Terraform Example for CI/CD with GitHub Actions

1. Login to GCP: Authenticate with Google Cloud

D:\Terraform-Training\terraform\_1.9.2\_windows\_amd64\GCP\demo-1>gcloud auth application-default login

1. Verify the Authentication

D:\Terraform-Training\terraform\_1.9.2\_windows\_amd64\GCP\demo-1>gcloud auth application-default print-access-token

ya29.a0AcM612xih9TwIRzezhSuMfpVmEAu6JNiykRQOj0cn2YS\_T8YeeZPh19KmKIZMQwLt9MdZUS5vngVboN\_vQxQctwTGv7hmWrxI9MymcXB2kEfzSolqYpeFxADTmASrplz52RbkXNBUyoYB2OPSLS0R3DtlTSdv6l\_HoaaaCgYKAbISARASFQHGX2Mi7NgFnhdnFxlGmj9Q-mRBNw0171

**Step 1: Setting Up Terraform Configuration**

First, ensure you have your Terraform configuration files set up. For instance, you might have a file main.tf to define your infrastructure:

**provider "google" {**

**project = "projectkk-terraform"**

**region = "us-central1"**

**}**

**resource "google\_compute\_instance" "vm\_instance" {**

**name = "my-instance"**

**machine\_type = "n1-standard-1"**

**zone = "us-central1-a"**

**boot\_disk {**

**initialize\_params {**

**image = "** **debian-cloud/debian-10-buster-v20210817"**

**}**

**}**

**network\_interface {**

**network = "default"**

**access\_config {}**

**}**

**}**

A screenshot of a computer error

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Step 2: Complete Terraform Setup with CI/CD**

**To run the .github/workflows/terraform.yml workflow in GitHub Actions, you need to follow these steps:**

**Step-by-Step Guide**

1. **Add the Workflow File: Ensure the .github/workflows/terraform.yml file is present in your repository.**
2. **Create a Service Account and Generate a Key:**
   * **Go to the Google Cloud Console.**
   * **Navigate to IAM & Admin > Service Accounts.**
   * **Create a new service account or use an existing one.**
   * **Grant the necessary permissions to the service account (e.g., roles like Editor or specific roles like Compute Admin).**
   * **Create a JSON key for this service account and download it.**
3. **Add the Service Account Key to GitHub Secrets:**
   * **Go to your GitHub repository.**
   * **Navigate to Settings > Secrets and variables > Actions.**
   * **Click on New repository secret.**
   * **Add a secret with the name GCP\_CREDENTIALS and paste the content of the JSON key file.**
4. **Push Code to GitHub:**
   * **Commit and push your Terraform configuration and the workflow file to your repository. The workflow will automatically trigger on the specified events (e.g., push or pull request to the main branch).**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Terraform.yml**

**name: 'Terraform CI/CD'**

**on:  
 push:  
 branches:  
 - main  
 pull\_request:  
 branches:  
 - main**

**jobs:  
 terraform:  
 name: 'Terraform'  
 runs-on: ubuntu-latest**

**steps:  
 - name: 'Checkout GitHub Action'  
 uses: actions/checkout@v2**

**- name: 'Setup Terraform'  
 uses: hashicorp/setup-terraform@v1  
 with:  
 terraform\_version: 1.0.0**

**- name: 'Authenticate to Google Cloud'  
 run: echo "${{ secrets.GCP\_CREDENTIALS }}" > $HOME/gcloud.json**

**- name: 'Setup gcloud CLI'  
 uses: google-github-actions/setup-gcloud@v0.3.0  
 with:  
 service\_account\_key: ${{ secrets.GCP\_CREDENTIALS }}  
 export\_default\_credentials: true**

**- name: 'Terraform Init'  
 run: terraform init**

**- name: 'Terraform Plan'  
 run: terraform plan**

**- name: 'Terraform Apply'  
 if: github.ref == 'refs/heads/main'  
 run: terraform apply -auto-approve**

**Steps to Follow**

1. **Create/Update the .github/workflows/terraform.yml File:**
   * **Follow the steps mentioned in the previous response to create or update the .github/workflows/terraform.yml file in your repository.**
2. **Store Secrets in GitHub:**
   * **Ensure your Google Cloud service account key is stored securely in GitHub Secrets as GCP\_CREDENTIALS. To add a secret:**
     + **Go to your GitHub repository.**
     + **Click on Settings.**
     + **Click on Secrets and variables > Actions.**
     + **Click on New repository secret.**
     + **Add GCP\_CREDENTIALS as the name and paste your service account JSON key as the value.**
3. **Commit and Push Changes:**
   * **If you're using the Git CLI, commit and push the changes to your repository.**

**git add .github/workflows/terraform.yml  
git commit -m "Add Terraform GitHub Actions workflow"  
git push origin main**

1. **Check Workflow Runs:**
   * **Once the workflow file is added, GitHub Actions will automatically pick it up and run the specified jobs on push and pull request events for the main branch. You can check the status of the workflow runs in the Actions tab of your repository on GitHub.**

**By following these steps, you should be able to resolve the issue and have a working Terraform CI/CD pipeline in your GitHub repository.**

**cd D:\GCP-Terraform\First-Demo  
git init  
git add .github/workflows/terraform.yml  
git commit -m "Add Terraform GitHub Actions workflow"  
git remote add origin your-remote-repository-url  
git push origin main**

**tep-by-Step Guide**

1. **Verify File Existence Ensure that the file .github/workflows/terraform.yml exists in the correct directory.**
2. **Check Directory Structure Verify the directory structure to make sure that the .github/workflows/terraform.yml file is correctly placed.**
3. **Create Directory and File (if missing) If the .github/workflows directory or the terraform.yml file does not exist, create them:  
     
   mkdir -p .github/workflows  
   touch .github/workflows/terraform.yml**
4. **Verify Current Directory Ensure you are in the correct directory where your Git repository is initialized.  
     
   cd D:\GCP-Terraform\First-Demo**
5. **List Files to Confirm List the files in the .github/workflows directory to confirm the terraform.yml file exists:  
     
   dir .github\workflows**

**Example Steps**

**Here’s a sequence of commands you can follow to ensure the file exists and to add it to the Git repository:**

**cd D:\GCP-Terraform\First-Demo**

**# Create directories if they don't exist  
mkdir -p .github/workflows**

**# Create the terraform.yml file if it doesn't exist  
touch .github/workflows/terraform.yml**

**# Verify the file exists  
dir .github\workflows**

**# Add the file to the repository  
git add .github/workflows/terraform.yml**

**# Commit the changes  
git commit -m "Add Terraform GitHub Actions workflow"**

**# Push the changes to the remote repository  
git push origin main**

**If you execute git push origin main and it still results in errors, here’s a checklist to resolve potential issues:**

1. **Verify Branch Name: Confirm that your local branch is indeed named main:**

**bash**

**Copy code**

**git branch**

**If you don’t see main, you’re likely on a different branch (e.g., master).**

1. **Push to Existing Branch: If your branch is master and you want to push to main on the remote, you might need to first create the main branch on your local repository if it doesn’t exist:**

**bash**

**Copy code**

**git branch main**

**Switch to the new branch:**

**bash**

**Copy code**

**git checkout main**

**Then push:**

**bash**

**Copy code**

**git push origin main**

1. **Set Upstream Branch: If you’re pushing a branch for the first time, you might need to set the upstream branch:**

**bash**

**Copy code**

**git push --set-upstream origin main**

1. **Check Remote Repository: Ensure that the remote repository exists and is accessible. Verify the remote URL:**

**bash**

**Copy code**

**git remote -v**

1. **Ensure You Have Access: Make sure you have the necessary permissions to push to the repository. You might need to authenticate with your Git hosting service (e.g., GitHub, GitLab) if prompted.**
2. **Resolve Conflicts: If the branch you’re trying to push to exists on the remote but has diverged from your local branch, you might need to pull the latest changes:**

**bash**

**Copy code**

**git pull origin main**

**o push new changes to your Git repository, follow these steps:**

1. **Make Your Changes: Edit, add, or delete files as needed in your local repository.**
2. **Stage Your Changes: Add the changes to the staging area with git add. You can stage specific files or all changes:**
   * **Stage specific files:**

**bash**

**Copy code**

**git add <file1> <file2>**

* + **Stage all changes:**

**bash**

**Copy code**

**git add .**

1. **Commit Your Changes: Commit the staged changes with a descriptive message:**

**bash**

**Copy code**

**git commit -m "Your commit message here"**

1. **Push Your Changes: Push the committed changes to the remote repository. Replace main with your branch name if necessary:**

**bash**

**Copy code**

**git push origin main**

**If you’re working with a branch other than main, replace main with the name of your branch. For example, if your branch is named feature-branch:**

**bash**

**Copy code**

**git push origin feature-branch**

**Make sure your local branch is up to date with the remote branch to avoid conflicts. If there are updates on the remote branch, you may need to pull the latest changes before pushing.**

**4o mini**

**Invalid workflow file: .github/workflows/terraform.yml#L43 You have an error in your yaml syntax on line 43**

**ChatGPT**

**It looks like there’s a syntax error in your GitHub Actions workflow file at line 43. YAML syntax issues can be tricky, so here are some common things to check:**

1. **Indentation: YAML is sensitive to indentation. Make sure you use consistent spaces (usually 2 or 4 spaces) and avoid mixing tabs and spaces.**
2. **Colons and Hyphens: Ensure that you have colons followed by a space and that list items start with a hyphen followed by a space.**
3. **Quotes: If you’re using quotes, make sure they match and are correctly placed.**
4. **Syntax Structure: Verify that the structure of your workflow is correct according to GitHub Actions syntax.**

**Here’s a general template for a Terraform workflow in GitHub Actions for reference:**

**yaml**

**Copy code**

**name: Terraform**

**on:**

**push:**

**branches:**

**- main**

**jobs:**

**terraform:**

**runs-on: ubuntu-latest**

**steps:**

**- name: Checkout code**

**uses: actions/checkout@v3**

**- name: Set up Terraform**

**uses: hashicorp/setup-terraform@v2**

**with:**

**terraform\_version: 1.5.0**

**- name: Terraform Init**

**run: terraform init**

**- name: Terraform Plan**

**run: terraform plan**

**- name: Terraform Apply**

**run: terraform apply -auto-approve**

**env:**

**TF\_VAR\_example: ${{ secrets.TF\_VAR\_EXAMPLE }}**

**If you can share the snippet around line 43, I can help identify the issue more precisely.**

**4o mini**

**name: 'Terraform CI/CD' on: push: branches: - main pull\_request: branches: - main jobs: terraform: name: 'Terraform' runs-on: ubuntu-latest steps: - name: 'Checkout GitHub Action' uses: actions/checkout@v2 - name: 'Setup Terraform' uses: hashicorp/setup-terraform@v1 with: terraform\_version: 1.0.0 - name: 'Authenticate to Google Cloud' run: echo "${{ secrets.GCP\_CREDENTIALS }}" > $HOME/gcloud.json - name: 'Setup gcloud CLI' uses: google-github-actions/setup-gcloud@v0.3.0 with: service\_account\_key: ${{ secrets.GCP\_CREDENTIALS }} export\_default\_credentials: true - name: 'Terraform Init' run: terraform init - name: 'Terraform Plan' run: terraform plan - name: 'Terraform Apply' if: github.ref == 'refs/heads/main' run: terraform apply -auto-approve <<<<<<< HEAD ======= >>>>>>> 170f7fae5ab6eaee55aa4f1926af22139530bbb1**

**ChatGPT**

**It looks like there’s a merge conflict marker (<<<<<<< HEAD, =======, >>>>>>> 170f7fae5ab6eaee55aa4f1926af22139530bbb1) in your YAML file, which indicates that there was a conflict during a merge operation. You need to resolve these conflict markers manually.**

**Here’s how to fix it:**

1. **Remove Conflict Markers: Find and remove the conflict markers and choose the correct code to keep. The conflict markers are:**
   * **<<<<<<< HEAD — marks the beginning of the conflicting changes from the current branch.**
   * **======= — separates the conflicting changes.**
   * **>>>>>>> 170f7fae5ab6eaee55aa4f1926af22139530bbb1 — marks the end of the conflicting changes from the branch you’re merging.**
2. **Corrected YAML Example: Here’s what your file might look like after resolving the conflict:**

**yaml**

**Copy code**

**name: 'Terraform CI/CD'**

**on:**

**push:**

**branches:**

**- main**

**pull\_request:**

**branches:**

**- main**

**jobs:**

**terraform:**

**name: 'Terraform'**

**runs-on: ubuntu-latest**

**steps:**

**- name: 'Checkout GitHub Action'**

**uses: actions/checkout@v2**

**- name: 'Setup Terraform'**

**uses: hashicorp/setup-terraform@v1**

**with:**

**terraform\_version: 1.0.0**

**- name: 'Authenticate to Google Cloud'**

**run: echo "${{ secrets.GCP\_CREDENTIALS }}" > $HOME/gcloud.json**

**- name: 'Setup gcloud CLI'**

**uses: google-github-actions/setup-gcloud@v0.3.0**

**with:**

**service\_account\_key: ${{ secrets.GCP\_CREDENTIALS }}**

**export\_default\_credentials: true**

**- name: 'Terraform Init'**

**run: terraform init**

**- name: 'Terraform Plan'**

**run: terraform plan**

**- name: 'Terraform Apply'**

**if: github.ref == 'refs/heads/main'**

**run: terraform apply -auto-approve**

1. **Commit the Changes: After resolving the conflict and saving the file, commit the changes:**

**bash**

**Copy code**

**git add .github/workflows/terraform.yml**

**git commit -m "Resolve merge conflict in Terraform CI/CD workflow"**

1. **Push the Changes: Push the updated workflow file to the remote repository:**

**bash**

**Copy code**

**git push origin main**

**This should resolve the issue, and your workflow file should be properly configured and free from merge conflicts.**

**4o mini**

**Not able to create new virtual instance GCP after pust the terraform.yml**

**ChatGPT**

**If you’re encountering issues with creating a new virtual instance in GCP after pushing your Terraform GitHub Actions workflow file, it’s essential to debug both the Terraform configuration and the GitHub Actions workflow. Here’s a step-by-step approach to troubleshoot and resolve the issue:**

1. **Check GitHub Actions Workflow Logs:**
   * **Go to your GitHub repository.**
   * **Click on the "Actions" tab.**
   * **Find the workflow run related to your recent push and click on it to see the detailed logs.**
   * **Look for any error messages or failed steps in the logs.**
2. **Verify Terraform Configuration: Ensure that your Terraform files (main.tf, variables.tf, etc.) are correctly configured for creating a virtual instance. Here’s a basic example of a Terraform configuration for creating a GCP virtual instance:**

**hcl**

**Copy code**

**provider "google" {**

**credentials = file("${path.module}/gcloud.json")**

**project = var.project\_id**

**region = var.region**

**}**

**resource "google\_compute\_instance" "default" {**

**name = "terraform-instance"**

**machine\_type = "e2-medium"**

**zone = "us-central1-a"**

**boot\_disk {**

**initialize\_params {**

**image = "debian-cloud/debian-10"**

**}**

**}**

**network\_interface {**

**network = "default"**

**access\_config {}**

**}**

**}**

**variable "project\_id" {**

**description = "The GCP project ID"**

**type = string**

**}**

**variable "region" {**

**description = "The GCP region"**

**type = string**

**default = "us-central1"**

**}**

1. **Check Service Account Permissions: Ensure that the service account used by your GitHub Actions workflow has the necessary permissions to create resources in GCP. The service account key is stored in secrets.GCP\_CREDENTIALS. The service account should have roles such as roles/compute.admin and roles/iam.serviceAccountUser.**
2. **Check for Errors in Terraform Plan/Apply Steps:**
   * **Ensure that the terraform plan and terraform apply steps in your workflow are executed correctly.**
   * **Look for specific error messages in the output of these steps.**
3. **Validate Terraform Files Locally: Before running the GitHub Actions workflow, validate and test your Terraform configuration locally:**

**bash**

**Copy code**

**terraform init**

**terraform plan**

**terraform apply -auto-approve**

1. **Update GitHub Actions Workflow: Ensure your .github/workflows/terraform.yml file is correctly configured. Here’s an updated example:**

**yaml**

**Copy code**

**name: 'Terraform CI/CD'**

**on:**

**push:**

**branches:**

**- main**

**pull\_request:**

**branches:**

**- main**

**jobs:**

**terraform:**

**name: 'Terraform'**

**runs-on: ubuntu-latest**

**steps:**

**- name: 'Checkout GitHub Action'**

**uses: actions/checkout@v2**

**- name: 'Setup Terraform'**

**uses: hashicorp/setup-terraform@v1**

**with:**

**terraform\_version: 1.0.0**

**- name: 'Authenticate to Google Cloud'**

**run: echo "${{ secrets.GCP\_CREDENTIALS }}" > $HOME/gcloud.json**

**- name: 'Setup gcloud CLI'**

**uses: google-github-actions/setup-gcloud@v0.3.0**

**with:**

**service\_account\_key: ${{ secrets.GCP\_CREDENTIALS }}**

**export\_default\_credentials: true**

**- name: 'Terraform Init'**

**run: terraform init**

**- name: 'Terraform Plan'**

**run: terraform plan**

**- name: 'Terraform Apply'**

**if: github.ref == 'refs/heads/main'**

**run: terraform apply -auto-approve**

1. **Review and Fix Any Errors:**
   * **Review the error messages in the GitHub Actions logs.**
   * **Make necessary adjustments to your Terraform configuration and workflow file.**
   * **Push the updated files to trigger the workflow again.**

**By following these steps, you should be able to identify and resolve the issue preventing the creation of a new virtual instance in GCP using your Terraform configuration and GitHub Actions workflow. If you need further assistance, feel free to share specific error messages or logs.**

**4o**

**how to get this : name: 'Authenticate to Google Cloud' run: echo "${{ secrets.GCP\_CREDENTIALS }}" > $HOME/gcloud.json - name: 'Setup gcloud CLI' uses: google-github-actions/setup-gcloud@v0.3.0 with: service\_account\_key: ${{ secrets.GCP\_CREDENTIALS }} export\_default\_credentials: true**

**ChatGPT**

**To set up the GitHub Actions workflow for authenticating to Google Cloud and configuring the gcloud CLI, follow these steps:**

**1. Create a Service Account and Download Credentials**

1. **Create a Service Account:**
   * **Go to the Google Cloud Console.**
   * **Navigate to IAM & Admin > Service Accounts.**
   * **Click Create Service Account.**
   * **Enter a name for the service account and click Create.**
   * **Assign necessary roles to the service account (e.g., roles/compute.admin, roles/iam.serviceAccountUser).**
   * **Click Done.**
2. **Generate a Key for the Service Account:**
   * **Select the newly created service account.**
   * **Go to the Keys tab.**
   * **Click Add Key > Create New Key.**
   * **Choose JSON and click Create.**
   * **A JSON file with the credentials will be downloaded.**

**2. Store the Service Account Key in GitHub Secrets**

1. **Go to Your GitHub Repository:**
   * **Navigate to the repository where you want to set up the workflow.**
2. **Add a New Secret:**
   * **Go to Settings > Secrets and variables > Actions.**
   * **Click New repository secret.**
   * **Name the secret GCP\_CREDENTIALS.**
   * **Copy the contents of the downloaded JSON file and paste it as the value of the secret.**
   * **Click Add secret.**

**3. Configure the GitHub Actions Workflow**

**Here is the updated GitHub Actions workflow file (.github/workflows/terraform.yml), including the steps for authenticating to Google Cloud and setting up the gcloud CLI:**

**yaml**

**Copy code**

**name: 'Terraform CI/CD'**

**on:**

**push:**

**branches:**

**- main**

**pull\_request:**

**branches:**

**- main**

**jobs:**

**terraform:**

**name: 'Terraform'**

**runs-on: ubuntu-latest**

**steps:**

**- name: 'Checkout GitHub Action'**

**uses: actions/checkout@v2**

**- name: 'Setup Terraform'**

**uses: hashicorp/setup-terraform@v1**

**with:**

**terraform\_version: 1.0.0**

**- name: 'Authenticate to Google Cloud'**

**run: echo "${{ secrets.GCP\_CREDENTIALS }}" > $HOME/gcloud.json**

**- name: 'Setup gcloud CLI'**

**uses: google-github-actions/setup-gcloud@v0.3.0**

**with:**

**service\_account\_key: ${{ secrets.GCP\_CREDENTIALS }}**

**export\_default\_credentials: true**

**- name: 'Terraform Init'**

**run: terraform init**

**- name: 'Terraform Plan'**

**run: terraform plan**

**- name: 'Terraform Apply'**

**if: github.ref == 'refs/heads/main'**

**run: terraform apply -auto-approve**

**Explanation**

1. **Authenticate to Google Cloud:**

**yaml**

**Copy code**

**- name: 'Authenticate to Google Cloud'**

**run: echo "${{ secrets.GCP\_CREDENTIALS }}" > $HOME/gcloud.json**

**This step creates a gcloud.json file in the home directory with the contents of the GCP\_CREDENTIALS secret.**

1. **Setup gcloud CLI:**

**yaml**

**Copy code**

**- name: 'Setup gcloud CLI'**

**uses: google-github-actions/setup-gcloud@v0.3.0**

**with:**

**service\_account\_key: ${{ secrets.GCP\_CREDENTIALS }}**

**export\_default\_credentials: true**

**This step configures the gcloud CLI to use the service account key from the GCP\_CREDENTIALS secret.**

**By following these steps, your GitHub Actions workflow should be able to authenticate to Google Cloud and set up the gcloud CLI, enabling the subsequent Terraform commands to execute successfully.**