Task 1: Deployment of a simple webpage on AWS

PHASE 2: DEVELOPMENT PHASE

Arrehquette Ewube Eyong

Cloud Programming

2026

Outline

- Introduction
- HTML
- Variable.tf
- Main.tf
- Output.tf

Introduction

This project uses Terraform to setup a highly available able and globally distributed static website on AWS. It includes four files:

- The html files which holds the index.html and error.html
- Variables.tf is where the system of variables are defined
- Main.tf defines the AWS resources
- Output.tf shows the results from the system after it has been executed

Html files

- The index html is the homepage of the website which loads the homepage that the user sees.

- The error html is displayed whenever the user tries to access a page that doesn’t exist or when the server encounters an issue. It also guides the user to the right homepage

Variable.tf

- Specifies the region where the AWS resources are located

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

- Specifies the unique S3 bucket name

  which will hold static website files

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

- Defines the home page

```
#The html error file
variable "index_main" {
  description = "The home page code name"
  default = "index.html"
}
```

- Defines the error page

```
#The html error file
variable "error_main" {
  description = "The home page code name"
  default = "error.html"
}
```

- Assigns an ID name to the S3 origin

```
#Gives the origin bucket an id
variable "s3_origin_id" {
  description = "S3 bucket origin ID"
  default     = "bucket-origin"
}
```

Main

- Configures Terraform to use the AWS provider and sets the region where all resources will be deployed

```
terraform {
    required_providers{
        aws = {
            source = "hashicorp/aws"
            version = "~> 4.16"
        }
    }
}

provider "aws" {
    region = var.aws_region
}
```

- Creates an S3 bucket where your static website

```
#Creates S3 bucket
resource "aws_s3_bucket" "s3bucket" {
    bucket = var.bucket_name
}
```

- Sets up the S3 bucket to function as a static website, specifying a homepage and an error page.

```
#Sets up the bucket 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
    }
}
```

- Sets ownership controls for the S3 bucket

```
#Defines the ownership controls
resource "aws_s3_bucket_ownership_controls" "static_website_bucket" {
  bucket = aws_s3_bucket.s3bucket.id

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

Main

- Allows public access to the S3 bucket

```
#Allows public access to the bucket
resource "aws_s3_bucket_public_access_block" "static_website_bucket" {
    bucket                  = aws_s3_bucket.s3bucket.id
    block_public_acls       = false
    block_public_policy     = false
    ignore_public_acls      = false
    restrict_public_buckets = false
}
```

- Creates a CloudFront Origin Access Identity (OAI) for the bucket. Allows CloudFront to fetch objects from the private S3 bucket securely.

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

- Grants the CloudFront OAI permission to read objects in the 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}/*"
            }]
    })
}
```

Main

- Uploads the files into the S3 bucket

```
#Uploads the files into 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
}
```

- Local variable that stores the bucket ID

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

- Creates the CloudFront distribution that will serve your website globally with low latency.

```
#Creates the CloudFront content delivery network
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
```

Main

- This block defines the default CloudFront cache behavior, specifying the origin, caching policy, protocol rules, and allowed HTTP methods

Allows only read only methods

Only GET/HEAD are cached

Tells CloudFront which origin to send request to

Redirects HHTP requests to HTTPS for secure access

Existing AWS-managed cache policy

```
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"
}
```

- The viewer certificate enables HTTPS on the CloudFront website
- Allows access from all locations

```
viewer_certificate {      #defines viewer certificate
  cloudfront_default_certificate = true
}
```

```
restrictions {           #allows access from all locations
  geo_restriction {
    restriction_type = "none"
    locations = []
  }
```

Output

- Outputs the public URL of the S3 website

```
#Gives the public URL for the S3 statis website
output "s3_url" {
  description = "Static website URL"
  value = aws_s3_bucket_website_configuration.static_website_bucket.website_endpoint
}
```

- Show the unique CloudFront distribution ID

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

- Outputs the usable HTTPS URL which users will access for the website

```
#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 of the CloudFront

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