Cloud Programming

TASK 1: Host a simple webpage on AWS

Phase 3: Finalisation phase

Arrehquette Ewube Eyong

IU International University

Table of Content

Introduction

The aim of this project is to host a static webpage on Amazon Web Services (AWS).

Objectives

The following are the main objectives of the project:

- High availability which makes it accessible at all times
- Scalable, adjustable to traffic levels
- Replicable and maintainable
- Globally accessible with low latency
- Secured

Solutions

The objectives are met by implementing the following tools:

- Amazon S3: It provides highly available, easily accessible, scalable and durable storage.
- Amazon CloudFront: Content Delivery Network (CDN) with edge locations used for content delivery from the nearest location to the user.
- Origin Access Identity (OAI): A security feature configured to allow only CloudFront access the S3 bucket, thereby making the website secure.
- Terraform: Infrastructure as Code (IaC) for replicability and management. It ensures the infrastructure can be recreated at anytime

Implementation

In the development of the website, there is a clear separation between

- Infrastructure configuration and
- Website Content

1. **Infrastructure Configuration**

   Includes the main.tf, variable.tf and output.tf which define resources, inputs and outputs for building the cloud environment.

   a. **Variable.tf** is used to input configurable parameters such as the AWS region and bucket name. This makes the infrastructure flexible and maintainable as it can adapt to different environments without changing core implementations.

```
variable "aws_region" {
  description ="AWS region"
  default     = "eu-north-1"
}

#Bucket name
variable "bucket_name" {
  description = " Distinct S3 bucket name"
  default = "ewube-bucket"
}
```

   b. **Output.tf** defines the values that are returned after deployment. It contains information such as the CloudFront ID which shows the unique identifier for CloudFront,

```
output "CloudFront_distribution_id" {
  description = "CloudFront ID"
  value = aws_cloudfront_distribution.cdn.id
}
```

the public URL for the website,

```
output "s3_url" {
  description = "Static website URL"
  value = aws_s3_bucket_website_configuration.static_website_bucket.website_endpoint
}
```

the CloudFront domain name outputs the secure HTTPS for the website and the CloudFront deployment status.

```
#Usable HTTPS URL to access the website
output "CloudFront_domain_name"{
  description = "CloudFront domain name"
  value = "https://${aws_cloudfront_distribution.cdn.domain_name}"
}

#Shows the current deployment state
output "CloudFront_status" {
  description = "Current deployment status of the CloudFront distribution"
  value = aws_cloudfront_distribution.cdn.status
}
```

c. **Main.tf** contains the core implementation of the architecture. It defines all the AWS resources including the S3 bucket for storage, CloudFront for global content delivery and OAI to access the S3 bucket.
The S3 bucket is configured for static website hosting,

```
resource "aws_s3_bucket_website_configuration" "static_website_bucket" {
    bucket = aws_s3_bucket.s3bucket.id

    index_document{
        suffix = var.index_main #htlm main file
    }
    error_document {
        key = var.error_main    #html error file
    }
}
```

defines the bucket's ownership controls,

```
resource "aws_s3_bucket_ownership_controls" "static_website_bucket" {
  bucket = aws_s3_bucket.s3bucket.id

  rule {
    object_ownership = "BucketOwnerPreferred"
  }
}
```

public access blocks but accessible only to CloudFront through OAI and grants read-only permission,

```
resource "aws_s3_bucket_public_access_block" "static_website_bucket" {
    bucket                  = aws_s3_bucket.s3bucket.id
    block_public_acls       = true
    block_public_policy     = true
    ignore_public_acls      = true
    restrict_public_buckets = true
}

#Allows CloudFront access the S3 bucket
resource "aws_cloudfront_origin_access_identity" "oai" {
  comment = "OAI for S3 bucket"
}

#Policies applied to the public
resource "aws_s3_bucket_policy" "public_read"{
    bucket = aws_s3_bucket.s3bucket.id
    policy = jsonencode({
        Version = "2012-10-17"
        Statement = [
            {
                Sid = "AllowCloudFrontOAI"
                Effect = "Allow"
                Principal ={
                  AWS = aws_cloudfront_origin_access_identity.oai.iam_arn
                }
                Action = "s3:GetObject"       #allows only reading
                Resource = "${aws_s3_bucket.s3bucket.arn}/*"
```

uploads website files to the S3 bucket.

```
resource "aws_s3_object" "website_files" {
  bucket = aws_s3_bucket.s3bucket.id
  for_each = fileset("AWS website/", "**/*.*")
  key    = each.value
  source = "AWS website/${each.value}"
  content_type = each.value
}

#Creates a local variable for the origin id
locals {
    s3_origin_id = var.s3_origin_id
}
```

CloudFront optimizes the website traffic flow with low latency by enabling caching. It servers the S3 static website securely over HTTPS and also redirect all traffic to HTTPS. It also allows global accessible without geo geo restrictions

```
resource "aws_cloudfront_distribution" "cdn" {
  depends_on = [
  aws_s3_bucket.s3bucket,
  aws_cloudfront_origin_access_identity.oai
  ]
  enabled             = true
  is_ipv6_enabled = true
  default_root_object = var.index_main


  default_cache_behavior {      #Defining the cache behavior
    allowed_methods = ["GET", "HEAD"]
    cached_methods = ["GET","HEAD"]
    target_origin_id       = "s3origin"
    viewer_protocol_policy = "redirect-to-https"
```

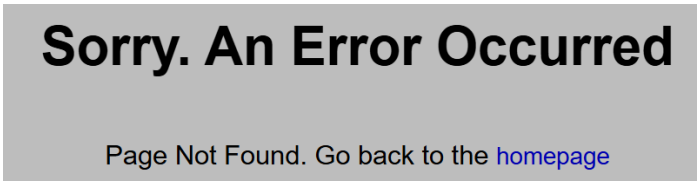
```
  cache_policy_id = "658327ea-f89d-4fab-a63d-7e88639e58f6"
}

origin {            #defines the origin of the distribution
  domain_name = aws_s3_bucket.s3bucket.bucket_regional_domain_name
  origin_id   = "s3origin"

  s3_origin_config {
    origin_access_identity =
s_cloudfront_origin_access_identity.oai.cloudfront_access_identity_path
  }
}
viewer_certificate {      #defines viewer certificate
  cloudfront_default_certificate = true
}
restrictions {           #allows access from all locations
  geo_restriction {
    restriction_type = "none"
    locations = []
```

2. **Website Content**

   This refers to the files that make up the static website and are stored in the S3 bucket. It includes the index and error html files which contained what is displayed on the browser when deployment was either successful or had an error.

   The Index html file is the main entry point of the website. It contains a simple layout used to verify that the deployment and content delivery through CloudFront are functioning properly. After deployment, the main home page will look like this:

The Error html files is the custom error page the custom error page displayed when the request resource cannot be found or access issues occurred. It also redirects the user to the main homepage (index html)

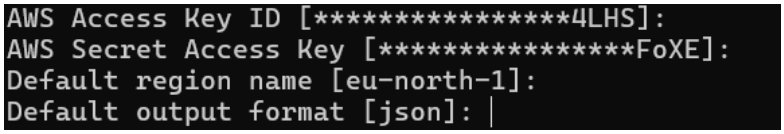


Deployment

The Infrastructure as Code used is Terraform which allows the cloud infrastructure to be defined, deployed and managed in a consistent, repeatable and reliable manner. The Amazon S3 bucket, CloudFront distribution and the OAI are defines within the terraform configuration files which ensures the infrastructure can be deployed whenever needed.

After the files' configuration and ensuring Terraform and AWS CLI are downloaded into your machine, we can start the deployment process.

- In the terminal, set up your AWS account by running "aws configure" and provide your AWS access key, AWS secret access key, default region name and output format



- Run "terraform init" to initialize the environment by downloading the necessary provider plugins and modules.
- Next, run "terraform plan". It previews the changes Terraform will make to the infrastructure.
- Finally run "terraform apply" to execute the planned changes. Terraform creates, updates or deletes resources to ensure the infrastructure matches the deployment configuration.
- At this point the infrastructure has been set up and the terminal displays the URL to access the webpage. If the files were deployed successfully, the homepage (index html) will be displayed otherwise the error page (error html) will be displayed.

Conclusion

This project demonstrates the successful deployment of a highly available and globally accessible static webpage using AWS services. It integrates Amazon S3 for storage, Amazon CloudFront for global content delivery, OAI for secure access and Terraform as IaC for automated, consistent and reusable deployment of the infrastructure.