Terraform Static Website Deployment on AWS (S3 + CloudFront)

This project uses Terraform to provision AWS infrastructure for deploying a static website. It leverages Amazon S3 for hosting and CloudFront for fast, secure content delivery. Deployment is automated using GitHub Actions for continuous integration and delivery.

Project Overview

This repository allows you to:

- Host static websites using AWS S3
- Securely distribute your content globally via AWS CloudFront
- Automate infrastructure provisioning using Terraform
- Automate deployments using GitHub Actions

Tech Stack

- **Terraform** – Infrastructure as Code
- **AWS S3** – Static Website Hosting
- **AWS CloudFront** – Content Delivery Network (CDN)
- **AWS ACM** – For enabling HTTPS (optional, for custom domain support)
- **GitHub Actions** – CI/CD Pipeline for Infrastructure Deployment

Project Structure

```
Terrsform-Site/
├── .github/workflows/
│   └── terraform.yml       # GitHub Actions workflow for Terraform
├── main.tf                 # Terraform resources for S3 and CloudFront
├── variables.tf            # Input variables for customization
├── outputs.tf              # Outputs like CloudFront URL
├── providers.tf            # AWS provider config
└── README.md               # This file
```

Features

- Automated infrastructure provisioning via Terraform
- Fast content delivery with CloudFront
- Optional HTTPS support with AWS ACM
- GitHub Actions for auto-deploying Terraform configurations
- Modular and scalable IaC setup

Prerequisites

- Terraform installed (Install Guide)
- AWS CLI configured with IAM credentials: `aws configure`
- GitHub repository secrets set:
    - `AWS_ACCESS_KEY_ID`

    - `AWS_SECRET_ACCESS_KEY`
    - Optionally: `AWS_REGION`

□ How to Use

1. **Fork & Clone the Repo**

   ```
   git clone https://github.com/YogiHarshil/Terrsform-Site.git
   cd Terrsform-Site
   ```

2. **Customize Variables**

   Edit `variables.tf` or add a `terraform.tfvars` file for:

   ```
   bucket_name   = "your-unique-s3-bucket-name"
   region        = "us-east-1"
   website_index = "index.html"
   ```

3. **Set Up GitHub Secrets**

   Go to your GitHub repo → **Settings** → **Secrets** → **Actions**, and add:

    - `AWS_ACCESS_KEY_ID`
    - `AWS_SECRET_ACCESS_KEY`
    - `AWS_REGION`

4. **Push Changes to Trigger CI/CD**

   Any push to `main` (or your specified branch) will trigger the GitHub Action to:

    - Initialize Terraform
    - Validate & Plan the infra
    - Apply the changes on AWS

5. **Upload Website Files to S3**

   After provisioning, upload your website files (like `index.html`, `styles.css`) to the specified S3 bucket via AWS CLI or Console.

□ Outputs

After successful deployment, Terraform will output:

- □ S3 Bucket Name
- □ CloudFront Distribution Domain

□ Clean Up

To tear down the infrastructure:

```
terraform destroy
```

Or trigger a "destroy" step via GitHub Actions (if implemented).

□ GitHub Actions: terraform.yml

Here’s a summary of what the workflow does:

- Runs on push to `main`
- Sets up Terraform and AWS credentials
- Runs `terraform init`, `plan`, and `apply`
- Automates infrastructure deployment

□ License

MIT License

□ Author

Harshil Yogi

GitHub: @YogiHarshil