Advanced DevOps Lab
Experiment 5

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	Advanced DevOps Lab

Aim: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.

Theory:

Terraform is an open-source “Infrastructure as Code” tool, created by HashiCorp.

A declarative coding tool, Terraform enables developers to use a high-level configuration language called HCL (HashiCorp Configuration Language) to describe the desired “end-state” cloud or on-premises infrastructure for running an application. It then generates a plan for reaching that end-state and executes the plan to provide the infrastructure.

Because Terraform uses simple syntax, can provision infrastructure across multiple cloud and on-premises data centres, and can safely and efficiently re-provision infrastructure in response to configuration changes, it is currently one of the most popular infrastructure automation tools available. If your organization plans to deploy a hybrid cloud or multi-cloud environment, you’ll likely want or need to get to know Terraform.

Why Terraform?

There are a few key reasons developers choose to use Terraform over other Infrastructure as Code tools:

Open source: Terraform is backed by large communities of contributors who build plugins to the platform. Regardless of which cloud provider you use, it’s easy to find plugins, extensions, and professional support. This also means Terraform evolves quickly, with new benefits and improvements added consistently.

Platform agnostic: Meaning you can use it with any cloud services provider. Most other IaC tools are designed to work with a single cloud provider.

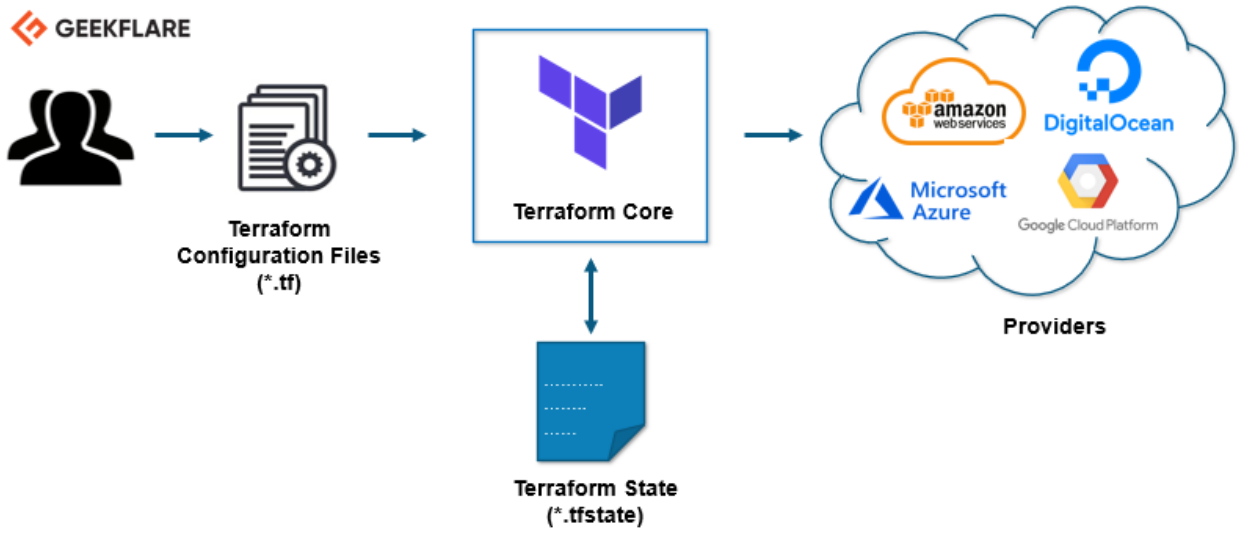
Immutable infrastructure: Most Infrastructure as Code tools create mutable infrastructure, meaning the infrastructure can change to accommodate changes such as a middleware upgrade or new storage server. The danger with mutable infrastructure is configuration drift—as the changes pile up, the actual provisioning of different servers or other infrastructure elements ‘drifts’ further from the original configuration, making bugs or performance issues difficult to diagnose and correct. Terraform provisions immutable infrastructure, which means that with each change to the environment, the current configuration is replaced with a new one that accounts for the change, and the infrastructure is reprovisioned. Even better, previous configurations can be retained as versions to enable rollbacks if necessary or desired.

How does Terraform work?

Terraform allows users to define their entire infrastructure simply by using configuration files and version control. When a command is given to deploy and run a server, database or load balancer, Terraform parses the code and translates it into an application programming interface (API) call to the resource provider. Because Terraform is open source, developers are always able to extend the tool's usefulness by writing new plugins or compiling different versions of existing plugins.

Terraform has two important components: Terraform Core and Terraform Plugins.

Terraform Core oversees the reading and interpolation of resource plan executions, resource graphs, state management features and configuration files. The core is composed of compiled binaries written in the Go programming language. Each compiled binary acts as a command-line interface (CLI) for communicating with plugins through remote procedure calls (RPC).



Terraform Plugins are responsible for defining resources for specific services. This includes authenticating infrastructure providers and initializing the libraries used to make API calls. Terraform Plugins are written in Go as executable binaries that can either be used as a specific service or as a provisioner. (Provisioner plugins are used to execute commands for a designated resource.)

Steps:

1. Find HashiCorp Authentication for their packages on their official website here.

   ```
   curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
   ```

   ```
   zenon@yeetus:~$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
   [sudo] password for zenon:
   OK
   ```

   Register the key to your system.

2. Add this repository to your system.

   ```
   sudo apt-add-repository "deb [arch=$(dpkg --print-architecture)]
   https://apt.releases.hashicorp.com $(lsb_release -cs) main"
   ```

   ```
   zenon@yeetus:~$ sudo apt-add-repository "deb [arch=$(dpkg --print-architecture)] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
   Get:1 https://apt.releases.hashicorp.com focal InRelease [4,419 B]
   Hit:2 http://in.archive.ubuntu.com/ubuntu focal InRelease
   Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
   Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
   Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
   Get:6 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
   Get:7 https://apt.releases.hashicorp.com focal/main amd64 Packages [33.2 kB]
   Get:8 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
   Get:10 http://in.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [545 kB]
   Get:11 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1,258 kB]
   Get:12 http://in.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [265 kB]
   Get:13 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [283 kB]
   Get:14 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [14.4 kB]
   Get:15 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [865 kB]
   Get:16 http://in.archive.ubuntu.com/ubuntu focal-updates/universe i386 Packages [641 kB]
   Get:17 http://in.archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [184 kB]
   Get:18 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 Metadata [361 kB]
   ```

3. Install Terraform

   ```
   sudo apt install terraform
   ```

   ```
   zenon@yeetus:~$ sudo apt install terraform
   Reading package lists... Done
   Building dependency tree
   Reading state information... Done
   The following NEW packages will be installed:
     terraform
   0 upgraded, 1 newly installed, 0 to remove and 53 not upgraded.
   Need to get 32.7 MB of archives.
   ```

4. Verify your installation.

   ```
   zenon@yeetus:~$ terraform --version
   Terraform v1.0.8
   on linux_amd64
   ```

Conclusion: In this experiment, we learned how to install Terraform on our Linux machines.