**TERRAFORM :**

* Automate & manage your infrastructure, your platform & services that run on that platform.
* It is open source, declarative ( define what end result you want) & imperative ( define exact steps how)
* Tool for infrastructure provisioning

**INFRASTRUCTURE :**

Application microservices docker containers +database container Build in aws

**PREPARE INFRASTRUCTURE:**

* Private network space
* Ec2 server instances
* Install docker & other tools
* Security

**INFRASTRUCTURE PROVISIONING:**

* PROVISIONING INFRASTRUCTURE
* DEPLOYING APPLICATIONS

**PROVISIONING INFRASTRUCTURE:**

* + - Create VPC ( Virtual Private Cloud )
    - Spin up servers
    - Install Docker
    - Create AWS users & permissions

|  |  |
| --- | --- |
| **ANSIBLE** | **TERRAFORM** |
| For configuring infrastructure | Better for provisioning infrastructure |
| More mature | Relatively new |
| - | More advanced in orchestration |

**MANAGING EXISTING INFRASTRUCTURE:**

* Create infrastructure
* Changes to infrastructure - using Terraform
* Automate the continuous changes to the infrastructure.

**REPLICATING THE INFRASTRUCTURE:**

Once the development environment is tested and ready to deploy, create a production environment which replicates the development environment and keep the development environment as it is to test new features and updates.

**HOW TERRAFORM WORKS:**

* Core provides execution plan by comparing the inputs from state & config file,
* Providers execute the plan by connecting to the platforms to carryout execution.

TERRAFORM ARCHITECTURE

CORE

PROVIDERS

TF-CONFIG

STATE

AWS | AZURE [IAAS]

FASTLY [SAAS]

KUBERNETES [PAAS]

**CORE :**

* + Compares current vs desired state ( config file)
  + Takes the input & plans what needs to be created / updated / destroyed.
    - **TF-CONFIG :** What to create / config
    - **STATE :** Current state of setup

**PROVIDERS:**

* + Through providers we can get access to resources
  + Create stuffs on different levels

EC2

**AWS**

Users

Namespaces

**KUBERNETES**  Services

Deployment

**CONFIGURATION FILE EXAMPLE :**

**# CONFIGURE AWS PROVIDER**

provider “aws”

{

version = “~> 2.0”

region = “us-east-1”

}

**# CREATE A VPC**

resource “aws\_vpc” “example”

{

cidr\_block = “10.0.0.0/16”

}

**# CONFIGURE KUBERNETES PROVIDER**

provider “Kubernetes”

{

config\_context\_auth\_info = ”ops”

config\_context\_cluster = “mycluster”

}

resource “Kubernetes\_namespace” “example”

{

metadata

{

name = “my-first-namespace”

}

}