Name - Priyanshu Lapkale
Class - TY B
Roll No - 322067
PRN - 22220008

Assignment 5

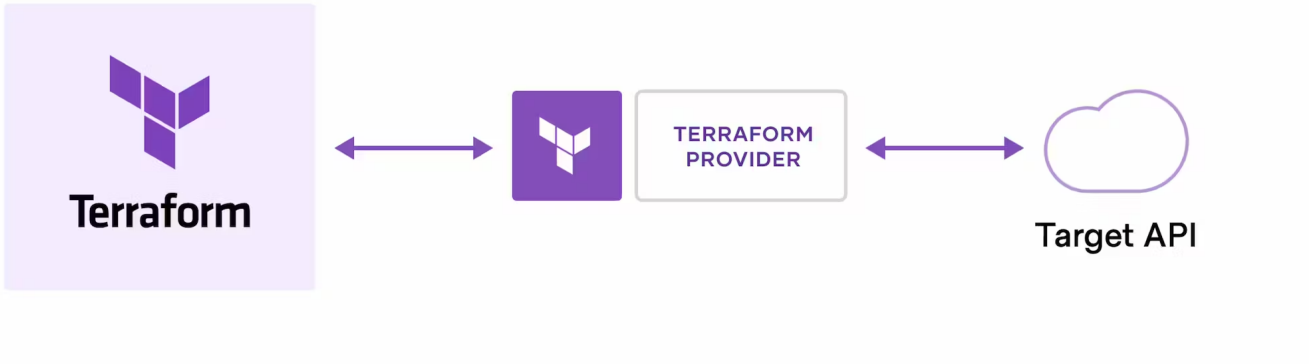
Write IaC using terraform to create EC2 machine on AWS or azure or google cloud. (Compulsory to use Input and output variable files)

What is 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.



HashiCorp and the Terraform community have already written thousands of providers to manage many different types of resources and services. You can find all publicly available providers on the Terraform Registry, including Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog, and many more.

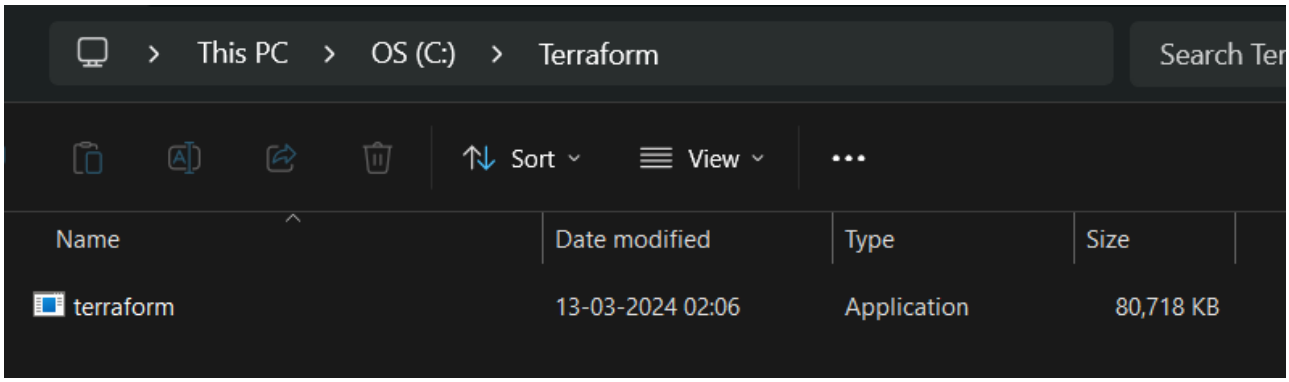
The core Terraform workflow consists of three stages:

- Write: You define resources, which may be across multiple cloud providers and services. For example, you might create a configuration to deploy an application on virtual machines in a Virtual Private Cloud (VPC) network with security groups and a load balancer.

- Plan: Terraform creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and your configuration.
- Apply: On approval, Terraform performs the proposed operations in the correct order, respecting any resource dependencies. For example, if you update the properties of a VPC and change the number of virtual machines in that VPC, Terraform will recreate the VPC before scaling the virtual machines.

Terraform Installation

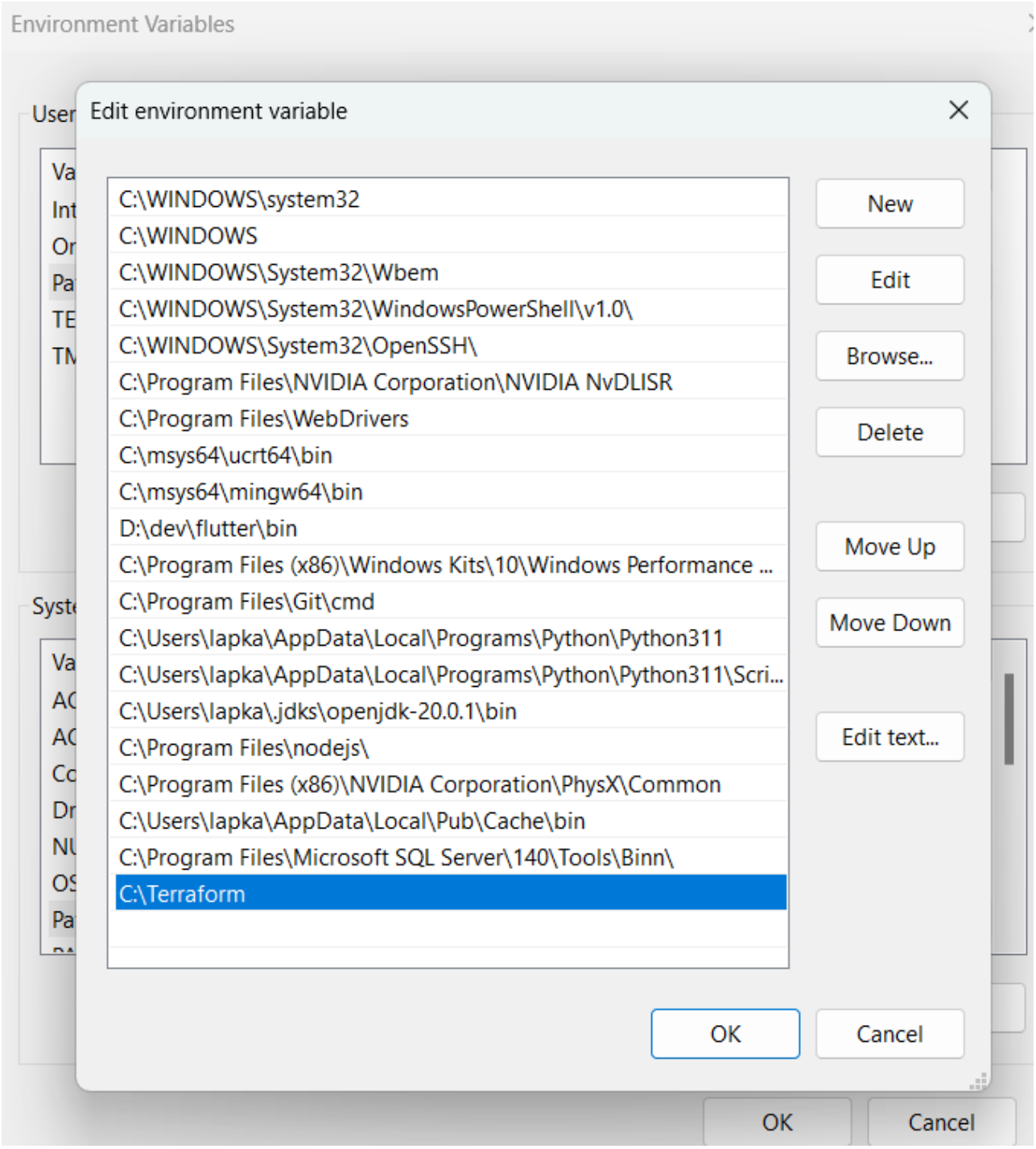
1. Download Terraform - Use the official HashiCorp link to download Terraform depending on your Operating System



2. Setup Environment Variable

- Click on “Search” → Search “Edit environment variables”
- Under “Advanced” click on “Environment Variables”
- Click “Path” under System Variables

- Select “Path” → “Add New” → “Terraform_Path” → Paste the path location example — C:\terraform



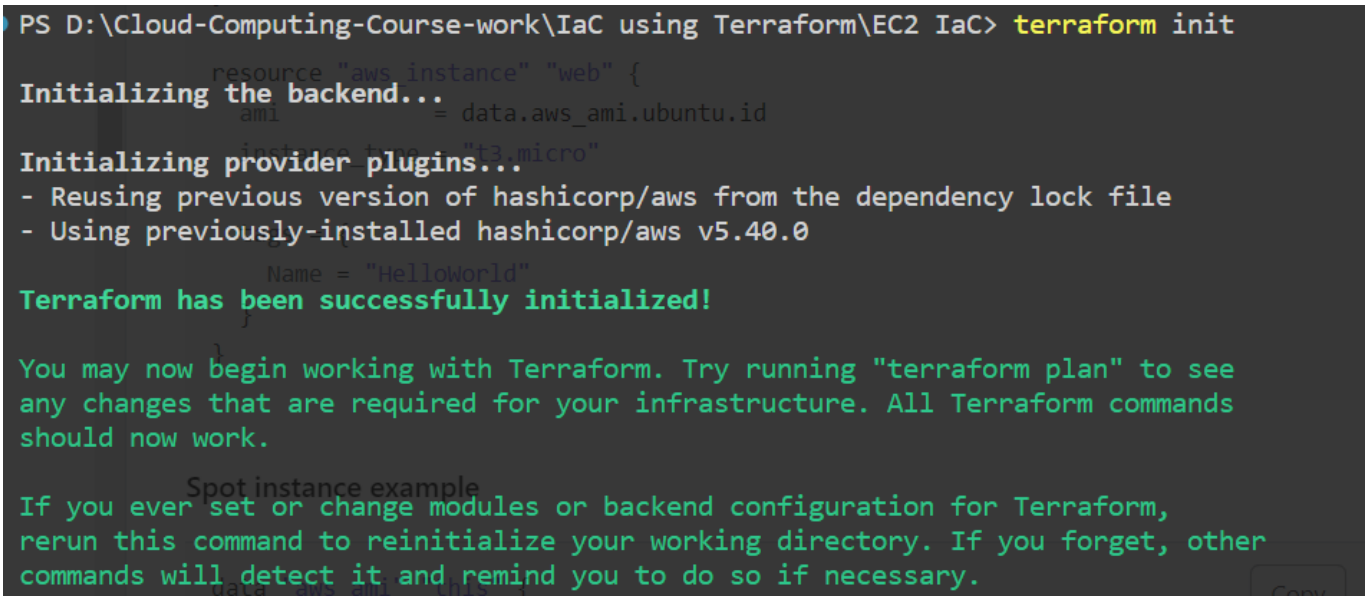
3. Check if it is installed

```
$ terraform -version
```

Creating EC2 using Terraform

1. Create a folder and initialize terraform-

```
$ terraform init
```



2. Create main.tf file and define the infrastructure of your EC2 instance -

```
provider "aws" {
  region = var.aws_region
  access_key = var.access_key
  secret_key = var.secret_key
}

resource "aws_instance" "ec2_instance" {
  count         = var.instance_count
  ami           = var.ami_id
  instance_type = var.instance_type
  tags = {
    Name = "Assignment 5"
  }
}

output "instance_public_ips" {
  value = aws_instance.ec2_instance[*].public_ip
}
```

3. Create variables.tf file where we'll define all the required variables -

```
variable "aws_region" {
  description = "The AWS region to deploy the EC2 instance in"
  type        = string
}
variable "access_key" {
  description = "The AWS region to deploy the EC2 instance in"
  type        = string
}
```

```
variable "secret_key" {
  description = "The AWS region to deploy the EC2 instance in"
  type        = string
}

variable "instance_type" {
  description = "The type of EC2 instance to launch"
  type        = string
}

variable "ami_id" {
  description = "The ID of the AMI to use for the EC2 instance"
  type        = string
}

variable "instance_count" {
  description = "The number of EC2 instances to launch"
  type        = number
}
```

4. Now create terraform.tfvars file where we'll give all input variable data -

```
aws_region     = "ap-south-1"
instance_type  = "t2.micro"
ami_id         = "ami-001843b876406202a"
access_key     = "******"
secret_key     = "******"
instance_count = 1
```

5. Run this command to know exact plan -

```
$ terraform plan
```

```
Terraform will perform the following actions:

  # aws_instance.ec2_instance[0] will be created
  + resource "aws_instance" "ec2_instance" {
      + ami                                  = "your_ami_id"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags_all                             = (known after apply)
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + instance_public_ips = [
      + (known after apply),
    ]
```

```
      + (known after apply),
    }
  ]
- subnet-id-output     = "subnet-0097c28c4ddab64f0" -> null
- vpc-id-output        = "vpc-0f196d5f1f04e8ab3" -> null
```

6. Run this command to execute your IaC -

```
$ terraform apply
```

This will show you plan again and ask for confirmation to apply changes

```
Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  ~ instance_public_ips = [
      - "13.233.255.191",
      + (known after apply),
    ]

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.ec2_instance[0]: Creating...
aws_instance.ec2_instance[0]: Still creating... [10s elapsed]
aws_instance.ec2_instance[0]: Still creating... [20s elapsed]
aws_instance.ec2_instance[0]: Creation complete after 21s [id=i-03bd813c398d66383]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

instance_public_ips = [
  "3.110.185.16",
]
```

Thus we've successfully created EC2 instance in AWS using Terraform -