QATIPv3 Google Cloud Lab2
Create a GCE instance with Terraform

Contents

Lab Objectives

In this lab, you will:

- Review the Terraform registry documentation on the AWS Provider
- Build a simple main.tf configuration file based on example code
- Deploy a GCE instance on Google Cloud
- Test and modify the deployment
- Destroy the deployment

Teaching Points

This lab takes you through the writing of code to create a GCE instance in Google Cloud.

Before you begin

1. Ensure you have completed Lab0 before attempting this lab.
2. In the IDE terminal pane, enter the following commands…

   cd ~/googlelabs/labs02

3. This shifts your current working directory to googlelabs/labs02. ***Ensure all commands are executed in this directory***
4. Close any open files and use the Explorer pane to navigate to and open the empty main.tf file in **googlelabs/lab02**

Solution

The solution to this lab can be found in **googlelabs/solutions/lab02**. Try to use this only as a last resort if you are struggling to complete the step-by-step processes

Task 1: Review the documentation and create a configuration file to deploy a GCE instance.

1. Review the Terraform Google Provider documentation:

   https://registry.terraform.io/providers/hashicorp/google/6.17.0/docs/resources/compute_instance

2. Click `Use Provider`
3. Copy the code block into the empty main.tf file in the labs/02 folder. For convenience, the code is listed below:

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
      version = "6.17.0"
    }
  }
}

provider "google" {
  # Configuration options
}
```

4. Click on the ‘Copy’ icon within the google_compute_instance example usage and paste this into main.tf beneath your current entries.

5. For this simple lab we will not being using a service account, so remove the **google_service_account** block

6. Within the **google_compute_instance** resource block, remove the **service account** sub-block. Also remove the **scratch_disk** sub-block, the **metadata** and the **metadata_startup_script** parameters

7. You should now be left with …

```
resource "google_compute_instance" "default" {
 name         = "my-instance"
 machine_type = "n2-standard-2"
 zone         = "us-central1-a"

 tags = ["foo", "bar"]

 boot_disk {
  initialize_params {
   image = "debian-cloud/debian-11"
   labels = {
    my_label = "value"
   }
  }
 }

 network_interface {
  network = "default"

  access_config {
   // Ephemeral public IP
  }
 }
}
```

Task 2: Run Terraform & Test

1. Run `**terraform init**` ensuring you are in the correct working directory.

2. Run **terraform plan**

3. An error will be returned indicating that we have not specified the project into which we want to provision the resource. This can be entered either as part of the provider block in which case it will apply to all resources, or on a per-resource basis as part of the resource block itself.

4. Update your provider block to include your project. This ..

```
provider "google" {
  project = "<your project id here>"
  # Configuration options
}
```

5. Run **terraform plan** -- review what will be created

6. Run **terraform apply** typing **yes** when prompted. Review output in the CLI. Notice that a persistent terraform.tfstate file now exists in your directory.

7. Select the **terraform.tfstate** file to view its contents

8. Scroll down around line 63 to view the unique id of the compute instance (yours will differ)…

9. Switch to the Console and search for **my-instance** in GCE. Click on the DETAILS tab to view the Instance ID, verifying that this is the instance terraform deployed…

10. Switch back to the IDE and within the instance resource block, change the size from “**ns-standard-2**” to “**e2-small**” to ascertain if this change invokes an in-place modification or a deletion/re-creation.

11. Run `**terraform plan**` Review the output and note that this is an 'update in-place'

```
Plan: 0 to add, 1 to change, 0 to destroy.
```

12. Run `**terraform apply**` typing `**yes**` when prompted.

13. A warning is shown regarding the updating of running instance.

14. Modify your main.tf and include the following line under the zone setting

    allow_stopping_for_update = true

```
machine_type = "e2-small"
zone         = "us-central1-a"
allow_stopping_for_update = true
```

15. Run `**terraform apply**` typing `**yes**` when prompted.

16. As the modification is made, use the console to verify that the id of the instance is unchanged. The update may take several minutes. Refresh the display periodically. Scroll down the DETAILS of the instance to verify its machine type has changed to **e2-small**

17. Switch back to the IDE and within the **boot_disk** sub-block, change the image from "debian-cloud/debian-11" to “**ubuntu-minimal-2404-noble-amd64-v20250725**”

18. Run `**terraform plan**` Review the output and note that this is a destroy and replace..

```
Plan: 1 to add, 0 to change, 1 to destroy.
```

19. Run `**terraform apply**` typing `**yes**` when prompted.

20. Switch to the console and watch (periodically refresh) as the old instance is first destroyed, and a new one is created with a new **id**

Name	my-instance
Instance ID	726716466820170867
Description	None

Task 3: Destroy your deployment

1. Switch back to the IDE and run `**terraform destroy**` review the output and type `**yes**`
2. Switch to the Console and verify the deletion of your instance
3. In the IDE, note how the content of the terraform.tfstate file is reduced.

***** Congratulations, you have completed this lab *****