St. Francis Institute of Technology, Mumbai-400 103

Department of Information Technology

A.Y. 2023-2024

Class: TE-ITA/B, Semester: V

Experiment – 7: To understand Terraform lifecycle, basic concepts / terminologies and install it on Windows/Linux machine.

Subject: **Advanced DevOps Lab**

1. **Aim:** To understand Terraform lifecycle, basic concepts/terminologies and install it on Windows /Linux machine.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand basic Terraform concepts
   - Perform installation of Terraform.
   - Write terraform scripts
3. **Outcomes:** After study of this experiment, the student will be able to
   - To be familiarized with infrastructure as code for provisioning, compliance, and management of any cloud infrastructure and d service.
4. **Prerequisite:** Fundamentals of cloud computing
5. **Requirements:** PC and Internet
6. **Pre-Experiment Exercise:**

Brief Theory:

Terraform

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. This includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc. Terraform can manage both existing service providers and custom in-house solutions.

Key Features

Infrastructure as Code:

You describe your infrastructure using Terraform's high-level configuration language in human- readable, declarative configuration files. This allows you to create a blueprint that you can version, share, and reuse.

Resource Graph

Terraform builds a resource graph and creates or modifies non-dependent resources in parallel. This allows Terraform to build resources as efficiently as possible and gives you greater insight into your infrastructure.

Change Automation

Terraform can apply complex change sets to your infrastructure with minimal human interaction. When you update configuration files, Terraform determines what changed and creates incremental execution plans that respect dependencies.

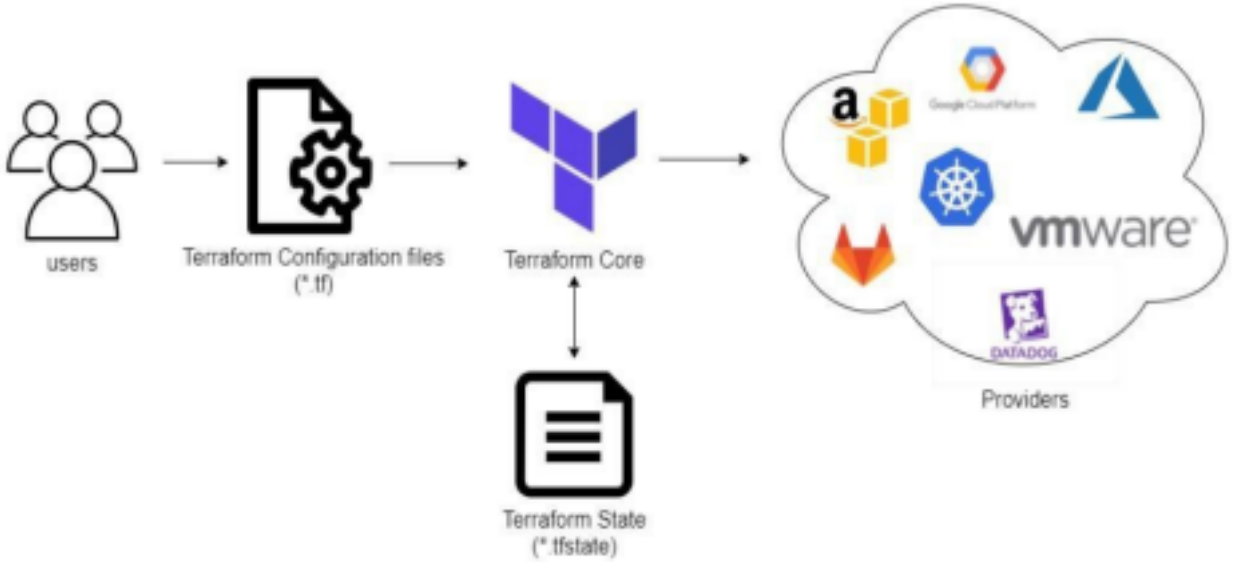
Terraform Life Cycle:

Terraform actually works, there's sort of two major components:

one is the **terraform core:** it takes the terraform configuration which is being provided by the user and then takes the terraform state which is managed by terraform itself. As such, this gets fed into the core that is responsible for figuring out what is that graph of our different resources for exemple how these different pieces relate to each other or what needs to be created/updated/destroyed, it does all the essential lifecycle management.

On the backside, terraform supports many different **providers**, such as: cloud providers (AWS,GCP,AZURE) and they also could be on-premise infrastructure (VMware, OpenStack.) But this support is not restricted or limited only to Infrastructure As A Service , terraform can

also manage higher level like Platform As A Service(Kubernetes, Lambdas..)or even Software As A Service (DataDog, GitHub..)



All of these are important pieces of the infrastructure, they are all part of the logical end-to-end delivery. Terraform has over a hundred providers for different technologies, and each provider gives terraform users access to their resources. It also gives you the

ability to create infrastructure at different levels.

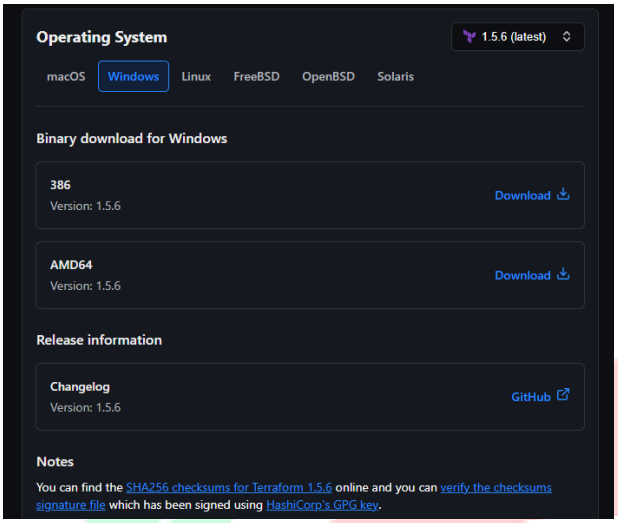
Trraform Core Concepts:

Below are the core concepts/terminologies used in Terraform:

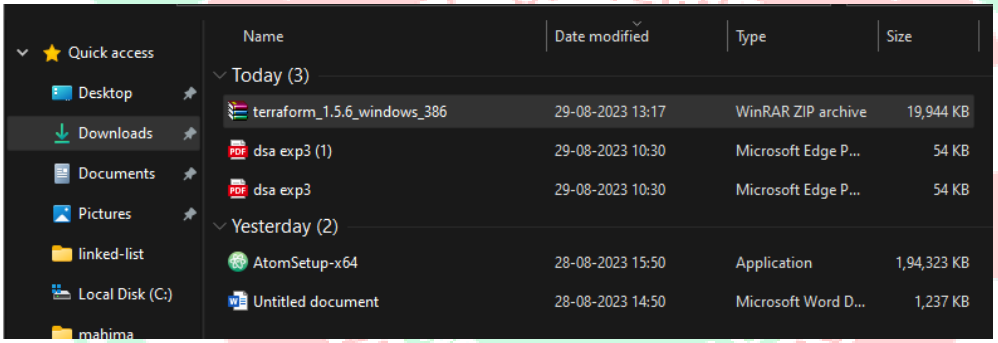
- **Variables:** Also used as input-variables, it is a key-value pair used by Terraform modules to allow customization.
- **Provider:** It is a plugin to interact with APIs of service and access its related resources.
- **Module:** It is a folder with Terraform templates where all the configurations are defined
- **State:** It consists of cached information about the infrastructure managed by Terraform and its related configurations.
- **Resources:** It refers to a block of one or more infrastructure objects (compute instances, virtual networks, etc.), which are used in configuring and managing the infrastructure.
- **Data Source:** It is implemented by providers to return information on external objects to terraform.
- **Output Values:** These are return values of a terraform module that can be used by other configurations.
- **Plan:** It is one of the stages 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.
- **Apply:** It is one of the stages where it applies the changes in the real/current state of the infrastructure in order to move to the desired state.

7. Laboratory Exercise

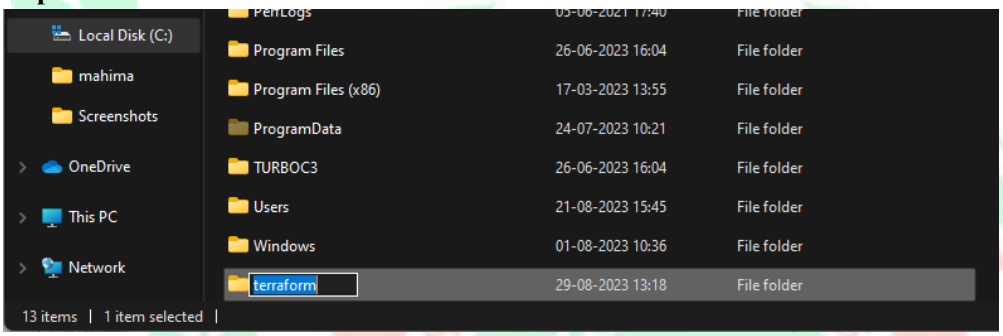
Step 1 : Download appropriate terraform package(.zip) from terraform.io/downloads.html for Windows



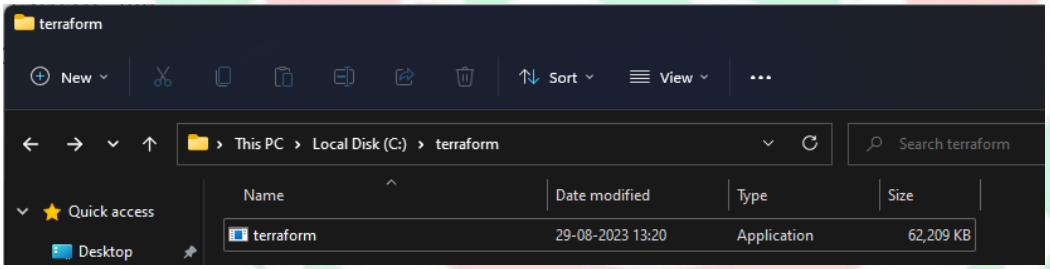
Step 2: Download Terraform for Windows 64-bit / (32-bit).



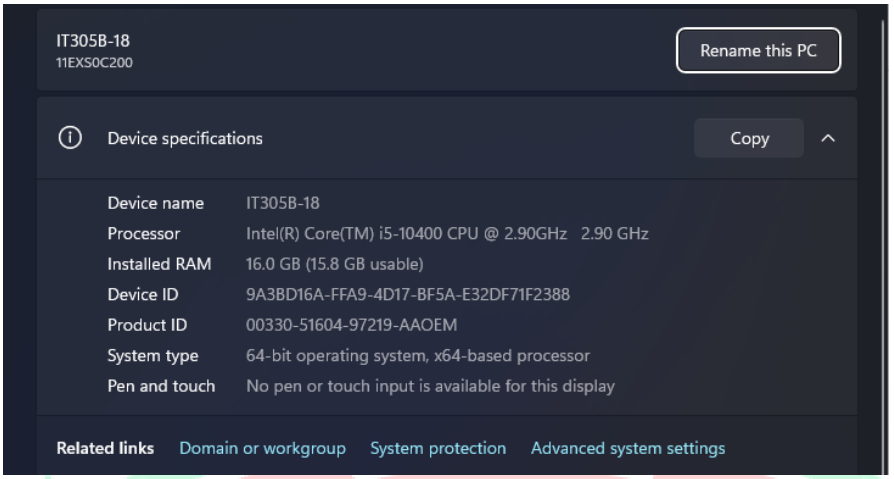
Step 3: Create a folder 'terraform' in drive C .



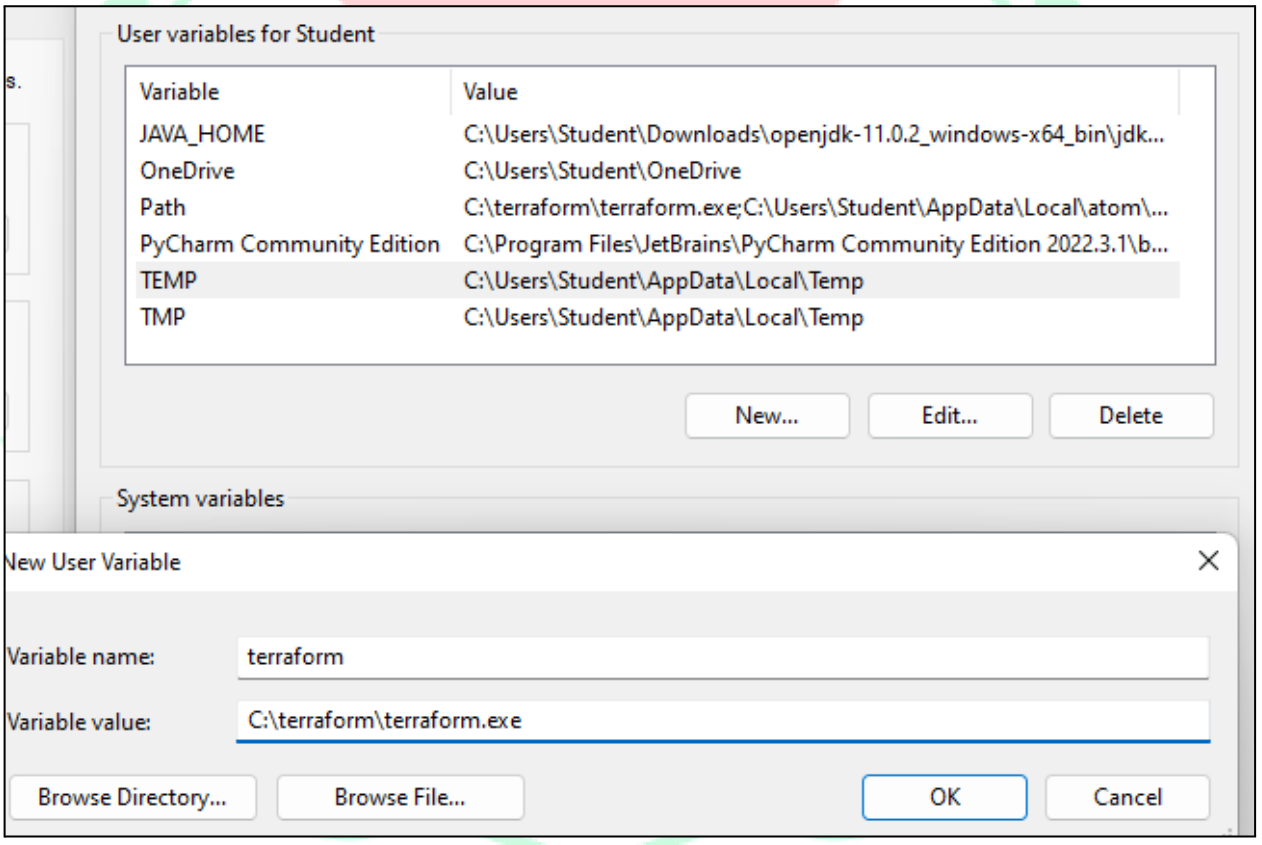
Step 4 : Extract downloaded zip in to this c:/terraform folder



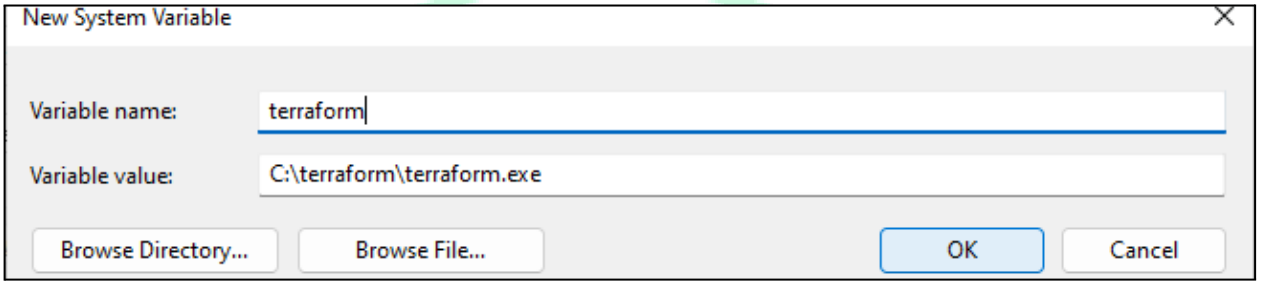
Step 5: Now we need to set path for terraform. Go to My computer/ This PC, right click, select propoerties, go to advanced system settings.



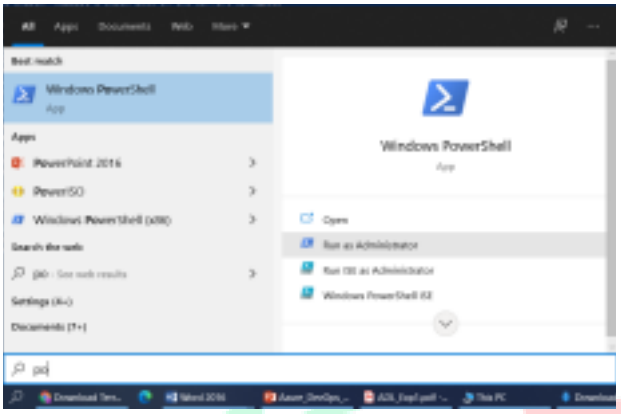
Step 6: click on environment variable



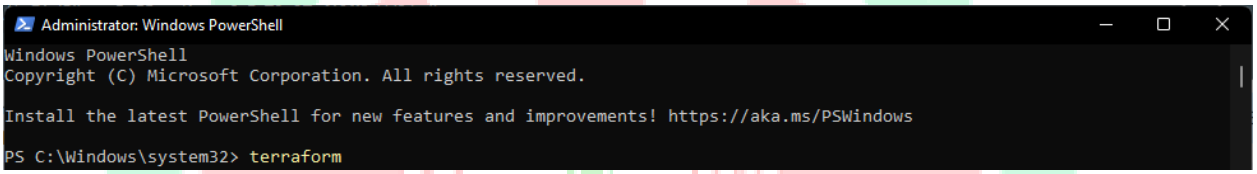
Step 7: click on New, give variable name = Path, click on browse directory, select c:/terraforms/terra….exe, OK



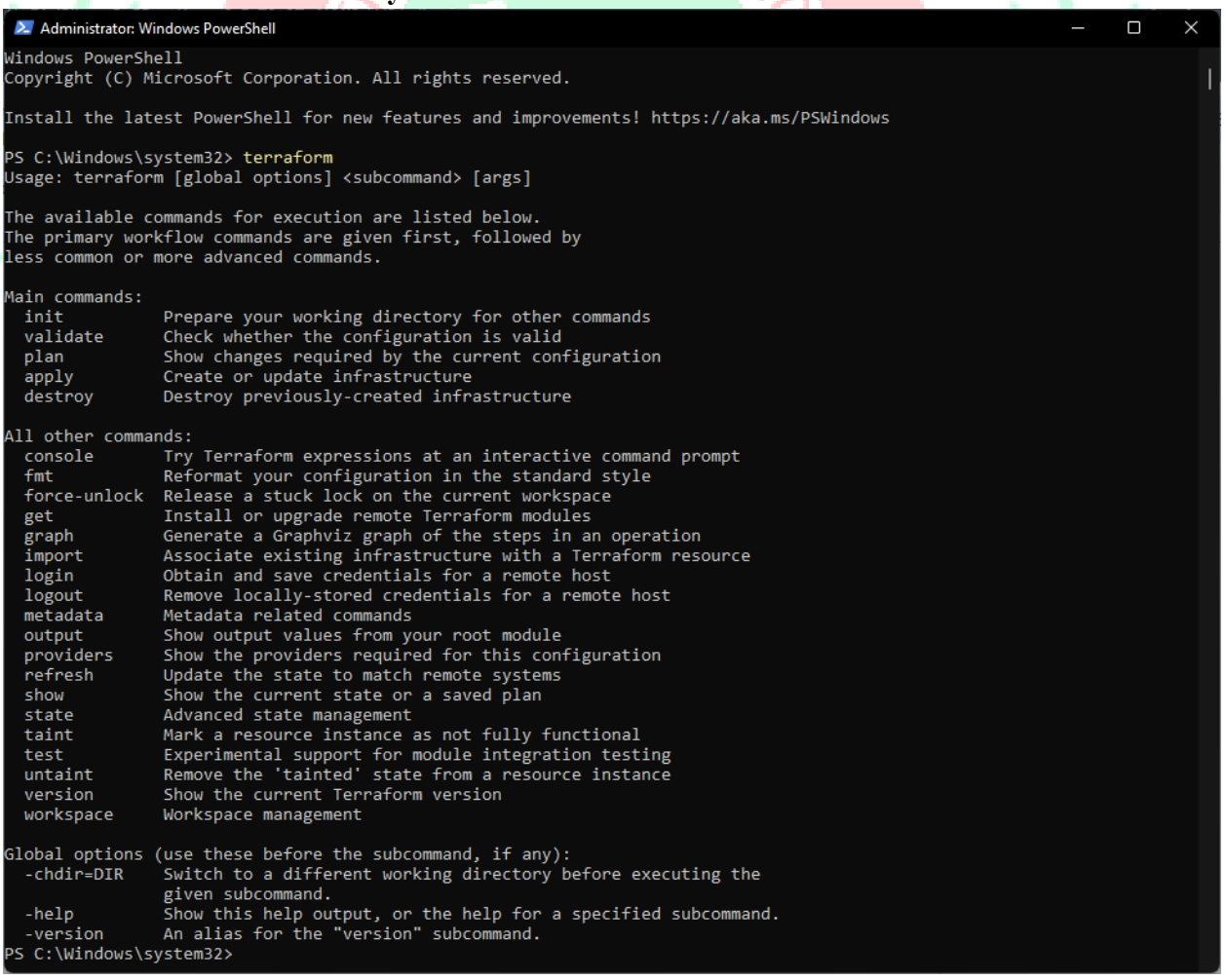
Step 8: Cross verify terraform installed properly or not . go to MS Powershell, run as a administrator



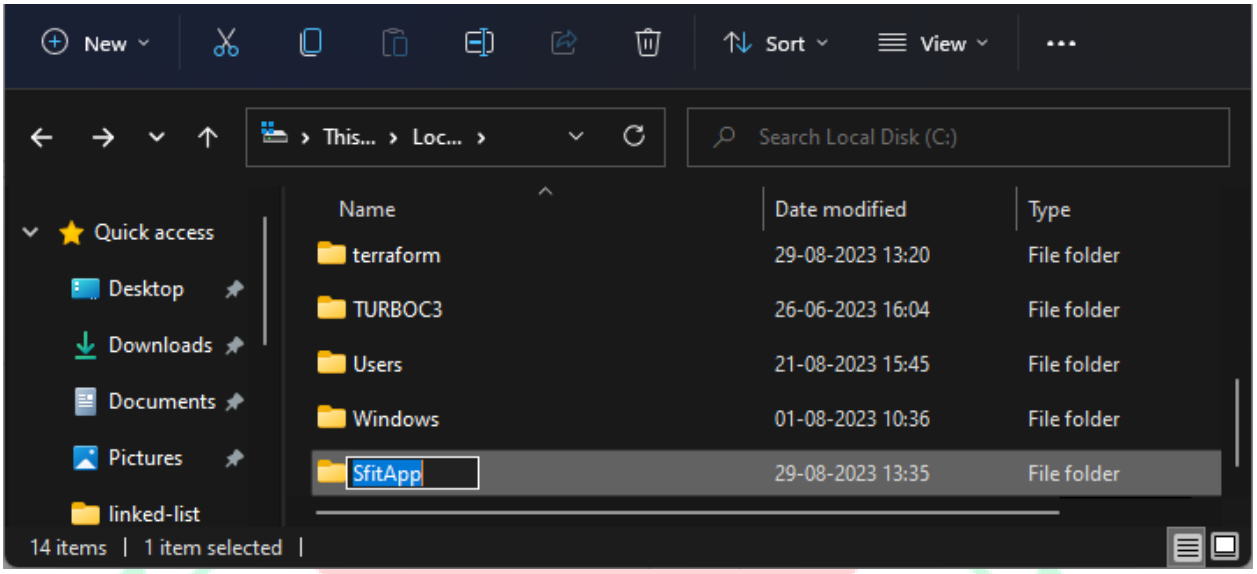
Step 9: Type terraform



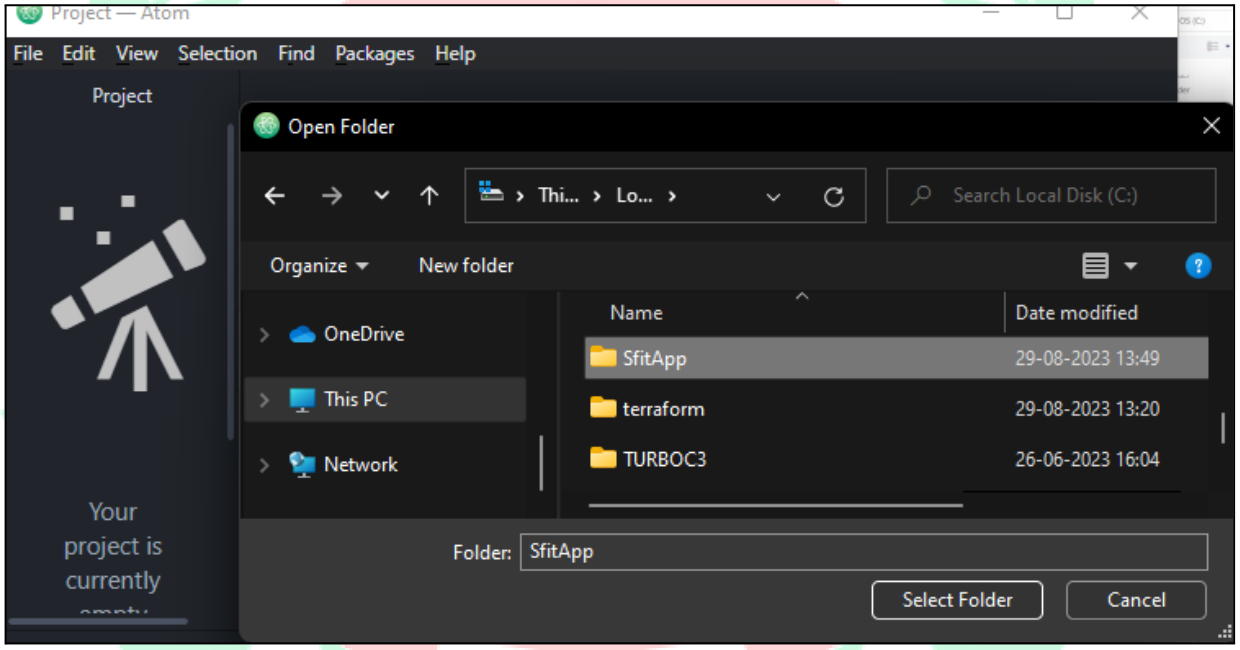
Step 10: You will find init, validate, plan, apply and destroy options means you have installed terraform succesfully.



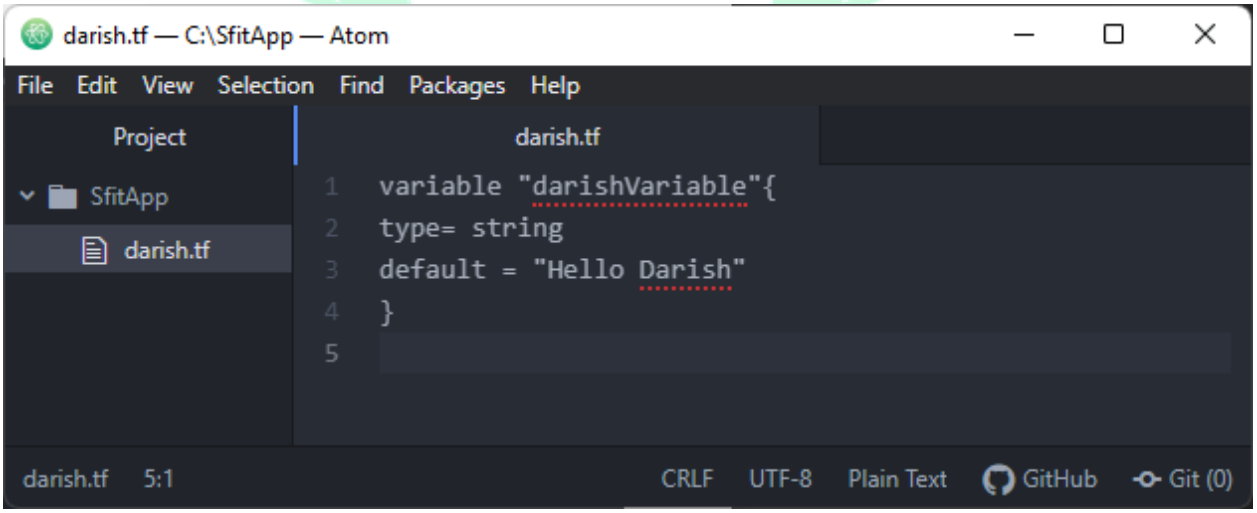
Step 11: Create a folder c:\SfitApp

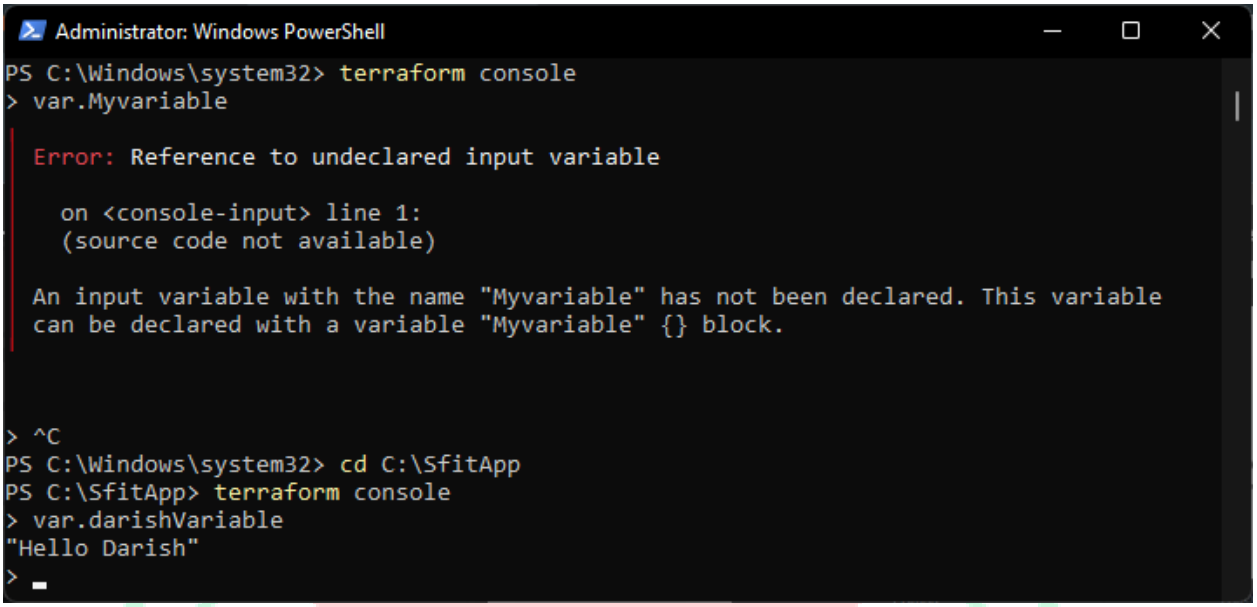


Step 12 : Open Atom/VS CODE Editor …Open folder SfitApp

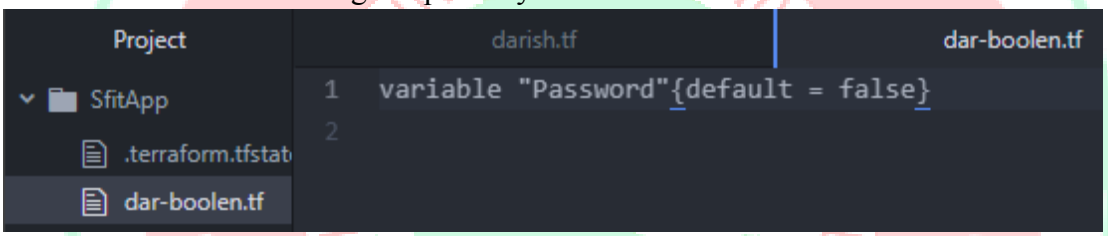


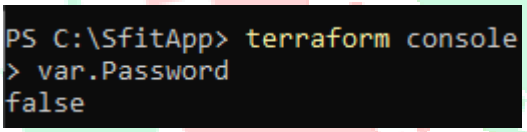
Step 13: write main.tf file with input variables. The input variables, like the one above, use a couple of different types: string, list, map, and Boolean.





Step 14: Check the output on command Prompt…Go to C:\SfitApp, Type Terraform Console You will get terraform prompt, run the .tf with var.MyVariable1, You will get welcome to SFIT msg. Step 15: try Boolean variable…Create Boolean.tf





8. Post-Experiments Exercise

A. Extended Theory:

- Terraform Vs. Kubernetes
- Terraform Vs. Ansible

B. Questions:

1. Name all version controls supported by Terraform.
2. What is TerraGrunt?

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

https://www.ibm.com/cloud/learn/terraform#toc-terraform--OoC-5III
https://www.simplilearn.com/terraform-interview-questions-and-answers-article
https://aws.amazon.com/microservices/
https://www.monkeyvault.net/docker-vs-virtualization/
https://cloudacademy.com/blog/docker-vs-virtualization/
https://www.terraform.io/docs/language/values/variables.html
