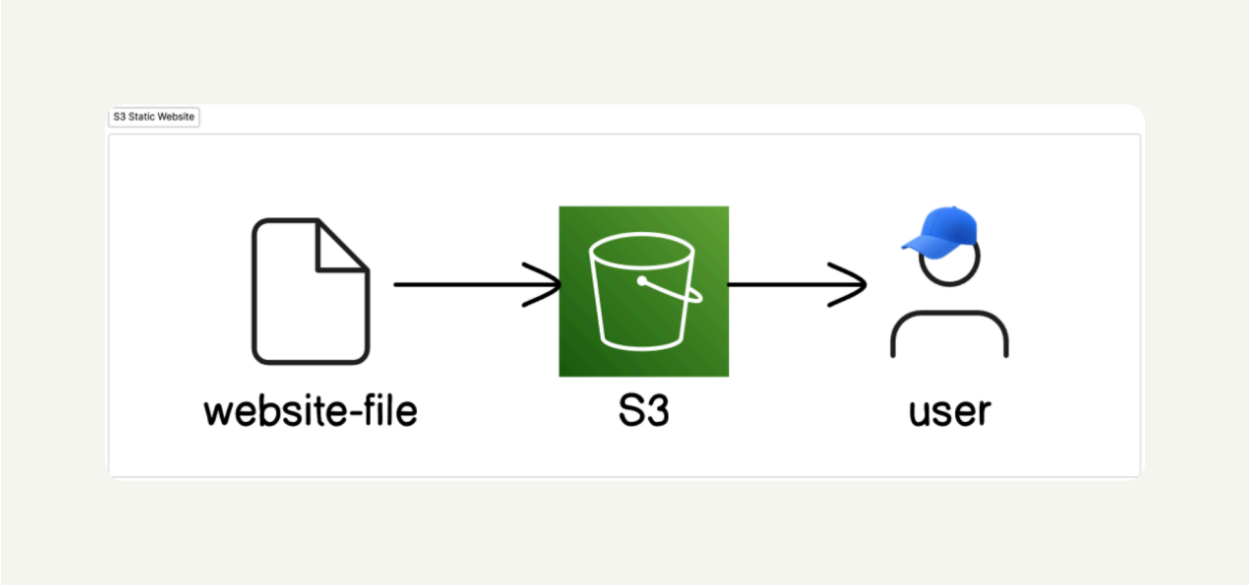
Static Website Hosted on Amazon S3 with IAM-Based Access Control

Introduction

My name is Mohammed Amir. In this project, I hosted a static website on Amazon S3 while enforcing secure access control using IAM policies. The goal was to deploy a publicly accessible website while granting a dedicated IAM user limited permissions to upload objects without the ability to delete them — following the Principle of Least Privilege.



Topics Covered :-

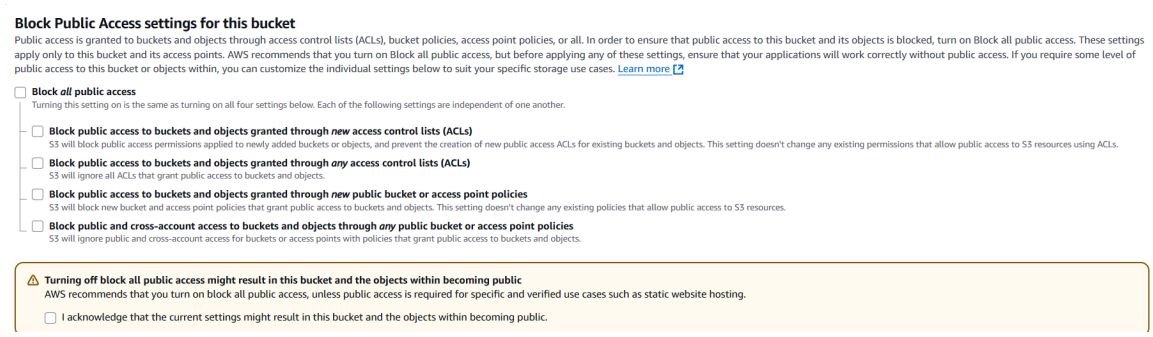
1. Amazon S3 (Simple Storage Service) :- **S3 is an object storage service that allows storing and retrieving any amount of data from anywhere on the web.**

- A Bucket is a container for storing objects (files).
- Each bucket has unique access permissions and configurations like Bucket , Versioning, Lifecycle rules.
- **Bucket Versioning** :- Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures.

  Example of Versioning in S3 Bucket:
  You upload a file report.docx to an S3 bucket with versioning enabled. If you upload report.docx again (same name), S3 saves both versions — you can access or restore any previous version anytime.

- Bucket Lifecycle :- Lifecycle policies automatically transition objects to cheaper storage classes or delete them after a set time, helping optimize cost and manage data retention. S3 offers multiple storage classes for different use cases, from frequent access to archival. Below are the main storage classes :-
  1. S3 Standard — For frequently accessed data like active content, websites, and applications.
  2. S3 Standard-IA (Infrequent Access) — For infrequently accessed data that still needs quick retrieval, like backups and disaster recovery.
  3. S3 One Zone-IA — For infrequently accessed data that can be stored in a single AZ, like secondary backups.
  4. S3 Intelligent-Tiering — For data with unknown or changing access patterns, automatically moves objects between s3 storage as per the access.

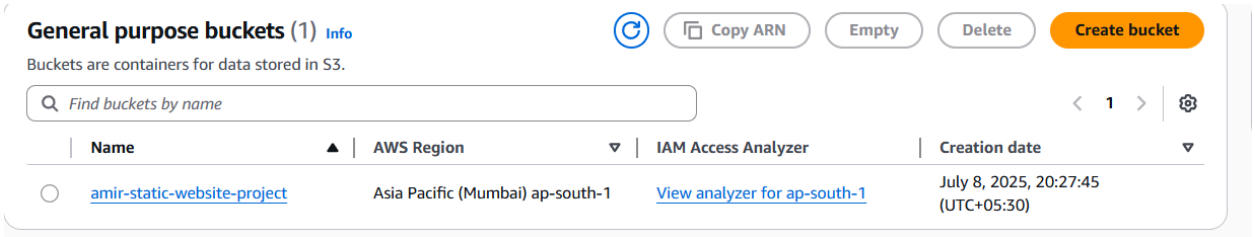
5. S3 Glacier — For archival data that requires occasional access, like compliance records.
6. S3 Glacier Deep Archive — For long-term archival of rarely accessed data, like regulatory archives.



Note :- While creating an S3 bucket if you want the objects to be publicly accessed then you must disable the block all public access.

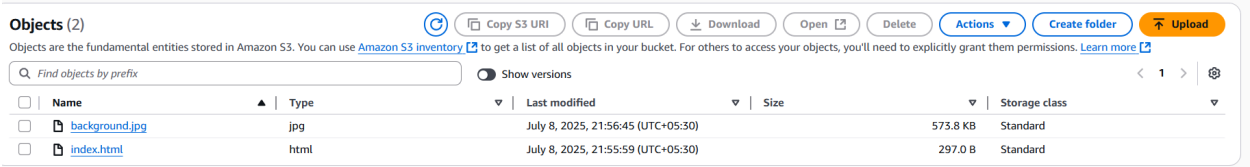
Bucket Name Rules : -

- Globally Unique
- Bucket names must be between 3 (min) and 63 (max) characters long.
- Bucket names can consist only of lowercase letters, numbers, periods (.), and hyphens (-).
- Bucket names must begin and end with a letter or number.



2. Objects in S3 :- **An Object in S3 is any file (like HTML, image, video) stored inside a bucket.**

- Each object has metadata and a unique key (filename).
- Objects can be public or private based on bucket policy or ACL.
- You upload all the required files for your website to work in the s3 bucket.

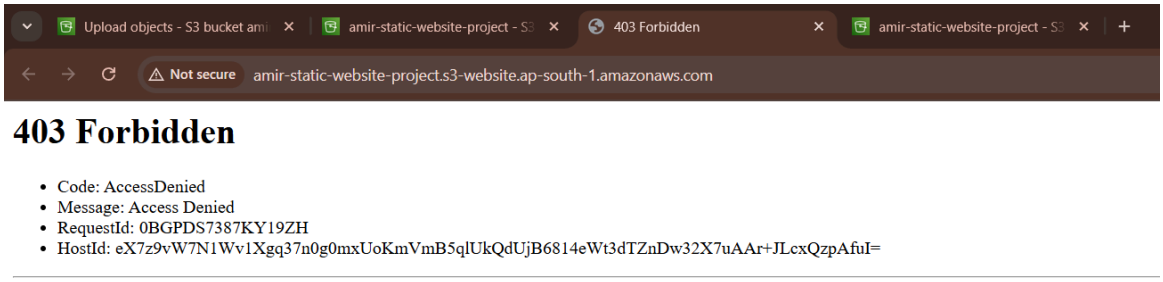


Metadata — Data about the object, such as its content type , size, creation date, encryption details, or custom tags.

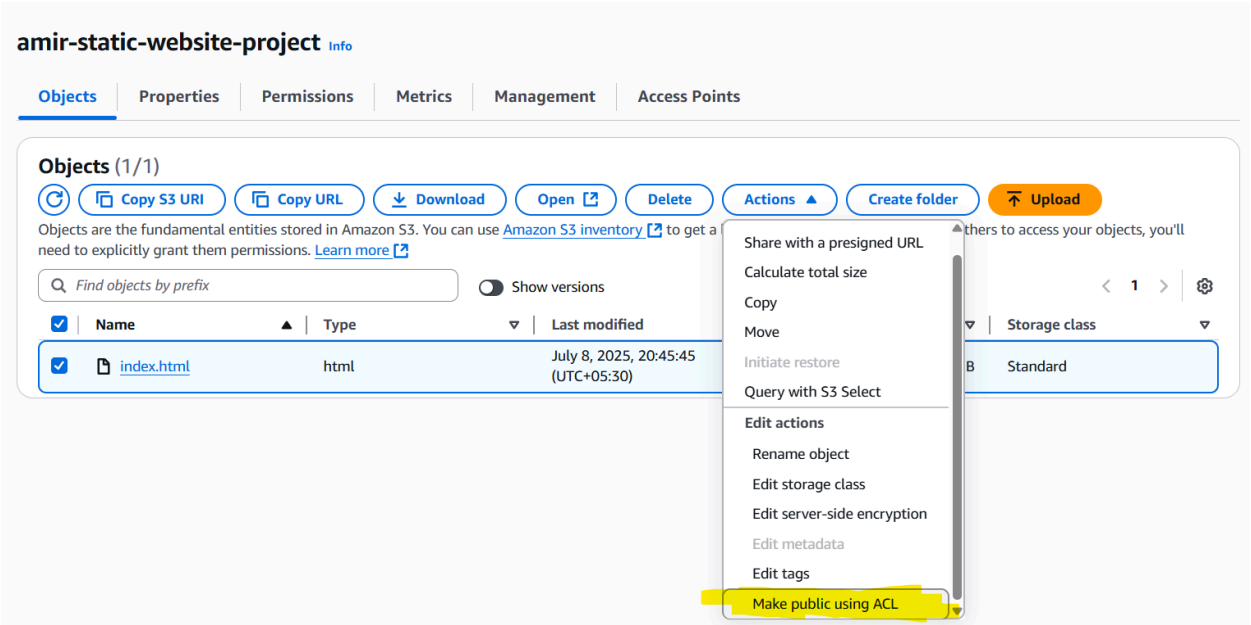
Unique Key — The name of the object (file name + path) that uniquely identifies it within the bucket.

3. 403 Access Denied Error

Common error when trying to access an object over the internet without **making the objects public using ACL.**

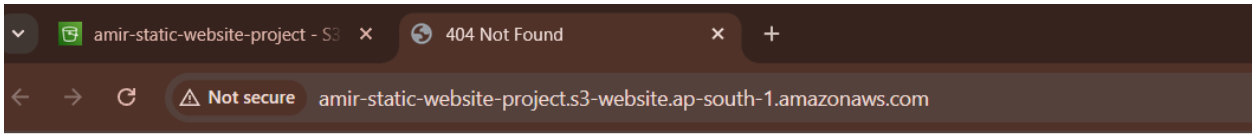


ACL (Access Control List) in Amazon S3 is a permissions mechanism that defines who can access an object or bucket.



4. 404 Not Found Error

Occurs when the static website hosting is not enabled or index document is not set.

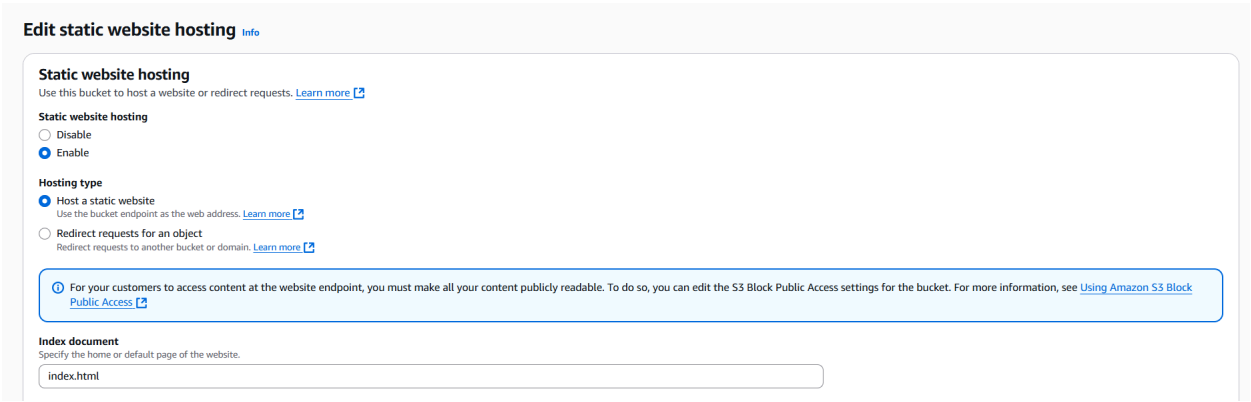


404 Not Found

- Code: NoSuchWebsiteConfiguration
- Message: The specified bucket does not have a website configuration
- BucketName: amir-static-website-project
- RequestId: 607STTD5TF0Y0DC0
- HostId: HmMADlpcrNFC4UVgezET9+FXaPcwrMISisli5u28JoGeDZwhH9WR4zxWaQW0YyJEXOeqAZQro4E=

Static Website — A website with fixed content that doesn't change unless edited manually, built using HTML, CSS, and JavaScript. Dynamic Website — A website that displays content dynamically based on user interaction or server-side processing, often using databases and backend scripting.

- **Solved by enabling static website hosting and specifying index.html.(landing page)**



5. IAM (Identity and Access Management) :- **AWS service to securely manage access to AWS resources by controlling who can do what.**

1 Created a Custom IAM Policy :- A JSON document that defines permissions (allow/deny) for users, groups, or roles to access AWS resources..

Policy Name: S3-StaticWebsite-UploadPolicy :-

Permissions defined in this policy Info

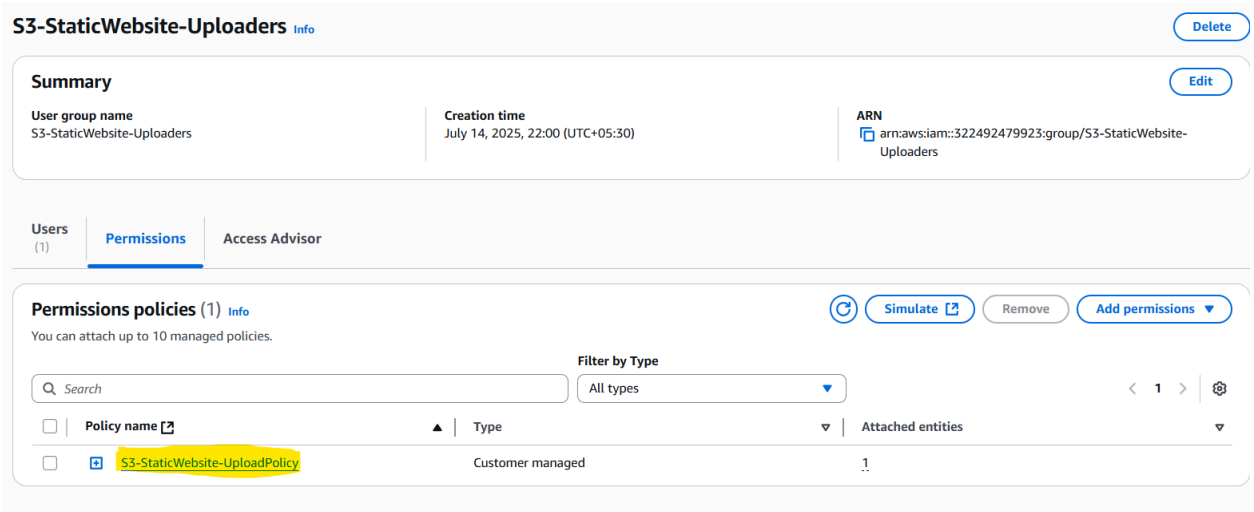
Permissions defined in this policy document specify which actions are allowed or denied.

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Sid": "UploadObject",
6              "Effect": "Allow",
7              "Action": [
8                  "s3:PutObject",
9                  "s3:GetObject"
10             ],
11             "Resource": "arn:aws:s3:::amir-static-website-project/*"
12         },
13         {
14             "Sid": "ListBucket",
15             "Effect": "Allow",
16             "Action": [
17                 "s3:ListAllMyBuckets",
18                 "s3:ListBucket"
19             ],
20             "Resource": "*"
21         }
22     ]
23 }
```

2 Created an IAM Group :- A collection of IAM users; permissions assigned to the group apply to all its members.

Group Name: S3-StaticWebsite-Uploaders.

Attached the custom policy S3-StaticWebsite-UploadPolicy to the group.



✅ Why This Approach?

Easier Management: Future users can be added to the group with inherited permissions.

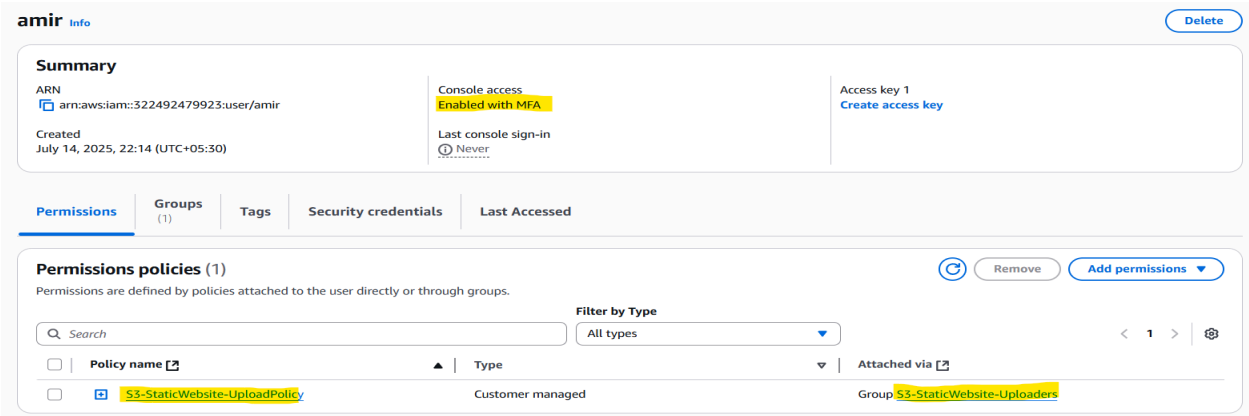
Policy Consistency: No need to attach policies individually.

Scalable Access Control: Aligns with IAM best practices.

3 Created IAM User :- An identity with credentials used by a person or application to access AWS services.

User Name: amir with **MFA enabled.**

Added to group: S3-StaticWebsite-Uploaders



✅ What is MFA (Multi-Factor Authentication) ?

MFA = An extra layer of security beyond just username & password.

It requires the user to provide two or more authentication factors to sign in :-

- You sign in with IAM username + password
- ✅ Then, you enter a 6-digit OTP code from an Authenticator app like Google Authenticator or AWS MFA app.

Policy validation using AWS IAM Policy Simulator

To ensure that the IAM permissions granted to the user were correct and functional, we used the AWS IAM Policy Simulator to simulate and test various S3 actions based on the attached policy.

The following actions were tested:

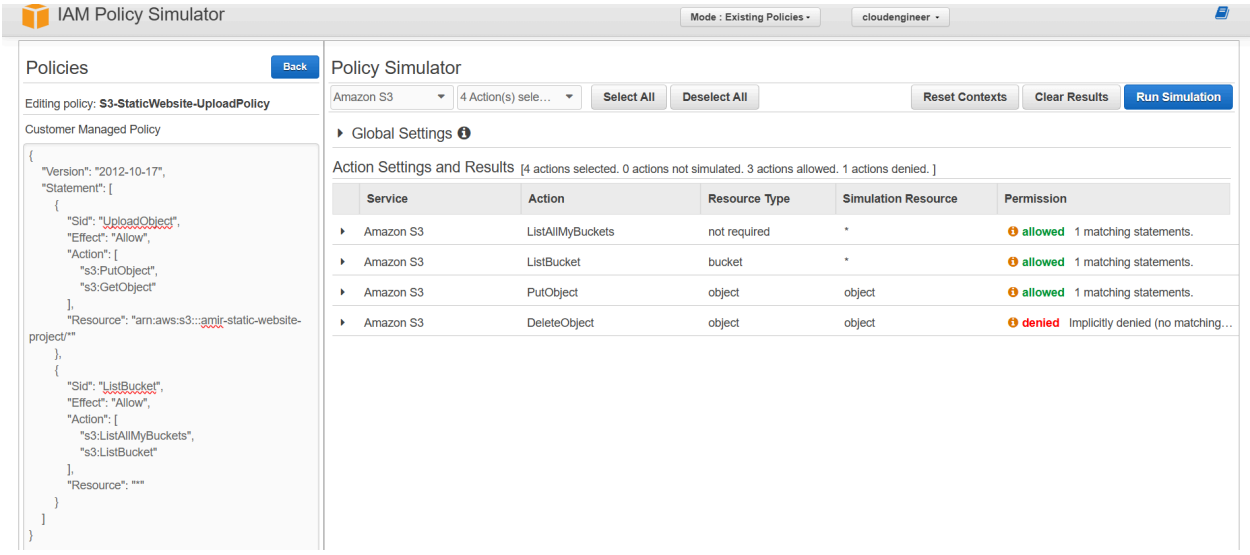
✅ s3:ListAllMyBuckets — Allowed (User can list all buckets in the account)

✅ s3:ListBucket — Allowed (User can list objects inside the allowed bucket)

✅ s3:PutObject — Allowed (User can upload objects to the bucket)

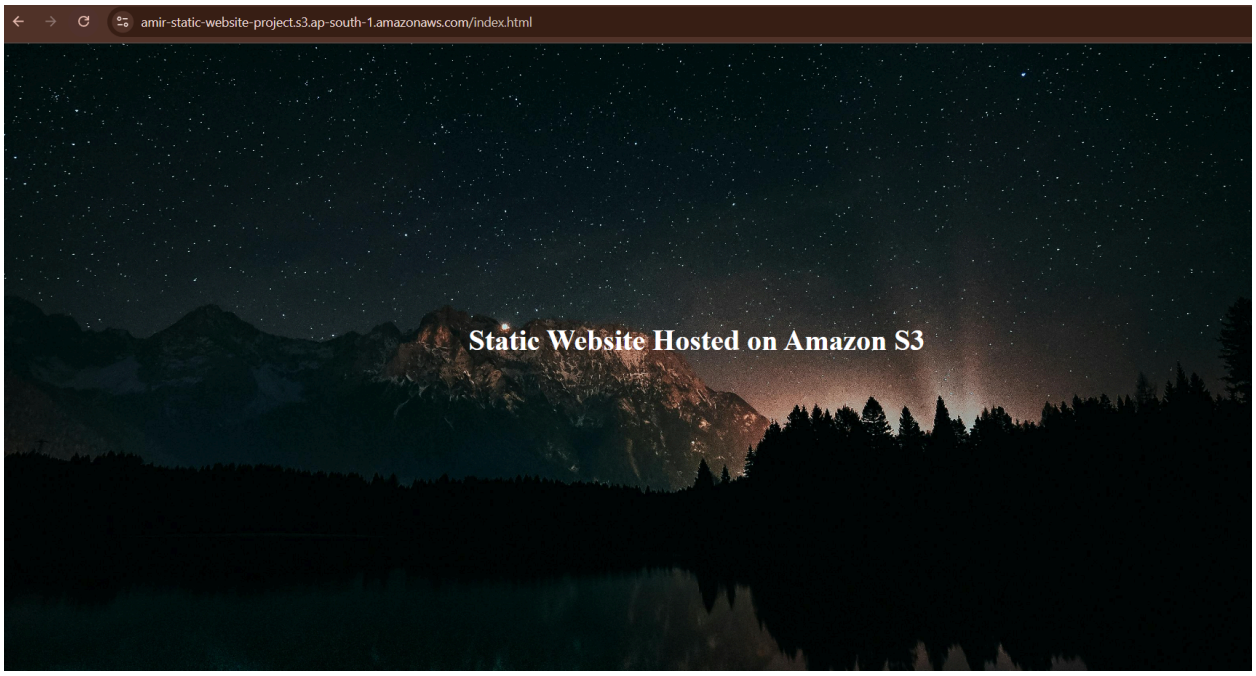
❌ s3:DeleteObject — Denied (As expected, since DeleteObject was not permitted in the policy , **By default, all actions are implicitly denied in AWS unless explicitly allowed**)

The simulator results confirmed that the IAM policy correctly enforces the intended permissions and restrictions.

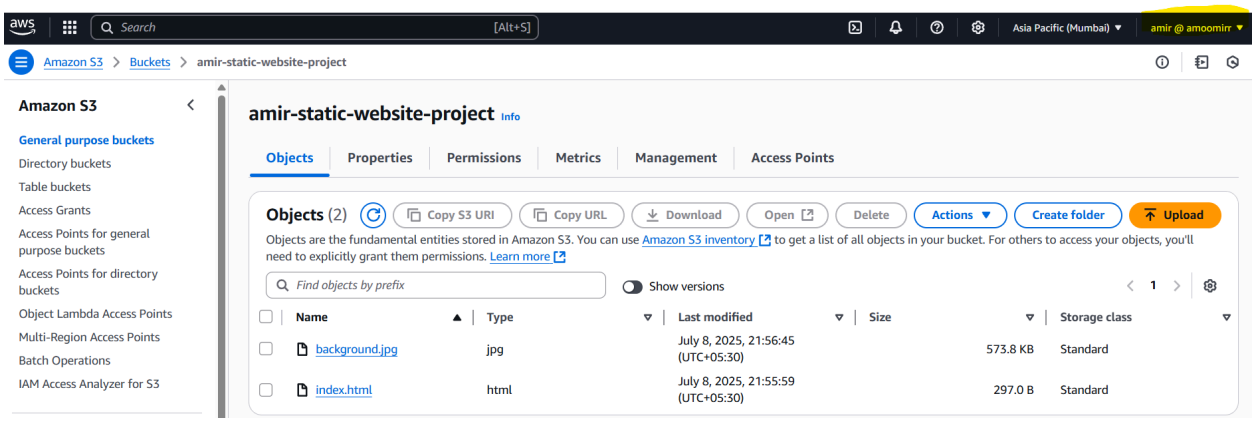


Website URL :-

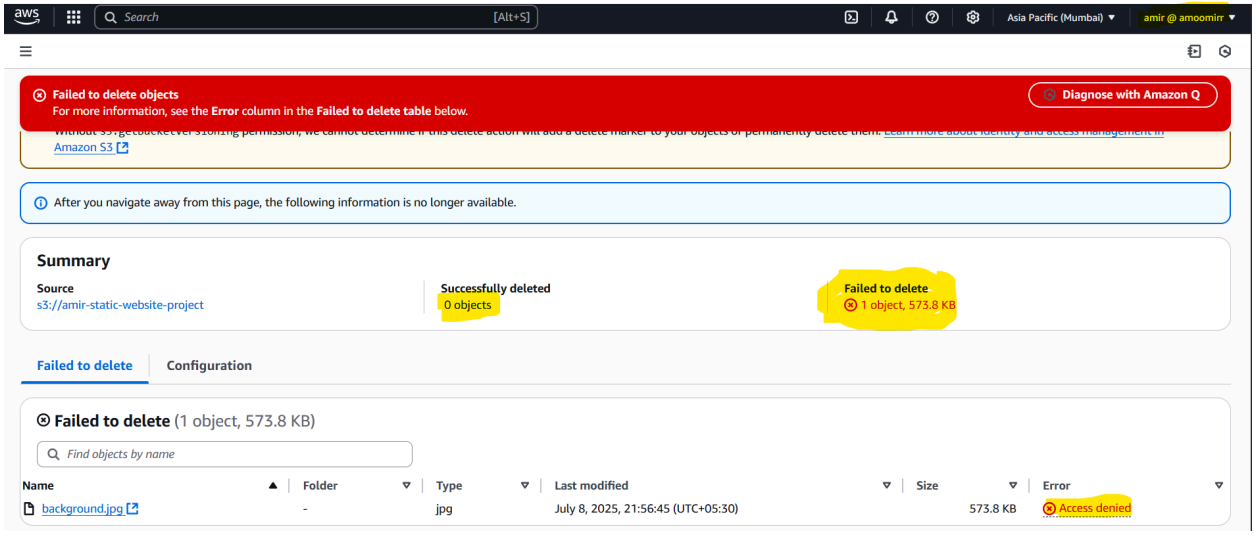
https://amir-static-website-project.s3.ap-south-1.amazonaws.com/index.html



User can list the bucket and objects :-



User cannot delete any objects as implicitly denied :-



✅ Conclusion and Project Outcome

In this project, we successfully hosted a static website on Amazon S3 with controlled access using AWS Identity and Access Management (IAM).
We ensured secure and controlled operations by following the principle of least privilege for IAM users.

Through this project, we achieved the following objectives:

- ✅ Created an Amazon S3 bucket and configured it for static website hosting.
- ✅ Uploaded web content and resolved common access errors (403 Forbidden, 404 Not Found).
- ✅ Created an IAM user and assigned permissions using a custom IAM policy
- ✅ Controlled access by allowing only specific actions — listing, uploading, and reading objects.
- ✅ Validated permissions using the AWS IAM Policy Simulator.
- ✅ Ensured sensitive actions like object deletion remained denied.