Lets get started

What are we going to see in this session?

- What is IAAS & IAC ?
- Introduction to Terraform
- Why Terraform?
- Terraform Vs other tools
- Common use cases of Terraform
- How Terraform works?
- Architecture of Terraform
- Terraform Core & Plugins.

What is IAAS & IAC ?

IAAS

- IAAS is Infrastructure as service.
- Where you can get Infrastructure on-demand. These type of services are provided by [AWS, GCP, Azure & Digital Ocean]

IAC

- IAC is Infrastructure as code.
- Through code you can pass set of instructions which is needed to provision and configure your infrastructure.

Terraform

- Using Terraform you can write your own code to setup your infrastructure.
- Hence basically its fit into IAC.

Introduction to Terraform

- Terraform is an open-source Infrastructure as a code software tool developed by Hashicorp.
- It helps users to define and provision datacentre infrastructure using language known as Hashicorp Configuration Language (HCL)

Why Terraform ?

Automation

Helps you to automate your infrastructure rather then doing things manually.

Flexible

Easy to manage multiple cloud providers such as [AWS, AZURE, GCP, Digital Ocean] and private infrastructure such as [VMware]

Open source

Free to deploy and use.

Managing State

- Helps to keep your infrastructure in certain state.
- For example : Your code is written for [2 web instances with 2 volumes & 1 Load balancer] If incase one of your instance is modified or deleted, you can still retain the machine with same configuration when you rerun the code.

Auditable

- By seeing the code, you can understand what is your infrastructure made of.
- Also you can keep your infrastructure change history in version control system like Git.

Easy to use

- Easy to install and use.
- Codes written are human readable.

Terraform Vs Other orchestration tools

Chef, Puppet & Ansible	Cloud Formation, Heat	Python Boto	Custom solutions
Configuration management tools are used to install and manage software on the machine that already exist	Cloud formation & heat are native solutions provided by there own cloud providers which helps to code the infrastructure into a configuration file.	Libraries like Boto are used to provide native access to cloud providers and services by using their API's.	Many organizations starts to setup there own infrastructure solutions either through simple scripts or web-based interfaces.
Terraform is predominantly used to automate the infrastructure itself.	Terraform does the same, you can code the infrastructure setup into configuration file and implement.	Terraform is very flexible and uses plugin based model to support providers.	Difficulty here is when infrastructure grows it might become tedious to mange your custom solutions.
Terraform is not an Configuration management tool, and it allows and existing tools to focus on there strengths.	Major difference here is Terraform is cloud-agnostic. It can work with multiple public & private providers.	Major difference here is libraries might give only low-level programmatic access through APIs but Terraform gives high-level access.	Terraform is designed to tackle these kind of challenges. It provides simple solution to mange your Infrastructure.

Common use cases of Terraform

Multi-Tier Applications:
The very common patter is 2-tire architecture [Pool of Web servers along with DB Servers] Additional tires may get added based on requirement. Terraform is very useful to handle in setting up this kind of Infrastructures.

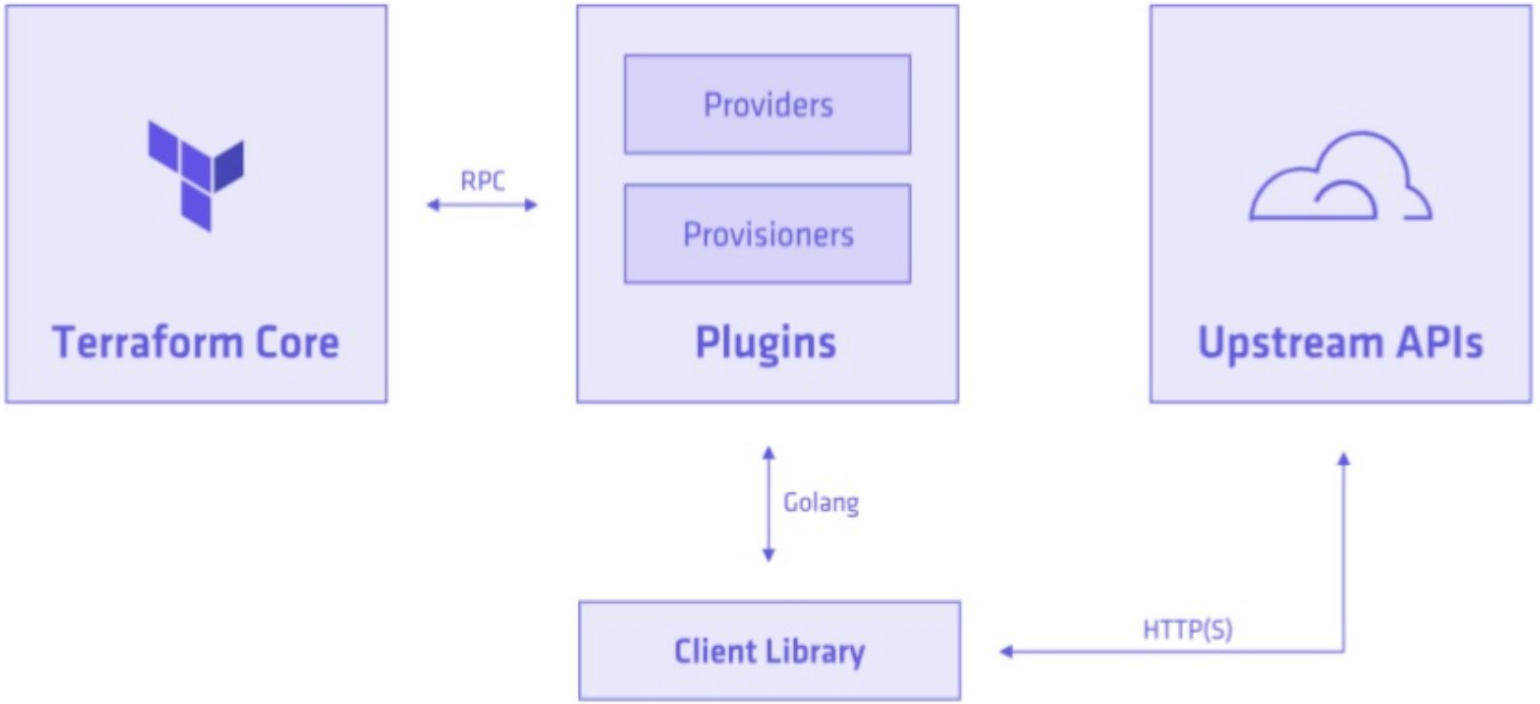
Disposable Environment :
Its very easy to setup an test environment when and where required and its not going to be complex anymore.

Multi-Cloud Deployment :
Its often attractive to spread infrastructure across multiple clouds to increase FT. This is vey much possible through Terraforms.

How Terraform Works?

- Terraform is built on a plugin-based architecture.
- This helps developers to extend Terraform by writing new plugins or compiling modified versions of existing plugins.
- Terraform is logically split into two main parts [Terraform Core & Terraform Plugins]
- Terraform Core :
  - Terraform Core uses remote procedure calls (RPC) to communicate with Terraform Plugins.
  - Terraform Core is written in Go programming language. The code is open source and you can find it Github
- Terraform Plugins :
  - Terraform Plugins are also written in Go programming language.
  - Plugins are used in configuration files.
  - Plugins consists of 2 things [Providers & Provisioners]

Architecture of Terraform



Terraform Core & Plugins

- The primary responsibility of Terraform core is
  - Reading & Interpreting configuration files and modules.
  - Resource state management.
  - Plan execution.
  - Communication with plugins over RPC.
- The primary responsibility of Terraform Plugins are
  - Ensure authentication with Infrastructure providers.
  - Taking care of RPC calls from Terraform core.

Providers & Provisioner we will see it more in coming sessions..

End of this topic !

Any questions?