Configure a VPC deployment with Terraform variables and iterations – Google Cloud

Contents

VPC deployment on GCP with variables and iterations

Overview

This lab will modify existing code that has been used successfully to deploy a 'hard-coded' basic VPC architecture

Objectives

There is a Solution section at the very end of these instructions. Try to use this only as a last resort if you are struggling to complete the step-by-step processes.

Lab Setup

For each lab, you are provided with a new Google Cloud project and a set of resources for a fixed time at no cost.

1. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time. There is no pause feature. You can restart if needed, but you have to start at the beginning.

2. When ready, click **Start lab.**

3. Note your lab credentials (Username and Password). You will use them to sign in to the Google Cloud Console.

4. Click **Open Google Console**.

5. Click **Use another account** and copy/paste credentials for this lab into the prompts. If you use other credentials, you'll receive errors or incur charges.

6. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Terraform and Cloud Shell Setup

1. In the Cloud Console, click Activate Cloud Shell

2. If prompted, click Continue.

3. Select the **Open in a new window** icon

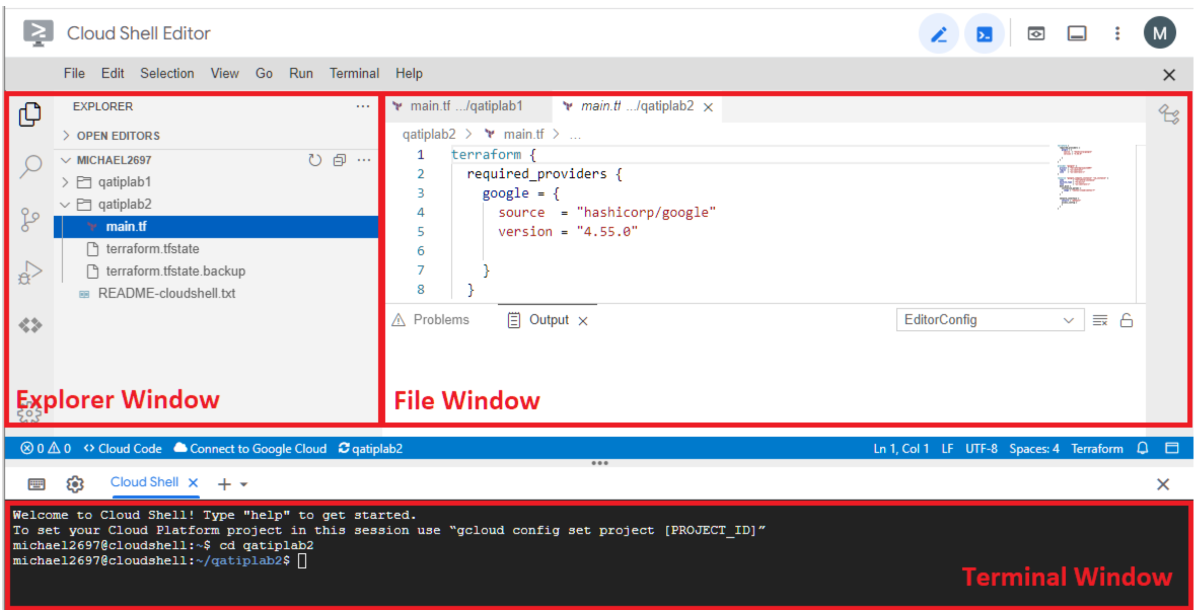
4. Select the **Open Editor** icon

5. Select **Open Home Workspace** if explorer pane does not display.

6. Create a directory for your Terraform configuration and initial files by running the following commands in the terminal:

```
cd ~
mkdir ./lab
cd ./lab
touch main.tf
```

7. In the Explorer, navigate to and click on main.tf to open the empty file for editing...



```
terraform {
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "4.55.0"
    }
  }
```

```
Welcome to Cloud Shell! Type "help" to get started.
To set your Cloud Platform project in this session use "gcloud config set project [PROJECT_ID]"
michael2697@cloudshell:~$ cd qatiplab2
michael2697@cloudshell:~/qatiplab2$
```

8. You now have 2 browser tabs open. The current tab will be referred to as the `IDE` throughout these instructions.

9. Switch to the Google console tab and close the cloud shell. This tab will be referred to as the `Console` throughout these instructions.

10. Switch back to the `IDE`

11. Execute all terraform commands from within the lab directory

Task 1. Load existing code

1. Switch to the **IDE** and copy the following code into main.tf, updating line 11 with your project ID…

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "4.55.0"
    }
  }
}

provider "google" {
  project = "{YOUR PROJECT ID HERE}"
  region = "us-central1"
}

resource "google_compute_network" "lab-vpc" {
    name = "lab-vpc"
    auto_create_subnetworks = false
}

resource "google_compute_subnetwork" "public-subnet" {
  name = "public-subnet"
  ip_cidr_range = "10.0.1.0/24"
  region        = "us-central1"
  network = google_compute_network.lab-vpc.id
}

resource "google_compute_subnetwork" "private-subnet" {
  name = "private-subnet"
  ip_cidr_range = "10.0.2.0/24"
  region        = "us-central1"
  network = google_compute_network.lab-vpc.id
}

resource "google_compute_router" "lab-router" {
   name = "lab-router"
  network = google_compute_network.lab-vpc.id
}

resource "google_compute_router_nat" "lab-nat" {
  name = "lab-nat"
  router = google_compute_router.lab-router.name
  region = google_compute_router.lab-router.region
  nat_ip_allocate_option = "AUTO_ONLY"
  source_subnetwork_ip_ranges_to_nat = "LIST_OF_SUBNETWORKS"

  subnetwork {
    name = google_compute_subnetwork.private-subnet.id
    source_ip_ranges_to_nat = ["ALL_IP_RANGES"]
  }
}

resource "google_compute_instance" "priv-vm" {
  name         = "priv-vm"
  machine_type = "e2-micro"
  zone         = "us-central1-a"
  allow_stopping_for_update = true
  tags = ["priv-subnet-vm"]

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
    }
  }
  network_interface {
    subnetwork = google_compute_subnetwork.private-subnet.id
  }
}

resource "google_compute_instance" "pub-vm" {
  name         = "pub-vm"
  machine_type = "e2-micro"
  zone         = "us-central1-a"
  allow_stopping_for_update = true
  tags = ["pub-subnet-vm"]

  boot_disk {
   initialize_params {
     image = "debian-cloud/debian-11"
    }
  }
  network_interface {
    subnetwork = google_compute_subnetwork.public-subnet.id

    access_config {
      // Ephemeral public IP
    }
  }
}

resource "google_compute_firewall" "lab-private-fw-rule" {
  name    = "lab-private-firewall"
  network = google_compute_network.lab-vpc.name

  allow {
    protocol = "all"
  }
  source_tags = ["pub-subnet-vm"]
  target_tags = ["priv-subnet-vm"]
}

resource "google_compute_firewall" "lab-public-fw-rule" {
  name    = "lab-public-firewall"
  network = google_compute_network.lab-vpc.name

  allow {
    protocol = "tcp"
    ports    = ["22"]
  }
  source_ranges =["0.0.0.0/0"]
  target_tags = ["pub-subnet-vm"]
}
```

2. Save main.tf

3. Run **terraform init**

4. Run **terraform apply**, entering **yes** when prompted. Authorize Cloud Shell if prompted.

5. A total of 9 resources should be added. If you have not completed the previous lab in which you created this deployment file then familiarize yourself with the components deployed using the console.

6. Switch to the IDE and run **terraform destroy** to clear the hard-coded deployment

Task 2. Introducing variables

Notice how main.tf has all argument values explicitly declared, the region, the zone, the subnet names etc. This is a 'hard-coded' file which limits its re-usability. We could duplicate, edit, and deploy a new version, but this is not optimal. Ideally we should be able re-use the same code-base but introduce variations as needed. This is where variables come into use.

Take a moment to review developer information regarding variables from Hashicorp at https://developer.hashicorp.com/terraform/language/values/variables

1. Create a file to hold our variables by entering **touch variable.tf** in terminal session

2. Open the new file using the IDE Explore and paste the following boiler-plate code into it.

```
variable "" {
  description = ""
  type        =
  default     = ""
}
```

3. We will create a variable for the region we are going to deploy to. Modify variable.tf to set us-east1 as our region

```
variable "deployment_region" {
  description = "Region of deployment"
  type        = string
  default     = "us-east1"
}
```

4. Save variable.tf

5. Navigate to the provider configuration block in main.tf (Lines 10-13)

6. Replace `"us-central1"` with **var.deployment_region** (no quotes). Note how variables are referenced by **var.** followed by the name of the variable, as defined in variable.tf

7. Save main.tf and run **terraform apply yes**

8. Note the error regarding 'lab-nat'. We have changed the default region and this change is applicable to resources that are created without explicit indication of the region, in this case the lab-router (lines 34-36). The NAT router resource (lines 39-50) references the lab-router, but it also references the private subnet. This subnet is being created in us-central1 still (Line23). A NAT router is a regional entity and cannot reference subnets from other regions. This shows how changing one argument may have a knock-on effect!

9. Run **terraform destroy yes** to roll back the partial deployment

10. Use Edit, Replace in the Cloud Shell Editor and replace **"us-central1"** with **var.deployment_region**. There should be 2 replacements.

11. Save main.tf and run **terraform apply**

12. We again have errors, this time relating to the compute instances. The 2 compute instance resources specify the networks that the instances should join (lines 65 and 82). Both of these subnets are in the region specified by the variable we declared. The zones where the instances themselves are to be created is still hard-coded as us-central1-a though (lines 55 and 72). This zone is not part of the region where the subnets are to exist and therefore the compute instance creation fails.

13. Switch to variable.tf and paste in the new block below.

```
variable "deployment_zone" {
  description = "Zone"
  type        = string
  default     = "us-east1-b"
}
```

14. In main.tf, change the hard-coded value for the 2 zone argument values from the current value "us-central1-a" to `var.deployment_zone` The IDE has a find/replace facility to assist in this.

15. Save both files and run `terraform apply` `yes` The 2 compute instance resources should now deploy without error. Switch to the console to verify this.

Try this yourself

16. Create a variable called **var.machine_size** with a default string value of **"e2-micro"** then replace the 2 hard-coded references to **machine_type** with this variable in main.tf. Apply these changes. No changes should occur as the current machine type has not been altered, we have just declared it as a variable.

17. Update the default value of var.machine_size to **"e2-small"** and apply the changes. The existing virtual machines are now updated from type e2-micro to e2-small.

Task 3. Simple Iterations

Our code currently deploys a single instance onto each of our 2 subnets. What is we wanted more ? We could duplicate the google_compute_instance blocks, renaming each to give us as many instances as we need. This is poor coding though, a better solution being to iterate through the same block multiple times.

1. Open variable.tf and paste the following block in...

   ```
   variable "instance_count" {
     description = "Number of instances in each subnet"
     type        = number
     default     = 2
   }
   ```

2. Open main.tf and paste the following argument line into the 2 google_compute_instance blocks, above the name argument.

   ```
   count = var.instance_count
   ```

3. Save both files and run `terraform apply` `yes`
4. Note the error regarding duplicate instance names. There are already machines deployed named priv-vm and pub-vm.
5. Update the Private instance name argument value to **"${format("${"priv-vm"}-${"%03d"}", count.index + 1)}"**
6. Update the Public instance name argument value to **"${format("${"pub-vm"}-${"%03d"}", count.index + 1)}"**
7. Apply these changes. Note that the original instances must be deleted because of the new naming convention and that a total of 4 instances are deployed, 2 per subnet. Verify this using the Console.
8. Destroy all 11 resources.

Solution

main.tf

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "4.55.0"
    }
```

```
  }
}

provider "google" {
  project = "qwiklabs-gcp-02-f70adfeaf09b"
  region = var.deployment_region
}

resource "google_compute_network" "lab-vpc" {
    name = "lab-vpc"
    auto_create_subnetworks = false
}

resource "google_compute_subnetwork" "public-subnet" {
  name = "public-subnet"
  ip_cidr_range = "10.0.1.0/24"
  region        = var.deployment_region
  network = google_compute_network.lab-vpc.id
}

resource "google_compute_subnetwork" "private-subnet" {
  name = "private-subnet"
  ip_cidr_range = "10.0.2.0/24"
  region        = var.deployment_region
  network = google_compute_network.lab-vpc.id
}

resource "google_compute_router" "lab-router" {
   name = "lab-router"
  network = google_compute_network.lab-vpc.id
}

resource "google_compute_router_nat" "lab-nat" {
  name = "lab-nat"
  router = google_compute_router.lab-router.name
  region = google_compute_router.lab-router.region
  nat_ip_allocate_option = "AUTO_ONLY"
  source_subnetwork_ip_ranges_to_nat = "LIST_OF_SUBNETWORKS"

  subnetwork {
    name = google_compute_subnetwork.private-subnet.id
    source_ip_ranges_to_nat = ["ALL_IP_RANGES"]
  }
}

resource "google_compute_instance" "priv-vm" {
  count = var.instance_count
  name         = "${format("${"priv-vm"}-${"%03d"}", count.index + 1)}"
  machine_type = var.machine_size
  zone         = var.deployment_zone
  allow_stopping_for_update = true
  tags = ["priv-subnet-vm"]

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
    }
  }
  network_interface {
    subnetwork = google_compute_subnetwork.private-subnet.id
  }
}

resource "google_compute_instance" "pub-vm" {
  count = var.instance_count
  name         = "${format("${"pub-vm"}-${"%03d"}", count.index + 1)}"
  machine_type = var.machine_size
  zone         = var.deployment_zone
  allow_stopping_for_update = true
  tags = ["pub-subnet-vm"]

  boot_disk {
   initialize_params {
      image = "debian-cloud/debian-11"
    }
  }
  network_interface {
    subnetwork = google_compute_subnetwork.public-subnet.id

    access_config {
      // Ephemeral public IP
    }
  }
}

resource "google_compute_firewall" "lab-private-fw-rule" {
  name    = "lab-private-firewall"
  network = google_compute_network.lab-vpc.name

  allow {
    protocol = "all"
  }
  source_tags = ["pub-subnet-vm"]
  target_tags = ["priv-subnet-vm"]
}
```

```
resource "google_compute_firewall" "lab-public-fw-rule" {
  name    = "lab-public-firewall"
  network = google_compute_network.lab-vpc.name

  allow {
    protocol = "tcp"
    ports    = ["22"]
  }
  source_ranges =["0.0.0.0/0"]
  target_tags = ["pub-subnet-vm"]
}
```

variable.tf

```
variable "deployment_region" {
  description = "Region of deployment"
  type        = string
  default     = "us-east1"
}

variable "deployment_zone" {
  description = "Zone"
  type        = string
  default     = "us-east1-b"
}

variable "machine_size" {
  description = "Type"
  type        = string
  default     = "e2-small"
}

variable "instance_count" {
  description = "Number of instances in each subnet"
  type        = number
  default     = 2
}
```