**Solution**

**Task 1. Create the configuration files**

Create the directory and the file for your new module:

**touch** [**main.tf**](http://main.tf)

**touch** [**variables.tf**](http://variables.tf)

**mkdir modules**

**cd modules**

**mkdir instances**

**cd instances**

**touch** [**instances.tf**](http://instances.tf)

**touch** [**outputs.tf**](http://outputs.tf)

**touch** [**variables.tf**](http://variables.tf)

**cd ..**

**mkdir storage**

**cd storage**

**touch** [**storage.tf**](http://storage.tf)

**touch** [**outputs.tf**](http://outputs.tf)

**touch** [**variables.tf**](http://variables.tf)

**cd**

Add the following to the each [variables.tf](http://variables.tf) file, and fill in the GCP Project ID:

**variable "region" {**

**default = "us-central1"**

**}**

**variable "zone" {**

**default = "us-central1-a"**

**}**

**variable "project\_id" {**

**default = "<FILL IN PROJECT ID>"**

**}**

Add this to the main.tf file **terraform {**

**required\_providers {**

**google = { source = "hashicorp/google"**

**version = "**[**3.55.0**](tel:3.55.0)**"**

**}**

**}**

**}**

**provider "google" {**

**project = var.project\_id**

**region =** [**var.region**](http://var.region)

**zone =** [**var.zone**](http://var.zone)

**}**

**module "instances" {**

**source = "./modules/instances"**

**}**

1. Initialize Terraform

**terraform init**

**Task 2. Import infrastructure**

1. Look for name of the instances

Open main.tf and this Navigate to Compute Engine > VM Instances. Click on tf-instance-1. Copy the Instance ID down somewhere to use later.

Navigate to Compute Engine > VM Instances. Click on tf-instance-2. Copy the Instance ID down somewhere to use later.

Next, navigate to modules/instances/[instances.tf](http://instances.tf).

Copy the following configuration into the file:

--------------------------------------------------------------

**resource "google\_compute\_instance" "tf-instance-1" {**

**name = "tf-instance-1" machine\_type = "n1-standard-1" zone =** [**var.zone**](http://var.zone)

**boot\_disk {**

**initialize\_params {**

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

**}**

**}**

**network\_interface {**

**network = "default"**

**}**

**metadata\_startup\_script = <<-EOT**

**#!/bin/bash**

**EOT**

**allow\_stopping\_for\_update = true**

**}**

**resource "google\_compute\_instance" "tf-instance-2" {**

**name = "tf-instance-2" machine\_type = "n1-standard-1" zone =** [**var.zone**](http://var.zone)

**boot\_disk {**

**initialize\_params {**

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

**}**

**}**

**network\_interface {**

**network = "default"**

**}**

**metadata\_startup\_script = <<-EOT**

**#!/bin/bash**

**EOT**

**allow\_stopping\_for\_update = true**

**}**

To import the first instance, use the following command, using the Instance ID for tf-instance-1 you copied down earlier.

**terraform import** [**module.instances.google\_compute\_instance.tf**](http://module.instances.google_compute_instance.tf)**-instance-1 <Instance ID - 1>**

To import the second instance, use the following command, using the Instance ID for tf-instance-2 you copied down earlier.

**terraform import** [**module.instances.google\_compute\_instance.tf**](http://module.instances.google_compute_instance.tf)**-instance-2 <Instance ID - 2>**

The two instances have now been imported into your terraform configuration. You can now optionally run the commands to update the state of Terraform. Type yes at the dialogue after you run the apply command to accept the state changes.

Use

**terraform plan**

**terraform apply**

====================== TASK 3: Configure a remote backend ======================

Create a bucket in the storage storage.tf module Add the following code to the modules/storage/[storage.tf](http://storage.tf) file:

**resource "google\_storage\_bucket" "storage-bucket" {**

**name = var.project\_id**

**location = "US"**

**force\_destroy = true**

**uniform\_bucket\_level\_access = true**

**}**

Add a local backend to your main.tf file:

Next, add the following to the [main.tf](http://main.tf) file:

**module "storage" {**

**source = "./modules/storage"**

**}**

Run the following commands to initialize the module and create the storage bucket resource. Type yes at the dialogue after you run the apply command to accept the state changes.

**terraform init**

**terraform apply**

Next, update the [main.tf](http://main.tf) file so that the terraform block looks like the following. Fill in your GCP Project ID for the bucket argument definition.

**terraform {**

**backend "gcs" {**

**bucket = "<FILL IN PROJECT ID>" prefix = "terraform/state"**

**}**

**required\_providers {**

**google = {**

**source = "hashicorp/google" version = "**[**3.55.0**](tel:3.55.0)**"**

**}**

**}**

**}**

--------------------------------------------

Run the following to initialize the remote backend. Type yes at the prompt.

**terraform init**

====================== TASK 4: Modify and update infrastructure ====================== Navigate to modules/instances/[instance.tf](http://instance.tf). Replace the entire contents of the file with the following: --------------------------------------------------------

**resource "google\_compute\_instance" "tf-instance-1" {**

**name = "tf-instance-1" machine\_type = "n1-standard-1" zone =** [**var.zone**](http://var.zone)

**boot\_disk {**

**initialize\_params {**

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

**}**

**}**

**network\_interface {**

**network = "default"**

**}**

**metadata\_startup\_script = <<-EOT**

**#!/bin/bash**

**EOT**

**allow\_stopping\_for\_update = true**

**}**

**resource "google\_compute\_instance" "tf-instance-2" {**

**name = "tf-instance-2" machine\_type = "n1-standard-1" zone =** [**var.zone**](http://var.zone)

**boot\_disk {**

**initialize\_params {**

**image = "debian-cloud/debian-11"**

**}**

**}**

**network\_interface {**

**network = "default"**

**}**

**metadata\_startup\_script = <<-EOT**

**#!/bin/bash**

**EOT**

**allow\_stopping\_for\_update = true**

**}**

**resource "google\_compute\_instance" "tf-instance-3" {**

**name = "tf-instance-3" machine\_type = "n1-standard-1" zone =** [**var.zone**](http://var.zone)

**boot\_disk {**

**initialize\_params {**

**image = "debian-cloud/debian-11"**

**}**

**}**

**network\_interface {**

**network = "default"**

**}**

**metadata\_startup\_script = <<-EOT**

**#!/bin/bash**

**EOT**

**allow\_stopping\_for\_update = true**

**}**

Run the following commands to initialize the module and create/update the instance resources. Type yes at the dialogue after you run the apply command to accept the state changes.

**terraform init**

**terraform apply**

====================== TASK 5: Taint and destroy resources ======================

Taint the tf-instance-3 resource by running the following command:

**terraform taint** [**module.instances.google\_compute\_instance.tf**](http://module.instances.google_compute_instance.tf)**-instance-3**

Run the following commands to apply the changes:

**terraform init**

**terraform apply**

Remove the tf-instance-3 resource from the [instances.tf](http://instances.tf) file. Delete the following code chunk from the file.

**resource "google\_compute\_instance" "tf-instance-3" {**

**name = "tf-instance-3" machine\_type = "n1-standard-1" zone =** [**var.zone**](http://var.zone)

**boot\_disk { initialize\_params { image = "debian-cloud/debian-10" }**

**}**

**network\_interface {**

**network = "default"**

**}**

**metadata\_startup\_script = <<-EOT**

**#!/bin/bash**

**EOT**

**allow\_stopping\_for\_update = true**

**}**

Run the following commands to apply the changes. Type yes at the prompt.

**terraform apply**

====================== TASK 6: Use a module from the Registry ======================

Copy and paste the following into the [main.tf](http://main.tf) file:

**module "vpc" {**

**source = "terraform-google-modules/network/google" version = "~>** [**6.0.0**](tel:3.2.2)**" project\_id = var.project\_id network\_name = "terraform-vpc" routing\_mode = "GLOBAL" subnets = [**

**{**

**subnet\_name = "subnet-01" subnet\_ip = "**[**10.10.10.0/24**](http://10.10.10.0/24)**" subnet\_region = "us-central1" },**

**{ subnet\_name = "subnet-02" subnet\_ip = "**[**10.10.20.0/24**](http://10.10.20.0/24)**" subnet\_region = "us-central1" subnet\_private\_access = "true" subnet\_flow\_logs = "true" description = "This subnet has a description" }**

**]**

**}**

Run the following commands to initialize the module and create the VPC. Type yes at the prompt.

**terraform init terraform apply**

Navigate to modules/instances/[instances.tf](http://instances.tf). Replace the entire contents of the file with the following:

**resource "google\_compute\_instance" "tf-instance-1" {**

**name = "tf-instance-1" machine\_type = "n1-standard-2" zone =** [**var.zone**](http://var.zone) **allow\_stopping\_for\_update = true boot\_disk { initialize\_params {**

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

**}**

**}**

**network\_interface {**

**network = "terraform-vpc" subnetwork = "subnet-01"**

**}**

**}**

**resource "google\_compute\_instance" "tf-instance-2" {**

**name = "tf-instance-2" machine\_type = "n1-standard-2" zone =** [**var.zone**](http://var.zone) **allow\_stopping\_for\_update = true boot\_disk {**

**initialize\_params {**

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

**}**

**}**

**network\_interface {**

**network = "terraform-vpc"**

**subnetwork = "subnet-02" }**

**}**

Run the following commands to initialize the module and update the instances. Type yes at the prompt.

**terraform init terraform apply**

====================== TASK 7: Configure a firewall ======================

Add the following resource to the [main.tf](http://main.tf) file and fill in the GCP Project ID:

**resource "google\_compute\_firewall" "tf-firewall" {**

**name = "tf-firewall" network = "projects/<PROJECT\_ID>/global/networks/terraform-vpc" allow {**

**protocol = "tcp" ports = ["80"] }**

**source\_tags = ["web"]**

**source\_ranges = ["**[**0.0.0.0**](tel:0.0.0.0)**/0"]**

**}**

Run the following commands to configure the firewall. Type yes at the prompt.

**terraform init terraform apply**