**Task: Host a Static Website on S3 with Versioning**

**Name: Abhishek Hengade**

**Batch: B-56**

**S3 Static Website Hosting & Versioning :**

This document provides an in-depth, step-by-step guide to deploying a secure and resilient static website using Amazon S3. It covers the foundational concepts of bucket configuration, public access, versioning, and the final deployment steps.

**## 1. S3 Bucket Creation and Configuration**

The S3 bucket is the core container for your website's assets. Its initial configuration is crucial for security, management, and functionality.

* **Bucket Name and Region**
  + **Name**: Your bucket name must be globally unique. It's a best practice to name it after your domain (e.g., www.example.com) for easier management with services like Route 53.
  + **Region**: The AWS Region should be chosen based on proximity to your primary user base to minimize latency and improve website load times.
* **Object Ownership and Access Control Lists (ACLs)**
  + **ACLs**: This is a legacy method for managing permissions on a per-object basis. By selecting **"ACLs enabled"** and **"Bucket owner preferred,"** you ensure that you, as the bucket owner, retain full control over all objects, even if uploaded by other accounts. While bucket policies are now the recommended approach for most use cases, understanding ACLs is still valuable.
* **Public Access Settings**
  + By default, S3 blocks all public access for security. To host a public website, you must disable **"Block all public access."** This action doesn't make your objects public; it only removes the top-level restriction, allowing you to grant public access through more granular methods like ACLs or bucket policies later.
* **Bucket Versioning (Critical Feature)**
  + Enabling **Versioning** is a key practice for data durability and disaster recovery. It instructs S3 to create a complete history of all objects in the bucket.
  + **How it Works**: When a file is overwritten, S3 creates a new version instead of replacing the old one. When a file is deleted, S3 creates a "delete marker" but preserves all previous versions.
  + **Benefits**:
    - **Undelete**: You can easily recover from accidental deletions by simply removing the delete marker.
    - **Rollback**: If a new version of your website has issues, you can instantly roll back to a previous, stable version of any file.

**## 2. Uploading and Managing Website Content**

With the bucket configured, you can now upload your website's static files.

* **Uploading Files**
  + The S3 console allows you to upload entire folders, preserving your website's directory structure (e.g., /css, /images, /js). S3 represents these folders as object prefixes.

**## 3. Enabling Public Access and Static Hosting**

These final steps turn your S3 bucket into a publicly accessible web server.

* **Granting Public Read Access**
  + Files in a public bucket are not public by default. You must explicitly grant read permissions. Using the **"Make public using ACL"** action is a quick way to apply public-read permissions to all your website's files, allowing anyone on the internet to view them.
* **Enabling Static Website Hosting**
  + This feature tells S3 to serve your content through an HTTP endpoint.
  + **Index Document**: This is the default file served when a user accesses your site's root URL (e.g., http://www.example.com/). This is almost always index.html.
  + **Error Document**: It's a best practice to specify a custom error page (e.g., error.html). This provides a better user experience than the default XML error message when a visitor requests a page that doesn't exist.

**## 4. Accessing Your Live Website**

Once configured, S3 provides a unique URL to access your website.

* **Website Endpoint**
  + The URL follows the format: http://<bucket-name>.s3-website.<Region>.amazonaws.com.
  + You can use this URL for testing or map it to a custom domain name using Amazon Route 53 for a professional appearance. Navigating to this URL will now serve your index.html page and all its assets.

