Lab 2: S3 Buckets

A Outline

Amazon Simple Storage Service (Amazon S3) is used to store objects within data buckets. They are the simplest form of storage in AWS and offer cheap access to data. Unfortunately they have been involved with a range of data leaks. This is mainly due to poor configuration of the buckets. We can replicate S3 buckets over different geographical regions and can apply an access policy on buckets. They also allow public access to data and can be encrypted on the server. S3 storage costs $0.023 per GB for the first 50TB. Thus a 1TB data storage will cost around $23 per month. In this lab we will create and deleted buckets and files, and also implement encryption on these.

B Creating and accessing a bucket

First, from the AWS CLI, list your buckets with "aws s3 ls":

```
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 ls
2022-11-16 10:52:53 billbucket33
2022-11-16 10:45:41 napierbillbucket
```

We can see that there are two buckets that have already been created. Now we will create a bucket. Think of a name, and create the bucket, and then list the buckets to see if the bucket has been created:

```
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 mb s3://billbucket333
make_bucket: billbucket333
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 ls
2022-11-16 10:52:53 billbucket33
2022-11-16 10:57:06 billbucket333
2022-11-16 10:45:41 napierbillbucket
```

Now create a bucket with the same name.

What happens and why do you think this happens?

Now we will create a file named "1.txt" using nano:

```
nano 1.txt
Hello
^X
```

Next, we will copy this file into the newly created bucket, and then list the bucket:

```
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 cp 1.txt s3://billbucket333/1.txt
upload: ./1.txt to s3://billbucket333/1.txt
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 ls s3://billbucket333
2022-11-16  1:11:28          6 1.txt
```

C Accessing buckets from AWS Console

Now go to your AWS Console (Figure 1) and find your bucket.

Which is the name of the bucket you have created?

Which region has the bucket been created in?

At what time was the bucket created?

Now, view the contents of the bucket (Figure 2).

What is the name of the file in the bucket?

What are the contents of the file in the bucket?

Determine the URI of the file (Figure 3):

Determine the URL of the file (Figure 4):

What is the difference between the URI and the URL?

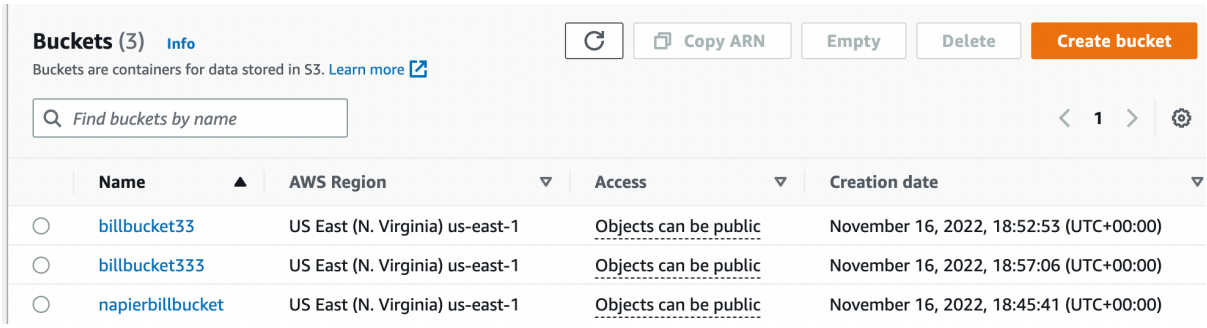

Figure 1: Listing of buckets

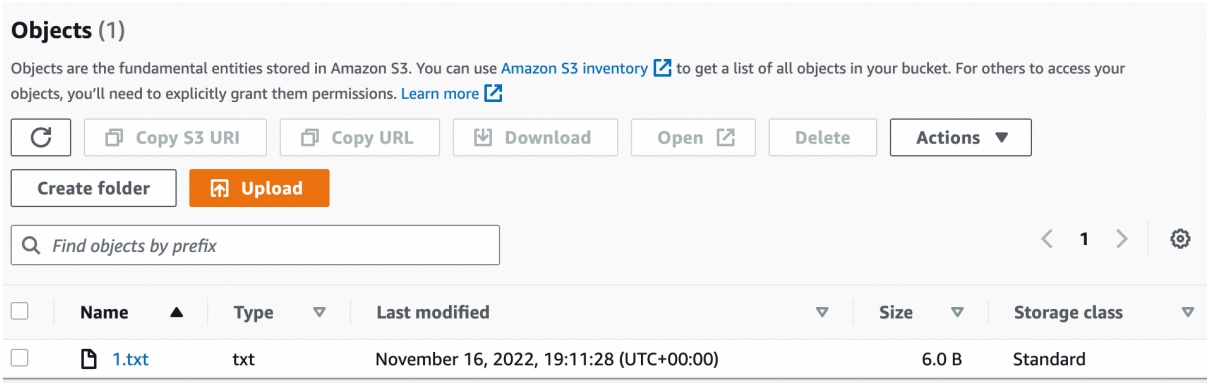

Figure 2: Viewing content of the bucket

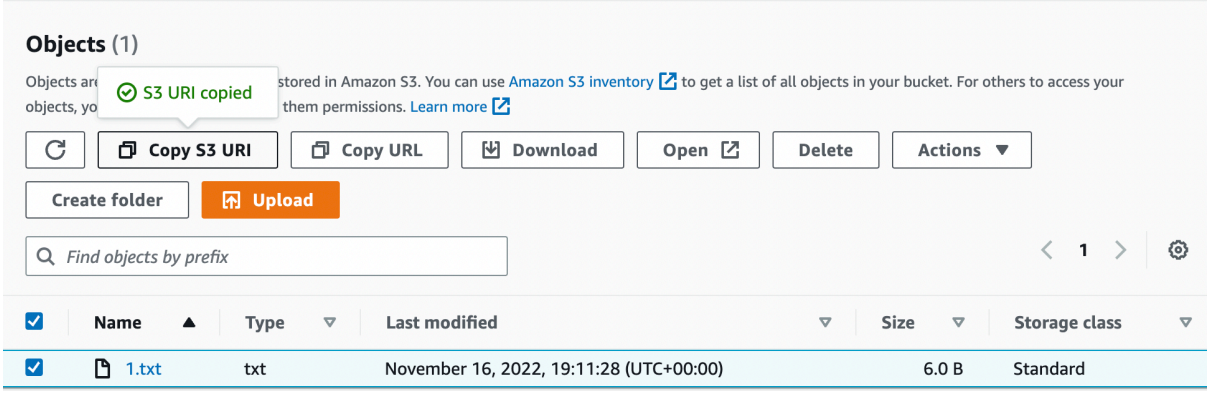

Figure 3: AWS URI

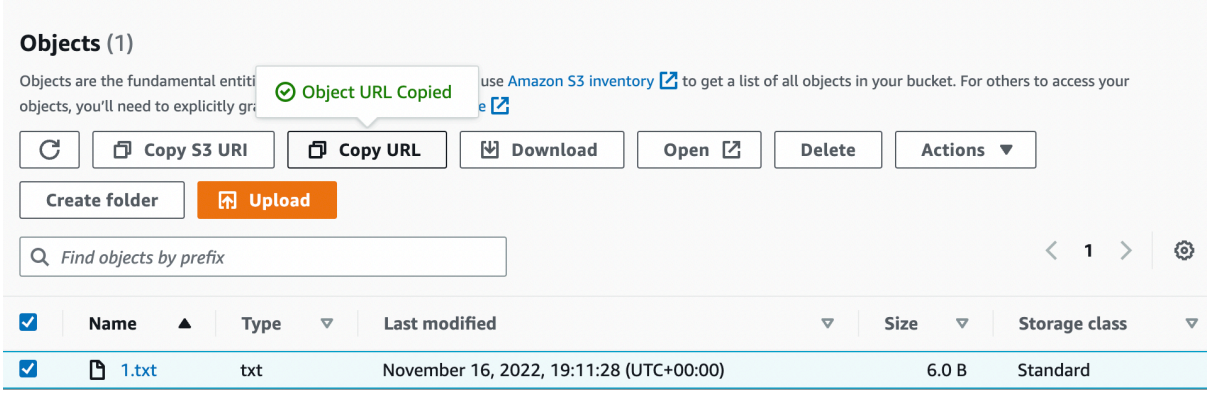

Figure 4: AWS URL

S3 does not default to public access. To check this, open the URL in a Web browser, and check that you do not have access to the file (Figure 5).

Figure 5: URL access not allowed

Now, we will enable public access. For this, select Action, and then "Share with a preassigned URL" (Figure 6).

Now setup a share of one minute (Figure 7)

Can you now access the contents of the file (Figure 8)?

Wait for one minute. Can you now access the file?

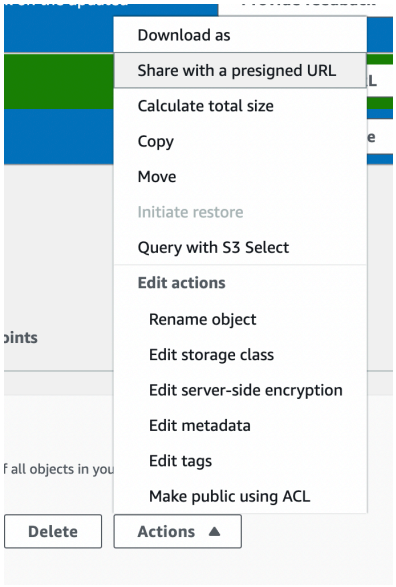

Figure 6: Enable public sharing

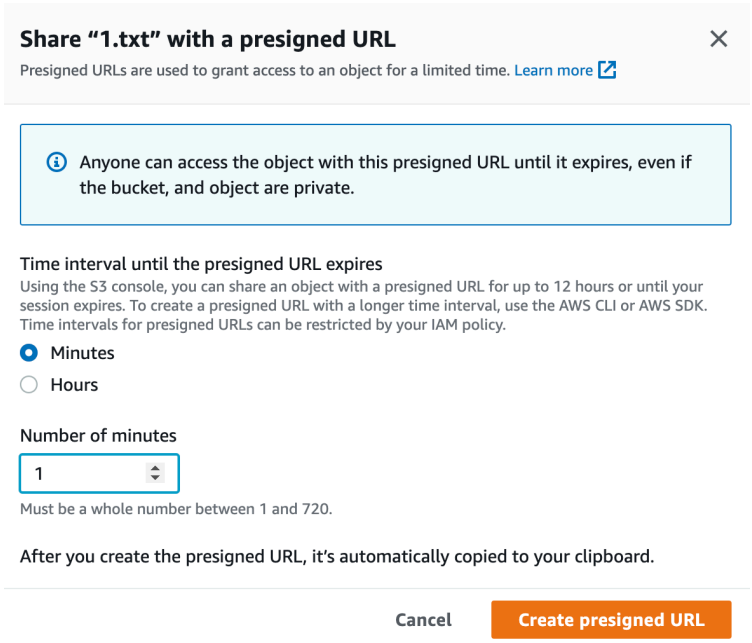

Figure 7: Enable access for one minute

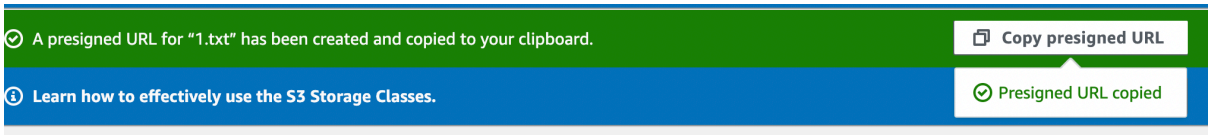

Figure 8: Copying the preassigned URL

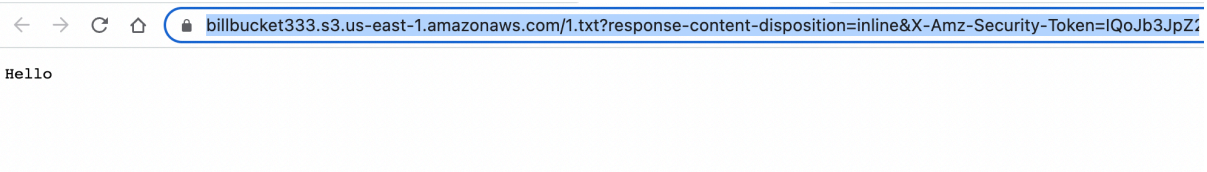

Figure 9: Public access

D Creating a folder from AWS Console

Now we will create a folder in our bucket from the AWS Console. For this select “Create Folder”, and add a folder named “doc” (Figure 10). Next, add a file from your local computer to the folder (Figure 11).

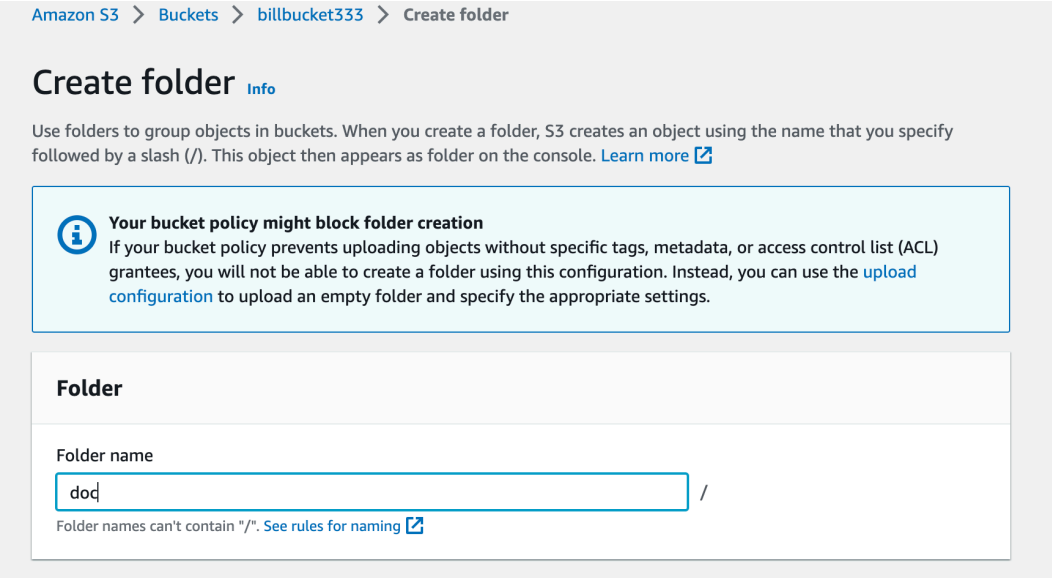

Figure 10: Creating a folder in a bucket

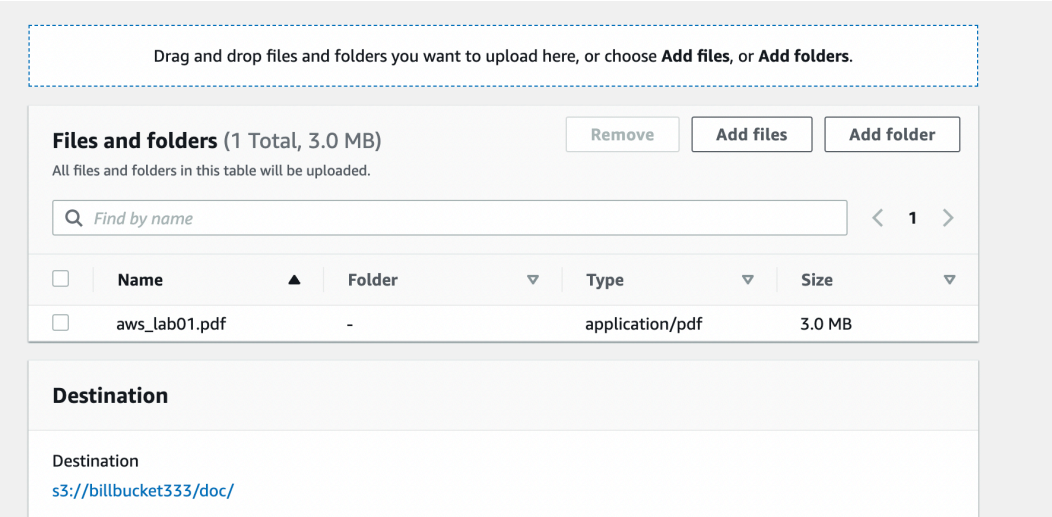

Figure 11: Creating a folder in a bucket

Now go back to your AWS CLI, and list your bucket:

```
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 ls s3://billbucket333
                           PRE doc/
2022-11-16 11:11:28          6 1.txt
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 ls s3://billbucket333/doc/
2022-11-16 11:27:46          0
2022-11-16 11:29:37    3147199 aws_lab01.pdf
```

Now we will copy the file from your bucket into your CLI:

```
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 cp
s3://billbucket333/doc/aws_lab01.pdf aws_lab01.pdf
download: s3://billbucket333/doc/aws_lab01.pdf to ./aws_lab01.pdf
ddd_v1_w_pcq_1801880@runweb67382:~$ ls
1.txt  aws_lab01.pdf
```

Has the file been created on your CLI?

Now, we will delete the file from the bucket, using “aws s3 rm”:

```
ddd_v1_w_pcq_1801880@runweb67382:~$ aws s3 rm
s3://billbucket333/doc/aws_lab01.pdf
delete: s3://billbucket333/doc/aws_lab01.pdf
```

Check on the CLI that the file has been deleted? [Yes/No]

Check on the Console that the file has been deleted? [Yes/No]

D Accessing buckets from Python

Apart from using the CLI and the AWS Console, we can use Python. For this we can integrate with the Boto3 library. From the CLI, create the following file using nano:

```
import boto3
import pprint
s3 = boto3.client("s3")

response = s3.create_bucket(Bucket="billbucket33331")
print(pprint.pprint(response))
response = s3.list_buckets()

print('Buckets:')
for bucket in response['Buckets']:
    print(f'  {bucket["Name"]}')
```

Now run the program, and observe the output:

```
ddd_v1_w_pcq_1801880@runweb67382:~$ python makebucket.py
{'Location': '/billbucket33331',
 'ResponseMetadata': {'HTTPHeaders': {'content-length': '0',
                      'date': 'Wed, 16 Nov 2022 19:42:53 GMT',
                      'location': '/billbucket33331',
                      'server': 'AmazonS3',
```

```
                        'x-amz-id-2':
'gKdRfFiZoZuA4pwmim2MemScAqcvy/xNN/QOyvPRCAWlkDXOPjyJ4UgbICPOnjWLrQKzVKewK
Vo=', 'x-amz-request-id': 'QC0K80Z83A19M56C'},
                       'HTTPStatusCode': 200,
                       'HostId':
'gKdRfFiZoZuA4pwmim2MemScAqcvy/xNN/QOyvPRCAWlkDXOPjyJ4UgbICPOnjWLrQKzVKewK
Vo=',
                       'RequestId': 'QC0K80Z83A19M56C',
                       'RetryAttempts': 0}}
None
Buckets:
  billbucket33
  billbucket333
  billbucket33331
  napierbillbucket
```

Has the bucket been created successfully? Yes/No

Now run your script again. What happens, and why?

E Deleting a bucket in Python

We can also delete a bucket with Python. For this we use the delete_bucket() method.

```
ddd_v1_w_pcq_1801880@runweb67382:~$ cp makebucket.py deletebucket.py
ddd_v1_w_pcq_1801880@runweb67382:~$ nano deletebucket.py

import boto3
import pprint
s3 = boto3.client("s3")

response = s3.delete_bucket(Bucket="billbucket33331")
print(pprint.pprint(response))
response = s3.list_buckets()

print('Buckets:')
for bucket in response['Buckets']:
    print(f'  {bucket["Name"]}')
```

Now run this program, and verify that your bucket has been deleted:

```
ddd_v1_w_pcq_1801880@runweb67382:~$ python deletebucket.py
{'ResponseMetadata': {'HTTPHeaders': {'date': 'Wed, 16 Nov 2022 19:44:55
GMT',
                       'server': 'AmazonS3',
                       'x-amz-id-2':
'8Ib/DPHy+GuScpYdF+0LGnBzKJJaZSeF4VUPMhy7t8W/LmzNv01z/7cqLnx/dubDuHRIe5ZwS
5Y=',
                       'x-amz-request-id': '5X9PBMM74XF7FDTW'},
                       'HTTPStatusCode': 204,
                       'HostId':
'8Ib/DPHy+GuScpYdF+0LGnBzKJJaZSeF4VUPMhy7t8W/LmzNv01z/7cqLnx/dubDuHRIe5ZwS
5Y=',
                       'RequestId': '5X9PBMM74XF7FDTW',
                       'RetryAttempts': 0}}
None
Buckets:
  billbucket33
  billbucket333
```

F Encrypting folders and files with S3-managed encryption keys

If you are worried about files being accessed on the server, it is possible to apply simple encryption to a folder or a set of files. First select the files and folders that you want to encryption for server-side encryption (Figure 12).

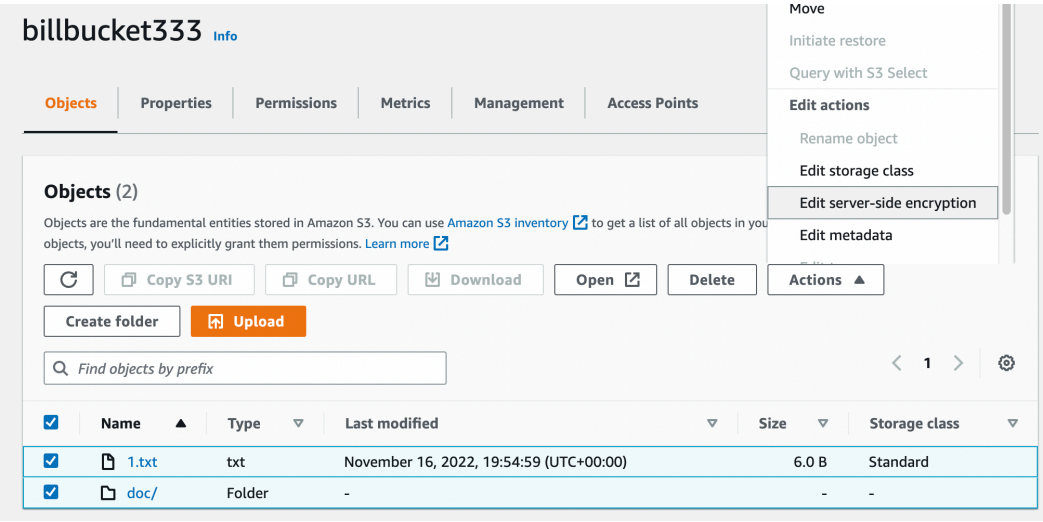

Figure 12: Creating a folder in a bucket

Next enable server-side encryption, and to use an Amazon S3 encryption key. This key will be stored within AWS (Figure 13). After we encryption, we should get a verification (Figure 14).

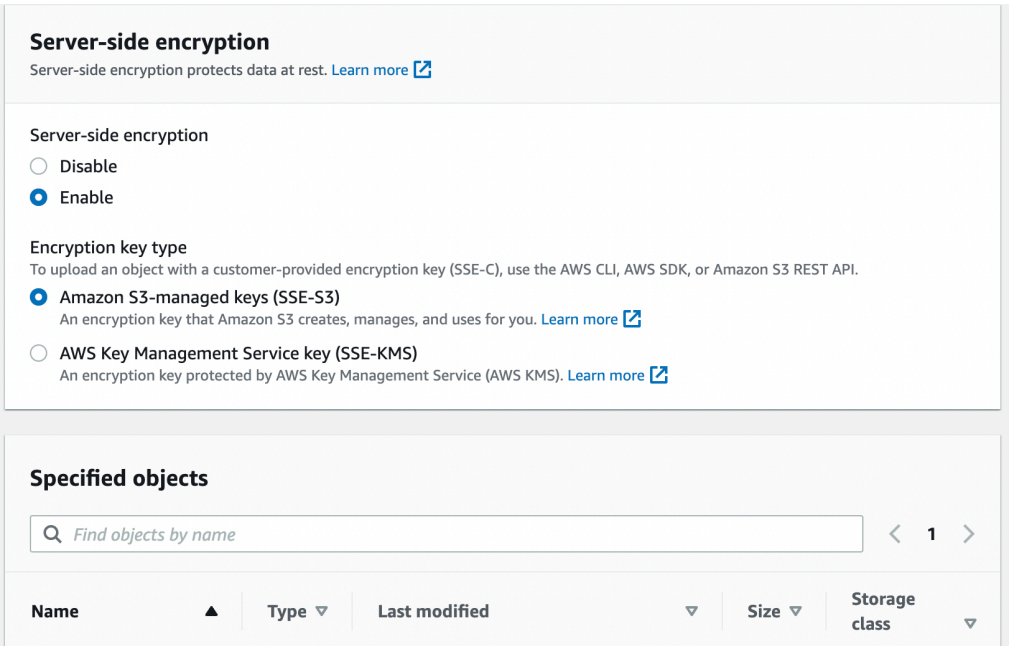

Figure 13: Enabling encryption

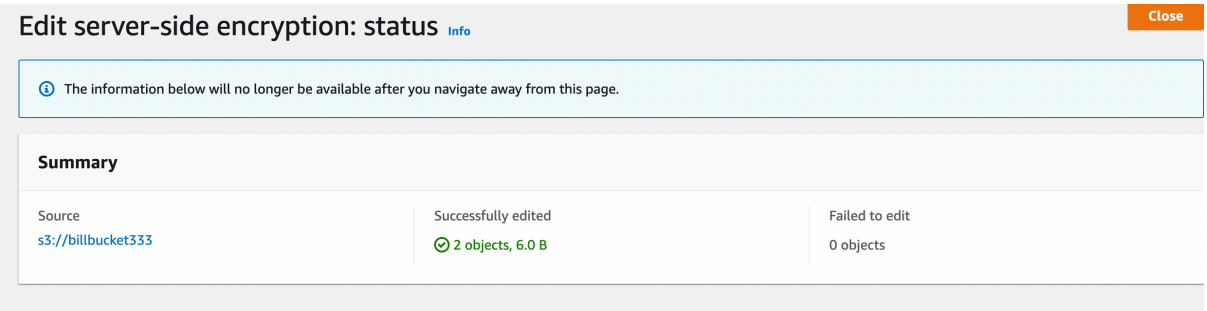

Figure 14: Verification of encryption

G Encrypting folders and files with KMS

With KMS (Key Manage Service) we can create an encryption key, and then encrypt the S3 folder or files. First, go to KMS (Key Management Service), and create a key (Figure 15).

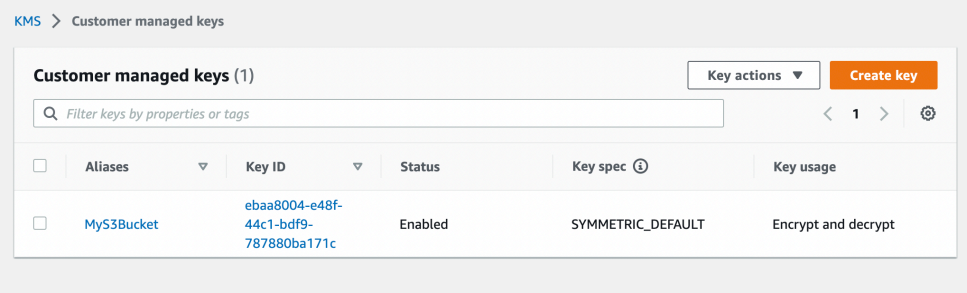

Figure 15: KMS

Next we will create an AES key that will be used for encryption and decryption (Figure 16).

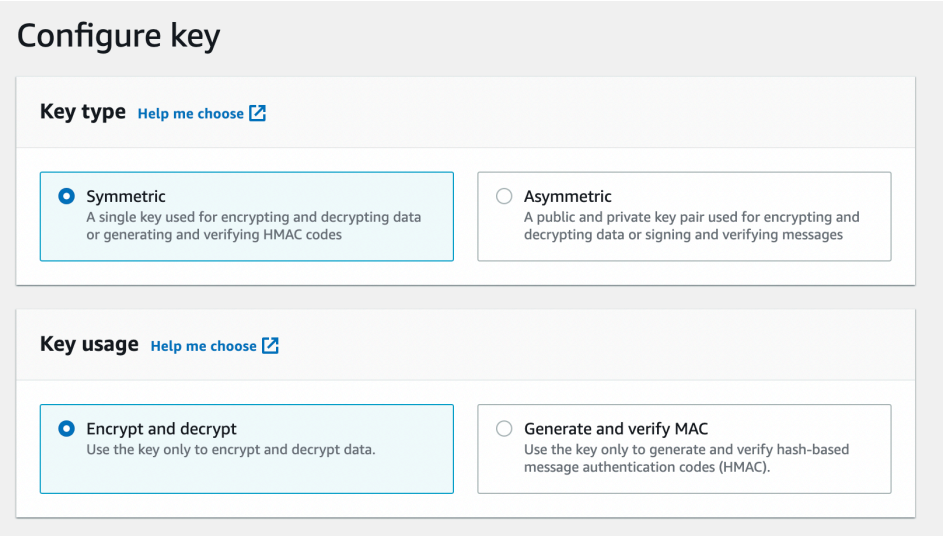

Figure 16: Creating an AES encryption key

Give the key a name and a description (Figure 17). In this case, name the key “MyKeyBucket”.

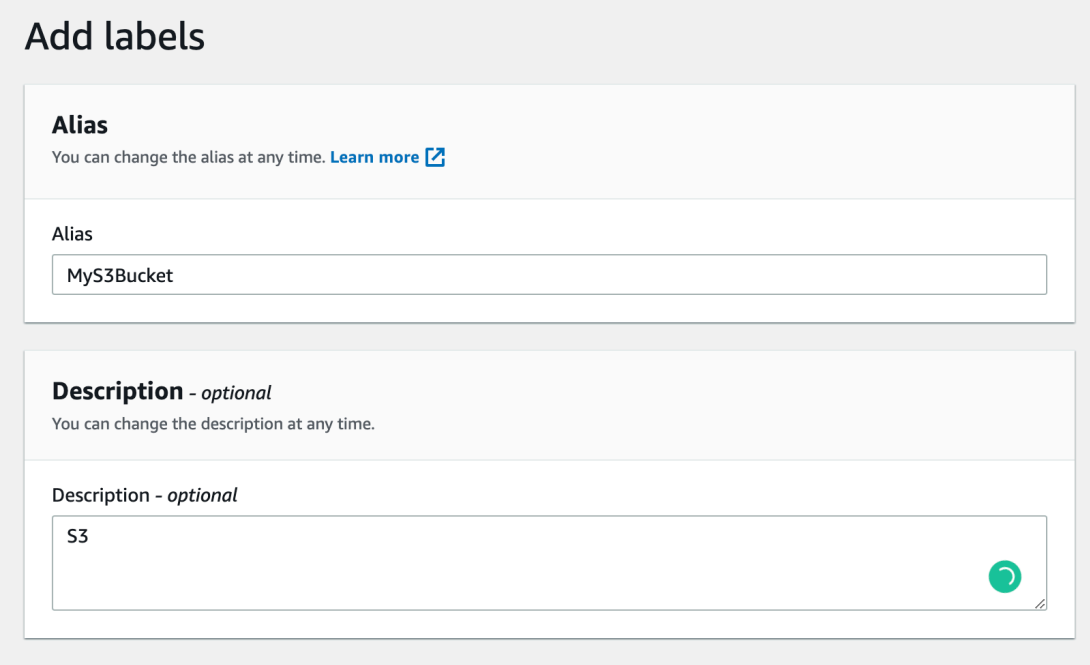

Figure 17: Creating an AES encryption key

We can then define both the administrative permissions (Figure 18) and the usage permissions (Figure 19) on the key. In this case, we will select all of the permissions (but in real-life we would lock down the usage of the key).

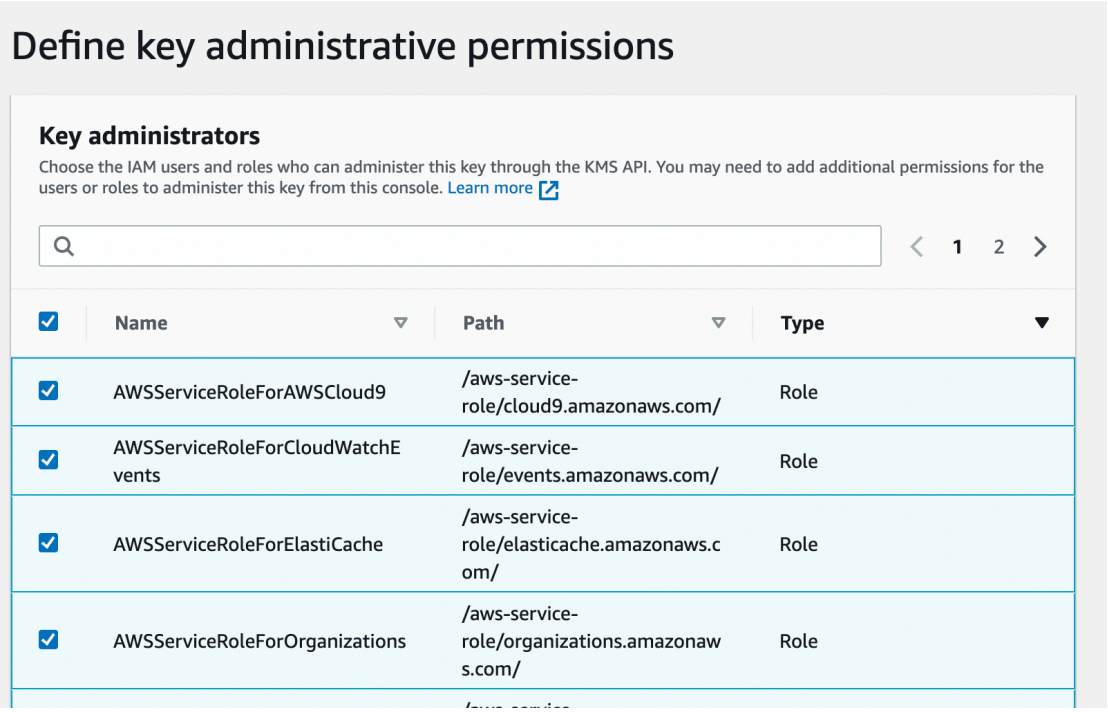

Figure 18: Admin policy

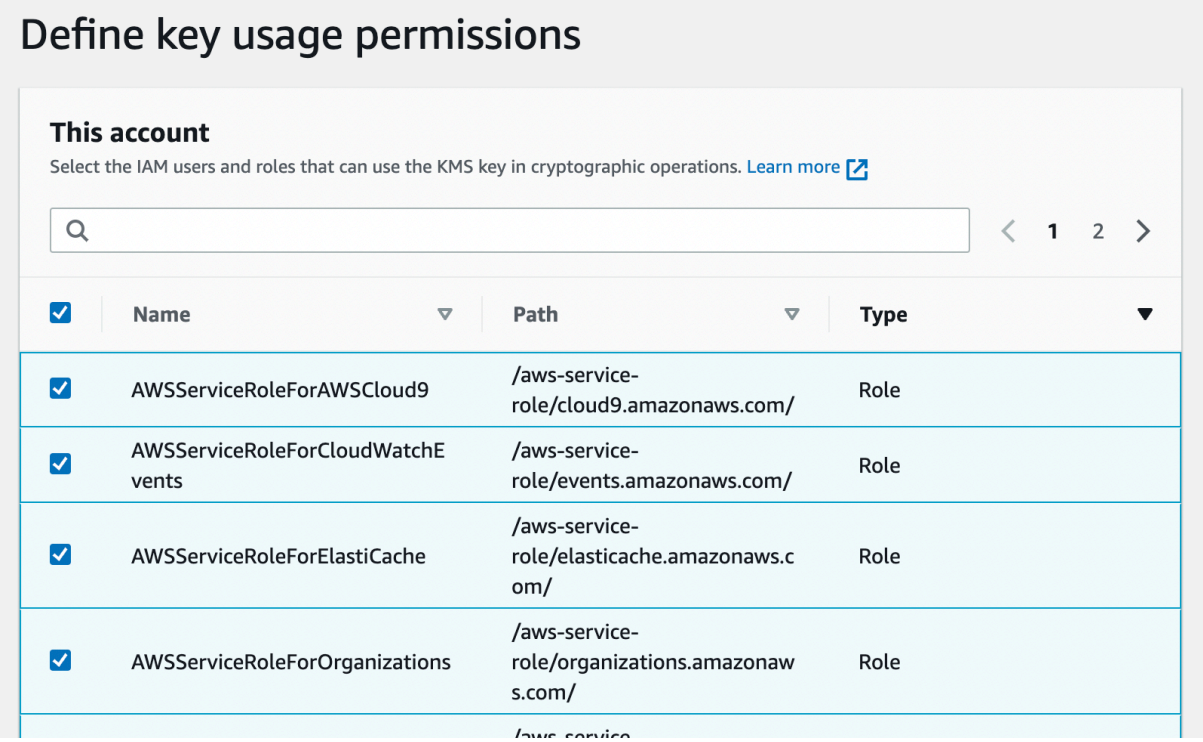

Figure 19: Usage policy

Finally we create our key (Figure 20).

What are some of the parameters that the key uses:

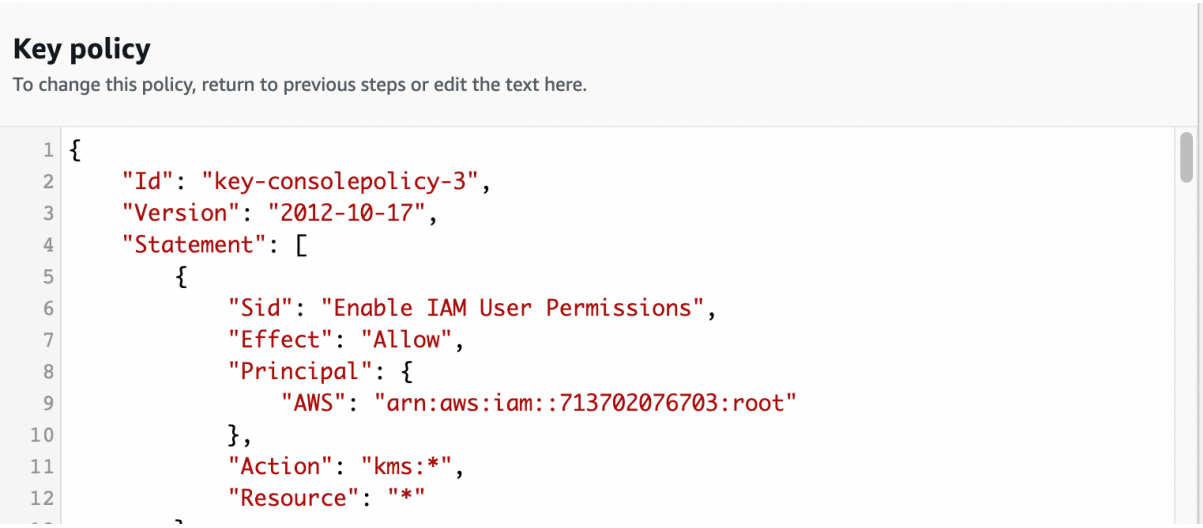

Figure 20: Key policy

Now go back to S3, and select the 1.txt file, and select Server-side encryption. Next select "AWS Management Service Key" (Figure 21), and select the ARN of the key that you have just created.

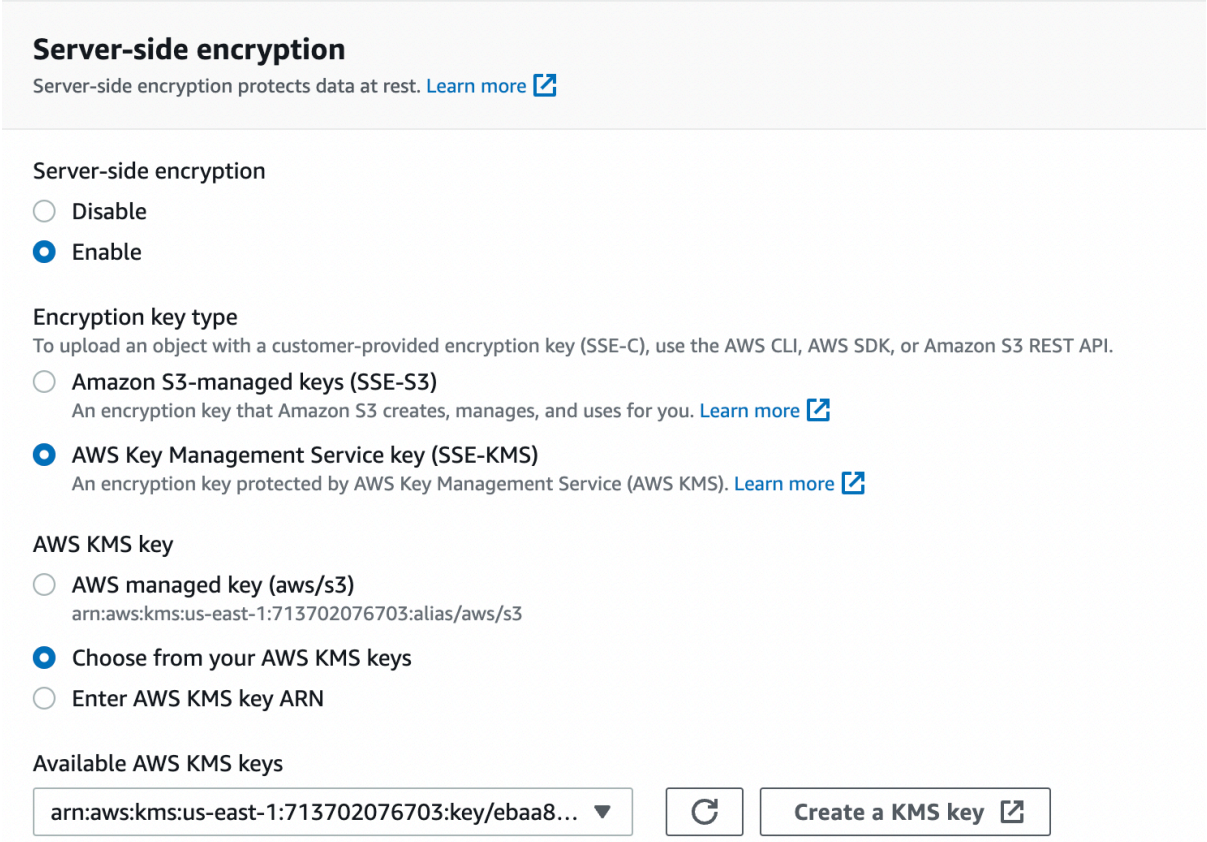

Figure 21: Applying KMS key

Did the encryption of the file succeed? Yes/No