Advanced DevOps Lab
Experiment 5

Name: Yash Rahate
Class: D15B
Roll no.: 48

Aim:

To understand terraform lifecycle,core concepts/terminologies andinstall it on a Linux Machine and Windows.

Theory:

Terraform Overview:

Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp. It allows you to define, provision, and manage infrastructure across multiple cloud providers (like AWS, Azure, GCP) using declarative configuration files. This approach enables automated and reproducible deployments, scaling, and management of infrastructure.

Core Concepts and Terminologies:

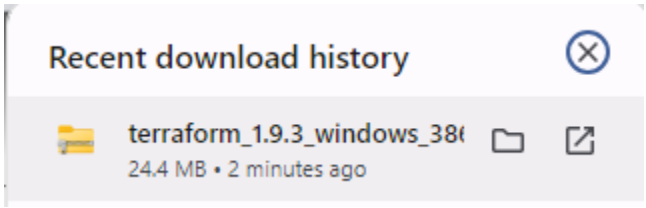
1. **Providers**:
    - Providers are responsible for managing the lifecycle of infrastructure resources. Each provider (like AWS, Azure, GCP) exposes its resources that can be managed using Terraform.
    - Example: provider "aws" { ... } defines the AWS provider.
2. **Resources**:
    - Resources represent the infrastructure components (such as servers, databases, or networking components) you want to create or manage.
    - Example: resource "aws_instance" { ... } defines an AWS EC2 instance.
3. **State**:
    - Terraform maintains a state file (terraform.tfstate) that tracks the current state of infrastructure. This state is used to determine changes between the existing infrastructure and the desired state defined in the configuration.
4. **Modules**:

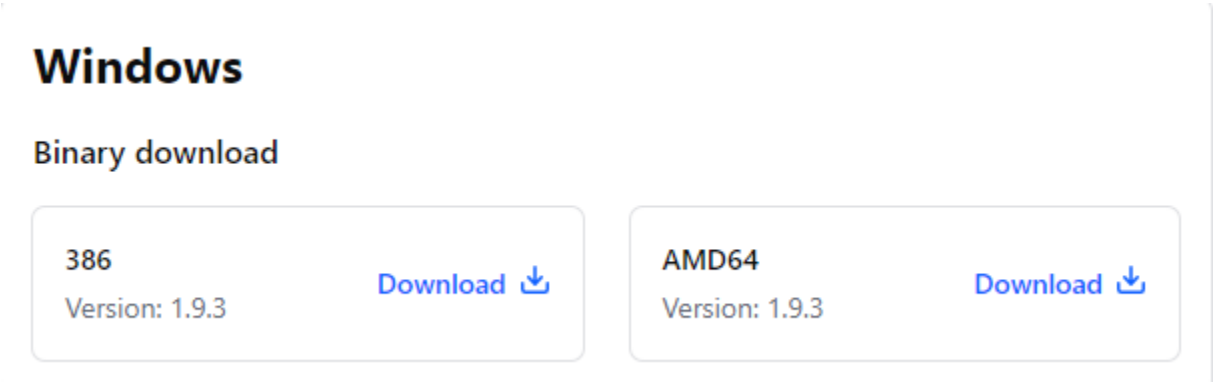
- Modules are self-contained packages of Terraform configurations that can be reused and shared. A module can consist of multiple resources.

5. **Variables**:
   - Variables allow you to parameterize the Terraform configurations, making them flexible and reusable. They can be provided through the command line, files, or environment variables.
6. **Outputs**:
   - Outputs allow you to extract and display information about your infrastructure after applying configurations. They can also be passed between modules.
7. **Terraform CLI Commands**:
   - terraform init: Initializes the working directory by downloading providers and configuring the backend.
   - terraform plan: Shows the execution plan, listing the changes that will be made.
   - terraform apply: Applies the desired state and provisions the infrastructure.
   - terraform destroy: Deletes all resources defined in the configuration.
8. **Lifecycle of Terraform**:
   - **Write**: Define infrastructure as code in configuration files.
   - **Initialize**: Initialize Terraform by downloading providers and setting up the workspace.
   - **Plan**: Preview the changes Terraform will make to the infrastructure.
   - **Apply**: Apply the changes to provision or modify the infrastructure.
   - **Destroy**: Clean up resources created by Terraform.

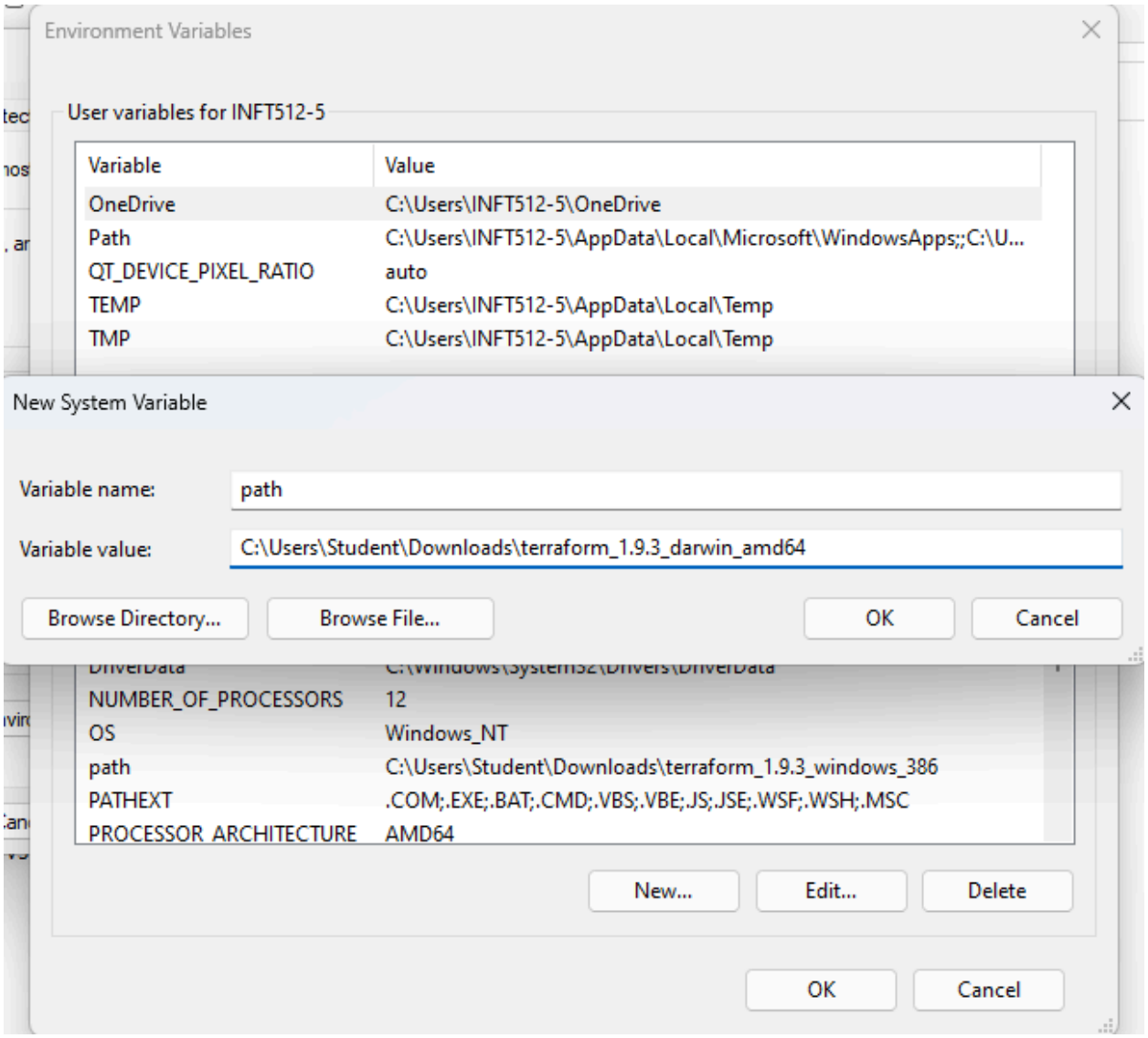
Installing Terraform on a Windows Machine:

1. Downloading Terraform from the Hashicorp website.

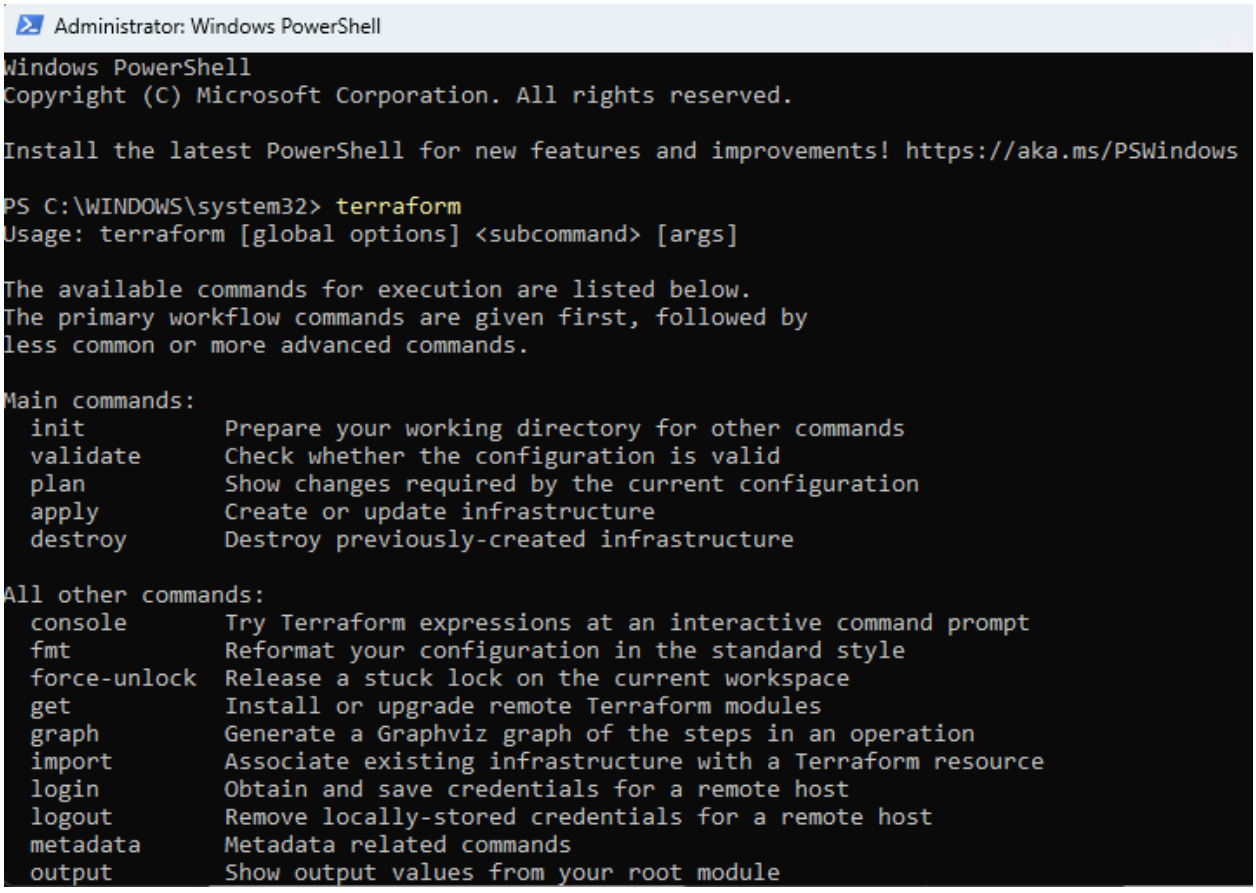




2. Setting path of the terraform in the Environment variables from settings.



3. Working of terraform on Windows Powershell



Conclusion:

Terraform simplifies the process of infrastructure management by providing a declarative, human-readable configuration language. By understanding core concepts like providers, resources, and state, users can define and manage complex infrastructures across multiple platforms. Installing Terraform on a Windows machine is straightforward and involves downloading the tool and setting the appropriate system path. Once installed, you can efficiently manage cloud infrastructure in a consistent and automated way, which is ideal for scaling applications and environments.