

## LAB 1 : GCP-Create a Project Module in Terraform

(Kavitha B)

### Experimenting with GCP

Below is the main.tf code -

Specifies provider is google, machine type, tags, network and can update when VM is running.

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
    }
  }
}
provider "google" {
  region = "us-central1"
  zone   = "us-central1-c"
}

resource "google_compute_instance" "terraform" {
  name         = "terraform"
  machine_type = "n1-standard-1"
  tags         = ["web", "dev"]
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
    }
  }
  network_interface {
    network = "default"
    access_config {
    }
  }
  allow_stopping_for_update = true
}
```

Terraform installation can be verified using the `Terraform -version` command.

Commands executed:

`terraform init`

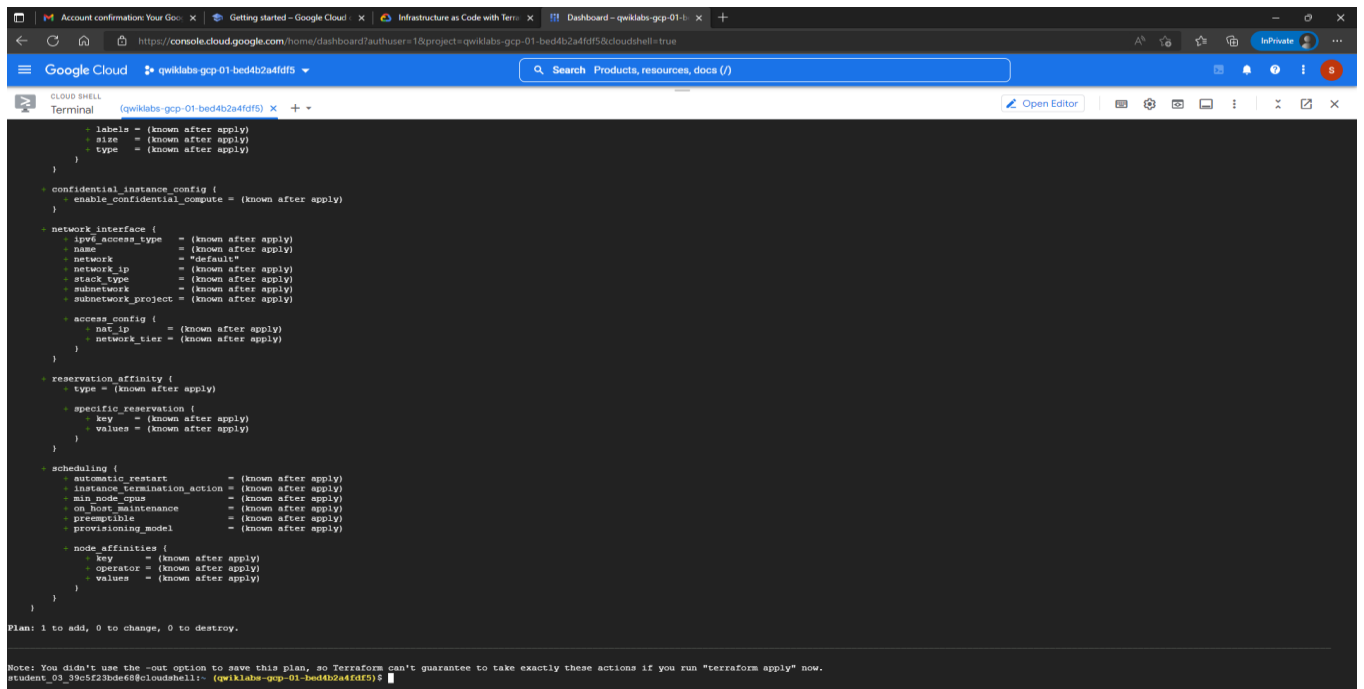
`terraform plan`

`terraform apply`

`terraform destroy`

# LAB 1 : GCP-Create a Project Module in Terraform

## (Kavitha B)



```
labels = (known after apply)
size   = (known after apply)
type   = (known after apply)
}

confidential_instance_config {
  enable_confidential_compute = (known after apply)
}

network_interface {
  ipv6_access_type = (known after apply)
  name             = (known after apply)
  network          = "default"
  network_ip       = (known after apply)
  stack_type       = (known after apply)
  subnetwork       = (known after apply)
  subnetwork_project = (known after apply)
}

access_config {
  nat_ip        = (known after apply)
  network_tier  = (known after apply)
}

reservation_affinity {
  type = (known after apply)
}

specific_reservation {
  key   = (known after apply)
  values = (known after apply)
}

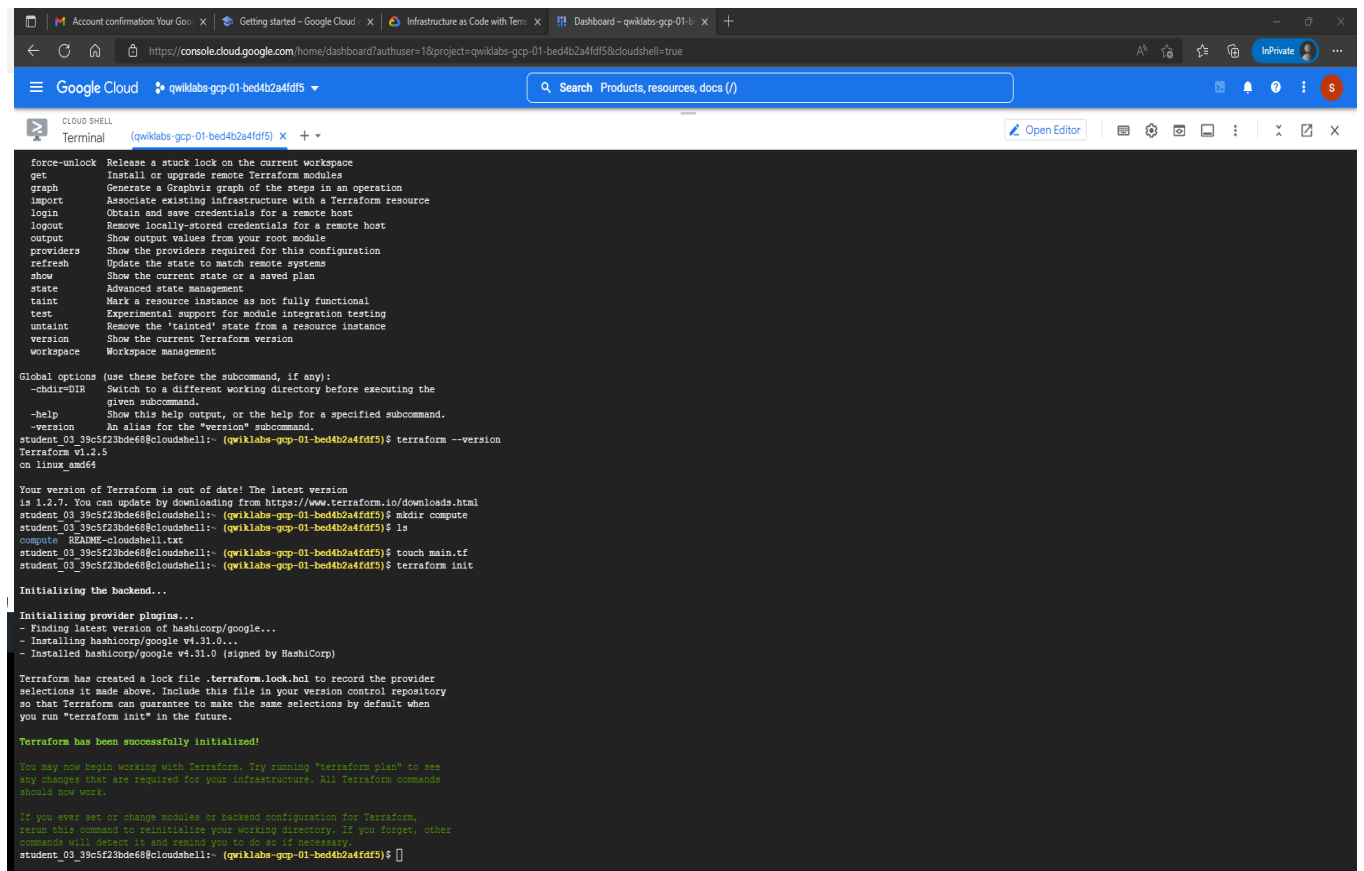
scheduling {
  automatic_restart    = (known after apply)
  instance_termination_action = (known after apply)
  min_node_cpus        = (known after apply)
  on_host_maintenance  = (known after apply)
  preemptible          = (known after apply)
  provisioning_model    = (known after apply)
}

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

Plan: 1 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

student\_03\_39c5f23bde68@cloudshell:~ (qwiklabs-gcp-01-bed4b2a4fd5) %



```
force-unlock Release a stuck lock on the current workspace
get          Install or upgrade remote Terraform modules
graph       Generate a Graphviz graph of the steps in an operation
import      Associate existing infrastructure with a Terraform resource
login       Obtain and save credentials for a remote host
logout      Remove locally-stored credentials for a remote host
output      Show output values from your root module
providers   Show the providers required for this configuration
refresh     Update the state to match remote systems
show        Show the current state or a saved plan
state       Advanced state management
taint       Mark a resource instance as not fully functional
test        Experimental support for module integration testing
untaint     Remove the "tainted" state from a resource instance
version     Show the current Terraform version
workspace   Workspace management

Global options (use these before the subcommand, if any):
-chdir=DIR  Switch to a different working directory before executing the
            given subcommand.
-help      Show this help output, or the help for a specified subcommand.
-version   An alias for the "version" subcommand.
student_03_39c5f23bde68@cloudshell:~ (qwiklabs-gcp-01-bed4b2a4fd5) $ terraform --version
Terraform v1.2.5
on linux_amd64

Your version of Terraform is out of date! The latest version
is 1.2.7. You can update by downloading from https://www.terraform.io/downloads.html
student_03_39c5f23bde68@cloudshell:~ (qwiklabs-gcp-01-bed4b2a4fd5) $ mkdir compute
student_03_39c5f23bde68@cloudshell:~ (qwiklabs-gcp-01-bed4b2a4fd5) $ ls
compute README-cloudshell.txt
student_03_39c5f23bde68@cloudshell:~ (qwiklabs-gcp-01-bed4b2a4fd5) $ touch main.tf
student_03_39c5f23bde68@cloudshell:~ (qwiklabs-gcp-01-bed4b2a4fd5) $ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/google...
- Installing hashicorp/google v4.31.0...
- Installed hashicorp/google v4.31.0 (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_03_39c5f23bde68@cloudshell:~ (qwiklabs-gcp-01-bed4b2a4fd5) %
```