Lab06

Managing Google Cloud S3 Storage using Terraform

Contents

Scenario

You are working for a startup that runs a lightweight media-sharing platform. The platform team needs a secure, scalable way to:

- Store static media files (e.g. images, text)
- Prevent public access to objects over unsecured connections
- Automatically transition unused content to cold storage
- Allow temporary access to select files for sharing or testing

You have been asked to use Terraform to provision the required infrastructure on Google Cloud Platform (GCP).

Business Requirements

Bucket Setup

1. Create a Google Cloud Storage (GCS) bucket in `us-central1` with a globally unique name

2. Enable **versioning** on the bucket

Secure Access Controls

3. Implement a **bucket policy** to **deny access to any user using HTTP** (unencrypted transport)

Static File Upload

4. Upload all files from a local directory called static_files

5. Ensure each object has the correct **MIME type** assigned, based on file extension

Storage Management

6. Apply a lifecycle policy to: - Transition objects to Nearline or Coldline storage after 30 days. - Delete them after 90 days.

Generate Secure URL

7. Generate a **Signed URL** that provides **read-only** access to Teide.jpeg

8. Set the URL to **expire in 1 hour**

Verification

9. Post the generated **Signed URL** into the class chat window so the instructor can test access

Your Files

10. Use **googlelabs/lab06** as the root module location for your Terraform code. In that directory is a folder called static_files containing sample files of various types. Also provided is mime.json for MIME type association.

Rules

11. Use Terraform only — no manual actions in the GCP Console unless debugging.

12. You may use the GCP Console to verify deployments, but not to modify resources.

13. The bucket name must include your initials or student ID to ensure global uniqueness (e.g. googlelabs`-mcg-static-5678`).

14. The lab environment is pre-authenticated — no need to handle GCP credentials in your code.

Success Criteria

15. GCS bucket appears in the GCP Console with versioning enabled.
16. Public access is blocked by IAM policy.
17. All files in `static_files/` are uploaded with correct content type metadata.
18. Lifecycle rules are visible and correctly configured.
19. Signed URL for `Teide.jpeg` works in the browser (valid for 1 hour).
20. You share the Signed URL in the chat window.S3 bucket appears in the AWS Console with versioning enabled

Solution

21. A proposed solution to this challenge can be found in the solutions folder. Only use this as a last resort.

Student Reference Guide – GCS Terraform Challenge

GCS Bucket Basics
Access Control
Upload Multiple Files with MIME Type
Generating Signed URLs
Loading and Using JSON in Terraform