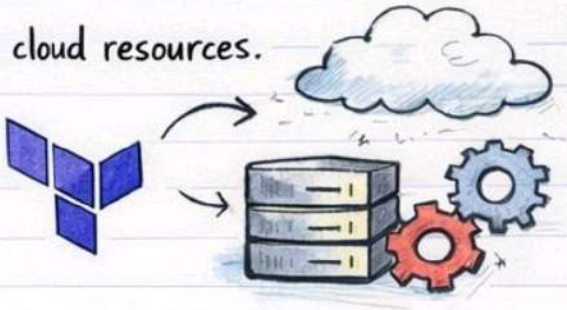
TERRAFORM DAY 1:

FUNDAMENTALS + SETUP

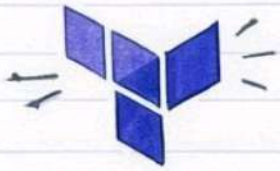
1. What is Terraform?

- Infrastructure as Code (IaC) tool to manage cloud resources.
  - Cloud-Agnostic
  - Version-Controlled
  - Repeatable & Scalable
  - Immutable Infrastructure



2. Terraform Architecture

- Terraform CLI
- Providers (AWS, Azure, GCP)
- State File
- Resources



3. Install Terraform

- Download & Add to PATH
- terraform version

4. Core Commands

- terraform init - Initialize
- terraform plan - Preview Changes
- terraform apply - Deploy Infra
- terraform destroy - Remove Infra

5. Basic File Structure

- provider.tf
- main.tf
- variables.tf
- outputs.tf

6. First Terraform Code:

(AWS EC2 Example)

```
provider "aws" {
  region = "us-east-1"
}
resource "aws_instance "demo" {
   ami = "ami-0obcd1234"
   instance_type = "t2.micro"
}
```

7. Workflow Steps

- Write Code
- terraform init
- terraform plan
- terraform apply

TERRAFORM DAY 2:

VARIABLES, OUTPUTS & STATE

1. What Are Variables?

- Reusable inputs for dynamic Terraform code (like parameters).

t2.micro → dev

Uses them like:

```
instance_type = var.instance_type
```

2. Variable Types:

- string
- number
- bool
- list (string)
- map(string)
- object({}), object(+) x N

3. Outputs:

- Expose resource attributes.

```
output "instance_ip" {
  value = aws_instance
  demo.public_ip
}
```

Example:
- bucket = "tf-state-bucket"
- key = "prod/terraform.tfstate"

4.

- terraform.tfstate

- Tracks real infra
- Never edit manually!
- Sensitive data inside
- Remote backend recommended!

5. State Commands:

- terraform state list
- terraform state show <resource>
- terraform state rm <resource>

5. State Commands:

- Remote State ss
- Enable State Encrypt
- Enable State Versioning→

TERRAFORM DAY 3:

RESOURCES, MODULES & DEPLOYMENT

1. Resource Dependencies

- Terraform Auto-Detects Dependencies.

```
depends_on = [aws_security_group.sg]
```

2. Meta-Arguments

- count
- for_each
- depends_on
- lifecycle

3. Lifecycle Rules

- Control resource lifecycle.

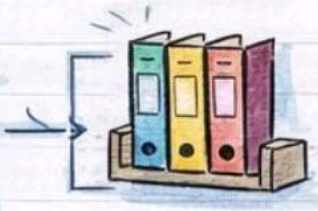
```
lifecycle {
    prevent_destroy = true
    ignore_changes = [tags]
}
```

- Reusable infrastructure
- Isolated code
- DRY (Don't Repeat Yourself)1

4. Terraform Modules (Most Important)

- Reusable Infrastructure components

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



Example:

```
module "ec2" {
    source = "../modules/ec2"
}
```

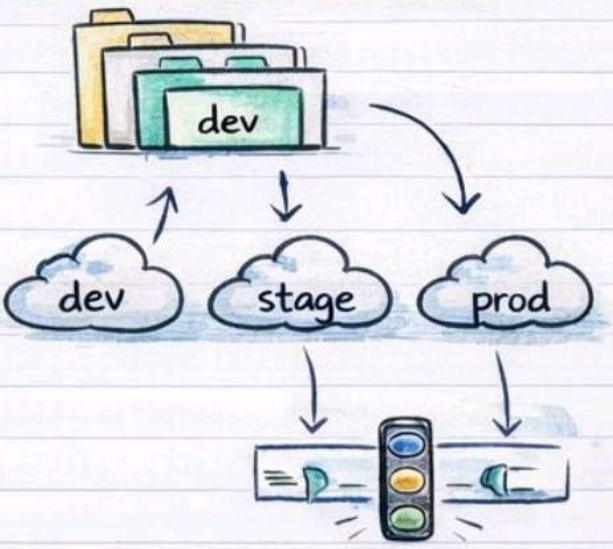
5. Reference Terraform Registry:

- Find terrat.fform.io
- - VPC module
- - EKS module
- - RDS module

6. Environment Separation

- Into folders: dev
  stage
  prod

6. Environment Separation



Best Practices:

- Use Modules for Everything
- One Module = One Purpose
- Version Your Modules

TERRAFORM DAY 4:

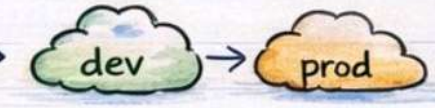
ADVANCED & SECURITY

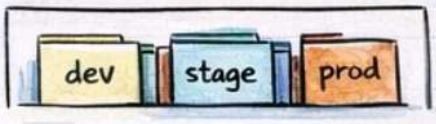
(PRODUCTION LEVEL)

1. Workspaces (Multi-Environments)

- Separate State for Each Env (dev, stage, prod).

```
terraform workspace new dev
terraform workspace select prod
terraform workspace list
```





- Alternatives: Separate Folders (dev, stage, prod)

2. Secrets Management (NOT HARDCODED)

- BAD:
  - password = "admin123"
- GOOD:
  - AWS Secrets Manager
  - Azure Key Vault
  - Environment Variables

Environment Variables

- ✔ export AWS_ACCESS_KEY_ID=xxx
- ✔ export AWS_SECRET_ACCESS_KEY=yyy

Never Commit Your Secrets!

4. Data Sources

- Get Info about Existing Resources.

```
data "aws_vpc" "default" {
  default = true
}
```

5. Terraform fmt

- terraform validate

6. Drift Detection

- Catch Manual Infrastructure Changes

```
terraform plan
```

5. Terraform Formatting & Validation

- terraform fmt
- terraform validate

Best Practices (Production Level)

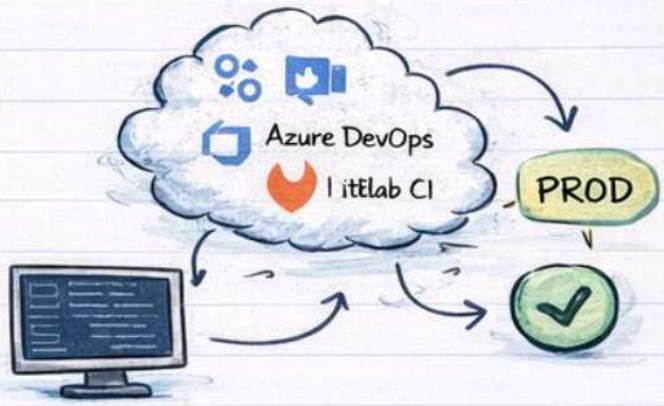
- ✔ NEVER COMMIT STATE FILES !!
- ✔ Use .gitignore (Exclude *.tfstate)
- ✔ Least Privilege IAM Policies (Secure)

TERRAFORM DAY 5:

CI/CD + PRODUCTION LEVEL

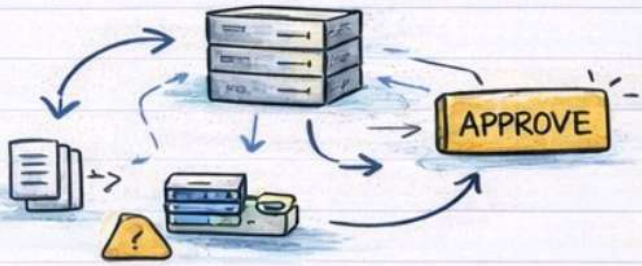
1. Terraform in CI/CD

- Typical pipeline:
  - Validate
  - Plan
  - Apply (Manual Approve)
- Tools: GitHub Actions
  - Azure DevOps
  - GitLab CI



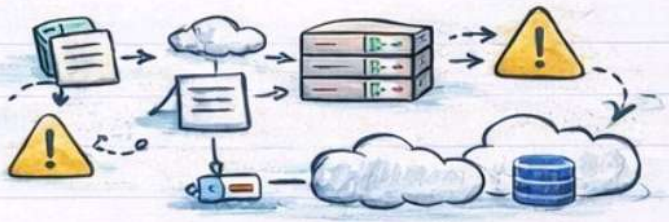
2. Example GitHub Actions Flow

- Checkout Code
- Setup Terraform
- Run Plan
- Apply with Approval (Manual)



3. Real-World Production Project (Practice)

- VPC
- Subnets
- Internet Gateway
- EC2
- Security Groups
- Load Balancer



→ Highly Recommended!

4. Terraform Folder Structure (Production)

- modules/
- envs/
  - dev
  - prod
- backend.tf
- provider.tf

5. Terraform Interview Questions

- What is Terraform state?
- Difference between count and for_each?
- How to secure state file?
- How Terraform handles dependencies?
- What are modules?

READY TO WORK AS
CLOUD / DEVOPS / PLATFORM ENGINEER!

TERRAFORM REAL INTERVIEW TICKETS

(WITH EXPECTED ACTION)

TICKET 1: Terraform apply failed due to state lock

- Error: Error acquiring the state lock

What interviewer expects:

- Terraform uses state locking to prevent consurrent changes
- Lock usually stored in

→ terraform force-unlock <LOCK_ID>

Resolution:

- Another pipeline running?
- Previous apply crashed?

TICKET 3: Sensitive data exposed in terraform.tfstate

Scenario: Passwords visible in state file

Expected Answer:

- Same backend / Same state file
- Fix Options:
  - Separate backend keys:
  - Use workspaces. →

TICKET 4: Multiple environments overwriting each other

Root Cause: Dev apply affects prod infra

Fix Options: Same backend / same state file

```
↳ terraform workspace new dev
↳ terraform workspace new prod
```

TICKET 5: Terraform apply failed – provider version mismatch

- Fix: Lock provider versions
  - terraform apply -var-file-def-vives

TICKET 3: Two EC2 instances needed dynamically | Expected tHecks:

- Plan done, apply blocked
- / Large resources (EKS, RDS)
- / API throitling
- / Dependency chaining

TICKET 9: Terraform plans long time to apply failed in CI/CD

- Plan done, apply blocked
- Expected Answer:

```
lifecycle {
  ignore_changes = wurt
}
```

TICKET 8: Two EC2 instances needed dynamically

- Count = var_instance_demo

or for_each = var_instances

TICKET 10: Terraform cannot find existing VPC

Solution (Data Source):

- data "aws_vpc" "default"{
  default = true

TICKET 11: Tive code deleted terraform.Istaic to prod

Solution:

```
terraform import
  aws_instance.demo i-0123456789
```

NEXT I CAN GIVE YOU:

- ☑ POFs ☑ Mock Terraform interview (Q & A)
- ☑ End-torend Terraform ANX/Azure project
- ☑ Terraform cheatsheet (1 pags)