Adv. Devops Experiment no. 5

Name: Rohan Lalchandani
Class: D15A Roll no.: 25

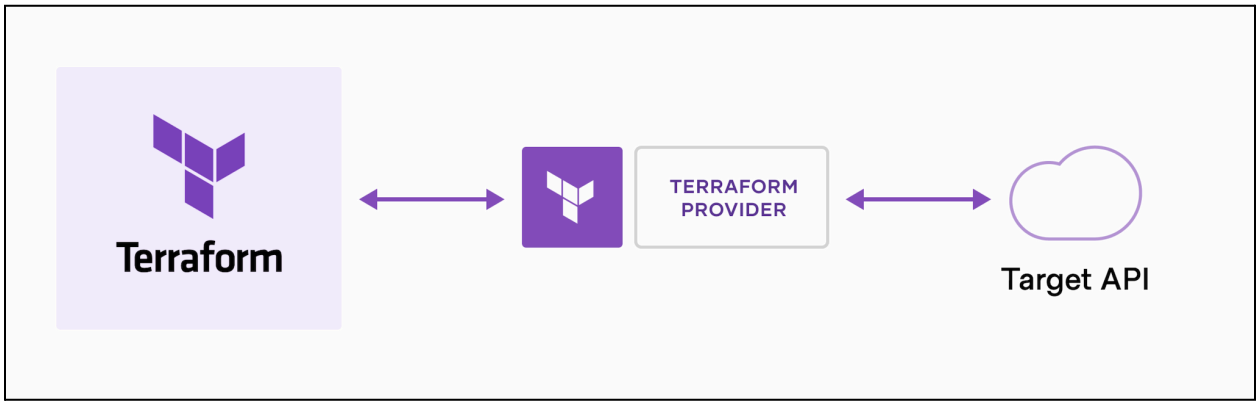
Aim: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.

Theory:

Terraform:
HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

How does Terraform work?

Terraform creates and manages resources on cloud platforms and other services through their application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.



Key Features

1. Infrastructure as Code (IaC)

- **Declarative Configuration:** Users describe the desired final state of infrastructure, and Terraform determines the steps to achieve that state.
- **Version Control:** Infrastructure configurations can be versioned and treated similarly to application code, allowing for easy tracking of changes and collaboration.
- **Reusability:** Modules and configurations can be reused across different projects, promoting consistency and reducing duplication.

2. Multi-Cloud Support

- **Providers:** Terraform supports a wide range of cloud providers such as AWS, Azure, Google Cloud Platform, and many others, including on-premises solutions.
- **Abstraction:** It provides a consistent workflow across different providers, enabling users to manage heterogeneous environments seamlessly.
- **Portability:** Easy to migrate or replicate infrastructure across different cloud environments.

3. Resource Management

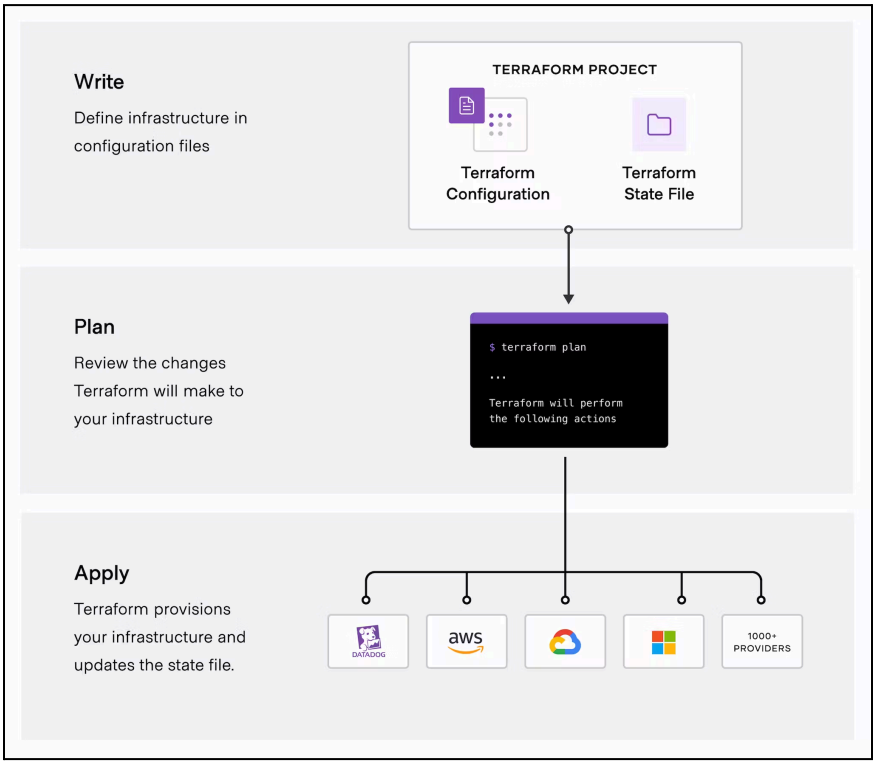
- **Dependency Graphing:** Terraform automatically understands and manages dependencies between resources, ensuring correct order of operations.
- **Plan and Apply:** The `terraform plan` command shows a preview of changes before they are applied, while `terraform apply` executes the changes, providing a safe and predictable workflow.

4. Extensibility

- **Modules:** Users can create and use modules to organize and encapsulate infrastructure configurations, promoting modularity and best practices.
- **Community and Ecosystem:** A vibrant community contributes modules, plugins, and support, enhancing Terraform's capabilities and usability.
- **Custom Plugins:** Ability to develop custom providers and plugins to extend functionality as needed.

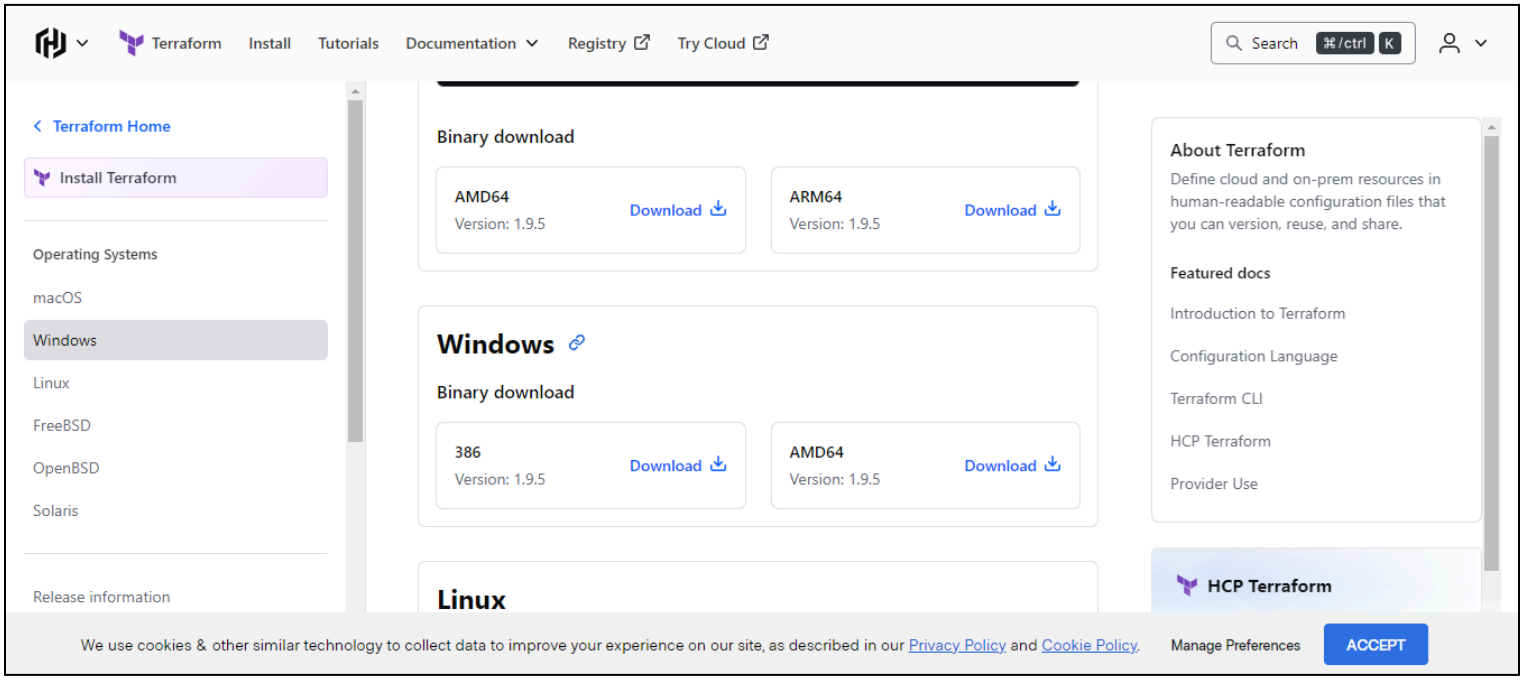
Common Use Cases

- **Provisioning Cloud Infrastructure:** Setting up and managing cloud resources like virtual machines, networks, and storage.
- **Managing Multi-Cloud Environments:** Coordinating resources across different cloud platforms for redundancy and optimization.
- **Infrastructure Migration:** Facilitating the migration of resources from one environment to another through codified configurations.

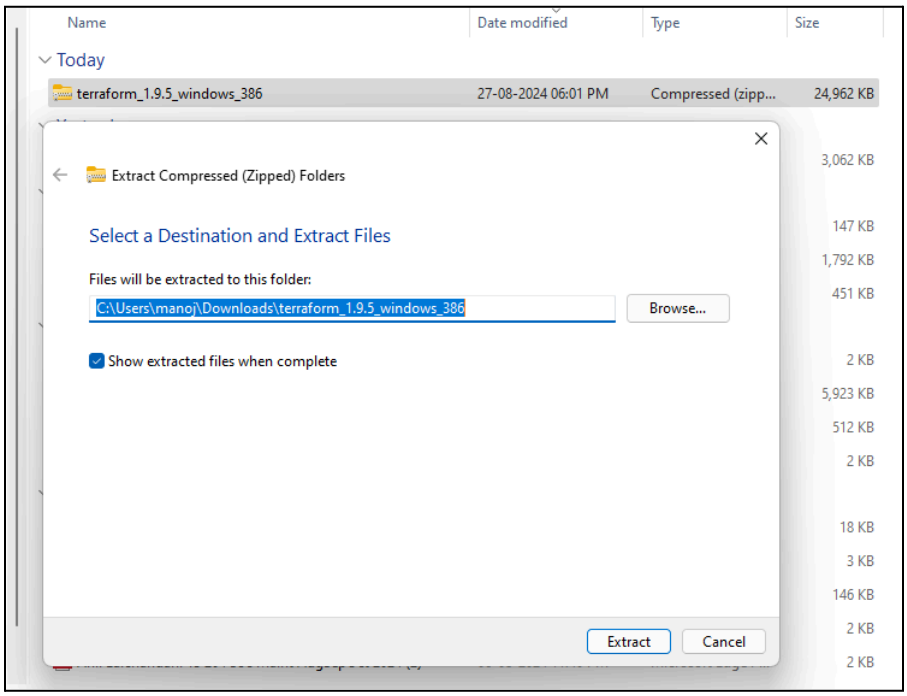


Implementation:

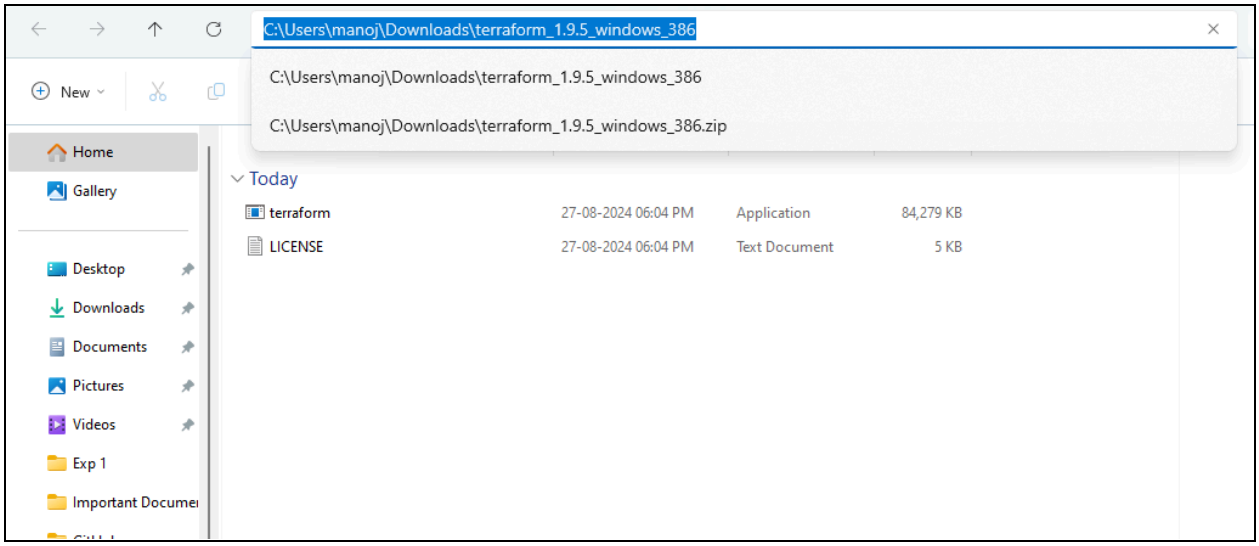
1) Go to website [“https://www.terraform.io/downloads.html”](https://www.terraform.io/downloads.html). Select your OS, in this case we are installing on Windows, download 386.



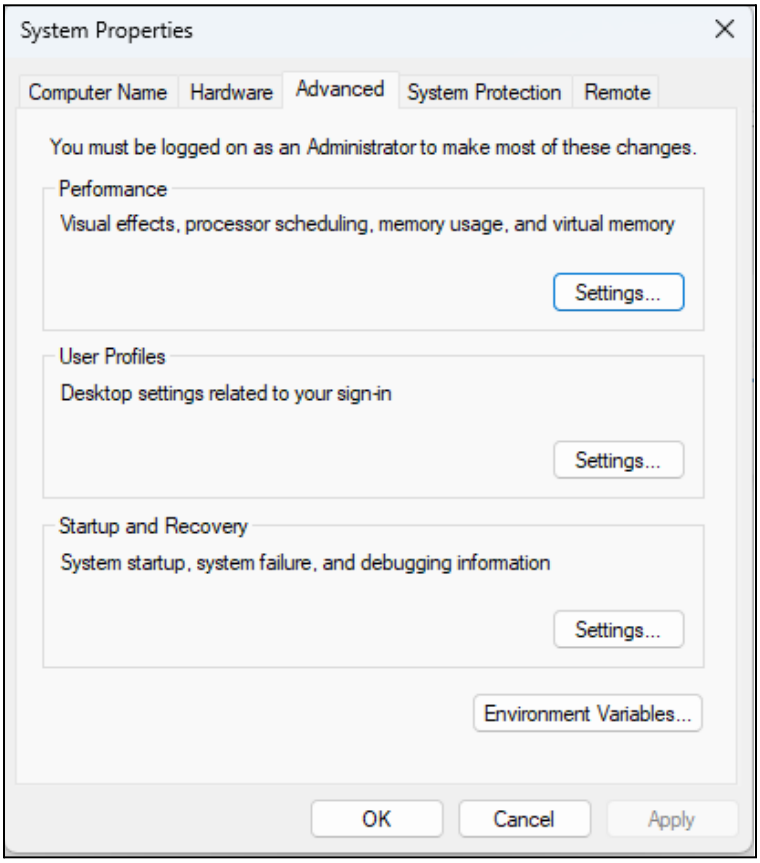
2) Extract the downloaded zip folder.



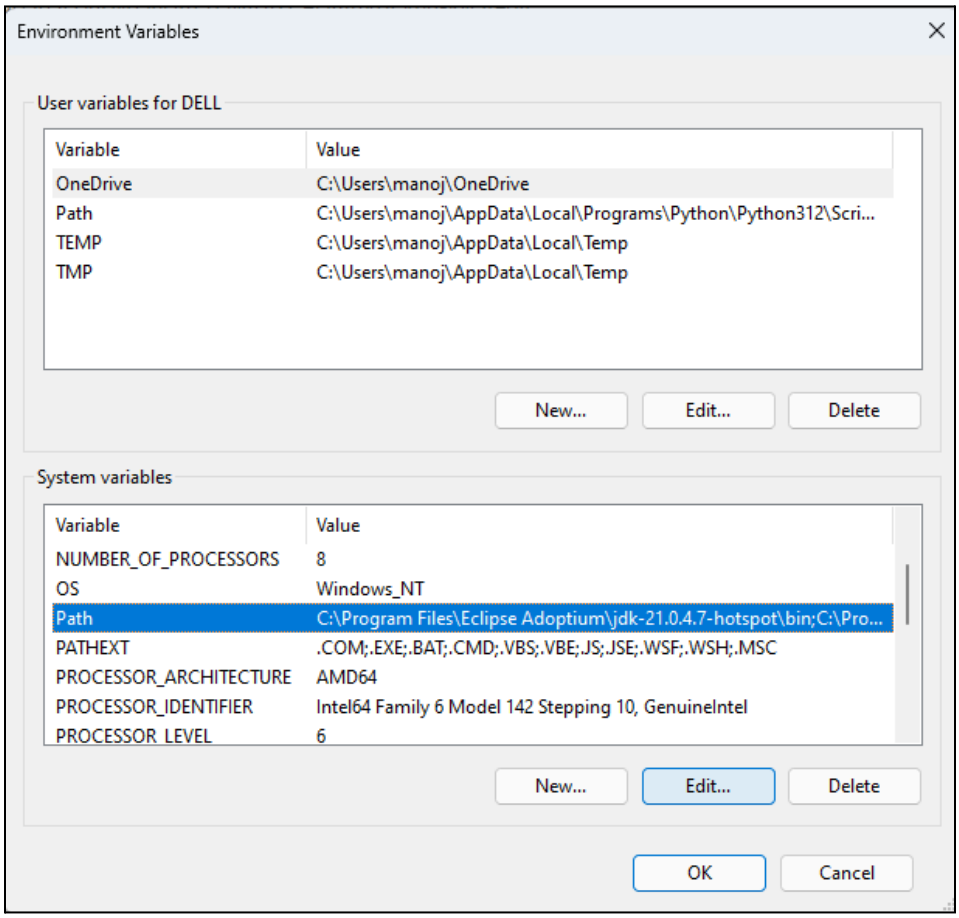
3) Go to the extracted folder and copy the path of the folder as shown below.



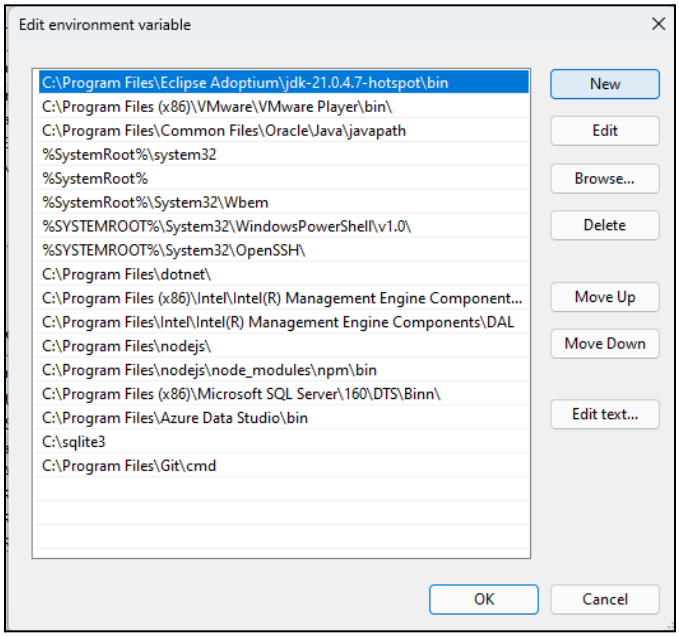
4) Search for “Environment variables” in the Start menu. In the dialog box click on environment variables.



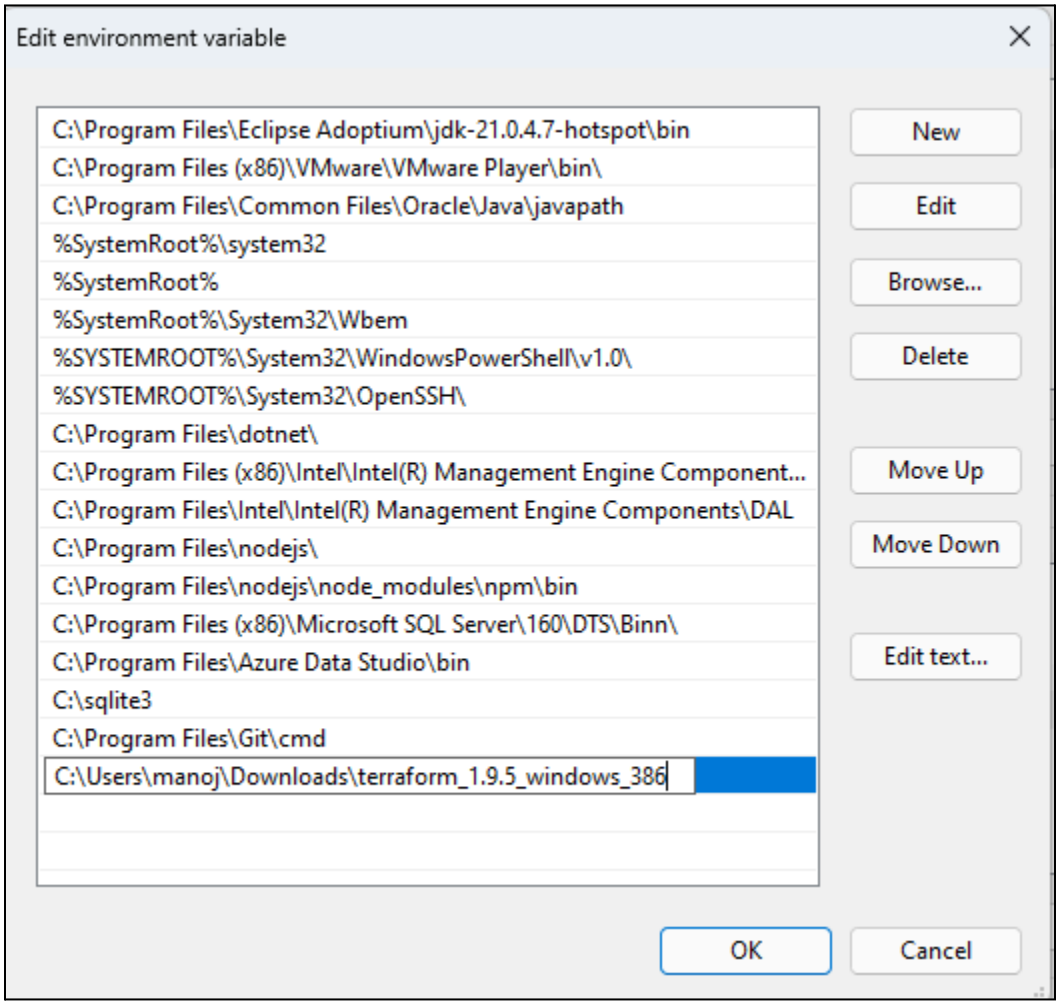
5) Click on path and click on edit as shown below.



6) Click on New.



7) Paste the path of the extracted folder and click on ok.



8) Open cmd and run the command “terraform --version” to check if it is successfully installed.