Report: Amazon S3 Bucket Configuration

Name: Chaithanya M

Date: July 14, 2025

Topic: Amazon S3 Bucket Configuration

1. Introduction

Amazon Simple Storage Service (Amazon S3) is a highly scalable, durable, and secure object storage service offered by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data from anywhere on the internet at any time. S3 provides a simple web interface and API for developers and system administrators to manage data, making it suitable for a wide range of use cases including backups, data lakes, static website hosting, and application data storage. Each object is stored in a bucket and can be up to 5 TB in size, making it ideal for storing large volumes of structured or unstructured data.

S3 is engineered for 99.999999999% (11 9's) durability and offers features such as access control, versioning, server-side encryption, event notifications, lifecycle policies, and integration with other AWS services.

2. Objective

- Review the setup and configuration of an S3 bucket.
- Understand its usage for data storage, static website hosting, or integration with other AWS services.
- Assess access permissions, versioning, and encryption settings.
- Identify potential misconfigurations or security vulnerabilities.
- Provide recommendations to ensure data integrity, availability, and secure access.

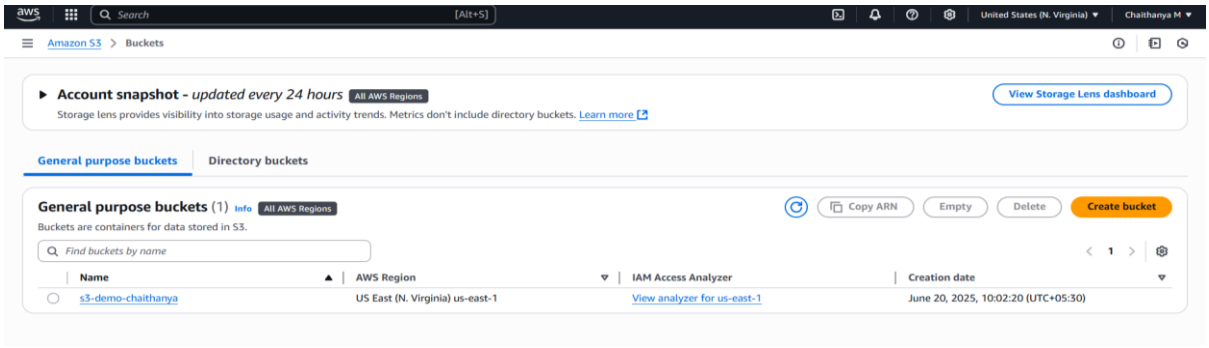
3. Prerequisites

- Active AWS account with billing enabled.
- IAM user or role with necessary S3 permissions (create, list, read, write).
- AWS CLI installed and configured, or access to AWS Management Console.
- Valid bucket name following S3 naming conventions.

4. Implementation

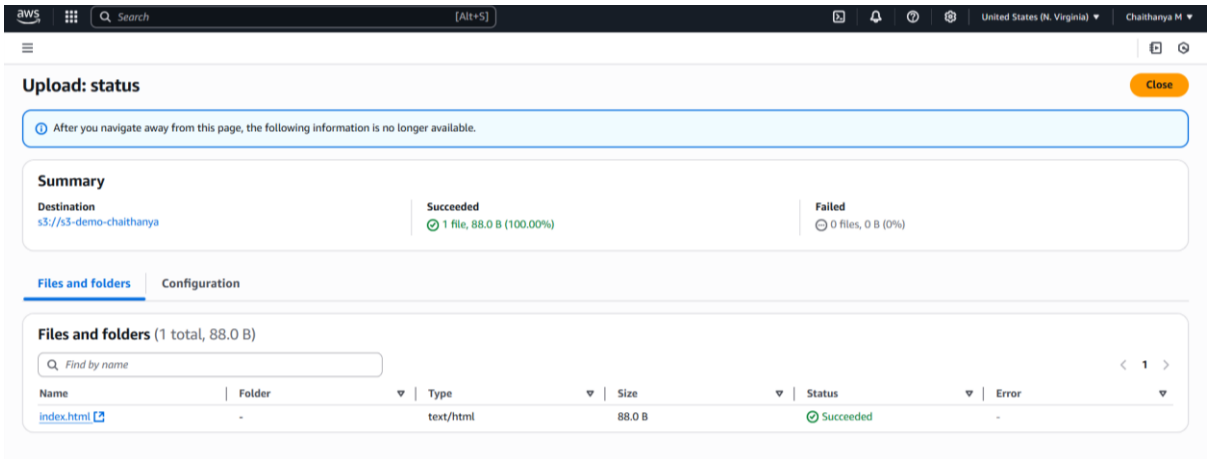
Step 1: Create a New S3 Bucket

- Click the “Create bucket” button.
- Fill in the required details:
    - Bucket name**:** s3-demo-chaithanya
    - AWS Region**:** us-east-1



Step 2: Upload Objects to the Bucket

- Select your newly created bucket.
- Click “Upload” > Add index.html file**.**

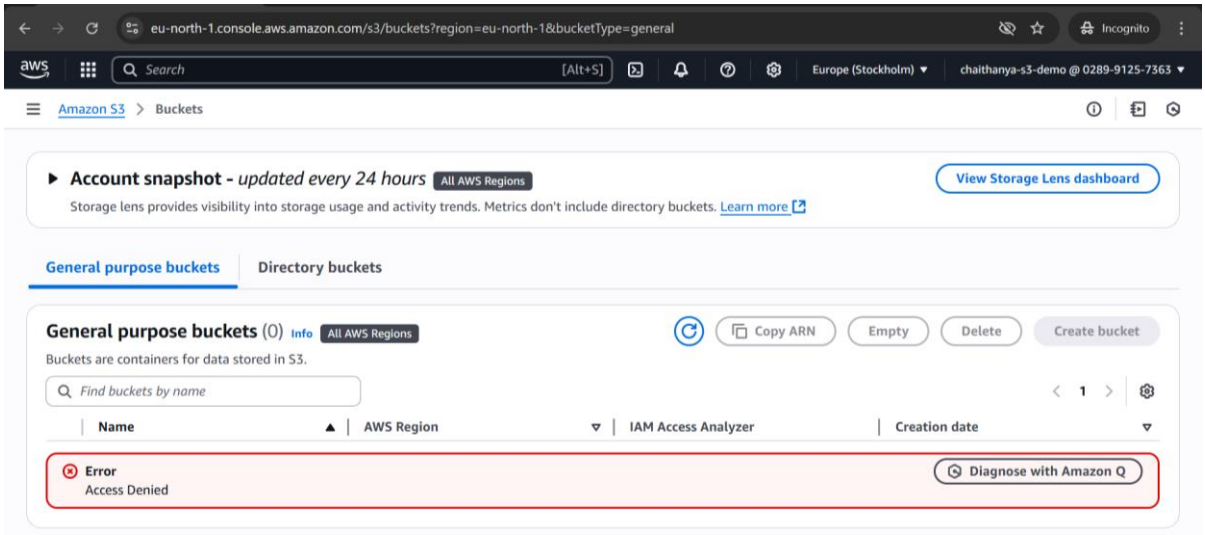


Step 3: Add a New IAM User

- User Name**:** chaithanya-s3-demo
- Access Type**:**
    - Check Programmatic access (for CLI/SDK use).
    - Check AWS Management Console access if you want GUI access.
    - Set a password for console access.

Step 4: check Permissions for the IAM User

This confirms that access is correctly denied by default.

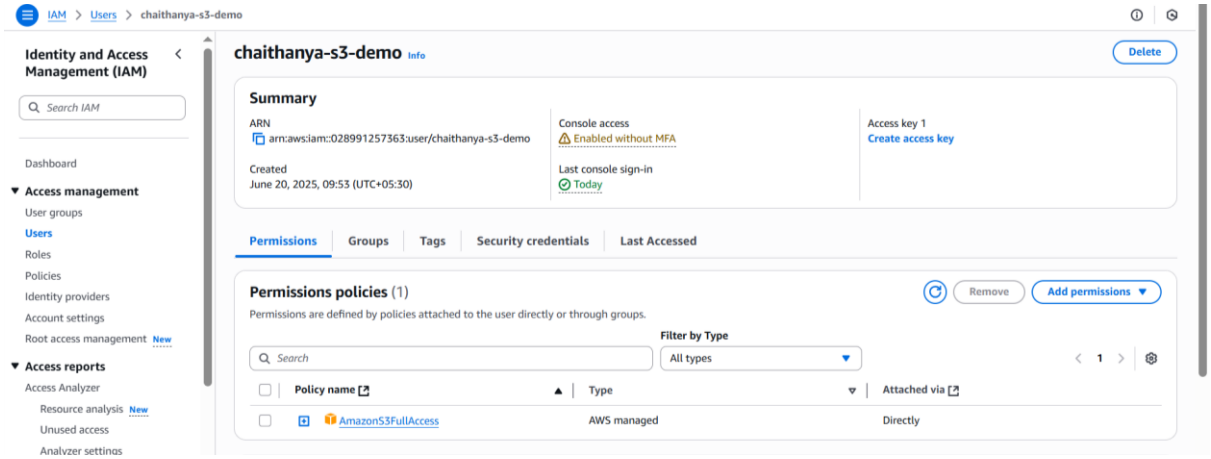


Step 5: Grant S3 Bucket Permissions to the IAM User

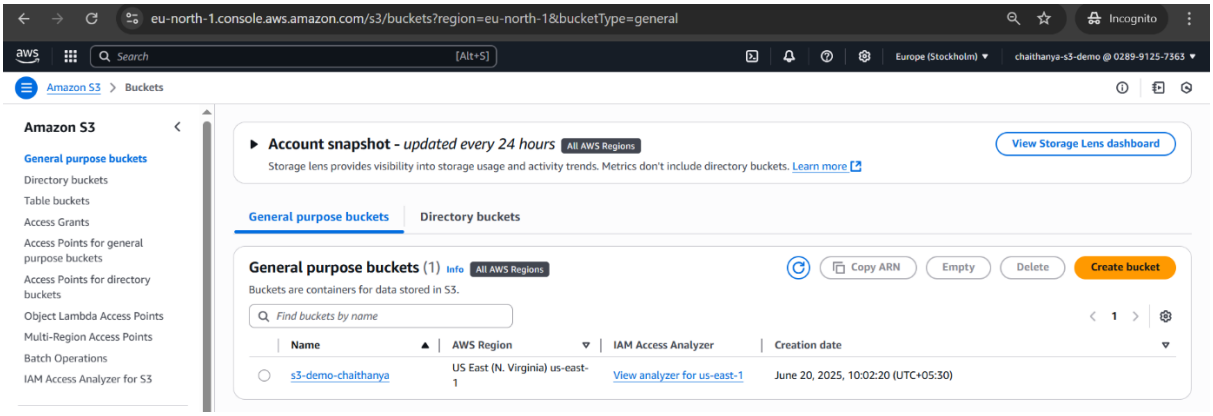
Attach Policy to Allow S3 Bucket Access

Go to IAM Console → Users → chaithanya-s3-demo → Add permissions and:

- Choose Attach policies directly, Select AmazonS3FullAccess.



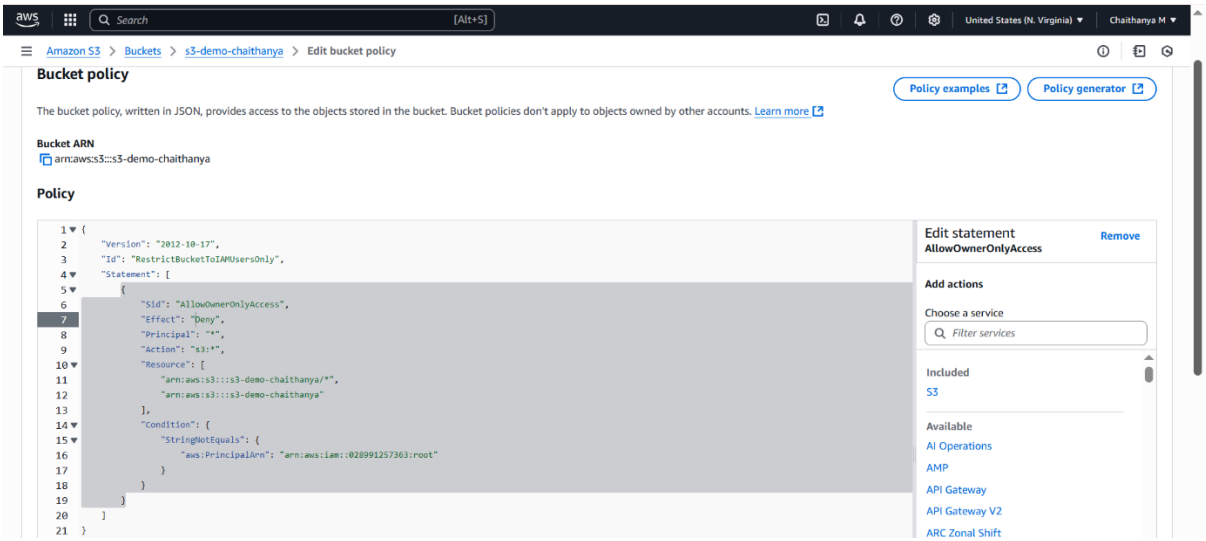
This confirms access is granted after permission is applied.

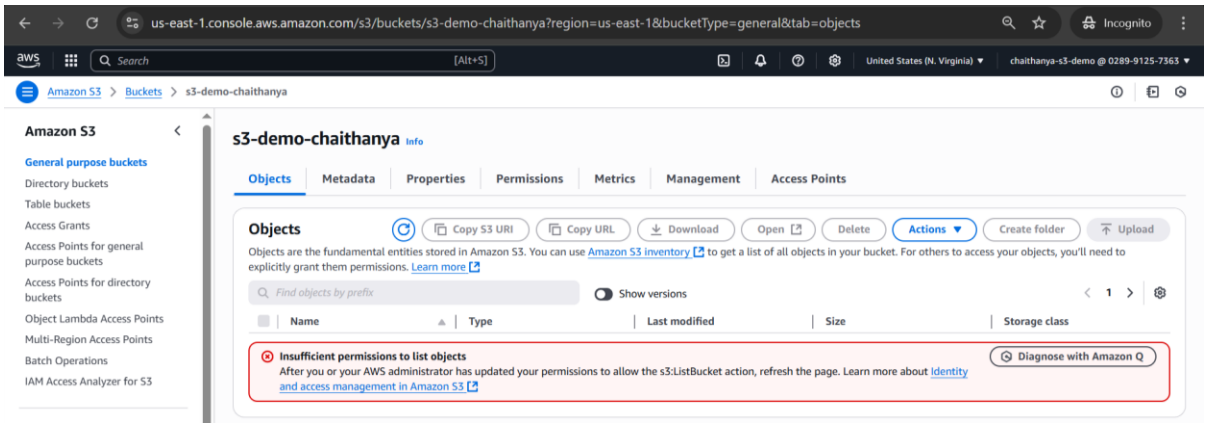


Step 6: Configure Bucket Policy to Restrict Access

As part of securing the s3-demo-chaithanya bucket, a bucket policy was added to explicitly deny access to all users except the root user of the AWS account. This was done by applying a deny rule in the bucket policy that blocks all S3 actions unless the request is made by the account root user. This step ensures that no other IAM users, roles, or external identities can access the bucket until proper permissions are deliberately granted.

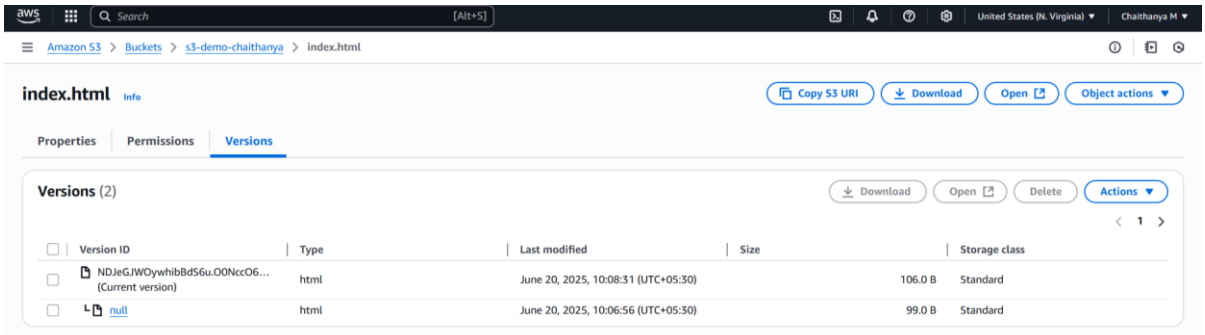

```
{
  "Version": "2012-10-17",
  "Id": "RestrictBucketToIAMUsersOnly",
  "Statement": [
    {
      "Sid": "AllowOwnerOnlyAccess",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::s3-demo-chaithanya/*",
        "arn:aws:s3:::s3-demo-chaithanya"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::028991257363:root"
        }
      }
    }
  ]
}
```





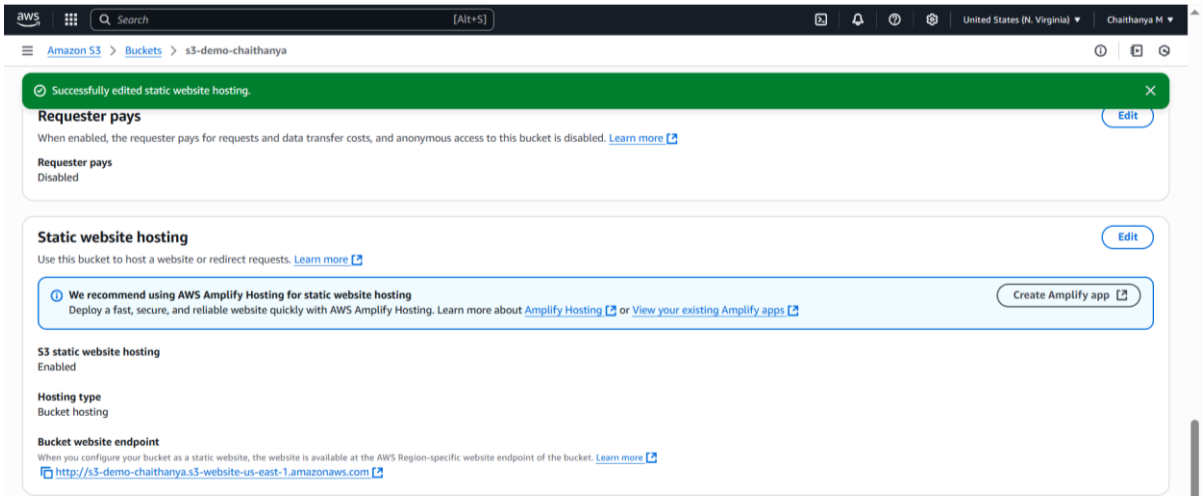
Step 7: Enable Bucket Versioning

Versioning was enabled on the s3-demo-chaithanya bucket to retain multiple versions of objects.This helps protect data from accidental overwrites or deletions. It was configured from the Properties tab by turning on the Bucket Versioning option. Once enabled, each object update is stored as a new version with a unique ID.



Step 8: Enable Static Website Hosting

Static website hosting was enabled on the s3-demo-chaithanya bucket to serve web content directly from S3. This was configured by navigating to the Properties tab and enabling Static website hosting. An index document (e.g., index.html) is specified. The bucket policy was later adjusted to allow public read access for web access.



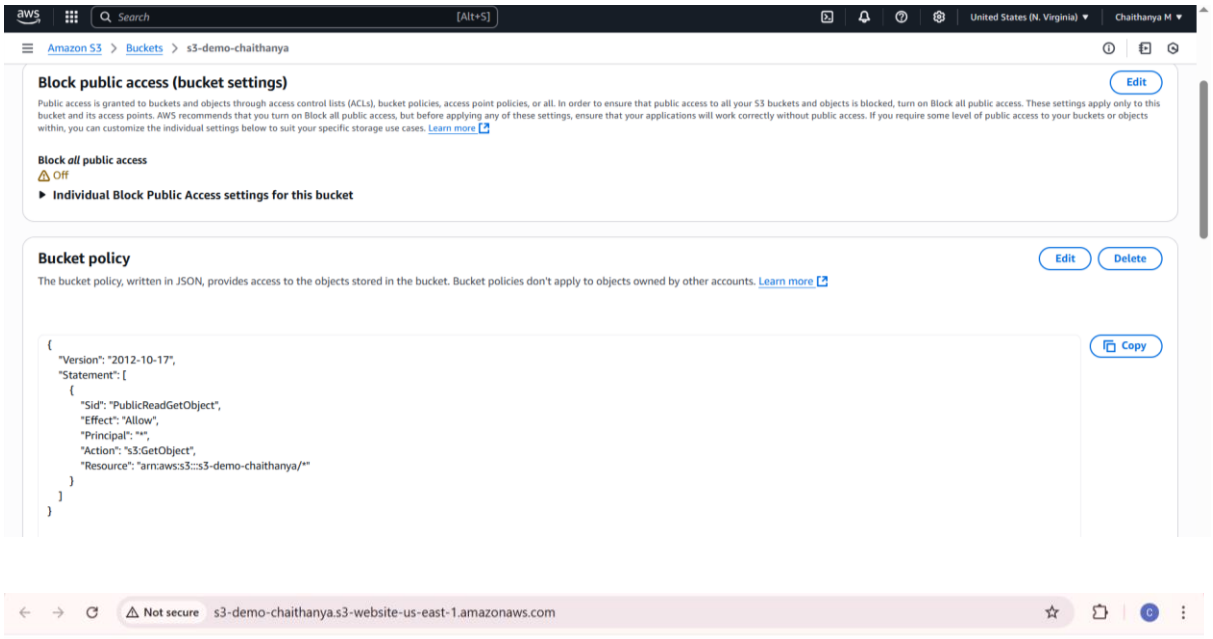
Still, I can’t able to access page.

Step 9: Update Bucket Policy for Static Website Access

To make the S3 bucket publicly accessible for static website hosting, the “Block all public access” setting was turned off, and a bucket policy was added to allow public read access. The following policy was applied:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
```

```
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::s3-demo-chaithanya/*"
  }
 ]
}
```



Demonsrating AWS s3 bucket versioning property

5. Conclusion

The S3 bucket configuration process demonstrated the full setup lifecycle, from creation and versioning to static website hosting and access control. Versioning was enabled to preserve object history and protect against unintentional overwrites or deletions. Static website hosting was successfully configured by enabling the hosting option, uploading required files (e.g., index.html), and updating the bucket policy to allow public read access.

Initially, public access was blocked due to restrictive settings and an incomplete bucket policy, resulting in a 403 Forbidden error. This issue was resolved by disabling the "Block all public access" setting and applying a proper bucket policy with the required s3: GetObject permissions. The correction emphasizes the importance of aligning bucket settings, object-level permissions, and IAM policies when hosting content publicly.