

## DEFINE AND PROVISION AN APPLICATION NETWORK INFRASTRUCTURE USING TERRAFORM

Terraform can be used to securely provision cloud infrastructure in a way that's repeatable and consistent. This way, I can easily and quickly make any required changes. I will be using terraform to create a network and firewall rules.

### First Step: Clone The Terraform Repo

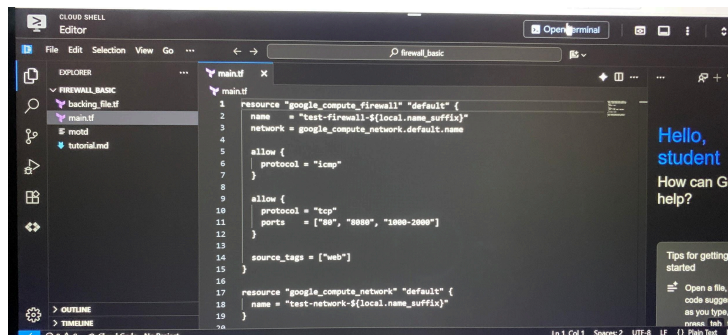
I cloned the terraform example repository using the cloud shell terminal. The terraform example contains the configuration file which I used to provision the firewall rules.

This command shown below, clones the terraform example directory.

```
cloudshell_open --repo_url "https://github.com/terraform-google-modules/docs-examples.git"
--print_file "./motd" --dir "firewall_basic" --page "editor" --tutorial "./tutorial.md" --open_in_editor
"main.tf" --force_new_clone
```

This command performs the following actions

1. Clones the “**terraform-google-modules**”
2. Switches to the “**firewall\_basic**” directory
3. Checks the cloned files, for example “[tutorial.md](#)”
4. Opens “[main.tf](#)” in the cloud shell editor



Once the cloning is complete, I'll be at the “~/cloudshell\_open/docs-examples/firewall\_basic” location in the terminal. My cloud shell prompt displays the following output

```
student_01_c2e095df84e2@cloudshell:~/cloudshell_open/docs-examples/firewall_basic (qwiklabs-gcp-04-fde36f013e65)$
```

I typed the “ls” command into the cloud shell terminal to list the content of directory

I noticed that several files in the directory have been downloaded

1. backing\_file.tf
2. [main.tf](#)
3. motd
4. [tutorial.md](#)

I typed the “cat [main.tf](#)” command into the cloud shell terminal to analyze the configuration of the firewall. The [main.tf](#) file is the configuration file that defines the resources that terraform will create. Two resources will be created: a firewall rule “**google\_compute\_firewall**” named

“test-firewall-`{local.name_suffix}`” with rules to allow ICMP and TCP traffic from ports 80, 8080, and 1000-2000 and a VPC network. The variable “`{.name_suffix}`” is a local variable that automatically generates unique names for resources.

```
resource "google_compute_firewall" "default" {
  name = "test-firewall-{local.name_suffix}"
  network = google_compute_network.default.name

  allow {
    protocol = "icmp"
  }

  allow {
    protocol = "tcp"
    ports = ["80", "8080", "1000-2000"]
  }

  source_tags = ["web"]
}

resource "google_compute_network" "default" {
  name = "test-network-{local.name_suffix}"
}
```

## Second step: Deploy the VPC network and firewall

I deployed a new VPC network and a new firewall rule. This task provided me with hands-on-experience with building a VPC network and subnets.

I typed the “`export GOOGLE_CLOUD_PROJECT= Project ID`” command into the terminal which sets the project ID.

I typed the “`terraform init`” command into the terminal which initializes the Terraform script

The output returned the message above stating that the Terraform has been successfully initialized. Take a moment to examine the output. I noticed that Terraform created a new firewall and VPC network.

```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/random...
- Finding latest version of hashicorp/google...
- Installing hashicorp/random v3.5.1...
- Installed hashicorp/random v3.5.1 (signed by HashiCorp)
- Installing hashicorp/google v4.83.0...
- Installed hashicorp/google v4.83.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
student_02_446f55c078@cloudshell:~/cloudshell/open/docs-examples/firewall_basic [quiklabs-gcp-02-f2bc9c812802]$
```

Once the initialization is complete, I typed the “`terraform apply`” command into the terminal which applies changes and deploys the Terraform script. The command prompt requested a value which I typed “`yes`”.

```
+ bgp_inter_region_cost = (known after apply)
+ delete_default_routes_on_create = false
+ gateway_ipv4 = (known after apply)
+ id = (known after apply)
+ internal_ipv6_range = (known after apply)
+ mtu = (known after apply)
+ name = (known after apply)
+ network_firewall_policy_enforcement_order = "FIRST_CLASSIC_FIREWALL"
+ network_id = (known after apply)
+ numeric_id = (known after apply)
+ project = "quiklabs-gcp-01-e0680d1d3ff1"
+ routing_mode = (known after apply)
+ self_link = (known after apply)
}

# random_pet.suffix will be created
+ resource "random_pet" "suffix" {
+   id = (known after apply)
+   length = 2
+   separator = "-"
+ }

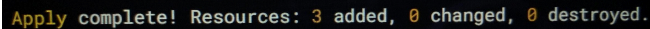
Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes
```

This started creating the VPC network and firewall rules.

Once completed, the output returned ;

A screenshot of a terminal window with a dark background. The text 'Apply complete! Resources: 3 added, 0 changed, 0 destroyed.' is displayed in a light-colored font.

```
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

This means that the VPC and firewall have been successfully deployed.

### **Final Step: verify the deployment of the resources**

I verified that the newly created VPC and firewall rules have been successfully deployed.

I noticed two VPC networks, “**default**” and the newest one I just created, “**test-network**”. I accessed the details of “**test-network**”

From “**firewall**” under “**protocols and ports**” I noticed the firewall rules are the same rules as defined in the configuration file: Allow and tcp: 80, 1000-2000, 8080 icmp.

Following these steps, I successfully built a VPC network and subnet using terraform and the cloud shell. This lab provides the foundation for me to develop advanced automated solutions that can be given to systems administrators to use with terraform.

By creating the VPC network and firewall, I have gained a better understanding of how it enables me to automate the process of provisioning and modifying firewall rules.