Simple Tutorial: Using Terraform in GitHub Actions

Prerequisites

1. Terraform installed locally (for testing your configuration before committing). 2. AWS Account with IAM credentials. 3. GitHub Repository to hold your Terraform code. 4. GitHub Actions enabled in your repo.

1. Create Terraform Configuration

Inside your repo, create a folder terraform/ and a file main.tf:

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

resource "aws_s3_bucket" "demo_bucket" {
  bucket = "my-demo-terraform-gha-bucket"
  acl    = "private"
}
```

2. Add AWS Credentials to GitHub Secrets

1. In your GitHub repo → Settings → Secrets and variables → Actions → New repository secret. 2. Add: - AWS_ACCESS_KEY_ID - AWS_SECRET_ACCESS_KEY

3. Create GitHub Actions Workflow

Inside your repo, create: .github/workflows/terraform.yml

```
name: Terraform CI

on:
  push:
    branches: [ "main" ]

jobs:
  terraform:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v3

      - name: Terraform Init
        run: terraform init
        working-directory: ./terraform

      - name: Terraform Validate
        run: terraform validate
        working-directory: ./terraform

      - name: Terraform Plan
        run: terraform plan
        working-directory: ./terraform
        env:
```

```
      AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
      AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}

  - name: Terraform Apply
    if: github.ref == 'refs/heads/main'
    run: terraform apply -auto-approve
    working-directory: ./terraform
    env:
      AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
      AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
```

4. Run the Workflow

Commit & push your repo. Go to GitHub → Actions tab. You'll see your Terraform job running.

5. Clean Up

When done, you can remove resources:
`terraform destroy`