#30 Days of Terraform

#Day_01

1.1) Before getting into the terraform we need to know three things.

1. What is IaC
2. Why do we need terraform?
3. How does it work?

1.2) Why Infrastructure as Code?

- **Consistency**: Identical environments across dev, staging, and production
- **Time Efficiency**: Automated provisioning saves hours of manual work
- **Cost Management**: Easy to track costs and automate cleanup
- **Scalability**: Deploy to hundreds of servers with same effort as one
- **Version Control**: Track changes in Git
- **Reduced Human Error**: Eliminate manual configuration mistakes
- **Collaboration**: Team can work together on infrastructure

1.3) How Terraform Works

Write Terraform files → Run Terraform commands → Call AWS APIs through Terraform Provider

Terraform Workflow Phases:

1. `terraform init` - Initialise the working directory
2. `terraform validate` - Validate the configuration files
3. `terraform plan` - Create an execution plan
4. `terraform apply` - Apply the changes to reach desired state
5. `terraform destroy` - Destroy the infrastructure when needed

Topic : What is IaC

Provisioning of Resources through Code

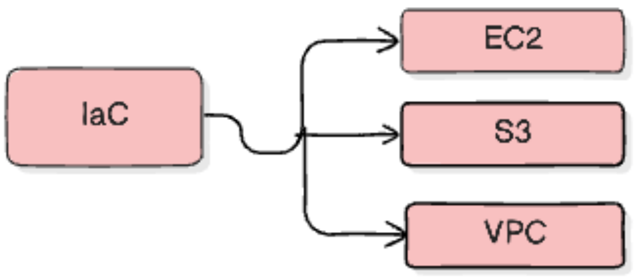

Fig 1.1. IaC Flowchart

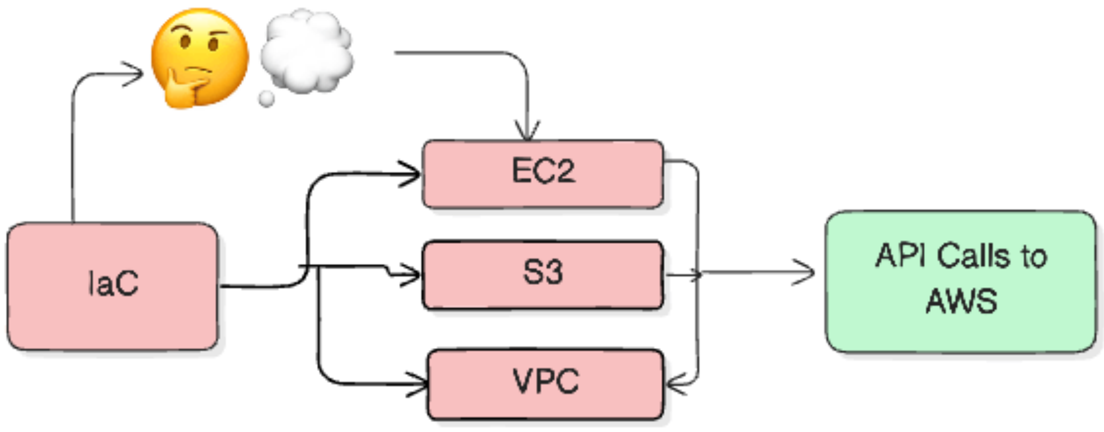

Fig 1.2. How Does terraform Interacts with AWS?

eraser