**INFRASTRUCTURE :**

**Infrastructure as Code (IaC):** Utilize a tool like AWS CloudFormation or Terraform to define the infrastructure as code. This allows for consistent provisioning and easy replication of the environment.

**Static Website Hosting**: Use Amazon S3 to host the static website content. S3 provides high availability, scalability, and low latency for serving static content.

**SSL/TLS Certificate**: Obtain an SSL/TLS certificate to enable HTTPS. This can be done using AWS Certificate Manager (ACM) for free SSL certificates that are automatically managed

**Security Group**: Configure security groups to restrict access to only necessary ports (e.g., HTTP, HTTPS) and IP ranges. This helps enforce network security and prevent unauthorized access.

**Automated Deployment**: Use AWS CodePipeline and AWS CodeDeploy to automate the deployment process. This ensures consistent and reliable deployments with built-in rollback mechanisms.

**Monitoring and Logging**: Set up Amazon CloudWatch alarms and logging to monitor the health and performance of the web application. This includes metrics such as latency, error rates, and traffic patterns.

TERRAFORM CONFIGURATION:

provider "aws" {

region = "us-east-1"

}

# Create an S3 bucket for website hosting

resource "aws\_s3\_bucket" "static\_website\_bucket" {

bucket = "my-static-website-bucket"

acl = "public-read"

website {

index\_document = "index.html"

}

# Enable versioning

versioning {

enabled = true

}

}

# Create an ACM certificate

resource "aws\_acm\_certificate" "example" {

domain\_name = "example.com"

validation\_method = "DNS"

tags = {

Environment = "Production"

}

}

# Create CloudFront distribution

resource "aws\_cloudfront\_distribution" "static\_website\_distribution" {

origin {

domain\_name = aws\_s3\_bucket.static\_website\_bucket.bucket\_regional\_domain\_name

origin\_id = "S3Origin"

s3\_origin\_config {

origin\_access\_identity = ""

}

}

enabled = true

is\_ipv6\_enabled = true

default\_root\_object = "index.html"

aliases = ["example.com"]

default\_cache\_behavior {

allowed\_methods = ["DELETE", "GET", "HEAD", "OPTIONS", "PATCH", "POST", "PUT"]

cached\_methods = ["GET", "HEAD"]

target\_origin\_id = "S3Origin"

forwarded\_values {

query\_string = false

cookies {

forward = "none"

}

}

viewer\_protocol\_policy = "redirect-to-https"

min\_ttl = 0

default\_ttl = 3600

max\_ttl = 86400

}

restrictions {

geo\_restriction {

restriction\_type = "none"

}

}

viewer\_certificate {

acm\_certificate\_arn = aws\_acm\_certificate.example.arn

ssl\_support\_method = "sni-only"

}

}

# Security Group for Web Server

resource "aws\_security\_group" "web\_sg" {

name = "web\_sg"

description = "Allow HTTP and HTTPS inbound traffic"

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "web\_sg"

}

}

**DEPLOYMENT SCRIPT (deploy.sh)**

1. #!/bin/bash
2. terraform init
3. terraform apply -auto-approve
4. echo "Validating deployment..."
5. curl -I http://example.com 2>/dev/null | grep "HTTP/1.1 301 Moved Permanently"
6. curl -I https://example.com 2>/dev/null | grep "HTTP/2 200"
7. echo "Deployment and validation complete!"

**EXECUTION:**

* terraform init
* chmod +x deploy.sh