Building Scalable Infrastructure in GCP using Terraform Module

v20241201

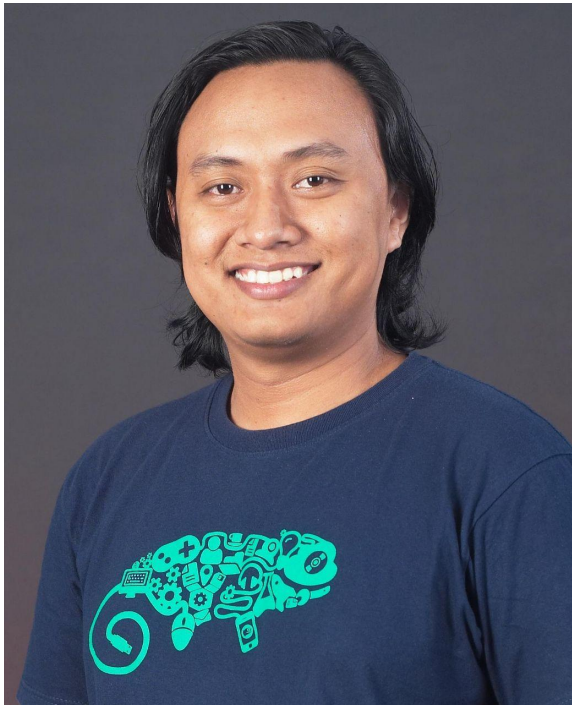
Hello,

Nama saya

Estu Fardani

Praktisi Cloud Specialist
(k8s, terraform, cloud, cicd)

@tuanpembual



Survey Pernah (?)

- Pakai GNU/Linux
- Pakai Git
- Pakai GCP
- Pakai terraform
- Jadi Sysadmin, SRE, SysEng, DevOps, Cloud Platform

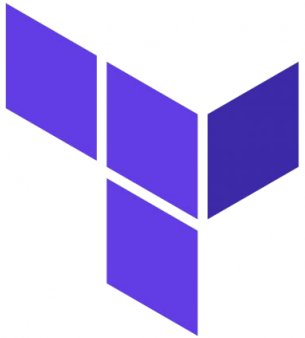
Terraform

Definisi

Terraform adalah **tool open-source** yang dikembangkan oleh HashiCorp.

Digunakan untuk mendefinisikan, mengelola, dan mengimplementasikan infrastruktur IT secara otomatis menggunakan konsep **Infrastructure as Code (IaC)**.

Opsi lain: opentofu



HashiCorp
Terraform

Infrastructure as a Code (IaC)

Praktik penyediaan dan pengelolaan infrastruktur komputasi melalui berkas konfigurasi yang dapat dibaca mesin, yang memungkinkan otomatisasi dan konsistensi

Motivasi Implementasi IaC

Otomatisasi dan Efisiensi

IaC memungkinkan pengaturan infrastruktur secara otomatis menggunakan kode, sehingga mengurangi kebutuhan akan konfigurasi manual yang memakan waktu. Infrastruktur bisa disiapkan dengan cepat dan efisien, terutama dalam skala yang besar.

Motivasi Implementasi IaC

Konsistensi dan Reproduksi

Dengan IaC, konfigurasi infrastruktur menjadi konsisten di berbagai lingkungan (development, staging, production). Karena semua konfigurasi didefinisikan dalam kode, infrastruktur dapat dengan mudah direplikasi, dan mengurangi resiko miskonfigurasi.

Motivasi Implementasi IaC

Version Control

Infrastruktur dapat diperlakukan seperti kode aplikasi lainnya, yang memungkinkan versioning, audit, dan rollback jika terjadi masalah. Hal ini memberi kemampuan untuk melacak perubahan infrastruktur dan menerapkan perubahan dengan aman.

Motivasi Implementasi IaC

Kolaborasi Tim yang Lebih Baik

Tim dapat bekerja bersama pada berkas konfigurasi infrastruktur, sama seperti mengelola kode aplikasi. Ini memungkinkan kolaborasi yang lebih baik, karena semua anggota tim dapat melihat, memeriksa, dan mengubah konfigurasi secara transparan.

Motivasi Implementasi IaC

Mengurangi Risiko Human Error

Proses manual sering kali rentan terhadap kesalahan manusia, seperti salah konfigurasi atau perubahan yang tidak terdokumentasi. Dengan IaC, risiko ini berkurang karena infrastruktur didefinisikan secara sistematis dan dapat diuji sebelum diterapkan.

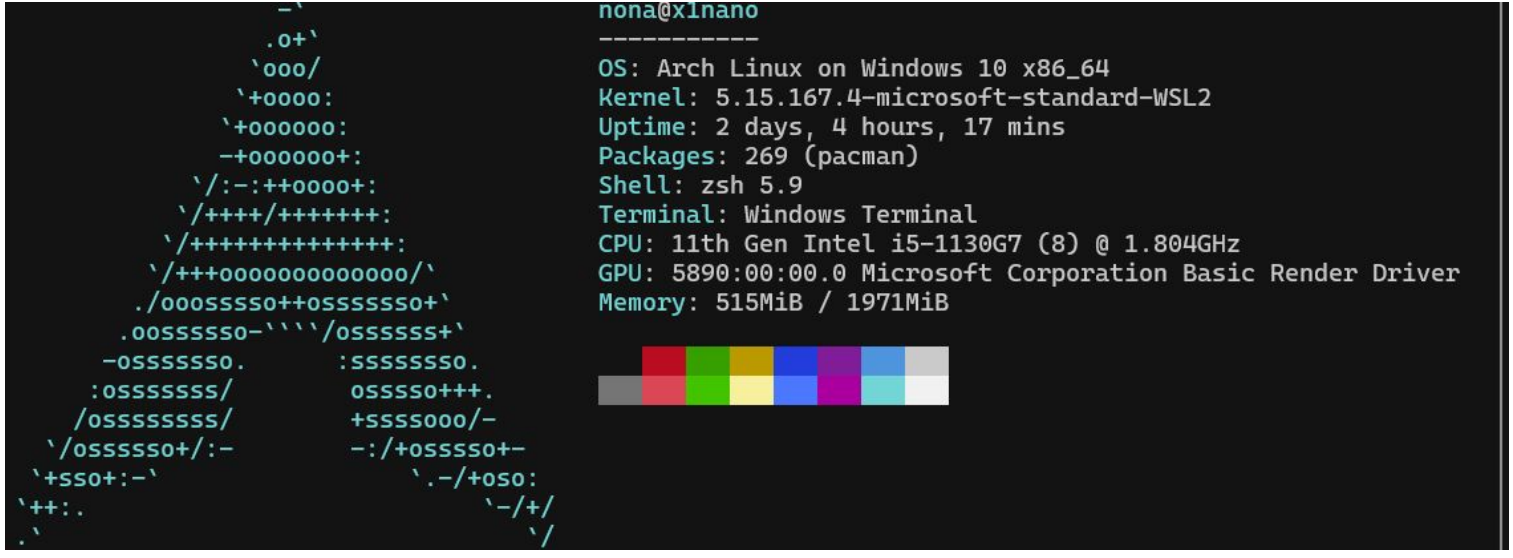
Motivasi Implementasi IaC

Integrasi dengan CI/CD

IaC memungkinkan integrasi yang mudah dengan pipeline CI/CD, sehingga infrastruktur dapat diterapkan atau diperbarui secara otomatis bersamaan dengan rilis kode aplikasi, mempercepat siklus pengembangan dan deployment.

Workshop Environment

- OS: GNU/Linux or WSL2 or OSX
- Terraform Version: 1.8.x
- GCP SDK Installed
- GCP Account
- Git
- Internet



Praktek Satu

Pemasangan Terraform

Terraform dipasang dalam bentuk binary

- Unduh dari situs resmi
- Ekstraksi dan pindahkan pada path /usr/local/bin

```
[11:19:20] ~/vagrant ››› wget https://releases.hashicorp.com/
terraform/1.8.1/terraform_1.8.1_linux_amd64.zip
```

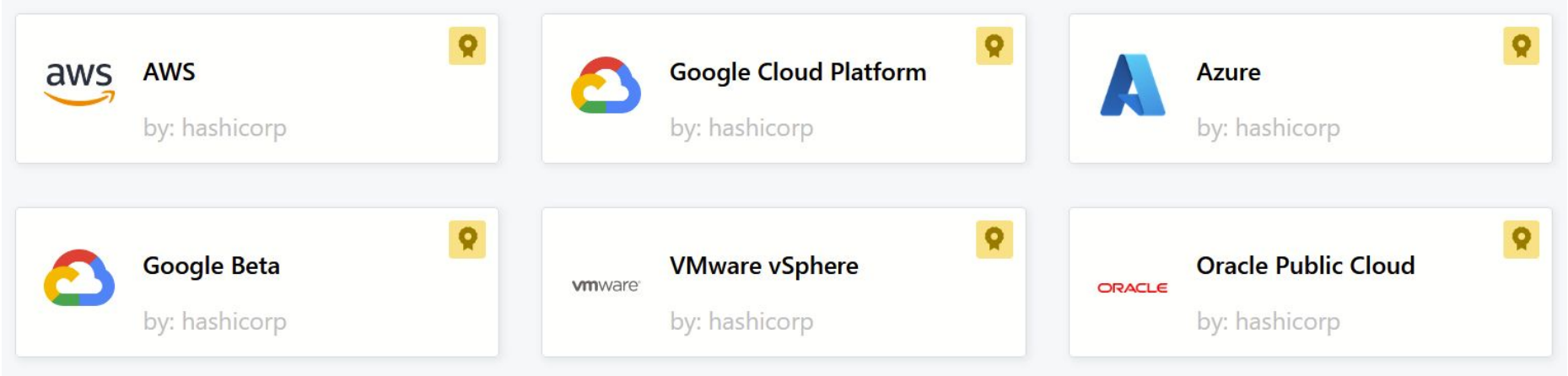
Konfig-Konfig di lokal

- GCloud SDK install
- GCloud Config

```
mkdir ~/bin
wget curl -O
https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-linux-x86_64.tar.gz
tar -xf google-cloud-cli-linux-x86_64.tar.gz
./google-cloud-sdk/install.sh
source ~/.zshrc
gcloud init
gcloud auth login
gcloud auth application-default login
```

TL,DR;

- Konfigurasi server itu ditulis dalam format HCL
- Disimpan dalam berkas *.tf
- Hanya bisa pada cloud provider yang sudah didukung



Membuat Resource

- Clone repo:
  git clone https://github.com/tuanpembual/devfest-gcp-terraform-module.git
- Checkout pada tag v1.0
  git checkout v1.0
- Periksa kode

Praktek Dua

Membuat Resource

Periksa kode di repo: https://github.com/tuanpembual/devfest-gcp-terraform-module.git

Berkas terraform.tf

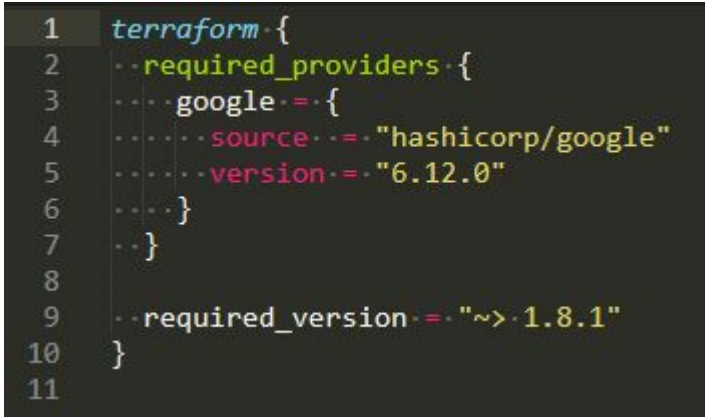
Berisi tentang konfigurasi:

- Provider (line 3-6) adalah plugin yang memungkinkan Terraform berinteraksi dengan layanan di GCP.
- Binary version terraform

Provider GCP memberi Anda akses untuk membuat resource seperti VM instance, network, database dan berbagai resource lainnya di Google Cloud.

Dokumentasi provider Google dapat dilihat di

https://registry.terraform.io/providers/hashicorp/google/6.12.0/docs



Buat sumber daya

Terraform akan membaca semua berkas *.tf pada folder bekerja. Contoh:

- Buat VM di vm.tf
- Buat db di cloudsql.tf

Caranya:

- Buka dokumentasi di https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/compute_instance

Mengoreksi isi berkas vm.tf

- Buat project di gcp dulu
- Buat VM
- Buat Service Account

Resource adalah komponen infrastruktur yang dikelola oleh Terraform. Resource ini bisa berupa virtual machine, database, bucket, jaringan, dan lainnya di GCP.

Resource digunakan untuk mendefinisikan apa saja yang ingin Anda buat di infrastruktur cloud. Dalam contoh di samping, kita akan membuat sebuah VM instance di Google Compute Engine

```
resource "google_service_account" "vm_sa" {
  project      = "jakarta-369305"
  account_id   = "vm-sa-estu"
  display_name = "Custom SA for VM Instance"
}

resource "google_compute_instance" "vm_estu" {
  project      = "jakarta-369305"
  name         = "vm-estu"
  machine_type = "n2-standard-1"
  zone         = "asia-southeast2-a"

  deletion_protection = true

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-11"
      size  = 20
      type  = "pd-balanced"
      labels = {
        name = "disk-vm-estu"
      }
    }
  }

  network_interface {
    network = "default"

    access_config {
      // Ephemeral public IP
    }
  }

  metadata_startup_script = "echo hi > /test.txt"

  service_account {
    email  = google_service_account.vm_sa.email
    scopes = ["cloud-platform"]
  }
}
```

Mengoreksi isi berkas cloudsql.tf

Yang diatar:

- Type DB
- Versi DB
- Region
- Dkk

```
resource "google_sql_database_instance" "db_estu" {
  project          = "jakarta-369305"
  name             = "db-estu"
  database_version = "POSTGRES_15"
  region           = "asia-southeast2"

  deletion_protection = false

  settings {
    tier      = "db-f1-micro"
    disk_size = 10

    disk_autoresize       = true
    disk_autoresize_limit = 11

    backup_configuration {
      enabled                        = true
      location                       = "asia-southeast2"
      point_in_time_recovery_enabled = true
    }

    insights_config {
      query_insights_enabled = true
    }

    maintenance_window {
      day  = 5
      hour = 20
    }

    availability_type = "ZONAL"
  }

  lifecycle {
    ignore_changes = [
      settings[0].disk_size
    ]
  }
}

```

Contoh Implementasi Resource dengan Terraform

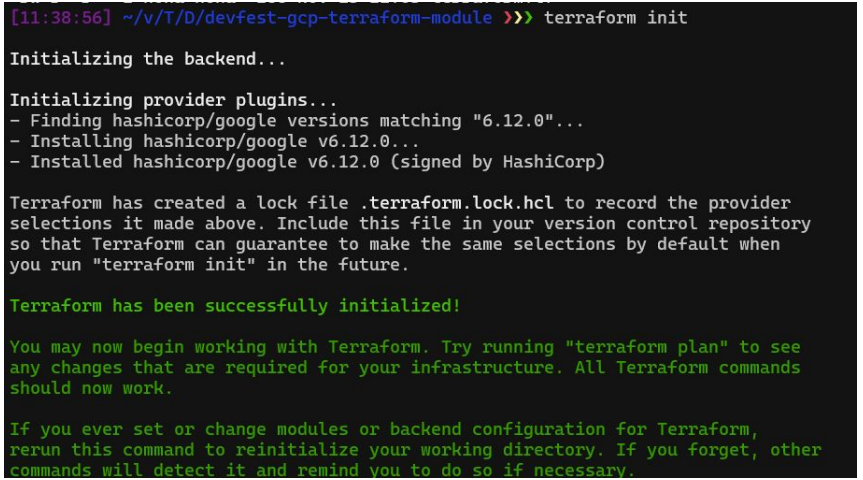
Contoh beberapa sistem yang dibuat dengan menggunakan terraform diantaranya adalah

- Cloud SQL
- IAM Access
- Memory Store (redis)
- Secret Manager
- Pub/Sub
- VPC (Virtual Private Cloud)
- GCLB (Google Cloud Load Balancer)
- GCS (Google Cloud Storage)
- GKE (Google Kubernetes Engine)
- GCE (Google Compute Engine)
- Cloud Armor
- Artifact Registry

Praktek Perintah Dasar

Cuma ada 3 yang sering dipakai:

terraform init #inisiasi
terraform plan #plan aja
terraform apply #plan dan apply



Implementasi Membuat Resource

```
git checkout v2.0
terraform init
terraform plan
terraform apply
```

```
Terraform will perform the following actions:

  # google_compute_instance.vm_estu will be created
  + resource "google_compute_instance" "vm_estu" {
      + can_ip_forward          = false
      + cpu_platform            = (known after apply)
      + creation_timestamp      = (known after apply)
      + current_status          = (known after apply)
      + deletion_protection     = true
      + effective_labels        = {
          + "goog-terraform-provisioned" = "true"
        }
      + id                      = (known after apply)
      + instance_id             = (known after apply)
      + label_fingerprint       = (known after apply)
      + machine_type            = "n2-standard-2"
      + metadata_fingerprint    = (known after apply)
      + metadata_startup_script = "echo hi > /test.txt"
      + min_cpu_platform        = (known after apply)
      + name                    = "vm-estu"
      + project                 = "jakarta-369305"
      + self_link               = (known after apply)
      + tags_fingerprint        = (known after apply)
      + terraform_labels        = {
          + "goog-terraform-provisioned" = "true"
        }
      + zone                    = "asia-southeast2-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)
```

```
google_service_account.vm_sa: Creating...
google_service_account.vm_sa: Still creating... [10s elapsed]
google_service_account.vm_sa: Creation complete after 14s [id=projects/jakarta-369305/serviceAccounts/vm-sa-estu@jakarta-369305.iam.gserviceaccount.com]
```

Praktek Tiga

Terraform - State

Kita pada umumnya tidak berinteraksi langsung dengan berkas Terraform State. Terraform state adalah berkas yang menyimpan status dari infrastruktur yang dikelola oleh Terraform.

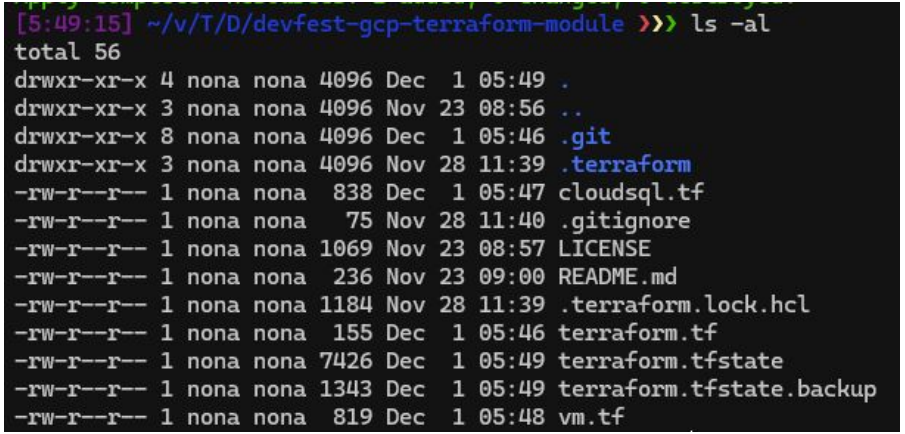
State melacak resource yang sudah dibuat, diperbarui, atau dihapus oleh Terraform, dan digunakan untuk memetakan antara resource di konfigurasi Terraform dengan resource yang ada di *upstream* (GCP)

Fungi State

- Menjaga Konsistensi
- Dependency Tracking

Kondisi Saat Ini

- TF state = database
- Disimpan di lokal `terraform.tfstate`
- Kalo hilang di lokal, tf apply akan delete and create resource baru? :D
- Tidak bisa kolaborasi
- Maka~



Managemen Backend TF State

Menyimpan TF state di awan, caranya:

- Simpan di GCS
- Simpan di AWS S3
- Terraform Cloud/Enterprise Backend

Cara mengatur TF state di GCS

Langkah-langkahnya:

1. Buat bucket gcs
2. Atur Terraform.tf
3. Migrate TF State

```
resource "google_storage_bucket" "terraform_state" {
  project       = "jakarta-369305"
  name          = "terraform-state-backend-estu"
  location      = "asia-southeast2"
  force_destroy = true

  versioning {
    enabled = true
  }

  lifecycle_rule {
    action {
      type = "Delete"
    }
    condition {
      age = 90
    }
  }
}

```

Membuat GCS via terraform

Langkahnya:

```
git checkout v3.0
terraform apply
```

```
Terraform will perform the following actions:

  # google_storage_bucket.terraform_state will be created
  + resource "google_storage_bucket" "terraform_state" {
      + effective_labels            = {
          + "goog-terraform-provisioned" = "true"
        }
      + force_destroy               = true
      + id                          = (known after apply)
      + location                    = "ASIA-SOUTHEAST2"
      + name                        = "terraform-state-backend-estu"
      + project                     = "jakarta-369305"
      + project_number              = (known after apply)
      + public_access_prevention    = (known after apply)
      + rpo                         = (known after apply)
      + self_link                   = (known after apply)
      + storage_class               = "STANDARD"
      + terraform_labels            = {
          + "goog-terraform-provisioned" = "true"
        }
      + uniform_bucket_level_access = (known after apply)
      + url                         = (known after apply)

      + lifecycle_rule {
          + action {
              + type          = "Delete"
                # (1 unchanged attribute hidden)
            }
          + condition {
              + age                   = 90
              + matches_prefix        = []
              + matches_storage_class = []
              + matches_suffix        = []
              + with_state            = (known after apply)
                # (3 unchanged attributes hidden)
            }
        }

      + versioning {
          + enabled = true
        }
    }

Plan: 1 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

google_storage_bucket.terraform_state: Creating...
google_storage_bucket.terraform_state: Creation complete after 2s [id=terraform-state-backend-estu]
```

Mengatur Backend dan Migrasi TF State

Ubah berkas terraform.tf (baris 10-14), kemudian run:

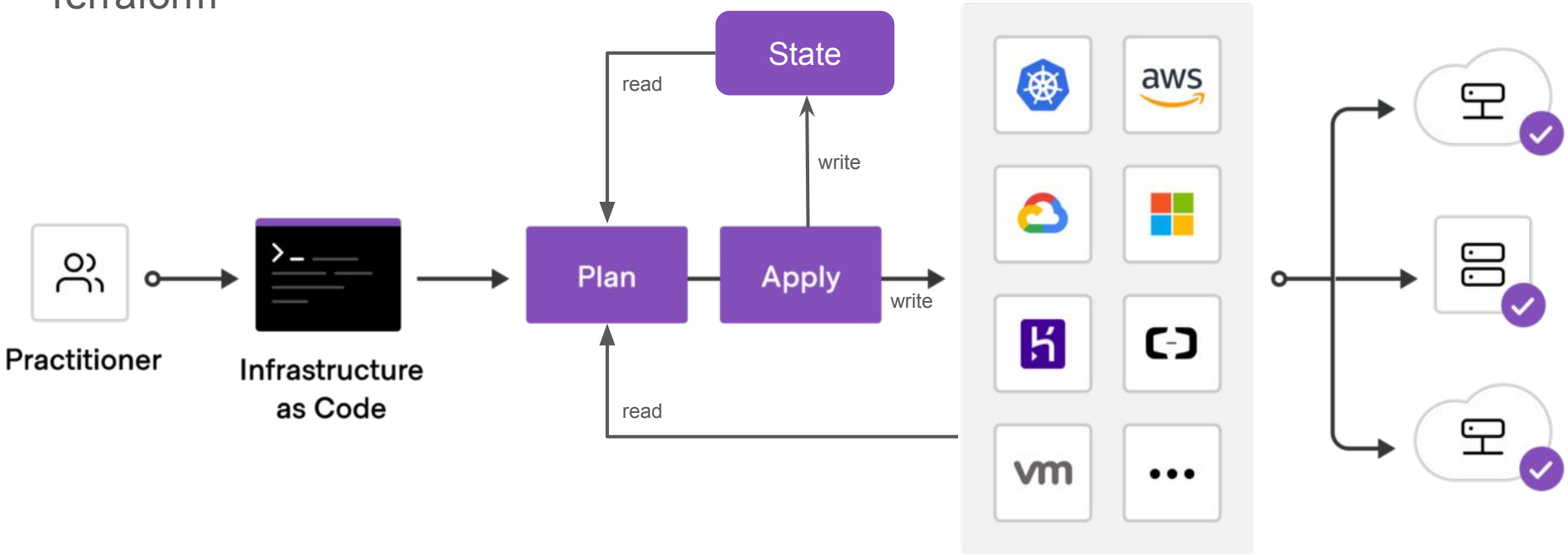
`terraform init`

```
1  terraform {
2    required_providers {
3      google = {
4        source  = "hashicorp/google"
5        version = "6.12.0"
6      }
7    }
8
9    required_version = "~> 1.8.1"
10
11   backend "gcs" {
12     bucket = "terraform-state-backend-estu"
13     prefix = "state"
14   }
15 }
16
```

```
[10:40:03] ~/v/T/D/devfest-gcp-terraform-module ››› terraform init

Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "gcs" backend. No existing state was found in the newly
  configured "gcs" backend. Do you want to copy this state to the new "gcs"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes


Successfully configured the backend "gcs"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Reusing previous version of hashicorp/google from the dependency lock file
- Using previously-installed hashicorp/google v6.12.0

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.
```

Terraform - Cara Kerja

Terraform



Praktek Empat

Terraform - Modules

```
resource "google_service_account" "vm_sa" {
  project      = "jakarta-369305"
  account_id   = "vm-sa-estu"
  display_name = "Custom SA for VM Instance"
}

module "vm_estu" {
  source                 = "./modules/compute"
  name                   = "vm-estu"
  metadata_startup_script = "echo hi > /test.txt"

  service_account = google_service_account.vm_sa.email
}
```

Module adalah kumpulan resource Terraform yang *reusable*.

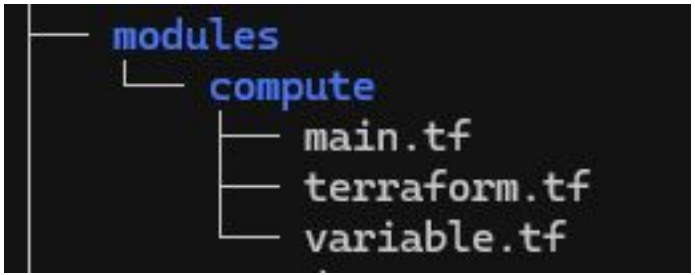
Layaknya fungsi pada bahasa pemrograman module membantu mengelompokkan resource yang umum dan memungkinkan penggunaan kembali konfigurasi dengan cara yang lebih terstruktur.

Kapan module digunakan?

- Standarisasi resource
  - Contoh: Untuk compute engine harus menggunakan OS debian-11
- Mengurangi baris kode yang ditulis pengguna terraform (engineer)

Menulis Module

- Siapkah folder khusus
- Tambahkan berkas module
  - terraform.tf
  - main.tf
  - variable.tf
- Refer path module ke berkas sumberdaya tf (vm.tf)



Menulis Module (2)

Berkas terraform.tf berisi:

- Provider version
- Binary version

```
terraform {
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = ">= 6.12.0"
    }
  }
  required_version = ">= 1.8.1"
}

```

Menulis Module (3)

Berkas variable.tf berisi:

- List value mandatory
- Tulis value yang ada di berkas vm.tf
- Isi default value sebagai inputan yang tidak lagi diubah-ubah

```
variable "name" {
  type = string
}

variable "machine_type" {
  type    = string
  default = "n2-standard-2"
}

variable "project" {
  type    = string
  default = "jakarta-369305"
}

variable "network" {
  type    = string
  default = "projects/jakarta-369305/global/networks/default"
}

variable "subnetwork" {
  type    = string
  default = "projects/jakarta-369305/regions/asia-southeast2/subnetworks/default"
}

variable "zone" {
  type    = string
  default = "asia-southeast2-a"
}

variable "deletion_protection" {
  type    = bool
  default = true
}

variable "disk_image" {
  type    = string
  default = "debian-cloud/debian-11"
}

variable "disk_size" {
  type    = number
  default = 20
}
```

Menulis Module (4)

Berkas main.tf berisi:

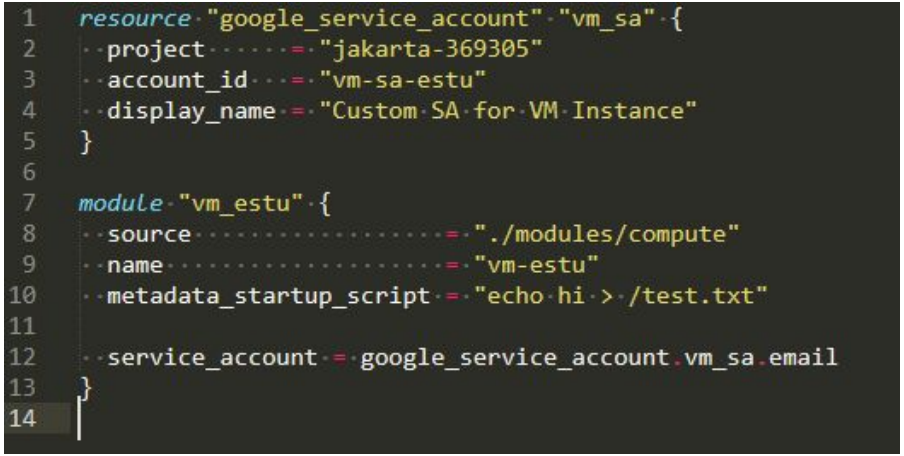
- Main config as write code
- Deklarasi lokal jika dibutuhkan

```
locals {
  metadata = {
    "enable-oslogin"        = "true"
    "block-project-ssh-keys" = "true"
  }
}

resource "google_compute_instance" "compute" {
  name                = var.name
  machine_type        = var.machine_type
  zone                = var.zone
  project             = var.project
  deletion_protection = var.deletion_protection

  boot_disk {
    initialize_params {
      image  = var.disk_image
      size   = var.disk_size
      type   = var.disk_type
      labels = var.disk_labels
    }
  }

  network_interface {
    network    = var.network
    subnetwork = var.subnetwork

    dynamic "access_config" {
      for_each = var.is_public ? [true] : []

      content {
        nat_ip = var.public_ip_address
      }
    }
  }

  metadata_startup_script = var.metadata_startup_script

  metadata = merge(var.metadata, local.metadata)

  service_account {
    email  = var.service_account
    scopes = var.scopes
  }
}
```

Menggunakan Module

Ubah Berkas vm.tf:

- Pasang path module
- Hapus baris yang sudah diatur dari modul
- Atur variable value yang dibutuhkan



Implementasi Penggunaan Module

```
git checkout v4.0
terraform init
terraform plan
terraform apply
```

```
[11:40:42] ~/v/T/D/devfest-gcp-terraform-module ››› terraform init && terraform plan

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/google from the dependency lock file
- Using previously-installed hashicorp/google v6.12.0

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.
google_storage_bucket.terraform_state: Refreshing state... [id=terraform-state-backend-estu]
```

```
# module.vm_estu.google_compute_instance.compute will be created
+ resource "google_compute_instance" "compute" {
    + can_ip_forward          = false
    + cpu_platform            = (known after apply)
    + creation_timestamp      = (known after apply)
    + current_status          = (known after apply)
    + deletion_protection     = true
    + effective_labels        = {
        + "goog-terraform-provisioned" = "true"
      }
    + id                      = (known after apply)
    + instance_id             = (known after apply)
    + label_fingerprint       = (known after apply)
    + machine_type            = "n2-standard-1"
    + metadata                = {
        + "block-project-ssh-keys" = "true"
        + "enable-oslogin"         = "true"
      }
    + metadata_fingerprint    = (known after apply)
    + metadata_startup_script = "echo hi > /test.txt"
    + min_cpu_platform        = (known after apply)
    + name                    = "vm-estu"
    + project                 = "jakarta-369305"
    + self_link               = (known after apply)
    + tags_fingerprint        = (known after apply)
    + terraform_labels        = {
        + "goog-terraform-provisioned" = "true"
      }
    + zone                    = "asia-southeast2-a"
```

Praktek Lima

Clean up

- Set baris kode sebagai komentar, `terraform apply`

Next todo: Expand beyond boundaries

Apa ya...

- Terraform Import State
- Koding di terraform
  - Validation
- Integrasi CICD
  - Request Resource
  - Plan
  - Apply
  - Drift

```
#-an-05-2023
locals {
  #-sharding-is-for-computer_test-instances
  cloudsql_computer_test_shard_count  = 20
  cloudsql_bank_soal_shard_count      = 4
  sizing_name                         = "medium"
  sql_sizing = {
    "production" = {
      log          = { cpu : 8, memory : 16, connection_limit : "2000", activation_policy : "ALWAYS" }
      auth         = { cpu : 24, memory : 96, connection_limit : "40000", activation_policy : "ALWAYS" }
      soal         = { cpu : 16, memory : 32, connection_limit : "20000", activation_policy : "ALWAYS" }
      computer_test = { cpu : 24, memory : 128, connection_limit : "20000", activation_policy : "ALWAYS" }
      migrator     = { cpu : 2, memory : 4, connection_limit : "1000", activation_policy : "ALWAYS" }
    },
```

```
72  count = local.cloudsql_bank_soal_shard_count == 1 ? 1 : local.cloudsql_bank_soal_shard_count
73
74  project            = module.project.project_id
75  environment        = module.project.environment
76  disk_size          = 500
77  instance_name      = "bank-soal-db-${count.index}"
78  network            = google_compute_network.vpc_anbk_production.id
79  database_version   = "POSTGRES_14"
80  deletion_protection = true
81
82  cpu = (
83    length(lookup(local.bank_soal_patch, "${count.index}", [])) > 0 && contains(keys(local.sql_sizing), local.sizing_name) ?
84    lookup(local.bank_soal_patch, "${count.index}").cpu :
85    lookup(local.sql_sizing, local.sizing_name).soal.cpu
86  )
```

Catatan

- Repo ujicoba: https://github.com/tuanpembual/devfest-gcp-terraform-module

Q&A

1.