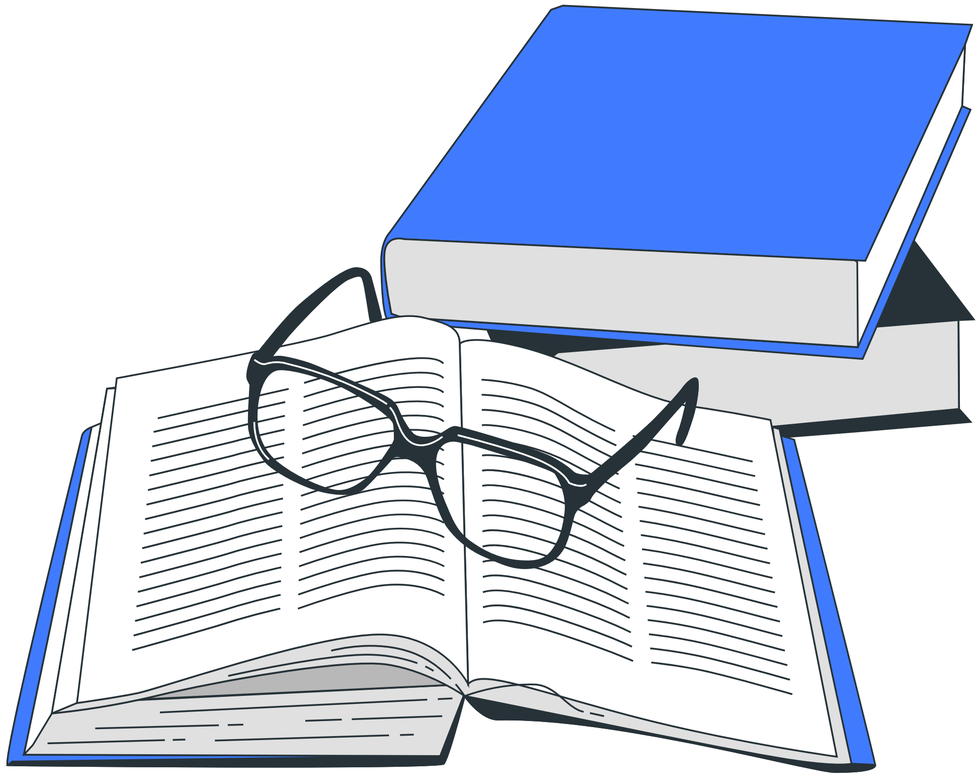
Task 1: Host a simple web-page on AWS

01

Table of Contents



Description Of The Task

To ensure a highly available and low-latency website on AWS, I set up a basic 'index.html' webpage using **Amazon S3** for storage and **AWS CloudFront** for global content delivery.

Amazon S3 serves as storage for the webpage's static content like HTML, CSS, and images. I created an S3 bucket and upload the basic HTML file there.

To distribute the webpage globally I used **AWS CloudFront**, a content delivery network. It caches content across various locations worldwide, minimizing latency for users from different regions.

Infrastructure as Code (IaC) tools like **Terraform** streamline the setup and management of AWS services.

AWS Provider

Set up AWS credentials

- **AWS Access Key:** Identify an account
- **AWS Secret Key:** The corresponding secret key pairs with an access key

Set an AWS access key and secret key as an environmental variables

- setx AWS_ACCESS_KEY_ID "YOUR_ACCESS_KEY"
- setx AWS_SECRET_ACCESS_KEY "YOUR_SECRET_KEY"

AWS Provider

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

S3 Bucket

aws_s3_bucket

- Create an S3 bucket resource named **s3bucket**
- Set its name as specified in the variable **var.bucket_name**
- Assign tags to the bucket in the variable **var.bucket_tags**

```
# Step 2: Create an S3 Bucket for Website Hosting
resource "aws_s3_bucket" "s3bucket" {
  bucket = var.bucket_name

  tags = var.bucket_tags
}
```

S3 Bucket Configuration

aws_s3_bucket_website_configuration

- Create an S3 bucket configuration named **static_website_bucket**
- Use the ID of an existing S3 bucket **aws_s3_bucket.s3bucket.id**
- Set the suffix (filename) of the default document in the variable **var.index_document_suffix**

```
# Step 3: Set up the S3 Bucket Configuration
resource "aws_s3_bucket_website_configuration" "static_website_bucket" {
  bucket = aws_s3_bucket.s3bucket.id

  index_document {
    suffix = var.index_document_suffix
  }
}
```

aws_s3_bucket_ownership_controls

- Create an S3 bucket configuration named **static_website_bucket**
- Use the ID of an existing S3 bucket **aws_s3_bucket.s3bucket.id**
- **rule:** This block defines a rule within the ownership controls configuration.
- Set the object ownership configuration for the bucket in the variable **var.object_ownership**

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

  rule {
    object_ownership = var.object_ownership
  }
}
```

aws_s3_bucket_public_access_block

- Create an S3 bucket configuration named **static_website_bucket**
- Use the ID of an existing S3 bucket **aws_s3_bucket.s3bucket.id**
- **block_public_acls, block_public_policy, ignore_public_acls, restrict_public_buckets:** Settings for blocking public access to S3 bucket
- Set all settings **false**, meaning public access is not blocked

```
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
}
```

aws_s3_bucket_acl

- Create an ACL configuration named **static_website_bucket**
- **depends_on:** This block specifies dependencies for this resource.The ACL configuration will only be applied after the specified resources
- Use the ID of an existing S3 bucket **aws_s3_bucket.s3bucket.id**
- Set the ACL (Access Control List) for the S3 bucket in the variable **var.acl_setting**

```
resource "aws_s3_bucket_acl" "static_website_bucket" {
  depends_on = [
    aws_s3_bucket_ownership_controls.static_website_bucket,
    aws_s3_bucket_public_access_block.static_website_bucket,
  ]

  bucket = aws_s3_bucket.s3bucket.id
  acl    = var.acl_setting
}
```

S3 Bucket Policy

```
# Step 4: Set the S3 Bucket Policy
resource "aws_s3_bucket_policy" "s3_bucket_policy" {
  bucket = aws_s3_bucket.s3bucket.bucket

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Sid       = "PublicReadGetObject"
        Effect    = "Allow"
        Principal = "*"
        Action    = "s3:GetObject"
        Resource = [
          aws_s3_bucket.s3bucket.arn,
          "${aws_s3_bucket.s3bucket.arn}/*"
        ]
      }
    ]
  })
}
```

aws_s3_bucket_acl

- Create an S3 Bucket Policy named **s3_bucket_policy**
- Use the ID of an existing S3 bucket **aws_s3_bucket.s3bucket.id**
- **policy = jsonencode({ ... }):** This section defines the policy using JSON formatting and the jsonencode function.
- Set the policy to allow public read access **("Effect": "Allow")**

Upload Website Files

aws_s3_object

- Create an S3 objects within the bucket named **website_files**
- Use the ID of an existing S3 bucket **aws_s3_bucket.s3bucket.id**
- **for_each = fileset("website files/", "**/*.*"):** The fileset function moves through the local directory called **"website files/"** and includes all files **(**/*.*)** within this directory.
- The **each.value** refers to the current filename being processed from the for_each set.
- **source = "website files/${each.value}":** Specifies the source of the file to be uploaded.
- **content_type = each.value:** Sets the content type of the file being uploaded in S3.

```
# Step 5: Upload Website Files to the S3 Bucket
resource "aws_s3_object" "website_files" {
  bucket       = aws_s3_bucket.s3bucket.id
  for_each     = fileset("website files/", "**/*.*")
  key          = each.value
  source       = "website files/${each.value}"
  content_type = each.value
}
```

Cloud Distribution

```
# Step 6: Set up a CloudFront Distribution
locals {
  s3_origin_id = var.s3_origin_id
}

resource "aws_cloudfront_distribution" "s3_distribution" {
  depends_on = [aws_s3_bucket.s3bucket]

  origin {
    domain_name = aws_s3_bucket.s3bucket.bucket_regional_domain_name
    origin_id   = local.s3_origin_id
  }

  enabled             = true
  is_ipv6_enabled     = var.ipv6_enabled
  default_root_object = var.default_root_object

  viewer_certificate {
    cloudfront_default_certificate = true
  }
```

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

  default_cache_behavior {
    cache_policy_id        = var.cache_behavior.cache_policy_id
    viewer_protocol_policy = var.cache_behavior.viewer_protocol_policy
    allowed_methods        = var.cache_behavior.allowed_methods
    cached_methods         = var.cache_behavior.cached_methods
    target_origin_id       = local.s3_origin_id
  }
}
```

aws_cloudfront_distribution

- **locals:** Defines a local value called **s3_origin_id** using the variable **var.s3_origin_id.**
- Create an S3 objects within the bucket named **s3_distribution**
- Use the ID of an existing S3 bucket **aws_s3_bucket.s3bucket.id**
- **depends_on = [aws_s3_bucket.s3bucket]:** This CloudFront distribution resource depends on the creation of the S3 bucket
- **origin { ... }:** The origin settings for the CloudFront distribution. It uses the S3 bucket as the origin by specifying the S3 bucket's regional domain name and the **origin_id** from the local value.
- **enabled = true:** The CloudFront distribution is enabled.
- IPv6 is enabled for the CloudFront distribution, using the value from the variable **var.ipv6_enabled**.
- Specifies the default root object **"index.html"** for the distribution from the variable **var.default_root_object**.
- **viewer_certificate { ... }:** Configures the viewer's SSL/TLS certificate.
- **restrictions { ... }:** Configures restrictions on the distribution.
- **default_cache_behavior { ... }:** Defines the default cache behavior for the CloudFront distribution.

Output

```
# Step 7: Get the Outputs
output "cloudfront_id" {
  description = "Cloudfront ID"
  value       = aws_cloudfront_distribution.s3_distribution.id
}

output "cloudfront_url" {
  description = "Cloudfront distribution URL (HTTPS)"
  value       = "https://${aws_cloudfront_distribution.s3_distribution.domain_name}"
}

output "s3_url" {
  description = "S3 hosting URL"
  value       = aws_s3_bucket_website_configuration.static_website_bucket.website_endpoint
}
```

cloudfront_id

- output **"cloudfront_id":** Specifies the CloudFront distribution ID.
- The **aws_cloudfront_distribution.s3_distribution.id** resource represents the ID of the CloudFront distribution.

cloudfront_url

- output **"cloudfront_url":** Specifies the CloudFront distribution URL.
- The **aws_cloudfront_distribution.s3_distribution.domain_name** resource provides the CloudFront distribution URL as an HTTPS URL.

s3_url

- output **"s3_url":** Specifies the CloudFront distribution ID.
- The **aws_s3_bucket_website_configuration.static_website_bucket.website_endpoint** resource represents the endpoint URL where the S3 bucket content is hosted as a static website.