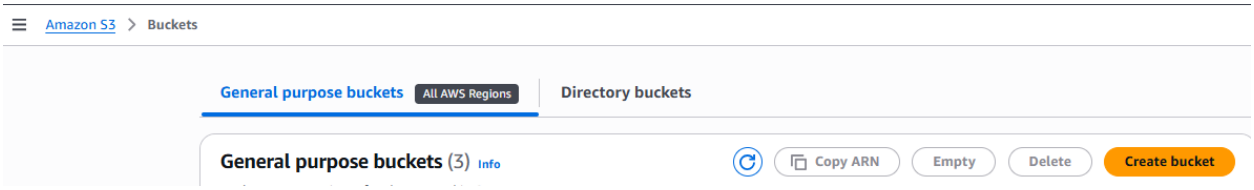
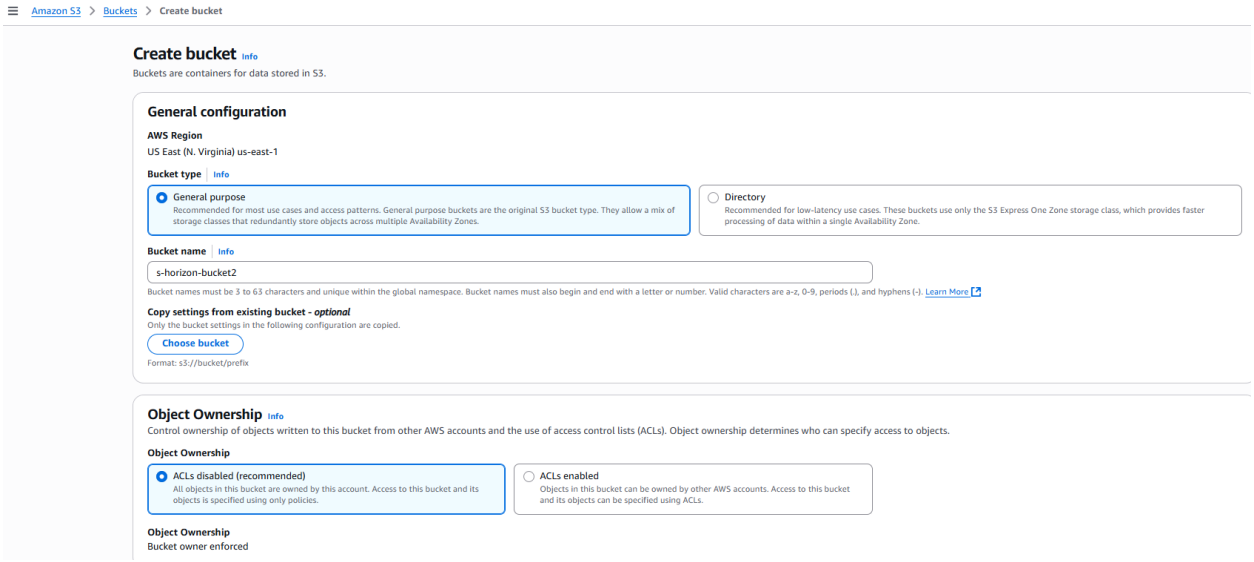
1. Create an S3 bucket and upload some objects to S3.

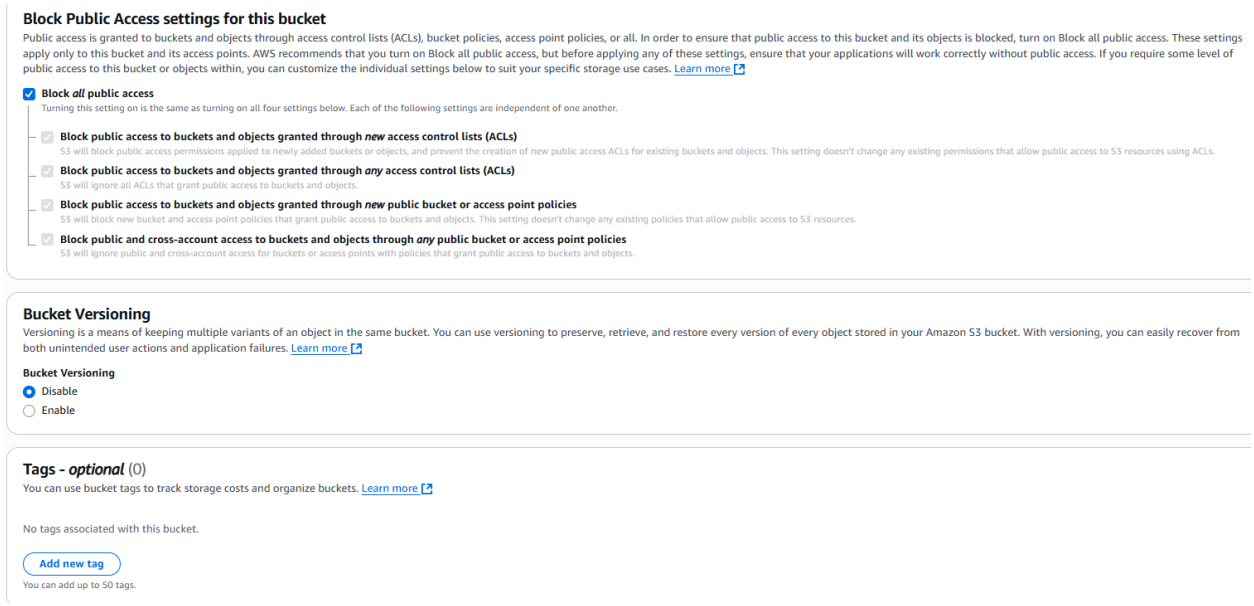
Login to aws a/c and goto s3 service and click on Create bucket:



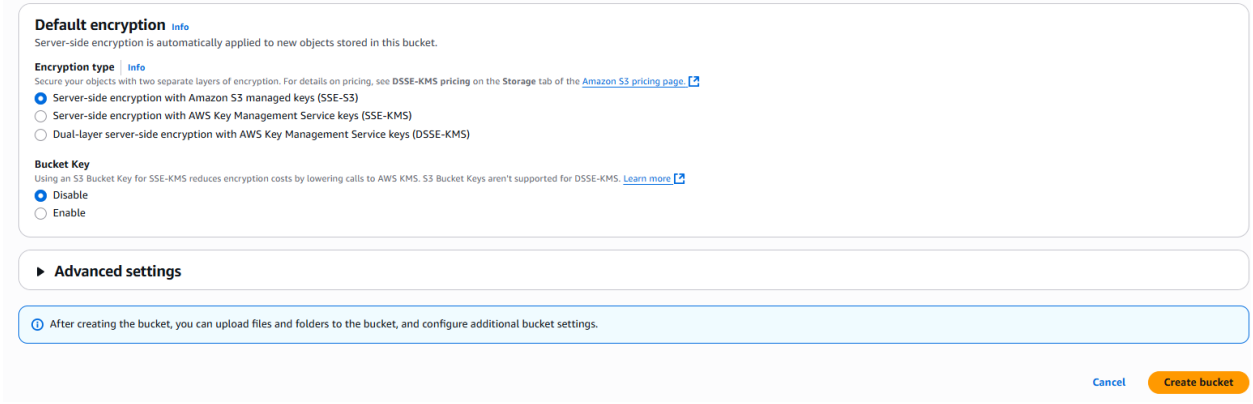
Give the bucket name which should be unique across the aws accounts:



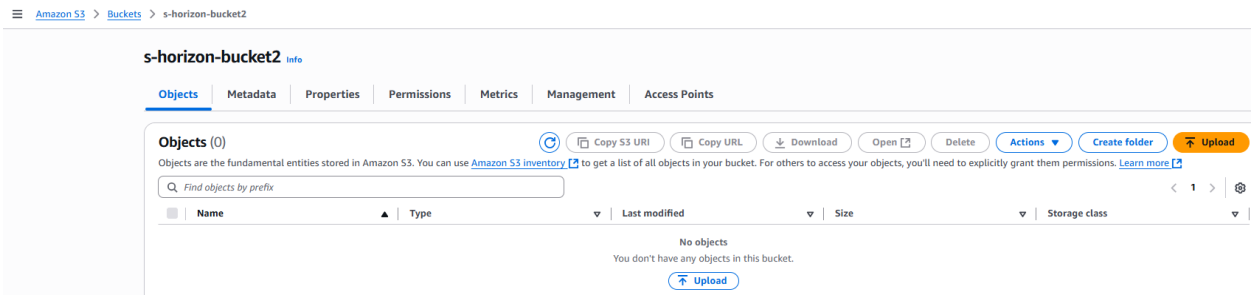
Block the public access and disable the versioning:



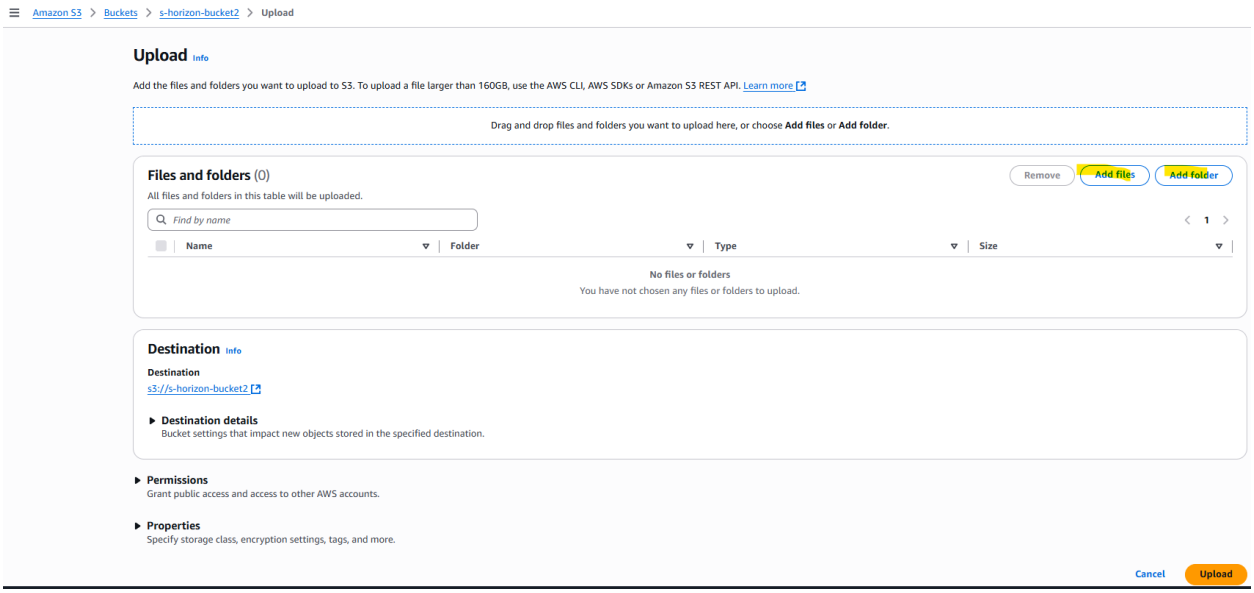
And create the bucket:



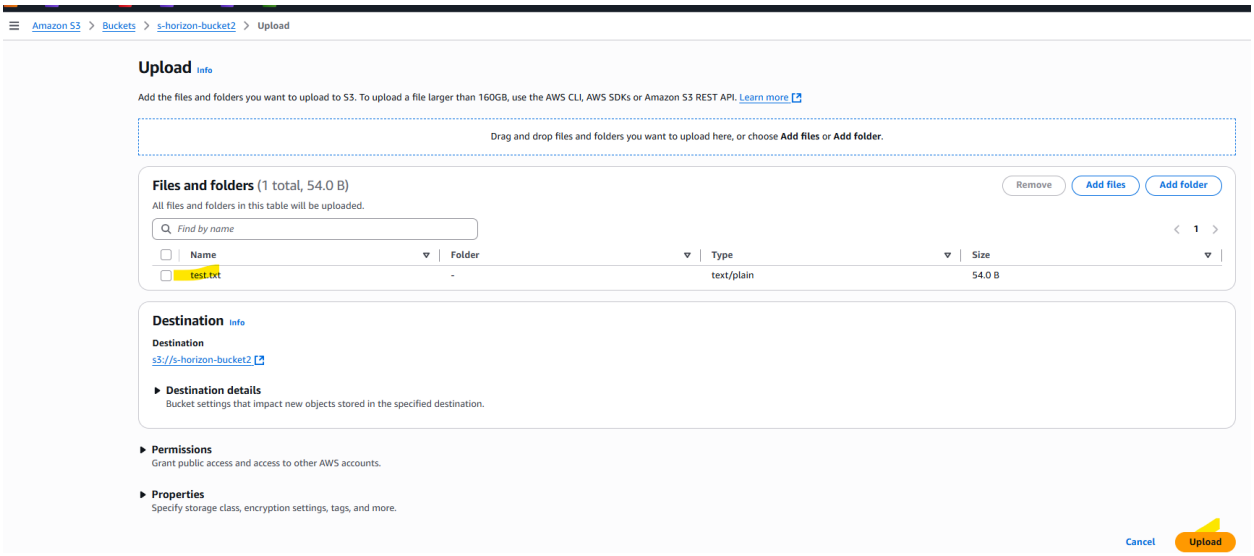
Goto the bucket and click on upload:



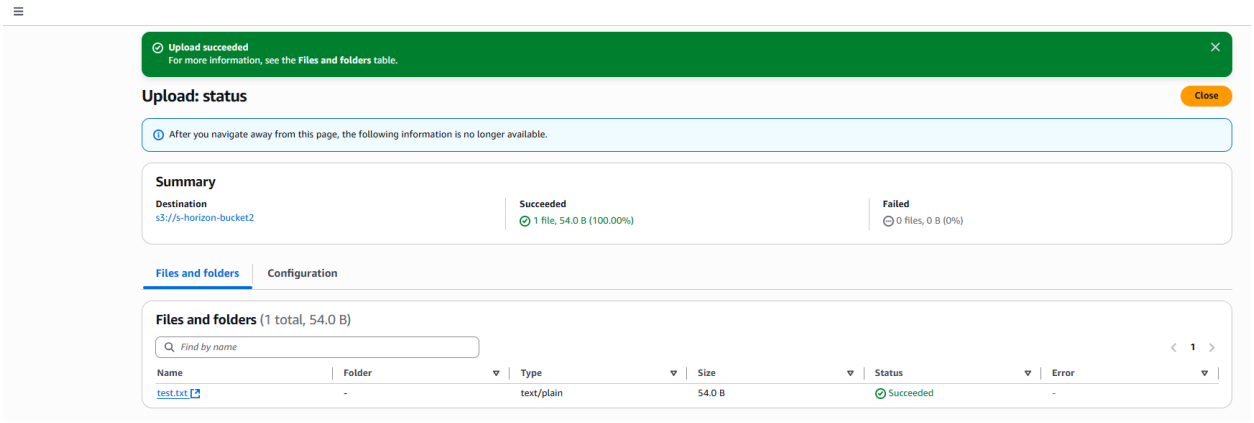
Click on Add file/Add folder:



Add any file/folder and upload:



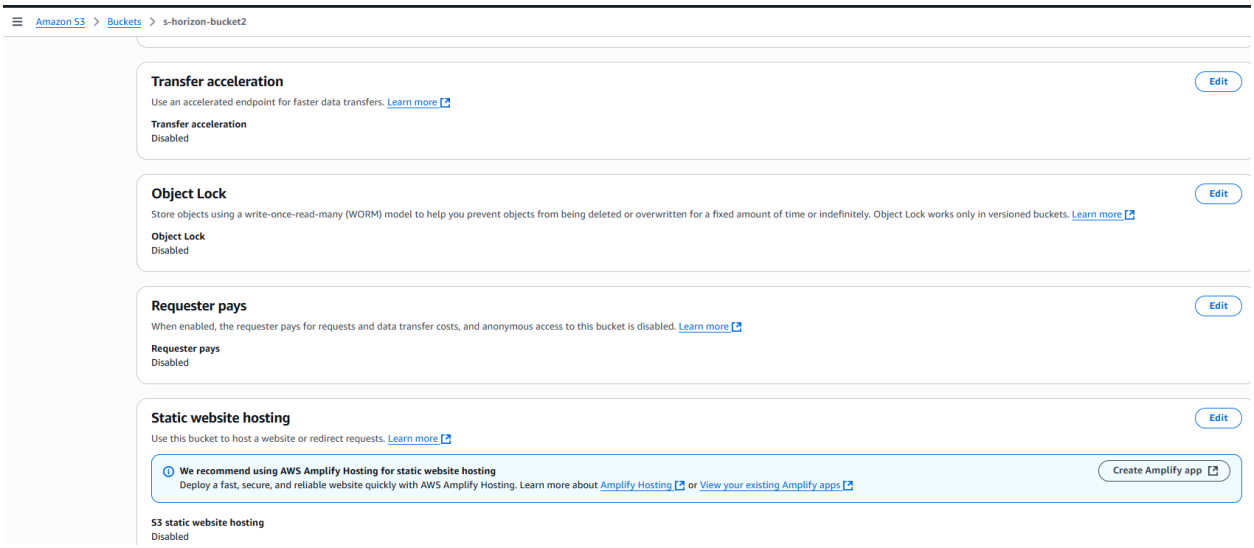
Check the file got uploaded:



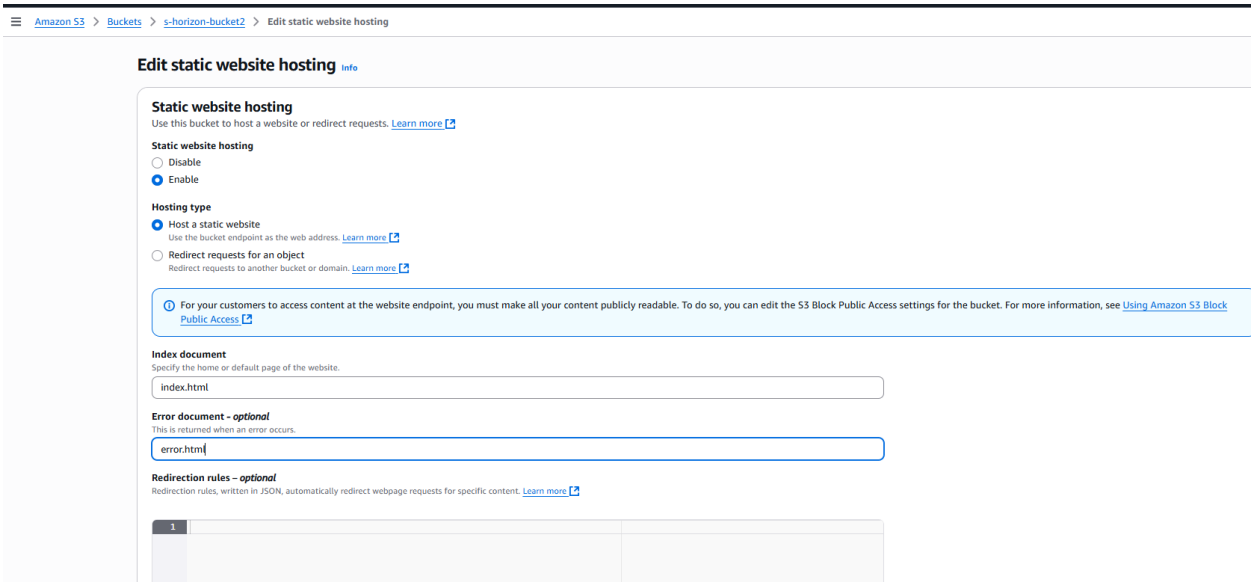
---

2. Deploy a static website in the S3 bucket.

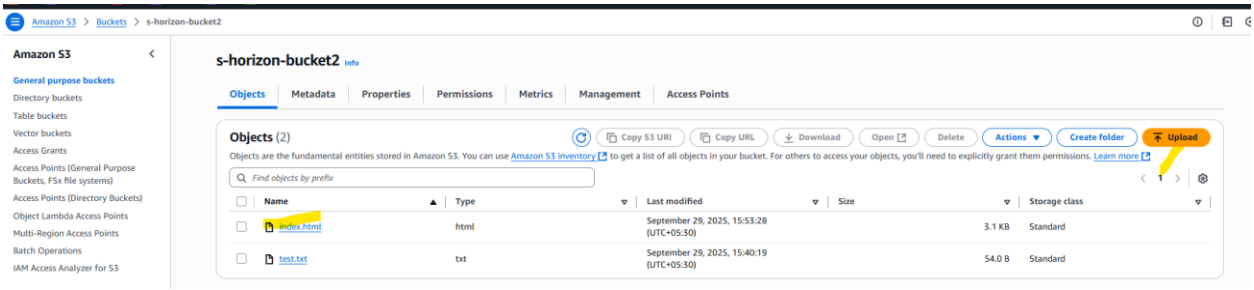
Goto the bucket->Properties and click on ‘Static Website Hosting’ ->Edit:



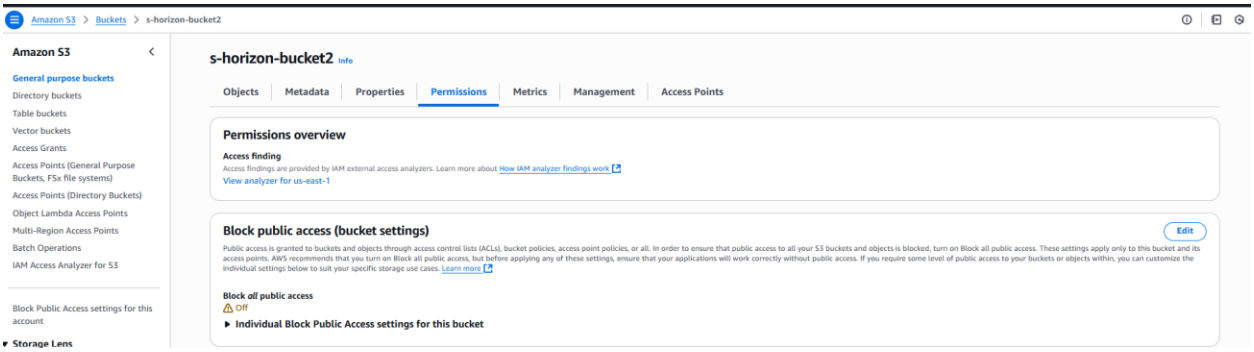
Enable the ‘static website hosting’ and give the ‘index.html’ and ‘error.html’ and save:



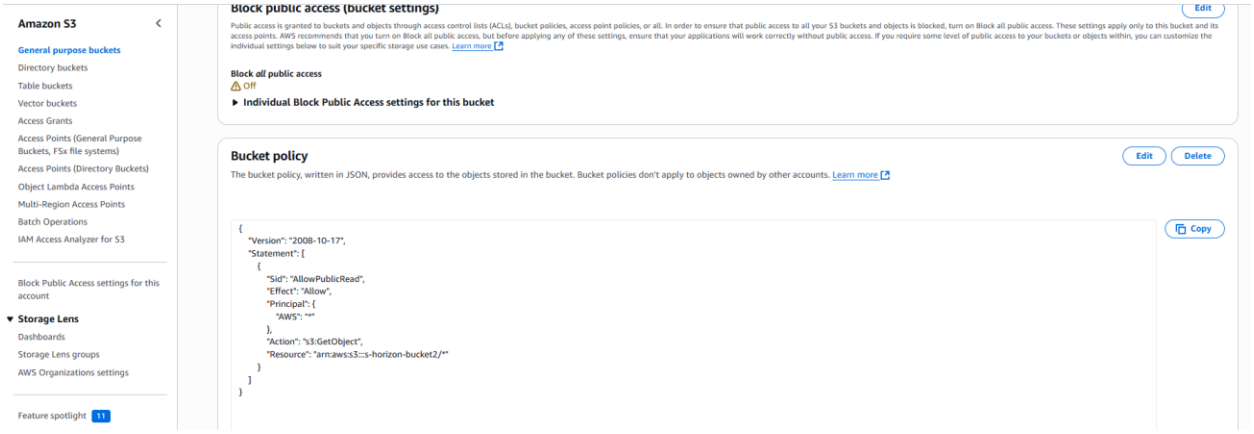
Create a file with name ‘index.html’ and upload to the above created bucket:



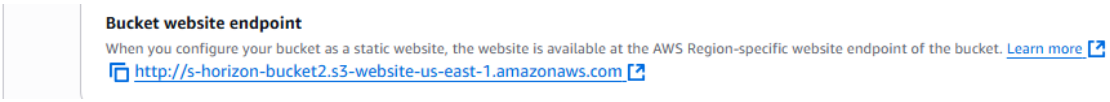
Then remove the checkbox from ‘Block public access”, :



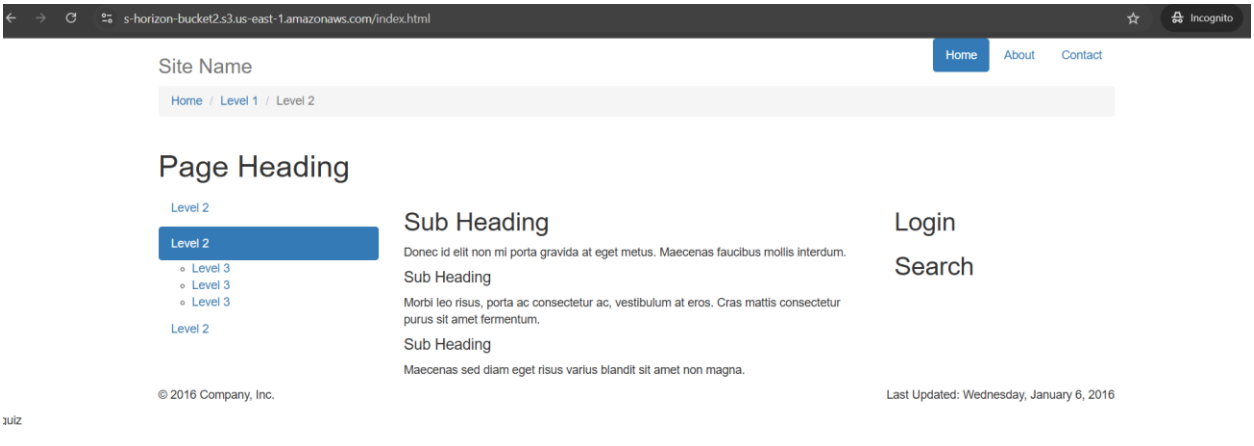
And remove the ACL’s and attach policy:



Goto the properties of the bucket and in the down side we get the url:



Will get the static website:

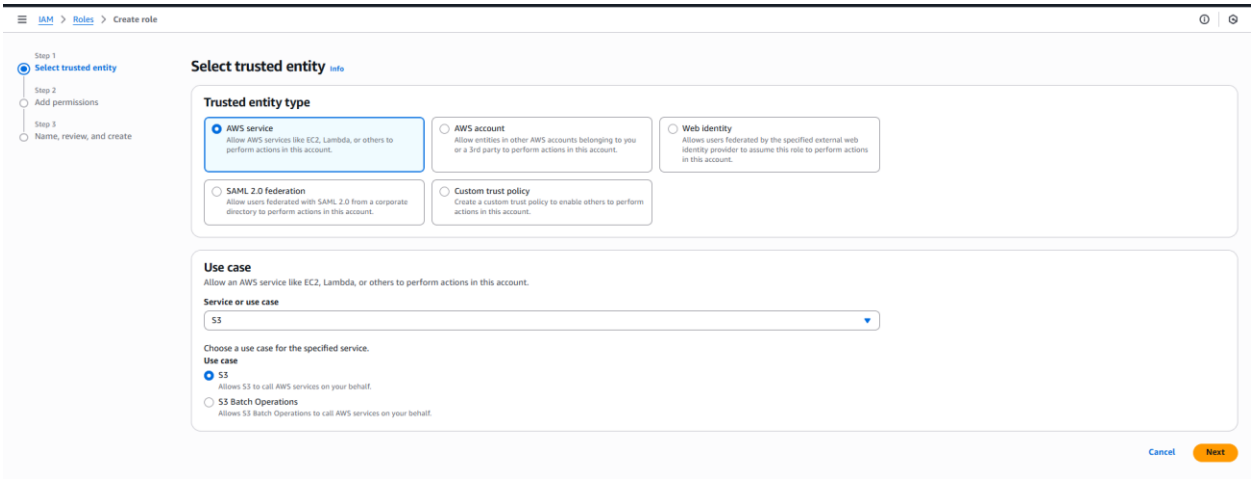


----------done----------

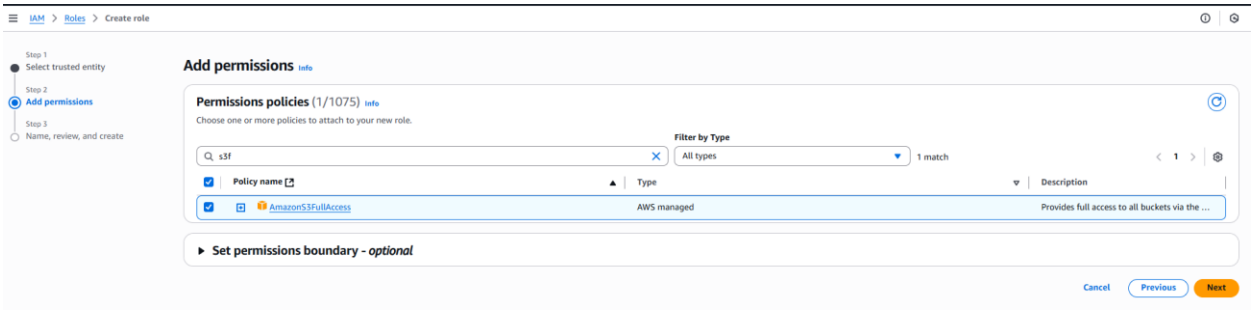
3. Enable cross-region replication on S3 buckets.

Create 2 buckets in 2 different Regions and enable the Versioning for both buckets.

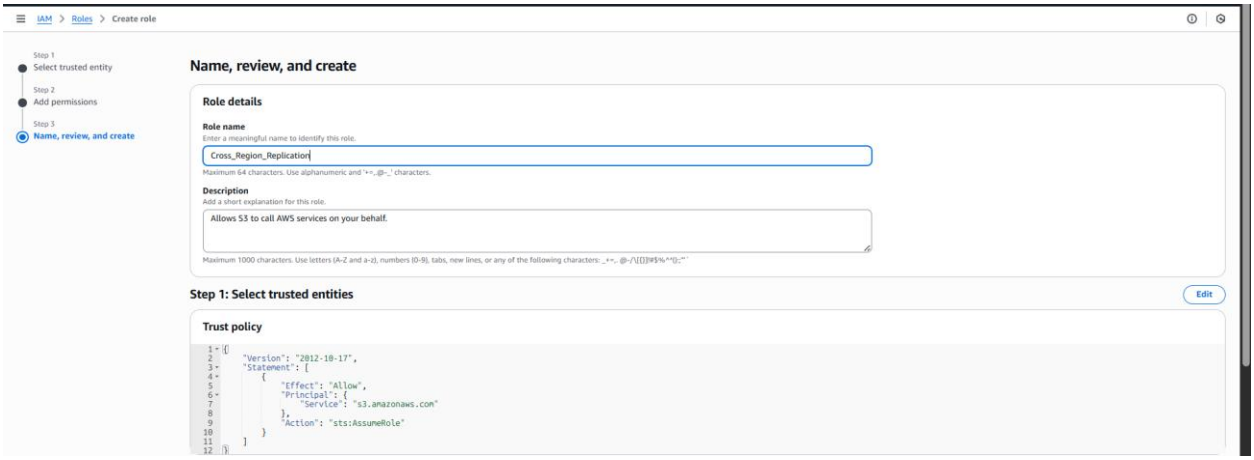
Create IAM role with ‘aws service’ and give the service as ‘s3’ and click on Next:



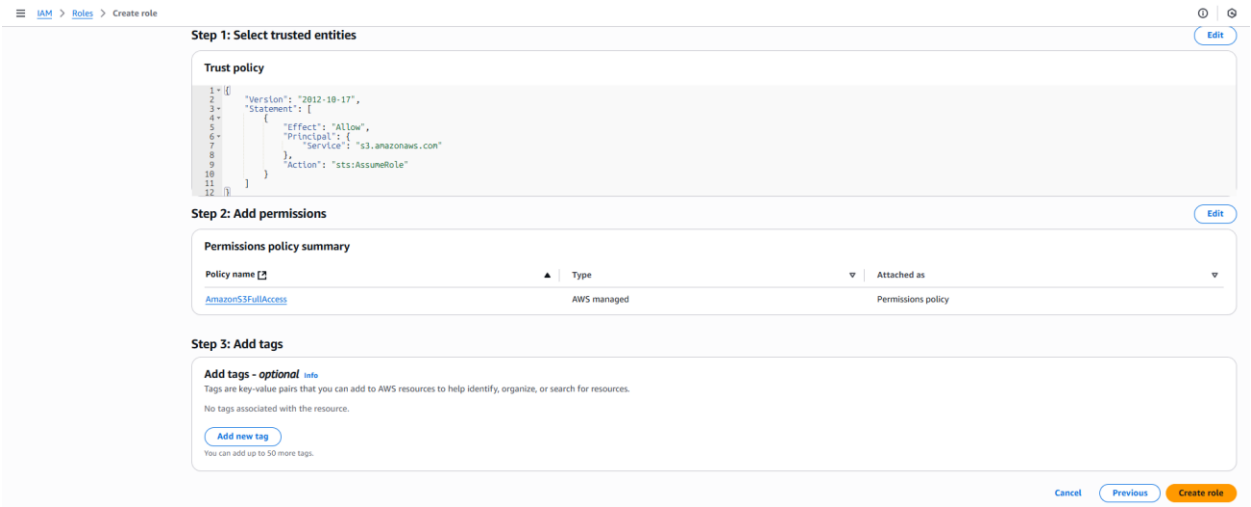
and select the policy ‘AmazonS3FullAccess’:



Give the role name:



And click on ‘Create Role’:



And Create another policy and attach to this role :

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersionForReplication",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
        }
    ]
}
```

Note: here replace the name of the bucket of source and destination with your own source and destination buckets, below is also same policy of json:

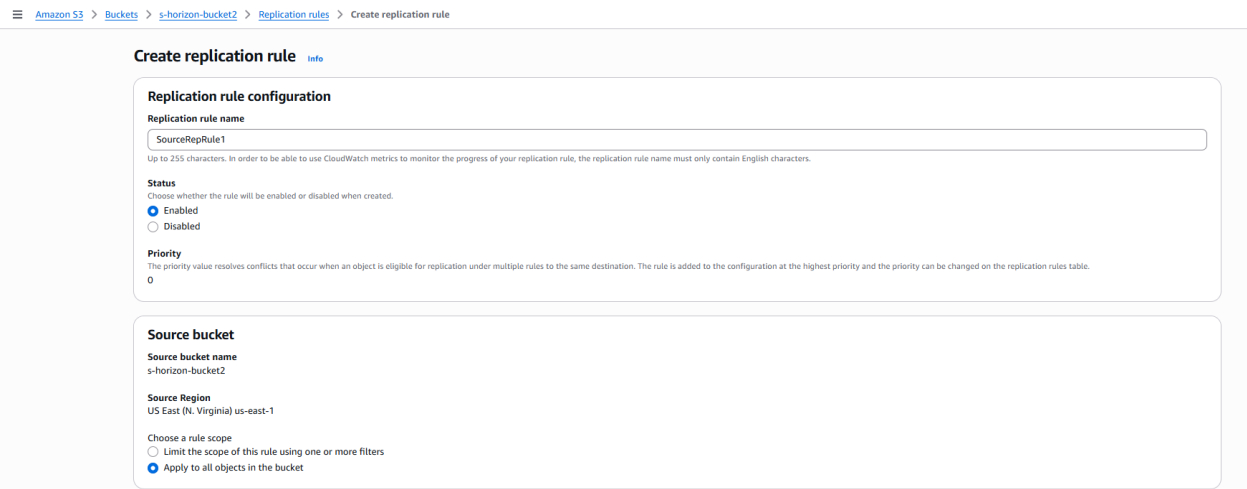
{

```
"Version":"2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetReplicationConfiguration",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-source-bucket"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObjectVersionForReplication",
      "s3:GetObjectVersionAcl",
      "s3:GetObjectVersionTagging"
    ],
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
```

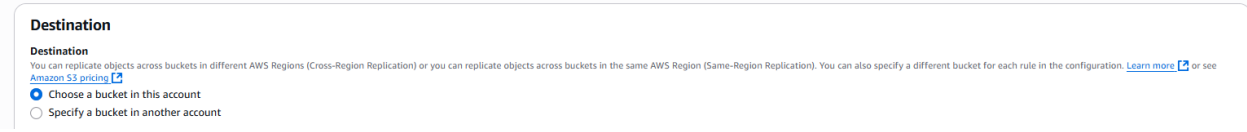
```
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    }
  ]
}
```

Activating Cross region Replication

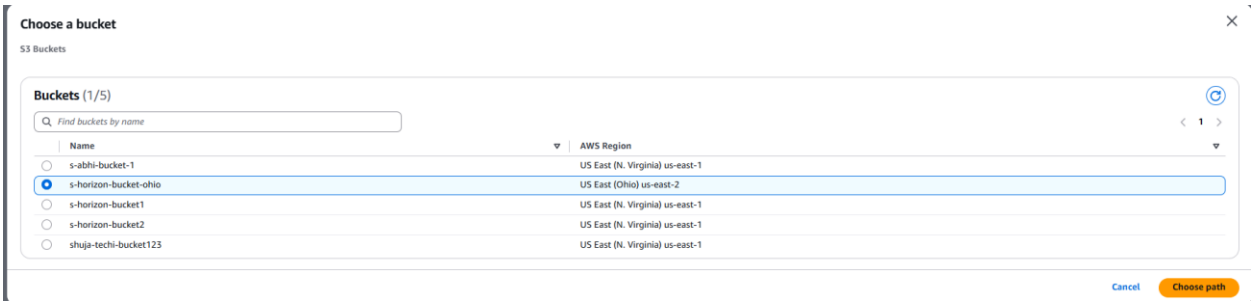
- Click on the Source bucket. Go to the management and Click on the management tab.
- Create Replication Rule, give rule name and source bucket choose ‘Apply to all objects in the bucket’ because I want to apply the rule to all the objects in the buckets:

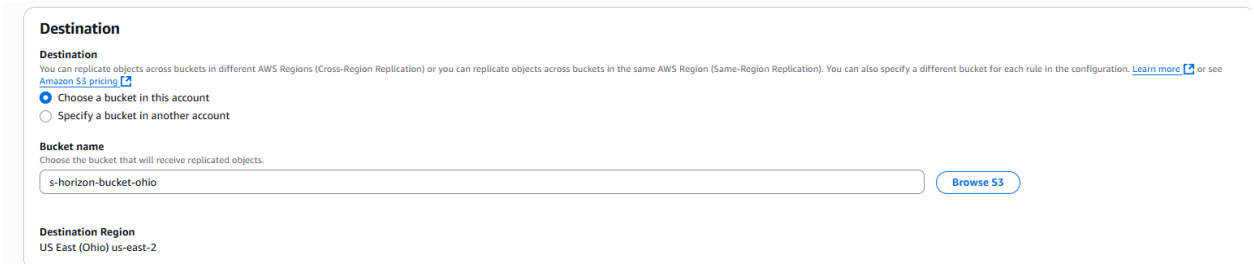


Scroll down and choose the Destination :choose first option since the destination bucket is also in the same account:

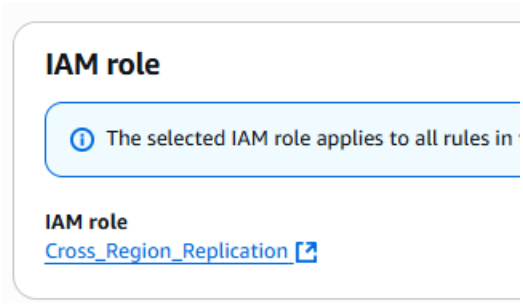


Click on Browse S3 and choose your destination bucket:

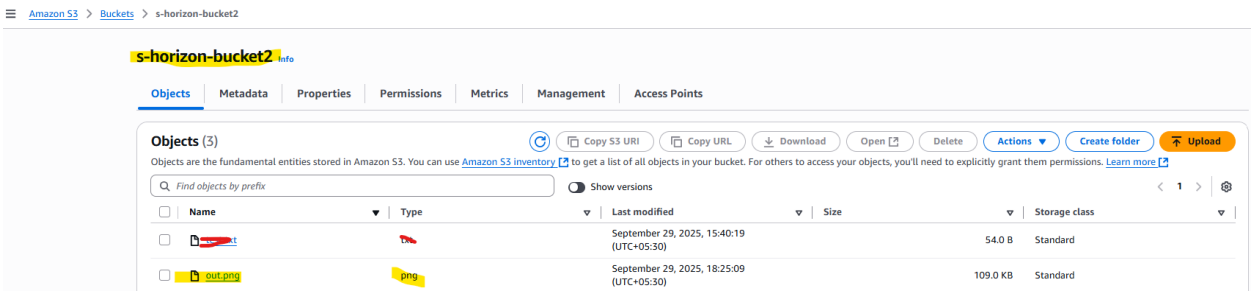




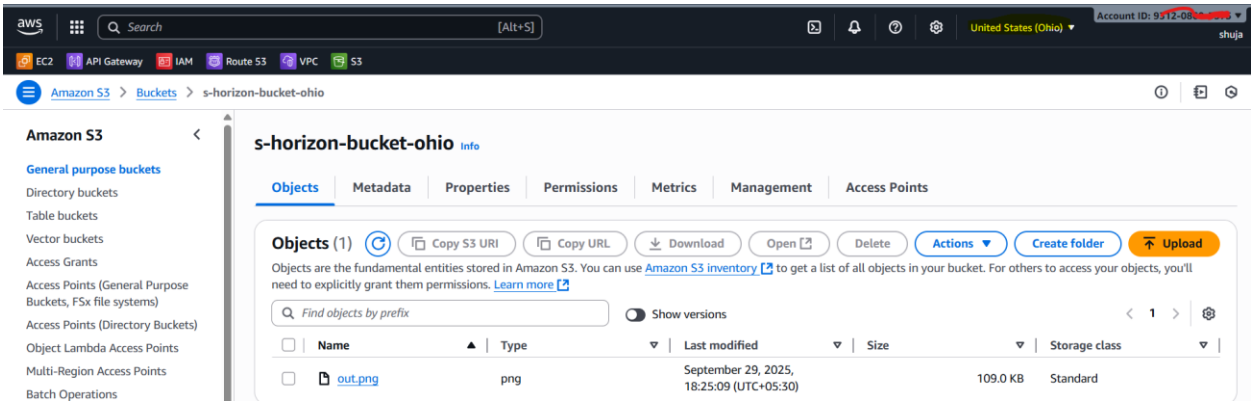
Next, select the IAM role: [the role which has created above :



Upload files to Source Bucket to Replicate to the Destination Bucket.



- If you now check in the two buckets, files are uploaded successfully in the source bucket as well as the destination bucket due to CRR.



----------done------

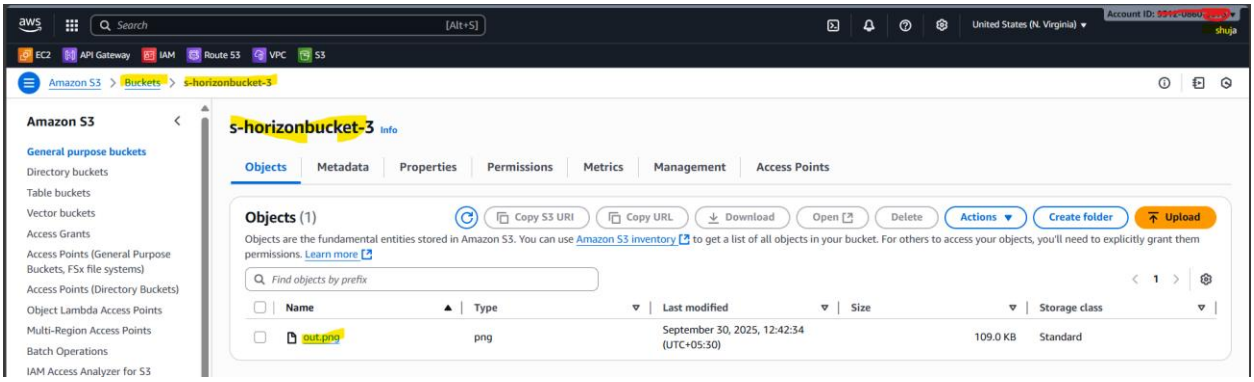
4. Configure a bucket policy so only the Admin user can see the objects of the S3 bucket.

   Create a bucket/to existing bucket add the below policy :

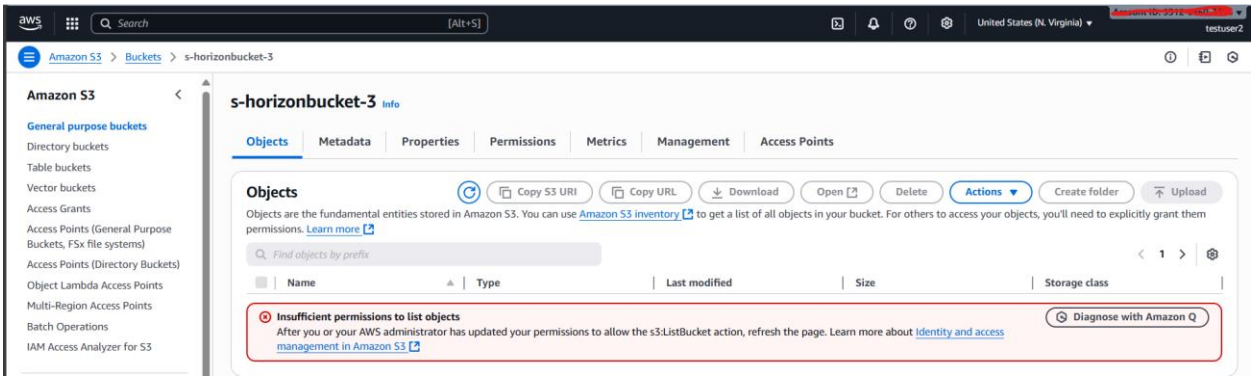
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAdminFullRead",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::931208603575:user/testuser"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::s-horizonbucket-3",
                "arn:aws:s3:::s-horizonbucket-3/*"
            ]
        },
```

```
        {
            "Sid": "DenyAllExceptAdmin",
            "Effect": "Deny",
            "NotPrincipal": {
                "AWS": "arn:aws:iam::931208603575:user/testuser"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::s-horizonbucket-3",
                "arn:aws:s3:::s-horizonbucket-3/*"
            ]
        }
    ]
}
```

With user ‘testuser’ is able to access all the buckets.



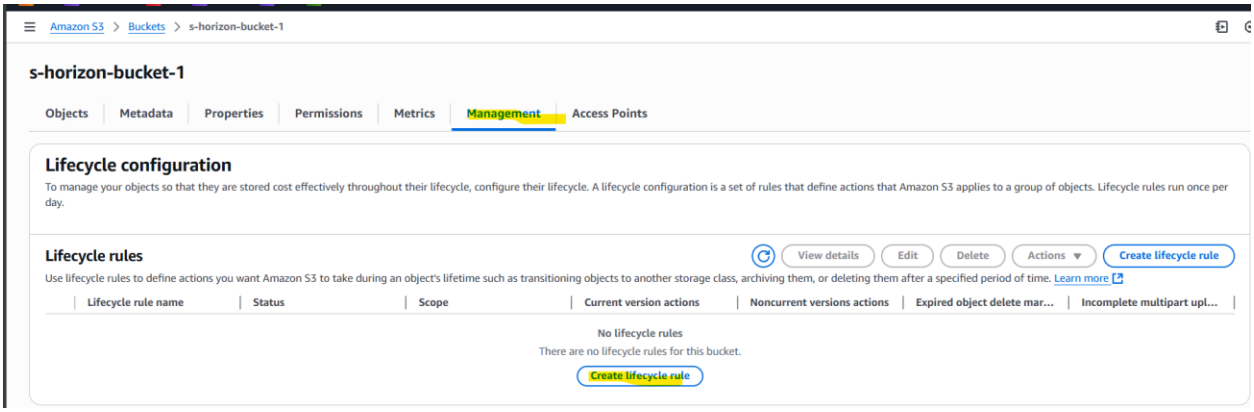
Login with ‘testuser2’ which cannot access the objects of the bucket:



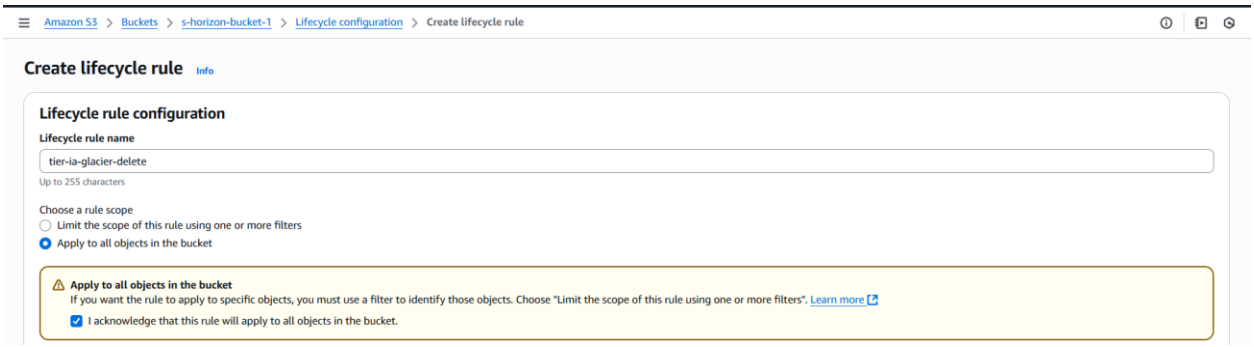
----------done---------

5. Set up lifecycle policies to automatically transition or delete objects based on specific criteria.

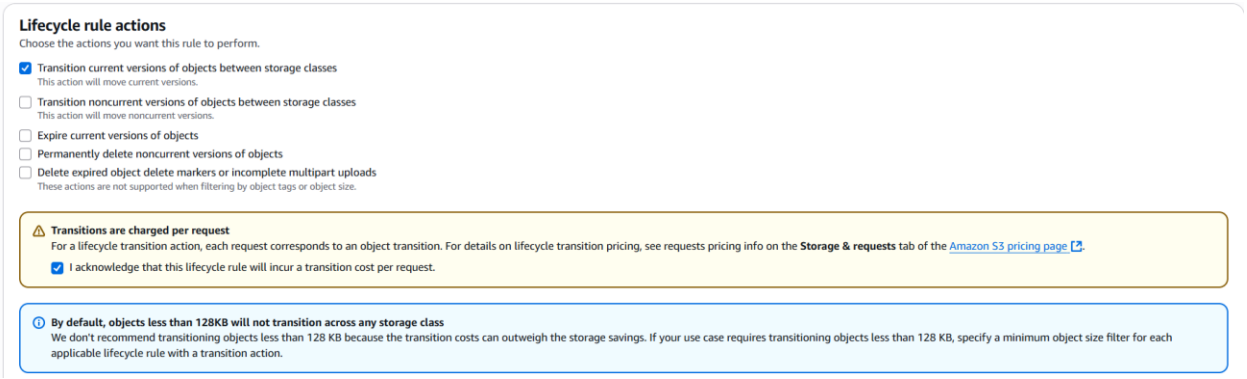
Select your bucket and Goto Management tab:



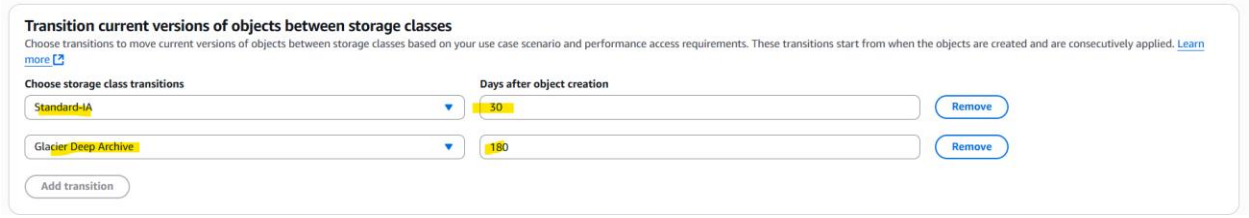
Give the name for the rule and select any filter like if you want for specific objects or all objects you want to do transition, here I want to do for all objects object:



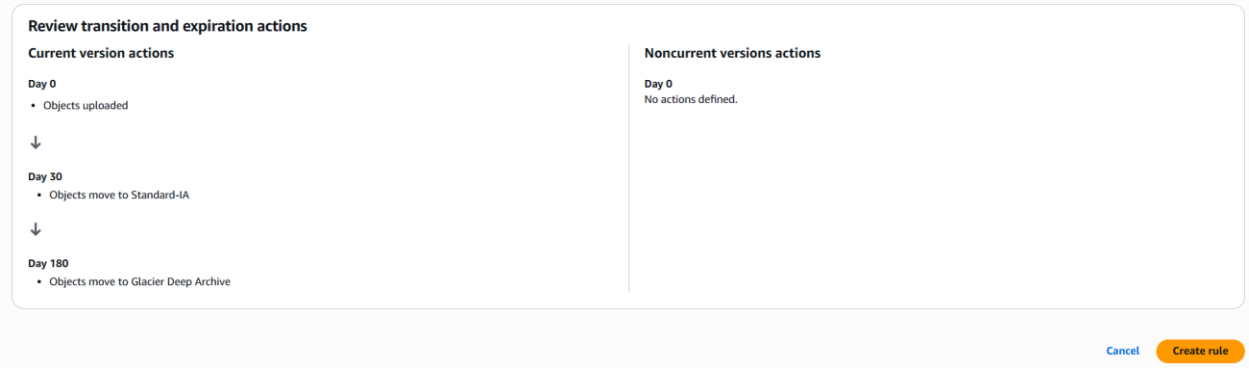
And select which version you want to do transition: here I want to do transition for current version:



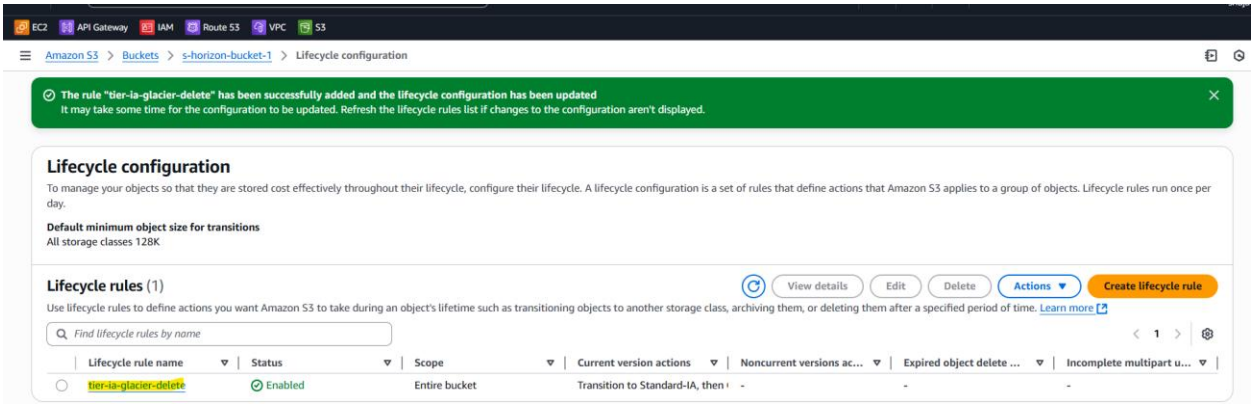
Select your required storage class and no. of days after object creation:



And click on Create rule:



Then a transition rule will get created which is Enabled:



----------done----------

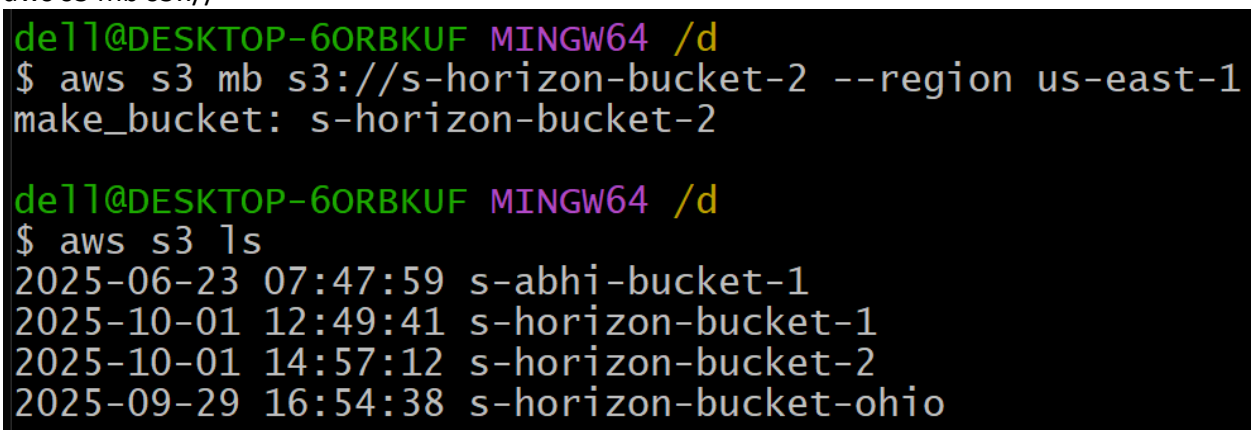
6. Push some objects to S3 using the AWS CLI.

Configure credentials / profile

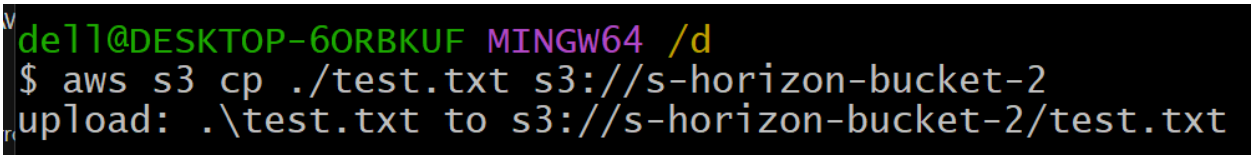
Use aws configure for a default profile, or set a named profile (recommended for multiple accounts):
Using a named profile:
aws configure --profile myprofile
Or export environment variables (temporary / CI usage):

export AWS_ACCESS_KEY_ID="AKIA..."
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCY"
export AWS_DEFAULT_REGION="ap-south-1"
on Windows PowerShell:
$env:AWS_ACCESS_KEY_ID="AKIA..."
Use --profile myprofile on commands to use a named profile.

Create an S3 bucket
aws s3 mb s3://my-bucket-name --region <region> --profile myprofile
aws s3 mb s3:://



Upload a single file (simple)
aws s3 cp ./localfile.txt s3://my-bucket-name/path/in/bucket/localfile.txt --profile myprofile



Common options:

- Make public (be careful!): --acl public-read
- Server-side encryption (SSE-S3): --sse AES256
- SSE with KMS: --sse aws:kms --sse-kms-key-id arn:aws:kms:...

Example with SSE and metadata:

aws s3 cp ./index.html s3://my-bucket-name/site/index.html \

--content-type "text/html" \

--metadata author=avinaash,env=prod \

--sse AES256 \

--profile myprofile

Upload a directory (multiple files) — sync (recommended) or recursive cp

Sync (uploads new/changed files):

aws s3 sync ./local-folder s3://my-bucket-name/folder-path --profile myprofile

```
dell@DESKTOP-6ORBKUF MINGW64 /d
$ aws s3 sync ./dir1/ s3://s-horizon-bucket-2/folder/
upload: dir1\file2.txt to s3://s-horizon-bucket-2/folder/file2.txt
upload: dir1\file1.txt to s3://s-horizon-bucket-2/folder/file1.txt
upload: dir1\file3.txt to s3://s-horizon-bucket-2/folder/file3.txt
```

Recursive copy (uploads entire tree):

aws s3 cp ./local-folder s3://my-bucket-name/folder-path --recursive --profile myprofile

Filter uploads:

include only .html files

aws s3 sync ./local-folder s3://my-bucket-name/folder-path --exclude "*" --include "*.html" --profi

Verify uploaded objects

List objects:

aws s3 ls s3://my-bucket-name/folder-path/ --recursive --profile myprofile

```
dell@DESKTOP-6ORBKUF MINGW64 /d
$ aws s3 ls s3://s-horizon-bucket-2/folder --recursive
2025-10-01 15:10:18          0 folder/file1.txt
2025-10-01 15:10:17          0 folder/file2.txt
2025-10-01 15:10:18          0 folder/file3.txt
```

Advanced: multipart / large files

aws s3 cp and sync automatically use multipart uploads for large files. For manual control use s3api multipart commands (create-multipart-upload, upload-part, complete-multipart-upload)

----------done----------

7. Write a Bash script to create an S3 bucket.

```
dell@DESKTOP-6ORBKUF MINGW64 /d
$ cat create_bucket.sh
#!/bin/bash

bucket_name=s-horizon-bucket-3
aws s3 mb s3://${bucket_name} --region us-east-1
if [ $? -eq 0 ]; then
echo "Bucket created"
else
echo "Bucket not created"
fi
```

Execute the script

```
dell@DESKTOP-6ORBKUF MINGW64 /d
$ ./create_bucket.sh
make_bucket: s-horizon-bucket-3
Bucket created
```

```
#!/usr/bin/env bash


# Usage: ./create_s3_bucket.sh <bucket-name> <region> [profile]


BUCKET_NAME=$1

REGION=$2

PROFILE=${3:-default}


# --- Function: Print error and exit ---

error_exit() {
```

```
  echo "❌ ERROR: $1"
  exit 1
}

# --- Step 1: Check input ---
if [[ -z "$BUCKET_NAME" || -z "$REGION" ]]; then
  echo "Usage: $0 <bucket-name> <region> [profile]"
  exit 1
fi

# --- Step 2: Validate bucket name ---
if [[ ! $BUCKET_NAME =~ ^[a-z0-9.-]{3,63}$ ]]; then
  error_exit "Invalid bucket name. Must be 3–63 chars, lowercase, numbers, dots, or hyphens."
fi
if [[ $BUCKET_NAME =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]]; then
  error_exit "Bucket name cannot look like an IP address."
fi
if [[ $BUCKET_NAME =~ ^[\.-] || $BUCKET_NAME =~ [\.-]$ ]]; then
  error_exit "Bucket name cannot start or end with '.' or '-'."
fi

# --- Step 3: Check AWS CLI installed ---
if ! command -v aws &>/dev/null; then
  error_exit "AWS CLI is not installed. Install AWS CLI v2."
fi
```

```
# --- Step 4: Check AWS credentials ---
if ! aws sts get-caller-identity --profile "$PROFILE" &>/dev/null; then
  error_exit "AWS CLI credentials not configured for profile [$PROFILE]. Run: aws configure --profile $PROFILE"
fi

# --- Step 5: Check if bucket already exists ---
EXISTING=$(aws s3api head-bucket --bucket "$BUCKET_NAME" --profile "$PROFILE" 2>&1)

if [[ $? -eq 0 ]]; then
  echo "☑ Bucket [$BUCKET_NAME] already exists and you own it."
  exit 0
elif echo "$EXISTING" | grep -q '403'; then
  error_exit "Bucket [$BUCKET_NAME] already exists but is owned by another account."
elif echo "$EXISTING" | grep -q '404'; then
  echo "ⓘ Bucket [$BUCKET_NAME] does not exist, proceeding with creation..."
else
  error_exit "Unexpected error while checking bucket: $EXISTING"
fi

# --- Step 6: Create bucket ---
if [[ "$REGION" == "us-east-1" ]]; then
  CREATE_OUT=$(aws s3api create-bucket --bucket "$BUCKET_NAME" --region "$REGION" --profile "$PROFILE" 2>&1)
else
```

```
  CREATE_OUT=$(aws s3api create-bucket --bucket "$BUCKET_NAME" --region "$REGION" \
    --create-bucket-configuration LocationConstraint="$REGION" --profile "$PROFILE" 2>&1)
fi

if [[ $? -ne 0 ]]; then
  error_exit "Failed to create bucket: $CREATE_OUT"
fi

# --- Step 7: Verify bucket creation ---
if aws s3api head-bucket --bucket "$BUCKET_NAME" --profile "$PROFILE" 2>/dev/null; then
  echo " 🎉 SUCCESS: Bucket [$BUCKET_NAME] created in region [$REGION] using profile [$PROFILE]."
else
  error_exit "Bucket creation failed verification step."
Fi
```

---

aws s3api is the **low-level interface** of the AWS CLI for S3. Unlike aws s3 (which is higher-level and easier for uploads/downloads), aws s3api exposes the **full S3 API** with all options.

Here are some **practical examples**:

1. Create a bucket

bash

```
# For us-east-1
aws s3api create-bucket --bucket my-unique-bucket --region us-east-1

# For other regions (must specify LocationConstraint)
aws s3api create-bucket \
  --bucket my-unique-bucket \
  --region ap-south-1 \
  --create-bucket-configuration LocationConstraint=ap-south-1
```

2. List all buckets in your account

bash

```
aws s3api list-buckets
```

3. Upload (Put) an object

bash

```
aws s3api put-object \
  --bucket my-unique-bucket \
  --key folder1/hello.txt \
  --body ./hello.txt
```

4. Download (Get) an object

bash

```
aws s3api get-object \
  --bucket my-unique-bucket \
  --key folder1/hello.txt \
  ./downloaded-hello.txt
```

---

5. List objects in a bucket

bash

```
aws s3api list-objects \
  --bucket my-unique-bucket \
  --prefix folder1/
```

6. Get object metadata (HEAD request)

bash

```
aws s3api head-object \
  --bucket my-unique-bucket \
  --key folder1/hello.txt
```

7. Enable versioning on a bucket

bash

```
aws s3api put-bucket-versioning \
  --bucket my-unique-bucket \
  --versioning-configuration Status=Enabled
```

8. Block all public access

bash Copy code

```
aws s3api put-bucket-public-access-block \
  --bucket my-unique-bucket \
  --public-access-block-configuration \
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
```

----------done----------

8. Upload a 1 GB file to S3 using the CLI.

Uploading a **1 GB file** to Amazon S3 using the AWS CLI is straightforward. The good news: you don't need to worry about splitting the file — the AWS CLI automatically uses **multipart upload** for large files (anything over **8 MB** by default).

Generate a 1 GB test file (Linux/Mac)

```
dell@DESKTOP-6ORBKUF MINGW64 /d
$ dd if=/dev/zero of=bigfile.bin bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 0.500256 s, 2.1 GB/s
```

Upload the 1 GB file to S3

Option A: High-level aws s3 (easiest)

aws s3 cp ./bigfile.bin s3://my-bigfile-bucket/uploads/bigfile.bin

```
dell@DESKTOP-6ORBKUF MINGW64 /d
$ aws s3 cp ./bigfile.bin s3://s-horizon-bucket-3
upload: .\bigfile.bin to s3://s-horizon-bucket-3/bigfile.bin
```

Option B: Low-level aws s3api (fine-grained control)

aws s3api put-object \

--bucket my-bigfile-bucket \

--key uploads/bigfile.bin \

--body ./bigfile.bin

Verify upload

aws s3 ls s3://my-bigfile-bucket/uploads/

Or check metadata:

aws s3api head-object \

--bucket my-bigfile-bucket \

--key uploads/bigfile.bin

(Optional) Speed it up

- Use **multipart concurrency**:

aws s3 cp ./bigfile.bin s3://my-bigfile-bucket/uploads/ --expected-size 1073741824

- Add **--storage-class STANDARD_IA** if you want cheaper storage.
- Use --sse AES256 for encryption.

----------completed----------