

①

# TERRAFORM

→ Hashicorp Language

competitors

pulumi

crossplane

Why Terraform?

Universal approach

- AWS (CLOUD FORMATION)
- AZURE (RESOURCE MGR)
- GCP [CLOUD DEPLOYMENT MGR (CDM)]
- OPENSACK

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 = "A12345"  
  instance-type = "t3-micro" # specify  
  }  
  appropriate  
  AMI ID
```

← initializes terraform  
configuration (terraform)

←

terraform init

→ it reads from  
main.tf

↓  
downloads

→ through aws access keys  
it connects to "aws api"

provider

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

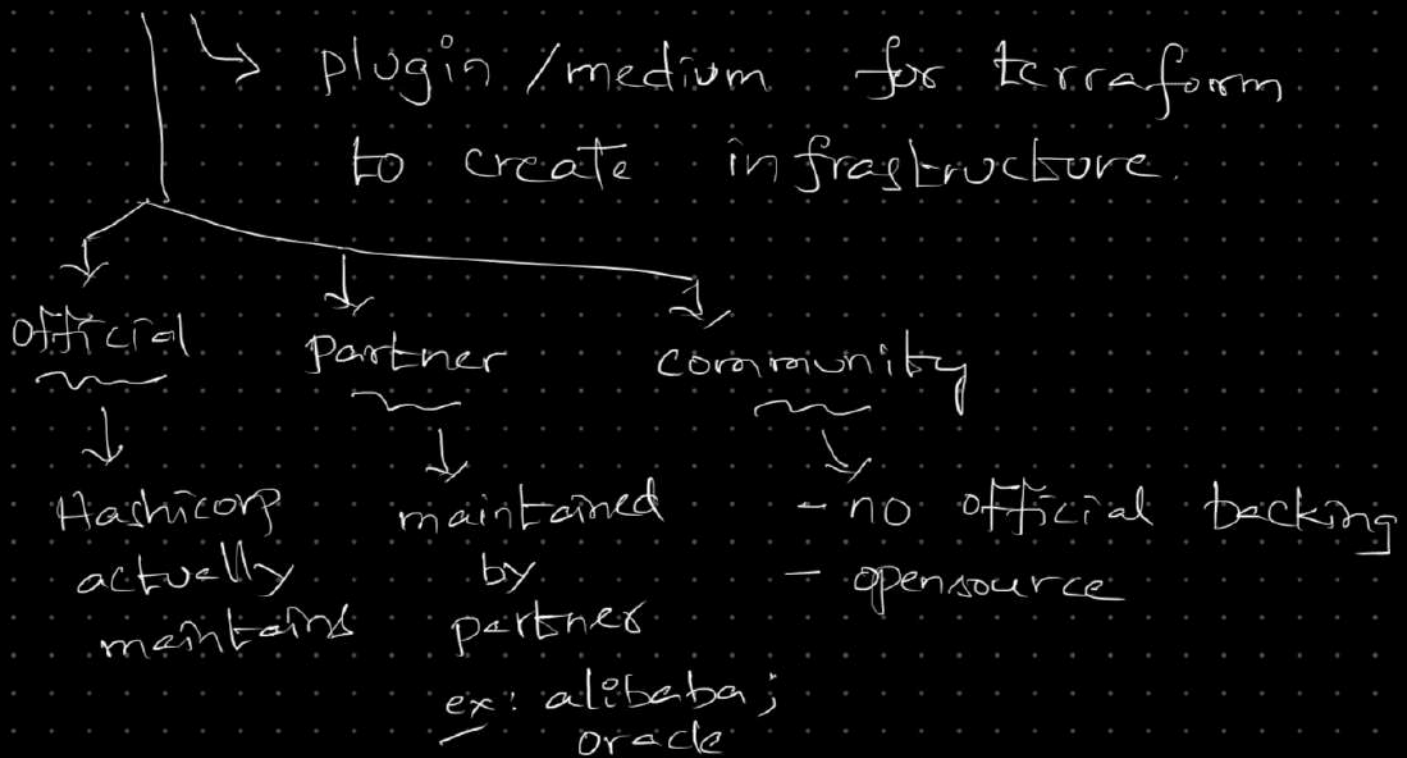
PROJECT 1: Create ec2-instance using terraform

<sup>sd</sup> before creating ec2 using terraform,  
lets revise 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  
No 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

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

## PROVIDERS



**IQ**

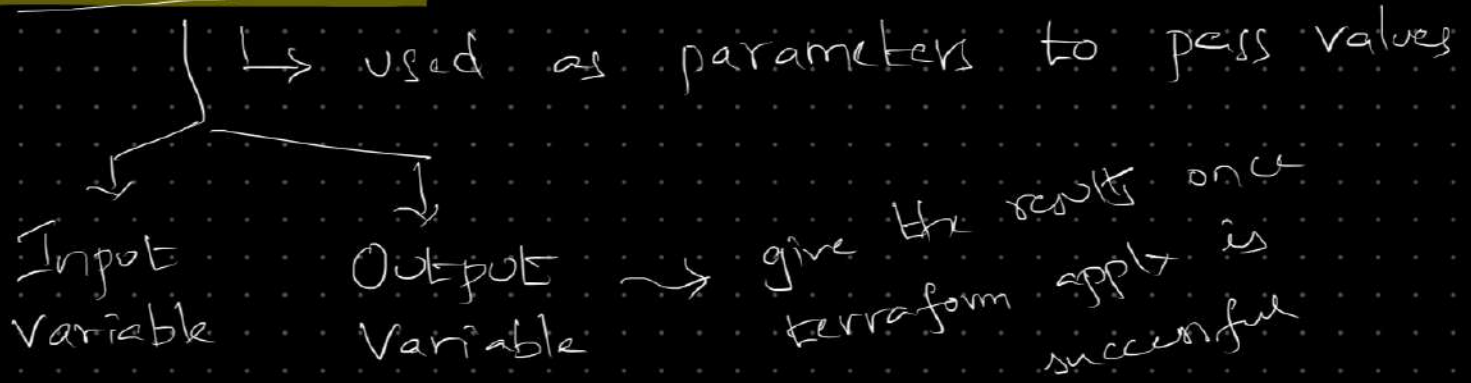
How do you setup terraform in multi region? → using alias

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

→ refer documentation and write code block for each provider.



# 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

}

name of the variable

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

actual value of variables

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