# Getting Started with Terraform

In this lab, you will install and use the CLI version of Terraform to work through the lifecycle of a simple infrastructural configuration. The development environment you will use is Google Cloud Shell, an interactive command-line interface facilitated through an automatically provisioned Google Compute Engine instance. Though this environment does deliver some convenience, the installation and workflow principles covered in this lab are applicable on other platforms as well.

**This lab will take approximately 30 minutes to complete.**

## Prerequisites

In order to perform this lab, you will need:

* A Google Cloud Platform account. If you don’t have one, follow [this link](https://console.cloud.google.com/freetrial) to create one for free.
* A running instance of Google Cloud Shell. Once logged in to the Google Cloud Console using your Google Cloud Platform account, click the **Activate Cloud Shell** icon ( ) at the top-right corner to start the shell.

## Installing Terraform

In this section, you will install Terraform into your Google Cloud Shell environment.

1. Download the Terraform archive  
   In the Google Cloud Shell, execute the following command to download a .zip file containing the most recent version of the Terraform binary targeting 64-bit Linux systems (this is the underlying compute environment providing your Google Cloud shell). The command below references version 0.12.18, which represents the most current version at the time of this writing. If desired, visit the Terraform release page (<https://releases.hashicorp.com/terraform/>) to see an up-to-date list of Terraform releases, and substitute a more recent version in the following command:

wget https://releases.hashicorp.com/terraform/0.12.18/terraform\_0.12.18\_linux\_amd64.zip

1. Extract the archive  
   Execute the following command to extract the contents of the archive you downloaded:

unzip terraform\_0.12.18\_linux\_amd64.zip

This will extract a stand-alone binary that can be installed in any place desired.

1. Install the binary  
   Copy the **terraform** binary to any place readily accessible by your **PATH** setting. In the case of a Google Cloud Shell environment, *$HOME/gopath/bin* is a reasonable choice. Execute the following command to create this directory and relocate the binary there:

mkdir -p $HOME/gopath/bin && mv terraform $\_

## Creating a Terraform Configuration

In this section, you will create an HCL file that provisions a tiny compute instance in the **us-east4** region (located in Northern Virginia, USA). This instance will boot from a standard Debian 9 boot image.

**NOTE:** because this lab is being performed in a Google Cloud Shell environment, API authentication is already set up. In a different environment, you would need to take additional steps to address authentication requirements – but to keep the learning exercise simple, this task is overlooked.

1. Create a Terraform configuration file  
   Create a Terraform configuration file called **test.tf**:

cat >> test.tf << EOF  
resource "google\_compute\_instance" "vm\_instance" {  
 name = "terraform-instance"  
 machine\_type = "f1-micro"  
 zone = "us-east4-b"  
 boot\_disk {  
 initialize\_params {  
 image = "debian-cloud/debian-9"  
 }  
 }  
 network\_interface {  
 network = "default"  
 access\_config {  
 }  
 }  
}  
EOF

1. Initialize Terraform  
   Prepare Terraform to process this configuration – a step which involves downloading plugins and setting up a context for Terraform to monitor state – by executing the following command:

terraform init

1. Inspect running instances  
   Prior to doing anything further, execute the following command to list any instances that may be running (at this point there should be none). This command will also prompt you to enable the compute API if it isn’t already – if this is the case, choose **y**.

gcloud compute instances list

## Provisioning the Instance

With an initialized Terraform configuration in place, provisioning operations may now be performed.

1. Apply the configuration  
   Instruct Terraform to devise and execute a plan to achieve the state described in your configuration file by executing the following command:

terraform apply

Terraform will summarize the actions it will perform and prompt you to continue. Respond with **yes** to perform those actions.

1. Inspect the results  
   Since provisioning a single **f1-micro** compute instance takes little time, the operation should complete quickly. Once it completes, execute the following command to view your running instance:

gcloud compute instances list

1. Connect to the instance  
   Execute the following command to connect to your running instance via ssh:

gcloud compute ssh terraform-instance --zone=us-east4-b

**NOTE:** you will need to respond to prompts as requested to initialize your ssh configuration the first time you connect to your instance.

1. Modify the instance environment  
   Make a small but noticeable change to your instance’s run-time environment. Execute the following command to change the login message to a custom string:

sudo sh -c 'echo "Hello from my instance" > /etc/motd'

1. Return to the Google Cloud Shell  
   When you are done inspecting the environment on the newly provisioned instance, issue the **exit** command to return to your Google Cloud Shell prompt. If desired, connect back to the instance to view the effect of the changes you made; you should now see an updated login message.

## Applying Modifications

Inevitably, your configuration requirements will change over time. You may need to add new resources, or adjust parameters of existing ones. In this section, you will adjust the machine type of your provisioned instance to a value that will provide more compute power.

1. Modify the configuration  
   Open the **test.tf** file and modify the value for **machine\_type**. Change the value from *f1-micro* to *g1-small*. Also, add the following parameter setting in a new line below your **machine\_type** declaration; this will allow Terraform to stop any running instances it needs to in order to satisfy the configuration change:

allow\_stopping\_for\_update = "true"

1. Apply the changes  
   With the configuration changes in place, execute the following command to apply them:

terraform apply

As before, respond with **yes** when prompted.

1. Inspect the results  
   Applying changes this time will take a little longer, as your instance must first be stopped before a new updated instance can be created. Once it completes, execute the following command to connect to the newly updated instance:

gcloud compute ssh terraform-instance --zone=us-east4-b

Notice that the custom login message from the previous section has been retained. This is evidence that Terraform destroyed and recreated only the resources needed to satisfy the change – in this case the compute instance. The underlying disk from which the instance boots was retained as there was no need to destroy and recreate that resource.

1. Return to the Google Cloud Shell  
   When you are done inspecting the environment on the newly provisioned instance, issue the **exit** command to return to your Google Cloud Shell prompt.

## Cleaning Up

1. Destroy resources  
   To remove all resources that Terraform created, execute the following command:

terraform destroy

As with the **apply** command, Terraform summarizes the actions it will perform to delete your configuration. Respond with **yes** to perform those actions.

1. Inspect the results  
   Execute the following command to verify that the instance is gone:

gcloud compute instances list

## Summary

In this lab, you worked through the basic lifecycle of a small infrastructural configuration by provisioning an instance, applying changes to VM parameters, and ultimately removing the instance. Even with the instance removed, you now know that the environment can be recreated with a single command – illustrating the power and convenience of an IaC tool like Terraform.