**Terraform Workflow for Provisioning Google Compute Engine (GCE) Instances**

This document outlines the **step-by-step Terraform process**, including **configuration**, **troubleshooting**, and solutions applied to common issues.

**📌 1. Terraform Project Structure**

Organize the Terraform project with modules for reusability:

terraform-gce/

│── main.tf

│── variables.tf

│── terraform.tfvars

│── outputs.tf

│── modules/

│ └── compute\_instance/

│ ├── main.tf

│ ├── variables.tf

│ ├── outputs.tf

**📌 2. Define Terraform Variables (variables.tf)**

Declare global variables **without default values**:

hcl

variable "project\_id" {

description = "GCP Project ID"

type = string

}

variable "region" {

description = "GCP Region"

type = string

}

variable "zone" {

description = "GCP Zone"

type = string

}

variable "machine\_type" {

description = "Machine type for the VM"

type = string

}

variable "image" {

description = "OS image for the VM"

type = string

}

variable "instance\_name" {

description = "Name of the VM instance"

type = string

}

**📌 3. Create a Reusable Module (modules/compute\_instance/main.tf)**

Encapsulate VM provisioning inside a **Terraform module**:

hcl

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

name = var.instance\_name

machine\_type = var.machine\_type

zone = var.zone

boot\_disk {

initialize\_params {

image = var.image

}

}

network\_interface {

network = "default"

access\_config {

// Assigns a public IP

}

}

}

**📌 4. Define Module-Specific Variables (modules/compute\_instance/variables.tf)**

Declare input variables inside the module:

hcl

variable "instance\_name" {

description = "Name of the VM instance"

type = string

}

variable "machine\_type" {

description = "Machine type for the VM"

type = string

}

variable "zone" {

description = "GCP Zone"

type = string

}

variable "image" {

description = "OS image for the VM"

type = string

}

**📌 5. Define Module Outputs (modules/compute\_instance/outputs.tf)**

Ensure outputs **pass values back** to the root module:

hcl

output "instance\_public\_ip" {

description = "Public IP of the VM instance"

value = google\_compute\_instance.vm\_instance.network\_interface[0].access\_config[0].nat\_ip

}

output "instance\_internal\_ip" {

description = "Internal IP of the VM instance"

value = google\_compute\_instance.vm\_instance.network\_interface[0].network\_ip

}

**📌 6. Reference the Module in main.tf (Root Level)**

Call the **module from the root Terraform configuration**:

hcl

provider "google" {

project = var.project\_id

region = var.region

credentials = file("C:\\GCPCICD\\GCP\_GCE\\credentials.json")

}

module "compute\_instance" {

source = "./modules/compute\_instance"

instance\_name = var.instance\_name

machine\_type = var.machine\_type

zone = var.zone

image = var.image

}

**📌 7. Define terraform.tfvars for Input Values**

Move **all values into** .tfvars **for flexibility**:

hcl

project\_id = "gce03021991"

region = "us-central1"

zone = "us-central1-a"

machine\_type = "e2-medium"

image = "debian-cloud/debian-11"

instance\_name = "terraform-gce-instance"

**📌 8. Retrieve Outputs in Root Level (outputs.tf)**

Reference **module outputs at the root level**:

hcl

output "instance\_public\_ip" {

description = "Public IP of the VM instance"

value = module.compute\_instance.instance\_public\_ip

}

output "instance\_internal\_ip" {

description = "Internal IP of the VM instance"

value = module.compute\_instance.instance\_internal\_ip

}

**🛠 Issues Faced & Fixes Applied**

**🔹 Issue #1: Terraform Initialization Error – Duplicate Output Definition**

**Problem:** Terraform failed due to duplicate instance\_ip output. **Fix:** ✅ Removed duplicate instance\_ip output from both module and root outputs.tf. ✅ Renamed outputs to instance\_public\_ip and instance\_internal\_ip for uniqueness.

**🔹 Issue #2: Terraform Variables Not Loaded from terraform.tfvars**

**Problem:** Terraform prompted for manual input, ignoring terraform.tfvars. **Fix:** ✅ Verified terraform.tfvars exists in the same directory. ✅ Explicitly used terraform apply -var-file="terraform.tfvars". ✅ Ensured **all variables were defined in** variables.tf.

**🔹 Issue #3: Terraform Still Showing Outputs After Removing outputs.tf**

**Problem:** Terraform displayed outputs despite deletion of outputs.tf. **Fix:** ✅ Terraform **retained outputs in state (**terraform.tfstate**)**. ✅ Ran terraform refresh to update state. ✅ Deleted .terraform, terraform.tfstate, and terraform.tfstate.backup.

**🔹 Issue #4: Terraform Destroy Still Asked for tfvars File**

**Problem:** Terraform prompted for missing variable values during destroy. **Fix:** ✅ Passed terraform.tfvars explicitly during destruction:

sh

terraform destroy -var-file="terraform.tfvars"

✅ Verified all variables were **correctly defined in** variables.tf.

**🔹 Issue #5: Persistent Outputs After Resource Destruction**

**Problem:** After running terraform destroy, Terraform still displayed old outputs. **Fix:** ✅ **Removed all Terraform state files:**

sh

rm -rf .terraform terraform.tfstate terraform.tfstate.backup

✅ **Reinitialized Terraform from scratch:**

sh

terraform init

✅ **Verified that no lingering outputs remained:**

sh

terraform state list

**✅ Terraform Commands Used**

sh

terraform init

terraform apply -var-file="terraform.tfvars"

terraform destroy -var-file="terraform.tfvars"

terraform refresh

rm -rf .terraform terraform.tfstate terraform.tfstate.backup

terraform state list

terraform show

**🎯 Final Outcome**

* **Resolved duplicate outputs**
* **Successfully automated variable handling**
* **Properly cleaned up Terraform state**
* **Ensured environment flexibility using** terraform.tfvars
* **Reset Terraform state to remove past outputs**