**Build Infrastructure with Terraform on Google Cloud: Challenge Lab**

Experiment Lab schedule 1 hour 30 minutes universal\_currency\_alt5 Creditsshow\_chart Intermediate

**GSP345**



**Overview**

In a challenge lab you’re given a scenario and a set of tasks. Instead of following step-by-step instructions, you will use the skills learned from the labs in the course to figure out how to complete the tasks on your own! An automated scoring system (shown on this page) will provide feedback on whether you have completed your tasks correctly.

When you take a challenge lab, you will not be taught new Google Cloud concepts. You are expected to extend your learned skills, like changing default values and reading and researching error messages to fix your own mistakes.

To score 100% you must successfully complete all tasks within the time period!

This lab is recommended for students who have enrolled in the [Build Infrastructure with Terraform on Google Cloud](https://www.cloudskillsboost.google/course_templates/636) course. Are you ready for the challenge?

Topics tested:

* Import existing infrastructure into your Terraform configuration.
* Build and reference your own Terraform modules.
* Add a remote backend to your configuration.
* Use and implement a module from the Terraform Registry.
* Re-provision, destroy, and update infrastructure.
* Test connectivity between the resources you've created.

**Setup and requirements**

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 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.

To complete this lab, you need:

* Access to a standard internet browser (Chrome browser recommended).

**Note:** Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

* Time to complete the lab---remember, once you start, you cannot pause a lab.

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

**Challenge scenario**

You are a cloud engineer intern for a new startup. For your first project, your new boss has tasked you with creating infrastructure in a quick and efficient manner and generating a mechanism to keep track of it for future reference and changes. You have been directed to use [Terraform](https://www.terraform.io/) to complete the project.

For this project, you will use Terraform to create, deploy, and keep track of infrastructure on the startup's preferred provider, Google Cloud. You will also need to import some mismanaged instances into your configuration and fix them.

In this lab, you will use Terraform to import and create multiple VM instances, a VPC network with two subnetworks, and a firewall rule for the VPC to allow connections between the two instances. You will also create a Cloud Storage bucket to host your remote backend.

**Note:**At the end of every section, plan and apply your changes to allow your work to be successfully verified. Since we will be updating many terraform files in this lab make sure to use the correct file path and maintain the correct indentation.

**Task 1. Create the configuration files**

1. In Cloud Shell, create your Terraform configuration files and a directory structure that resembles the following:

main.tf

variables.tf

modules/

└── instances

├── instances.tf

├── outputs.tf

└── variables.tf

└── storage

├── storage.tf

├── outputs.tf

└── variables.tf

Copied!

content\_copy

1. Fill out the variables.tf files in the root directory and within the modules. Add three variables to each file: region, zone, and project\_id. For their default values, use , <filled in at lab start>, and your Google Cloud Project ID.

**Note:**You should use these variables anywhere applicable in your resource configurations.

1. Add the Terraform block and the [Google Provider](https://registry.terraform.io/providers/hashicorp/google/latest/docs) to the main.tf file. Verify the **zone** argument is added along with the **project** and **region** arguments in the Google Provider block.
2. Initialize Terraform.

**Task 2. Import infrastructure**

1. In the Google Cloud Console, on the **Navigation menu**, click **Compute Engine > VM Instances**. Two instances named **tf-instance-1** and **tf-instance-2** have already been created for you.

**Note:**by clicking on one of the instances, you can find its **Instance ID**, **boot disk image**, and **machine type**. These are all necessary for writing the configurations correctly and importing them into Terraform.

1. [Import](https://www.terraform.io/docs/cli/commands/import.html#example-import-into-module) the existing instances into the **instances** module. To do this, you will need to follow these steps:

* First, add the module reference into the main.tf file then re-initialize Terraform.
* Next, write the [resource configurations](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/compute_instance) in the instances.tf file to match the pre-existing instances.
  + Name your instances tf-instance-1 and tf-instance-2.
  + For the purposes of this lab, the resource configuration should be as minimal as possible. To accomplish this, you will only need to include the following additional arguments in your configuration: machine\_type, boot\_disk, network\_interface, metadata\_startup\_script, and allow\_stopping\_for\_update. For the last two arguments, use the following configuration as this will ensure you won't need to recreate it:
* metadata\_startup\_script = <<-EOT
* #!/bin/bash
* EOT

allow\_stopping\_for\_update = true

Copied!

content\_copy

* Once you have written the resource configurations within the module, use the terraform import command to import them into your **instances** module.

1. Apply your changes. Note that since you did not fill out all of the arguments in the entire configuration, the apply will **update the instances in-place**. This is fine for lab purposes, but in a production environment, you should make sure to fill out all of the arguments correctly before importing.

Click *Check my progress* to verify the objective.

Import infrastructure.

Check my progress

**Task 3. Configure a remote backend**

1. Create a [Cloud Storage bucket resource](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/storage_bucket) inside the storage module. For the bucket **name**, use **Bucket Name**. For the rest of the arguments, you can simply use:
   * location = "US"
   * force\_destroy = true
   * uniform\_bucket\_level\_access = true

**Note:**You can optionally add output values inside of the outputs.tf file.

1. Add the module reference to the main.tf file. Initialize the module and apply the changes to create the bucket using Terraform.
2. Configure this storage bucket as the [remote backend](https://www.terraform.io/docs/language/settings/backends/gcs.html) inside the main.tf file. Be sure to use the **prefix** terraform/state so it can be graded successfully.
3. If you've written the configuration correctly, upon init, Terraform will ask whether you want to copy the existing state data to the new backend. Type yes at the prompt.

Click *Check my progress* to verify the objective.

Configure a remote backend.

Check my progress

**Task 4. Modify and update infrastructure**

1. Navigate to the **instances** module and modify the **tf-instance-1** resource to use an e2-standard-2 machine type.
2. Modify the **tf-instance-2** resource to use an e2-standard-2 machine type.
3. Add a third instance resource and name it **Instance Name**. For this third resource, use an e2-standard-2 machine type. Make sure to change the machine type to e2-standard-2 **to all the three instances**.
4. Initialize Terraform and apply your changes.

**Note:**Optionally, you can add output values from these resources in the outputs.tf file within the module.

Click *Check my progress* to verify the objective.

Modify and update infrastructure.

Check my progress

**Task 5. Destroy resources**

1. Destroy the third instance **Instance Name** by removing the resource from the configuration file. After removing it, initialize terraform and apply the changes.

Click *Check my progress* to verify the objective.

Destroy resources.

Check my progress

**Task 6. Use a module from the Registry**

1. In the Terraform Registry, browse to the [Network Module](https://registry.terraform.io/modules/terraform-google-modules/network/google/6.0.0).
2. Add this module to your main.tf file. Use the following configurations:

* Use version 6.0.0 (different versions might cause compatibility errors).
* Name the VPC **VPC Name**, and use a **global** routing mode.
* Specify **2** subnets in the region, and name them subnet-01 and subnet-02. For the subnets arguments, you just need the **Name**, **IP**, and **Region**.
* Use the IP 10.10.10.0/24 for subnet-01, and 10.10.20.0/24 for subnet-02.
* You do **not** need any secondary ranges or routes associated with this VPC, so you can omit them from the configuration.

1. Once you've written the module configuration, initialize Terraform and run an apply to create the networks.
2. Next, navigate to the instances.tf file and update the configuration resources to connect **tf-instance-1** to subnet-01 and **tf-instance-2** to subnet-02.

**Note:**Within the instance configuration, you will need to update the **network** argument to VPC Name, and then add the **subnetwork** argument with the correct subnet for each instance.

Click *Check my progress* to verify the objective.

Use a module from the Registry.

Check my progress

**Task 7. Configure a firewall**

* Create a [firewall rule](https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/compute_firewall) resource in the main.tf file, and name it **tf-firewall**.
  + This firewall rule should permit the **VPC Name** network to allow ingress connections on *all* IP ranges (0.0.0.0/0) on **TCP port 80**.
  + Make sure you add the source\_ranges argument with the correct IP range (0.0.0.0/0).
  + Initialize Terraform and apply your changes.

**Note:**To retrieve the required network argument, you can inspect the state and find the **ID** or **self\_link** of the google\_compute\_network resource you created. It will be in the form projects/PROJECT\_ID/global/networks/VPC Name.

Click *Check my progress* to verify the objective.

Configure a firewall.

Check my progress

**Congratulations!**

If you managed to import, create, modify, and configure the architecture within the allotted time, you've shown great competency in your Terraform skills! In this lab, you started by importing two pre-configured VMs into Terraform and creating a Cloud Storage bucket to configure your backend. You then added another instance and practiced modifying and updating the resource configurations within your modules. Lastly, you used a module from the Terraform Registry to create a VPC with two subnets, connected the instances to it, and created a firewall rule to allow connections between them.



Earn your next skill badge

This self-paced lab is part of the [Build Infrastructure with Terraform on Google Cloud](https://www.cloudskillsboost.google/course_templates/636) skill badge course. Completing this skill badge course earns you the badge above, to recognize your achievement. Share your badge on your resume and social platforms, and announce your accomplishment using #GoogleCloudBadge.

Next steps

Now that you have shown competency in Terraform, you may want to register for the [HashiCorp Infrastructure Automation Certification](https://www.hashicorp.com/certification/terraform-associate" \t "_blank). The Terraform Associate certification is for Cloud Engineers specializing in operations, IT, or development who know the basic concepts and skills associated with open source HashiCorp Terraform. Candidates will be best prepared for this exam if they have professional experience using Terraform in production, but performing the exam objectives in a personal demo environment may also be sufficient. So, if you want to take your skills to the next level and verify your infrastructure automation abilities, we recommend that you review the [study guide](https://learn.hashicorp.com/tutorials/terraform/associate-study?in=terraform/certification) and the [exam review](https://learn.hashicorp.com/tutorials/terraform/associate-review?in=terraform/certification), and then [register for the exam](https://hashicorp-certifications.zendesk.com/hc/en-us/articles/360049382552) when you're ready. Good luck!

Google Cloud training and 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 May 14, 2024**

**Lab Last Tested May 14, 2024**

Copyright 2024 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.

[navigate\_beforePrevious](https://www.cloudskillsboost.google/course_sessions/14498931/labs/464835)

Nextnavigate\_next