Main.tf file

# Configure the AWS Provider and required providers

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 4.0"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

#----------------------------------------

# S3 Bucket Configuration

#----------------------------------------

# Create the main S3 bucket

resource "aws\_s3\_bucket" "website" {

  bucket        = var.bucket\_name

  force\_destroy = true

}

# Enable versioning for backup and recovery

resource "aws\_s3\_bucket\_versioning" "website" {

  bucket = aws\_s3\_bucket.website.id

  versioning\_configuration {

    status = "Enabled"

  }

}

# Configure the bucket for static website hosting

resource "aws\_s3\_bucket\_website\_configuration" "website" {

  bucket = aws\_s3\_bucket.website.id

  index\_document {

    suffix = "index.html"

  }

  error\_document {

    key = "error.html"

  }

}

# Enable server-side encryption for security

resource "aws\_s3\_bucket\_server\_side\_encryption\_configuration" "website" {

  bucket = aws\_s3\_bucket.website.id

  rule {

    apply\_server\_side\_encryption\_by\_default {

      sse\_algorithm = "AES256"

    }

  }

}

#----------------------------------------

# CloudFront Configuration

#----------------------------------------

# Create Origin Access Identity for CloudFront

resource "aws\_cloudfront\_origin\_access\_identity" "website" {

  comment = "OAI for ${var.bucket\_name} website"

}

# Configure bucket policy to allow CloudFront access

resource "aws\_s3\_bucket\_policy" "website" {

  bucket = aws\_s3\_bucket.website.id

  policy = jsonencode({

    Version = "2012-10-17"

    Statement = [

      {

        Sid    = "PublicReadGetObject"

        Effect = "Allow"

        Principal = {

          AWS = aws\_cloudfront\_origin\_access\_identity.website.iam\_arn

        }

        Action   = "s3:GetObject"

        Resource = "${aws\_s3\_bucket.website.arn}/\*"

      }

    ]

  })

}

# Set bucket ownership controls

resource "aws\_s3\_bucket\_ownership\_controls" "website" {

  bucket = aws\_s3\_bucket.website.id

  rule {

    object\_ownership = "BucketOwnerPreferred"

  }

}

# Block all public access - content will be served through CloudFront

resource "aws\_s3\_bucket\_public\_access\_block" "website" {

  bucket = aws\_s3\_bucket.website.id

  block\_public\_acls       = false

  block\_public\_policy     = false

  ignore\_public\_acls      = false

  restrict\_public\_buckets = false

}

# Create CloudFront distribution

resource "aws\_cloudfront\_distribution" "website" {

  enabled             = true

  is\_ipv6\_enabled     = true

  default\_root\_object = "index.html"

  price\_class         = "PriceClass\_100" # Use only North America and Europe endpoints

  # Origin configuration for S3

  origin {

    domain\_name = aws\_s3\_bucket.website.bucket\_regional\_domain\_name

    origin\_id   = "S3Origin"

    s3\_origin\_config {

      origin\_access\_identity = aws\_cloudfront\_origin\_access\_identity.website.cloudfront\_access\_identity\_path

    }

  }

  # Default cache behavior for most content

  default\_cache\_behavior {

    allowed\_methods  = ["GET", "HEAD", "OPTIONS"]

    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  # 1 hour

    max\_ttl                = 86400 # 24 hours

    compress               = true

  }

  # Specific cache behavior for PDF files

  ordered\_cache\_behavior {

    path\_pattern     = "assets/pdfs/\*"

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

    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            = 86400   # 24 hours

    max\_ttl                = 2592000 # 30 days

    compress               = true

  }

  # Geographic restrictions

  restrictions {

    geo\_restriction {

      restriction\_type = "none"

    }

  }

  # SSL/TLS certificate configuration

  viewer\_certificate {

    cloudfront\_default\_certificate = true

  }

  # Custom error response configuration

  custom\_error\_response {

    error\_code         = 404

    response\_code      = 404

    response\_page\_path = "/error.html"

  }

}

#----------------------------------------

# File Upload Configuration

#----------------------------------------

# Resource to handle website file uploads

resource "null\_resource" "upload\_files" {

  triggers = {

    always\_run = "${timestamp()}"

  }

  provisioner "local-exec" {

    interpreter = ["PowerShell", "-Command"]

    command     = <<-EOT

      if (aws s3 sync ./website/ s3://${aws\_s3\_bucket.website.id} --delete) {

        Write-Host "Website files uploaded successfully"

        aws cloudfront create-invalidation --distribution-id ${aws\_cloudfront\_distribution.website.id} --paths "/\*"

      } else {

        Write-Host "Failed to upload website files"

        exit 1

      }

    EOT

  }

  depends\_on = [

    aws\_s3\_bucket.website,

    aws\_cloudfront\_distribution.website,

    aws\_s3\_bucket\_policy.website

  ]

}

Outputs.tf file

output "cloudfront\_domain\_name" {

  description = "Domain name of the CloudFront distribution"

  value       = aws\_cloudfront\_distribution.website.domain\_name

}

output "cloudfront\_distribution\_id" {

  description = "ID of the CloudFront distribution"

  value       = aws\_cloudfront\_distribution.website.id

}

output "s3\_bucket\_name" {

  description = "Name of the S3 bucket"

  value       = aws\_s3\_bucket.website.id

}

output "s3\_bucket\_arn" {

  description = "ARN of the S3 bucket"

  value       = aws\_s3\_bucket.website.arn

}

Variables.tf file

variable "aws\_region" {

  description = "AWS region"

  type        = string

  default     = "us-east-2"

}

variable "bucket\_name" {

  description = "my-website-bucket"

  type        = string

}

Terraform.tfvars