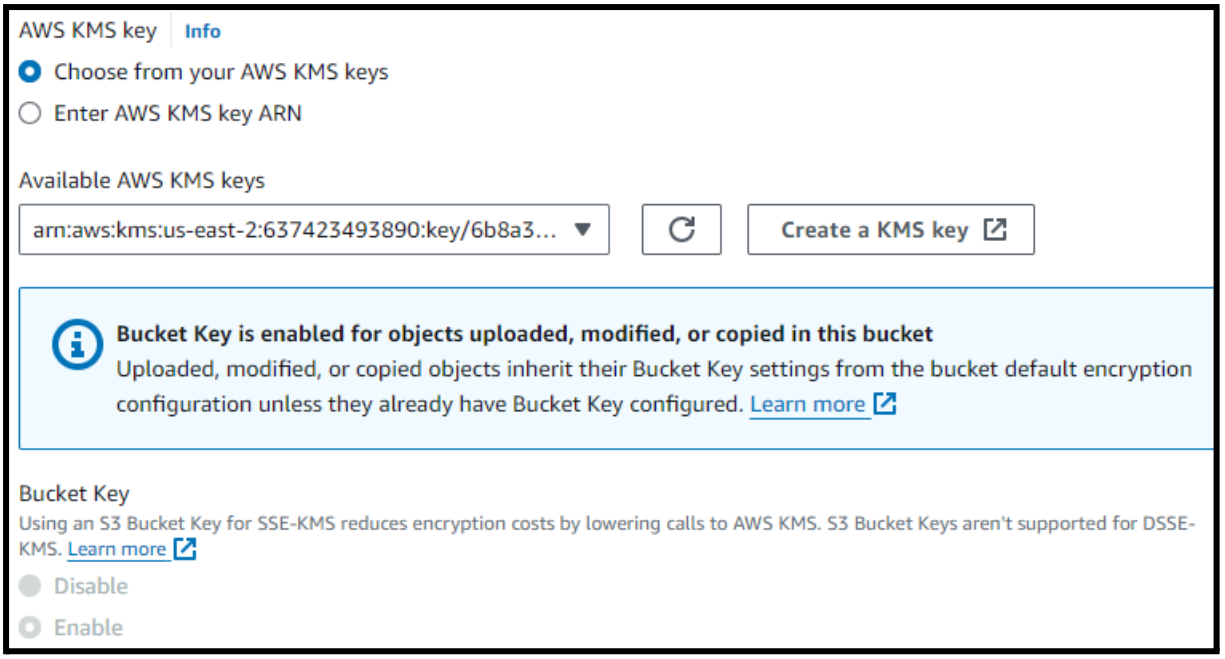


Step 4: Configuring Encryption on S3 Objects

1. **Manually encrypt specific objects using the KMS key**:
    - During the upload process, select the encryption option.
    - Choose the previously created KMS key for encryption to secure sensitive files.

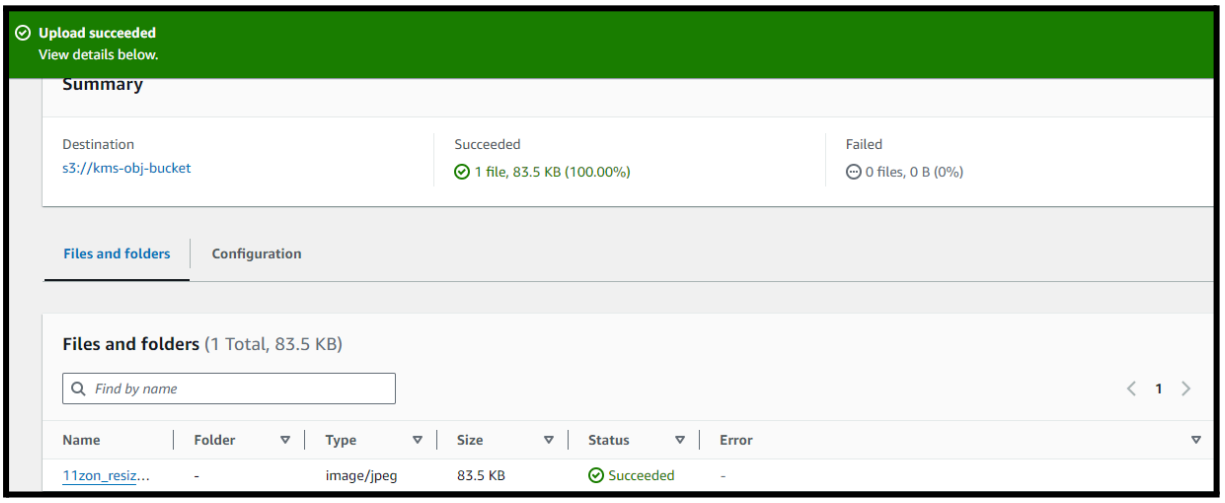






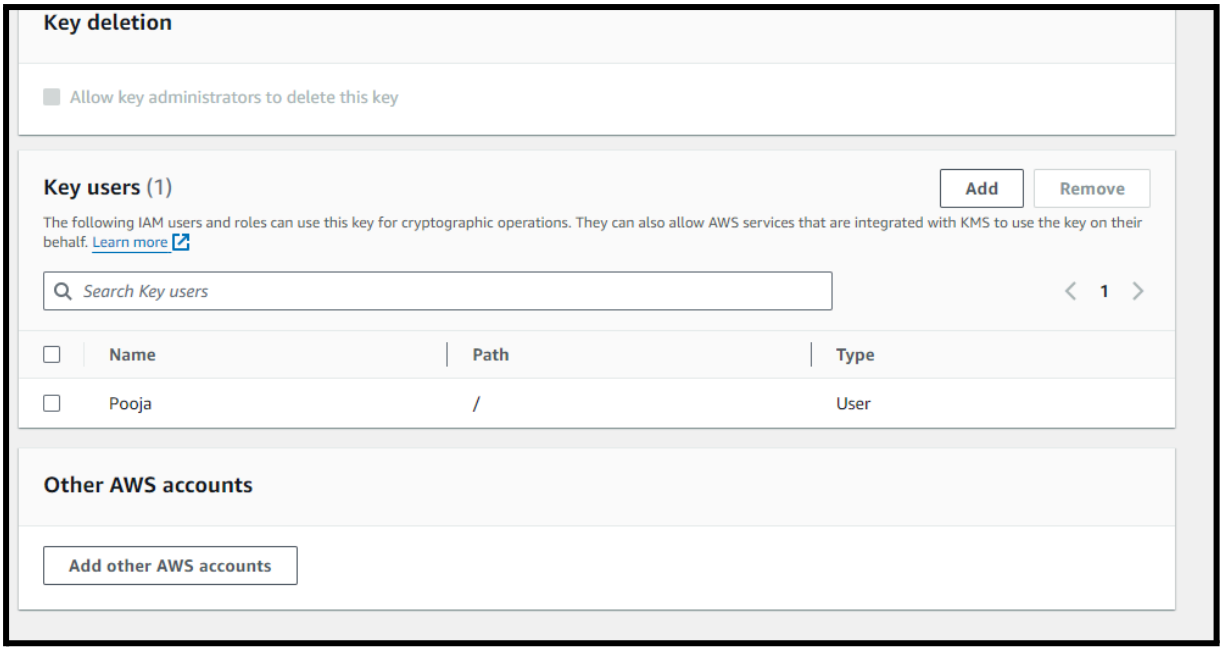
Step 5: Uploading Data

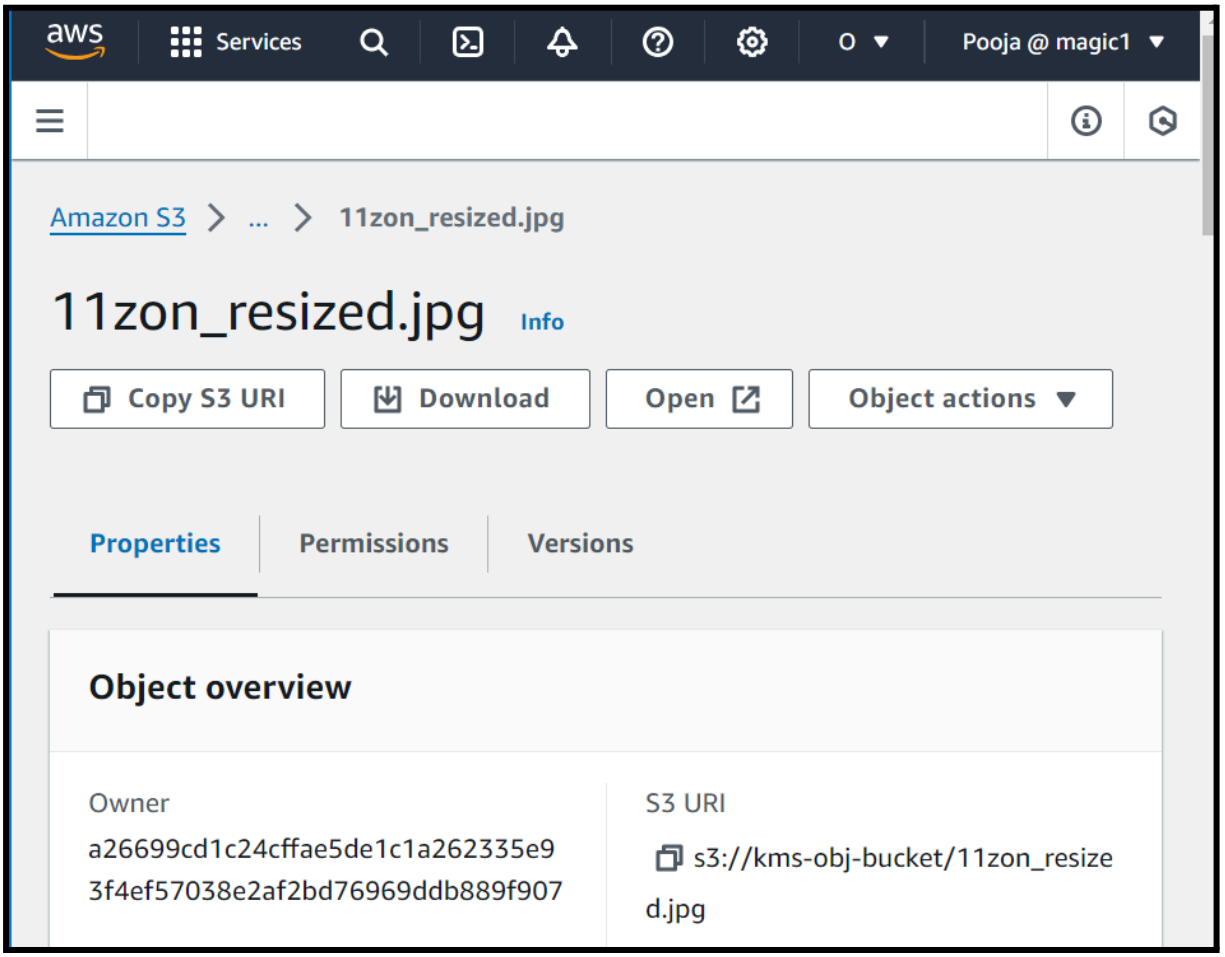
1. **Upload both encrypted and unencrypted objects**:
    - Admin, as the primary user, should upload files to the S3 bucket.
    - Selectively apply encryption to sensitive files during the upload.

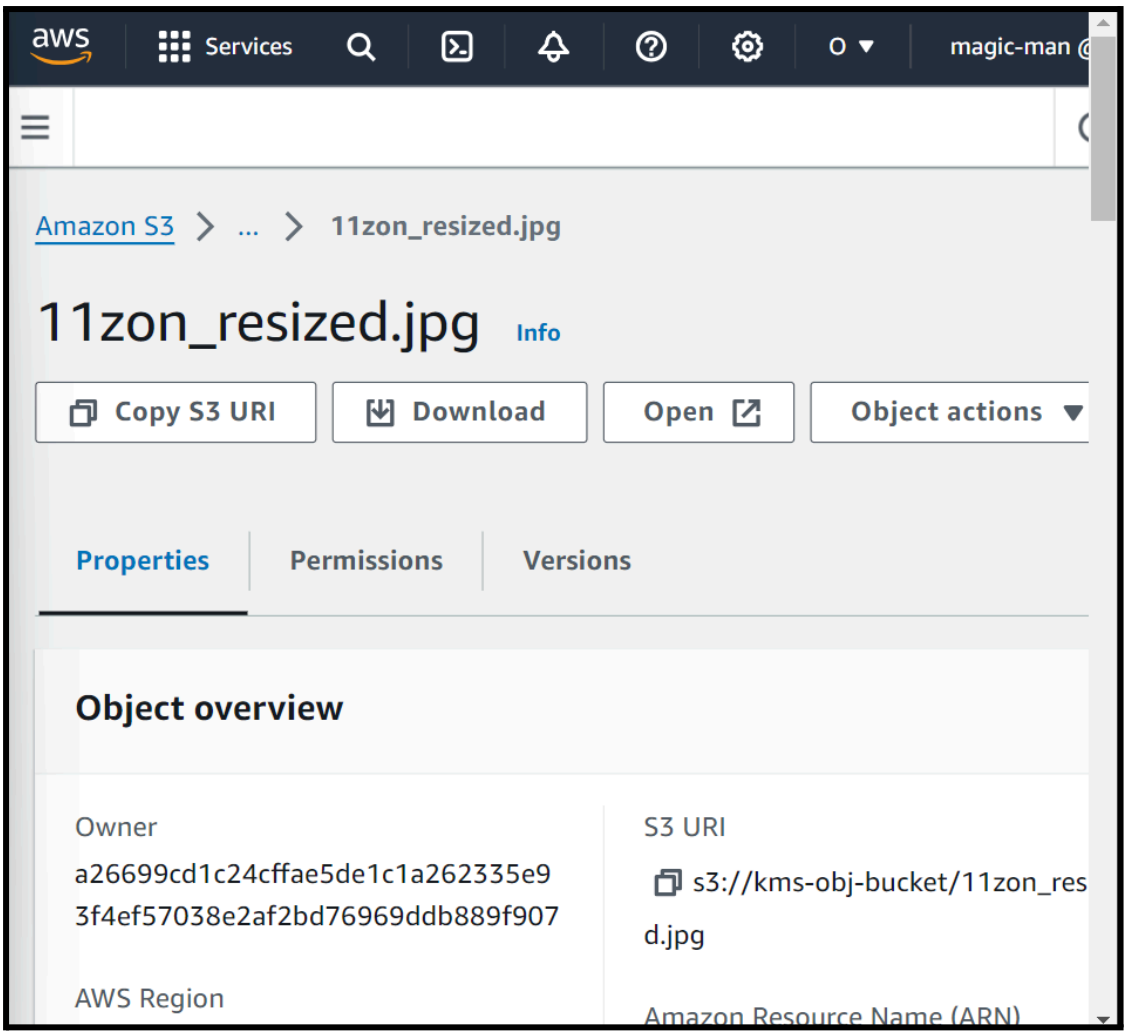


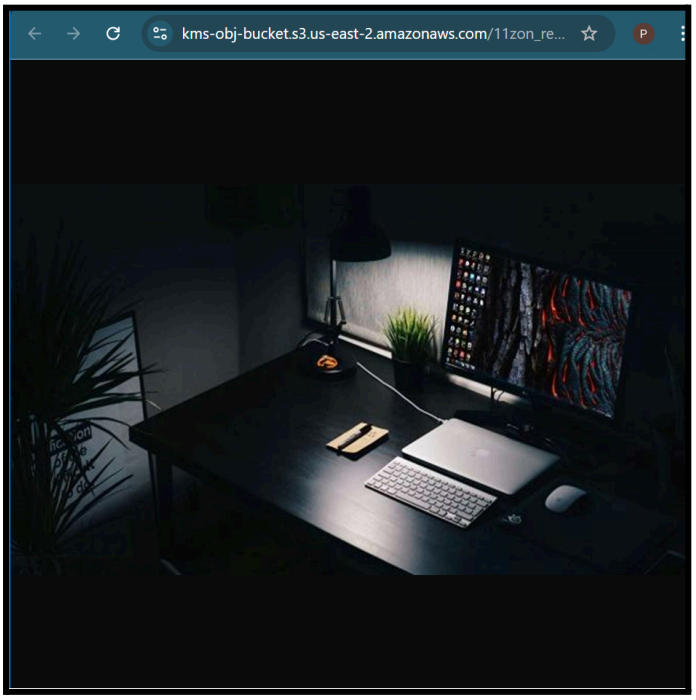
User Access and Permissions

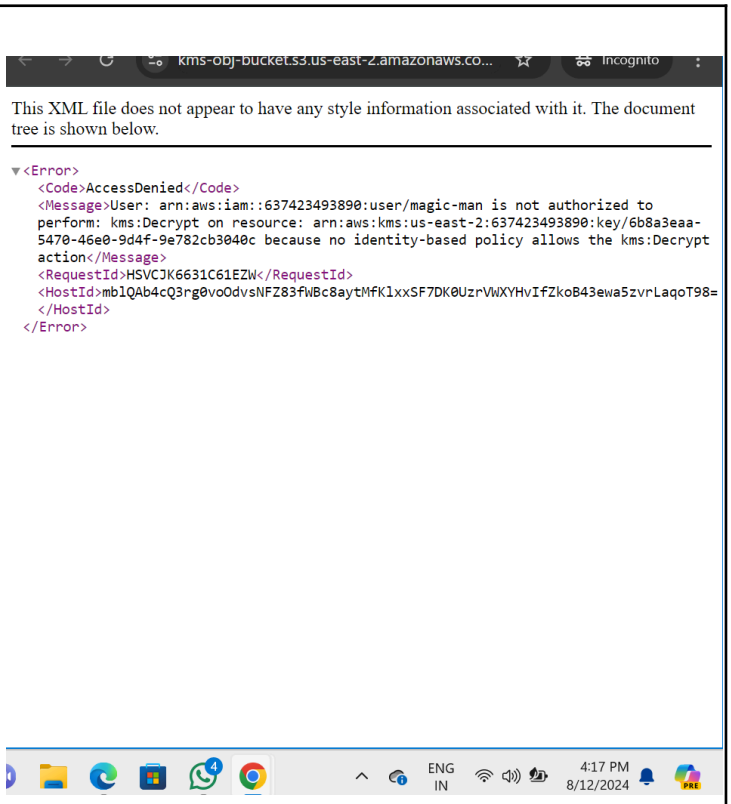
- **Configure IAM roles and policies** to ensure that both user1 and User2 can read all objects.
- **Set decryption permissions** using the KMS key for both user1 to access encrypted files securely.


Key deletion
Allow key administrators to delete this key
Key users (1)
Add
Remove
The following IAM users and roles can use this key for cryptographic operations. They can also allow AWS services that are integrated with KMS to use the key on their behalf. Learn more
Search Key users
1
Name
Path
Type
Pooja
/
User
Other AWS accounts
Add other AWS accounts


aws
Services
Pooja @ magic1
Amazon S3
...
11zon_resized.jpg
11zon_resized.jpg
Info
Copy S3 URI
Download
Open
Object actions
Properties
Permissions
Versions
Object overview
Owner
a26699cd1c24cffae5de1c1a262335e9
3f4ef57038e2af2bd76969ddb889f907
S3 URI
s3://kms-obj-bucket/11zon_resize
d.jpg


aws
Services
magic-man @
Amazon S3
...
11zon_resized.jpg
11zon_resized.jpg
Info
Copy S3 URI
Download
Open
Object actions
Properties
Permissions
Versions
Object overview
Owner
a26699cd1c24cffae5de1c1a262335e9
3f4ef57038e2af2bd76969ddb889f907
AWS Region
S3 URI
s3://kms-obj-bucket/11zon_res
d.jpg
Amazon Resource Name (ARN)

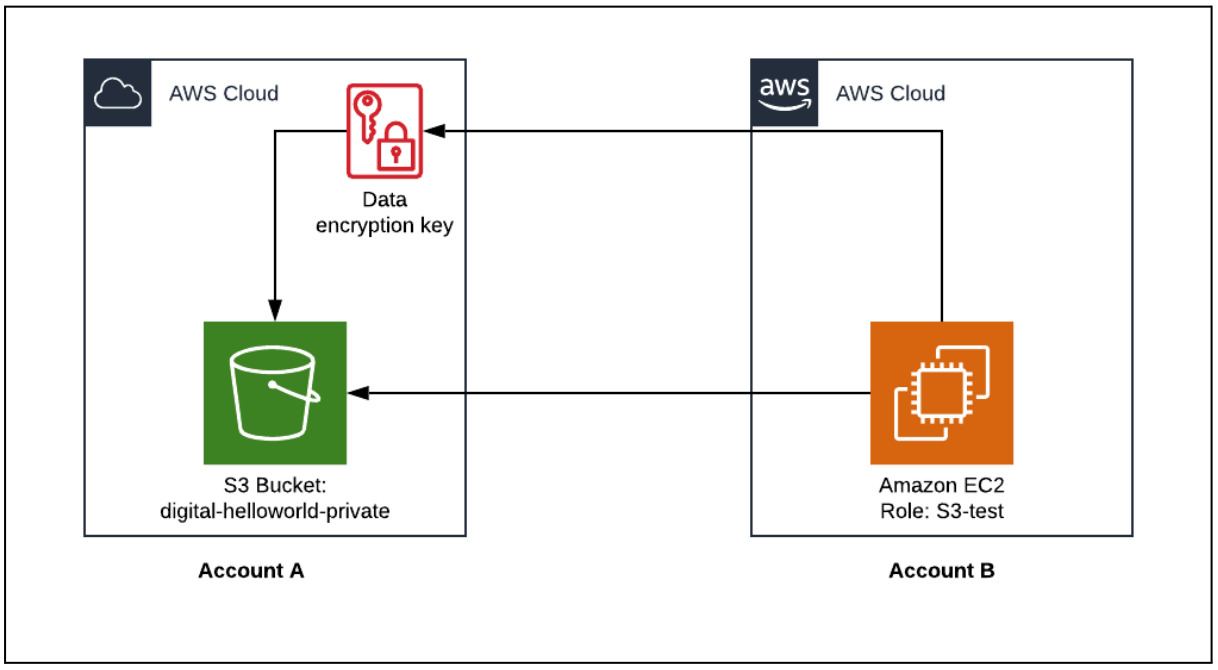




Conclusion

organisations can effectively manage data security in AWS using S3 and KMS. The read-only access coupled with selective encryption provides a robust security framework while maintaining straightforward access for authorised users.

AIM : Secure Cross-Account Data Sharing in AWS Using S3, IAM, and KMS



Amazon EC2 instance in one AWS account (Account B) needs access to a privately encrypted S3 bucket in another account (Account A). This is a common use case in environments where different teams or divisions manage resources separately but need to share data securely.

Objective

To securely share encrypted data stored in an S3 bucket in Account A with an EC2 instance in Account B, using AWS Identity and Access Management (IAM) roles and AWS Key Management Service (KMS) for encryption key access.

Architecture Overview

- **Account A** contains an S3 bucket (digital-helloworld-private) that uses a data encryption key managed by AWS KMS for encrypting data at rest.
- **Account B** has an EC2 instance with an IAM role (S3-test) which needs to access the encrypted data in Account A's S3 bucket.

Steps to Implement

Step 1: Configure IAM Roles and Policies

- **Account A**:
    1. Create a KMS key for S3 bucket encryption. Attach a key policy that allows the S3-test role in Account B to use this key for decryption.
    2. Modify the bucket policy of digital-helloworld-private to permit the S3-test role from Account B to perform s3:GetObject on the bucket.
- **Account B**:
    1. Create an IAM role (S3-test) with permissions to access S3. Attach policies that grant access to the S3 bucket in Account A and use the KMS key for decryption.
    2. Attach this IAM role to the EC2 instance.

Step 2: Enable Cross-Account Access

- Use AWS Resource Access Manager (RAM) if applicable, or ensure that trust relationships are configured correctly in IAM roles and policies to allow the EC2 instance in Account B to access resources in Account A.

Step 3: Data Encryption and Access

- Encrypt data in the S3 bucket using the specified KMS key in Account A before it is stored.
- Access the data from the EC2 instance in Account B by utilizing the assigned IAM role, which should include permissions to use the KMS key for decryption.

Step 4: Monitoring and Logging

- Implement AWS CloudTrail and AWS CloudWatch in both accounts to monitor access and operations on the S3 bucket and the EC2 instance, ensuring all accesses are logged and auditable.

Conclusion

This setup ensures that sensitive data stored in an S3 bucket can be securely shared between different AWS accounts while maintaining stringent security measures through encryption and controlled access. This approach not only

maximises data security but also adheres to principles of least privilege by restricting access to what is necessary.

Real-Life Corporate Use Case

In a corporate environment, such as a multinational corporation with multiple independent departments, the need to share sensitive financial reports or customer data securely between departments is common.

For instance, a finance department (Account A) could store encrypted budget reports in an S3 bucket, which need to be accessed by a data analytics team using an EC2 instance (Account B) for real-time financial analysis and forecasting.

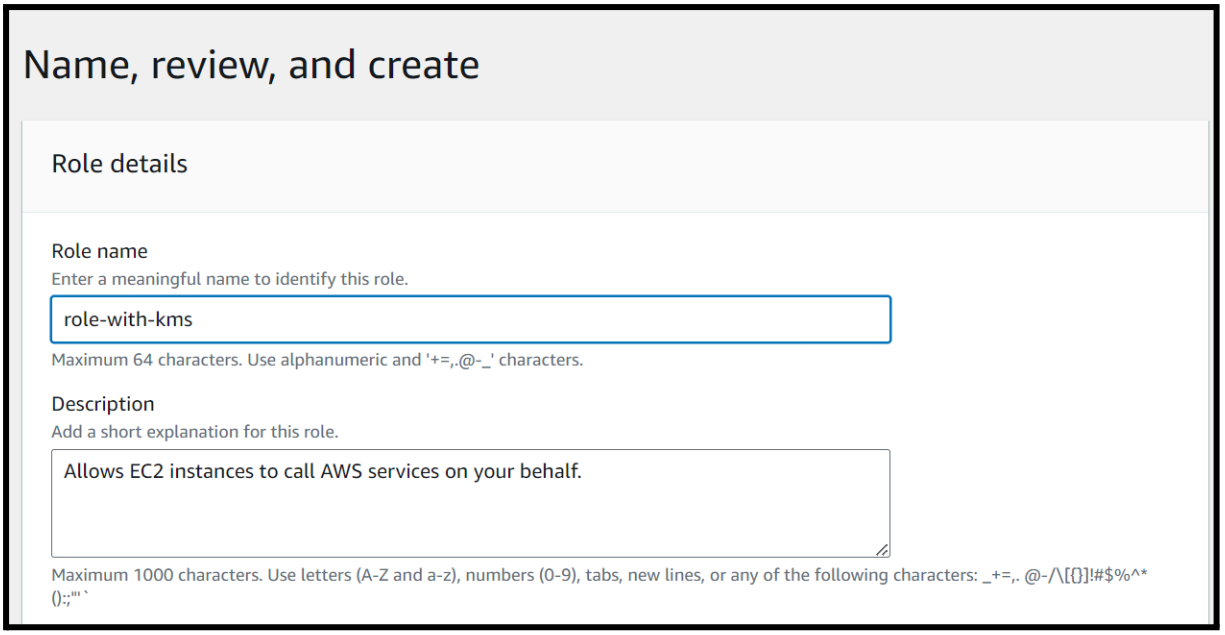
This architecture facilitates secure data sharing while complying with corporate governance and data privacy laws.

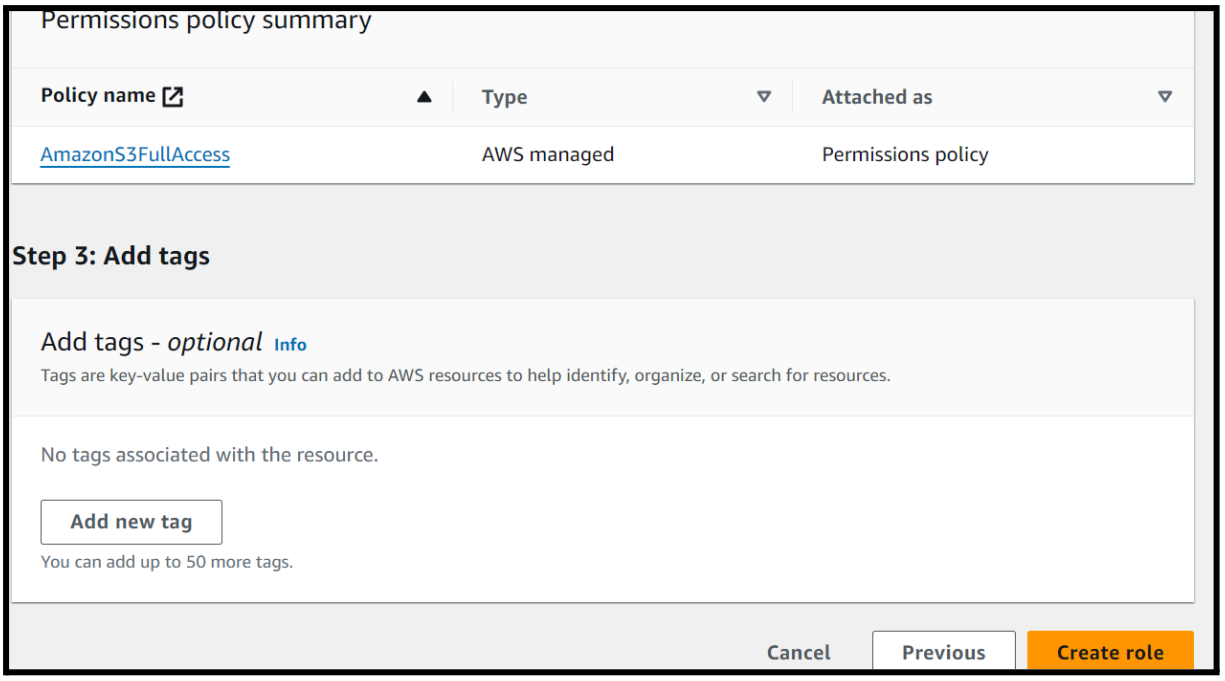
Securing Data with AWS KMS and OpenSSL on EC2

date : 09-8-24

AIM : KMS (KEY MANAGEMENT SERVICE)

Create IAM user and role :





Configure key

Key type Help me choose

- (●) **Symmetric**
  A single key used for encrypting and decrypting data or generating and verifying HMAC codes
- ( ) **Asymmetric**
  A public and private key pair used for encrypting and decrypting data, signing and verifying messages, or deriving shared secrets

Key usage Help me choose

- (●) **Encrypt and decrypt**
  Use the key only to encrypt and decrypt data.
- ( ) **Generate and verify MAC**
  Use the key only to generate and verify hash-based message authentication codes (HMAC).

▼ Advanced options

Key material origin

Key material origin is a KMS key property that represents the source of the key material when creating the KMS key. Help me choose

- ( ) **KMS - *recommended***
  AWS KMS creates and manages the key material for the KMS key.
- (●) **External (Import Key material)**
  You create and import the key material for the KMS key.
- ( ) **AWS CloudHSM key store**
  AWS KMS creates the key material in the AWS CloudHSM cluster of your AWS CloudHSM key store.
- ( ) **External key store**
  The key material for the KMS key is in an external key manager outside of AWS.

You can import key material from your key management infrastructure into AWS KMS and use it like any other AWS KMS key.

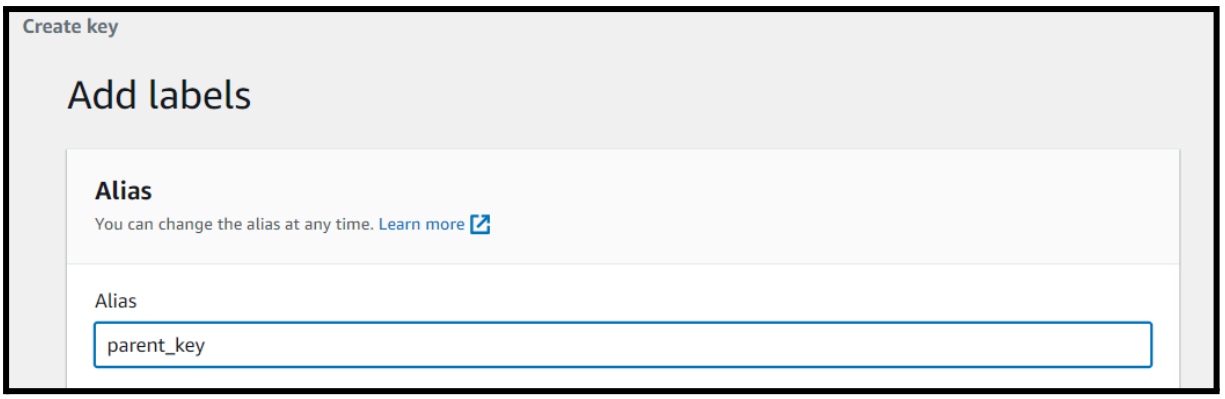
[ ] I understand the security and durability implications of using an imported key.

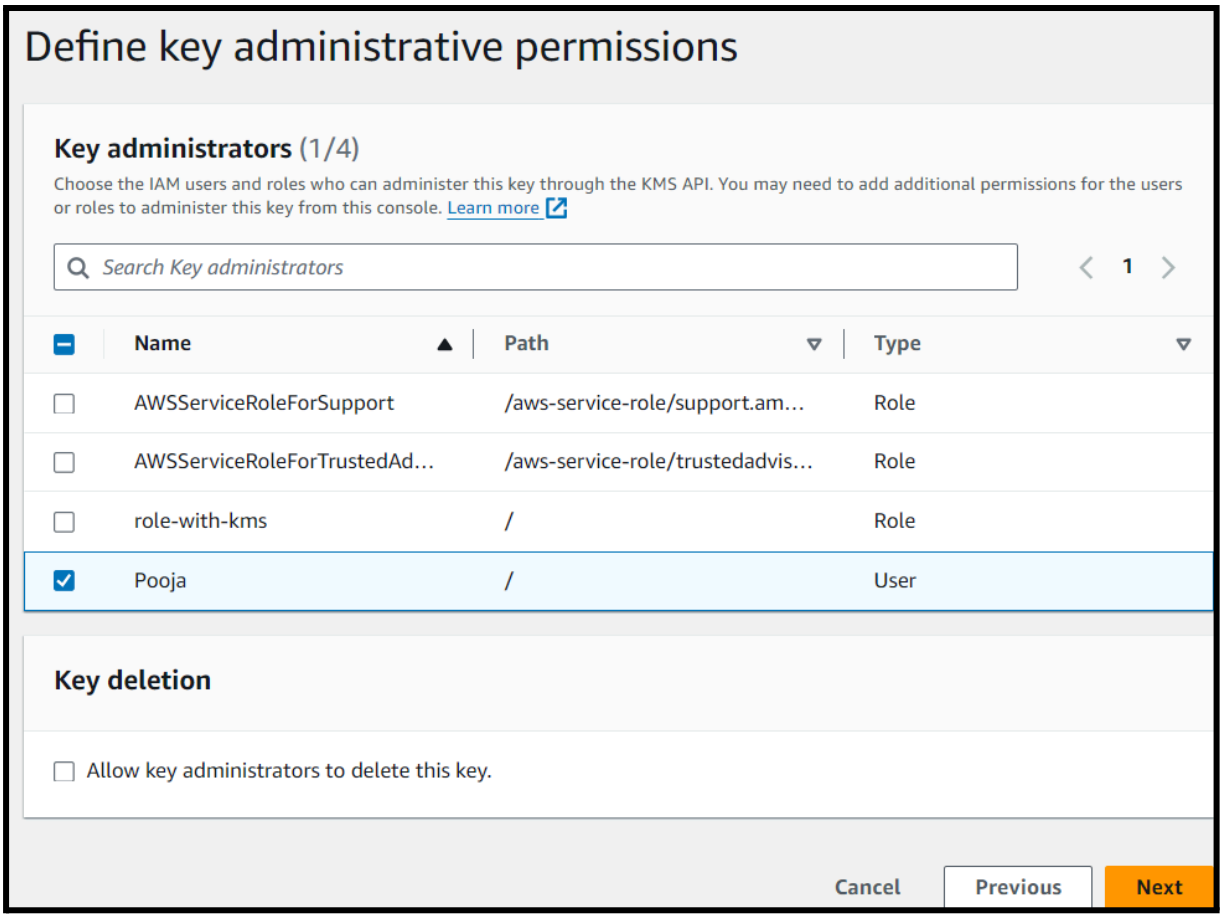
Regionality

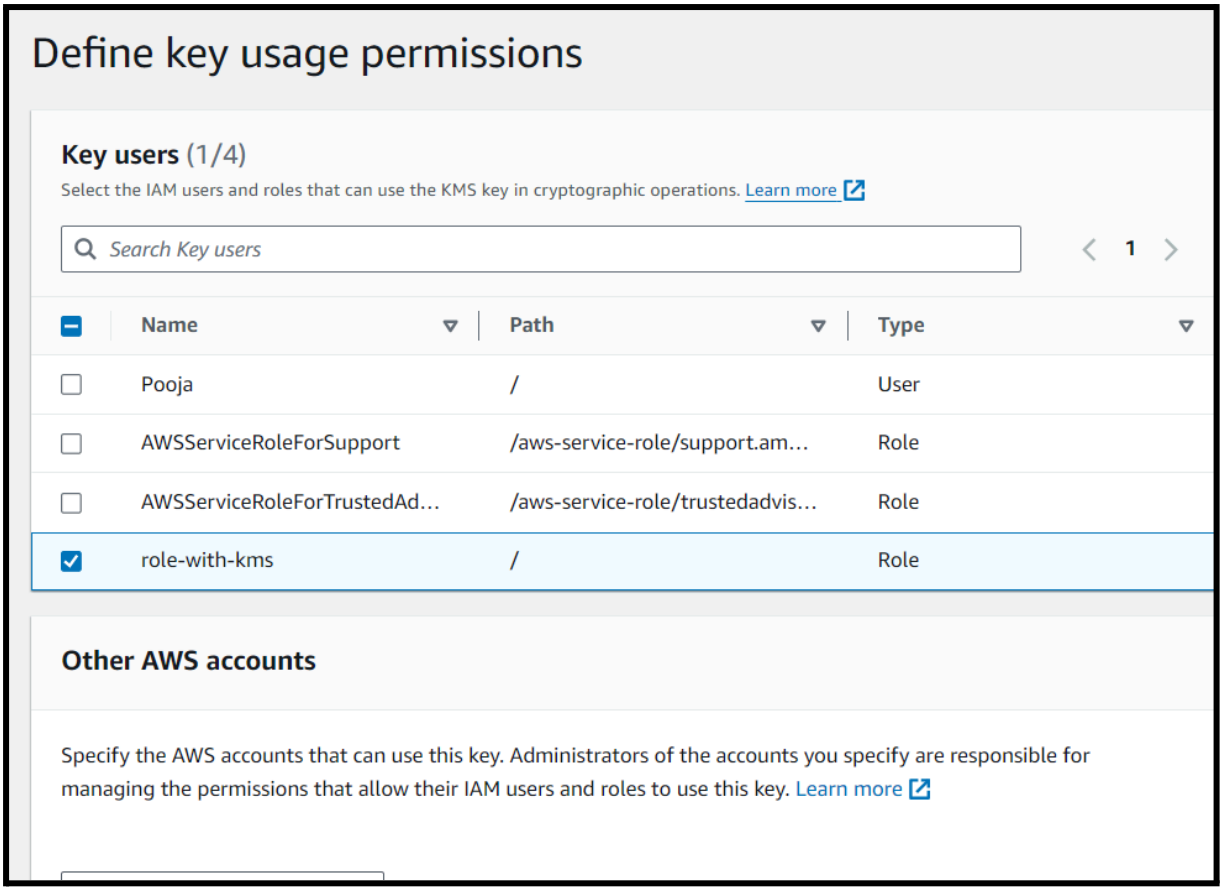
Create your KMS key in a single AWS Region (default) or create a KMS key that you can replicate into multiple AWS Regions. Help me choose

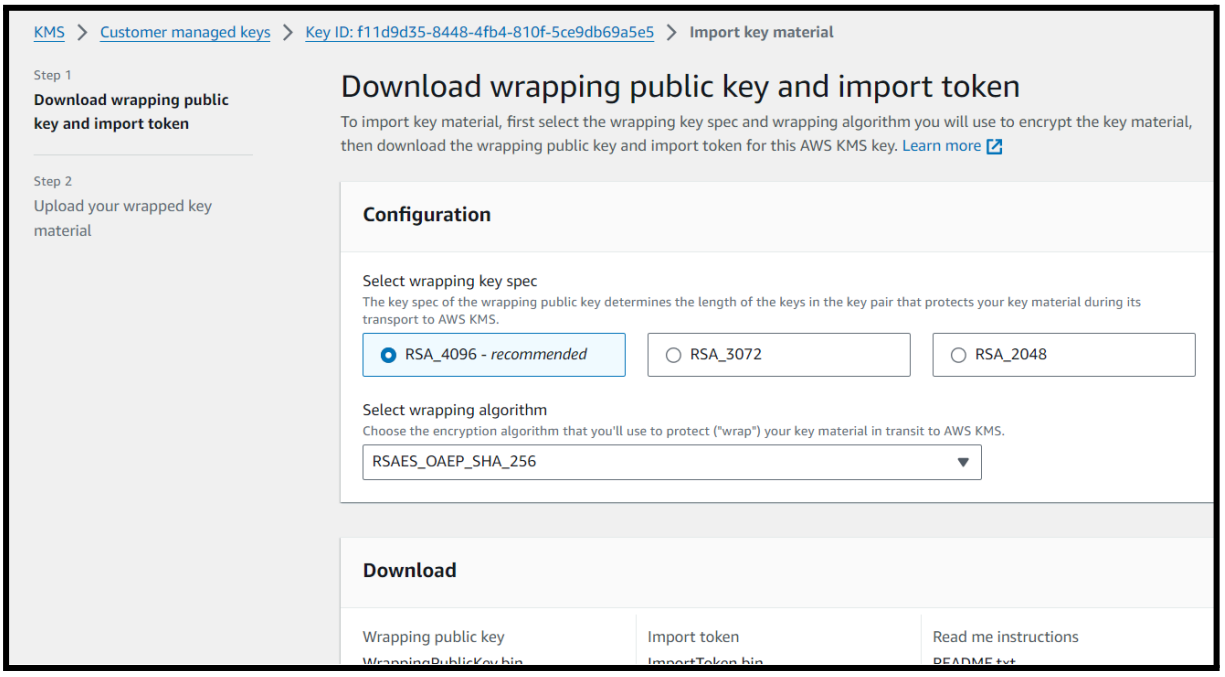
- (●) **Single-Region key**
  Never allow this key to be replicated into other Regions
- ( ) **Multi-Region key**
  Allow this key to be replicated into other Regions

Cancel Next

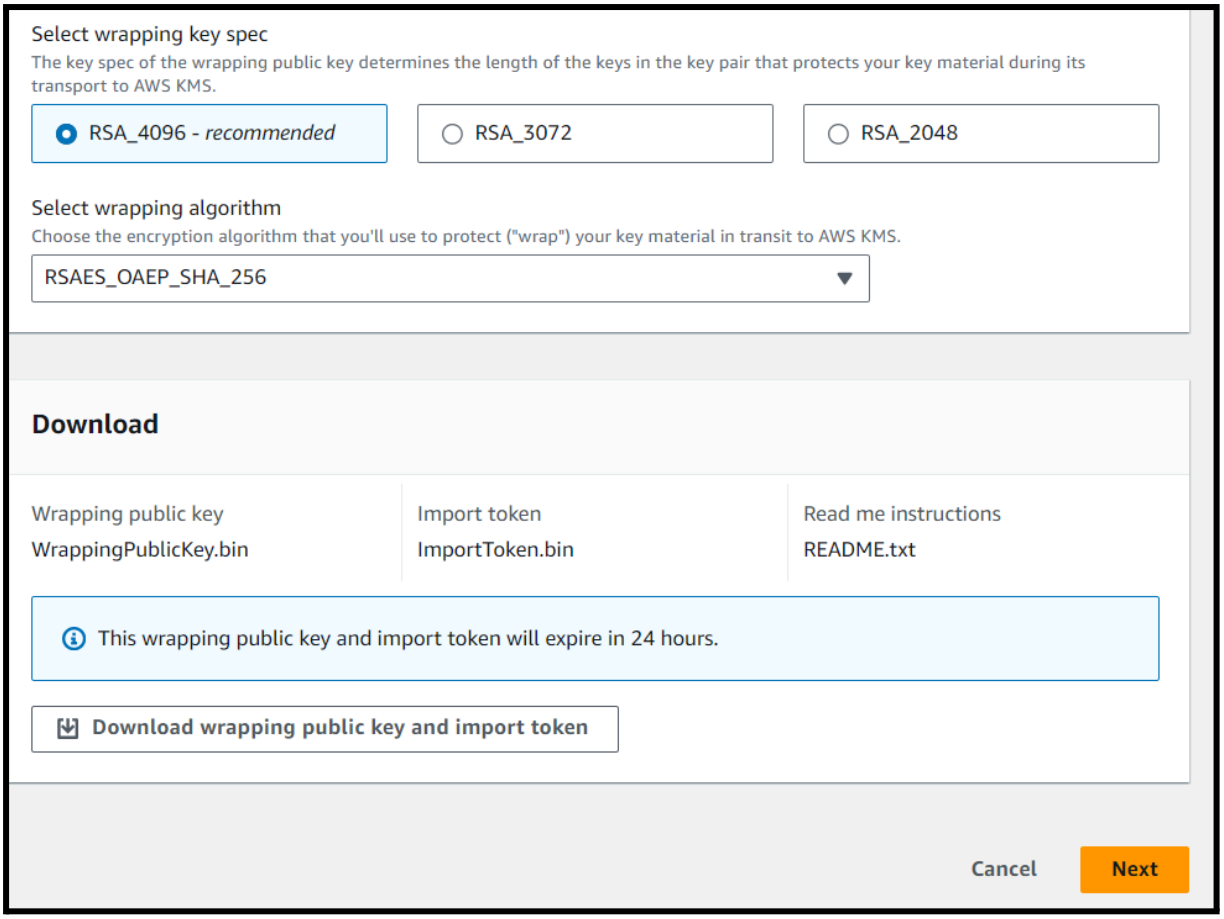

Create key
Add labels
Alias
You can change the alias at any time. Learn more
Alias
parent_key


Define key administrative permissions
Key administrators (1/4)
Choose the IAM users and roles who can administer this key through the KMS API. You may need to add additional permissions for the users or roles to administer this key from this console. Learn more
Search Key administrators
1
Name
Path
Type
AWSServiceRoleForSupport
/aws-service-role/support.am...
Role
AWSServiceRoleForTrustedAd...
/aws-service-role/trustedadvis...
Role
role-with-kms
/
Role
Pooja
/
User
Key deletion
Allow key administrators to delete this key.
Cancel
Previous
Next

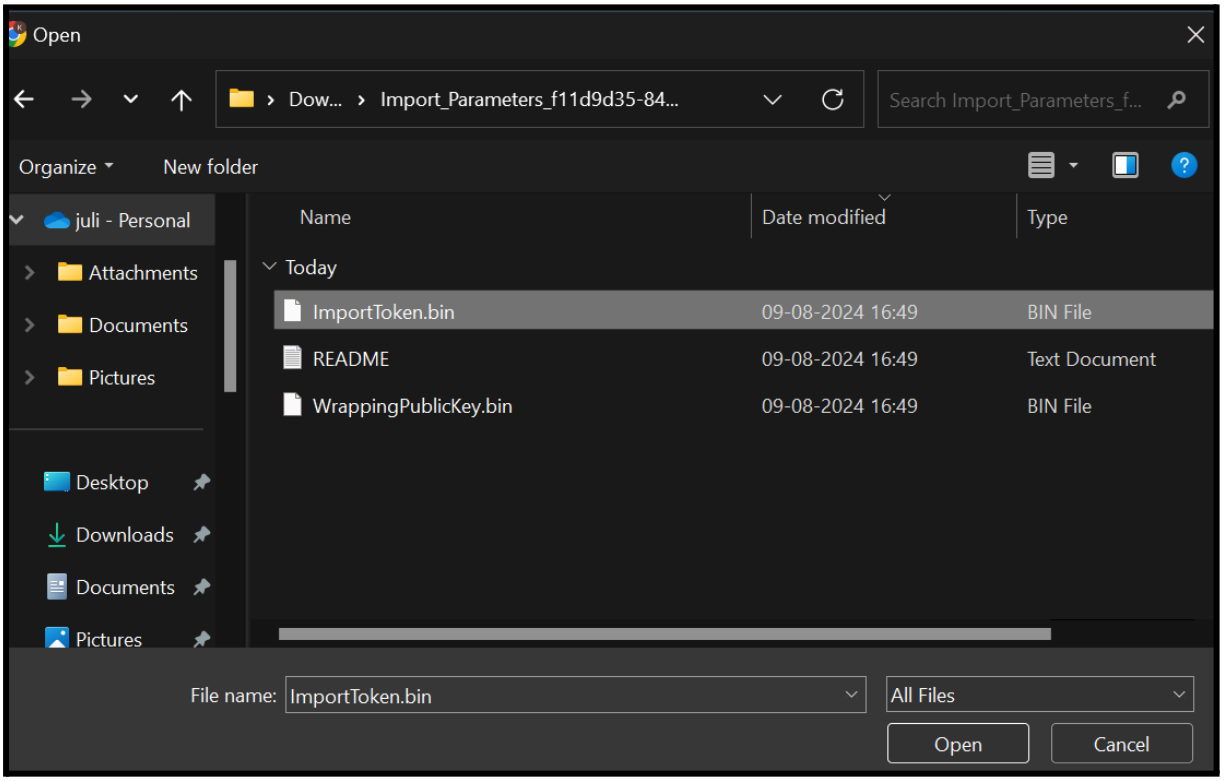




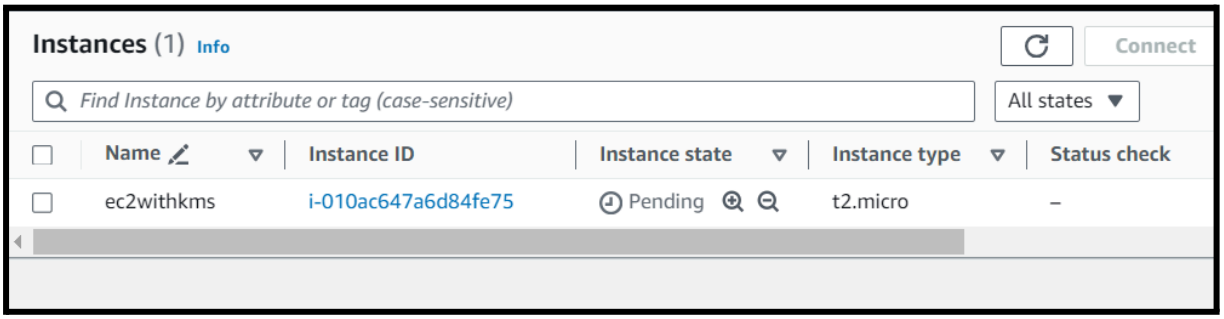
Download key



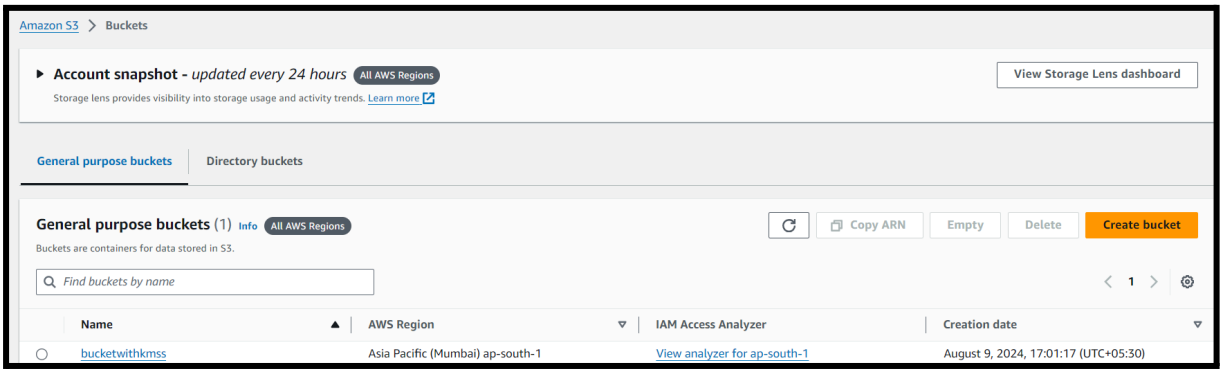
Import token



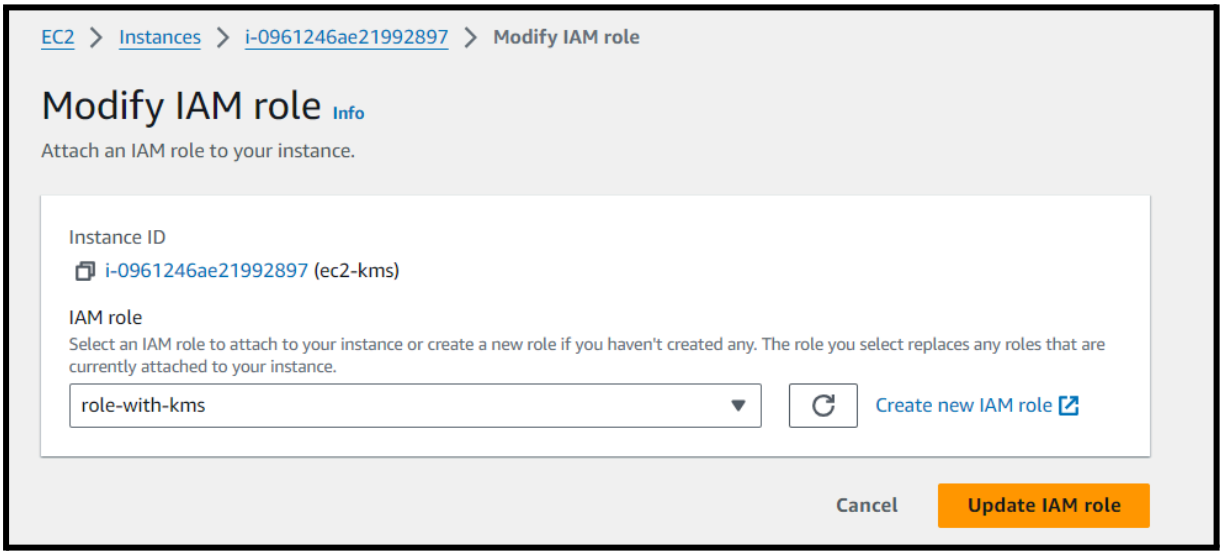
Launch an EC2

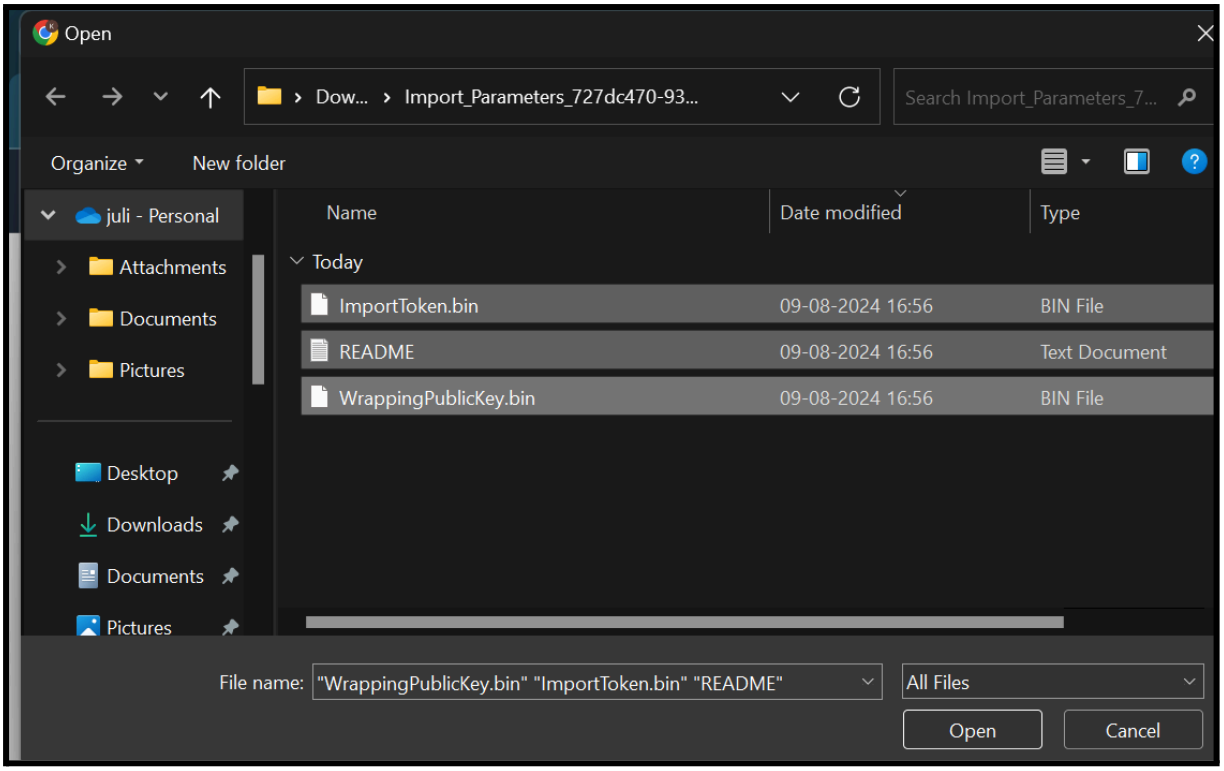


Create S3 bucket

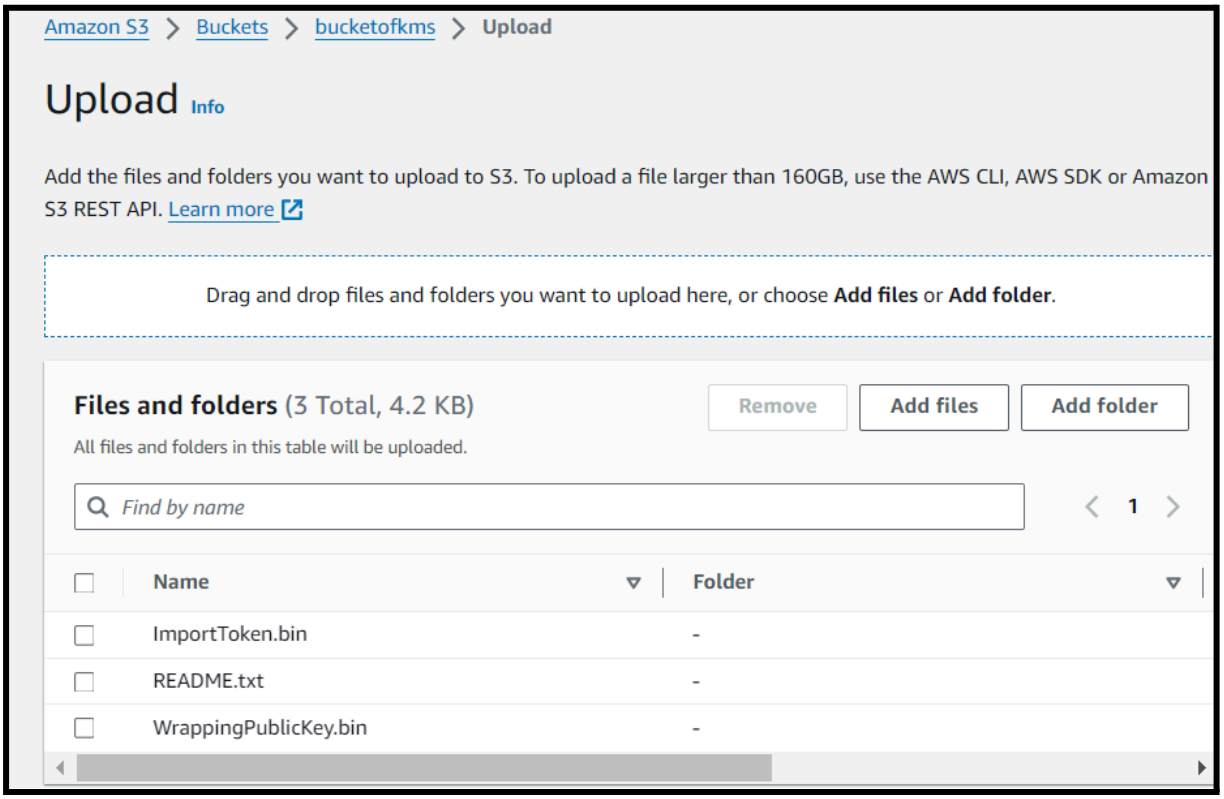


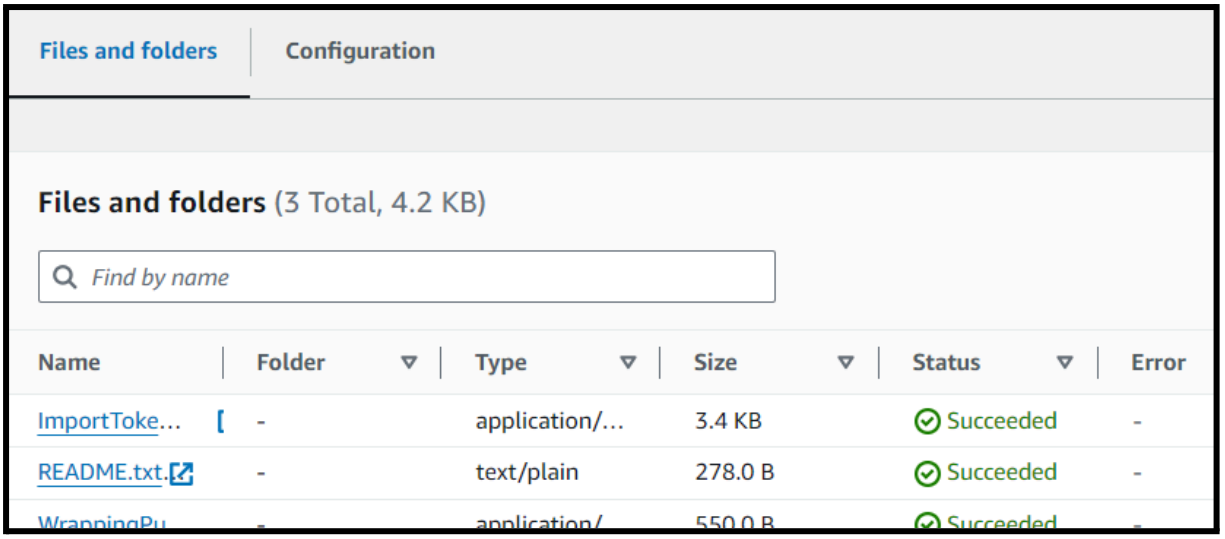
Connect EC2 instance




Open
Dow... › Import_Parameters_727dc470-93...
Search Import_Parameters_7...
Organize
New folder
juli - Personal
Attachments
Documents
Pictures
Desktop
Downloads
Documents
Pictures
Name
Date modified
Type
Today
ImportToken.bin
09-08-2024 16:56
BIN File
README
09-08-2024 16:56
Text Document
WrappingPublicKey.bin
09-08-2024 16:56
BIN File
File name: "WrappingPublicKey.bin" "ImportToken.bin" "README"
All Files
Open
Cancel

Upload in s3


Amazon S3 > Buckets > bucketofkms > Upload
Upload Info
Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more
Drag and drop files and folders you want to upload here, or choose Add files or Add folder.
Files and folders (3 Total, 4.2 KB)
Remove
Add files
Add folder
All files and folders in this table will be uploaded.
Find by name
1
Name
Folder
ImportToken.bin
-
README.txt
-
WrappingPublicKey.bin
-




```
Complete!
[root@ip-172-31-0-12 ec2-user]# yum upgrade -y
Last metadata expiration check: 0:01:06 ago on Fri Aug  9 11:49:29 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-0-12 ec2-user]# open ssl version
bash: open: command not found
[root@ip-172-31-0-12 ec2-user]# openssl version
OpenSSL 3.0.8 7 Feb 2023 (Library: OpenSSL 3.0.8 7 Feb 2023)
[root@ip-172-31-0-12 ec2-user]# pwd
/home/ec2-user
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketwithkmss /home/ec2-user --recursive
fatal error: Unable to locate credentials
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketofkms /home/ec2-user --recursive
fatal error: Unable to locate credentials
[root@ip-172-31-0-12 ec2-user]# aws --version
aws-cli/2.15.30 Python/3.9.16 Linux/6.1.102-108.177.amzn2023.x86_64 source/x86_64.amzn.2023 prompt/off
[root@ip-172-31-0-12 ec2-user]# aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
[root@ip-172-31-0-12 ec2-user]#
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketofkms /home/ec2-user --recursive
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketofkms /home/ec2-user --recursive
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketofkms /home/ec2-user --recursive
download: s3://bucketofkms/ImportToken.bin to ./ImportToken.bin
download: s3://bucketofkms/WrappingPublicKey.bin to ./WrappingPublicKey.bin
download: s3://bucketofkms/README.txt to ./README.txt
[root@ip-172-31-0-12 ec2-user]#
```

openssl version

aws s3 cp s3://bucketname /home/ec2-user --recursive

openssl rand -out plaintextkeymaterial.bin 32

openssl pkeyutl -in plaintextkeymaterial.bin -out EncryptedKeyMaterial.bin -inkey <wrapping key> -keyform der -pubin -encrypt -pkeyopt rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:sha256

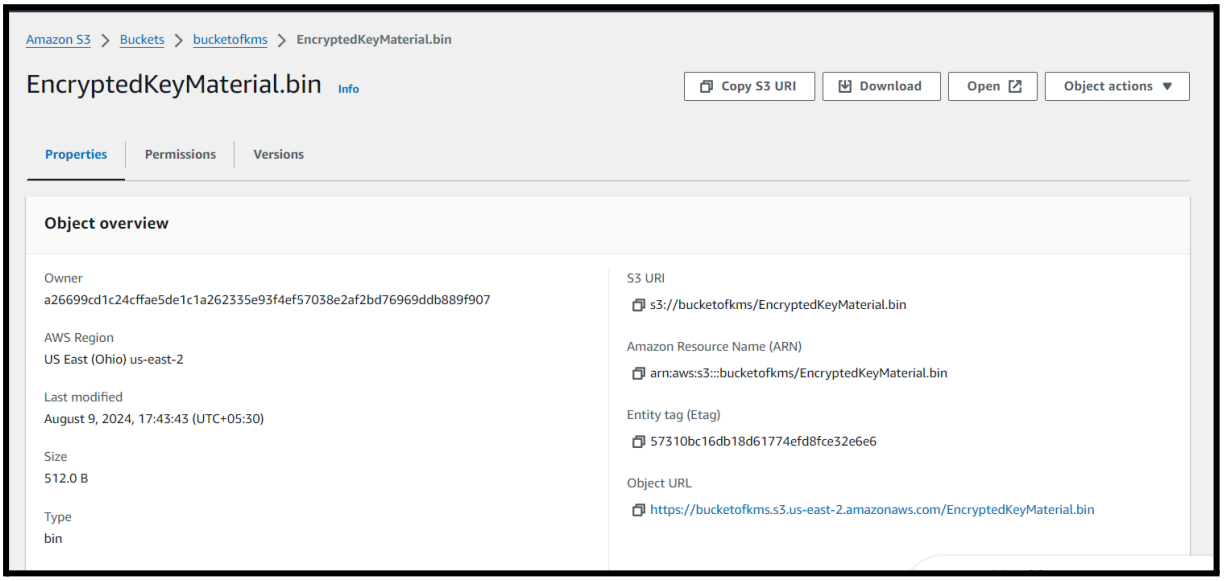
aws s3 cp /home/ec2-user s3://bucketname --recursive

```
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketofkms /home/ec2-user --recursive
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketofkms /home/ec2-user --recursive
[root@ip-172-31-0-12 ec2-user]# aws s3 cp s3://bucketofkms /home/ec2-user --recursive
download: s3://bucketofkms/ImportToken.bin to ./ImportToken.bin
download: s3://bucketofkms/WrappingPublicKey.bin to ./WrappingPublicKey.bin
download: s3://bucketofkms/README.txt to ./README.txt
[root@ip-172-31-0-12 ec2-user]#
openssl rand -out plaintextkeymaterial.bin 32
[root@ip-172-31-0-12 ec2-user]# openssl rand -out plaintextkeymaterial.bin 32
[root@ip-172-31-0-12 ec2-user]# openssl pkeyutl -in plaintextkeymaterial.bin -out EncryptedKeyMaterial.bin -inkey WrappingPublicKey.bin -keyform der -pubin -encrypt -pkeyopt rsa_paddi
ode:oaep -pkeyopt rsa_oaep_md:sha256
[root@ip-172-31-0-12 ec2-user]#
[root@ip-172-31-0-12 ec2-user]# openssl pkeyutl -in plaintextkeymaterial.bin -out EncryptedKeyMaterial.bin -inkey WrappingPublicKey.bin -keyform der -pubin -encrypt -pkeyopt rsa_paddi
ode:oaep -pkeyopt rsa_oaep_md:sha256
[root@ip-172-31-0-12 ec2-user]# aws s3 cp /home/ec2-user s3://bucketofkms --recursive
upload: ./.bash_logout to s3://bucketofkms/.bash_logout
upload: ./ImportToken.bin to s3://bucketofkms/ImportToken.bin
upload: ./WrappingPublicKey.bin to s3://bucketofkms/WrappingPublicKey.bin
upload: ./EncryptedKeyMaterial.bin to s3://bucketofkms/EncryptedKeyMaterial.bin
upload: ./plaintextkeymaterial.bin to s3://bucketofkms/plaintextkeymaterial.bin
upload: ./README.txt to s3://bucketofkms/README.txt
upload: ./.bash_profile to s3://bucketofkms/.bash_profile
upload: .ssh/authorized_keys to s3://bucketofkms/.ssh/authorized_keys
upload: ./.bashrc to s3://bucketofkms/.bashrc
[root@ip-172-31-0-12 ec2-user]#
```

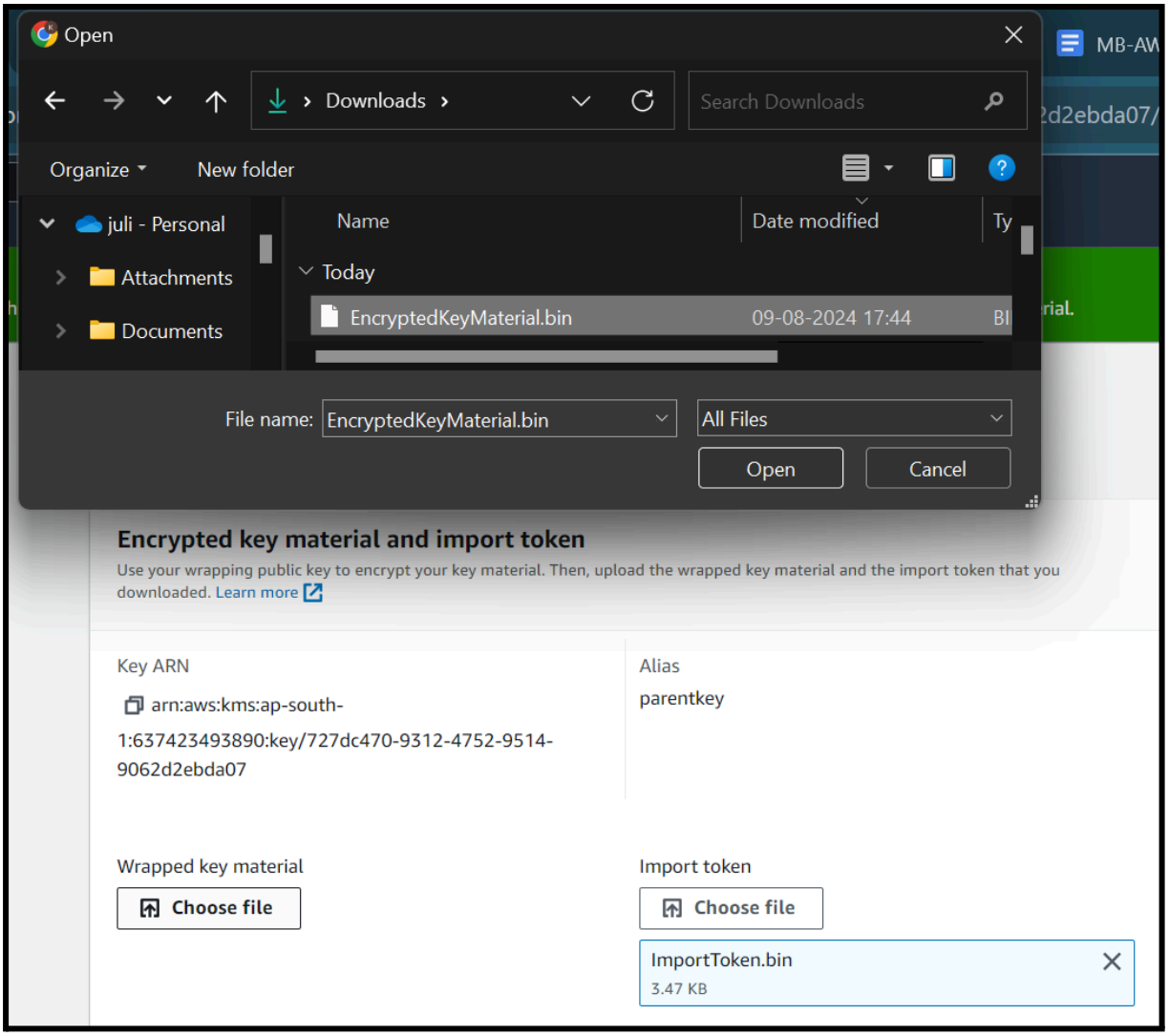
>aws s3 cp /home/ec2-user/EncryptedKeyMaterial.bin s3://bucketofkms

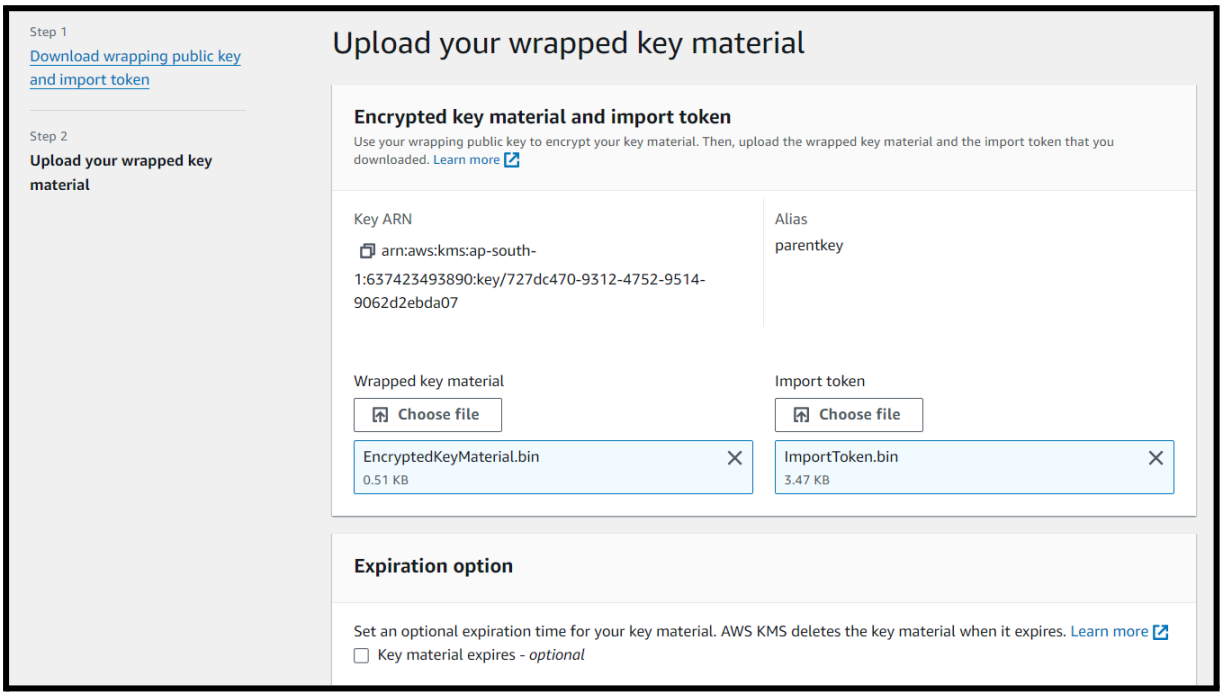
```
[root@ip-172-31-0-12 ec2-user]# aws s3 cp /home/ec2-user/EncryptedKeyMaterial.bin s3://bucketofkms
upload: ./EncryptedKeyMaterial.bin to s3://bucketofkms/EncryptedKeyMaterial.bin
[root@ip-172-31-0-12 ec2-user]#
```

Download encrypted file

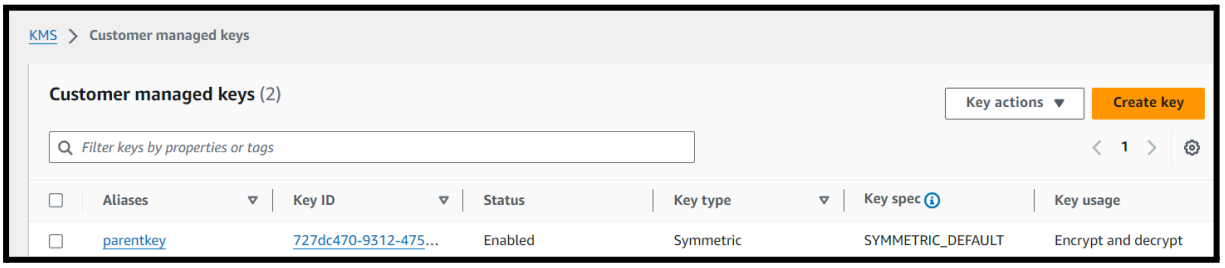


Upload


Open
Downloads
Search Downloads
Organize
New folder
juli - Personal
Attachments
Documents
Name
Date modified
Today
EncryptedKeyMaterial.bin
09-08-2024 17:44
File name: EncryptedKeyMaterial.bin
All Files
Open
Cancel
Encrypted key material and import token
Use your wrapping public key to encrypt your key material. Then, upload the wrapped key material and the import token that you downloaded. Learn more
Key ARN
arn:aws:kms:ap-south-1:637423493890:key/727dc470-9312-4752-9514-9062d2ebda07
Alias
parentkey
Wrapped key material
Choose file
Import token
Choose file
ImportToken.bin
3.47 KB


Step 1
Download wrapping public key and import token
Step 2
Upload your wrapped key material
Upload your wrapped key material
Encrypted key material and import token
Use your wrapping public key to encrypt your key material. Then, upload the wrapped key material and the import token that you downloaded. Learn more
Key ARN
arn:aws:kms:ap-south-1:637423493890:key/727dc470-9312-4752-9514-9062d2ebda07
Alias
parentkey
Wrapped key material
Choose file
EncryptedKeyMaterial.bin
0.51 KB
Import token
Choose file
ImportToken.bin
3.47 KB
Expiration option
Set an optional expiration time for your key material. AWS KMS deletes the key material when it expires. Learn more
Key material expires - optional

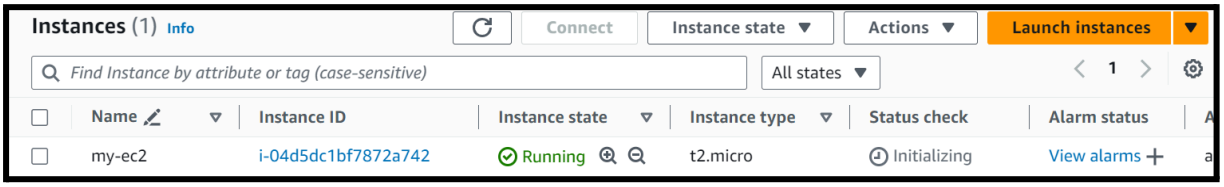
Key is enabled now



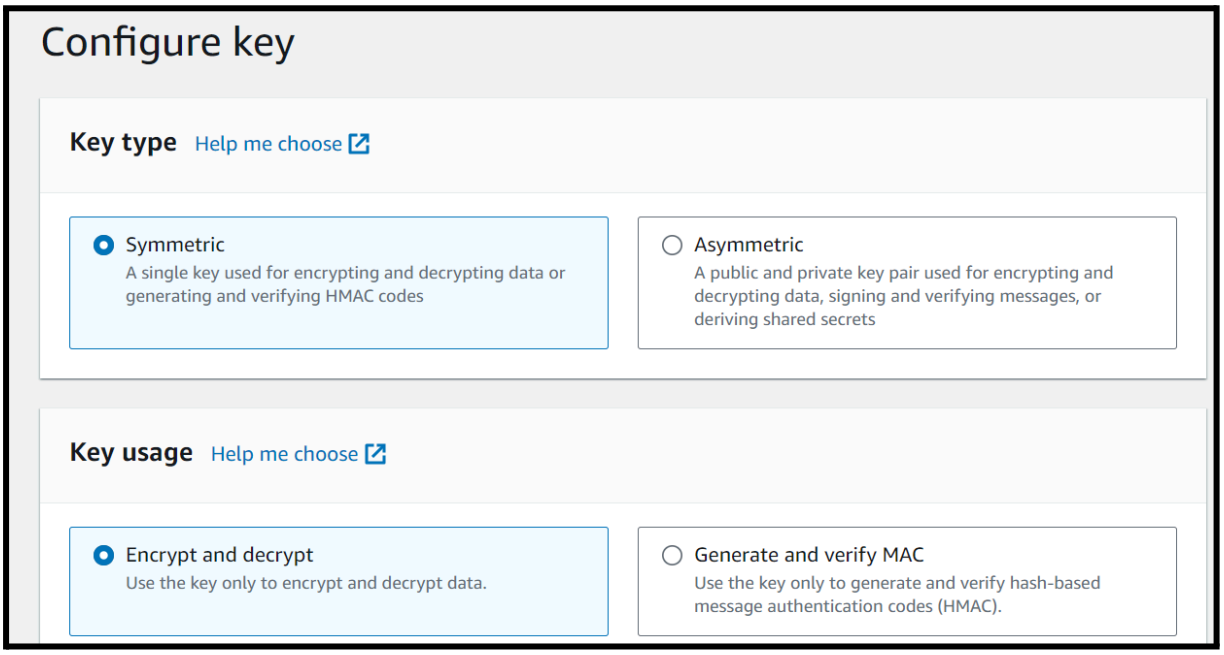
Date : 12-8-2024

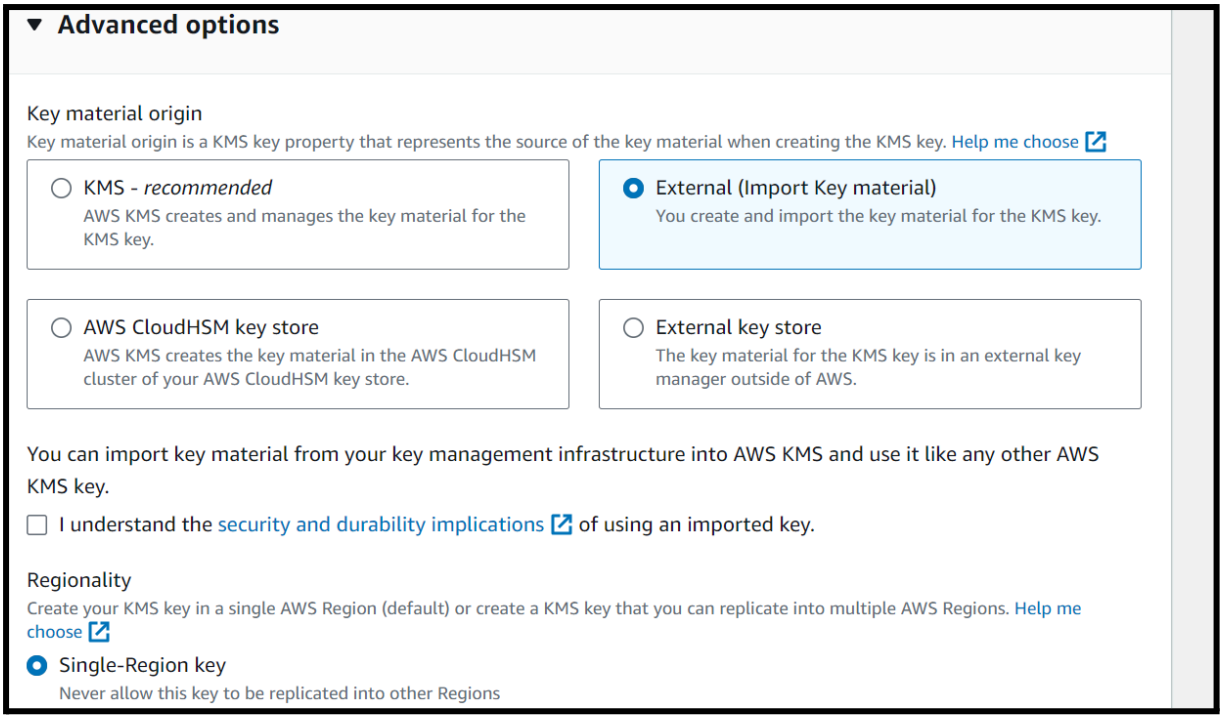
AIM : KEY MANAGEMENT SERVICE WITH EC2

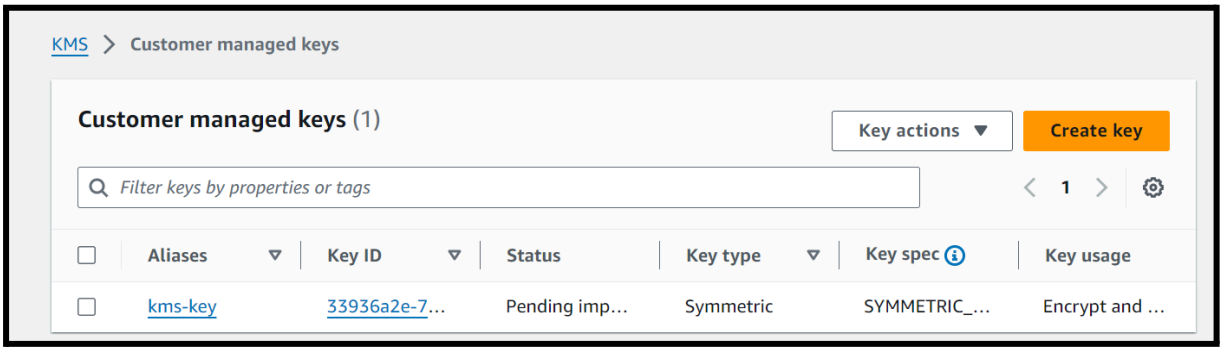
Step 1 : launch an EC2 machine

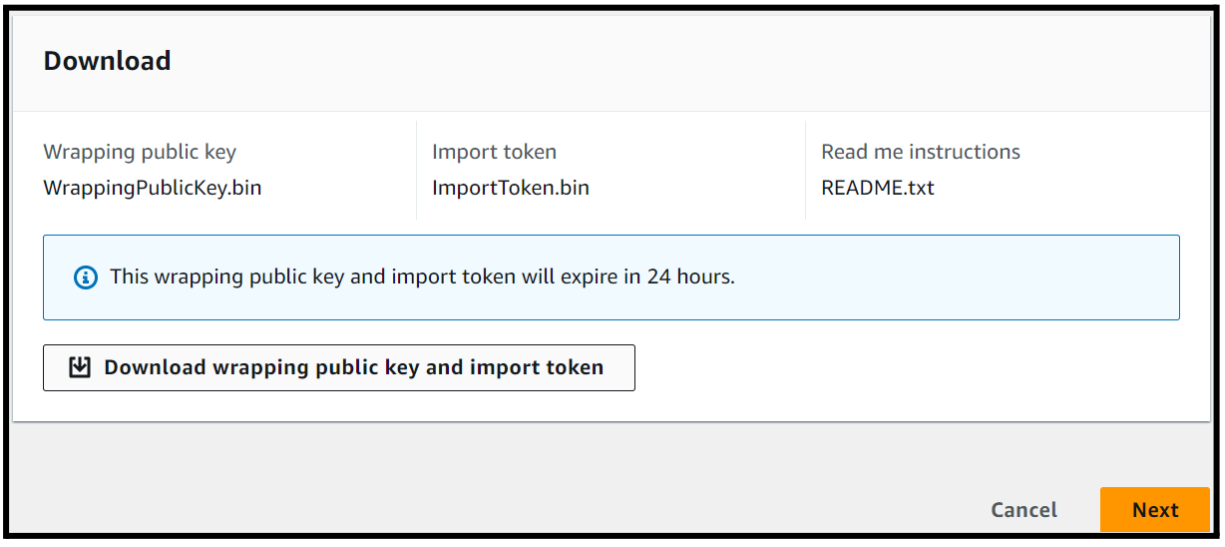


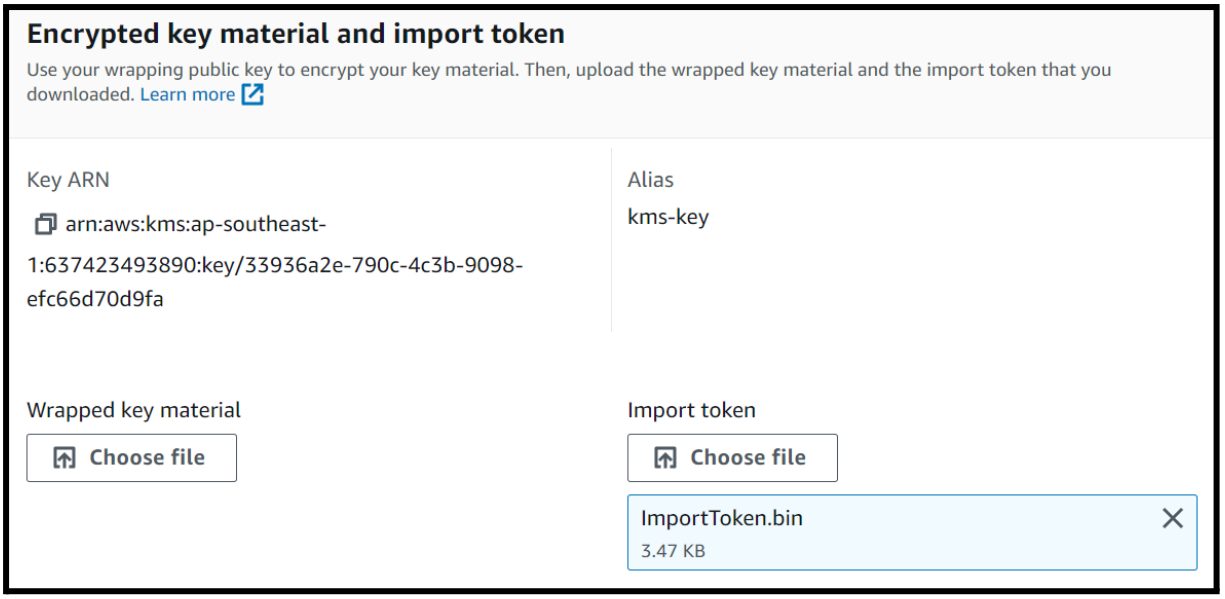
Step 2 : create KMS



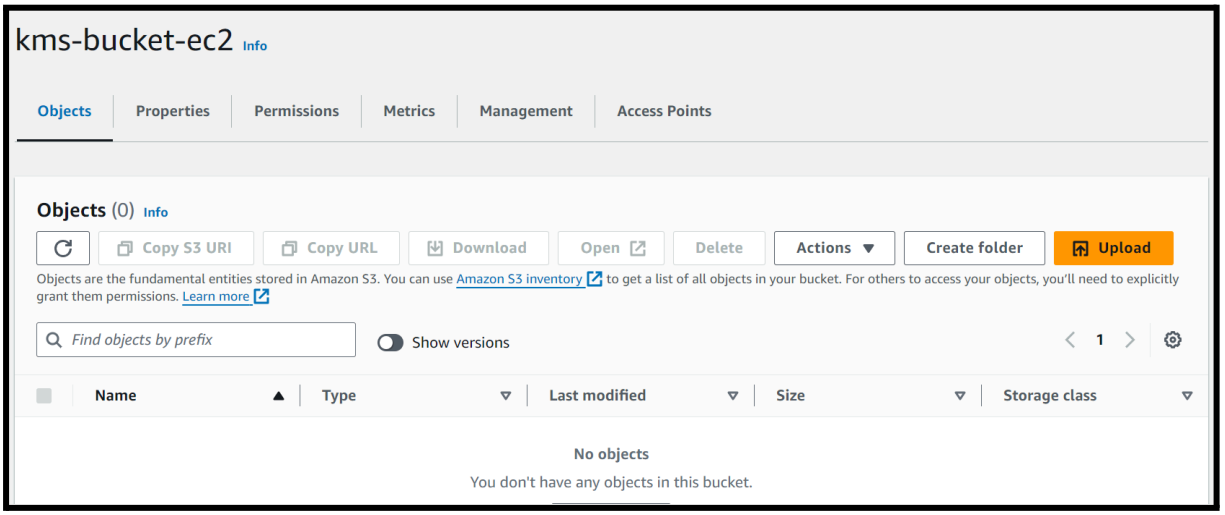

Advanced options
Key material origin
Key material origin is a KMS key property that represents the source of the key material when creating the KMS key. Help me choose
KMS - recommended
AWS KMS creates and manages the key material for the KMS key.
External (Import Key material)
You create and import the key material for the KMS key.
AWS CloudHSM key store
AWS KMS creates the key material in the AWS CloudHSM cluster of your AWS CloudHSM key store.
External key store
The key material for the KMS key is in an external key manager outside of AWS.
You can import key material from your key management infrastructure into AWS KMS and use it like any other AWS KMS key.
I understand the security and durability implications of using an imported key.
Regionality
Create your KMS key in a single AWS Region (default) or create a KMS key that you can replicate into multiple AWS Regions. Help me choose
Single-Region key
Never allow this key to be replicated into other Regions


KMS
Customer managed keys
Customer managed keys (1)
Key actions
Create key
Filter keys by properties or tags
1
Aliases
Key ID
Status
Key type
Key spec
Key usage
kms-key
33936a2e-7...
Pending imp...
Symmetric
SYMMETRIC_...
Encrypt and ...

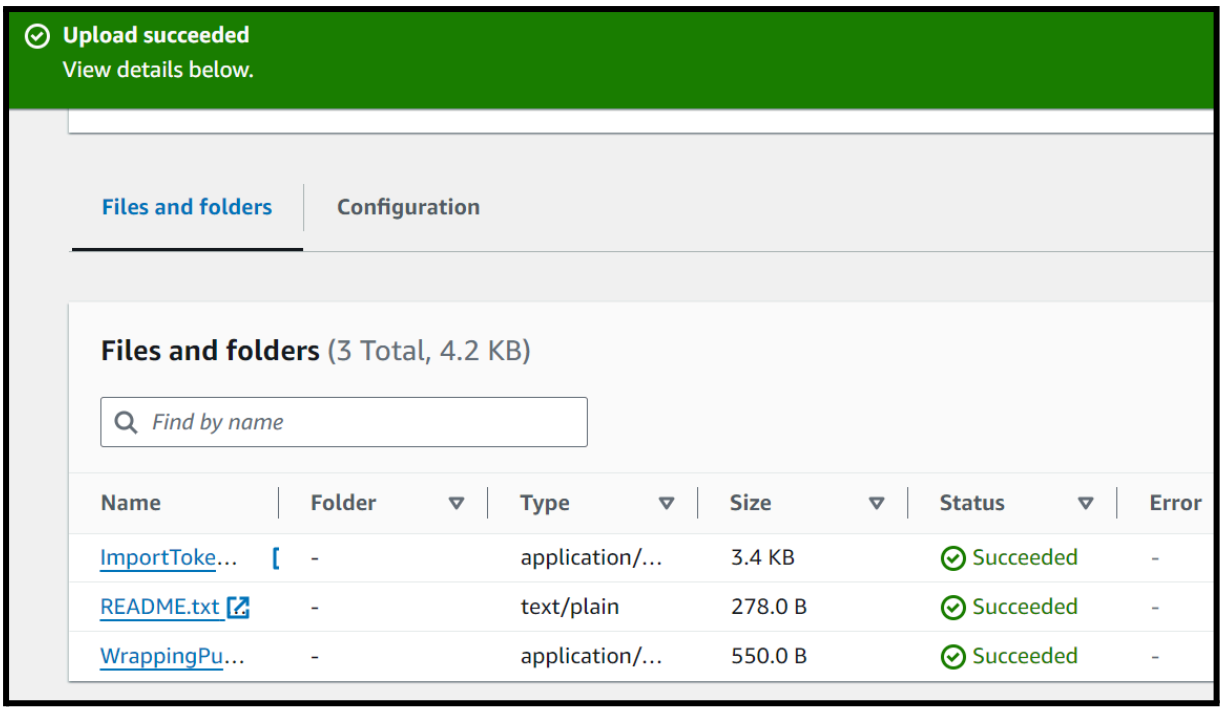

Download
Wrapping public key
WrappingPublicKey.bin
Import token
ImportToken.bin
Read me instructions
README.txt
This wrapping public key and import token will expire in 24 hours.
Download wrapping public key and import token
Cancel
Next



Create s3 bucket



Upload files



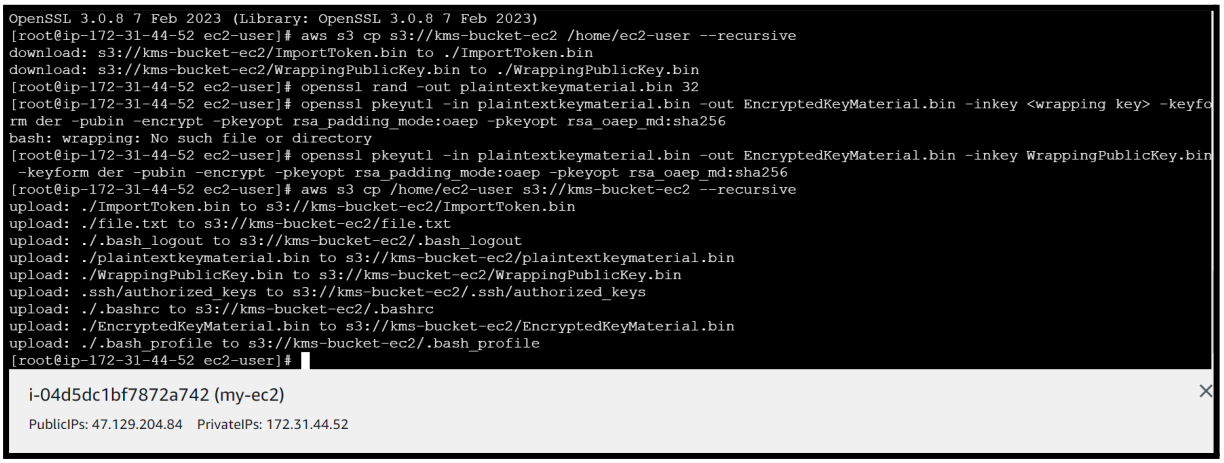
openssl version

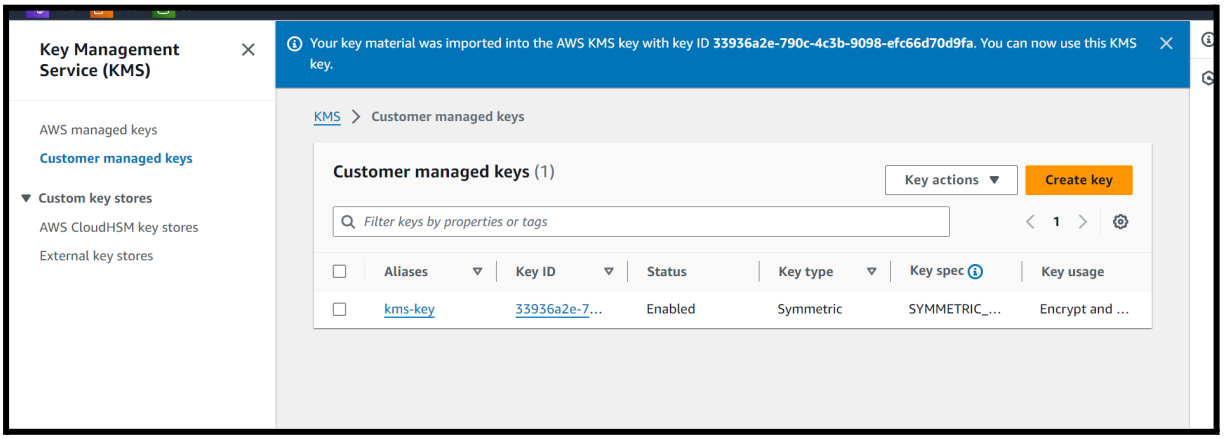
aws s3 cp s3://bucketname /home/ec2-user --recursive

openssl rand -out plaintextkeymaterial.bin 32

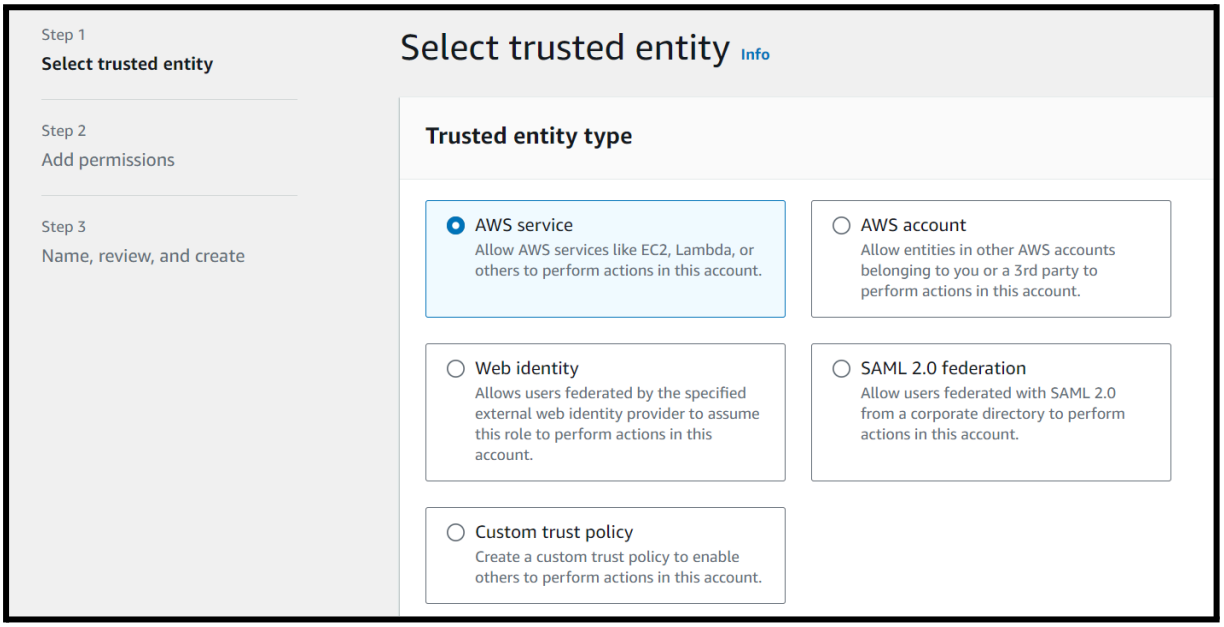
openssl pkeyutl -in plaintextkeymaterial.bin -out EncryptedKeyMaterial.bin -inkey <wrapping key> -keyform der -pubin -encrypt -pkeyopt rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:sha256

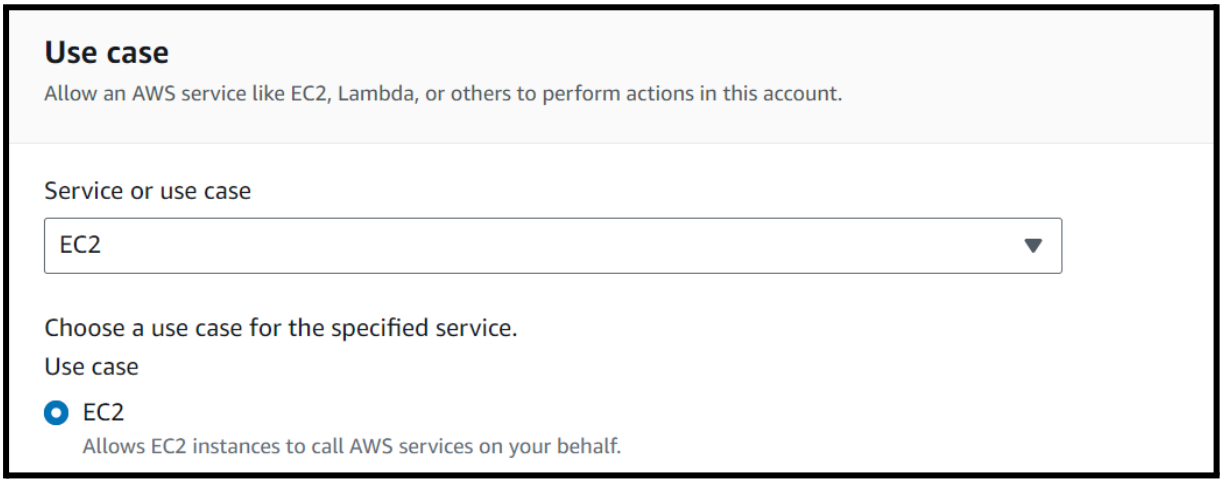
aws s3 cp /home/ec2-user s3://bucketname --recursive

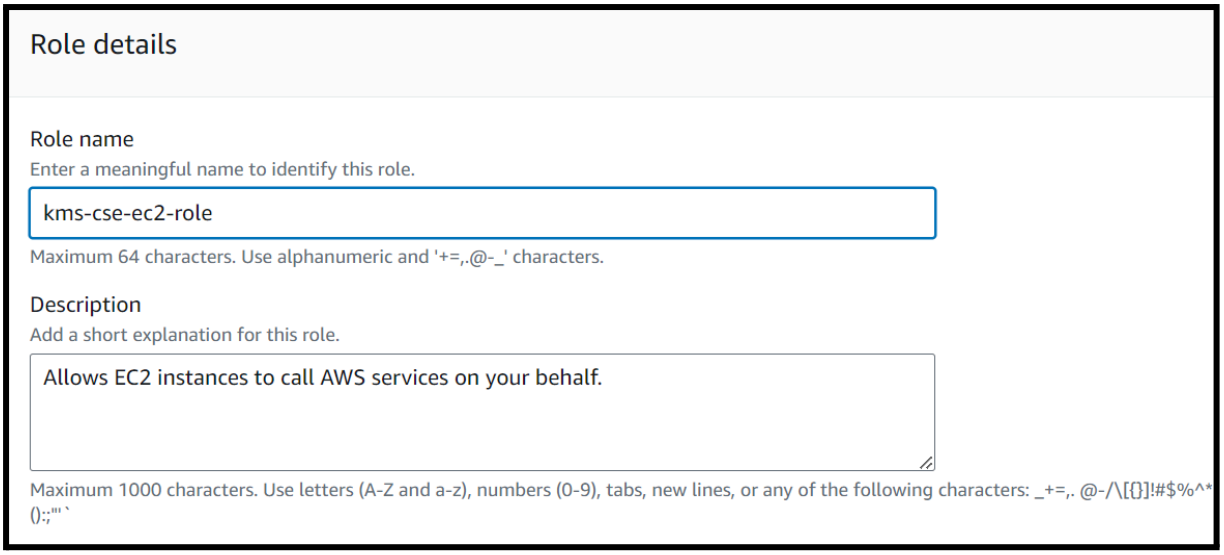


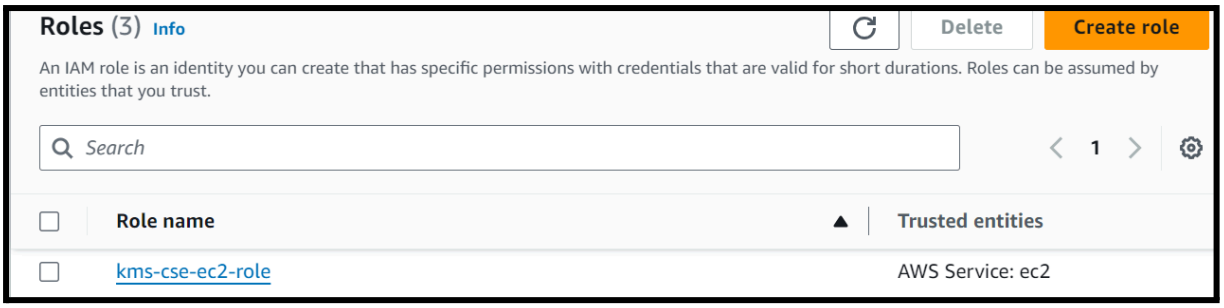


Step 3 : create role

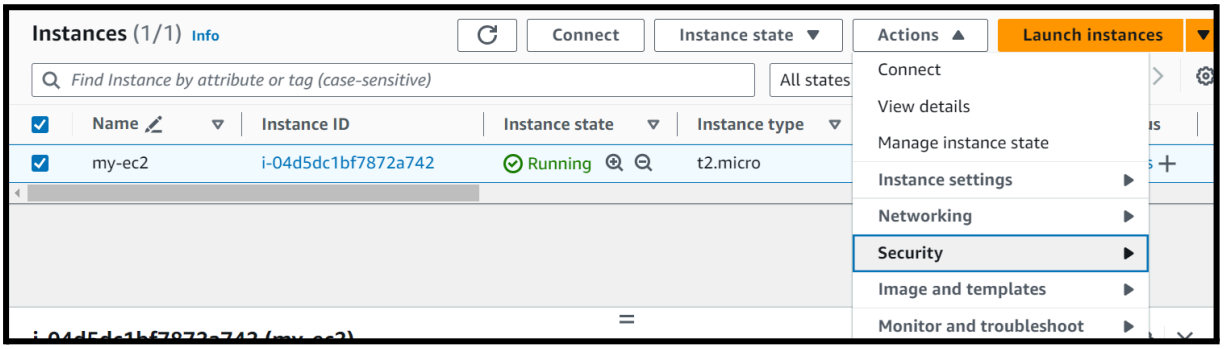


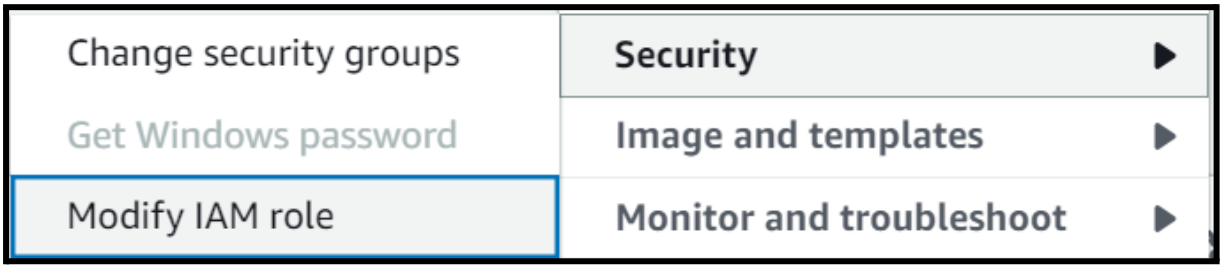




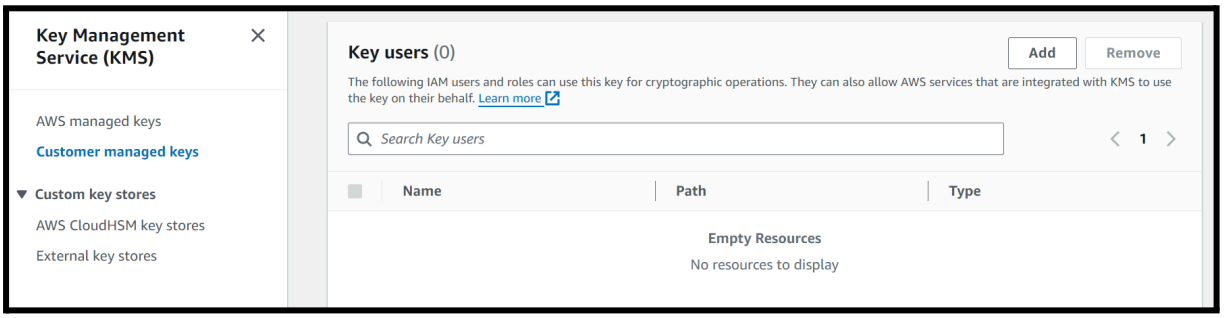


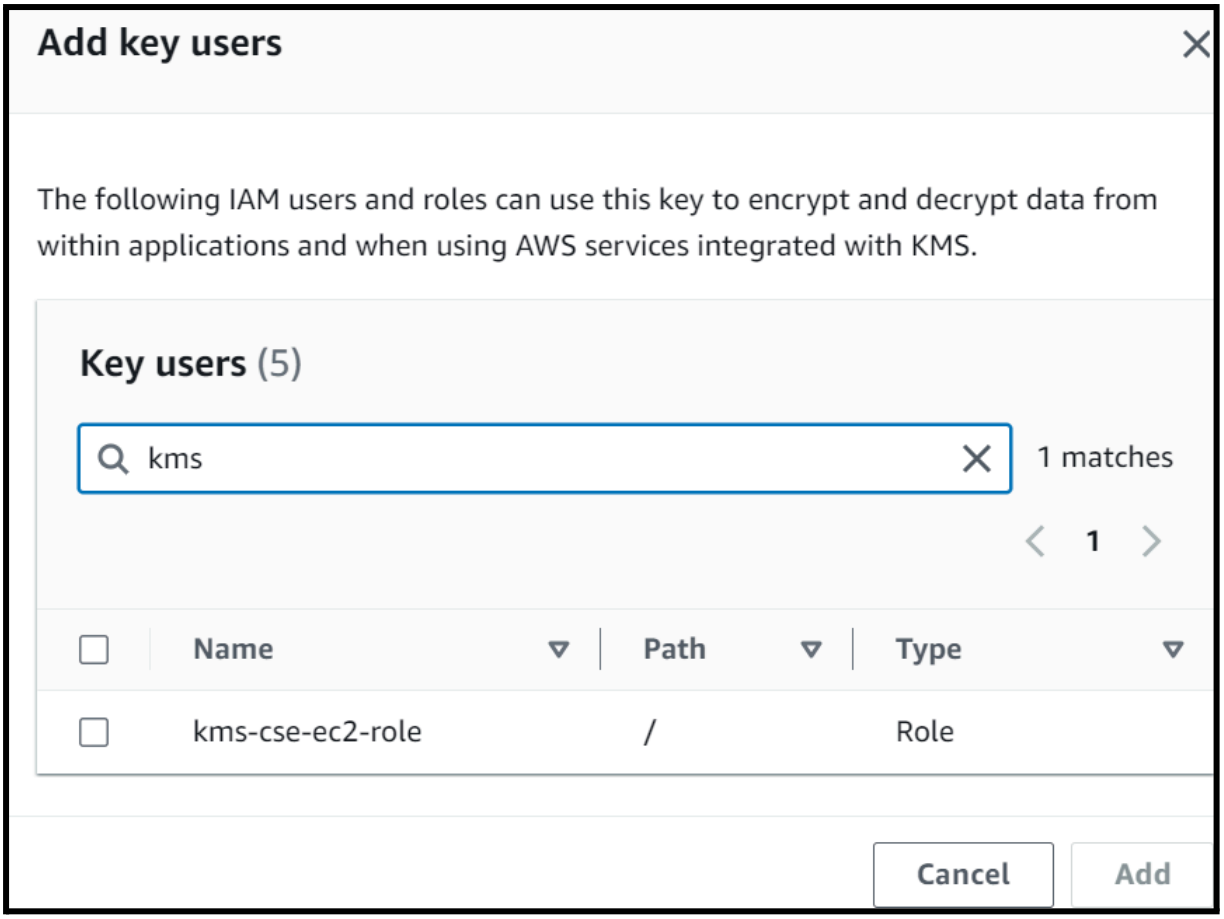
Go to > ec2 > modify security





KMS > key users > add IAM role





Connect EC2 machine

1. **Create a file with data which you want to ecrypt**

echo "Hello,Pooja! Encrypt Me." > file.txt

```
[ec2-user@ip-172-31-44-52 ~]$ sudo su
[root@ip-172-31-44-52 ec2-user]# yum update -y
Last metadata expiration check: 0:14:19 ago on Mon Aug 12 10:59:03 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-44-52 ec2-user]# yum upgrade -y
Last metadata expiration check: 0:14:27 ago on Mon Aug 12 10:59:03 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-44-52 ec2-user]# echo "Hello,Pooja! Encrypt Me." > file.txt
[root@ip-172-31-44-52 ec2-user]#
```

i-04d5dc1bf7872a742 (my-ec2)

PublicIPs: 47.129.204.84 PrivateIPs: 172.31.44.52

Generate data key using CMK

> aws kms generate-data-key --key-id alias/kms-key --key-spec AES_256 --region ap-southeast-1

```
[root@ip-172-31-44-52 ec2-user]# echo "Hello, Pooja! Please encrypt me ." > file.txt
[root@ip-172-31-44-52 ec2-user]# aws kms generate-data-key --key-id alias/kms-key --key-spec AES_256 --region ap-southeast-1
{
    "CiphertextBlob": "AQIDAHgh/QSZkrrkP+tAydUWGaUNhoxg44VQxAiTEW7NYikXUwEpRTK5vCJ8QTdHe4nA2fGzAAAAfjB8BgkqhkiG9w0BBwagbzBtAgEAMGgGCSqGS
Ib3DQEHATAeBglghkgBZQMEAS4wEQQMCQ8O6+ZSy0sd9ftqAgEQgDvl0qIINxOUw74TX9rOubKKrImbi1NKOQ074rUEN8E6n7S2B5EIC+0PZivDtYpKZOOsJ+KJMK5h7smlEQ==",
    "Plaintext": "v/SjBZL9hWrfAscs3oQBuDo6S5F8G/JsFOzlkpqGzFk=",
    "KeyId": "arn:aws:kms:ap-southeast-1:637423493890:key/33936a2e-790c-4c3b-9098-efc66d70d9fa"
}
[root@ip-172-31-44-52 ec2-user]#
```

Create a data key file which contain decoded Plaintext

> echo v/SjBZL9hWrfAscs3oQBuDo6S5F8G/JsFOzlkpqGzFk= | base64 --decode > datakey

```
[root@ip-172-31-44-52 ec2-user]# echo AQIDAHgh/QSZkrrkP+tAydUWGaUNhoxg44VQxAiTEW7NYikXUwEpRTK5vCJ8QTdHe4nA2fGzAAAAfjB8BgkqhkiG9w0BBwagbz
BtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMCQ8O6+ZSy0sd9ftqAgEQgDvl0qIINxOUw74TX9rOubKKrImbi1NKOQ074rUEN8E6n7S2B5EIC+0PZivDtYpKZOOsJ+
KJMK5h7smlEQ== | base64 --decode > encrypteddatakey
[root@ip-172-31-44-52 ec2-user]#
```

```
[root@ip-172-31-44-52 ec2-user]# echo v/SjBZL9hWrfAscs3oQBuDo6S5F8G/JsFOzlkpqGzFk= | base64 --decode > datakey
[root@ip-172-31-44-52 ec2-user]#
```

7. Create a encrypted data key file which contains decoded

> echo 7. Create a encrypted data key file which contains decoded

echo <ciphertext-bob> | base64 --decode > encrypteddatakey

| base64 --decode > encrypteddatakey

Encrypt file using data key

openssl enc -in ./file.txt -out ./encrypted-file.txt -e -aes256 -k fileb://./datakey

```
[root@ip-172-31-44-52 ec2-user]# ll
total 32
-rw-r--r--. 1 root root  512 Aug 12 12:07 EncryptedKeyMaterial.bin
-rw-r--r--. 1 root root 3466 Aug 12 11:58 ImportToken.bin
-rw-r--r--. 1 root root  550 Aug 12 11:58 WrappingPublicKey.bin
-rw-r--r--. 1 root root   32 Aug 12 12:18 datakey
-rw-r--r--. 1 root root   64 Aug 12 12:22 encrypted-file.txt
-rw-r--r--. 1 root root  184 Aug 12 12:21 encrypteddatakey
-rw-r--r--. 1 root root   34 Aug 12 12:15 file.txt
-rw-r--r--. 1 root root   32 Aug 12 12:06 plaintextkeymaterial.bin
[root@ip-172-31-44-52 ec2-user]# cat file.,,txt
cat: file.,,txt: No such file or directory
[root@ip-172-31-44-52 ec2-user]# cat file.txt
Hello, Pooja! Please encrypt me .
[root@ip-172-31-44-52 ec2-user]#
```

Remove datakey and datafile

rm datakey

rm file.txt

```
[root@ip-172-31-44-52 ec2-user]# rm datakey
rm: remove regular file 'datakey'? y
[root@ip-172-31-44-52 ec2-user]# rm file.txt
rm: remove regular file 'file.txt'? y
[root@ip-172-31-44-52 ec2-user]#
```

.Decrypt "encrypted data key"
aws kms decrypt --ciphertext-blob fileb://./encrypteddatakey --region ap-south-1

```
[root@ip-172-31-44-52 ec2-user]# aws kms decrypt --ciphertext-blob fileb://./encrypteddatakey --region ap-southeast-1
{
    "KeyId": "arn:aws:kms:ap-southeast-1:637423493890:key/33936a2e-790c-4c3b-9098-efc66d70d9fa",
    "Plaintext": "v/SjBZL9hWrfAscs3oQBuDo6S5F8G/JsFOzlkpqGzFk=",
    "EncryptionAlgorithm": "SYMMETRIC_DEFAULT"
}
```

Create a Data Key file which contains plaintext
echo <plaintext> | base64 --decode > datakey

Decrypt "encrypted data file" using the data key
openssl enc -in ./encrypted-file.txt -out ./file.txt -d -aes256 -k fileb://./datakey

```
[root@ip-172-31-44-52 ec2-user]# echo v/SjBZL9hWrfAscs3oQBuDo6S5F8G/JsFOzlkpqGzFk= | base64 --decode > datakey
[root@ip-172-31-44-52 ec2-user]# openssl enc -in ./encrypted-file.txt -out ./file.txt -d -aes256 -k fileb://./datakey
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[root@ip-172-31-44-52 ec2-user]# ll
total 32
-rw-r--r--. 1 root root  512 Aug 12 12:07 EncryptedKeyMaterial.bin
-rw-r--r--. 1 root root 3466 Aug 12 11:58 ImportToken.bin
-rw-r--r--. 1 root root  550 Aug 12 11:58 WrappingPublicKey.bin
-rw-r--r--. 1 root root   32 Aug 12 12:29 datakey
-rw-r--r--. 1 root root   64 Aug 12 12:22 encrypted-file.txt
-rw-r--r--. 1 root root  184 Aug 12 12:21 encrypteddatakey
-rw-r--r--. 1 root root   34 Aug 12 12:29 file.txt
-rw-r--r--. 1 root root   32 Aug 12 12:06 plaintextkeymaterial.bin
[root@ip-172-31-44-52 ec2-user]#
```

```
[root@ip-172-31-44-52 ec2-user]# cat file.txt
Hello, Pooja! Please encrypt me .
[root@ip-172-31-44-52 ec2-user]#
```