🗂️ AWS S3 Overview and Introduction

📘 What is Amazon S3?

Amazon S3 (Simple Storage Service) is a **fully managed object storage service** provided by AWS. It allows you to **store and retrieve any amount of data**, at any time, from anywhere on the internet.

Think of S3 as a **cloud-based drive** where you can upload files (called *objects*) into containers (called *buckets*).

🌐 **Features**:

- 🌍 Global availability
- 🧱 Object storage (not file or block storage)
- 🛡️ Built-in security, access controls, and encryption
- 🔄 Integration with other AWS services

🧰 AWS S3 Use Cases

Here are common ways people and organizations use S3:

- 📸 **Media storage** (images, videos, audio)
  📦 **Data backup and restore**
- 🧪 **Big Data analytics**
- 💾 **Static website hosting**
- 🚀 **Application data storage**
- 🧠 **Machine Learning training datasets**

What is an S3 Bucket?

An **S3 bucket** is like a **folder** where your data (objects) are stored in S3.

- 🪣 Each bucket has a **globally unique name**.
- 📍 Buckets exist within a specific **AWS Region**.
- 🛡️ Permissions, versioning, logging, and encryption are set at the bucket level.

Example:
A bucket named `my-photo-bucket` might contain thousands of image files.

Create Bucket

The process to create a new S3 bucket involves several key configuration steps:

🌍 AWS Region

- Choose the region where your bucket will reside.
- Data stored in S3 is region-specific (e.g., `us-east-1`, `ap-south-1`).
- Helps reduce latency and comply with data residency requirements.

📂 Bucket Type

- **General Purpose Buckets**:
    - The standard bucket type used for most workloads.
    - Supports all S3 features like versioning, lifecycle rules, and access control.
- **Directory Buckets** *(Newer Feature)*:
    - Designed for large-scale, analytics-type workloads.
    - Optimized for performance in data lakes and compatible with file-like access patterns.

🏷️ Bucket Name

- Must be **globally unique** across all AWS accounts.
- Must follow DNS naming rules (e.g., no uppercase letters, no underscores).
- Examples: `my-project-logs`, `company-data-bucket`

🔁 Copy Settings from Existing Bucket *(Optional)*

- Clone configurations like permissions, encryption, and versioning from an existing bucket.
- Useful for consistency in multi-bucket environments.

👥 Object Ownership

- Determines who owns newly uploaded objects:
  - **Bucket owner preferred** (recommended for team/shared environments)
  - **Object writer** (for individual uploads)
- Impacts cross-account access and billing.

🚫 Block Public Access Settings for this Bucket

- Strongly recommended to **keep public access blocked** unless explicitly needed.
- Combines multiple settings:
  - Block public ACLs
  - Block public bucket policies
  - Ignore public ACLs
  - Restrict public bucket policies

♻️ Bucket Versioning

- Enables **multiple versions** of the same object.
- Helps with accidental deletes/overwrites.
- Useful for backups and auditing.

🔐 Default Encryption

- Automatically encrypts all new objects in the bucket.
- Options:
    - **SSE-S3**: AWS-managed keys
    - **SSE-KMS**: AWS Key Management Service
    - **SSE-C**: Customer-provided keys

Click Create

📤 Uploading Files to S3 Bucket

Once your S3 bucket is created, you can easily upload files such as images, videos, documents, and more.

🖱️ Click "Upload"

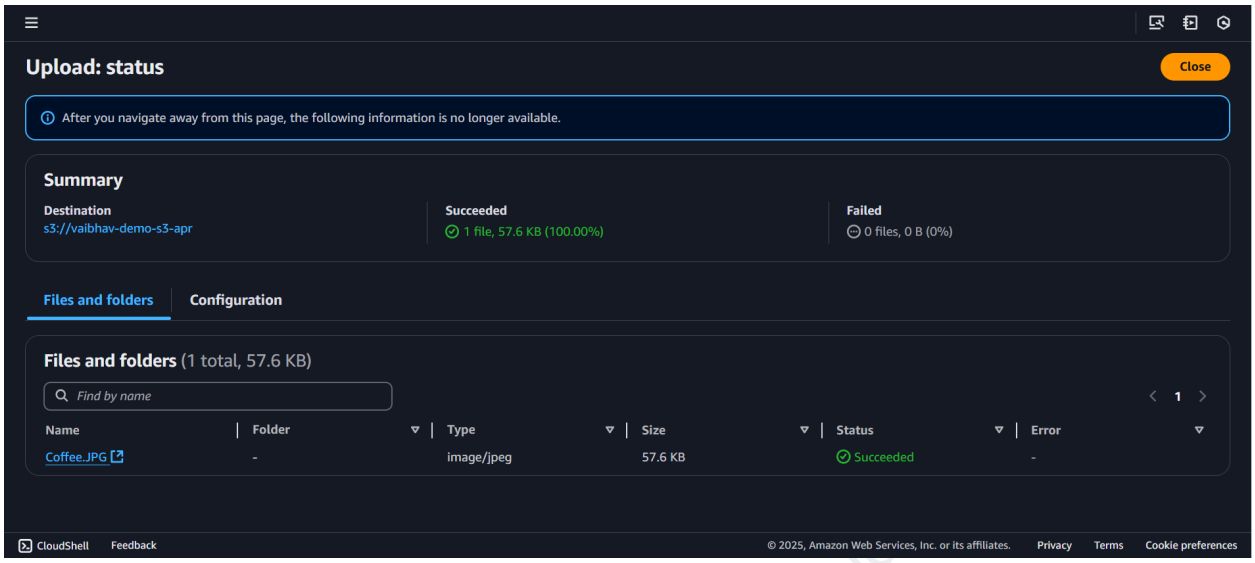
- Open your S3 bucket from the AWS Management Console.
- Click the **"Upload"** button at the top of the bucket page.

🖼️ Select Image (or Other File)

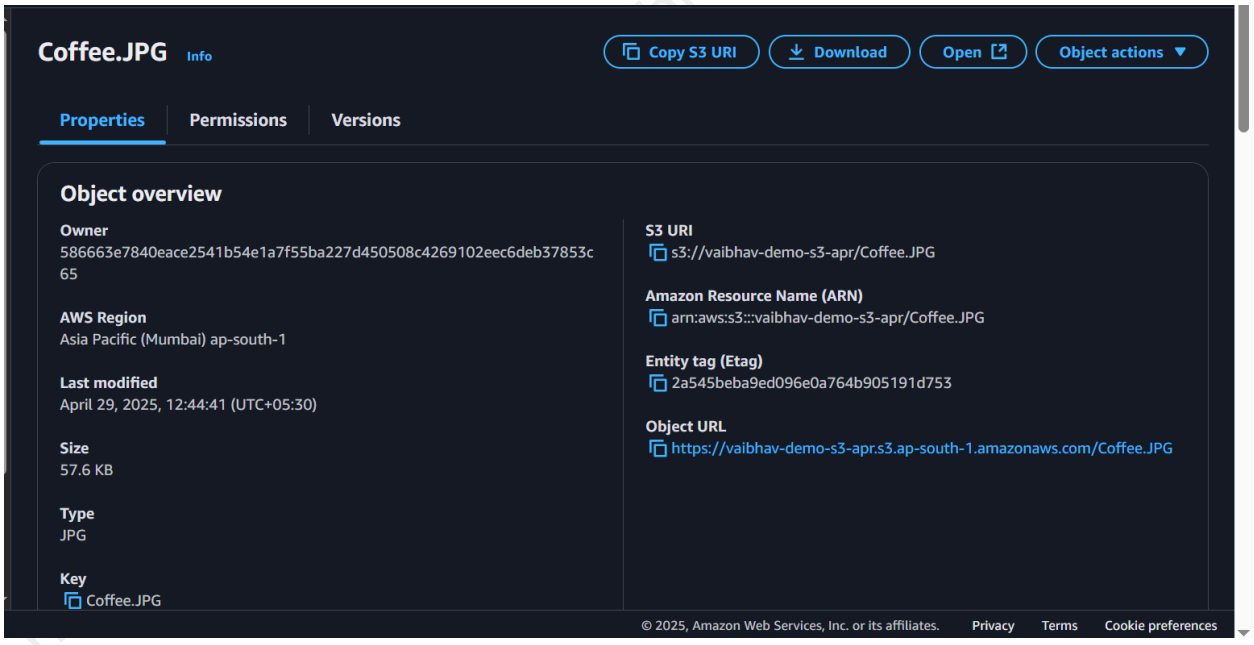
- In the upload wizard:
    - Click **"Add Files"**.
    - Browse and select an **image**, **video**, or any other file from your local system.
    - You can upload multiple files at once if needed.

⬆️ Click "Upload"

- After selecting the file(s), click the **"Upload"** button.
- The file(s) will be uploaded to the bucket.
- You can monitor the upload progress and view completion status.



Click on the File or Image that you uploaded and you can all the details and properties of that file.



👁️ Viewing Uploaded File in S3

Once you've uploaded a file (like an image), there are two ways to try accessing it:

🧭 Click "Open"

- From the S3 console:
    - Navigate to your uploaded object.
    - Click the **"Open"** button.
    - If you're logged in and have permissions, the image/file will open directly in a new tab.

🔗 Copy "Object URL"

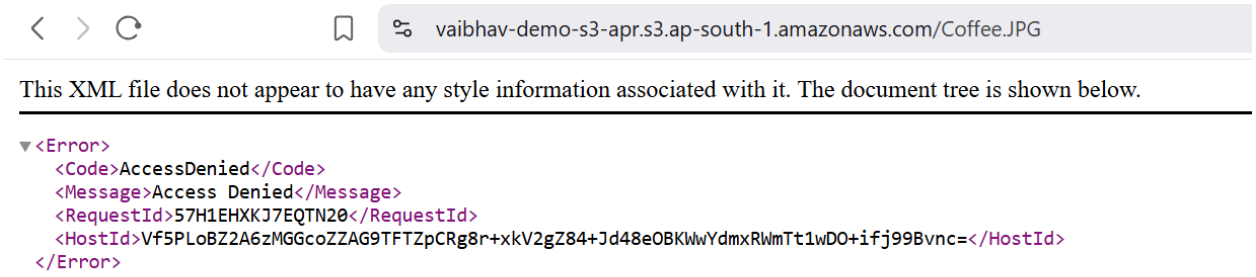
- Every object in S3 gets a unique **Object URL** (e.g., `https://bucket-name.s3.amazonaws.com/image.jpg`).
- You can copy this and try to open it in a web browser.

❌ Access Denied Issue

- When you try to open the Object URL directly in a browser, you will often see:

  "Access Denied"

- This happens because:
    - The object is **not publicly accessible**.
    - S3 blocks public access by default for security.
    - The object or bucket lacks a policy that allows public reads.



🔐 S3 Security

S3 offers multiple layers of security to control **who can access your data** and **how it is protected**.

👤 User-Based Security

- Managed using **IAM (Identity and Access Management)**.

🗒️ IAM Policies

- Attach policies to **users**, **groups**, or **roles**.
- Define what actions a user can perform on S3 (e.g., list, upload, delete).

Resource-Based Security

📄 Bucket Policies

- Define access rules directly on the bucket.
- Use **JSON** to specify access for users, accounts, or the public.

📄 Object ACLs (Access Control Lists)

- Set permissions **per object**.
- Limited use; better to prefer IAM or bucket policies for clarity and security.

🗒️ Bucket ACLs

- Older way to share buckets between AWS accounts.
- Still supported, but generally discouraged in favor of bucket policies.

🔐 Encryption

- Protects data **at rest**.
- Options include:
    - **SSE-S3** (Server-side encryption with S3-managed keys)
    - **SSE-KMS** (KMS-managed keys)
    - **SSE-C** (Customer-provided keys)

📜 S3 Bucket Policies

Bucket policies are **JSON-based** documents used to control access at the bucket level.

🗒️ Structure of a Bucket Policy

Here's a sample bucket policy and a breakdown of its components:

json Copy Edit

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-bucket-name/*"
    }
  ]
}
```

🔍 Explanation of Fields

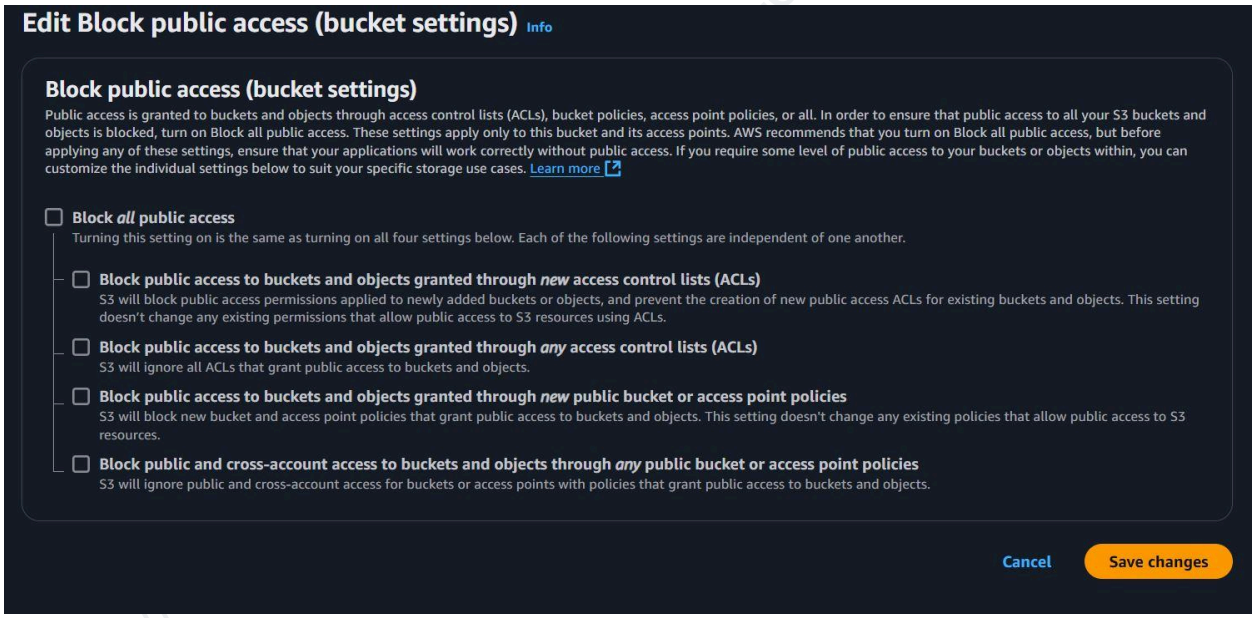
Field	Description
`"Version"`	Policy language version. Always use `"2012-10-17"`.
`"Statement"`	Main policy block that includes access rules.
`"Sid"`	Statement ID. Optional but useful for naming.
`"Effect"`	`"Allow"` or `"Deny"` access.
`"Principal"`	Who the policy applies to. `"*"` means everyone (public).
`"Action"`	The operation being allowed (e.g., `s3:GetObject`).
`"Resource"`	The specific bucket or object ARN (e.g., `"arn:aws:s3:::my-bucket-name/*"`).

🧪 S3 Bucket Policy Hands-On: Accessing an Image via Public Object URL

This step-by-step guide shows how to make an S3 object (e.g., an image) publicly accessible using a bucket policy.

🛑 Step 1: Disable "Block Public Access"

1. Go to your S3 bucket in the AWS Management Console.
2. Click on the Permissions tab.
3. Find "Block public access (bucket settings)" → Click Edit.
4. Uncheck the box labeled "Block all public access".
5. Confirm the warning and click Save Changes.

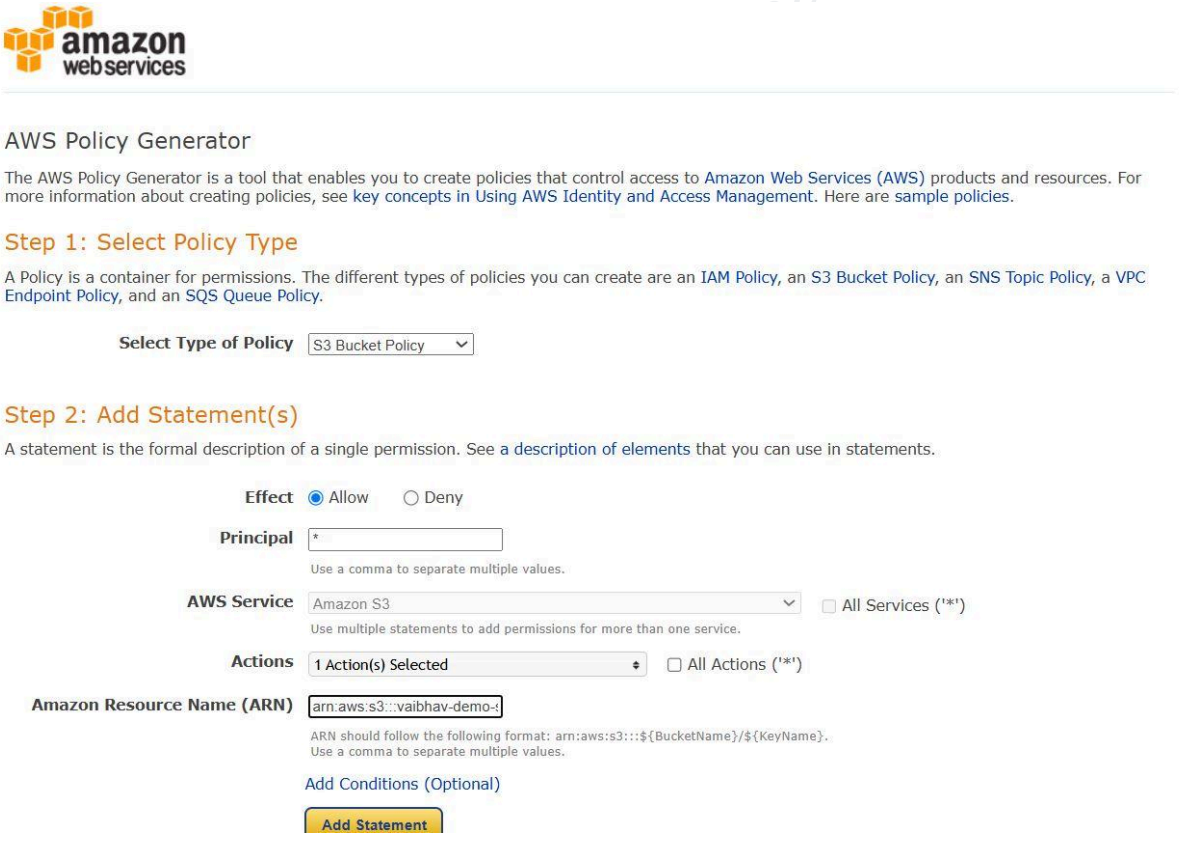


📜 Step 2: Add a Bucket Policy to Allow Public Read Access

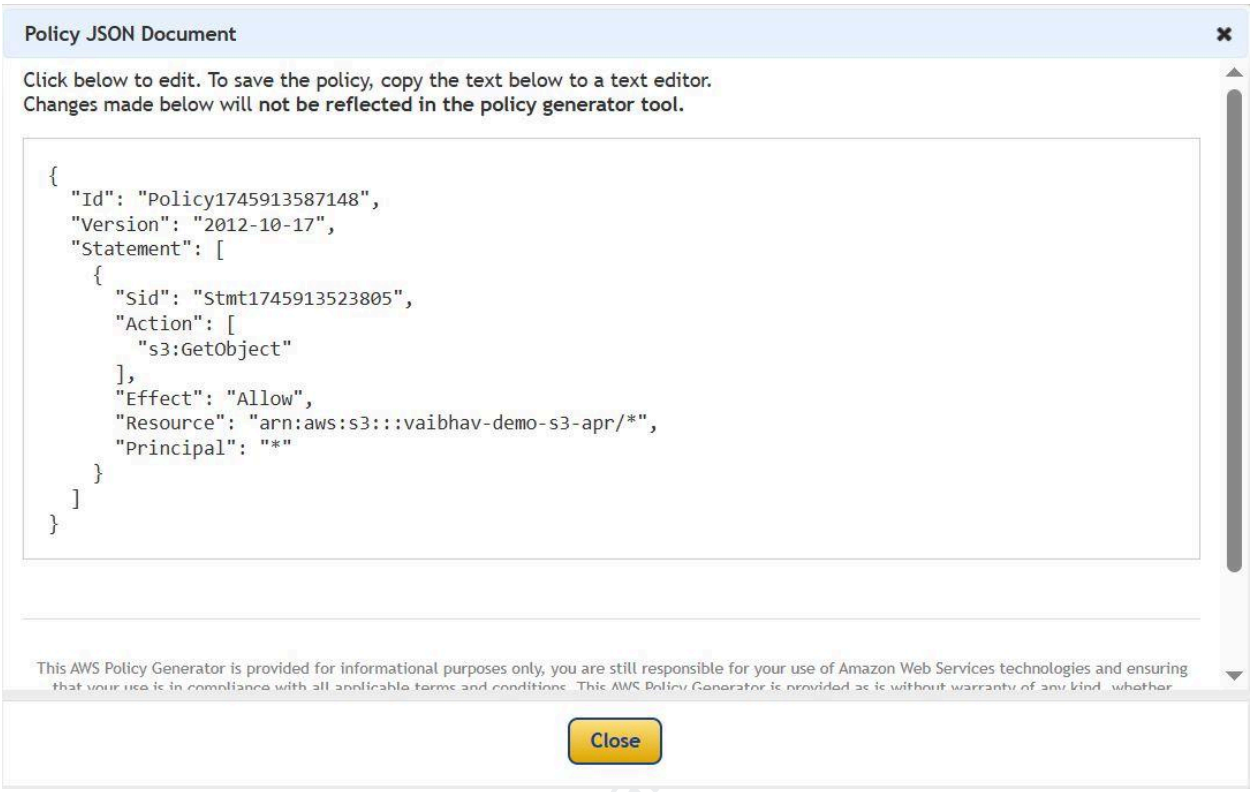
1. In the same Permissions tab, scroll down to Bucket Policy and click Edit.
2. Click Policy Generator to create your policy:

Select Type: S3 Bucket Policy

- Effect: Allow
- Principal: `*` (for public access)
- Action: `GetObject`
- ARN:
  - Go to your bucket → Copy the Bucket ARN
  - Append `/*` at the end (this means the policy applies to all objects in the bucket)
  - Example: `arn:aws:s3:::my-bucket-name/*`

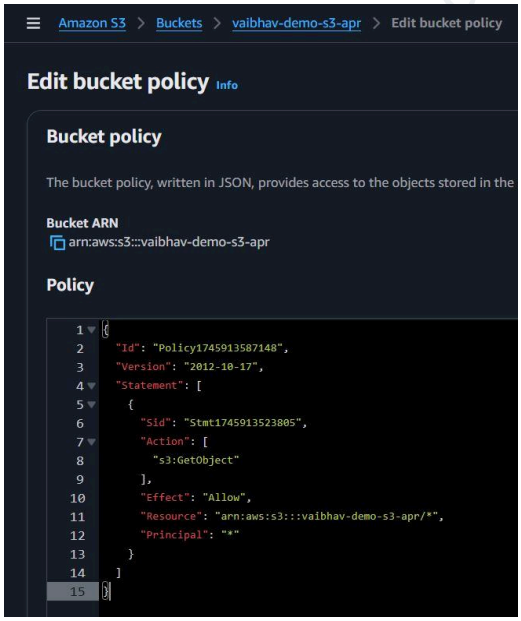


3. Click Add Statement, then Generate Policy.



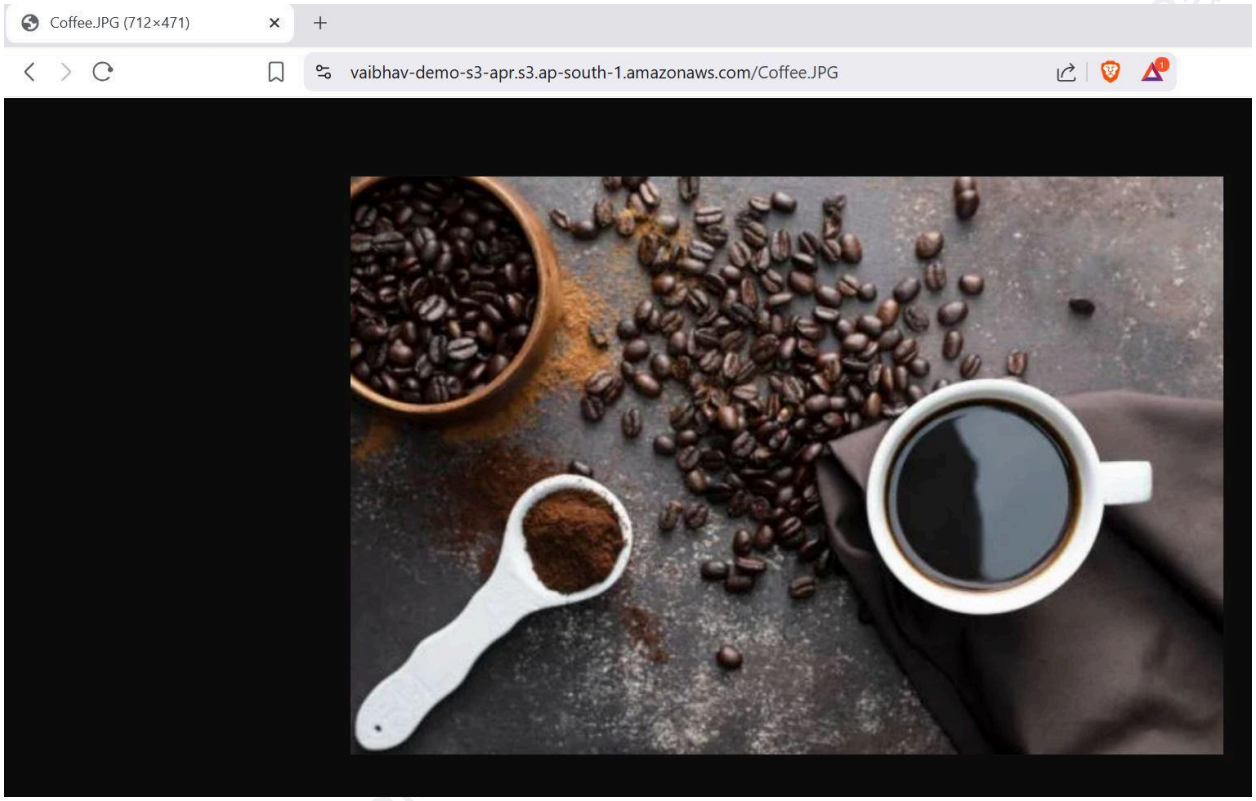
4. Copy the generated JSON.

5. Paste it into the Bucket Policy editor and click Save Changes.



🌐 Step 3: Test the Public Access

1. Go to the Objects tab in your bucket.
2. Select the uploaded image (or any file).
3. Copy the "Object URL" displayed on the top.
4. Paste it into your browser.



✅ Now, your image is **publicly accessible** using its URL—thanks to the **bucket policy** you just applied!

🌐 S3 Static Website Hosting Hands-On

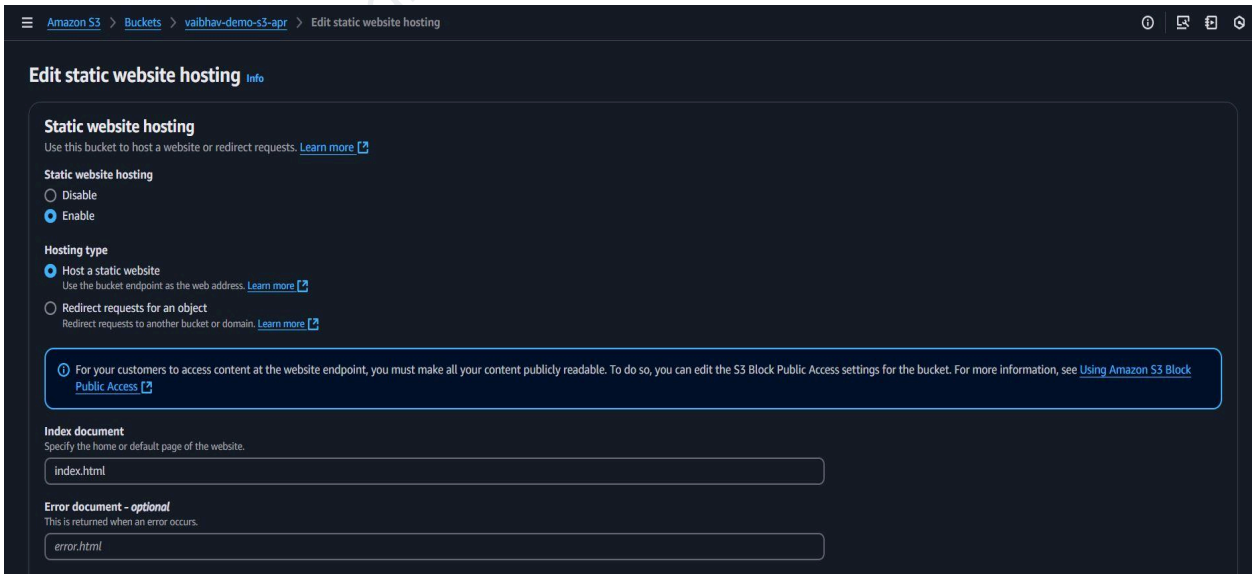
In this hands-on, you will host a simple static website using Amazon S3. The website will display images stored in your bucket (e.g., `coffee.jpg` and `beach.jpg`).

🖼️ Step 1: Upload Another Image

1. Upload a new image to your bucket:
    - Example: `beach.jpg`
2. Now your bucket contains at least two images:
    - `coffee.jpg`
    - `Beach.jpg`

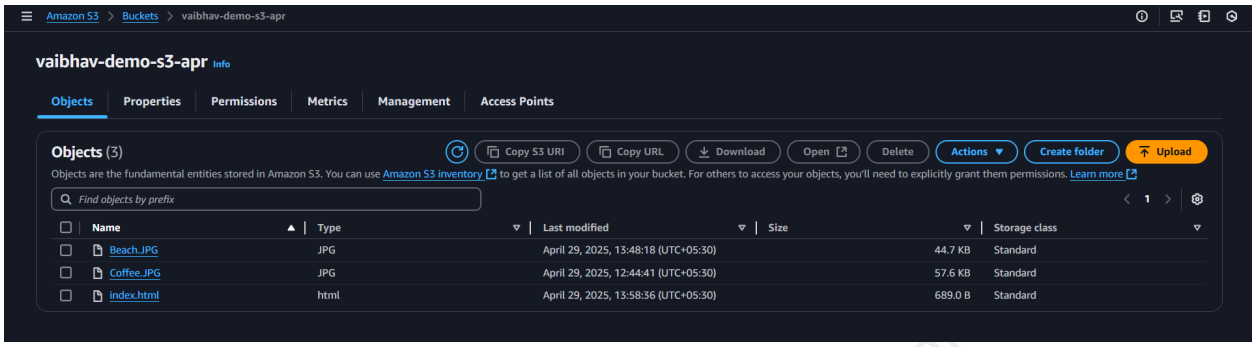
⚙️ Step 2: Enable Static Website Hosting

1. Go to your **S3 Bucket** in the AWS Console.
2. Navigate to the **Properties** tab.
3. Scroll to **Static Website Hosting** → Click **Edit**.
4. Check **Enable**.
5. Select **Hosting type**: Host a static website
6. In the **Index document** field, type index.html
7. Click **Save Changes**.



📄 Step 3: Upload the HTML File

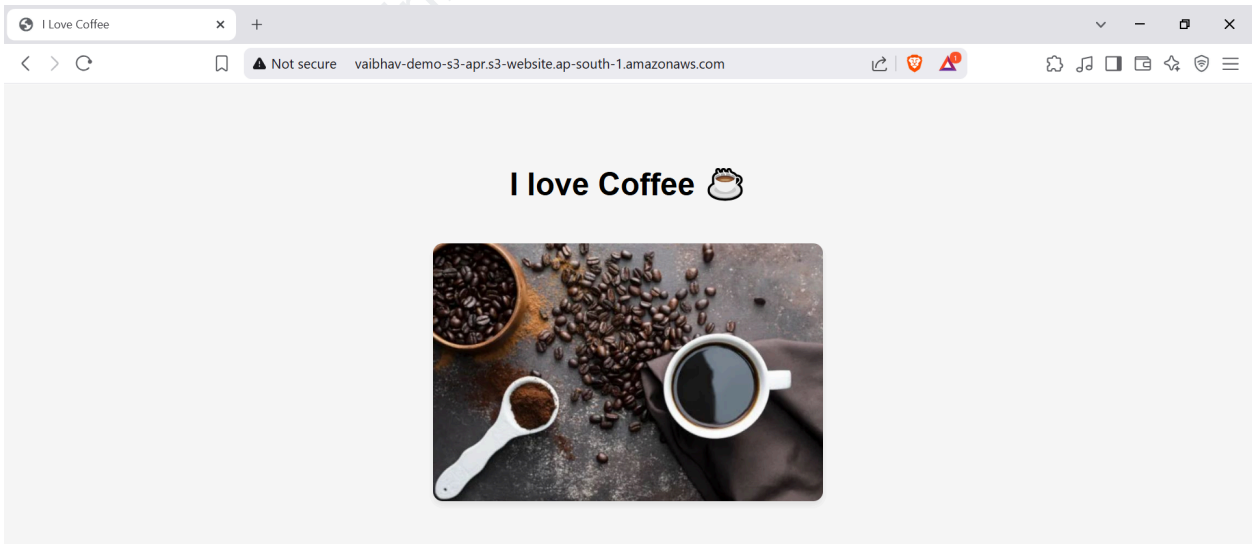
1. Upload your index.html file to the same bucket.
    - Make sure it references your image objects using relative or full S3 paths.



🔗 Step 4: Access the Static Website

1. Go to **Properties** → **Static Website Hosting**.
2. Copy the **Bucket website endpoint** (e.g., http://my-bucket.s3-website-us-east-1.amazonaws.com)
3. Paste it into your browser.

✅ You should now see your static website displaying the uploaded images!

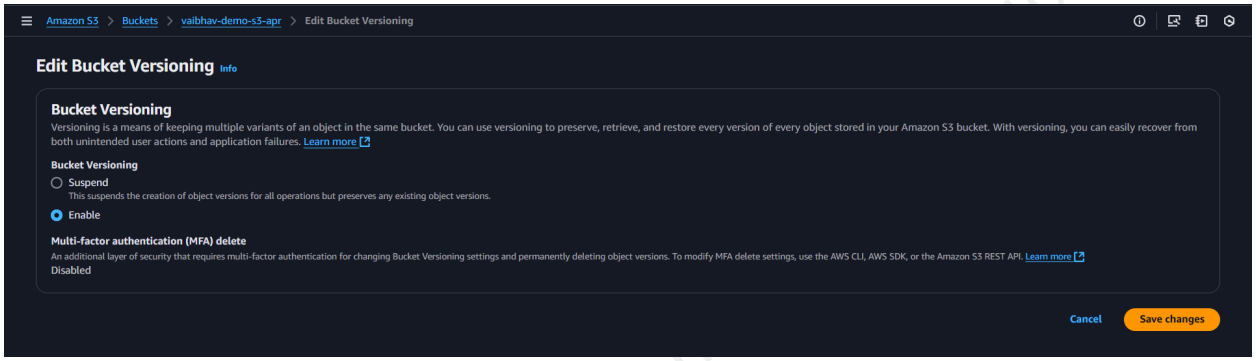


🧪 S3 Versioning Hands-On Guide

Understand Amazon S3 Versioning, its use cases, and how to use it to manage object changes (e.g., overwrites, restores).

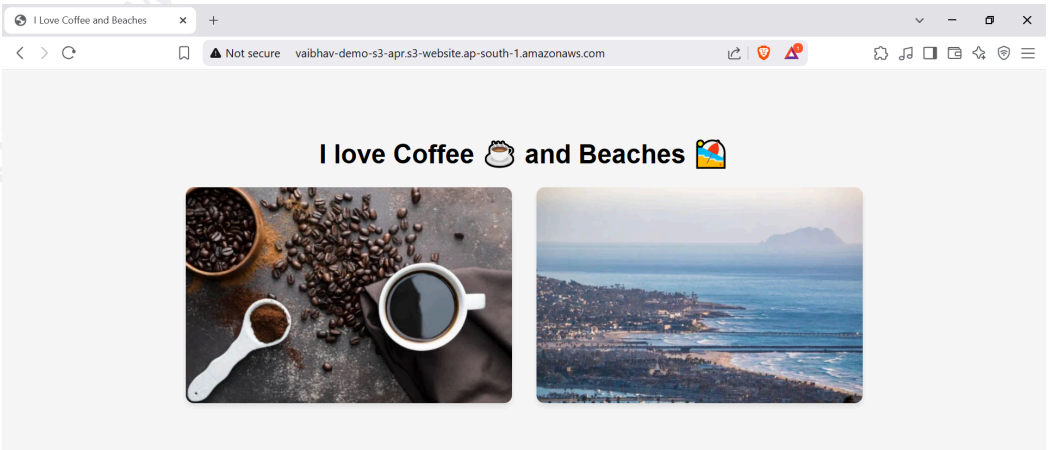
🔸 Step 1: Enable Versioning on the S3 Bucket

1. Go to the S3 bucket in the AWS Console.
2. Click on the “Properties” tab.
3. Scroll to “Bucket Versioning”.
4. Click Edit → select “Enable” → click Save Changes.



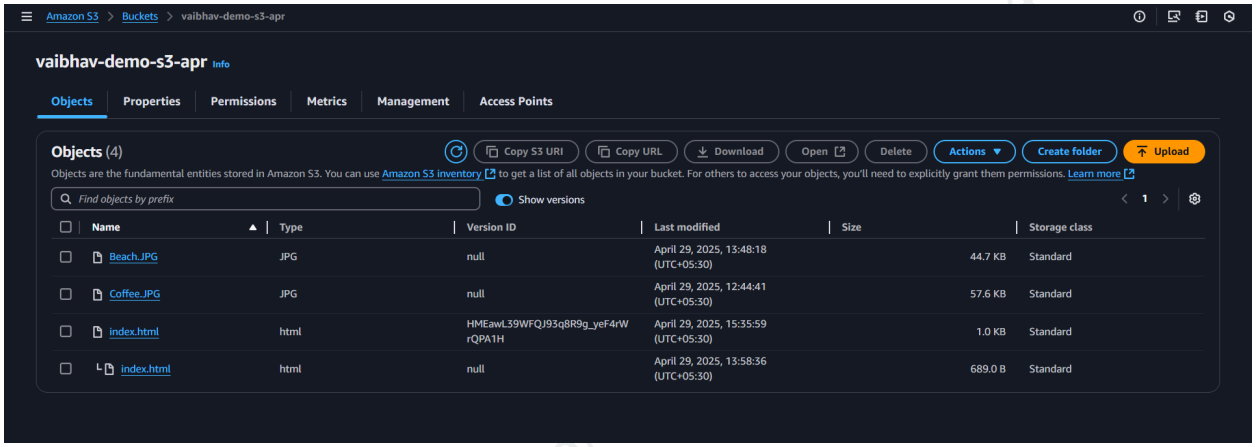
🔸 Step 2: Modify an Object in the Bucket

1. In your bucket, locate your existing object: `index.html`.
2. Upload a new version of `index.html` (e.g., with added content or image references like `Beach.jpg`).
    - This action overwrites the existing file but **preserves the old version**.



🔸 Step 3: View Object Versions

1. In the bucket's “Objects” tab, click the “Show versions” toggle (upper right).
2. You will now see:
    - All historical versions of the `index.html` file.
    - The current version will have a unique version ID.
    - The original file (before versioning was enabled) will have a Version ID: `null`.



🔹 What is a "null" version ID in S3?

When S3 versioning is not yet enabled, any object uploaded to the bucket is stored without a version ID. These objects are given a **"null"** version ID by default.

Once versioning is enabled, all newly uploaded versions of an object will receive a unique version ID, but the previously existing object with no version ID is still retained as the ***null version***.

🔹 Use Case in Your Example

- Before versioning was enabled, we uploaded `index.html`, `Coffee.JPG`, and `Beach.JPG` — all of these were marked with `Version ID = null`.
- After enabling versioning, when we uploaded a new version of `index.html`, S3 created a new version with a unique ID, while keeping the original "null" version intact.

✅ Use Cases for S3 Versioning

- Accidental Deletion Protection: Deleted objects can be restored using previous versions.
- Audit Trail: Track changes over time for compliance or debugging.
- Data Recovery: Recover overwritten files with minimal effort.
  Immutable Archives: Maintain historical versions for backup or legal records.

🌍 What is AWS S3 Replication?

AWS S3 Replication is a feature that allows automatic, asynchronous copying of objects across **buckets**. It helps meet compliance, disaster recovery, and data locality requirements.

Replication **only works between buckets** that have **versioning enabled**.

🔁 Types of S3 Replication

✅ CRR – Cross-Region Replication

- Replicates objects **from one AWS Region to another**.
- Ideal for:
  - **Disaster recovery**
  - **Compliance** (e.g., storing copies in a different legal jurisdiction)
  - **Low-latency access** from different geographical locations

✅ SRR – Same-Region Replication

- Replicates objects **within the same AWS Region**.
- Ideal for:
  - **Log aggregation**
  - **Replication across different accounts**
  - **Compliance/internal backup** needs within a region

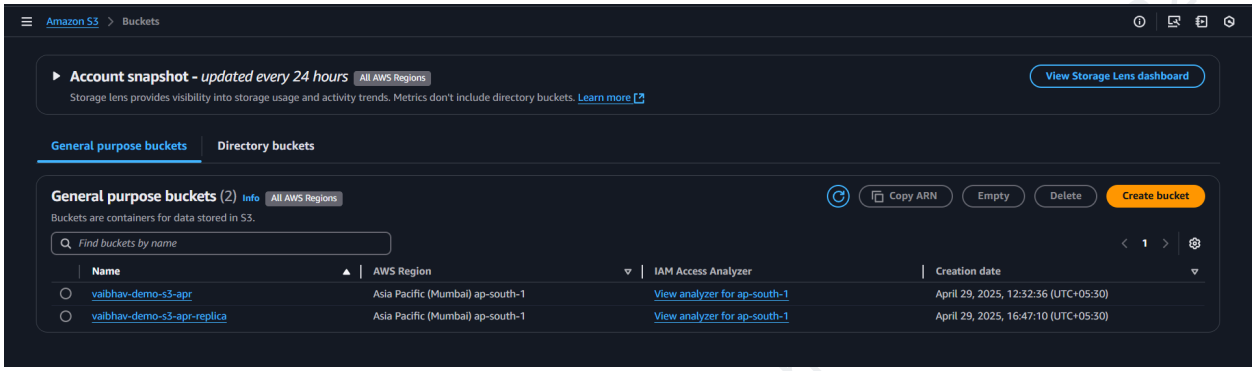
⚠️ Key Facts about S3 Replication

- ✅ **Only new objects are replicated** after replication is enabled.
- Replication only works when the Versioning is Enabled.
- 🕒 **Existing objects are not automatically replicated**.
  - To replicate them, use **S3 Batch Replication**.
- 🔄 **Replication is not transitive** (no *chaining*).
  - Example: If Bucket A replicates to Bucket B, and Bucket B replicates to Bucket C, **Bucket C will not receive objects from Bucket A**.

AWS S3 Replication Hands-On

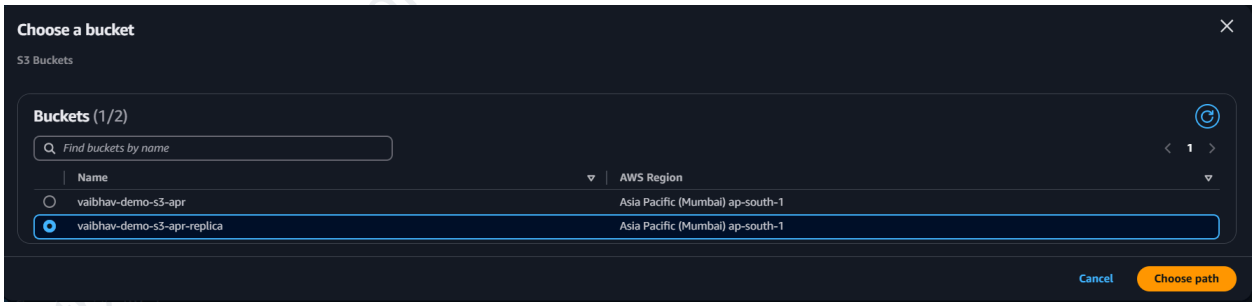
🏗️ Bucket Setup

- You must have an **original/source bucket** already created.
- Create a **new bucket** in the same region — this will be the **replica/destination bucket**.

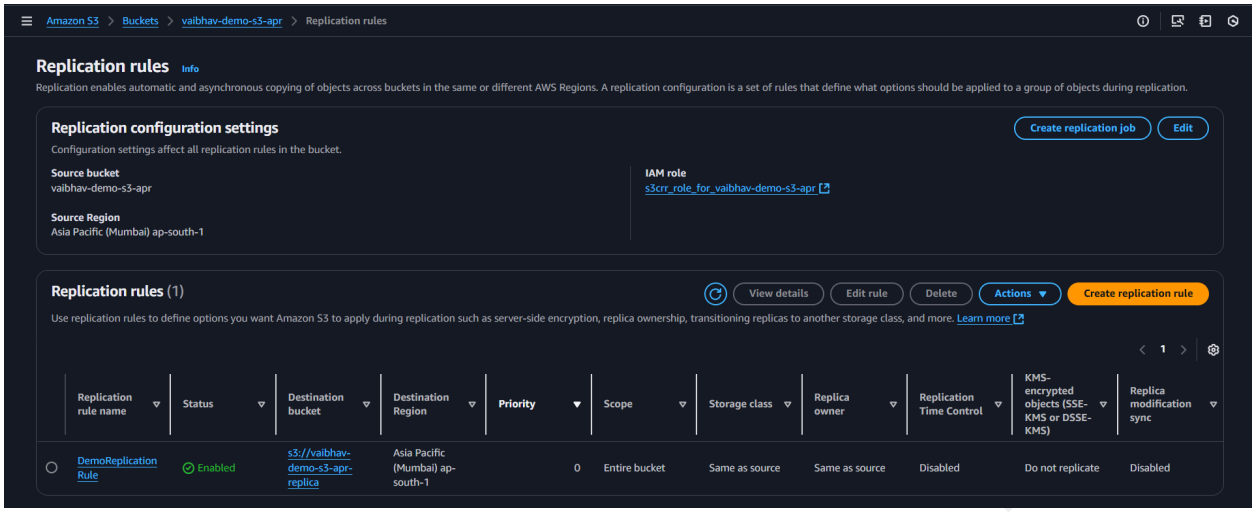


⚙️ Setting Up Replication

1. Go to the **Source Bucket** → **Management** tab.
2. Under **Replication Rules**, click **"Create replication rule"**.
3. Set a **name** for the rule.
   For **Destination**:
   - Choose **“Bucket in this account”**.
   - Click **Browse** and select the **new replica bucket**.

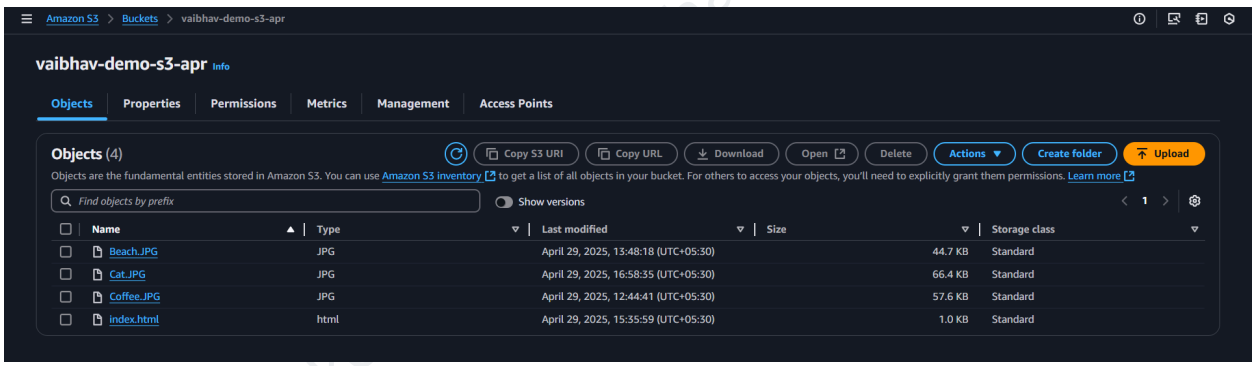


4. For **IAM Role**:
   - Choose **“Create new role”**.
5. Click **Save**.

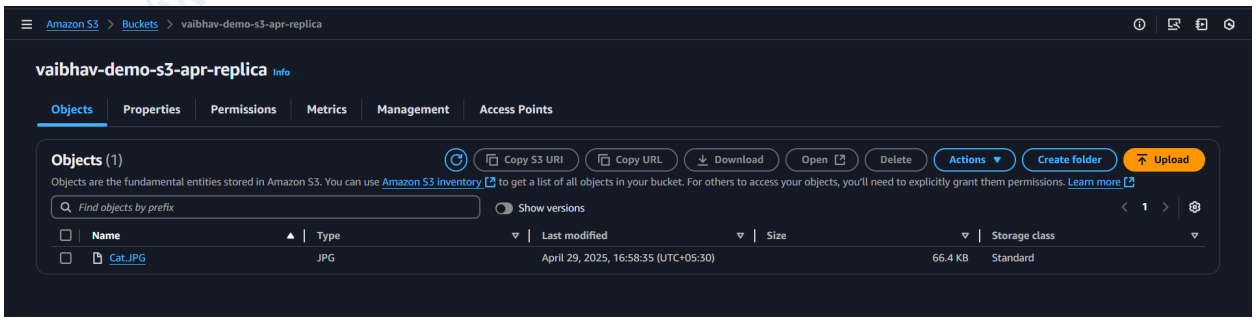


📂 Uploading Files

- Upload a file (e.g., `Cat.jpg`) into the **Source bucket**.



- Check the **Replica bucket** — `Cat.jpg` will appear automatically.



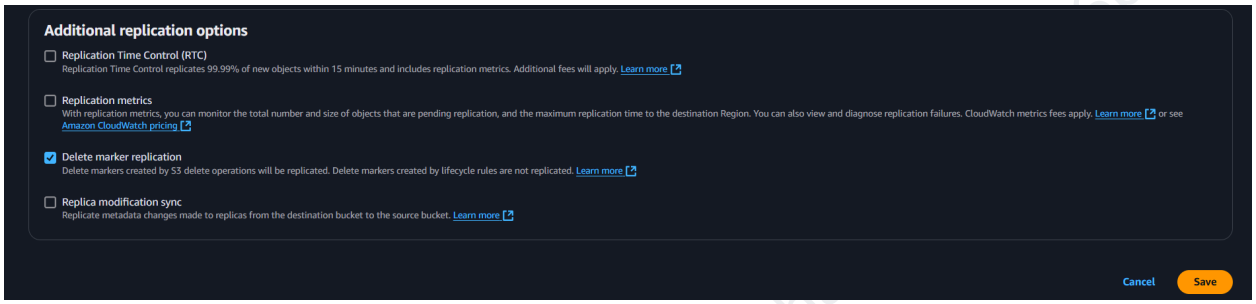
- Both buckets will show the **same Version ID** for the file.

🚫 Delete Marker Behavior

By default, **delete markers are not replicated**.

You can change this by:

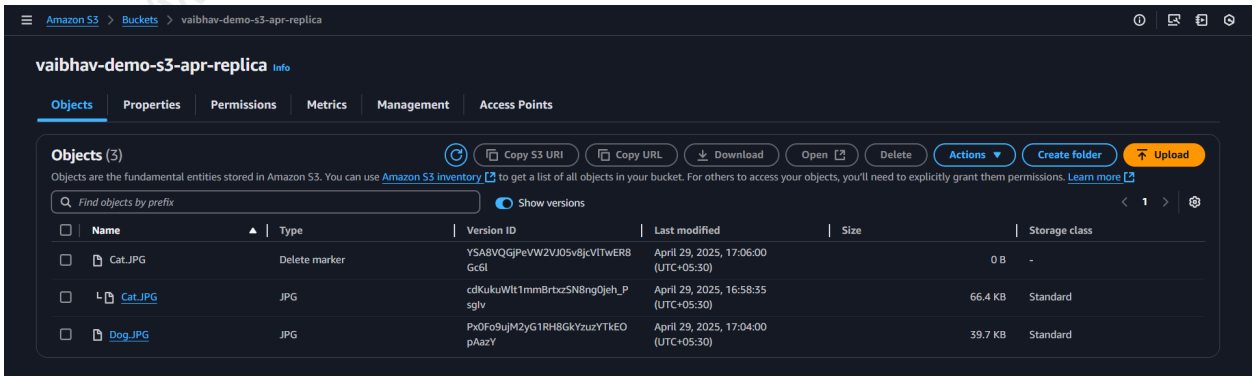
1. Editing the replication rule.
2. Going to **Additional replication options**.



3. Enabling:
   ✅ **Delete marker replication**
   (Only delete markers created by **manual delete operations** are replicated — not those from **lifecycle rules**.)

🧪 Testing Delete Marker Replication

1. Upload another file like `Dog.jpg` to the **Source bucket**.
2. Delete `Cat.jpg` from the **Source bucket**.
   - A **delete marker** is created.
3. Check the **Replica bucket** — the **delete marker for `Cat.jpg`** should now appear (if delete marker replication is enabled).



🗃️ AWS S3 Storage Classes

Each **S3 storage class** offers different durability, availability, and cost based on data access patterns. All classes (except **S3 Glacier** and **S3 Glacier Deep Archive**) provide **11 9's durability** — `99.999999999%`.

💡 S3 Intelligent-Tiering

- **Durability**: 99.999999999%
- **Availability**: 99.9%
- **Purpose**: Automatically moves data between frequent and infrequent tiers based on usage.
- **Use Case**: Unknown or unpredictable access patterns.

📦 S3 Standard

- **Durability**: 99.999999999%
- **Availability**: 99.99%
- **Purpose**: Default class for frequently accessed data.
- **Use Case**: Websites, apps, big data analytics.

🕑 S3 Standard–IA (Infrequent Access)

- **Durability**: 99.999999999%
- **Availability**: 99.9%
- **Purpose**: For data accessed less frequently, but requires rapid access when needed.
- **Use Case**: Backups, disaster recovery.

🧊 S3 One Zone–IA

- **Durability**: 99.999999999% (in a single AZ)
- **Availability**: 99.5%
- **Purpose**: Lower-cost option for infrequent access stored in one AZ only.
- **Use Case**: Re-creatable data, secondary backups.

🧬 S3 Glacier

- **Durability**: 99.999999999%
- **Availability**: Not published
- **Purpose**: Low-cost archival with retrieval times from minutes to hours.
- **Use Case**: Long-term backups, digital archives.

🧬❄️ S3 Glacier Deep Archive

- **Durability**: 99.999999999%
- **Availability**: Not published
- **Purpose**: Lowest-cost option for long-term archiving (retrieval in hours).
- **Use Case**: Compliance archives, rarely accessed data.

🔁 S3 Reduced Redundancy Storage (Legacy)

- **Durability**: 99.99%
- **Availability**: 99.99%
- **Purpose**: For non-critical, reproducible data.
- **Note**: **Not recommended** for most modern use cases.