

①

TERRAFORM

→ HashiCorp Language

Why Terraform ?

competitors { pulumi, crossplane }

Universal approach

- AWS (CloudFormation)
- AZURE (Resource Manager)
- GCP [Cloud Deployment Manager (CDM)]
- OPENSTACK [HCL]

Converted as
AWS API or
Azure API ...

GitHub Codespace } Sandbox or container env
github provides for free (60 hrs)
> dev 2 CPU, 4 GB RAM

main.tf ← holds main configuration
of terraform script

②

provider "aws" {
region = "us-east-1"

}

This step will validate if terraform has access to "aws" (i.e., aws api)

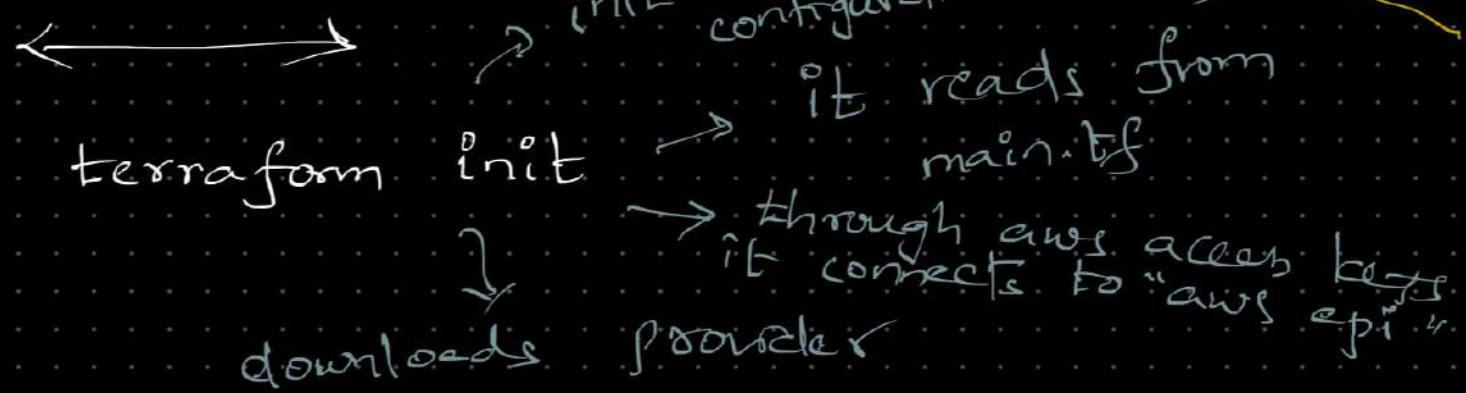
example

resource "aws-instance" "example" {

ami = "ami-12345"

instance-type = "t3.micro" # specify

}



Terraform plan ← like a dry-run

← shows the resources it is going to create

Terraform apply ← if all the details (ami, instance type ...) are correct then it creates ec2-instance.

Note : Hashicorp HCL }
Hashicorp Terraform }
 } plugins in vs code

Key-pairs on AWS side } useful to login
 } to instance

PROJECT 1 : Create EC2-instance using terraform

So before creating EC2 using terraform,

lets review on what is required to
create an EC2 instance manually

- ① AWS account, IAM → Terraform uses this to talk to AWS
- ② Region → You must choose where the server will run
- ③ AMI → This is the OS Image of your server
 ↳ AMI = No EC2
- ④ Instance Type → This decides CPU & RAM
- ⑤ Network (VPC + Subnet) → Your EC2 must live inside the network
 ↳ default: default VPC
- ⑥ Security Group (Firewall) → controls:
 - which ports are open
 - who can access EC2
- ⑦ Key Pair (optional)
 } SSH into server
 } without this, you cannot login to E

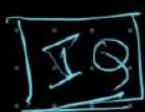
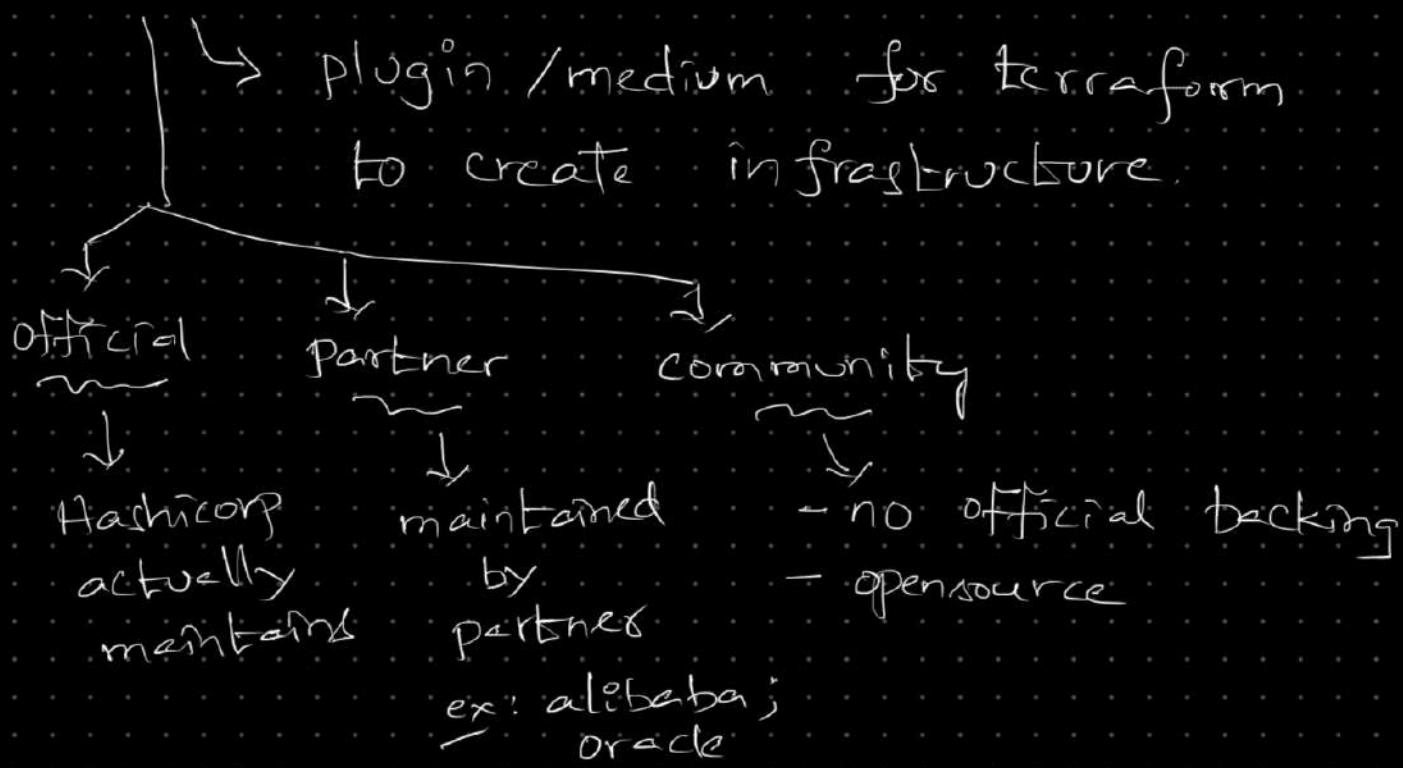
DAY 2

TERRAFORM

①

- ① Understanding Providers & Resources
- ② Variables & Outputs in Terraform
- ③ Conditional Expressions & Functions
- ④ Debugging & Formatting Terraform Files

PROVIDERS



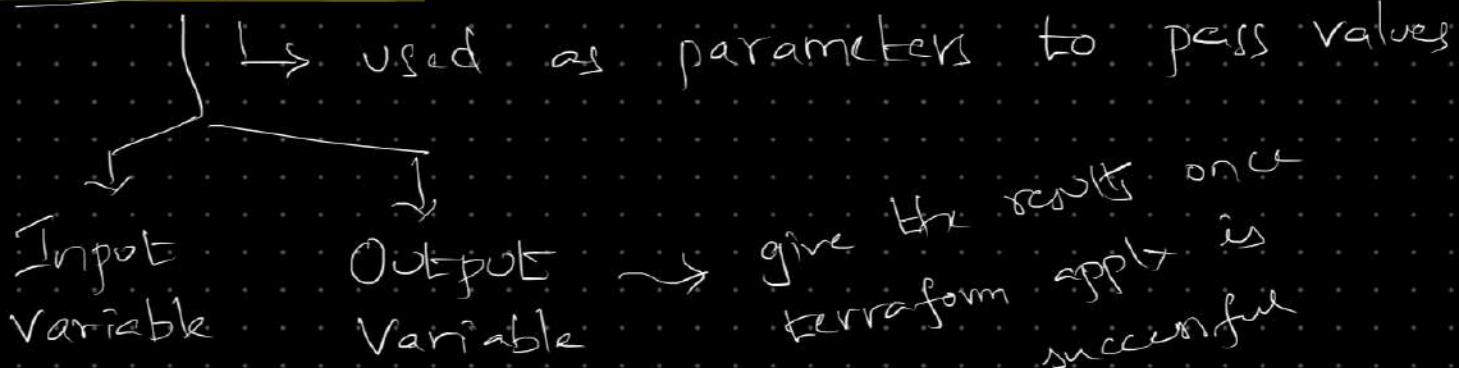
How do you setup terraform in multi region? → using alias

How do you setup terraform in multi cloud? or multi provider?

↳ refer documentation and write code block for each provider.

(2)

VARIABLES



example : <Input Variables>

resource "aws_instance" "example-instance" {
 ami = var.ami_id ← calling a variable
 instance_type = var.instance_type

(define variables either in separate file or in main.tf)

variable "ami_id" {

description = "EC2 AMI ID" ← not mandatory
 type = string

}

variable "instance-type"

description = "EC2 Instance Type"

type = string

default = "t3.micro"

}

Example :

output "public-ip" {

description = "Public IP address of EC2"

value = aws.instance.example-instance.public-ip

}

You are telling terraform to give me that information

real time ↴

→ main.tf

→ provider.tf

→ output.tf

→ input.tf

→ terraform.tfvars *

Adv ↓

easy to change

the variable values / pass the values in this file

ex: dev → value1

stage → value2

prod → value3

} easy to pass

} different values

if you

change the name of tfvars file, then
terraform apply example.tfvars ↴

CONDITIONAL EXPRESSIONS

(4)

→ like if/else

Syntax:

```
[condition ? true-val : false-val]
```

example:

production_subnet-cidr = 10.0.1.0/24

dev_subnet-cidr = 10.0.2.0/24

cidr-blocks = var.environment == "production" ?
 ↳ condition

[var.production-subnet-cidr] ; [var.dev-subnet.cidr]

↓
true

↓
false

BUILT-IN FUNCTIONS

example:

```
output "my-map" {  
    value = map(var.keys, var.values)  
}
```

↳ here, this built-in function provides output in key-value format

ex: "name" = "Esha"
 "age" = 25