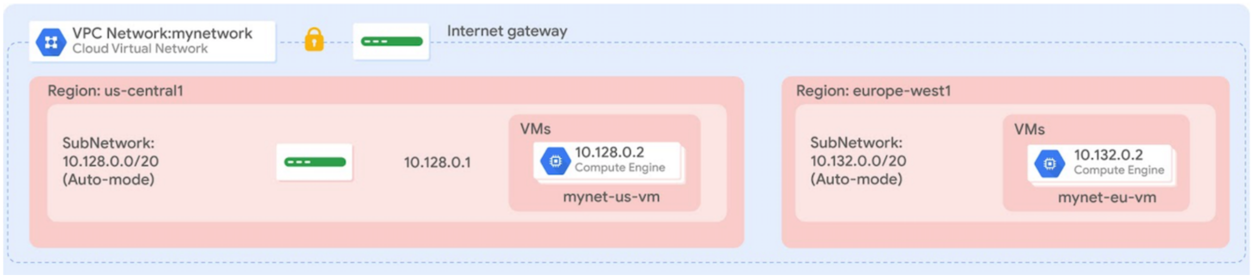
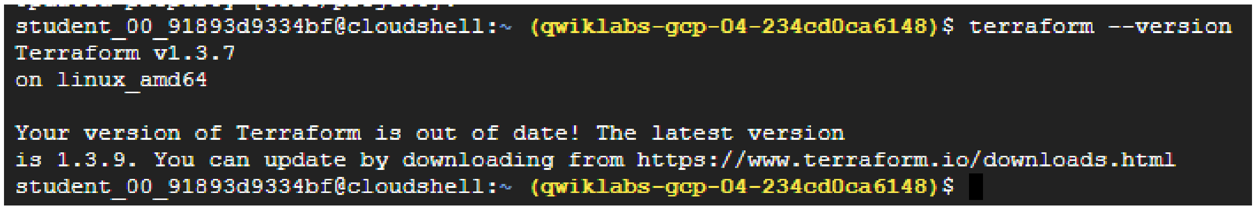
Terraform enables you to safely and predictably create, change, and improve infrastructure. It is an open-source tool that codifies APIs into declarative configuration files that can be shared among team members, treated as code, edited, reviewed, and versioned.
In this lab, you create a Terraform configuration with a module to automate the deployment of Google Cloud infrastructure. Specifically, you deploy one auto mode network with a firewall rule and two VM instances, as shown in this diagram:



In this lab, you learn how to perform the following tasks:

- Create a configuration for an auto mode network
- Create a configuration for a firewall rule
- Create a module for VM instances
- Create and deploy a configuration
- Verify the deployment of a configuration

Configure your Cloud Shell environment to use Terraform.
Install Terraform
Terraform is now integrated into Cloud Shell. Verify which version is installed.
terraform --version



```
student_00_91893d9334bf@cloudshell:~ (qwiklabs-gcp-04-234cd0ca6148)$ terraform --version
Terraform v1.3.7
on linux_amd64

Your version of Terraform is out of date! The latest version
is 1.3.9. You can update by downloading from https://www.terraform.io/downloads.html
student_00_91893d9334bf@cloudshell:~ (qwiklabs-gcp-04-234cd0ca6148)$
```

4. To create a directory for your Terraform configuration, run the following command:

```
mkdir tfinfra
```

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. Initialize Terraform by setting Google as the provider.

1. To create a new file inside **tfinfra** folder, `right-click` on **tfinfra** folder and then click **New File**.
2. Name the new file **provider.tf**, and then open it.
3. Copy the code into `provider.tf`:

```
provider "google" {}
```

4. To initialize Terraform, run the following command:

```
cd tfinfra
```

```
terraform init
```
Create the auto mode network **mynetwork** along with its firewall rule and two VM instances (**mynet_us_vm** and **mynet_eu_vm**).

Configure mynetwork

Create a new configuration, and define **mynetwork**.

1. To create a new file inside **tfinfra**, `right-click` on **tfinfra** folder and then click **New File**.
2. Name the new file **mynetwork.tf**, and then open it.
3. Copy the following base code into `mynetwork.tf`:

```
# Create the mynetwork network
resource [RESOURCE_TYPE] "mynetwork" {
name = [RESOURCE_NAME]
# RESOURCE properties go here
}
```
This base template is a great starting point for any Google Cloud resource. The **name** field allows you to name the resource, and the **type** field allows you to specify the Google Cloud resource that you want to create. You can also define properties, but these are optional for some resources.

4. In `mynetwork.tf`, replace `[RESOURCE_TYPE]` with `"google_compute_network"` (with the quotes).

Note: The **google_compute_network** resource is a VPC network.

5. In `mynetwork.tf`, replace `[RESOURCE_NAME]` with `"mynetwork"` (with the quotes).
6. Add the following property to `mynetwork.tf`:

```
auto_create_subnetworks = "true"
```
By definition, an auto mode network automatically creates a subnetwork in each region. Therefore, you are setting **auto_create_subnetworks** to **true**.

7. Verify that **mynetwork.tf** file look like this:

```
# Create the mynetwork network
resource "google_compute_network" "mynetwork" {
name = "mynetwork"
# RESOURCE properties go here
auto_create_subnetworks = "true"
}
```
Define a firewall rule to allow HTTP, SSH, RDP, and ICMP traffic on mynetwork.

1. Add the following base code to `mynetwork.tf`:

```
# Add a firewall rule to allow HTTP, SSH, RDP and ICMP traffic on
mynetwork
resource [RESOURCE_TYPE] "mynetwork-allow-http-ssh-rdp-icmp" {
name = [RESOURCE_NAME]
# RESOURCE properties go here
}
```

2. In `mynetwork.tf`, replace `[RESOURCE_TYPE]` with `"google_compute_firewall"` (with the quotes).

Note: The **google_compute_firewall** resource is a firewall rule.

3. In `mynetwork.tf`, replace `[RESOURCE_NAME]` with `"mynetwork-allow-http-ssh-rdp-icmp"` (with the quotes).
4. Add the following property to `mynetwork.tf`:

```
network = google_compute_network.mynetwork.self_link
```

Note: Because this firewall rule depends on its network, you are using the **google_compute_network.mynetwork.self_link** reference to instruct Terraform to resolve these resources in a dependent order. In this case, the network is created before the firewall rule.

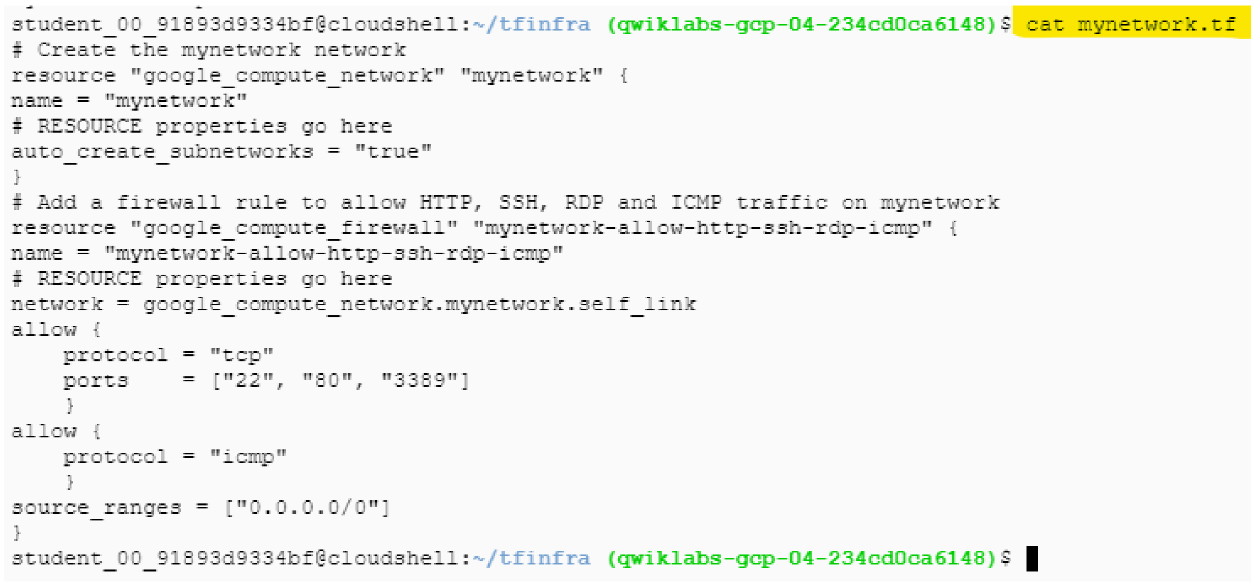
5. Add the following properties to `mynetwork.tf`:

```
allow {
    protocol = "tcp"
    ports    = ["22", "80", "3389"]
    }
allow {
    protocol = "icmp"
    }
source_ranges = ["0.0.0.0/0"]
```

The list of **allow** rules specifies which protocols and ports are permitted.

6. Verify that your `mynetwork.tf` file look like this:

```
# Create the mynetwork network
resource "google_compute_network" "mynetwork" {
name = "mynetwork"
# RESOURCE properties go here
auto_create_subnetworks = "true"
}
# Add a firewall rule to allow HTTP, SSH, RDP and ICMP traffic on
mynetwork
resource "google_compute_firewall" "mynetwork-allow-http-ssh-rdp-icmp"
{
name = "mynetwork-allow-http-ssh-rdp-icmp"
# RESOURCE properties go here
network = google_compute_network.mynetwork.self_link
allow {
    protocol = "tcp"
    ports    = ["22", "80", "3389"]
    }
allow {
    protocol = "icmp"
    }
source_ranges = ["0.0.0.0/0"]
}
```

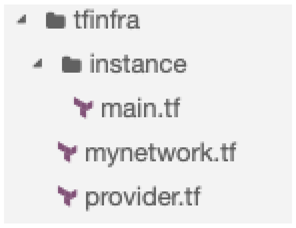


```
student_00_91893d9334bf@cloudshell:~/tfinfra (qwiklabs-gcp-04-234cd0ca6148)$ cat mynetwork.tf
# Create the mynetwork network
resource "google_compute_network" "mynetwork" {
name = "mynetwork"
# RESOURCE properties go here
auto_create_subnetworks = "true"
}
# Add a firewall rule to allow HTTP, SSH, RDP and ICMP traffic on mynetwork
resource "google_compute_firewall" "mynetwork-allow-http-ssh-rdp-icmp" {
name = "mynetwork-allow-http-ssh-rdp-icmp"
# RESOURCE properties go here
network = google_compute_network.mynetwork.self_link
allow {
    protocol = "tcp"
    ports    = ["22", "80", "3389"]
    }
allow {
    protocol = "icmp"
    }
source_ranges = ["0.0.0.0/0"]
}
student_00_91893d9334bf@cloudshell:~/tfinfra (qwiklabs-gcp-04-234cd0ca6148)$
```

Define the VM instances by creating a VM instance module. A module is a reusable configuration inside a folder. You will use this module for both VM instances of this lab.

1. To create a new folder inside **tfinfra**, select the **tfinfra** folder, and then click **File** > **New Folder**.
2. Name the new folder **instance**.
3. To create a new file inside **instance**, `right-click` on **instance** folder and then click **New File**.
4. Name the new file **main.tf**, and then open it.

You should have the following folder structure in Cloud Shell:



```
tfinfra
  instance
    main.tf
  mynetwork.tf
  provider.tf
```

5. Copy the following base code into **main.tf**:

```
resource [RESOURCE_TYPE] "vm_instance" {
  name = [RESOURCE_NAME]
  # RESOURCE properties go here
}
```

6. In `main.tf`, replace `[RESOURCE_TYPE]` with `"google_compute_instance"` (with the quotes).

Note: The **google_compute_instance** resource is a Compute Engine instance.

7. In `main.tf`, replace `[RESOURCE_NAME]` with `"${var.instance_name}"` (with the quotes).

Because you will be using this module for both VM instances, you are defining the instance name as an input variable. This allows you to control the name of the variable from mynetwork.tf. Learn more about input variables in the Terraform: Define Input Variables Guide.

8. Add the following properties to `main.tf`:

```
 zone         = "${var.instance_zone}"
  machine_type = "${var.instance_type}"
```

These properties define the zone and machine type of the instance as input variables.

9. Add the following properties to `main.tf`:

```
 boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
      }
  }
```

This property defines the boot disk to use the Debian 11 OS image. Because both VM instances will use the same image, you can hard-code this property in the module.

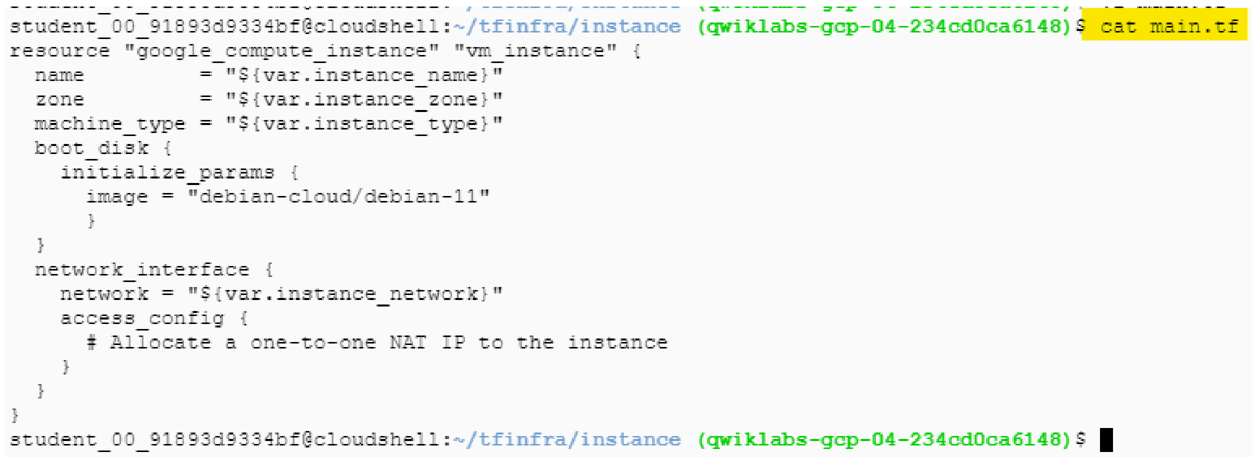
10. Add the following properties to `main.tf`:

```
 network_interface {
    network = "${var.instance_network}"
    access_config {
      # Allocate a one-to-one NAT IP to the instance
    }
  }
```

This property defines the network interface by providing the network name as an input variable and the access configuration. Leaving the access configuration empty results in an ephemeral external IP address (required in this lab). To create instances with only an internal IP address, remove the access_config section. For more information, see the Terraform documentation.

11. Verify that `main.tf` looks like this, including brackets `{}`

```
resource "google_compute_instance" "vm_instance" {
  name         = "${var.instance_name}"
  zone         = "${var.instance_zone}"
  machine_type = "${var.instance_type}"
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
      }
  }
  network_interface {
    network = "${var.instance_network}"
    access_config {
      # Allocate a one-to-one NAT IP to the instance
    }
  }
}
```



```
student_00_91893d9334bf@cloudshell:~/tfinfra/instance (qwiklabs-gcp-04-234cd0ca6148)$ cat main.tf
resource "google_compute_instance" "vm_instance" {
  name         = "${var.instance_name}"
  zone         = "${var.instance_zone}"
  machine_type = "${var.instance_type}"
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
      }
  }
  network_interface {
    network = "${var.instance_network}"
    access_config {
      # Allocate a one-to-one NAT IP to the instance
    }
  }
}
student_00_91893d9334bf@cloudshell:~/tfinfra/instance (qwiklabs-gcp-04-234cd0ca6148)$
```

12. To create a new file inside **instance**, `right-click` on **instance** folder and then click **New File**.
13. Name the new file **variables.tf**, and then open it.
14. Define the 4 input variables in `variables.tf`.

```
variable "instance_name" {}
variable "instance_zone" {}
variable "instance_type" {
  default = "e2-micro"
  }
variable "instance_network" {}
```

By giving **instance_type** a default value, you make the variable optional. The **instance_name**, **instance_zone**, and **instance_network** are required, and you will define them in `mynetwork.tf`.

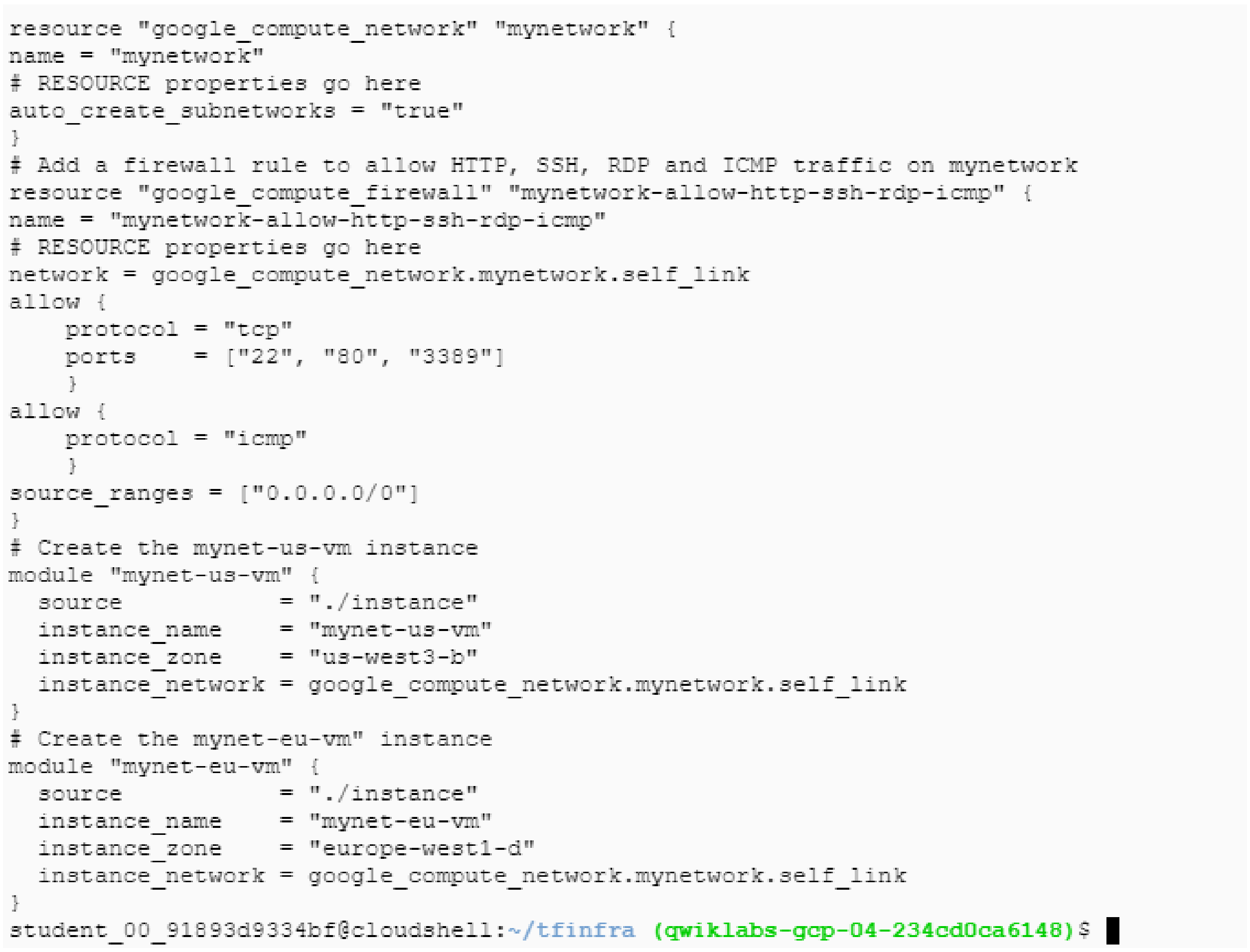
16. Add the following VM instances to `mynetwork.tf`:

```
# Create the mynet-us-vm instance
module "mynet-us-vm" {
  source           = "./instance"
  instance_name    = "mynet-us-vm"
  instance_zone    = "Zone"
  instance_network = google_compute_network.mynetwork.self_link
}
# Create the mynet-eu-vm" instance
module "mynet-eu-vm" {
  source           = "./instance"
  instance_name    = "mynet-eu-vm"
  instance_zone    = "europe-west1-d"
  instance_network = google_compute_network.mynetwork.self_link
}
```

These resources are leveraging the module in the **instance** folder and provide the name, zone, and network as inputs. Because these instances depend on a VPC network, you are using the **google_compute_network.mynetwork.self_link** reference to instruct Terraform to resolve these resources in a dependent order. In this case, the network is created before the instance.
Note: The benefit of writing a Terraform module is that it can be reused across many configurations. Instead of writing your own module, you can also leverage existing modules from the Terraform Module registry.

18. Verify that `mynetwork.tf` looks like this, including brackets {}

```
# Create the mynetwork network
resource "google_compute_network" "mynetwork" {
name = "mynetwork"
# RESOURCE properties go here
auto_create_subnetworks = "true"
}
# Add a firewall rule to allow HTTP, SSH, RDP and ICMP traffic on
mynetwork
resource "google_compute_firewall" "mynetwork-allow-http-ssh-rdp-icmp"
{
name = "mynetwork-allow-http-ssh-rdp-icmp"
# RESOURCE properties go here
network = google_compute_network.mynetwork.self_link
allow {
    protocol = "tcp"
    ports    = ["22", "80", "3389"]
    }
allow {
    protocol = "icmp"
    }
source_ranges = ["0.0.0.0/0"]
}
# Create the mynet-us-vm instance
module "mynet-us-vm" {
  source              = "./instance"
  instance_name       = "mynet-us-vm"
  instance_zone       = "Zone"
  instance_network = google_compute_network.mynetwork.self_link
}
# Create the mynet-eu-vm" instance
module "mynet-eu-vm" {
  source              = "./instance"
  instance_name       = "mynet-eu-vm"
  instance_zone       = "europe-west1-d"
  instance_network = google_compute_network.mynetwork.self_link
}
```



```
resource "google_compute_network" "mynetwork" {
name = "mynetwork"
# RESOURCE properties go here
auto_create_subnetworks = "true"
}
# Add a firewall rule to allow HTTP, SSH, RDP and ICMP traffic on mynetwork
resource "google_compute_firewall" "mynetwork-allow-http-ssh-rdp-icmp" {
name = "mynetwork-allow-http-ssh-rdp-icmp"
# RESOURCE properties go here
network = google_compute_network.mynetwork.self_link
allow {
    protocol = "tcp"
    ports    = ["22", "80", "3389"]
    }
allow {
    protocol = "icmp"
    }
source_ranges = ["0.0.0.0/0"]
}
# Create the mynet-us-vm instance
module "mynet-us-vm" {
  source           = "./instance"
  instance_name    = "mynet-us-vm"
  instance_zone    = "us-west3-b"
  instance_network = google_compute_network.mynetwork.self_link
}
# Create the mynet-eu-vm" instance
module "mynet-eu-vm" {
  source           = "./instance"
  instance_name    = "mynet-eu-vm"
  instance_zone    = "europe-west1-d"
  instance_network = google_compute_network.mynetwork.self_link
}
student_00_91893d9334bf@cloudshell:~/tfinfra (qwiklabs-gcp-04-234cd0ca6148)$
```

Create mynetwork and its resources
It's time to apply the mynetwork configuration.

1. To rewrite the Terraform configuration files to a canonical format and style, run the following command:

```
terraform fmt
```

The output should look like this:

```
mynetwork.tf
```

Note: If you get an error, revisit the previous steps to ensure that your configuration matches the lab instructions. If you cannot troubleshoot the issue of your configuration, download and then look at these finished configurations:

- mynetwork.tf
- main.tf
- provider.tf
- variables.tf

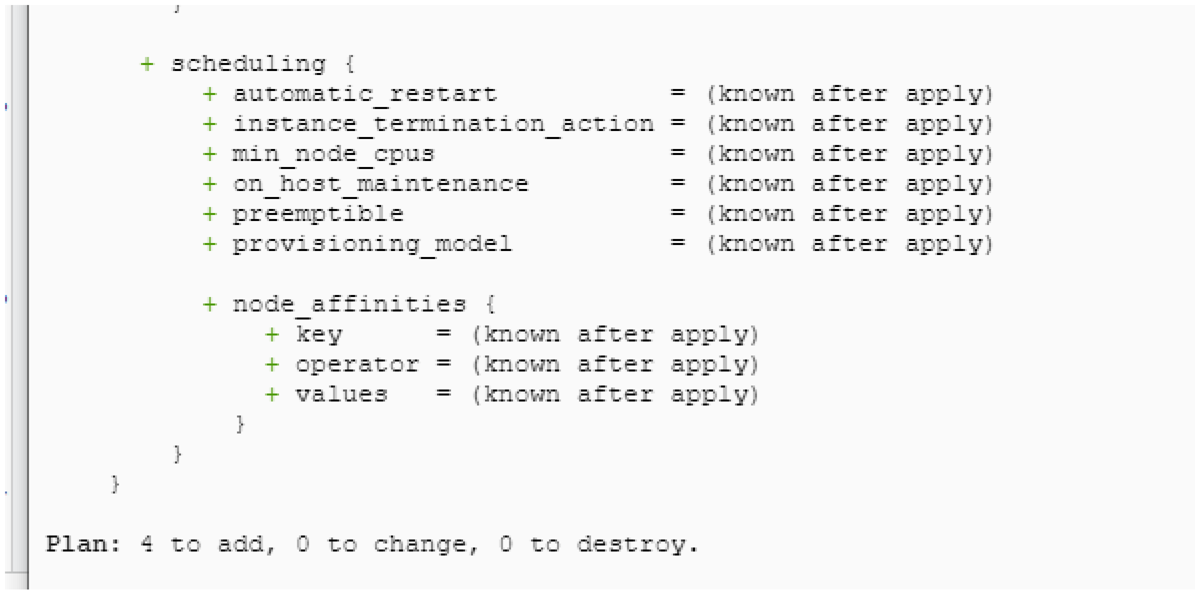
2. To initialize Terraform, run the following command:

```
terraform init
```

Note: If you get an error, revisit the previous steps to ensure that you have the correct folder/file structure. If you cannot troubleshoot the issue of your configuration, refer to the finished configurations linked above. When you have corrected the issue, re-run the previous command.

3. To create an execution plan, run the following command:

```
terraform plan
```



```
          }

      + scheduling {
          + automatic_restart           = (known after apply)
          + instance_termination_action = (known after apply)
          + min_node_cpus               = (known after apply)
          + on_host_maintenance         = (known after apply)
          + preemptible                 = (known after apply)
          + provisioning_model          = (known after apply)

          + node_affinities {
              + key      = (known after apply)
              + operator = (known after apply)
              + values   = (known after apply)
            }
        }
    }

Plan: 4 to add, 0 to change, 0 to destroy.
```

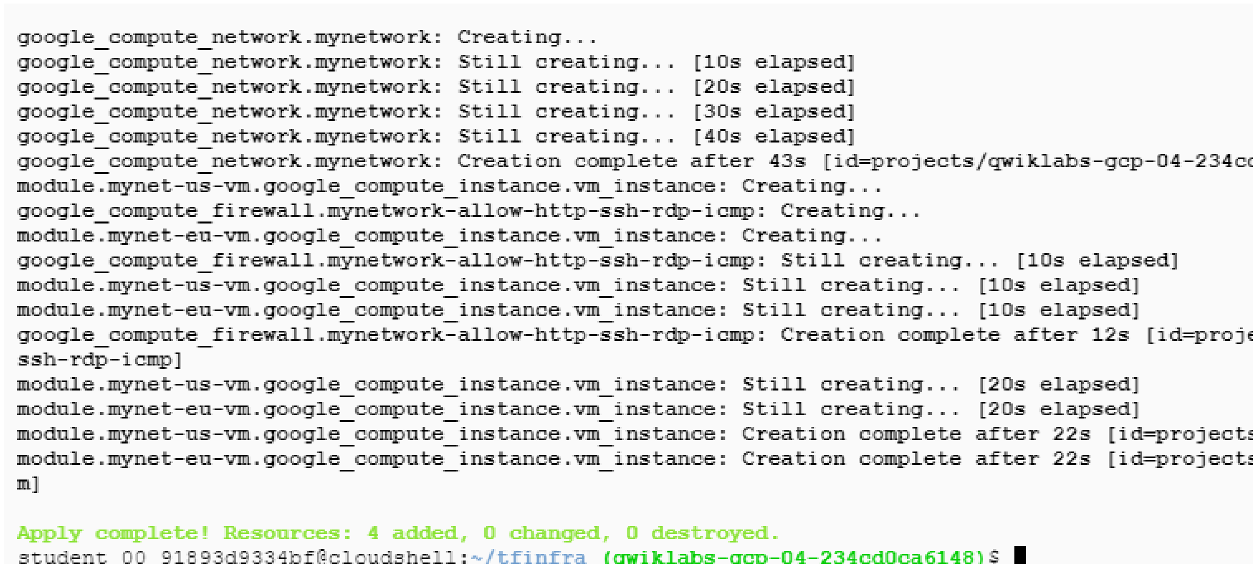
Terraform determined that the following 4 resources need to be added:

Name	Description
mynetwork	VPC network

mynetwork-allow-http-ssh-rdp-icmp	Firewall rule to allow HTTP, SSH, RDP and ICMP
mynet-us-vm	VM instance in Zone
mynet-eu-vm	VM instance in "europe-west1-d"

4. To apply the desired changes, run the following command:

```
terraform apply
```



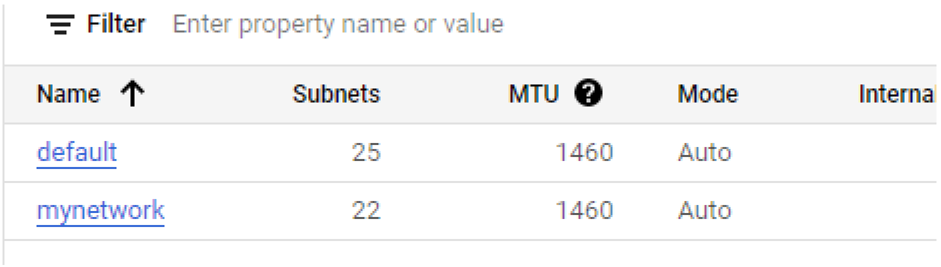
```
google_compute_network.mynetwork: Creating...
google_compute_network.mynetwork: Still creating... [10s elapsed]
google_compute_network.mynetwork: Still creating... [20s elapsed]
google_compute_network.mynetwork: Still creating... [30s elapsed]
google_compute_network.mynetwork: Still creating... [40s elapsed]
google_compute_network.mynetwork: Creation complete after 43s [id=projects/qwiklabs-gcp-04-234cd
module.mynet-us-vm.google_compute_instance.vm_instance: Creating...
google_compute_firewall.mynetwork-allow-http-ssh-rdp-icmp: Creating...
module.mynet-eu-vm.google_compute_instance.vm_instance: Creating...
google_compute_firewall.mynetwork-allow-http-ssh-rdp-icmp: Still creating... [10s elapsed]
module.mynet-us-vm.google_compute_instance.vm_instance: Still creating... [10s elapsed]
module.mynet-eu-vm.google_compute_instance.vm_instance: Still creating... [10s elapsed]
google_compute_firewall.mynetwork-allow-http-ssh-rdp-icmp: Creation complete after 12s [id=proje
ssh-rdp-icmp]
module.mynet-us-vm.google_compute_instance.vm_instance: Still creating... [20s elapsed]
module.mynet-eu-vm.google_compute_instance.vm_instance: Still creating... [20s elapsed]
module.mynet-us-vm.google_compute_instance.vm_instance: Creation complete after 22s [id=projects
module.mynet-eu-vm.google_compute_instance.vm_instance: Creation complete after 22s [id=projects
m]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
student_00_91893d9334bf@cloudshell:~/tfinfra (qwiklabs-gcp-04-234cd0ca6148)$
```

In the Cloud Console, verify that the resources were created.

Verify your network in the Cloud Console

1. In the Cloud Console, on the **Navigation menu** (☰), click **VPC network** > **VPC networks**.
2. View the **mynetwork** VPC network with a subnetwork in every region.

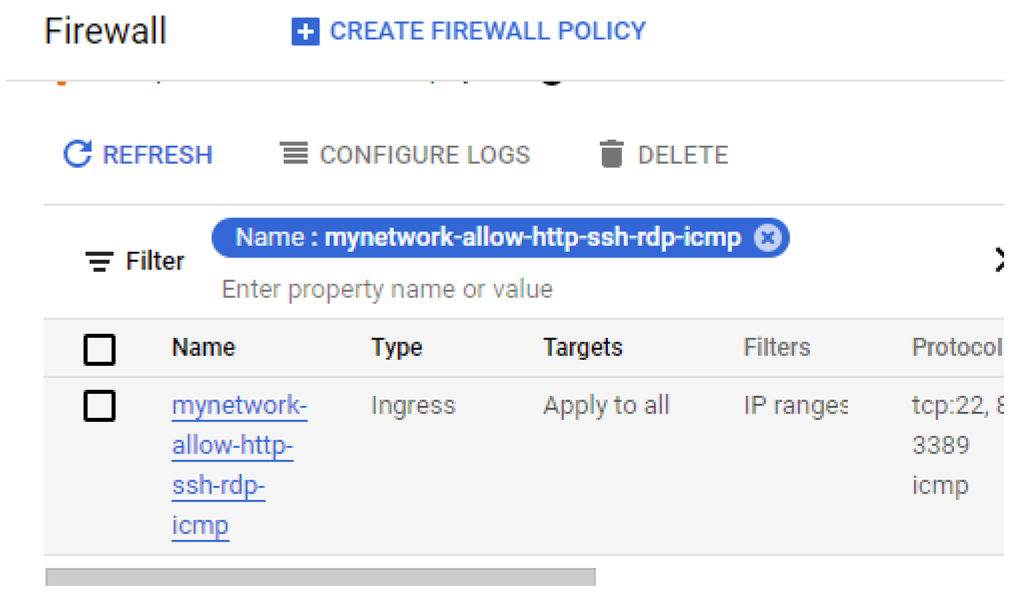


Filter Enter property name or value

Name ↑	Subnets	MTU	Mode	Interna
default	25	1460	Auto	
mynetwork	22	1460	Auto	

3. On the **Navigation menu**, click **VPC network** > **Firewall**.
4. Sort the firewall rules by **Network**.

5. View the **mynetwork-allow-http-ssh-rdp-icmp** firewall rule for **mynetwork**.


Firewall

CREATE FIREWALL POLICY

REFRESH CONFIGURE LOGS DELETE

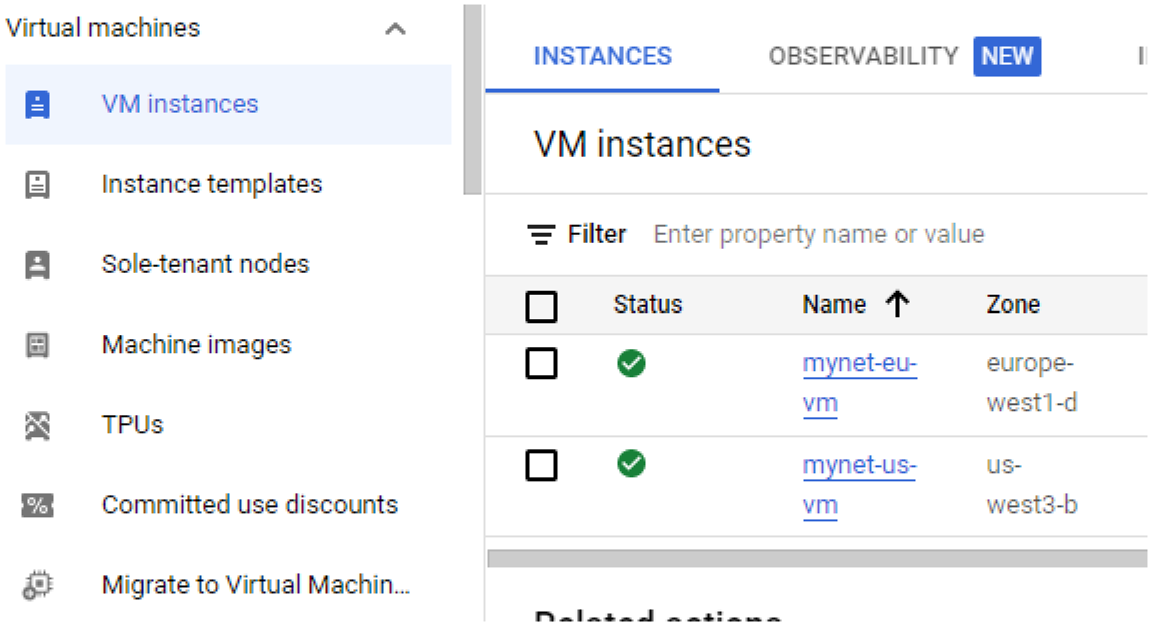
Filter Name : mynetwork-allow-http-ssh-rdp-icmp

Enter property name or value

Name	Type	Targets	Filters	Protocol
mynetwork-allow-http-ssh-rdp-icmp	Ingress	Apply to all	IP ranges	tcp:22, 8 3389 icmp

Verify your VM instances in the Cloud Console

1. On the **Navigation menu** (☰), click **Compute Engine** > **VM instances**.
2. View the **mynet-us-vm** and **mynet-eu-vm** instances.
3. Note the internal IP address for **mynet-eu-vm**.
4. For **mynet-us-vm**, click **SSH** to launch a terminal and connect.



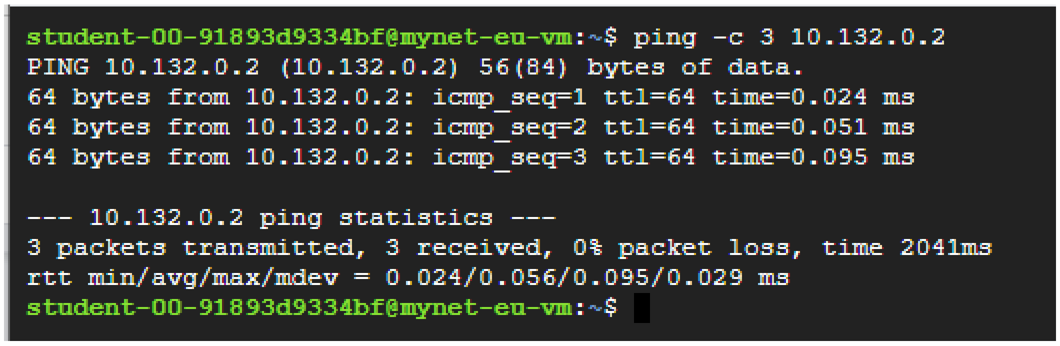
VM instances

Filter Enter property name or value

Status	Name ↑	Zone
✓	mynet-eu-vm	europe-west1-d
✓	mynet-us-vm	us-west3-b

5. To test connectivity to **mynet-eu-vm**'s internal IP address, run the following command in the SSH terminal (replacing mynet-eu-vm's internal IP address with the value noted earlier):

```
ping -c 3 <Enter mynet-eu-vm's internal IP here>
```



```
student-00-91893d9334bf@mynet-eu-vm:~$ ping -c 3 10.132.0.2
PING 10.132.0.2 (10.132.0.2) 56(84) bytes of data.
64 bytes from 10.132.0.2: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 10.132.0.2: icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from 10.132.0.2: icmp_seq=3 ttl=64 time=0.095 ms

--- 10.132.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.024/0.056/0.095/0.029 ms
student-00-91893d9334bf@mynet-eu-vm:~$
```

Note: This should work because both VM instances are on the same network, and the firewall rule allows ICMP traffic!

In this lab, you created a Terraform configuration with a module to automate the deployment of Google Cloud infrastructure. As your configuration changes, Terraform can create incremental execution plans, which allows you to build your overall configuration step by step.

The instance module allows you to re-use the same resource configuration for multiple resources while providing properties as input variables. You can leverage the configuration and module that you created as a starting point for future deployments.