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.

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 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.

<u>Procedure in Detail</u>

Task 1. Create the configuration files
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
```

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’. And Google Cloud Project ID.
Note: You should use these variables anywhere applicable in your resource configurations.
Add the Terraform block and the Google Provider to the main.tf file.
Verify the zone argument is added along with the project and region arguments in the Google Provider block.

*Initialize Terraform.

Task 2. Import infrastructure
In the Google Cloud Console, Two instances named tf-instance-1 and tf-instance-2 have already been created in th VM.

Note the Instance ID, boot disk image, and machine type. These are all needed for writing the configurations correctly and importing them into Terraform.

Import the existing instances into the instances module:
First, add the module reference into the **main.tf** file then *re-initialize* Terraform.
Next, write the resource configurations in the **instances.tf** file to match the pre-existing instances.
Name the instances **tf-instance-1** and **tf-instance-2**.

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

Apply your changes.

Task 3. Configure a remote backend

Create a Cloud Storage bucket resource inside the storage module. For the bucket name, use **tf-bucket-357397**. For the rest of the arguments, you can simply use:

```
location = "US"
force_destroy = true
uniform_bucket_level_access = true
```

Add the module reference to the main.tf file. Initialize the module and apply the changes to create the bucket using Terraform.

Configure this storage bucket as the remote backend inside the main.tf file. Be sure to use the prefix terraform/state so it can be graded successfully.

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.

Task 4. Modify and update infrastructure
Navigate to the instances module and modify the **tf-instance-1** resource to use an ‘e2-standard-2’ machine type.

Modify the **tf-instance-2** resource to use an ‘e2-standard-2’ machine type.

Add a **third** instance resource and name it ‘tf-instance-187533’.
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.

Initialize Terraform and apply your changes.

Task 5. Destroy resources

Destroy the **third** instance **tf-instance-187533** by removing the resource from the configuration file.
Initialize terraform and apply the changes after removing it.

Task 6. Use a module from the Registry
In the Terraform Registry, browse to the Network Module.

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, **tf-VPC-460994**, 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.

Initialize Terraform and run an apply to create the networks.

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’.

Task 7. Configure a firewall

-Create a firewall rule resource in the main.tf file, and name it ‘tf-firewall’.
: This firewall rule should permit the VPC, **tf-VPC-460994** 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