INTERNATIONALE
HOCHSCHULE

Tutor: Georgi Dimchev
Student: Zukhrakhon Gulomova
M/N: 92118181
Submitting date:

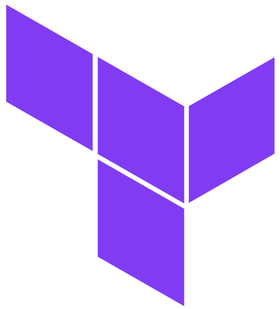
Bachelor / Artificial Intelligence

Cloud Programming
Portfolio Project (DLBSEPCP_E)

Host a simple webpage on AWS using Amazon S3 and CloudFront

Prerequisites

- An AWS account
- Terraform CLI
- Visual Studio Code

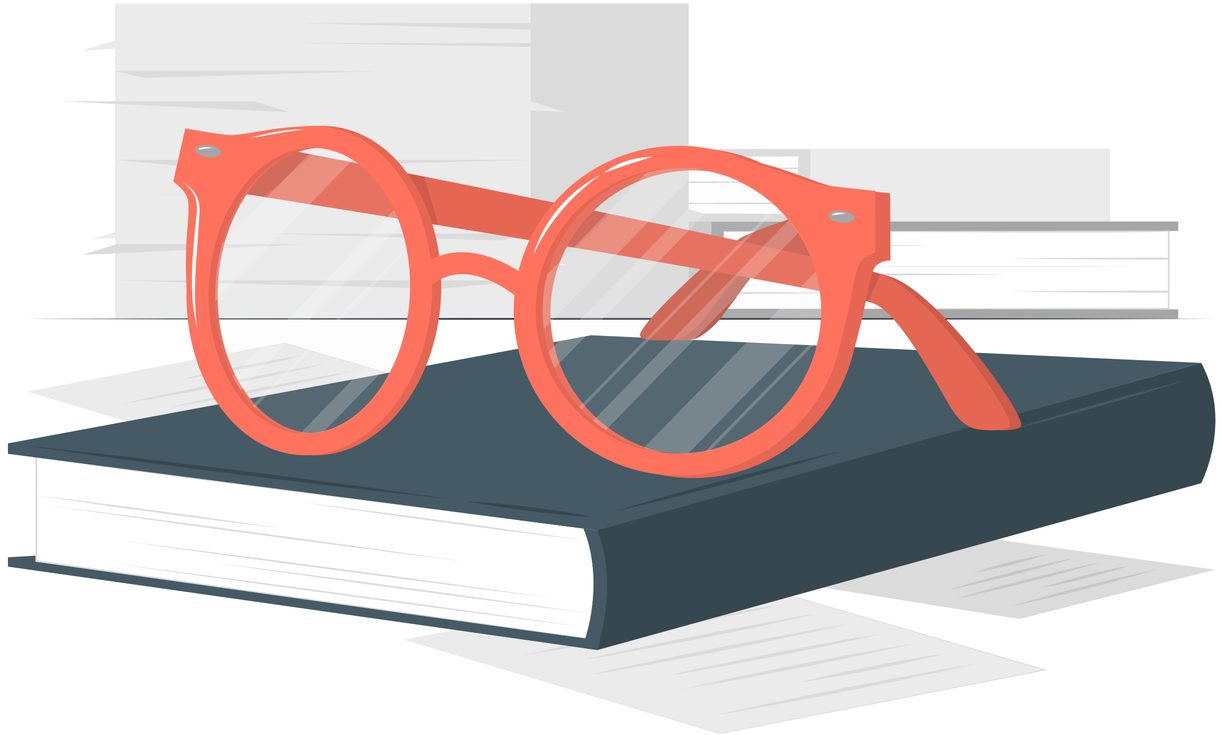


Services

- Amazon S3
- Amazon CloudFront

Table of Content

- Requirements
- AWS Provider Setup
- S3 Bucket for Static Website
- Uploading Files to S3 Bucket
- S3 Bucket Public Access
- S3 Bucket ACL
- S3 Bucket Policy
- S3 Bucket Website Configuration
- CloudFront Distribution
- Outputs



Requirements

1. **High Availability:**
    - Using Amazon S3 for static content hosting is highly available by design.
    - I've also integrated Amazon CloudFront for content delivery, which enhances availability by distributing content globally.
2. **Global Latency Avoidance:**
    - CloudFront helps in reducing latency by caching content at edge locations worldwide. This is a good approach for serving content with low latency to visitors from different geographic locations.
3. **Autoscaling for Increased Visitors:**
    - The current setup focuses on the frontend (S3 and CloudFront) and does not include any backend or server-side processing. For static websites, this is sufficient.
4. **Infrastructure as Code:**
    - Terraform script fulfills the requirement of using Infrastructure as Code.

AWS Provider Setup

providers.tf | main.tf ● | outputs.tf | datasources.tf

Project > main.tf > resource "aws_s3_bucket" "bucket1"

```
1   provider "aws" {
2     region      = "us-east-1"
3   }
4
```

S3 Bucket for Static Website

```
7   # S3 Bucket for Static Website
8   resource "aws_s3_bucket" "bucket1" {
9     bucket        = "zukhra-tf-bucket"
10    force_destroy = true
11
12    tags = {
13      Name        = "My bucket"
14      Environment = "Dev"
15    }
16  }
```

- Creating an S3 bucket for a static website
- Using the **‘aws_s3_bucket’** resource
- Setting tags for better organization

Uploading Files to S3 Bucket

```
18  # Uploading files to S3 Bucket
19  resource "aws_s3_object" "files" {
20    bucket       = aws_s3_bucket.bucket1.id
21    for_each     = fileset("website/", "**/*.*")
22    key          = each.value
23    source       = "website/${each.value}"
24    content_type = each.value
25  }
26
```

- Using '**aws_s3_object**' resource to upload files to the S3 bucket
- Utilizing '**fileset**' function for dynamic file uploads

S3 Bucket Public Access

```
27  # S3 Bucket Public Access
28  resource "aws_s3_bucket_ownership_controls" "ownership" {
29    bucket = aws_s3_bucket.bucket1.id
30    rule {
31      object_ownership = "BucketOwnerPreferred"
32    }
33  }
34
35  resource "aws_s3_bucket_public_access_block" "public_access_block" {
36    bucket = aws_s3_bucket.bucket1.id
37
38    block_public_acls       = false
39    block_public_policy     = false
40    ignore_public_acls      = false
41    restrict_public_buckets = false
42  }
43
```

- Managing public access to the S3 bucket
- Configuring ownership controls and public access block

S3 Bucket ACL

```
44  resource "aws_s3_bucket_acl" "s3_bucket_acl" {
45    bucket = aws_s3_bucket.bucket1.id
46    acl    = "public-read"
47
48    depends_on = [
49      aws_s3_bucket_ownership_controls.ownership,
50      aws_s3_bucket_public_access_block.public_access_block,
51    ]
52
53  }
54
```

- Setting Access Control List (ACL) for the S3 bucket
- Allowing public read access

S3 Bucket Policy

```
55  # S3 Bucket Policy
56  resource "aws_s3_bucket_policy" "bucket_policy" {
57    bucket = aws_s3_bucket.bucket1.bucket
58    policy = jsonencode(
59      {
60        "Version" : "2012-10-17",
61        "Statement" : [
62          {
63            "Sid" : "PublicReadGetObject",
64            "Effect" : "Allow",
65            "Principal" : "*",
66            "Action" : "s3:GetObject",
67            "Resource" : "${aws_s3_bucket.bucket1.arn}/*"
68          }
69        ]
70      }
71    )
72  }
73
```

- Defining a policy for the S3 bucket
- Allowing public read access to objects in the bucket

S3 Bucket Website Configuration

```
74  # S3 bucket website configuration
75  resource "aws_s3_bucket_website_configuration" "bucket_website_configuration" {
76    bucket = aws_s3_bucket.bucket1.id
77
78    index_document {
79      suffix = "index.html"
80    }
81  }
82
```

- Configuring the S3 bucket for static website hosting
- Specifying index document (e.g., index.html)

CloudFront Distribution

```
# CloudFront Distribution
locals {
  s3_origin_id = "myS3Origin"
}

resource "aws_cloudfront_distribution" "distribution" {
  enabled             = true
  is_ipv6_enabled     = true
  default_root_object = "index.html"

  origin {
    domain_name = aws_s3_bucket.bucket1.bucket_regional_domain_name
    origin_id   = local.s3_origin_id
  }

  viewer_certificate {
    cloudfront_default_certificate = true
  }

  restrictions {
    geo_restriction {
      restriction_type = "none"
      locations        = []
    }
  }

  default_cache_behavior {
    cache_policy_id        = "4135ea2d-6df8-44a3-9df3-4b5a84be39ad"
    viewer_protocol_policy = "redirect-to-https"
    allowed_methods        = ["DELETE", "GET", "HEAD", "OPTIONS", "PATCH", "POST", "PUT"]
    cached_methods         = ["GET", "HEAD"]
    target_origin_id       = local.s3_origin_id
  }
}
```

- Configuring default cache behavior and SSL certificate
- Leveraging caching strategies with cache policies
- Enforcing HTTPS with ‘**redirect-to-https**’ policy
- Specifying allowed and cached HTTP methods

Outputs

```
118  # Outputs
119  output "website_url" {
120    description = "Website URL (HTTPS)"
121    value       = aws_cloudfront_distribution.distribution.domain_name
122  }
123
124  output "s3_url" {
125    description = "S3 hosting URL (HTTP)"
126    value       = aws_s3_bucket_website_configuration.bucket_website_configuration.website_endpoint
127  }
128
```

- Website URL
- S3 URL