Amazon S3: Bucket creation and Objects Access Control using AWS CLI

JULEANNY NAVAS
AWS Cloud Computing

Introduction

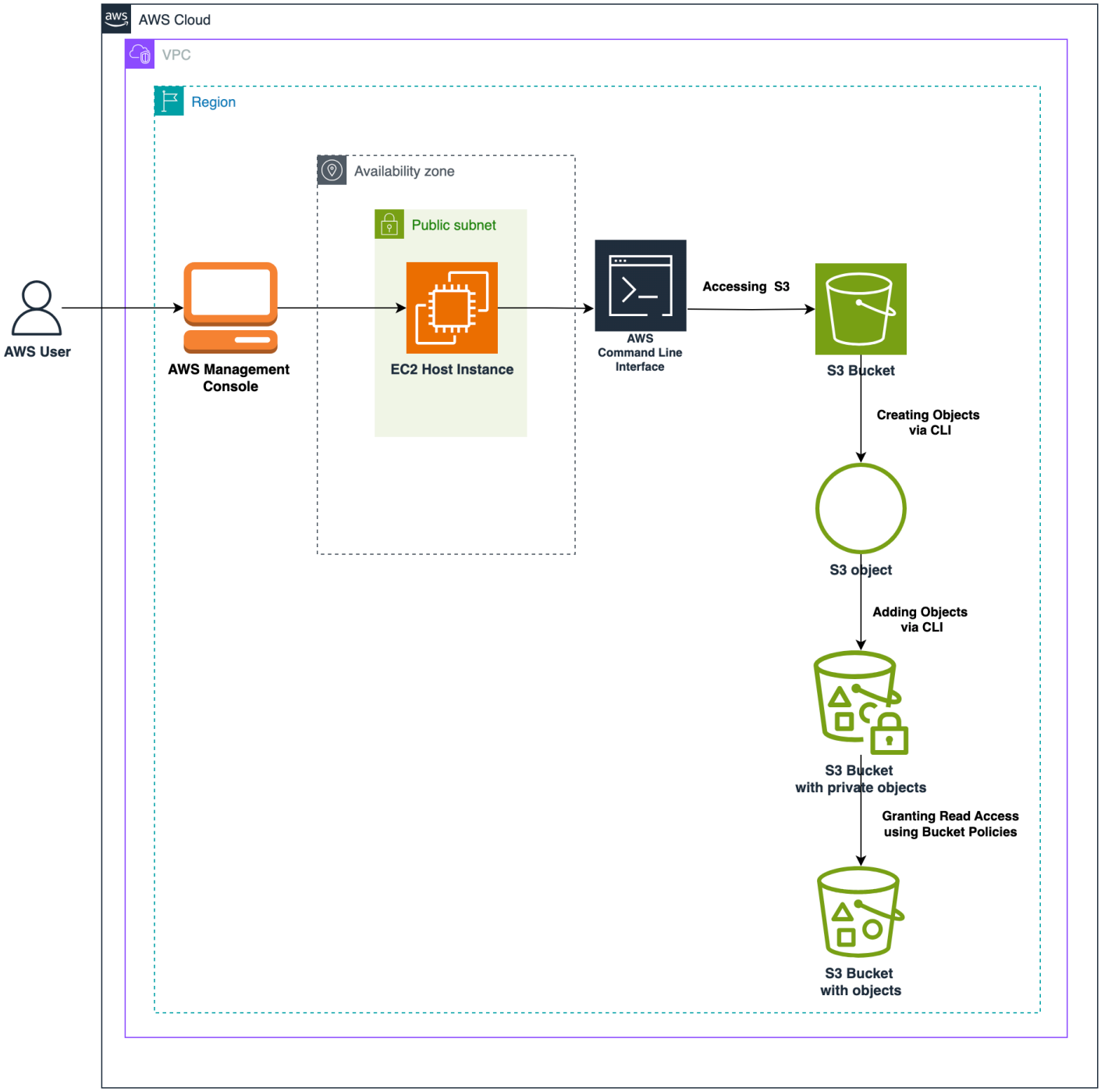
In this project, I explored the capabilities of **Amazon Simple Storage Service (Amazon S3)** by creating an **S3 bucket**, uploading objects, and configuring permissions to make those objects publicly accessible. Additionally, I used the **AWS Command Line Interface (AWS CLI)** to list bucket contents, demonstrating fundamental S3 operations.

This hands-on practice enhanced my understanding of **cloud storage, object permissions, and CLI-based AWS interactions**.

Technologies used

- **Amazon S3** – Scalable object storage
- **AWS CLI** – Command-line tool to interact with AWS services
- **Amazon EC2** – Instance for CLI execution
- **IAM (Identity and Access Management)** – Secure access management
- **app.diagram.net:** Architecture design visualization tool.

Architecture Overview



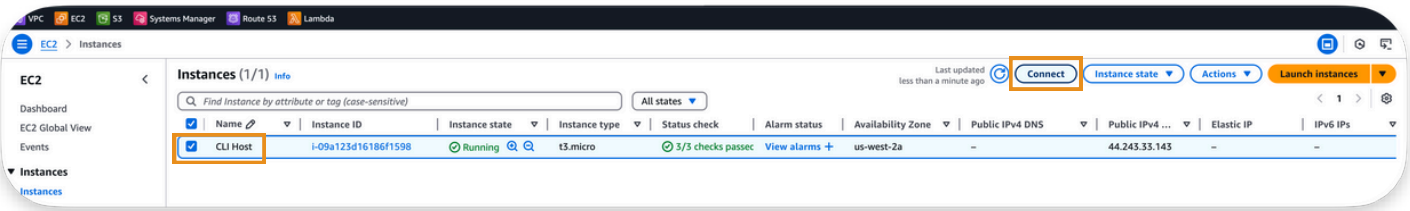
Step-by-Step Implementation

1 . Connecting to the CLI Host instance:

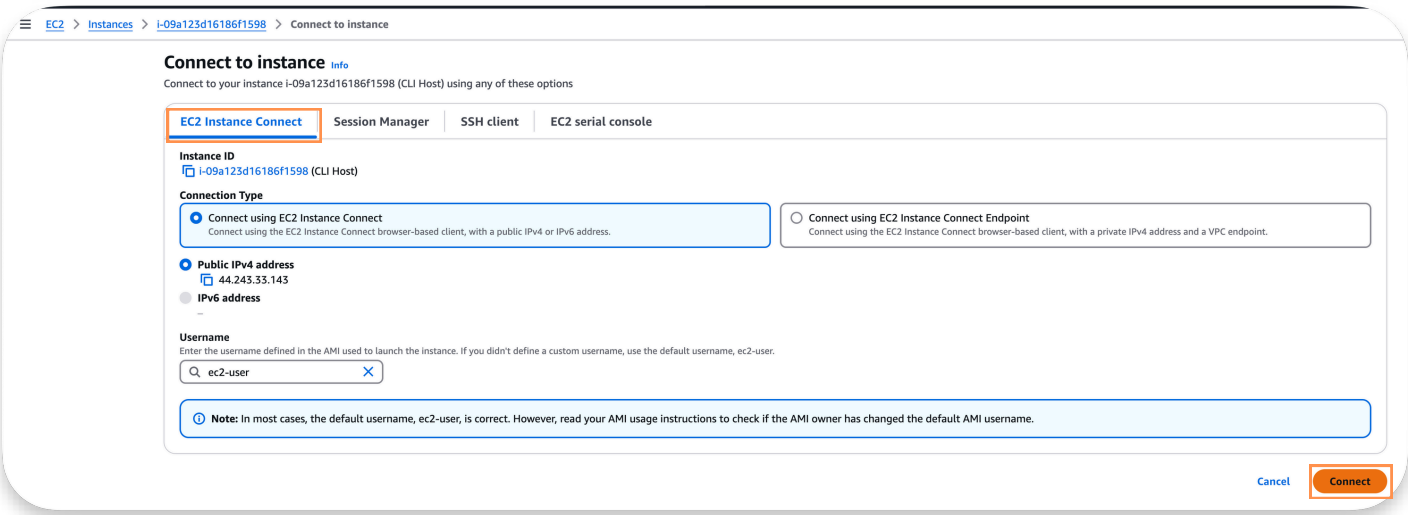
To interact with **AWS services** using the **AWS CLI**, I connected to the **EC2 CLI Host instance**, which had already been created.

Steps:

- Open the **AWS Management Console.**
- In the search bar enter and select **EC2.**
- In the navigation pane (left)n choose "**Instances**".
- Select the already created "**CLI Host instance**' → Click **Connect.**



- → Go to **EC2 Instance Connect** → Click **Connect.**



2 . Configuring the AWS CLI:

After connecting to the EC2 instance, I configured **AWS CLI** with credentials to authenticate and interact with AWS services.

```
  ,     #_
  ~\_  ####_        Amazon Linux 2
 ~~  \_#####\
 ~~     \###|       AL2 End of Life is 2026-06-30.
 ~~       \#/ ___
  ~~       V~' '->
   ~~~         /    A newer version of Amazon Linux is available!
     ~~._.   _/
        _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
      _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-              ~]$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-west-2
Default output format [None]: json
```

3 . Creating an Amazon S3 bucket:

I created an **Amazon S3 bucket** using AWS CLI to store objects.

```
[ec2-user@ip-              ~]$ aws s3 mb s3://my-s3-bucket-juna
make_bucket: my-s3-bucket-juna
```

4 . Uploading an object to the S3 bucket:

I uploaded a test file (*example.txt*) to the bucket.

Steps:

- Create an test file called example.txt and confirm the file is present in the directory listing

```
[ec2-user@ip-              ~]$ echo "This is a text file for S3 upload." > example.txt
[ec2-user@ip-              ~]$ ls -l
total 4
-rw-rw-r-- 1 ec2-user ec2-user 35 Mar 20 14:46 example.txt
```

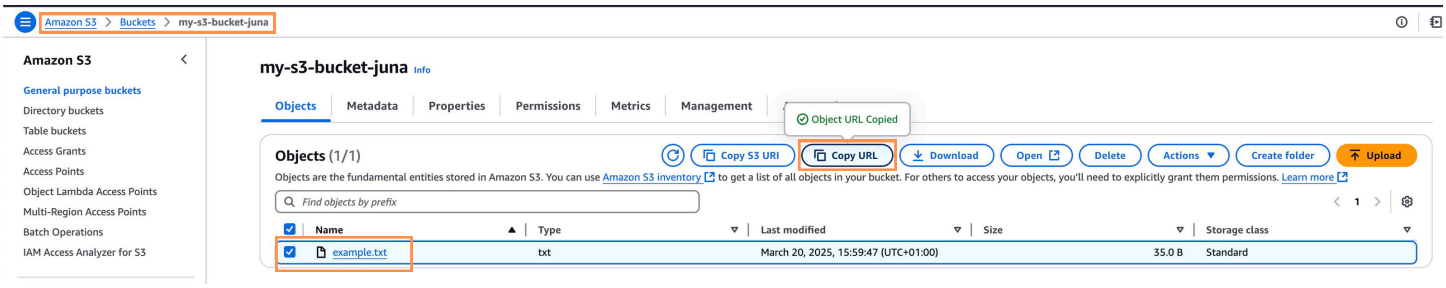
- Upload the *example.txt* file to the bucket.

```
[ec2-user@ip-              ~]$ aws s3 cp example.txt s3://my-s3-bucket-juna/
upload: ./example.txt to s3://my-s3-bucket-juna/example.txt
```

5 . Obtaining the uploaded object URL via AWS Management Console:

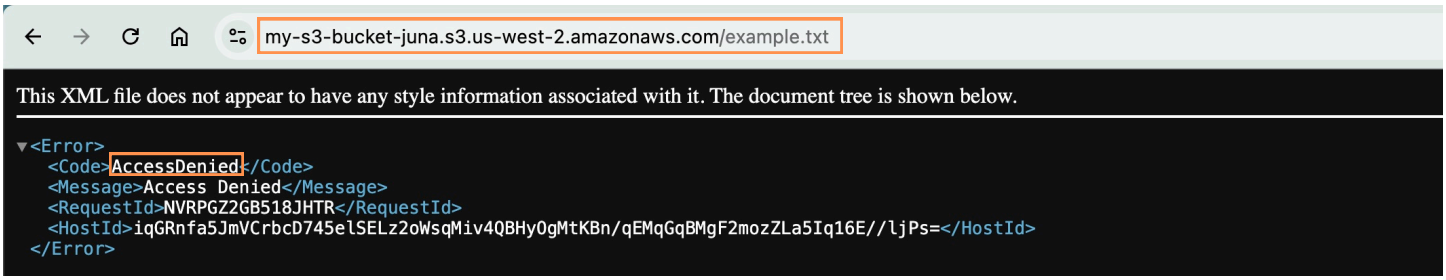
Steps:

- Go to **AWS Management Console** → Search the recently create **S3** bucket and select it → Select the uploaded object *example.txt* and → **Copy URL.**



6 . Accessing the object via Web Browser:

Initially, the uploaded object was **private**, so direct browser access was **not allowed**.



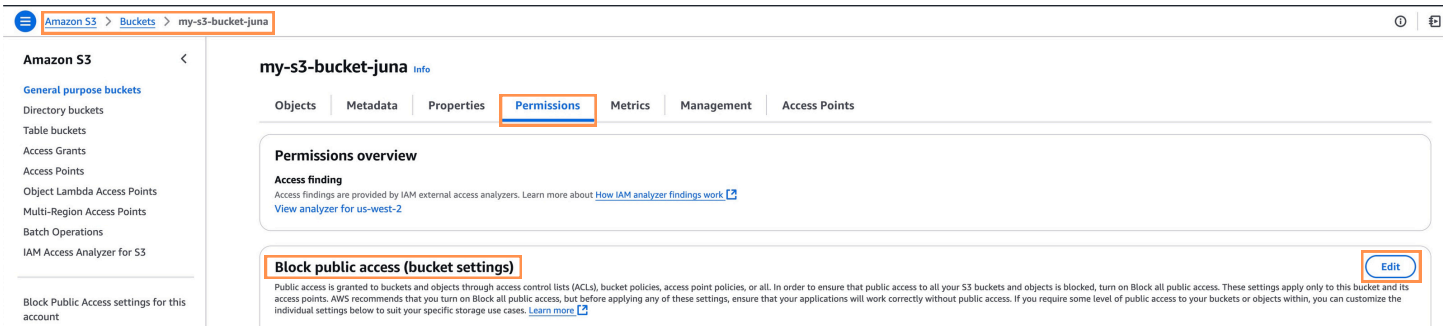
7 . Making the object publicly accesible:

By default, Amazon S3 **blocks public access** to all objects. To make only *example.txt* publicly accessible while keeping all other objects private, I applied a **bucket policy** that allows public read access to this specific object.

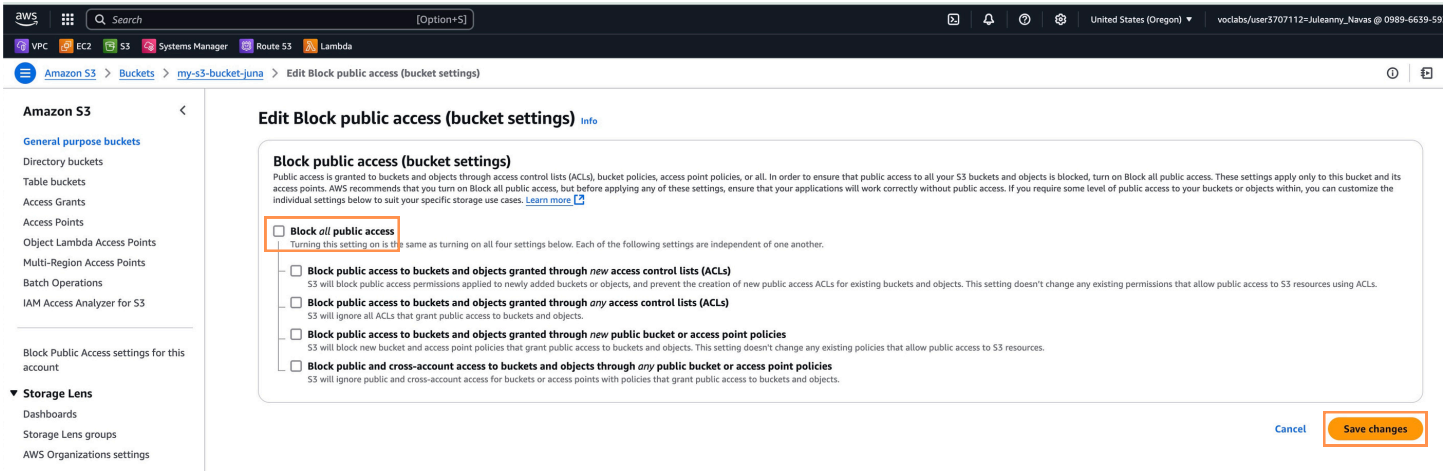
For more details, refer to the AWS Knowledge Center article:
🌐 Grant public read access to objects in Amazon S3 bucket

Steps:

- **Step 1: <u>Disable "Block Public Access"</u>**
  - Open the **AWS S3 Console**.
  - Select the bucket **my-s3-bucket-juna**.
  - Click on the **"Permissions"** tab.
  - Under **"Block public access (bucket settings)"**, click **Edit**.



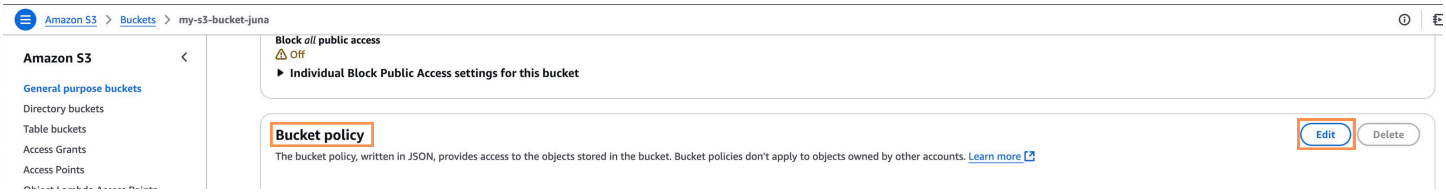
  - **Uncheck ✅ "Block all public access"**
  - Click **Save changes**.



- **Step 2: <u>Apply a Public Read Policy for *example.txt*</u>**

I used a **bucket policy** to allow public read access **only for example.txt**, keeping all other files private.

  - Scroll to the **"Bucket policy"** section and click **Edit**.
  - Paste the **bucket policy**, which allows public access ONLY to *example.txt*, keeping all other objects private. **Save changes.**



Amazon S3 > Buckets > my-s3-bucket-juna > Edit bucket policy

Amazon S3

- General purpose buckets
- Directory buckets
- Table buckets
- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight 11

AWS Marketplace for S3

Bucket policy

Policy examples | Policy generator

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Bucket ARN
arn:aws:s3:::my-s3-bucket-juna

Policy

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Sid": "PublicReadSpecificObject",
6        "Effect": "Allow",
7        "Principal": "*",
8        "Action": "s3:GetObject",
9        "Resource": "arn:aws:s3:::my-s3-bucket-juna/example.txt"
10     }
11   ]
12 }
```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

+ Add new statement

+ Add new statement

JSON Ln 1, Col 0

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

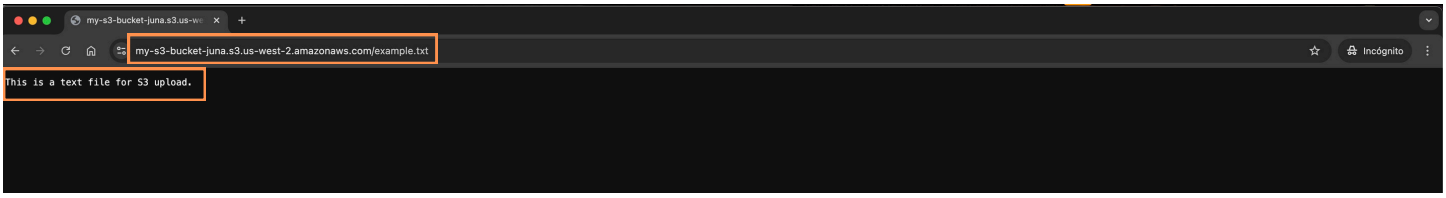
Preview external access

Cancel | Save changes

8 . Testing the policy:

I tested public access for *example.txt* object using its **direct URL.**

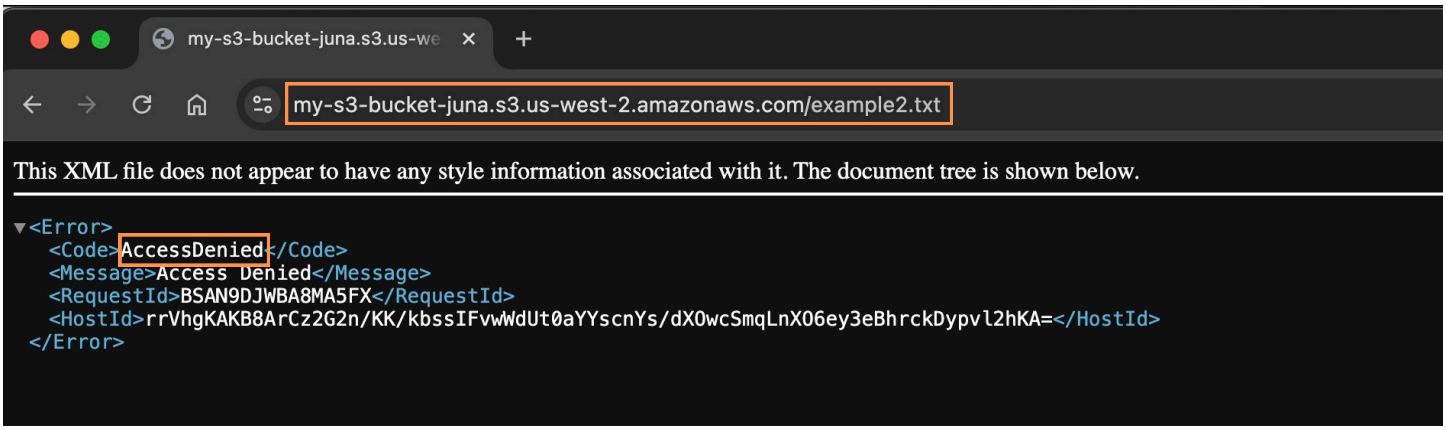
- **Test URL for *example.txt*** (Expected: ✅ Accessible)



- **Test URL for *example2.txt*** (Expected: ❌ Forbidden)
  - Creating example2.txt file and uploading to S3 bucket:

```
[ec2-user@ip-            ~]$ echo "This is a text file for S3 upload->2." > example2.txt
[ec2-user@ip-            ~]$ aws s3 cp example2.txt s3://my-s3-bucket-juna/
upload: ./example2.txt to s3://my-s3-bucket-juna/example2.txt
```

  - The request was blocked with a **Forbidden error**, confirming that only *example.txt* is publicly accessible, while *example2.txt* remains private.



9 . Listing the bucket contents:

To verify the uploaded objects, I listed the bucket contents.

```
[ec2-user@ip-            ~]$ aws s3 ls s3://my-s3-bucket-juna/
2025-03-20 14:59:47         35 example.txt
2025-03-20 15:37:09         38 example2.txt
```

Conclusions & Lessons Learned

- **Amazon S3 Provides Secure and Scalable Storage:**
  - Easy to create and manage object storage at scale.
  - Supports fine-grained access control through **bucket policies** and **IAM roles**.

- **AWS CLI is Powerful for Cloud Interactions:**
  - Automates storage operations without using the AWS Console.
  - Allows efficient bucket management, file uploads, and permission modifications.

- **Object Permissions Must Be Configured for Public Access:**
  - By default, objects are **private**, and permissions must be explicitly granted.
  - **Bucket policies** are the **recommended** method for controlling public access to specific objects, as they provide centralized, scalable, and auditable access control.

- **Pre-Signed URLs Provide Temporary Access:**

```
[ec2-user@ip-                    ~]$ aws s3 presign s3://my-s3-bucket-juna/example.txt
```

  - Ideal for controlled sharing without making objects public.
  - Ensures access expires after a set period, enhancing security.

Final Thoughts

This project provided hands-on experience with **Amazon S3, IAM permissions, and AWS CLI**. Successfully configuring and testing **public access, file uploads, and permission management** enhanced my understanding of **AWS storage security best practices**.

Additionally, I learned that **bucket policies** are the **best approach** for granting public access to specific objects, as they offer **better security, scalability, and management** compared to object ACLs.