Using GitHub Actions for Multi-Environment Deployments

Using GitHub Actions for Terraform-based multi-environment deployments allows teams to manage and automate infrastructure changes across development, staging, and production environments in a clean, modular way.

# Goal

Deploy to multiple environments (dev, staging, prod) using:  
- Separate Terraform variable files  
- GitHub Actions workflow inputs  
- Terraform commands with conditionals

# Example Directory Structure

gke-cluster/  
├── main.tf  
├── variables.tf  
├── dev.tfvars  
├── staging.tfvars  
├── prod.tfvars

# 🧠 Key Concepts

|  |  |
| --- | --- |
| Feature | Usage |
| workflow\_dispatch | Allows manual environment selection |
| GitHub environments | Isolate secrets and protect production |
| terraform apply -var-file | Apply configuration based on selected environment |

# GitHub Actions Workflow

Below is a complete workflow to support environment-based deployments using inputs:

name: Terraform GKE Multi-Env Deploy  
  
on:  
 workflow\_dispatch:  
 inputs:  
 environment:  
 description: 'Select Environment'  
 required: true  
 default: 'dev'  
 type: choice  
 options:  
 - dev  
 - staging  
 - prod  
 action:  
 description: 'Terraform Action'  
 required: true  
 default: 'plan'  
 type: choice  
 options:  
 - plan  
 - apply  
 - destroy  
  
jobs:  
 terraform:  
 name: Deploy to ${{ github.event.inputs.environment }}  
 runs-on: ubuntu-latest  
  
 defaults:  
 run:  
 working-directory: gke-cluster  
  
 environment: ${{ github.event.inputs.environment }}  
  
 steps:  
 - name: Checkout Repo  
 uses: actions/checkout@v4  
  
 - name: Set up Terraform  
 uses: hashicorp/setup-terraform@v3  
 with:  
 terraform\_version: 1.6.6  
  
 - name: Decode GCP credentials  
 run: echo "${{ secrets.GCP\_CREDENTIALS }}" | base64 --decode > "${HOME}/gcp-key.json"  
  
 - name: Set GOOGLE\_APPLICATION\_CREDENTIALS  
 run: echo "GOOGLE\_APPLICATION\_CREDENTIALS=${HOME}/gcp-key.json" >> $GITHUB\_ENV  
  
 - name: Authenticate to GCP  
 uses: google-github-actions/auth@v2  
 with:  
 credentials\_json: ${{ secrets.GCP\_CREDENTIALS }}  
  
 - name: Terraform Init  
 run: terraform init  
  
 - name: Terraform Validate  
 run: terraform validate  
  
 - name: Terraform Format Check  
 run: terraform fmt -check  
  
 - name: Terraform Plan  
 run: terraform plan -var-file=${{ github.event.inputs.environment }}.tfvars -input=false  
  
 - name: Terrascan Security Check  
 uses: accurics/terrascan-action@v1.3.0  
 with:  
 iac\_type: "terraform"  
 iac\_version: "v14"  
 policy\_type: "gcp"  
 iac\_dir: "gke-cluster"  
  
 - name: Terraform Apply or Destroy  
 if: github.event.inputs.action != 'plan'  
 run: |  
 if [[ "${{ github.event.inputs.action }}" == "apply" ]]; then  
 terraform apply -auto-approve -input=false -var-file=${{ github.event.inputs.environment }}.tfvars  
 elif [[ "${{ github.event.inputs.action }}" == "destroy" ]]; then  
 terraform destroy -auto-approve -input=false -var-file=${{ github.event.inputs.environment }}.tfvars

# 🔒 Secret Management Tips

When managing secrets in GitHub Actions for multi-environment deployments, follow these practices:  
- Use GitHub Environments to scope secrets (e.g., dev, staging, prod).  
- Store credentials like `GCP\_CREDENTIALS` in encrypted GitHub secrets.  
- Use `GITHUB\_ENV`, not hardcoded values, to access credentials securely.  
- Rotate secrets regularly and restrict access to sensitive environments.

# Multi-Environment Deployment Flow (Diagram)

The diagram below illustrates how GitHub Actions manages CI/CD for multiple environments.

