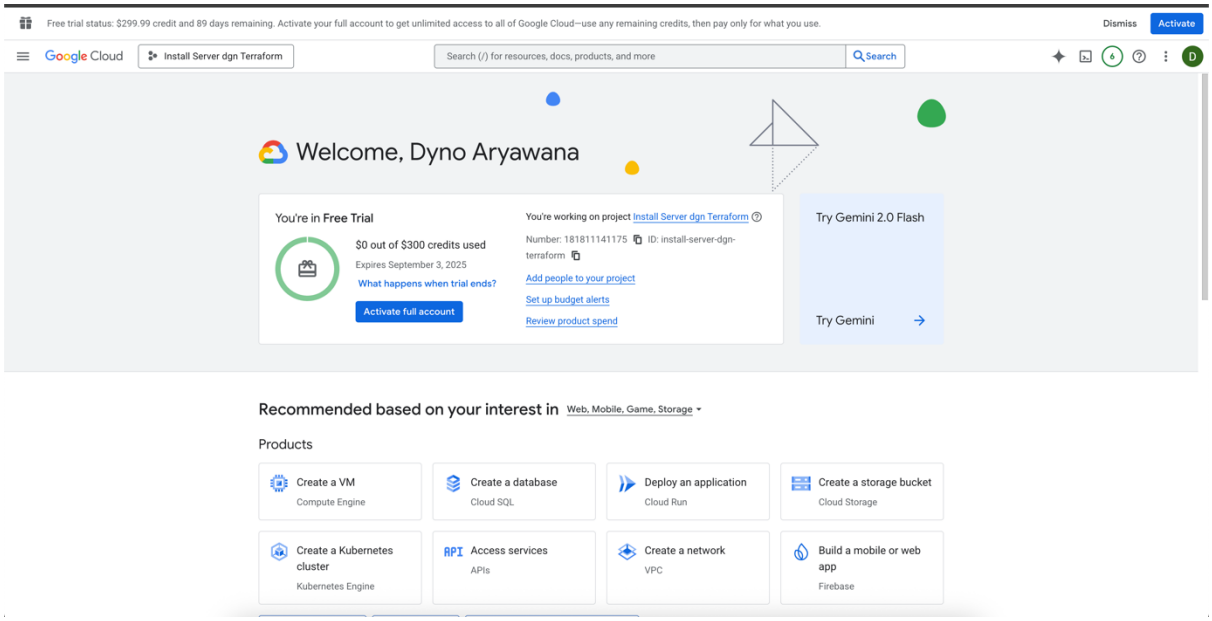
GCP Instance Deployment with Terraform

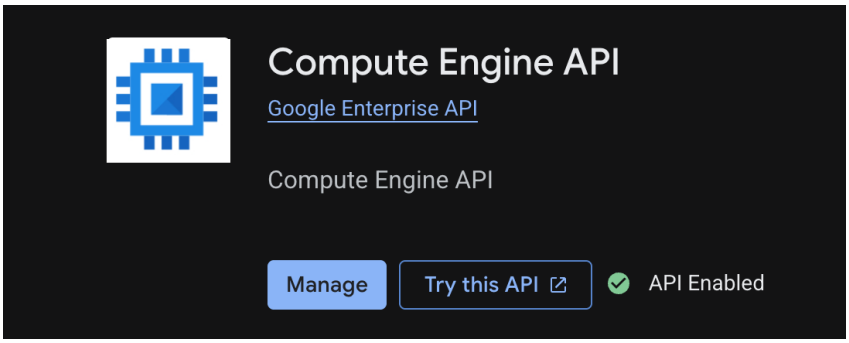
Currently streamlining my infrastructure provisioning on Google Cloud Platform (GCP) for a personal project, and I've chosen **Terraform** as my tool of choice. I'm focusing on defining and deploying a single virtual machine instance, allowing me to manage my resources efficiently and declaratively. This approach ensures consistency and repeatability as I build out my cloud environment.

---

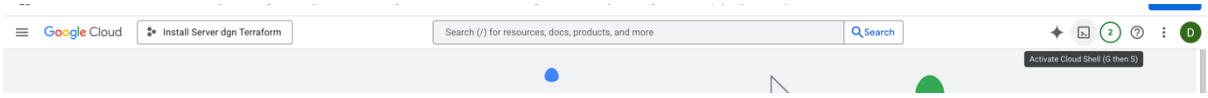
1. Log in into GCP Account and creat New Project :



2. Acivate Compute Engine API to make Terraform work :



3. Log in into Cloud Shell :



4. Make a new directory and creat file main.tf with explanation :

```
  GNU nano 7.2
# Konfigurasi provider Google Cloud
provider "google" {
  project = var.project_id   # ID project GCP (dari terraform.tfvars)
  region  = var.region        # Region tempat resource dibuat
  zone    = var.zone          # Zona tempat VM akan dijalankan
}

# Resource untuk membuat 1 VM instance di GCP
resource "google_compute_instance" "default" {
  name         = var.instance_name   # Nama VM, diambil dari variabel
  machine_type = var.machine_type    # Tipe mesin, misal "e2-micro"
  zone         = var.zone            # Zona tempat VM dijalankan

  # Konfigurasi disk utama (boot disk)
  boot_disk {
    initialize_params {
      image = var.image              # Sistem operasi yang digunakan (image GCP)
    }
  }

  # Konfigurasi jaringan
  network_interface {
    network = "default"              # Menggunakan jaringan default
    access_config {}                 # Menambahkan public IP agar bisa diakses dari internet
  }

  # Script yang otomatis dijalankan saat VM pertama kali nyala
  metadata_startup_script = <<-EOT
    #!/bin/bash
    apt update                      # Update repository package Ubuntu
    apt install -y nginx            # Install web server Nginx otomatis
  EOT
}
```

5. Creating variable with variables.tf :

```
  GNU nano 7.2
variable "project_id" {}
variable "region" {}
variable "zone" {}
variable "instance_name" {}
variable "machine_type" {}
variable "image" {}
```

6. Creating terraform.tfvars and explanation :

```
  GNU nano 7.2                                                                terraform.tfvars *
project_id     = "install-server-dgn-terraform"   # ID project GCP kamu (harus sesuai yang aktif di GCP Console)
region         = "us-central1"                    # Wilayah/region GCP tempat VM dibuat
zone           = "us-central1-a"                  # Zona spesifik dalam region (satu lokasi data center)
instance_name  = "vm-nginx-2"                     # Nama VM yang akan dibuat di GCP
machine_type   = "e2-micro"                       # Tipe VM, e2-micro = yang gratisan (Free Tier)
image          = "ubuntu-os-cloud/ubuntu-2004-lts" # Sistem operasi yang akan dipasang (Ubuntu 20.04 LTS)
```

7. Create outputs.tf to automatically display the public IP of the VM when Terraform apply is excecuted :

```
  GNU nano 7.2
output "vm_ip" {
  value = google_compute_instance.default.network_interface[0].access_config[0].nat_ip
}
```

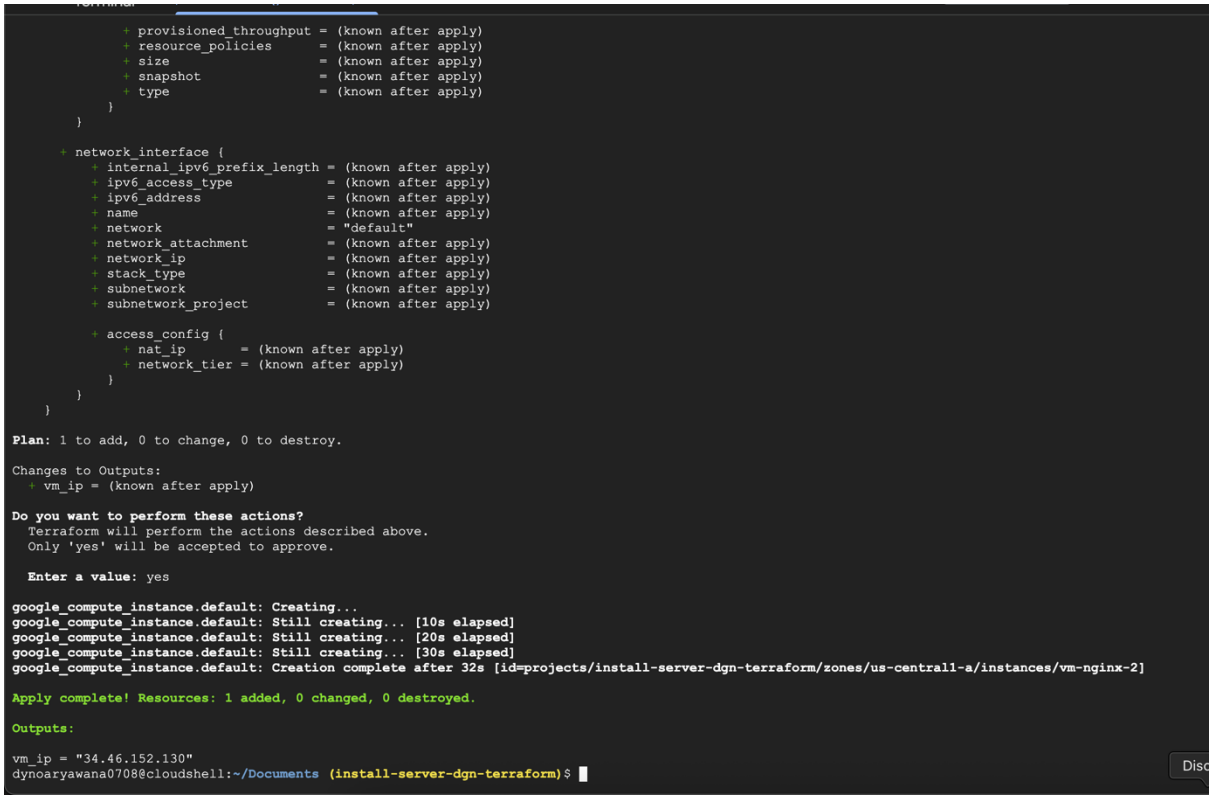
8. Run terraform init to initialize the Terraform project so that it is ready to use :

```
dynoaryawana0708@cloudshell:~/Documents (install-server-dgn-terraform)$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/google from the dependency lock file
- Using previously-installed hashicorp/google v6.38.0

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.
dynoaryawana0708@cloudshell:~/Documents (install-server-dgn-terraform)$
```

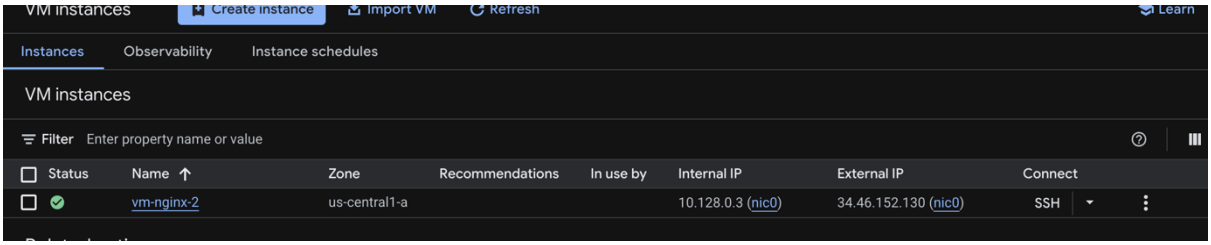
9. Run terraform plan to display the Terraform action plan before It is actually excecuted :

```
dynoaryawana0708@cloudshell:~/Documents (install-server-dgn-terraform)$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # google_compute_instance.default will be created
  + resource "google_compute_instance" "default" {
      + can_ip_forward          = false
      + cpu_platform            = (known after apply)
      + creation_timestamp      = (known after apply)
      + current_status          = (known after apply)
      + deletion_protection     = false
      + effective_labels        = {
          + "goog-terraform-provisioned" = "true"
        }
      + id                      = (known after apply)
      + instance_id             = (known after apply)
      + label_fingerprint       = (known after apply)
      + machine_type            = "e2-micro"
      + metadata_fingerprint    = (known after apply)
      + metadata_startup_script = <<-EOT
            #!/bin/bash
            apt update
            apt install -y nginx
        EOT
      + min_cpu_platform        = (known after apply)
      + name                    = "vm-nginx"
      + project                 = "install-server-dgn-terraform"
      + self_link               = (known after apply)
      + tags_fingerprint        = (known after apply)
      + terraform_labels        = {
          + "goog-terraform-provisioned" = "true"
        }
      + zone                    = "us-central1-a"

      + boot_disk {
          + auto_delete                = true
          + device_name                = (known after apply)
          + disk_encryption_key_sha256 = (known after apply)
          + guest_os_features          = (known after apply)
          + kms_key_self_link          = (known after apply)
          + mode                       = "READ_WRITE"
          + source                     = (known after apply)
```

10. Run terraform apply to run created project and the result will be like this. If successful, it will say “Apply complete” and the configured public will appear :



11. Check in the instance to see if the VM has been created or not :



Thank You