Since 1962

Vivekanand Education Society's

Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra

Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology A.Y. 2024-25

Advance DevOps Lab

Experiment 05

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

Roll No.	53
Name	Aryan Deepak Saraf
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements. LO3: To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure.
Grade:	

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 that allows users to define and manage infrastructure using a high-level configuration language called HashiCorp Configuration Language (HCL). By treating infrastructure as code, Terraform enables the automation of provisioning, modification, and management of resources across multiple cloud providers, ensuring consistency and repeatability.

Core Concepts and Terminologies

1. Infrastructure as Code (IaC): Terraform allows infrastructure to be defined, versioned, and managed as code, making it easier to automate the provisioning of resources like virtual machines, networks, and storage.

2. Terraform Configuration: Configurations are written in HCL or JSON, saved in `.tf` files, and describe the desired state of your infrastructure, including the resources to be created and their properties.

3. Providers: Providers are plugins that enable Terraform to interact with various cloud platforms and services. They manage specific resources via APIs.

4. Resources: Resources represent infrastructure components such as compute instances and databases. Terraform manages their lifecycle, ensuring they are created, updated, or destroyed based on the configuration.

5. State: Terraform's state file records the current state of the infrastructure, ensuring that Terraform accurately tracks resource changes and maintains the desired state.

6. Execution Plan: The execution plan (`terraform plan`) outlines the changes Terraform will make to achieve the desired state, allowing users to review the plan before applying it.

7. Apply: The `terraform apply` command executes the changes specified in the execution plan, updating the infrastructure to match the desired state.

8. Modules: Modules are reusable blocks of configuration that help organize and manage resources, promoting code reuse and consistency.

Terraform Lifecycle

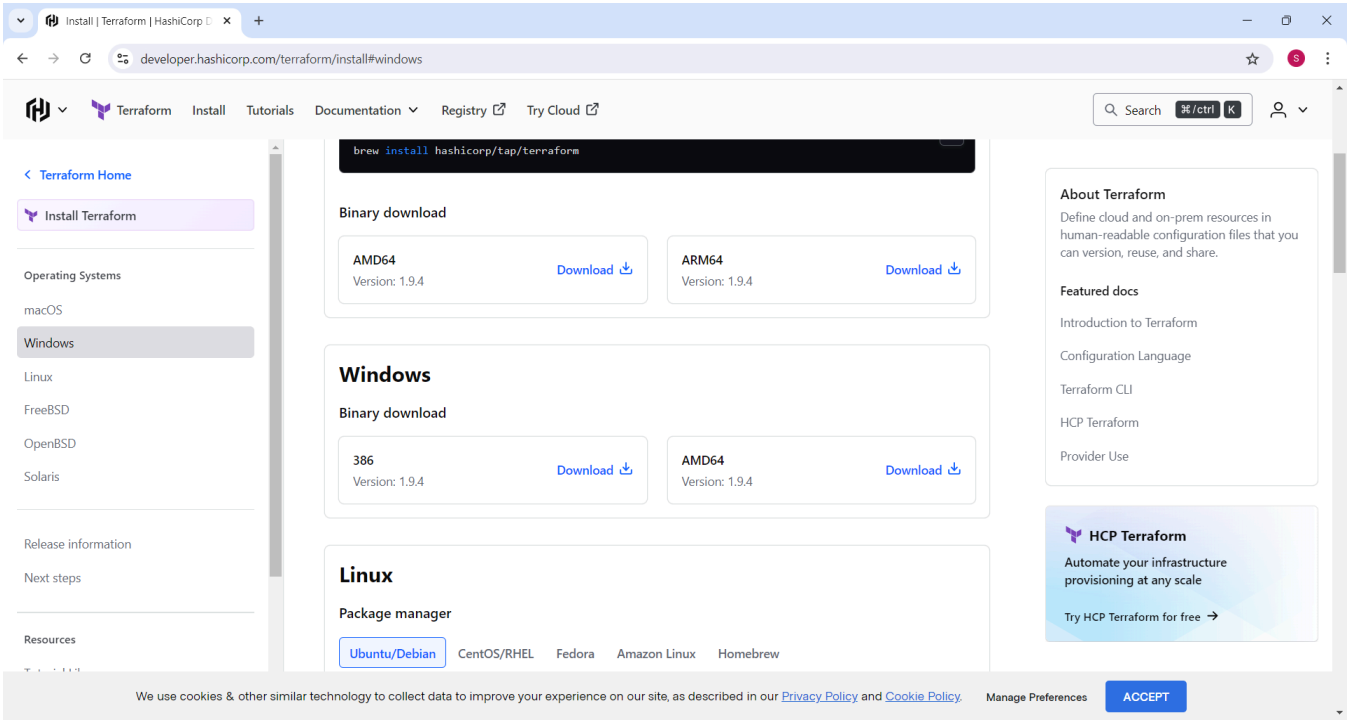
1. Initialization (`terraform init`): Initializes the working directory, downloads provider plugins, and prepares the environment for further commands.

2. Planning (`terraform plan`): Generates an execution plan by comparing the current state with the desired state defined in the configuration, detailing the actions Terraform will take.

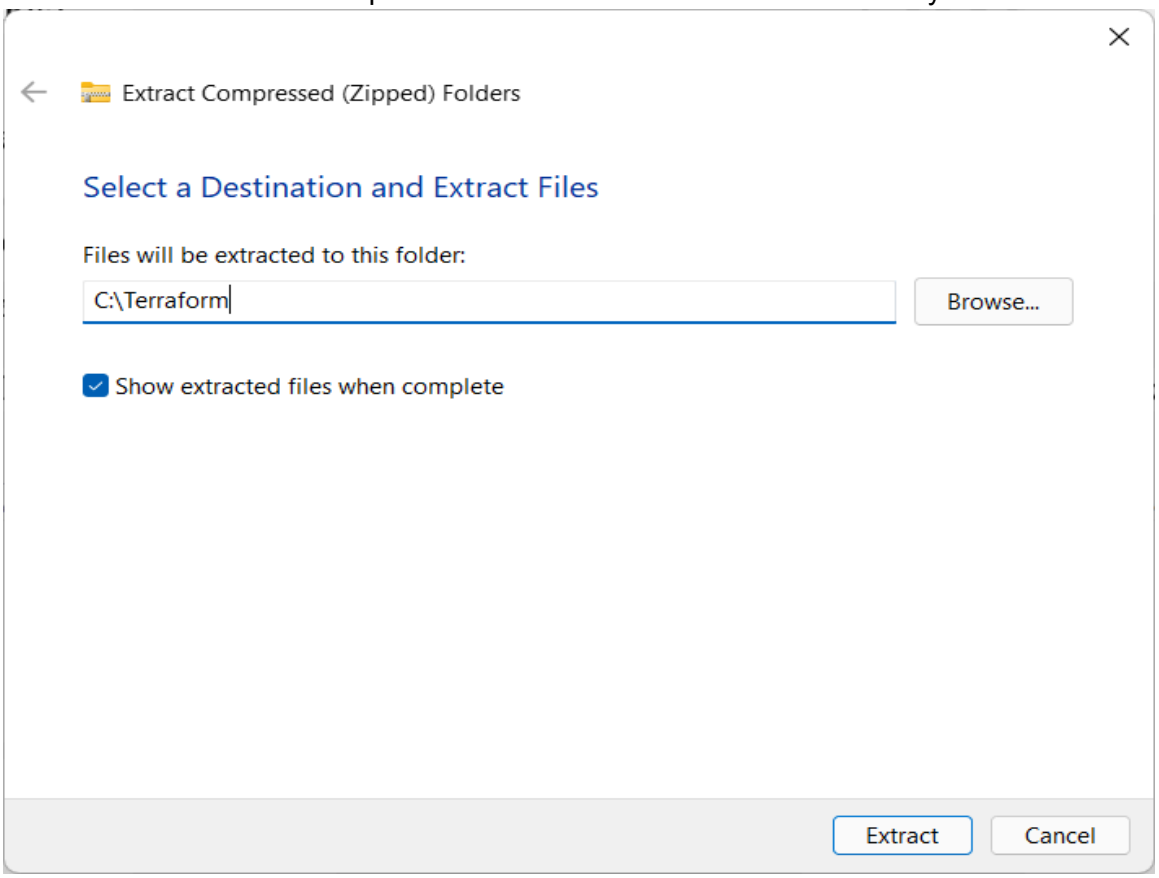
3. Applying (`terraform apply`): Executes the changes specified in the execution plan, creating or modifying resources to match the desired state.

Destroying (`terraform destroy`): Deletes all resources managed by the current configuration, useful for completely tearing down an environment.

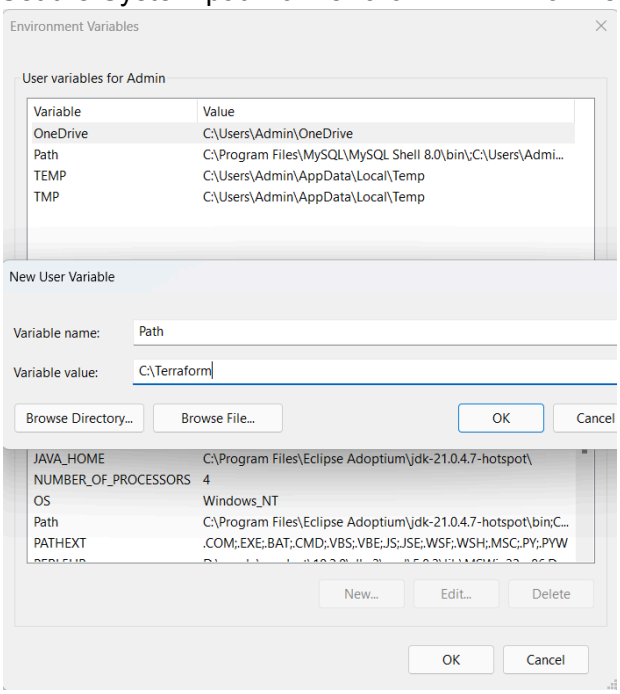
Download terraform



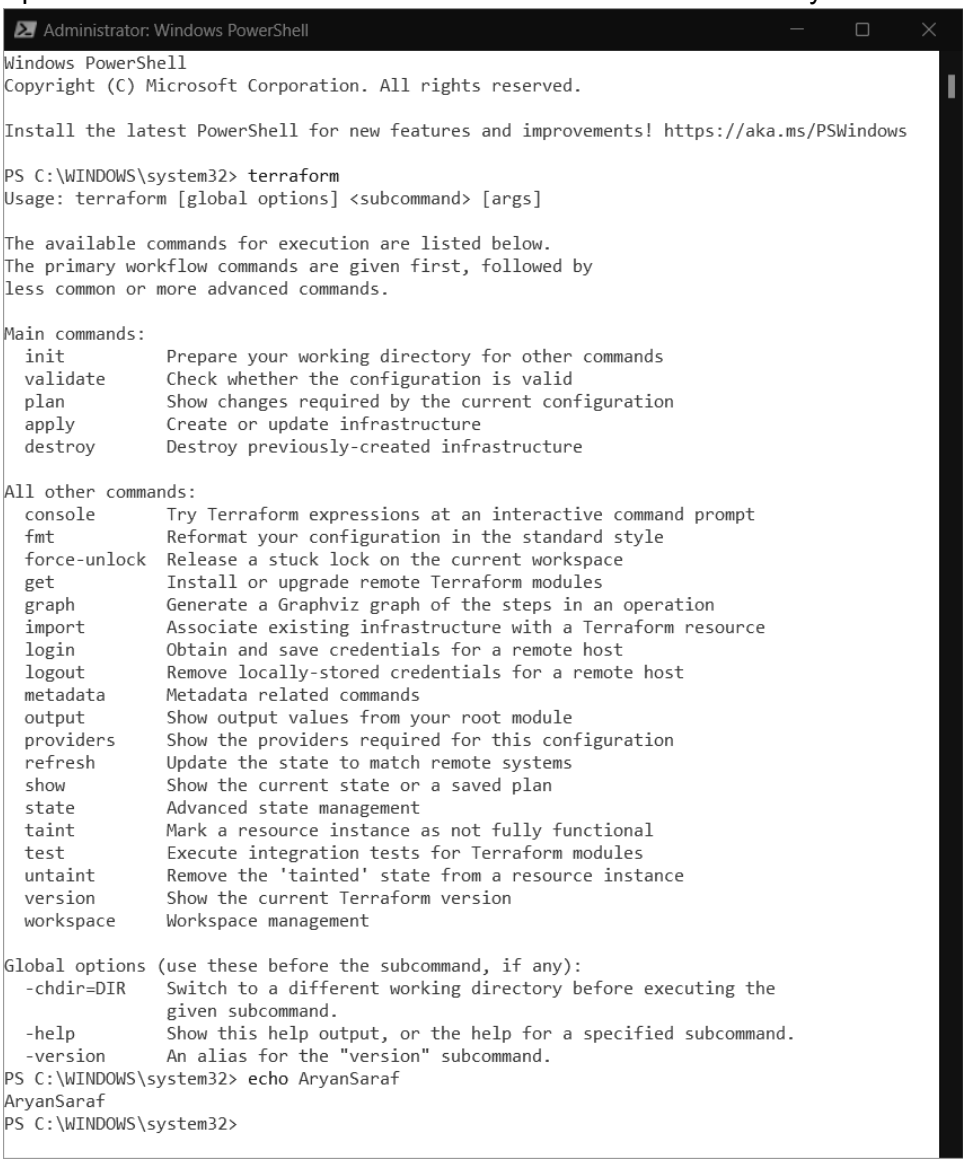
Extract the downloaded setup file Terraform.exe in C:\Terraform directory.



Set the System path for Terraform in Environment Variables



Open PowerShell with Admin Access and check its functionality.



CONCLUSION :

Terraform is a powerful tool for managing infrastructure as code, offering automation, consistency, and flexibility across various platforms. Understanding its core concepts and lifecycle is essential for effective infrastructure management.