Chapter 5 Lab: Installing Terraform and OpenTofu

Overview

In this lab, you will gain hands-on experience installing Terraform - an infrastructure as code tool that enables you to safely and predictably provision and manage infrastructure in any cloud. We will provision a Google Compute Engine using Terraform and then move on to implementing OpenTofu, which is a fork of Terraform that is open source, community-driven, and managed by the Linux Foundation.

Pre-Requisites

Ensure you have an Ubuntu 20.04 host. If not, then provision a virtual machine from Google Cloud Platform as per the lab in chapter 3.

Exercise 5.1: Install Terraform

In this exercise, we will install Terraform on our Ubuntu host.

1. Download the public signing key for the package repositories. To get started, execute the following commands:

```
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/hashicorp-archive-keyring.gpg
```

2. Add the appropriate **apt** repository:

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list
```

3. Update the **apt** package index and install Terraform:

```
sudo apt update && sudo apt install terraform
```

4. Verify the Terraform installation.

```
terraform --version
Terraform v1.7.4
on linux_amd64
```

Exercise 5.2: Set up the GCP Service Account and Configure Google Cloud SDK

In this exercise, we will install Google Cloud SDK, configure Google Cloud account to create a service account and use the credentials to create infrastructure using Terraform.

1. Before you install the **gcloud** CLI, update the packages:

```
sudo apt-get update
```

2. Install the **curl** and **apt-transport-https** packages:

```
sudo apt-get install apt-transport-https ca-certificates gnupg curl
```

3. Import the Google Cloud public key:

```
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor
-o /usr/share/keyrings/cloud.google.gpg
```

4. Add the **gcloud** CLI distribution URI as a package source:

```
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list
```

5. Update and install the **gcloud** CLI:

```
sudo apt-get update && sudo apt-get install google-cloud-cli
```

6. After installing the SDK, initialize it by running the following command and follow the prompts:

```
gcloud init

Welcome! This command will take you through the configuration of gcloud.

Your current configuration has been set to: [default]
```

```
You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

Choose the account you would like to use to perform operations for this
configuration:

Follow the prompts and enter your project details accordingly.
```

7. Create a Service Account:

You'll need to create a service account in your Google Cloud project. This service account will be used by applications and scripts to authenticate with Google Cloud APIs.

- Go to the Google Cloud Console: https://console.cloud.google.com/
- Navigate to IAM & Admin > Service accounts.
- Click on "Create service account" and follow the prompts to create a new service account.
- Grant the necessary permissions to the service account. At a minimum, you'll need the "Owner" role, but you can adjust permissions based on your requirements.
- After creating the service account, download the JSON key file associated with it. This file will be used for authentication.

8. Set Up Environment Variables:

Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to point to the JSON key file you downloaded:

```
export GOOGLE_APPLICATION_CREDENTIALS="/root/serviceaccount.json"
```

9. Verify Authentication: To verify that authentication is set up correctly, you can run a simple command using the Google Cloud SDK:

```
gcloud auth list
```

Exercise 5.3: Create a Google Cloud Compute Engine Instance Using Terraform

1. Create a directory to store our Terraform code:

```
mkdir -p ~/code
```

2. Create a Terraform configuration file and save the file as `main.tf` with the following configuration:

```
provider "google" {
  project = "<ADD YOUR PROJECT ID HERE>"
  region  = "us-central1"
  zone    = "us-central1-a"
}

resource "google_compute_instance" "my_instance" {
  name         = "my-instance"
  machine_type = "n1-standard-1"
  zone         = "us-central1-a"

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-10"
    }
  }

  network_interface {
    network = "default"
    access_config {
      // Ephemeral IP
    }
  }
}
```

Note: Replace " ADD YOUR PROJECT ID HERE " with your Google Cloud project ID. You can also customize other parameters like the instance name, machine type, zone, and image.

3. Initialize Terraform, navigate to the directory containing your Terraform configuration file (`main.tf`). Initialize the directory which will download the necessary providers.

```
cd ~/code
terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/google...
- Installing hashicorp/google v5.19.0...
- Installed hashicorp/google v5.19.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.
```

```
Terraform has been successfully initialized!
```

4. Plan the infrastructure; run **`terraform plan`** to see the execution plan. Terraform will show you what resources it will create, update, or delete based on your configuration.

```
terraform plan

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # google_compute_instance.my_instance will be created
  + resource "google_compute_instance" "my_instance" {
      + can_ip_forward       = false
      + cpu_platform         = (known after apply)
      + current_status       = (known after apply)
      + deletion_protection  = false
      + effective_labels     = (known after apply)
      + guest_accelerator    = (known after apply)

<OUTPUT Truncated>
```

5. Apply the Configuration: If the plan looks good, apply the configuration by running **`terraform apply`**. Terraform will create the Compute Engine instance as per your configuration.

```
terraform apply

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # google_compute_instance.my_instance will be created
  + resource "google_compute_instance" "my_instance" {
      + can_ip_forward       = false
      + cpu_platform         = (known after apply)
      + current_status       = (known after apply)
      + deletion_protection  = false

.
.
.
.

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

```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

google_compute_instance.my_instance: Creating...
google_compute_instance.my_instance: Still creating... [10s elapsed]
google_compute_instance.my_instance: Creation complete after 12s
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

6. Verify the Instance: After Terraform finishes applying the configuration, go to the Google Cloud Console to verify that the Compute Engine instance was created successfully.

That's it! You've successfully created a Google Cloud Compute Engine instance using Terraform. You can further extend your configuration to include additional resources or customize the instance properties as needed.

7. Cleanup by deleting the resource with a simple command:

```
terraform destroy

google_compute_instance.my_instance: Refreshing state...
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance]

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # google_compute_instance.my_instance will be destroyed
  - resource "google_compute_instance" "my_instance" {
      - can_ip_forward       = false -> null
      - cpu_platform         = "Intel Haswell" -> null
      - current_status       = "RUNNING" -> null
      - deletion_protection  = false -> null

.
.
..
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.
```

```
  Enter a value: yes

google_compute_instance.my_instance: Destroying...
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance]
google_compute_instance.my_instance: Still destroying...
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance, 10s
elapsed]
google_compute_instance.my_instance: Still destroying...
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance, 20s
elapsed]
google_compute_instance.my_instance: Destruction complete after 21s
```

Exercise 5.4: Setting up OpenTofu

OpenTofu is a fork of Terraform that is open source, community-driven, and managed by the Linux Foundation. OpenTofu is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. OpenTofu can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

1. OpenTofu is available on Snapcraft; we will install it by running:

```
snap install --classic opentofu

opentofu 1.6.2 from OpenTofu Core Team installed
```

2. Verify by executing :

```
tofu -version

OpenTofu v1.6.2
on linux_amd64
+ provider registry.terraform.io/hashicorp/google v5.19.0
```

3. Making use of the same `main.tf` used in the previous exercise, let's initialize and create the Google Cloud compute engine.

```
tofu init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/google...
- Installing hashicorp/google v5.19.0...
```

```
- Installed hashicorp/google v5.19.0 (signed, key ID 0C0AF313E5FD9F80)

Providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://opentofu.org/docs/cli/plugins/signing/

OpenTofu has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

OpenTofu has been successfully initialized!

You may now begin working with OpenTofu. Try running "tofu plan" to see
any changes that are required for your infrastructure. All OpenTofu commands
should now work.

If you ever set or change modules or backend configuration for OpenTofu,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

4. Make a plan and apply to create the infrastructure.

```
tofu plan

OpenTofu used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

OpenTofu will perform the following actions:

  # google_compute_instance.my_instance will be created
  + resource "google_compute_instance" "my_instance" {
      + can_ip_forward       = false
      + cpu_platform         = (known after apply)
      + current_status       = (known after apply)
      + deletion_protection  = false
.
.
.
Plan: 1 to add, 0 to change, 0 to destroy.

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________


Note: You didn't use the -out option to save this plan, so OpenTofu can't
guarantee to take exactly these actions if you run "tofu apply" now.
```

```
tofu apply

OpenTofu used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

OpenTofu will perform the following actions:

  # google_compute_instance.my_instance will be created
  + resource "google_compute_instance" "my_instance" {
      + can_ip_forward       = false
      + cpu_platform         = (known after apply)
      + current_status       = (known after apply)
      + deletion_protection  = false
..
.
.
.
Do you want to perform these actions?
  OpenTofu will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

google_compute_instance.my_instance: Creating...
google_compute_instance.my_instance: Still creating... [10s elapsed]
google_compute_instance.my_instance: Creation complete after 12s
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Congratulations, you have successfully created the infrastructure using OpenTofu.
```

5. Clean up destroying the infrastructure created with a simple command.

```
tofu destroy
Do you really want to destroy all resources?
  OpenTofu will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

google_compute_instance.my_instance: Destroying...
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance]
google_compute_instance.my_instance: Still destroying...
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance, 10s
elapsed]
```

```
google_compute_instance.my_instance: Still destroying...
[id=projects/calm-bliss-375608/zones/us-central1-a/instances/my-instance, 20s
elapsed]
google_compute_instance.my_instance: Destruction complete after 21s

Destroy complete! Resources: 1 destroyed.
```