Google Cloud

Manage your Cloud Infrastructure with Terraform

Prepared by: David Regalado

15

David Regalado Gutiérrez

Data Engineering, Big Data, and Machine Learning on GCP

Specializations in Google Cloud by Cousera

38

Google Cloud

Integrate with Machine Learning APIs

MACHINE LEARNING AND AI SKILL

Badges Google Cloud by Qwiklabs

Follow me: https://beacons.ai/davidregalado

BIG DATA CONFERENCE EUROPE 2020 ONLINE

November 24-26 Online

David Regalado

Workshop

Google Cloud Platform: The "What", "Where", and "How"

MACHINE LEARNING INDIA

LIVE

Machine Learning India presents

Talk Data To Me #4

DAVID REGALADO
Co-Founder, Data Engineering Latam

AKSHAY TOSHNIWAL
Community Manager: MLI

THE UNIVERSITY OF CHICAGO

EVENTO ONLINE

Cómo diseñar y escribir código en Python de manera creativa y eficiente a la hora de resolver problemas reales

Al comprender las nociones básicas de este lenguaje, los profesionales de data science pueden sumar más herramientas técnicas y luego obtener información relevante para sus proyectos.

Regístrate ahora >

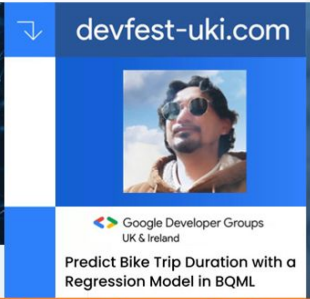
David Regalado

Co-Founder at Data Engineering LATAM

AI TIME JOURNAL PRESENTS

AI TIME JOURNAL PODCAST

Transitioning to a Career in Data Science with David Regalado


devfest-uki.com

Google Developer Groups
UK & Ireland

Predict Bike Trip Duration with a Regression Model in BQML

UNMSM - FISI

Construye una ETL con Python leyendo data de Spotify

David Regalado

- Co-Founder de Data Engineering LATAM
- Teaching Assistant at Global Alumni

Data Engineering Latam

Aforo: 500 personas
08 de octubre - 07:00 PM
Taller

@eneisoft

Meet Our Board Speakers

Thom Ives, Ph.D.
Lead Data Scientist
Founder of Integrated Machine Learning & AI
Lives In: Eagle Idaho, United States

David Oyelade
Chief Data Scientist
Digital Marketer
Making a DIFFERENCE with Big Data & Blockchain revolution.
Lives In: Lagos, Nigeria

David Regalado
Systems Engineer
Data Scientist
Teaching Assistant
Lives In: Lima Metropolitan Area, Peru

Khuyen Tran
Data Scientist Intern at Ocelot Consulting
Technical Writer
Lives In: Illinois, United States

Akshay Toshniwal
Data Scientist & AI Expert
AI Associate Editor
Data & AI Geek
Lives In: Maharashtra, India

Harpreet Sahota
Data Scientist
Statistician
Podcaster
Lives In: Manitoba, Canada

Pragati Jain
Sr. Data Analyst
Microsoft Data Platform MVP
Superuser at Power BI Community
Lives In: England, United Kingdom

Ask the passionate lifetime learner Thom Ives anything about data science

Who am I?

David Regalado

beacons.ai/davidregalado

I studied here

UNIVERSIDAD TECNOLÓGICA DEL PERÚ
Somos parte de Intercorp

Colorado
University

Jhons Hopkins
University

Massachusetts
Institute
of Technology

Stanford
University

I am certified by

Scrum.org
The Home of Scrum

Google Cloud Certified
Cloud Digital Leader

Google Cloud Certified
Associate
Cloud Engineer

Certified Entry-Level Python Programmer
OpenEDG Python Institute

Hackmakers
#WorldInnovationDay
Lead Mentor

I worked here

Telefónica

EQUIFAX

Claro

TERADATA

GRUPO sura

indra

AVANTICA
an indecomm company

ct

Data Engineering Latam

The Problem

Deploying safely and efficiently to the cloud can be tricky and error-prone. After all, we're only human, right?

We are talking about this in the context of production environments. Are you willing to risk making a mistake when configuring these kinds of services?

In addition, there are some services that can be created at the same time when there are no dependencies between them. It would be wise to deploy them in parallel.

The Solution

The Infrastructure as Code (IaC among friends) paradigm allows you to create configuration files with the desired state of the environment to be managed. These files will be read by whichever solution you choose.

There are many options, such as:

The Solution

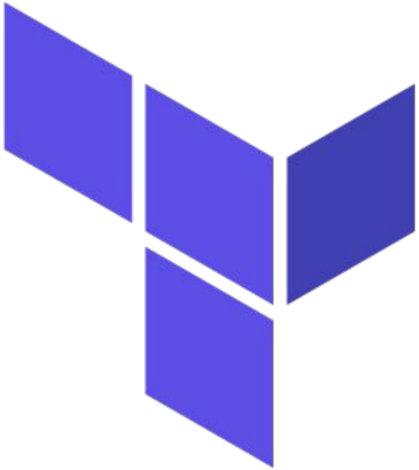
HashiCorp
Terraform

ANSIBLE

Cloud Deployment Manager

pulumi

AWS
CloudFormation



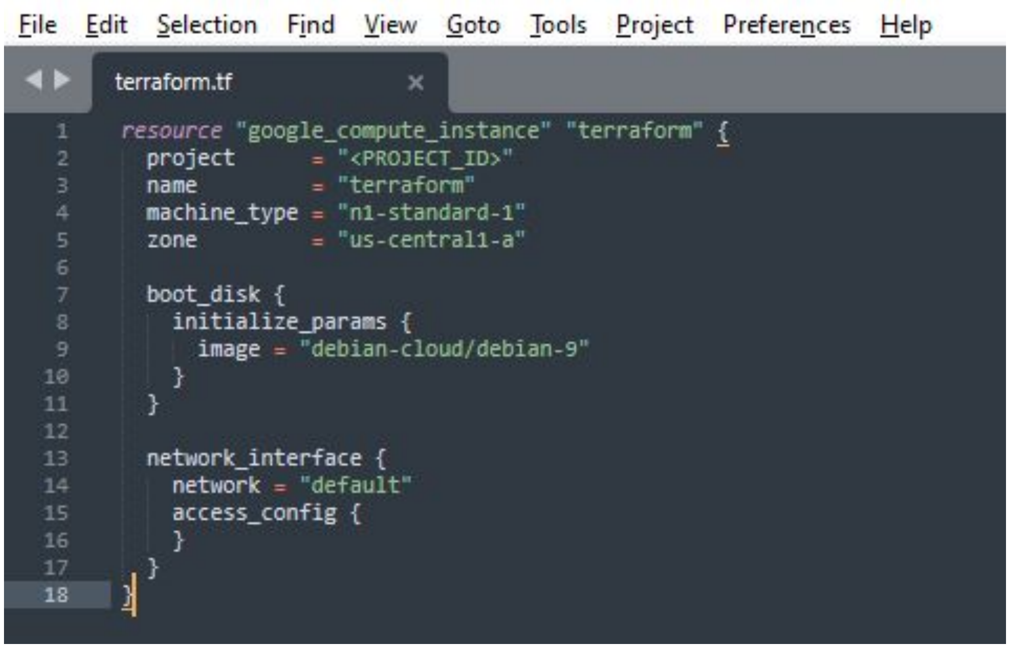
HashiCorp

Terraform

A Quick Example using Terraform

This is a configuration file.

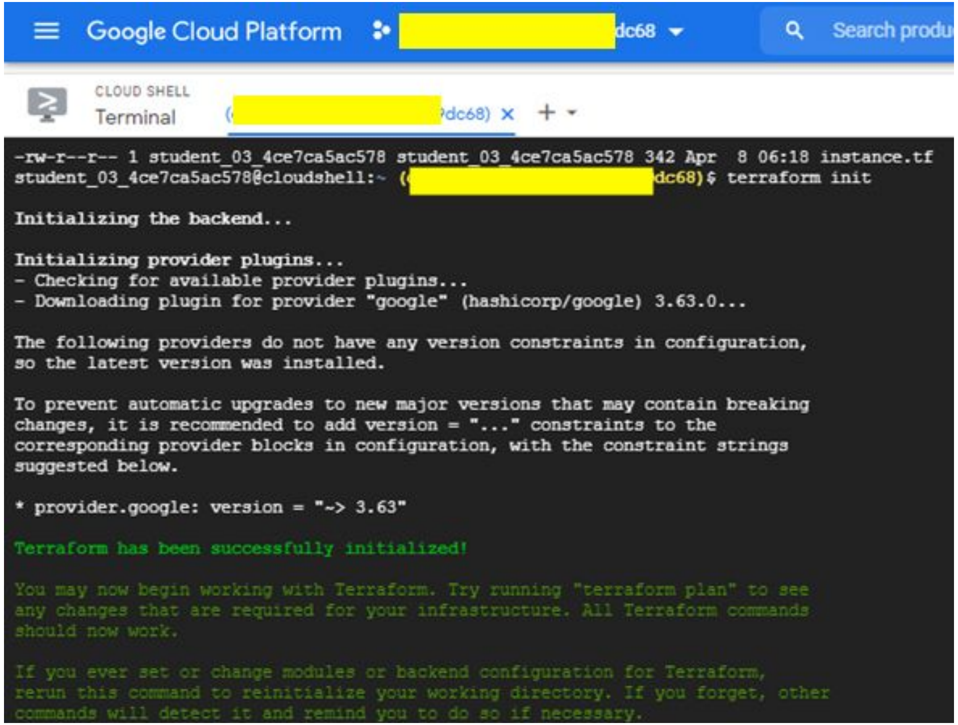
(Say hi!)



```
resource "google_compute_instance" "terraform" {
  project      = "<PROJECT_ID>"
  name         = "terraform"
  machine_type = "n1-standard-1"
  zone         = "us-central1-a"

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-9"
    }
  }

  network_interface {
    network = "default"
    access_config {
    }
  }
}
```

A Quick Example using Terraform

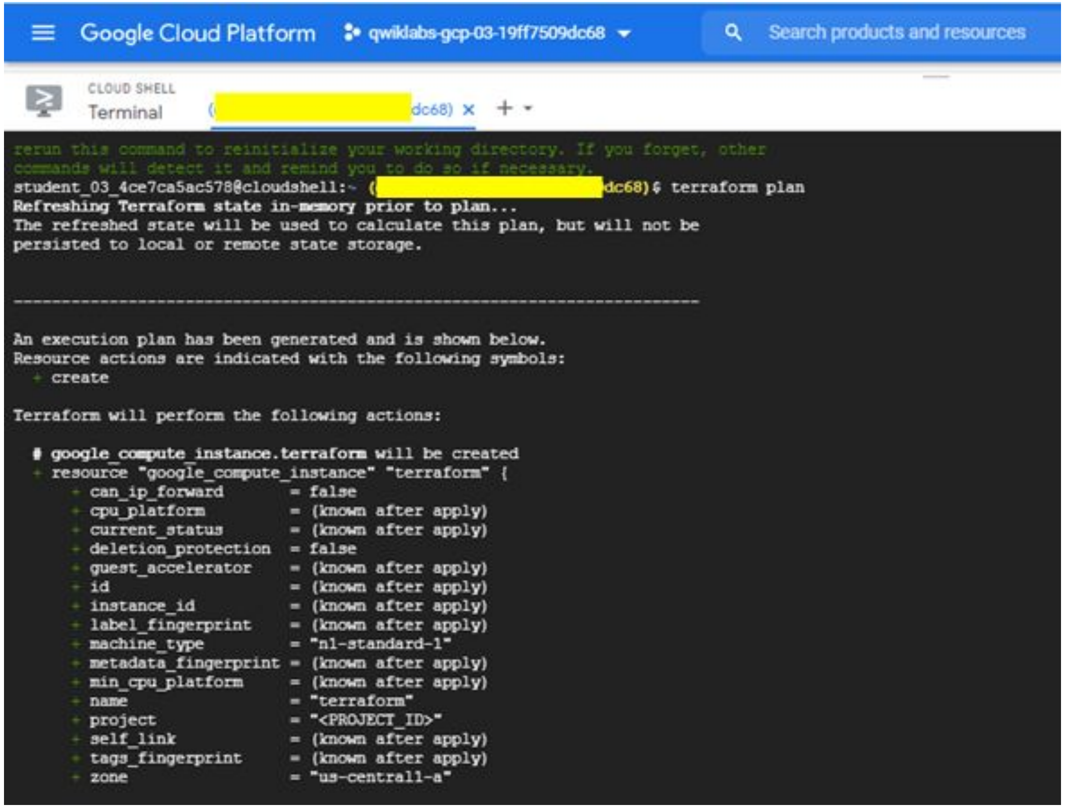
terraform init:
Download and install any provider binary (Google, AWS, Azure)



```
-rw-r--r-- 1 student_03_4ce7ca5ac578 student_03_4ce7ca5ac578 342 Apr  8 06:18 instance.tf
student_03_4ce7ca5ac578@cloudshell:~ ([redacted]dc68)$ terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "google" (hashicorp/google) 3.63.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.google: version = "~> 3.63"

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.
```

A Quick Example using Terraform

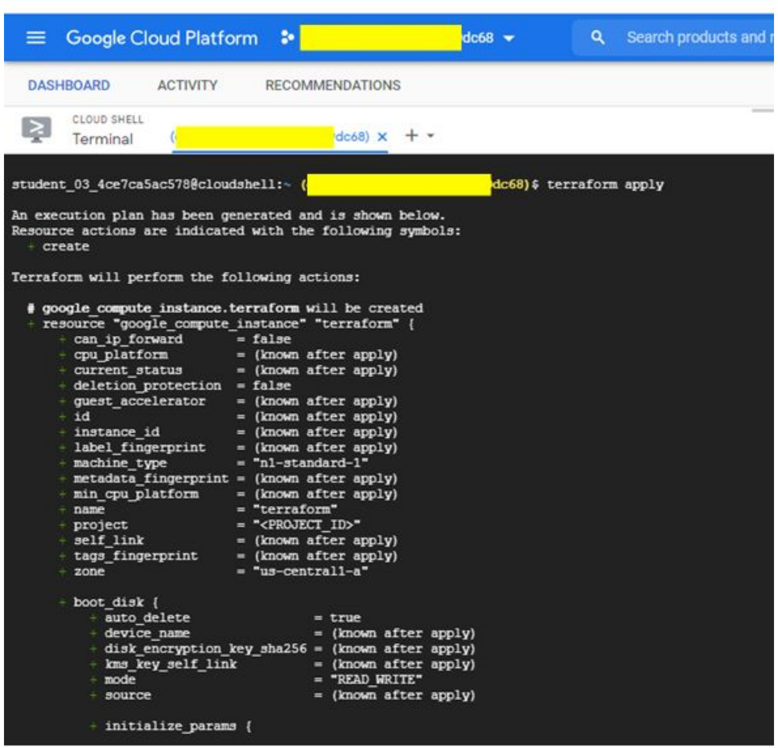
terraform plan: Performs a refresh and then determines what actions are necessary to achieve the desired state specified in the configuration files.



```
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_4ce7ca5ac578@cloudshell:~ ([redacted]dc68)$ terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # google_compute_instance.terraform will be created
  + resource "google_compute_instance" "terraform" {
      + can_ip_forward       = false
      + cpu_platform         = (known after apply)
      + current_status       = (known after apply)
      + deletion_protection  = false
      + guest_accelerator    = (known after apply)
      + id                   = (known after apply)
      + instance_id          = (known after apply)
      + label_fingerprint    = (known after apply)
      + machine_type         = "n1-standard-1"
      + metadata_fingerprint = (known after apply)
      + min_cpu_platform     = (known after apply)
      + name                 = "terraform"
      + project              = "<PROJECT_ID>"
      + self_link            = (known after apply)
      + tags_fingerprint     = (known after apply)
      + zone                 = "us-central1-a"
```

A Quick Example using Terraform

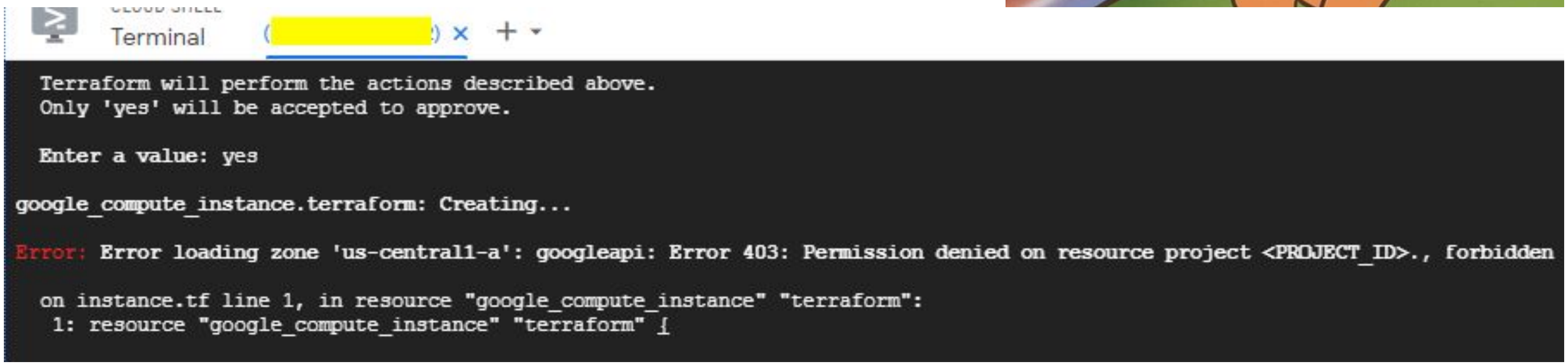
terraform apply: Reads the .tf configuration file, which shows the Execution Plan, and after approval, applies the actions in order to change real infrastructure.



```
student_03_4ce7ca5ac578@cloudshell:~ ([redacted]dc68)$ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # google_compute_instance.terraform will be created
  + resource "google_compute_instance" "terraform" {
      + can_ip_forward       = false
      + cpu_platform         = (known after apply)
      + current_status       = (known after apply)
      + deletion_protection  = false
      + guest_accelerator    = (known after apply)
      + id                   = (known after apply)
      + instance_id          = (known after apply)
      + label_fingerprint    = (known after apply)
      + machine_type         = "n1-standard-1"
      + metadata_fingerprint = (known after apply)
      + min_cpu_platform     = (known after apply)
      + name                 = "terraform"
      + project              = "<PROJECT_ID>"
      + self_link            = (known after apply)
      + tags_fingerprint     = (known after apply)
      + zone                 = "us-central1-a"

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

          + initialize_params {
```

A Quick Example using Terraform

Oops!

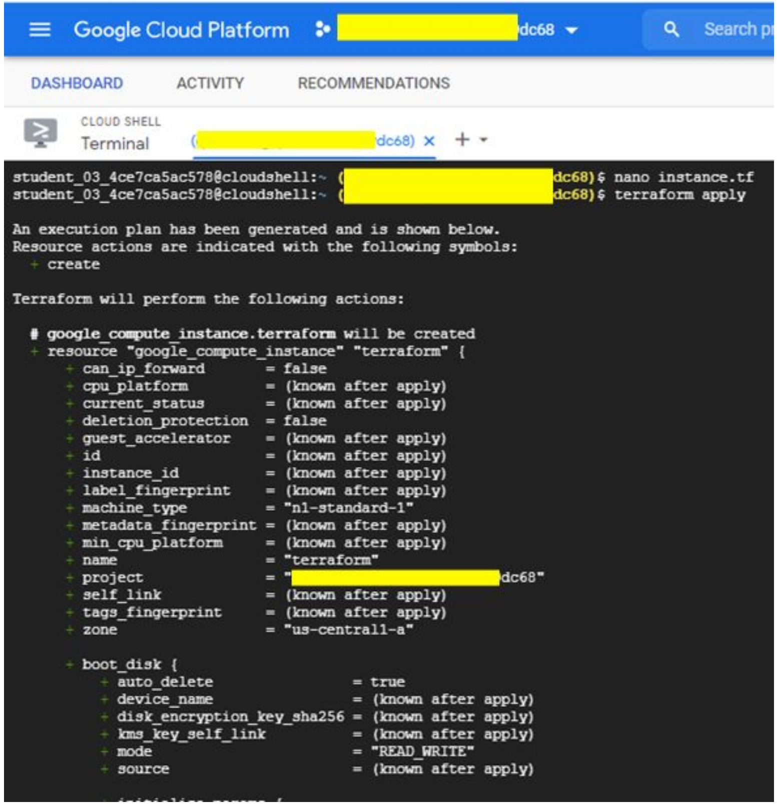


```
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

google_compute_instance.terraform: Creating...

Error: Error loading zone 'us-central1-a': googleapi: Error 403: Permission denied on resource project <PROJECT_ID>., forbidden

  on instance.tf line 1, in resource "google_compute_instance" "terraform":
   1: resource "google_compute_instance" "terraform" {
```

A Quick Example using Terraform

Tip of the day:

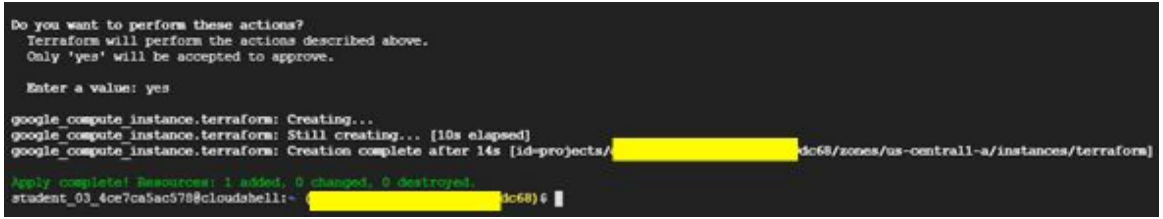
Remember to set the PROJECT_ID before running the apply command



```
student_03_4ce7ca5ac578@cloudshell:~ ([redacted]dc68)$ nano instance.tf
student_03_4ce7ca5ac578@cloudshell:~ ([redacted]dc68)$ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # google_compute_instance.terraform will be created
  + resource "google_compute_instance" "terraform" {
      + can_ip_forward       = false
      + cpu_platform         = (known after apply)
      + current_status       = (known after apply)
      + deletion_protection  = false
      + guest_accelerator    = (known after apply)
      + id                   = (known after apply)
      + instance_id          = (known after apply)
      + label_fingerprint    = (known after apply)
      + machine_type         = "n1-standard-1"
      + metadata_fingerprint = (known after apply)
      + min_cpu_platform     = (known after apply)
      + name                 = "terraform"
      + project              = "[redacted]dc68"
      + self_link            = (known after apply)
      + tags_fingerprint     = (known after apply)
      + zone                 = "us-central1-a"

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

A Quick Example using Terraform

Success!

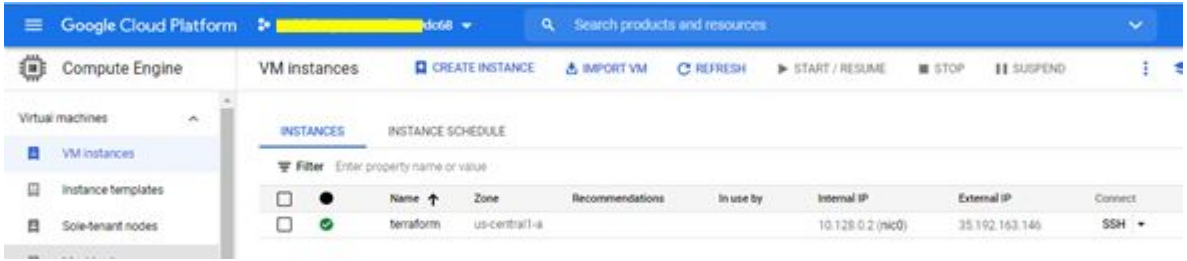


```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

google_compute_instance.terraform: Creating...
google_compute_instance.terraform: Still creating... [10s elapsed]
google_compute_instance.terraform: Creation complete after 14s [id=projects/[redacted]dc68/zones/us-central1-a/instances/terraform]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
student_03_4ce7ca5ac578@cloudshell:~ ([redacted]dc68)$
```

A Quick Example using Terraform

Really?

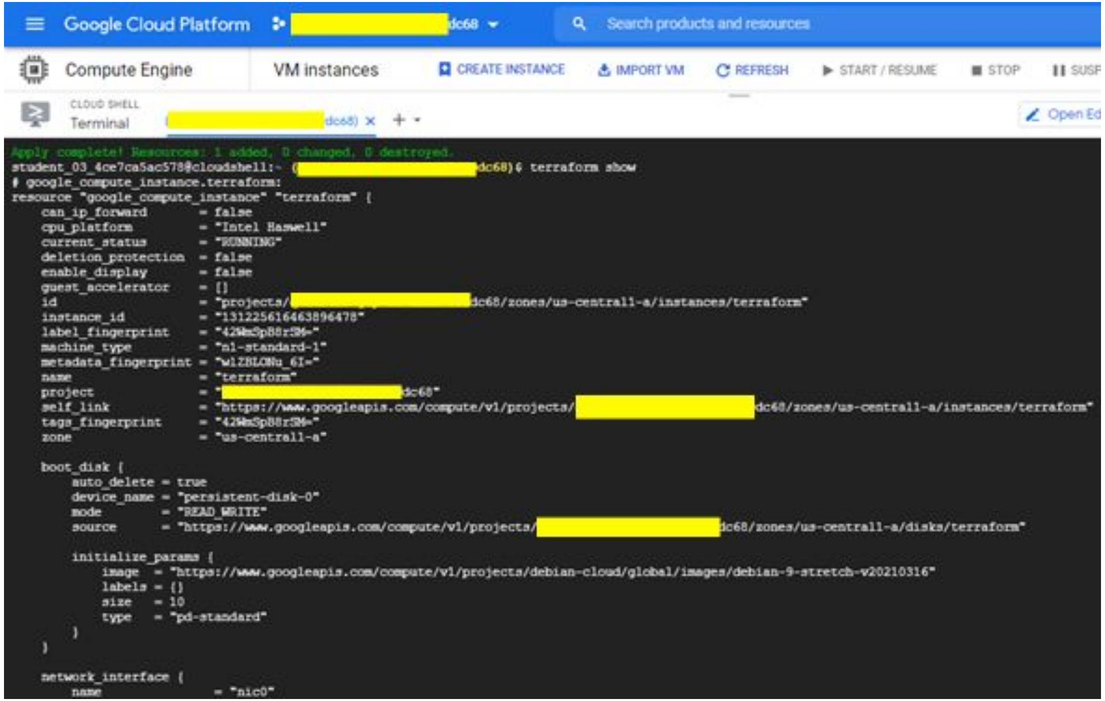
Of course!



□	●	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
□	✓	terraform	us-central1-a			10.128.0.2 (nic0)	35.192.163.146	SSH ▾

A Quick Example using Terraform

terraform show: Inspect the current state:



```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
student_03_4ce7ca5ac578@cloudshell:~ ([redacted]dc68)$ terraform show
# google_compute_instance.terraform:
resource "google_compute_instance" "terraform" {
    can_ip_forward       = false
    cpu_platform         = "Intel Haswell"
    current_status       = "RUNNING"
    deletion_protection  = false
    enable_display       = false
    guest_accelerator    = []
    id                   = "projects/[redacted]dc68/zones/us-central1-a/instances/terraform"
    instance_id          = "1312256164638964 78"
    label_fingerprint    = "42WmSpB8rSM="
    machine_type         = "n1-standard-1"
    metadata_fingerprint = "w1ZBLONu_6I="
    name                 = "terraform"
    project              = "[redacted]dc68"
    self_link            = "https://www.googleapis.com/compute/v1/projects/[redacted]dc68/zones/us-central1-a/instances/terraform"
    tags_fingerprint     = "42WmSpB8rSM="
    zone                 = "us-central1-a"

    boot_disk {
        auto_delete = true
        device_name = "persistent-disk-0"
        mode        = "READ_WRITE"
        source      = "https://www.googleapis.com/compute/v1/projects/[redacted]dc68/zones/us-central1-a/disks/terraform"

        initialize_params {
            image  = "https://www.googleapis.com/compute/v1/projects/debian-cloud/global/images/debian-9-stretch-v20210316"
            labels = {}
            size   = 10
            type   = "pd-standard"
        }
    }

    network_interface {
        name                 = "nic0"
```

THANKS

Does anyone have any questions?

👉 Follow me for more nerdy talks

🌐 https://beacons.ai/davidregalado