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

# **Theory :**

**Terraform** is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It allows users to define and provision infrastructure using a high-level configuration language known as HashiCorp Configuration Language (HCL) or JSON. Terraform supports a wide range of cloud providers, such as AWS, Azure, Google Cloud, and

on-premises solutions, enabling users to manage infrastructure across multiple environments consistently.

# **Core Concepts and Terminologies**

1. **Providers:**

Providers are plugins that allow Terraform to interact with various APIs of cloud providers, SaaS providers, and other services. Each provider requires configuration and manages resources for that specific service.

# 2. **Resources:**

Resources are the most fundamental elements in Terraform. They represent components of your infrastructure, such as virtual machines, databases, networks, and more.

# 3. **Modules:**

Modules are containers for multiple resources that are used together. A module can call other modules, creating a hierarchical structure. This makes it easier to organize and reuse code.

# 4. **State:**

Terraform maintains a state file that keeps track of the infrastructure managed by Terraform. The state file is crucial as it provides a mapping between the real-world resources and the configuration defined in Terraform.

# 5. **Variables:**

Variables in Terraform are used to make configurations dynamic and reusable. They can be defined in the configuration files and assigned values at runtime.

# 6. **Outputs:**

Outputs are used to extract information from the Terraform-managed infrastructure and display it after the execution of a Terraform plan or apply.

# **Terraform Lifecycle**

1. **Write:**

Write the configuration file (typically with .tf extension) using HCL to describe the desired infrastructure.

# 2. **Initialize (terraform init):**

Initialize the working directory containing the configuration files. This command downloads the necessary provider plugins and sets up the environment.

# 3. **Plan (terraform plan):**

Terraform creates an execution plan based on the configuration files. It compares the current state with the desired state and shows the changes that will be made.

# 4. **Apply (terraform apply):**

Apply the changes required to reach the desired state of the configuration. Terraform will prompt for confirmation before making any changes.

# 5. **Destroy (terraform destroy):**

Destroy the infrastructure managed by Terraform. This command is used to remove all resources defined in the configuration files.

Implementation:-









