Terraform

Raman Khanna

Introduction

Your Name

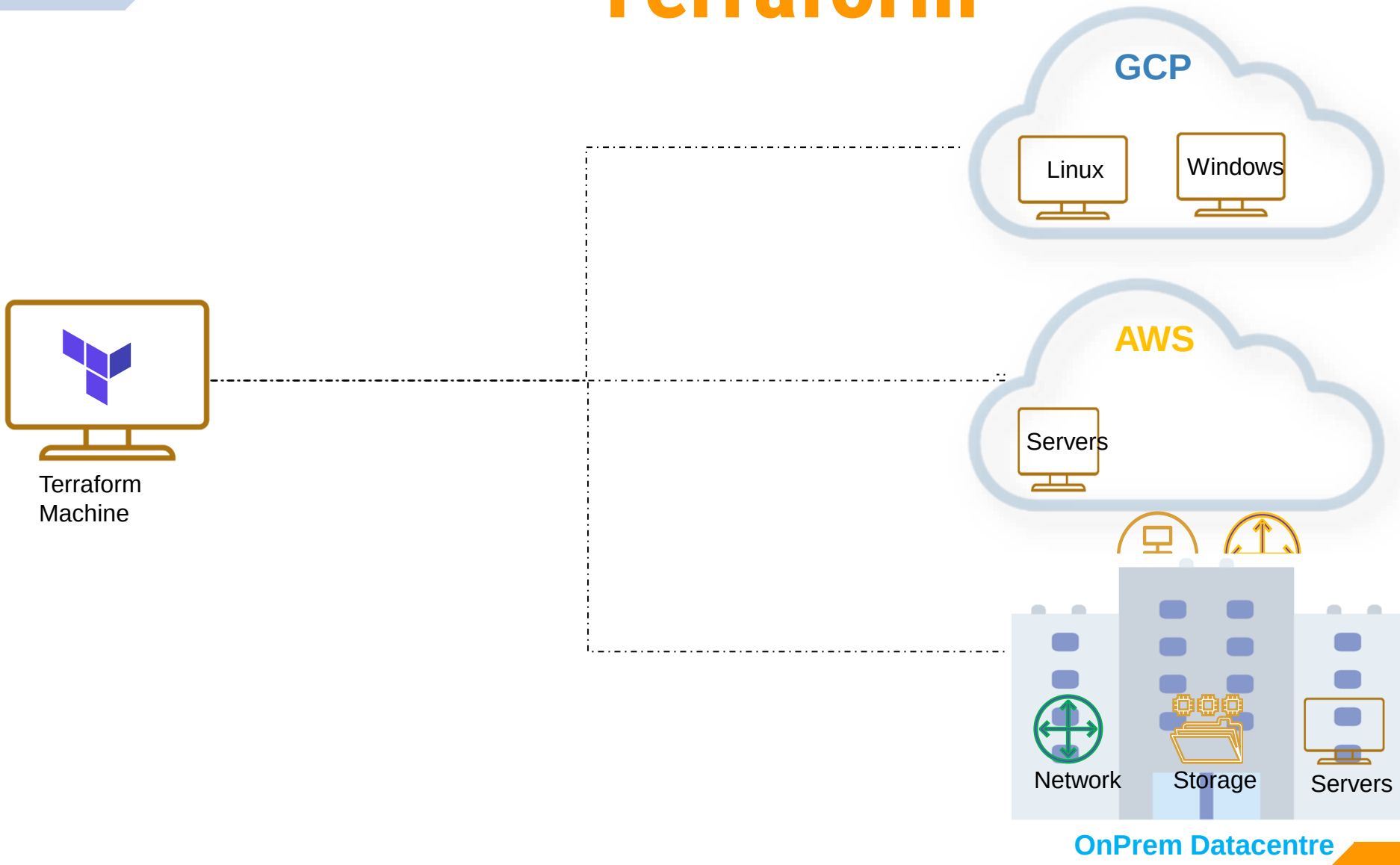
Total experience

Background – Development / Infrastructure / Database / Network

Experience on AWS Cloud and Terraform

What is Orchestration?

Terraform


GCP
Linux
Windows
AWS
Servers
Terraform
Machine
Network
Storage
Servers
OnPrem Datacentre

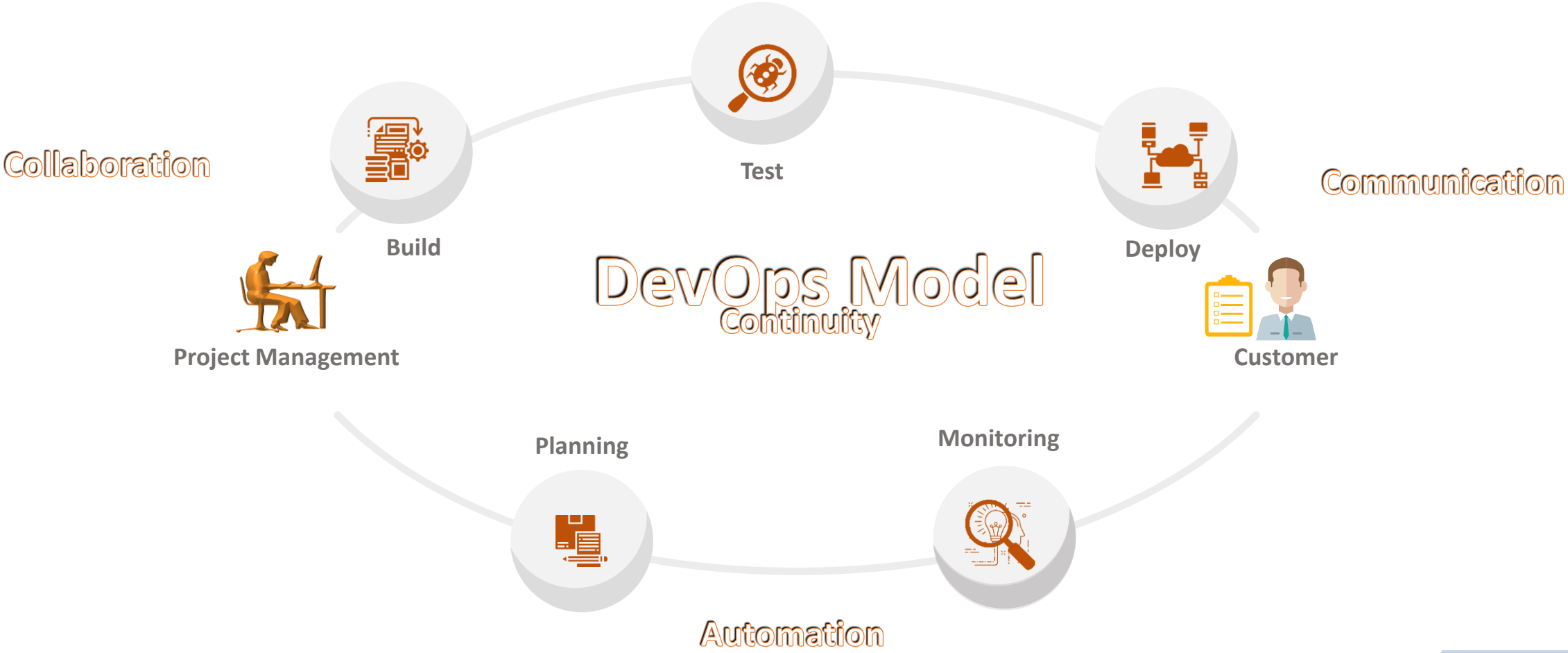
GUI vs CLI vs IAC

- **GUI (Graphical User Interface)**
  - ✓ Best for end user experience
  - ✓ Easy management
  - ✓ Bad for Automation
  - ✓ Not helpful for Administrators

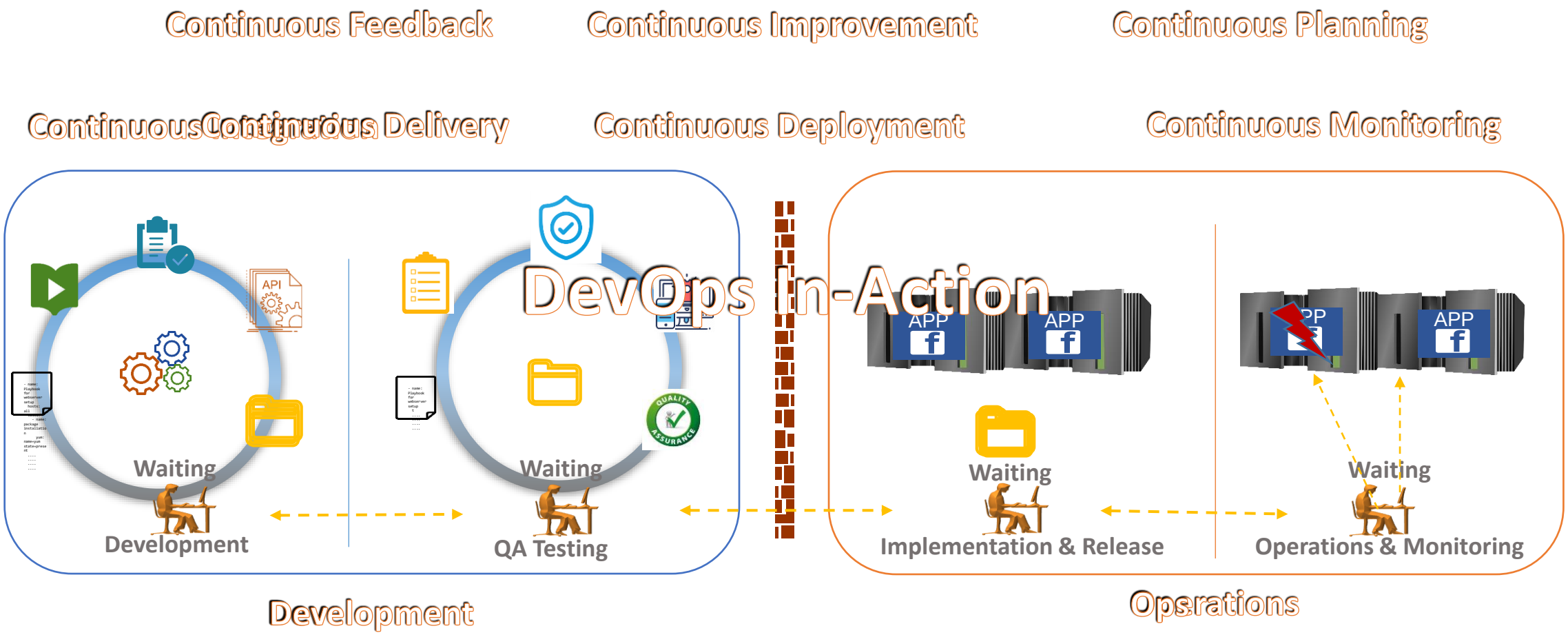
- **CLI (Command Line Interface)**
  - Best for Admin Experience
  - Easy management for Admin level tasks
  - Bad for end user experience
  - Bad for maintaining desired state and consistency

- **IaC (Infrastructure as Code)**
  - Best for Admin Experience
  - Easy management for Admin tasks
  - Easy to understand for end users too
  - Can easily maintain consistency and desired state
  - Infrastructure is written in files, so can be versioned

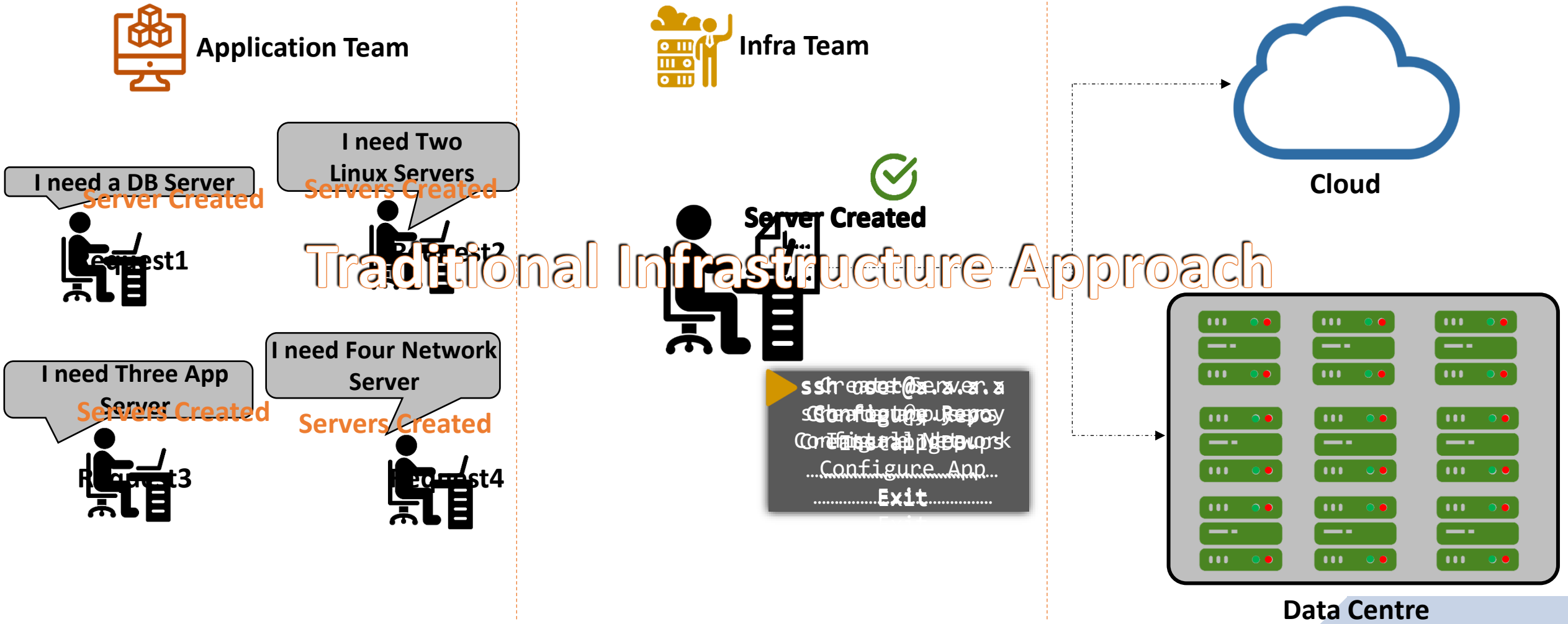
DevOps


Collaboration
Build
Test
Deploy
Communication
Project Management
DevOps Model
Continuity
Customer
Planning
Monitoring
Automation

DevOps in Action



Why DevOps IaC



Traditional Infrastructure Approach

IAC (Infrastructure as Code)

Application Team

I need DB Server

Server Created

Request1

I need Two Linux Servers

Servers Created

Request2

need Three Linux Servers

Servers Created

Request3

I need Four Linux Servers

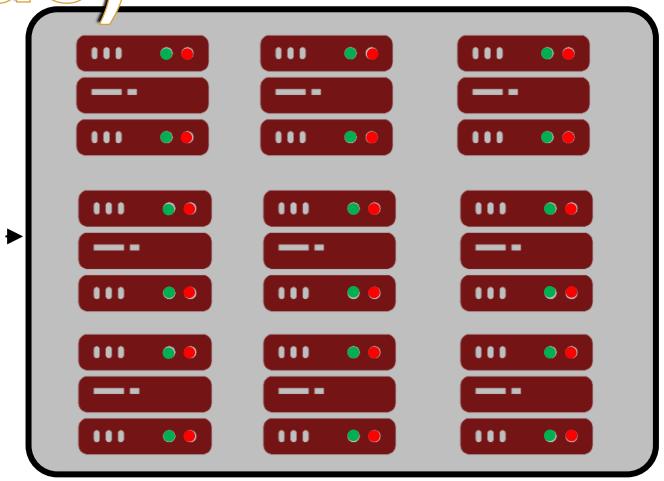
Servers Created

Request4

Infra Team

```
File is: main.tf
provider "aws" {
  region = "us-east-1"
}
resource "aws_instance" “requestfour" {
  count = "4"
  ami = "ami-03d5c68bab01f3496"
  instance_type = "t2.micro"
  tags = {
    Name = "DevOpsInAction"
  }
}
output "myawsserver" {
  value =
"${aws_instance.myawsserver.public_ip}"
}
```

Cloud



Data Centre

IaC is Managing Infrastructure in files rather than manually configuring resources in a user interface

Terraform

Terraform is an easy-to-use IT Orchestration & Automation Software for System Administrators & DevOps Engineers.

- It is the infrastructure as code offering from Hashicorp.
- It is a tool for building, changing, and managing infrastructure in a safe, repeatable way.
- Configuration language called the HashiCorp Configuration Language (HCL) is used to configure the Infrastructure.
- Compatible with almost all major public and private Cloud service provider

Terraform

Infrastructure as code (IAC)

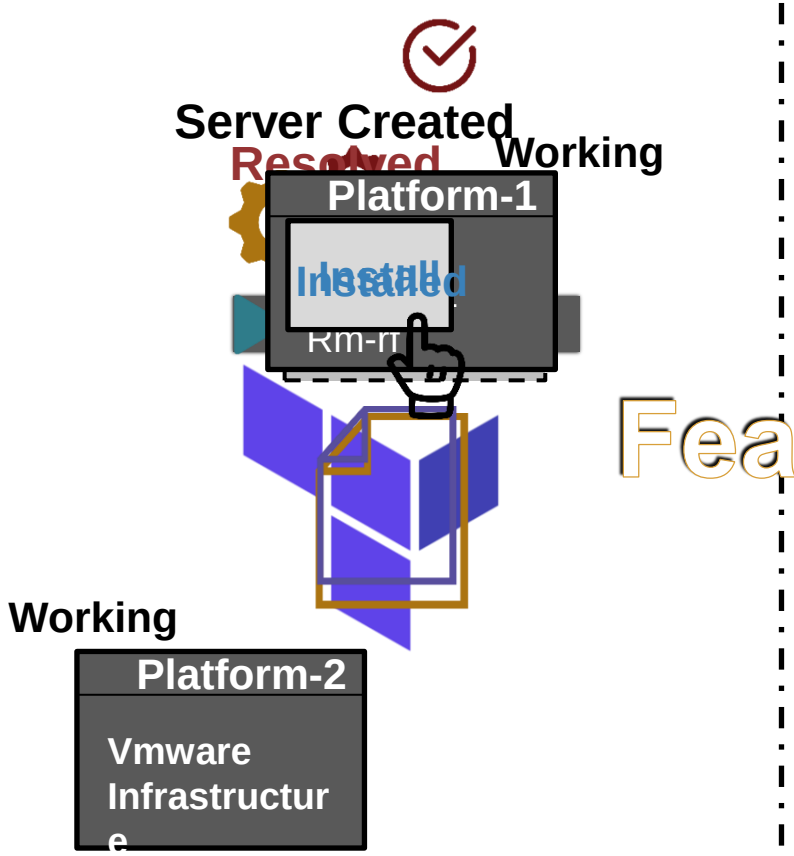
July 2014, HashiCorp

What is Terraform?

Opensource / Enterprise

HCL (Hashicorp Configuration Language)

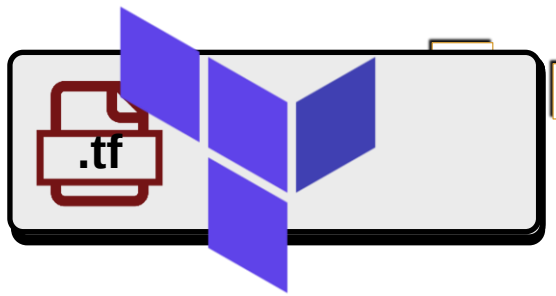
Terraform



Feature & Advantages

- Easy Installation
- Declarative in Nature
- Idempotent
- Intelligent Dependency Resolver
- Platform Agnostic
- Simple and easy to use

Terraform



Terraform Terminologies

Providers

Variables

Resources

Provisioners

DataSources

Outputs

Modules

File extension
.tf

Terraform

main.tf

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

Provider Block

```
resource "aws_instance" "myserver" {
  ami = "ami-030ff268bd7b4e8b5"
  instance_type = "t2.micro"
  tags = {
    Name = "DevOpsInAction"
  }
}
```

Resource Block

```
output "myserveroutputs" {
  description = "Display Servers Public IP"
  value = "${aws_instance.myserver.public_ip}"
}
```

Output Block

Terraform File (Sample Code)

Why Terraform?

- Infrastructure as Code – Write stuff in files, Version it, share it and collaborate with team on same.
- Declarative in Nature
- Automated provisioning
- Clearly mapped Resource Dependencies
- Can plan before you apply
- Consistent
- Compatible with multiple providers and infra can be combined on multiple providers
- 50+ list of official and verified providers
- Approx. 2500+ Modules readily available to work with
- Both Community and Enterprise versions available
- A best fit in DevOps IaC model

Why Terraform?

- **Platform Agnostic** – Manage Heterogeneous Environment
- **Perfect State Management** – Maintains the state and Refreshes the state before each apply action.

  Terraform state is the source of truth. If a change is made or a resource is appended to a configuration, Terraform compares those changes with the state file to determine what changes result in a new resource or resource modifications.
- **Confidence**: Due to easily repeatable operations and a planning phase to allow users to ensure the actions taken by Terraform will not cause disruption in their environment.

Terraform and its Peers

- Chef
- Puppet
- SaltStack
- Ansible
- CloudFormation
- Terraform
- Kubernetes

Terraform and its Peers

Many tools available in Market. Few things to consider, before selecting any tool:

- Configuration Management vs Orchestration
- Mutable Infrastructure vs Immutable Infrastructure
- Procedural vs Declarative

Terraform and its Peers

	Chef	Puppet	Ansible	SaltStack	CloudFormation	Terraform
Code	Open source	Open source	Open source	Open source	Closed source	Open source
Cloud	All	All	All	All	AWS only	All
Type	Config Mgmt	Config Mgmt	Config Mgmt	Config Mgmt	Orchestration	Orchestration
Infrastructure	Mutable	Mutable	Mutable	Mutable	Immutable	Immutable
Language	Procedural	Declarative	Declarative	Declarative	Declarative	Declarative
Architecture	Client/Server	Client/Server	Client-Only	Client/Server	Client-Only	Client-Only

Knowledge Checks

- What is Configuration Management?
- What is Orchestration?
- List a few available configuration Management tools.
- What are the Advantages of Terraform?

Summary: Terraform

Terraform is an easy-to-use IT Orchestration & Automation, Software for System Administrators & DevOps Engineers.

- Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently.
- Terraform can manage existing and popular service providers as well as custom in-house solutions.
- Maintain Desired State
- Highly scalable and can create a complete datacenters in minutes
- Agentless solution
- Declaration in nature than Procedural
- Uses Providers API to provision the Infrastructure
- Terraform creates a dependency graph to determine the correct order of operations.

Installation of Terraform on AWS Env.

Terraform Fundamentals

AWS CLI

AWS CLI

AWS CLI is a command based utility to manage AWS resources

The primary distribution method for the AWS CLI on Linux, Windows, and macOS is pip, a package manager for Python that provides an easy way to install, upgrade, and remove Python packages and their dependencies

http://docs.aws.amazon.com/cli/latest/userguide/installing.html

Requirements

- Python 2 version 2.6.5+ or Python 3 version 3.3+
- Windows, Linux, macOS, or Unix
- Pip package should be present (else install python-pip)

Install AWSCLI: pip install awscli --upgrade --user
For Windows, directly download the Windows installer from CLI webpage

AWS CLI

Lets install an AWSCLI
https://aws.amazon.com/cli

aws --version

aws help

aws ec2 help / aws s3 help / aws <anysubcommand> help

Configure your default keys and region:

root@ip-172-31-28-145:~# aws configure
AWS Access Key ID [None]: ##########
AWS Secret Access Key [None]: ################
Default region name [None]: us-west-2
Default output format [None]:
root@ip-172-31-28-145:~#

LAB 2: AWS CLI

Check the details for all running instances using CLI

- aws ec2 describe-instances | grep -i instanceID

Creation of an AWS Instance using CLI:

- aws ec2 run-instances --image-id ami-05fa00d4c63e32376 --instance-type t2.micro --key-name raman
- aws ec2 stop-instances --instance-ids i-02fedc26aa77154a6
- aws ec2 terminate-instances --instance-ids i-02fedc26aa77154a6
- aws s3 ls
- aws iam list-users

Providers

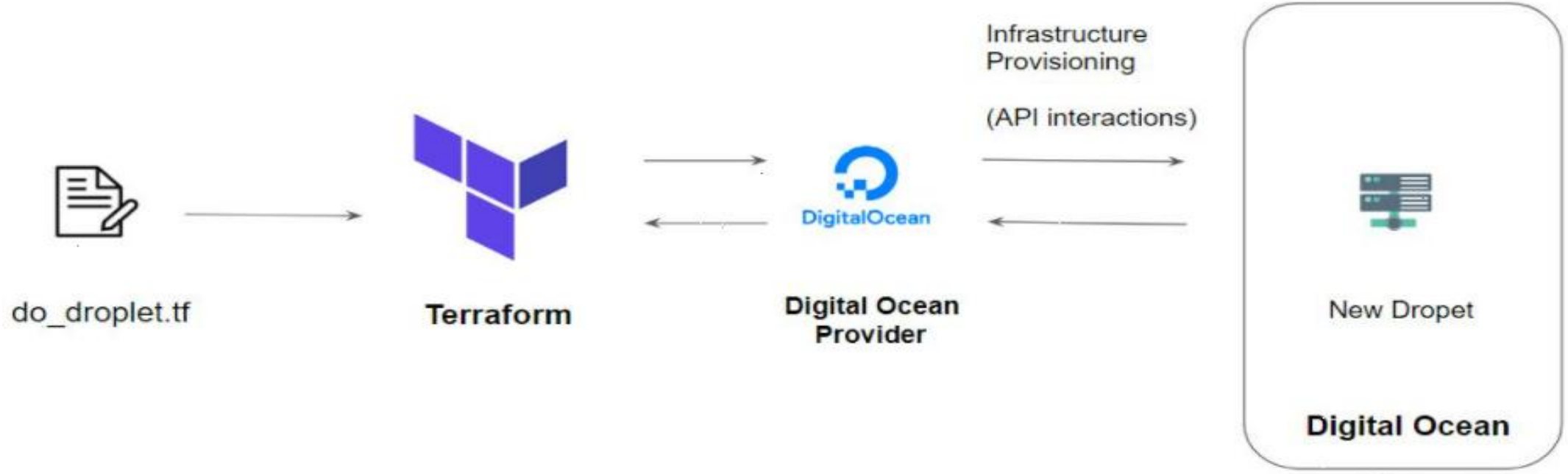
A provider is responsible for understanding API interactions and exposing resources over to a particular cloud service provider. Most providers configure a specific infrastructure platform (either cloud or self-hosted).

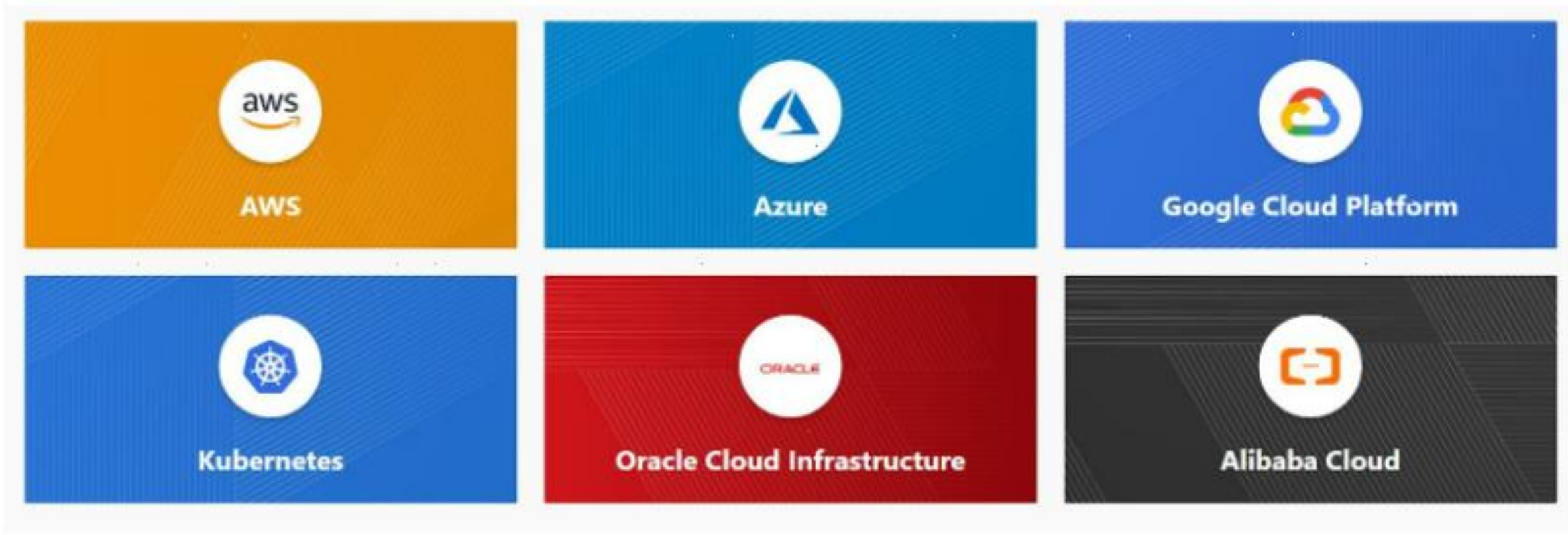
```
provider "aws" {
  region     = "us-east-2"
  access_key = "PUT-YOUR-ACCESS-KEY-HERE"
  secret_key = "PUT-YOUR-SECRET-KEY-HERE"
}
```

A provider is responsible for creating and managing resources.

https://registry.terraform.io/browse/providers

Overview of Provider Architecture :




aws
AWS
Azure
Google Cloud Platform
Kubernetes
ORACLE
Oracle Cloud Infrastructure
Alibaba Cloud

Resources

- Resources are the most important element in the Terraform language. Each resource block describes one or more infrastructure objects, such as virtual networks, compute instances, etc
-
```
resource "aws_instance" "web" {
 ami           = "ami-a1b2c3d4"
 instance_type = "t2.micro"
}
```

A resource block declares a resource of a given type ("aws_instance") with a given local name ("web"). The name is used to refer to this resource from elsewhere in the same Terraform module but has no significance outside that module's scope.

The resource type and name together serve as an identifier for a given resource and so must be unique within a module.

Resource names must start with a letter or underscore, and may contain only letters, digits, underscores, and dashes.

LAB 3: Creating first ec2 instance

..

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

Configuration files

- Whatever you want to achieve(deploy) using terraform will be achieved with configuration files.
- Configuration files ends with .tf extension (tf.json for json version).
- Terraform uses its own configuration language, designed to allow concise descriptions of infrastructure.
- The Terraform language is declarative, describing an intended goal rather than the steps to reach that goal.
- A group of resources can be gathered into a module, which creates a larger unit of configuration.
- As Terraform's configuration language is declarative, the ordering of blocks is generally not significant. Terraform automatically processes resources in the correct order based on relationships defined between them in configuration

Example

- You can write up the terraform code in hashicorp Language – HCL.
- Your configuration file will always endup with .tf extension

```
provider "aws" {
  region     = "us-east-2"
  access_key = "PUT-YOUR-ACCESS-KEY-HERE"
  secret_key = "PUT-YOUR-SECRET-KEY-HERE"
}

resource "aws_instance" "myec2" {
   ami = "ami-082b5a644766e0e6f"
   instance_type = "t2.micro"
}
  tags = {
    Name = "Techlanders-aws-ec2-instance"
  }
}
```