QATIP Intermediate
AWS Lab07
Managing AWS 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've been asked to use **Terraform** to provision the required infrastructure on AWS.

Business Requirements

Bucket Setup

1. Create an **S3 bucket** in us-east-1 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 files to **GLACIER** storage after 30 days
    - **Delete** them after 90 days

Generate Secure URL

7. Generate a **Pre-Signed S3 URL** that provides **read-only** access to Teide.jpeg
8. Set the URL to **expire in 1 hour**

Verification

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

Your Files

10. Use **aws-tf-int/labs/07** as the root moule location for your terraform code. In that directory is a folder called static_files containing sample files of various type. Also provided is mime.json file, to be used for mime type association.

Rules

11. Use Terraform only — no manual actions in the AWS Console unless debugging
12. You may use the AWS 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. qatipint-mcg-static-1234)
14. The lab environment is pre-authenticated — no need to handle AWS credentials in your code

Success Criteria

15. S3 bucket appears in the AWS Console with versioning enabled
16. Bucket policy prevents HTTP-based access
17. All files in static_files/ are uploaded with correct content type metadata
18. Lifecycle rules are visible and correctly configured
19. Pre-signed URL for Teide.jpeg works in the browser (valid for 1 hour)
20. You share the **Pre-Signed URL** in the chat window

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 – AWS S3 Terraform Challenge

AWS S3 Bucket Basics

Terraform Resource Docs:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket

AWS S3 Documentation:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-buckets-s3.html

Enable Versioning

Terraform Resource Docs:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_versioning

AWS S3 Versioning Concepts:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html

Deny HTTP Access (Bucket Policy)

Terraform Resource Docs:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_policy

Policy Example (Deny unencrypted transport):
https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-policy-for-config-rule/

Upload Multiple Files with MIME Type

Terraform Function Docs - fileset:
https://developer.hashicorp.com/terraform/language/functions/fileset

Terraform Function Docs - filemd5:
https://developer.hashicorp.com/terraform/language/functions/filemd5

Terraform Function Docs - regex:
https://developer.hashicorp.com/terraform/language/functions/regex

Terraform Function Docs - lookup:
https://developer.hashicorp.com/terraform/language/functions/lookup

Lifecycle Policies

Terraform Resource Docs:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_lifecycle_configuration

AWS S3 Lifecycle Rules Guide:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html

Generating Pre-Signed URLs

AWS CLI Documentation:
https://docs.aws.amazon.com/cli/latest/reference/s3/presign.html

Using Pre-Signed URLs:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/ShareObjectPreSignedURL.html

Loading and Using JSON in Terraform

Terraform Function Docs - jsondecode:
https://developer.hashicorp.com/terraform/language/functions/jsondecode