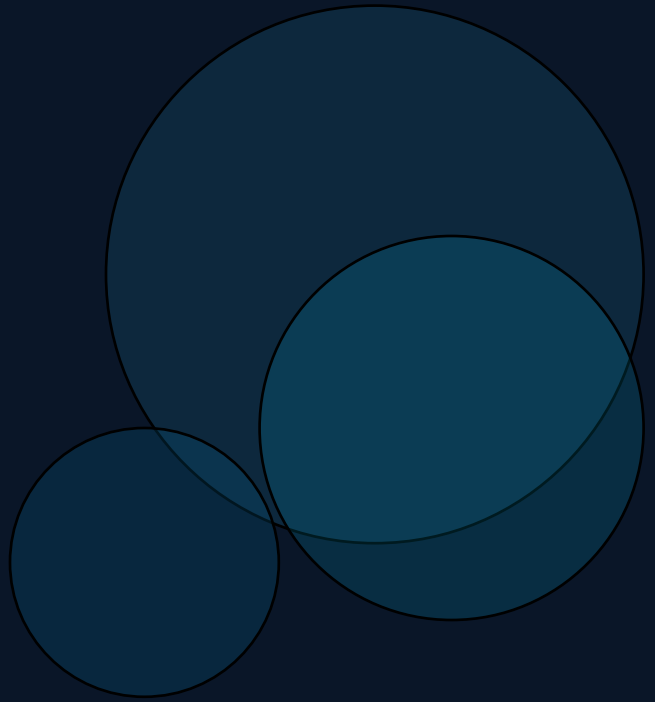
Introducing Terraform Modules: Making Infrastructure As Code Easier

Putu Sintia



Hi!! I’m Sintia

Balinese IE with 2 years experience in cloud infrastructure, DevOps and observability.

More about me, you can see in my linktree with scan the barcode here or go to the link below

https://linktr.ee/putusintia

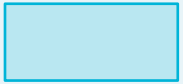
Agenda

01 **Selamat Tinggal Copy-Paste**

Identifikasi masalah & analogi modul

02 **Anatomi Terraform Module**

Root vs Child, Local vs Remote, struktur standar



03 **Live Demo: Bangun Cetakannya**

Praktik local module dari nol

03+ **Bonus: Terragrunt & common_tags**

Advanced — level up kode 100% DRY

04 **Kesimpulan & Q&A**

Rekap + next steps + diskusi

Before We Start

This session assumes you:

- **Know basic Terraform syntax (resource, variable, output)**
- **familiar with provider basics (for example in this session is AWS)**

Not there yet? No worries. Follow along — the concepts will still make sense.
Resources to catch up are shared at the end.

Bayangkan Skenario Ini...

Kamu diminta deploy infrastruktur untuk 2 environment:

Staging

```
aws_vpc
aws_subnet
aws_security_group
aws_instance
aws_eip
...dan seterusnya
```

COPY PASTE

Production

```
aws_vpc
aws_subnet
aws_security_group
aws_instance
aws_eip
...dan seterusnya
```

- **Melelahkan**
- **Rentan Error**
- **Sulit Dipelihara**

Lihat Perbedaannya

❌ Tanpa Modul

```
# staging/main.tf
resource "aws_instance" "stg" {
  ami           = "ami-abc123"
  instance_type = "t3.micro"
  sg_ids = [aws_sg.stg.id]
}

# production/main.tf  ← copy!
resource "aws_instance" "prd" {
  ami           = "ami-abc123"
  instance_type = "t3.large"
  sg_ids = [aws_sg.prd.id]
}
```

✅ Dengan Modul

```
# staging/main.tf
module "app" {
  source         = "../modules/ec2"
  instance_type  = "t3.micro"
  instance_count = 3
}

# production/main.tf
module "app" {
  source         = "../modules/ec2"
  instance_type  = "t3.large"
  instance_count = 5
}
```

Analogi: Terraform Module = Cetakan Agar-Agar

1

Satu Cetakan

Kamu membuat satu modul (blueprint) yang berisi semua logika infrastruktur

2

Dipanggil Berkali-kali

Staging, Production, UAT — semua memanggil modul yang sama

3

Beda Rasa

Hanya input/variabel nya yang berbeda per environment

Root vs Child Module

Root Module

Folder utama → tempat kamu jalankan terraform apply

Child Module

Folder 'cetakan' yang dipanggil oleh Root Module

Local vs Remote Module

Local

```
source = "./modules/ec2"
```

Subfolder di komputer / repo kamu sendiri

Remote

```
source = "git::github.com/..."
```

GitHub, Terraform Registry, S3, dll

Struktur Standar: Satu modul minimal = 3 file

`main.tf`
kode utama

`variables.tf`
input parameter

`outputs.tf`
nilai dikembalikan

Membangun Modul dari Nol

1

Pecah Folder

Pindahkan blok EC2 & Security Group ke direktori modules/

```
modules/
  ec2/
    main.tf
    variables.tf
    outputs.tf
```

2

Jadikan Dinamis (Inputs)

Gunakan variables.tf agar Root bisa kirim data berbeda

```
variable "instance_count" {
  type    = number
  default = 1
}
```

3

Lempar-Tangkap Data (Outputs)

Modul Security Group lempar ID, modul EC2 tangkap & pakai

```
# outputs.tf (sg module)
output "sg_id" {
  value = aws_security_group.main.id
}
```

Bagian 3+ : Level Up ke Terragrunt

Masalah Baru: Setiap resource wajib punya tag (Project, Environment, Version, ManagedBy).
Tanpa solusi, kita akan copy-paste tag di setiap modul dan environment. Kembali ke masalah semula.

Apa itu Terragrunt?

- Wrapper cerdas di atas Terraform
- Bukan pengganti — pelengkap!
- Tujuan: eliminasi redundansi kode
- Mantra: Don't Repeat Yourself (DRY)

Senjata Rahasia: common_tags

```
# terragrunt.hcl (root — 1 file!)
inputs = {
  common_tags = {
    Project     = "myapp"
    Environment = local.env
    Version     = "v1.2.0"
    ManagedBy   = "Terragrunt"
  }
}
```

Update versi? Ubah 1 baris angka → seluruh resource di AWS ter-update otomatis

Key Takeaways

1

Write Once, Use Anywhere

Satu modul dipanggil berkali-kali untuk environment berbeda

2

Kurangi Human Error

Perubahan di satu tempat berlaku untuk semua — tidak ada yang kelewatan

3

Kode Rapi & Maintainable

Struktur standar membuat kolaborasi tim jauh lebih mudah

4

DRY dengan Terragrunt

Eliminasi redundansi total — investasi setup awal yang sangat worthwhile

Resources untuk Belajar Lebih Lanjut

- developer.hashicorp.com/terraform/language/modules
- registry.terraform.io
- terragrunt.gruntwork.io/docs
- Buku: Terraform Up & Running (O'Reilly)

Ada Pertanyaan?

Terima Kasih!