**Terraform:-Infrastructure as a code**

* Terraform is a first infrastructure as a code tool with complete set of features.
* HCL (HashiCrop Control Language) is used to write terraform scripts.
* You can use terraform to provision resource on AWS, Azure, GCP. It is a cloud neutral.
* Some concepts related to infrastructure as a code easily implementable.
* It Automates the scripts and migrates scripts that AWS, cloud formation template(CFT) to Azure resource management to Heat Templates.
* Terraform convert the script to any service. So, we don’t need to learn so many Tools.

Using API(Application Programming Interface) you can programmically access to any application. Google, Git hub, Azure etc.

* Basic Commands in Terraform:-
* **Terraform init -** It initialize a working directory. First command to run after writing a code.
* **Terraform plan -** Create an execution plan, to check terraform state is upto date.
* **Terraform apply -** Build or changes infrastructure.
* **Terraform Destroy -** Terminate resources managed by terraform project.
* **Terraform state file:-** terraform.tfstate file is used to tracking the infrastructure. You can not store the state file in local machine or remotely in git. Because, it has sensitive information. Do not manipulate or update state file.

**Ideal Terraform setup (Remote back end)**

* Develop a script on Linux or VS Code.
* Put Terraform configure in Git hub (Version Control).
* Terraform state file should goes into remote backend. Remote backend means remote storage service. Ex:- Amazon S3 bucket, Azure storage.
* Integrated them with proper locking solutions. Ex:- Dynamo DB.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Terraform**  **User** | **Linux or**  **VS Code** | **Git**  **(Version Control** | **AWS S3**  **(Remote backend or remote storage)** | **Dynamo DB (Locking solutions)** |

**Modules :-** Modules are used to write reuse configuration that will be implemented in multiple locations of a terraform configuration.

Terraform module is a set of terraform configuration files in a single directory.

**Limitations in Terraform:-**

* state file is a single source of truth.
* Manual changes to the cloud provider can not be identified and auto-corrected.
* Git opps is not a friendly tool, don’t play well with flux or Argo CD.
* Can become very complex and difficult to manage.
* Trying to position to a configuration management tool as well.