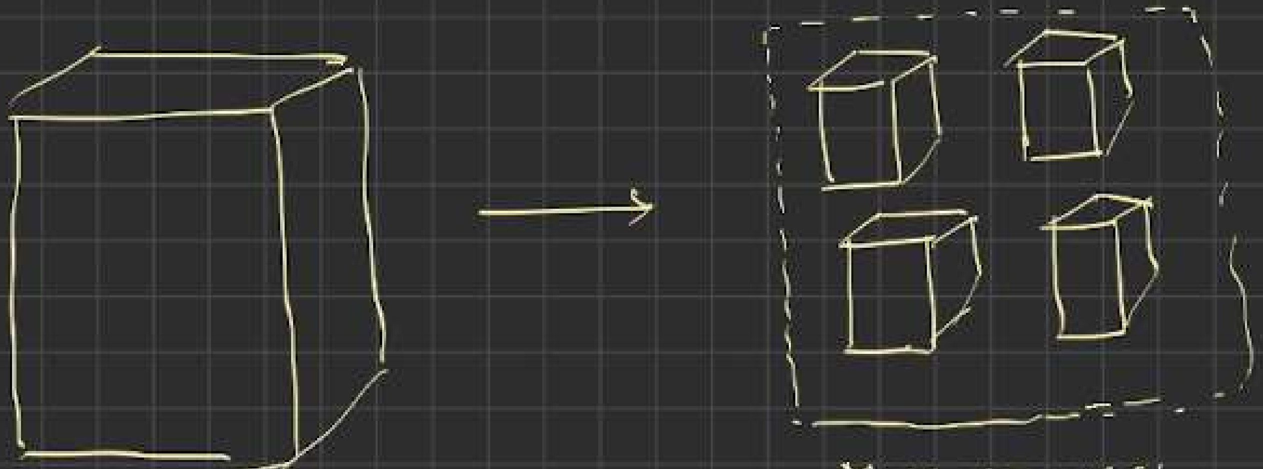
TERRAFORM MODULE ⤳ reusable code



MONOLITHIC

Disadv

→ takes lot of time to fix bug

→ maintainance is difficult

→ team coordination is challenging (support) as no ownership

MICROSERVICES

Adv

→ modular approach

→ module wise bug fixing is easy

→ managing is easy.

→ requires less downtime.

Terraform follows modular approach.

Adv of "terraform.tfvars" file

You can input values in terraform.tfvars file and you can git-ignore this file while pushing your code to github.

It means, anyone using your scripts doesn't have to know the values (ami, region, subnet-id etc) you have used, they can use their own tfvars file.

If you mention "terraform.tfvars" file,
when you run "terraform apply" ⤵
terraform automatically
checks for terraform.tfvars

Output variables/outputs in terraform

↳ why it is required?

Lets say your junior dev using your script,
can run your terraform script but don't have
permissions to check the resources in the console,
in this case, how will he know what is the
public IP address of the created EC2 instance?

Output → will display all the details.
↳ you will get the information of the
resource you have created.

Example: outputs.tf

```
output "public-ip-address" {
    value = aws_instance.example.public-ip
}
```

vs code plugins can help you with syntax

Q. using terraform apply → created EC2 instance. ⤵ instance is up & running.
what happens if I execute
"terraform apply" again without changing anything

Sol: If you accidently or wantedly run the
"terraform apply" again, it says:
No changes. Your infrastructure matches the
configuration. Terraform has compared your real infra
against your configuration and found no difference.

Note :

terraform registry → publicly available modules like dockerhub

Exercise : Terraform Modules practice

SENIOR DEV

↓

created terraform script for EC2 and S3
He has modularised the code.

JUNIOR DEV

↓

has to use module and spin EC2 & S3

Sol : Refer files under code.

SENIOR DEV

① create main.tf { providers / resources }

② create variables.tf

③ create terraform.tfvars

④ create outputs.tf { public IP / S3 name }

⑤ create modules ↓
ec2.tf
s3.tf

JUNIOR DEV

⑥ create main.tf
↓
call modules used by SENIOR DEV

⑦ terraform plan
terraform apply

⑧ validate the output