# LAB ASSIGNMENT 4

**AIM**: To understand terraform lifecycle, core concepts/terminologies and install it.

**LO3:** To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure.

**THEORY:**

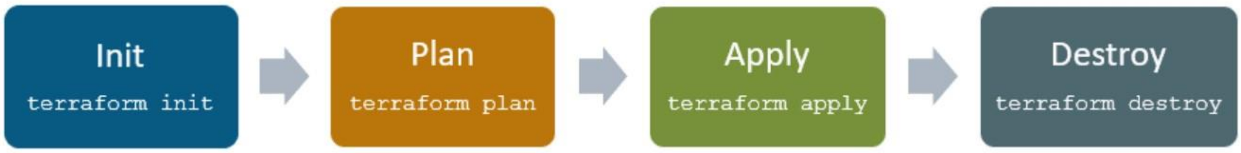
Terraform is one of the most popular Infrastructure-as-code (IaC) tool, used by DevOps teams to automate infrastructure tasks. It is used to automate the provisioning of your cloud resources. Terraform is an open-source, cloud-agnostic provisioning tool developed by HashiCorp and written in GO language.

Benefits of Terraform:

- Does orchestration, not just configuration management.
- Supports multiple providers such as AWS, Azure, Oracle, GCP, and many more.
- Provide immutable infrastructure where configuration changes smoothly.
- Uses easy to understand language, HCL (HashiCorp configuration language).
- Easily portable to any other provider.

## TERRAFORM LIFECYCLE

Terraform lifecycle consists of – **init**, **plan**, **apply**, and **destroy**.



1. **Terraform init** initializes the (local) Terraform environment. Usually executed only once per session.
2. **Terraform plan** compares the Terraform state with the as-is state in the cloud, build and display an execution plan. This does not change the deployment (read-only).
3. **Terraform apply** executes the plan. This potentially changes the deployment.
4. **Terraform destroy** deletes all resources that are governed by this specific terraform environment.

## TERRAFORM CORE CONCEPTS/TERMINOLOGIES

1. Variables: Terraform has input and output variables, it is a key-value pair. Input variables are used as parameters to input values at run time to customize our deployments. Output variables are return values of a terraform module that can be used by other configurations.

2. Provider: Terraform users provision their infrastructure on the major cloud providers such as AWS, Azure, OCI, and others. A provider is a plugin that interacts with the various APIs required to create, update, and delete various resources.

3. Module: Any set of Terraform configuration files in a folder is a module. Every Terraform configuration has at least one module, known as its root module.

4. State: Terraform records information about what infrastructure is created in a Terraform state file. With the state file, Terraform is able to find the resources it created previously, supposed to manage and update them accordingly.

5. Resources: Cloud Providers provides various services in their offerings, they are referenced as Resources in Terraform. Terraform resources can be anything from compute instances, virtual networks to higher-level components such as DNS records. Each resource has its own attributes to define that resource.

6. Data Source: Data source performs a read-only operation. It allows data to be fetched or computed from resources/entities that are not defined or managed by Terraform or the current Terraform configuration.

7. Plan: It is one of the stages in the Terraform lifecycle where it determines what needs to be created, updated, or destroyed to move from the real/current state of the infrastructure to the desired state.

8. Apply: It is one of the stages in the Terraform lifecycle where it applies the changes real/current state of the infrastructure in order to achieve the desired state.

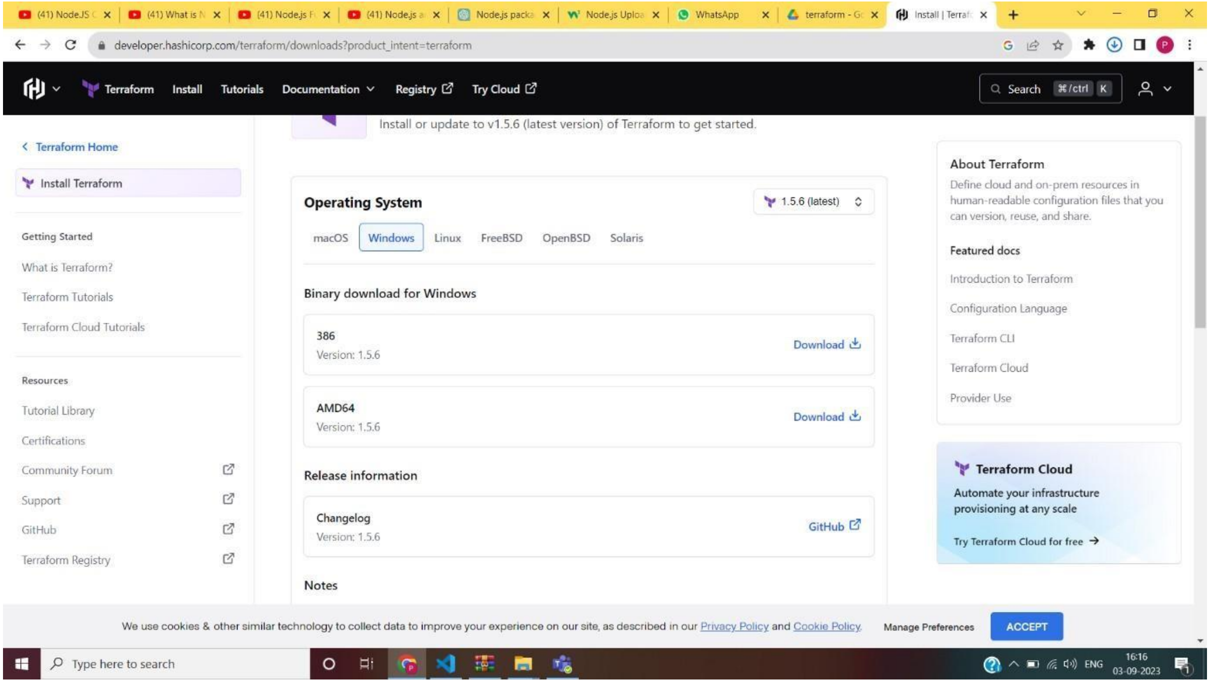
INSTALLATION:

1) Download Terraform

To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website

website: https://www.terraform.io/downloads.html

Select the Operating System Windows followed by either 32bit or 64 bit based on your OS type.



developer.hashicorp.com/terraform/downloads?product_intent=terraform

Install or update to v1.5.6 (latest version) of Terraform to get started.

**Operating System** — 1.5.6 (latest)

macOS | **Windows** | Linux | FreeBSD | OpenBSD | Solaris

Binary download for Windows

| | |
|---|---|
| 386<br>Version: 1.5.6 | Download |
| AMD64<br>Version: 1.5.6 | Download |

Release information

| | |
|---|---|
| Changelog<br>Version: 1.5.6 | GitHub |

Notes

About Terraform

Define cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.

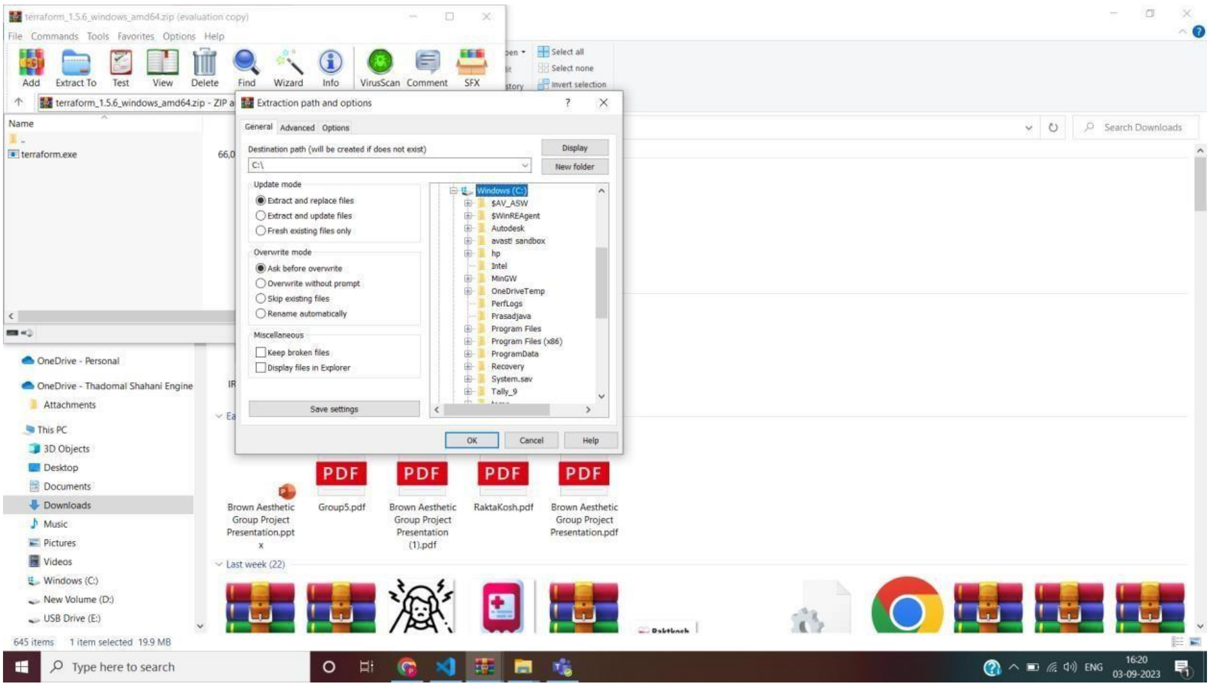
Terraform Cloud

Automate your infrastructure provisioning at any scale

Try Terraform Cloud for free →

We use cookies & other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie Policy.

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



terraform_1.5.6_windows_amd64.zip (evaluation copy)

Extraction path and options

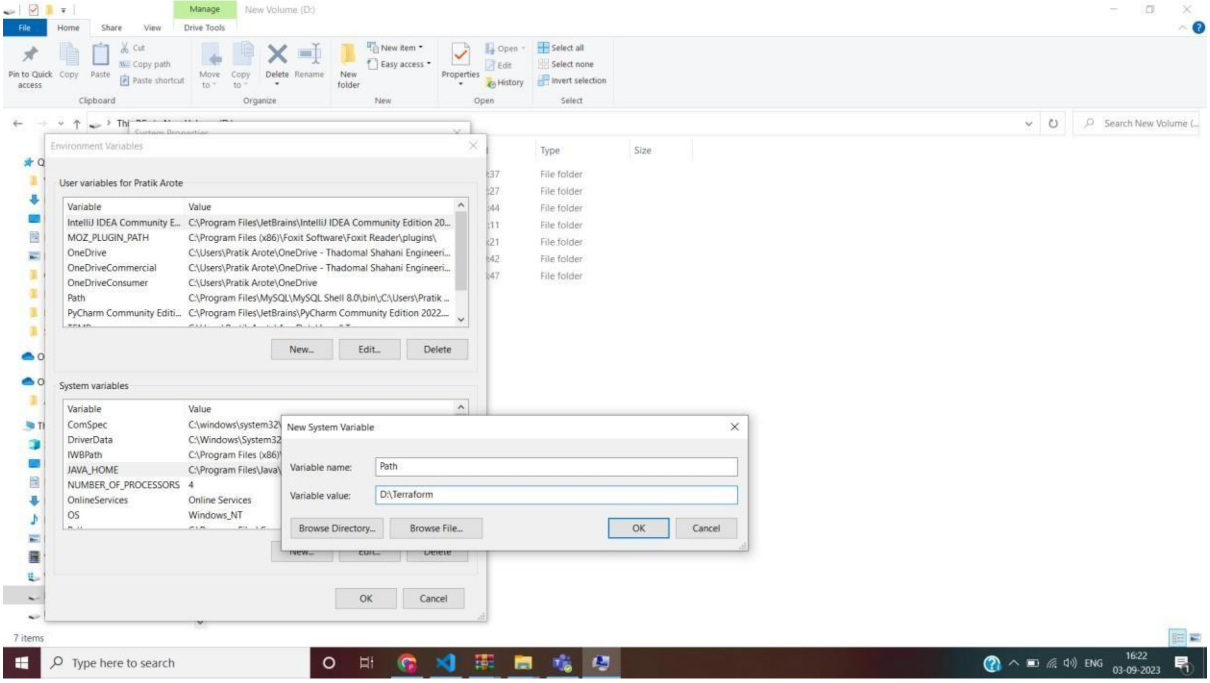
Destination path (will be created if does not exist): C:\

Update mode
● Extract and replace files
○ Extract and update files
○ Fresh existing files only

Overwrite mode
● Ask before overwrite
○ Overwrite without prompt
○ Skip existing files
○ Rename automatically

Miscellaneous
□ Keep broken files
□ Display files in Explorer

3. Set the System path for Terraform in Environment Variables.



User variables for Pratik Arote

| Variable | Value |
|---|---|
| IntelliJ IDEA Community E... | C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 20... |
| MOZ_PLUGIN_PATH | C:\Program Files (x86)\Foxit Software\Foxit Reader\plugins\ |
| OneDrive | C:\Users\Pratik Arote\OneDrive - Thadomal Shahani Engineeri... |
| OneDriveCommercial | C:\Users\Pratik Arote\OneDrive - Thadomal Shahani Engineeri... |
| OneDriveConsumer | C:\Users\Pratik Arote\OneDrive |
| Path | C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\Pratik ... |
| PyCharm Community Editi... | C:\Program Files\JetBrains\PyCharm Community Edition 2022.... |

System variables

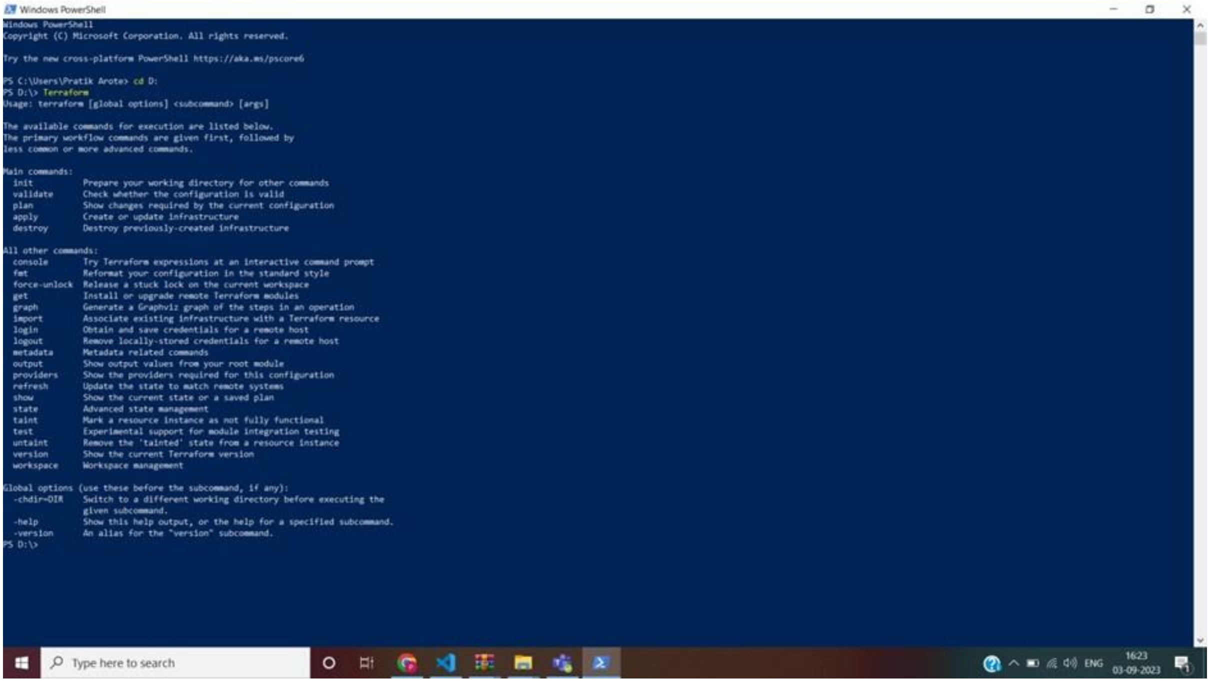
| Variable | Value |
|---|---|
| ComSpec | C:\windows\system32\ |
| DriverData | C:\Windows\System32 |
| IWBPath | C:\Program Files (x86) |
| JAVA_HOME | C:\Program Files\Java\ |
| NUMBER_OF_PROCESSORS | 4 |
| OnlineServices | Online Services |
| OS | Windows_NT |

New System Variable

Variable name: Path

Variable value: D:\Terraform

4. Open PowerShell with Admin Access. Open Terraform in PowerShell and check its functionality.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Pratik Arote> cd D:
PS D:\> Terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
  taint         Mark a resource instance as not fully functional
  test          Experimental support for module integration testing
  untaint       Remove the 'tainted' state from a resource instance
  version       Show the current Terraform version
  workspace     Workspace management

Global options (use these before the subcommand, if any):
  -chdir=DIR    Switch to a different working directory before executing the
                given subcommand.
  -help         Show this help output, or the help for a specified subcommand.
  -version      An alias for the "version" subcommand.
PS D:\>
```

## CONCLUSION:

Here, we studied the about the terraform lifecycle and terminologies/concepts of terraform and installed it on our system.