*[favorite\_border](https://google-run.qwiklabs.com/favorite.json?item_id=1208&item_type=CatalogItem)*

Add to favorites



**—/100**

**Checkpoints**

Create a VM instance in us-central1-a zone with Terraform

Check my progress

/ 100

**Terraform Fundamentals**

35 minutes 5 Credits

Rate Lab

**GSP156**

**Introduction**

Terraform enables you to safely and predictably create, change, and improve infrastructure. It is an open source tool that codifies APIs into declarative configuration files that can be shared amongst team members, treated as code, edited, reviewed, and versioned.

**Objectives:**

* Getting Started with Terraform in Google Cloud.
* Install Terraform from Installation Binaries.
* Create a VM instance infrastructure using Terraform.

**Setup and Requirements**

**Qwiklabs setup**

**Before you click the Start Lab button**

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

**What you need**

To complete this lab, you need:

* Access to a standard internet browser (Chrome browser recommended).
* Time to complete the lab.

**Note:** If you already have your own personal Google Cloud account or project, do not use it for this lab.

**Note:** If you are using a Pixelbook, open an Incognito window to run this lab.

**How to start your lab and sign in to the Google Cloud Console**

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.
2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

***Tip:*** Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**.

1. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

***Important:*** You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

1. Click through the subsequent pages:
   * Accept the terms and conditions.
   * Do not add recovery options or two-factor authentication (because this is a temporary account).
   * Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

**Note:** You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.

**Google Cloud Shell**

**Activate Cloud Shell**

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.

Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT\_ID*. For example:

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

gcloud auth list

(Output)

Credentialed accounts:

- <myaccount>@<mydomain>.com (active)

(Example output)

Credentialed accounts:

- google1623327\_student@qwiklabs.net

You can list the project ID with this command:

gcloud config list project

(Output)

[core]

project = <project\_ID>

(Example output)

[core]

project = qwiklabs-gcp-44776a13dea667a6

For full documentation of gcloud see the [gcloud command-line tool overview](https://cloud.google.com/sdk/gcloud).

**What is Terraform?**

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing, popular service providers as well as custom in-house solutions.

Configuration files describe to Terraform the components needed to run a single application or your entire datacenter. Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

The infrastructure Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

**Key Features**

**Infrastructure as Code**

Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.

**Execution Plans**

Terraform has a "planning" step where it generates an execution plan. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.

**Resource Graph**

Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform builds infrastructure as efficiently as possible, and operators get insight into dependencies in their infrastructure.

**Change Automation**

Complex changesets can be applied to your infrastructure with minimal human interaction. With the previously mentioned execution plan and resource graph, you know exactly what Terraform will change and in what order, avoiding many possible human errors.

**Verifying Terraform Installation**

Terraform comes pre-installed in Cloud Shell.

Verify that Terraform is available:

terraform

You should see help output similar to this:

Usage: terraform [--version] [--help] <command> [args]

The available commands for execution are listed below.

The most common, useful commands are shown first, followed by

less common or more advanced commands. If you're just getting

started with Terraform, stick with the common commands. For the

other commands, please read the help and docs before usage.

Common commands:

apply Builds or changes infrastructure

console Interactive console for Terraform interpolations

destroy Destroy Terraform-managed infrastructure

env Workspace management

fmt Rewrites config files to canonical format

get Download and install modules for the configuration

graph Create a visual graph of Terraform resources

import Import existing infrastructure into Terraform

init Initialize a Terraform working directory

output Read an output from a state file

plan Generate and show an execution plan

providers Prints a tree of the providers used in the configuration

push Upload this Terraform module to Atlas to run

refresh Update local state file against real resources

show Inspect Terraform state or plan

taint Manually mark a resource for recreation

untaint Manually unmark a resource as tainted

validate Validates the Terraform files

version Prints the Terraform version

workspace Workspace management

All other commands:

debug Debug output management (experimental)

force-unlock Manually unlock the terraform state

state Advanced state management

**Build Infrastructure**

With Terraform installed, you can dive right in and start creating some infrastructure.

**Configuration**

The set of files used to describe infrastructure in Terraform is simply known as a Terraform configuration. We're going to write our first configuration now to launch a single VM instance.

The format of the configuration files is [documented here](https://www.terraform.io/docs/configuration/index.html). We recommend using JSON for creating configuration files.

Create a configuration an instance.tf file with your favourite editor like vim, nano etc.:

nano instance.tf

Add the following content in file, Make sure to replace <PROJECT\_ID> with the Google Cloud project ID:

resource "google\_compute\_instance" "default" {

project = "<PROJECT\_ID>"

name = "terraform"

machine\_type = "n1-standard-1"

zone = "us-central1-a"

boot\_disk {

initialize\_params {

image = "debian-cloud/debian-9"

}

}

network\_interface {

network = "default"

access\_config {

}

}

}

**Ctrl + X** to save the file.

This is a complete configuration that Terraform is ready to apply. The general structure should be intuitive and straightforward.

The "resource" block in the instance.tf file defines a resource that exists within the infrastructure. A resource might be a physical component such as an VM instance.

The resource block has two strings before opening the block: the **resource type** and the **resource name**. For this lab the resource type is google\_compute\_instance and the name is terraform. The prefix of the type maps to the provider: google\_compute\_instance automatically tells Terraform that it is managed by the Google provider.

Within the resource block itself is the configuration needed for the resource.

Verify your new file has been added and that there are no other \*.tf files in your directory, since Terraform loads all of them:

ls

**Initialization**

The first command to run for a new configuration -- or after checking out an existing configuration from version control -- is terraform init. This will initialize various local settings and data that will be used by subsequent commands.

Terraform uses a plugin-based architecture to support the numerous infrastructure and service providers available. Each "Provider" is its own encapsulated binary distributed separately from Terraform itself. The terraform init command will automatically download and install any Provider binary for the providers to use within the configuration, which in this case is just the Google provider.

terraform init

The Google provider plugin is downloaded and installed in a subdirectory of the current working directory, along with various other bookkeeping files. You will see an "Initializing provider plugins" message. Terraform knows that you're running from a Google project and is getting Google resources.

Downloading plugin for provider "google" (2.12.0)...

The output specifies which version of the plugin is being installed, and suggests specifying this version in future configuration files to ensure that terraform init will install a compatible version.

The terraform plan command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

terraform plan

This command is a convenient way to check whether the execution plan for a set of changes matches your expectations without making any changes to real resources or to the state. For example, terraform plan might be run before committing a change to version control, to create confidence that it will behave as expected.

**Note:** The optional -out argument can be used to save the generated plan to a file for later execution with terraform apply.

**Apply Changes**

In the same directory as the instance.tf file you created, run terraform apply.

terraform apply

This output shows the Execution Plan, which describes the actions Terraform will take in order to change real infrastructure to match the configuration. The output format is similar to the diff format generated by tools like Git.

There is a + next to google\_compute\_instance.terraform, which means that Terraform will create this resource. Beneath that you'll see the attributes that will be set. When the value displayed is <computed>, it means that the value won't be known until the resource is created.

**Example Output:**

An execution plan has been generated and is shown below.

Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# google\_compute\_instance.default will be created

+ resource "google\_compute\_instance" "default" {

+ can\_ip\_forward = false

+ cpu\_platform = (known after apply)

+ deletion\_protection = false

+ guest\_accelerator = (known after apply)

+ id = (known after apply)

+ instance\_id = (known after apply)

+ label\_fingerprint = (known after apply)

+ machine\_type = "n1-standard-1"

+ metadata\_fingerprint = (known after apply)

+ name = "terraform"

+ project = "qwiklabs-gcp-42390cc9da8a4c4b"

+ self\_link = (known after apply)

+ tags\_fingerprint = (known after apply)

+ zone = "us-central1-a"

+ boot\_disk {

+ auto\_delete = true

+ device\_name = (known after apply)

+ disk\_encryption\_key\_sha256 = (known after apply)

+ kms\_key\_self\_link = (known after apply)

+ source = (known after apply)

+ initialize\_params {

+ image = "debian-cloud/debian-9"

+ labels = (known after apply)

+ size = (known after apply)

+ type = (known after apply)

}

}

+ network\_interface {

+ address = (known after apply)

+ name = (known after apply)

+ network = "default"

+ network\_ip = (known after apply)

+ subnetwork = (known after apply)

+ subnetwork\_project = (known after apply)

+ access\_config {

+ assigned\_nat\_ip = (known after apply)

+ nat\_ip = (known after apply)

+ network\_tier = (known after apply)

}

}

+ scheduling {

+ automatic\_restart = (known after apply)

+ on\_host\_maintenance = (known after apply)

+ preemptible = (known after apply)

+ node\_affinities {

+ key = (known after apply)

+ operator = (known after apply)

+ values = (known after apply)

}

}

}

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:

If the plan was created successfully, Terraform will now pause and wait for approval before proceeding. In a production environment, if anything in the Execution Plan seems incorrect or dangerous, it's safe to abort here. No changes have been made to your infrastructure.

For this case the plan looks acceptable, so type yes at the confirmation prompt to proceed.

Executing the plan will take a few minutes since Terraform waits for the VM instance to become available

After this, Terraform is all done!

**Test Completed Task**

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Create a VM instance in us-central1-a zone with Terraform.

In the Console, go to **Compute Engine** > **VM instances** to see the created VM instance.

Terraform has written some data into the terraform.tfstate file. This state file is extremely important; it keeps track of the IDs of created resources so that Terraform knows what it is managing.

You can inspect the current state using terraform show:

terraform show

**Example Output:**

# google\_compute\_instance.default:

resource "google\_compute\_instance" "default" {

can\_ip\_forward = false

cpu\_platform = "Intel Haswell"

deletion\_protection = false

guest\_accelerator = []

id = "terraform"

instance\_id = "3408292216444307052"

label\_fingerprint = "42WmSpB8rSM="

machine\_type = "n1-standard-1"

metadata\_fingerprint = "s6I5s2tjfKw="

name = "terraform"

project = "qwiklabs-gcp-42390cc9da8a4c4b"

self\_link = "https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-42390cc9da8a4c4b/zones/us-central1-a/instances/terraform"

tags\_fingerprint = "42WmSpB8rSM="

zone = "us-central1-a"

boot\_disk {

auto\_delete = true

device\_name = "persistent-disk-0"

source = "https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-42390cc9da8a4c4b/zones/us-central1-a/disks/terraform"

....

You can see that by creating this resource, you've also gathered a lot of information about it. These values can be referenced to configure additional resources or outputs.

If you want to go review the execution plan after it's been applied, you can use terraform plan command:

terraform plan

Congratulations! You've built your first infrastructure with Terraform. You've seen the configuration syntax, an example of a basic execution plan, and understand the state file.

**Test your Understanding**

Below are multiple choice questions to reinforce your understanding of this lab's concepts. Answer them to the best of your abilities.

**Congratulations**

**Finish Your Quest**

This self-paced lab is part of the [Managing Cloud Infrastructure with Terraform](https://google.qwiklabs.com/quests/44) Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge public and link to them in your online resume or social media account. [Enroll in this Quest](https://google.qwiklabs.com/learning_paths/44/enroll) and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](https://google.qwiklabs.com/catalog).

**Take Your Next Lab**

Continue your Quest with [Using a NAT Gateway with Kubernetes Engine](https://google.qwiklabs.com/catalog_lab/1009), or check out these suggestions:

* [Custom Providers with Terraform](https://google.qwiklabs.com/catalog_lab/1014)
* [HTTPS Content-Based Load Balancer with Terraform](https://google.qwiklabs.com/catalog_lab/1011)

**Next Steps / Learn More**

* [Terraform Community](https://www.terraform.io/community.html)
* [Terraform Google Examples](https://github.com/GoogleCloudPlatform/terraform-google-examples)

**Google Cloud Training & Certification**

...helps you make the most of Google Cloud technologies. [Our classes](https://cloud.google.com/training/courses) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](https://cloud.google.com/certification/) help you validate and prove your skill and expertise in Google Cloud technologies.

**Manual Last Updated August 6, 2019**

**Lab Last Tested August 6, 2019**

Copyright 2020 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated