

**Bansilal Ramnath Agarwal Charitable Trust's
VISHWAKARMA INSTITUTE OF INFORMATION
TECHNOLOGY,**

PUNE-48 Department of Information Technology

**ITUA32202: CLOUD COMPUTING
Assignment-5**

Shreyas Shripad Kulkarni

C2 Batch

Roll No. 333030

PRN: 22010443

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

THEORY:

1) What is Terraform




-> Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp. It allows developers to define and manage their infrastructure in a declarative way, using configuration files instead of manual processes.

With Terraform, infrastructure changes can be easily versioned, tested, and deployed across multiple cloud providers, such as AWS, Google Cloud Platform, and Microsoft Azure. Terraform's modular design allows for easy reuse of infrastructure code and promotes collaboration among team members. It automates the deployment of infrastructure, making it more efficient and reliable, reducing the risk of human error.

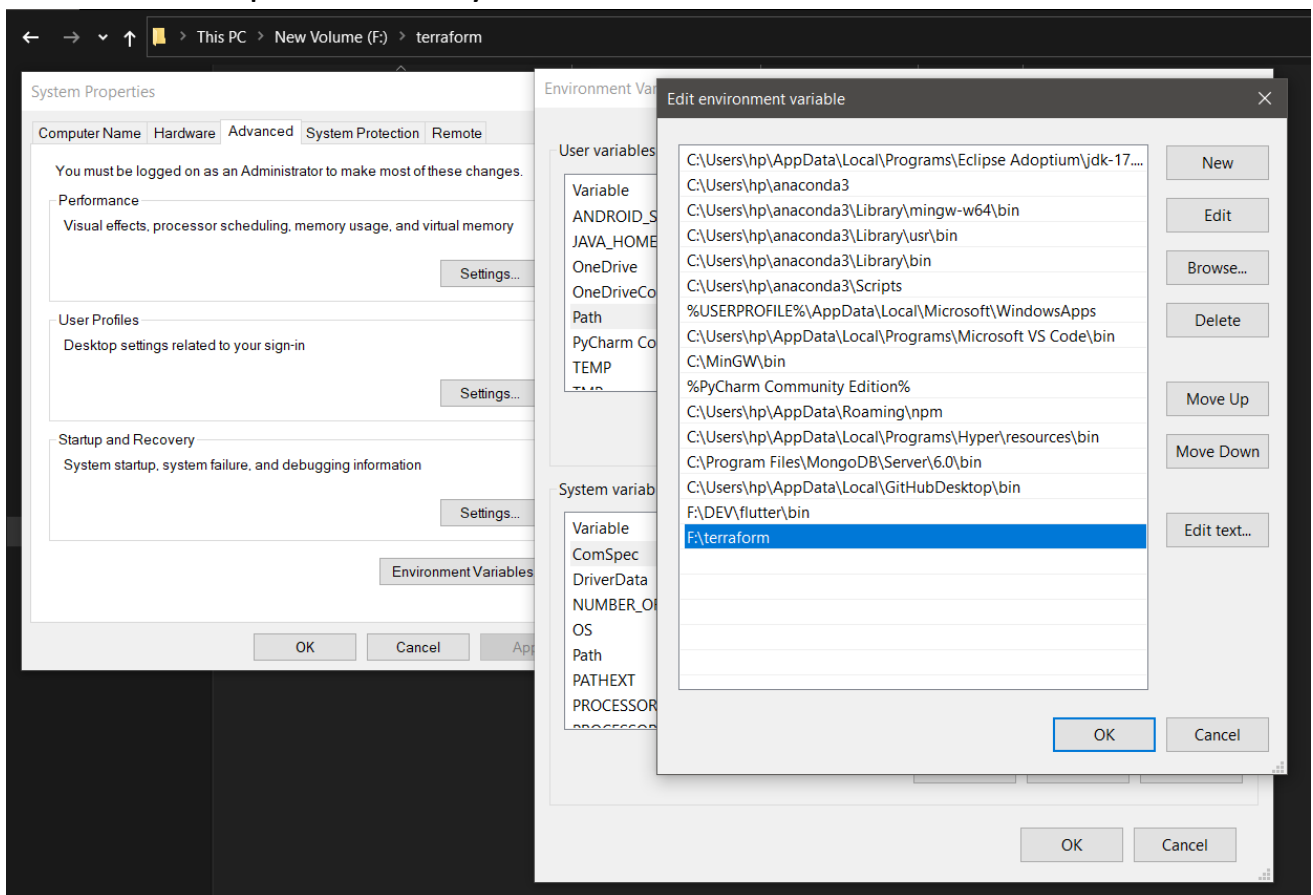
Terraform helps teams to manage their infrastructure as code, and provides greater agility and scalability to their projects.

2) Step-by-step screenshot to install and configure Terraform.

- i. Download Terraform from :
<https://developer.hashicorp.com/terraform/downloads>
- ii. This will download a zip file. We have to extract the zip file to desired location

 terraform_1.3.9_windows_amd64	27-02-2023 02:22	WinRAR ZIP archive	19,483 KB
 terraform	27-02-2023 17:29	File folder	
 terraform	15-02-2023 22:51	Application	60,697 KB

- iii. Add the path to the system environment variables



- iv. Check if installation is successful

```
D:\TF>terraform --version
Terraform v1.3.9
on windows_amd64
+ provider registry.terraform.io/hashicorp/aws v4.56.0
```

We have successfully installed terraform in our Windows System

3) Terraform script to create Infrastructure on any cloud platform (AWS or Azure or Google)

+ Create a main.tf file (terraform file)

Mention the provider. We are going to use AWS as our provider.

The region is Mumbai (ap-south-1)

The ami id of the VM is added along with other details.

```
main.tf x
main.tf > ...
1 terraform {
2     required_providers {
3         aws = {
4             source = "hashicorp/aws"
5             version = "~> 4.16"
6         }
7     }
8
9     required_version = ">= 1.2.0"
10 }
11
12 provider "aws" {
13     region = "ap-south-1" //Mumbai Region
14 }
15 //VM
16 resource "aws_instance" "app_server" {
17     ami = "ami-0e742cca61fb65051"
18     instance_type = "t2.micro"
19
20     tags = {
21         Name = "Instance-TF" //Name of the ec2 instance
22     }
23 }
24
```

- ✚ Using the `terraform init` command initialize the Terraform Project. This will download any required plugins and initialize the working directory.

```
D:\TF>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 4.16"...
- Installing hashicorp/aws v4.56.0...
- Installed hashicorp/aws v4.56.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

D:\TF>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions
the following symbols:
+ create

Terraform will perform the following actions:
```

- ✚ `terraform apply`
This will create or update the resources specified in your configuration files after giving the plan.

```
D:\TF>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions
the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.app_server will be created
+ resource "aws_instance" "app_server" {
  + ami              = "ami-0e742cca61fb65051"
  + 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_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)
```

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

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.app_server: Creating...
```

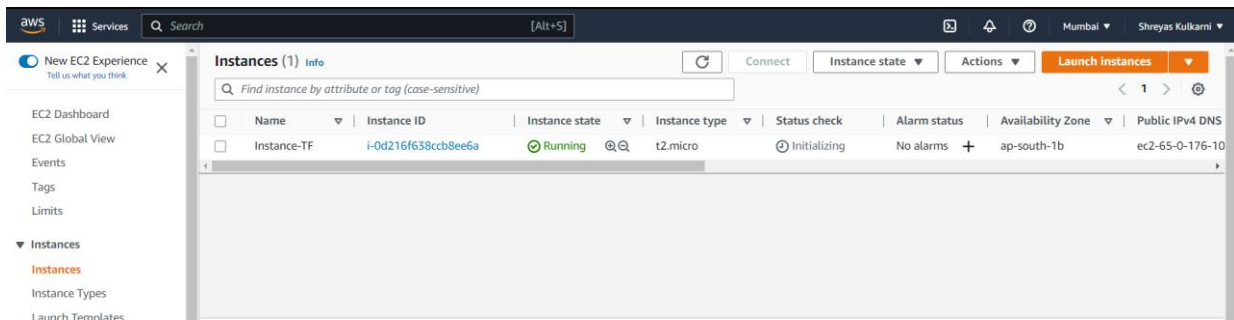
```
aws_instance.app_server: Still creating... [10s elapsed]
```

```
aws_instance.app_server: Still creating... [20s elapsed]
```

```
aws_instance.app_server: Creation complete after 22s [id=i-0d216f638ccb8ee6a]
```

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

✚ The Instance has been deployed Successfully.



✚ We can delete the instance using the `terraform destroy`.

This command will destroy & delete all the instances that were created using terraform.

```
D:\TF>terraform destroy
```

```
aws_instance.app_server: Refreshing state... [id=i-0d216f638ccb8ee6a]
```

Terraform used the selected providers to generate the following execution plan. Resources to be destroyed are shown in red.

```
- destroy
```

Terraform will perform the following actions:

```
# aws_instance.app_server will be destroyed
```

```
- resource "aws_instance" "app_server" {
```

```
  - ami                                = "ami-0e742cca61fb65051" -> null
```

```
  - arn                                = "arn:aws:ec2:ap-south-1:905151731728" -> null
```

```
  null
```

```
  - associate_public_ip_address        = true -> null
```

```
  - availability_zone                   = "ap-south-1b" -> null
```

```
  - cpu_core_count                      = 1 -> null
```

```
  - cpu_threads_per_core                = 1 -> null
```

```
  - disable_api_stop                    = false -> null
```

```
  - disable_api_termination             = false -> null
```

```
  - ebs_optimized                      = false -> null
```

Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

```
aws_instance.app_server: Destroying... [id=i-0d216f638ccb8ee6a]
aws_instance.app_server: Still destroying... [id=i-0d216f638ccb8ee6a, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-0d216f638ccb8ee6a, 20s elapsed]
aws_instance.app_server: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
```

<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check
<input type="checkbox"/>	Instance-TF	i-0d216f638ccb8ee6a	Terminated 🔍	t2.micro	-

USING INPUT & OUTPUT VARIABLE FILES


✚ DEFINING INPUT VARIABLES

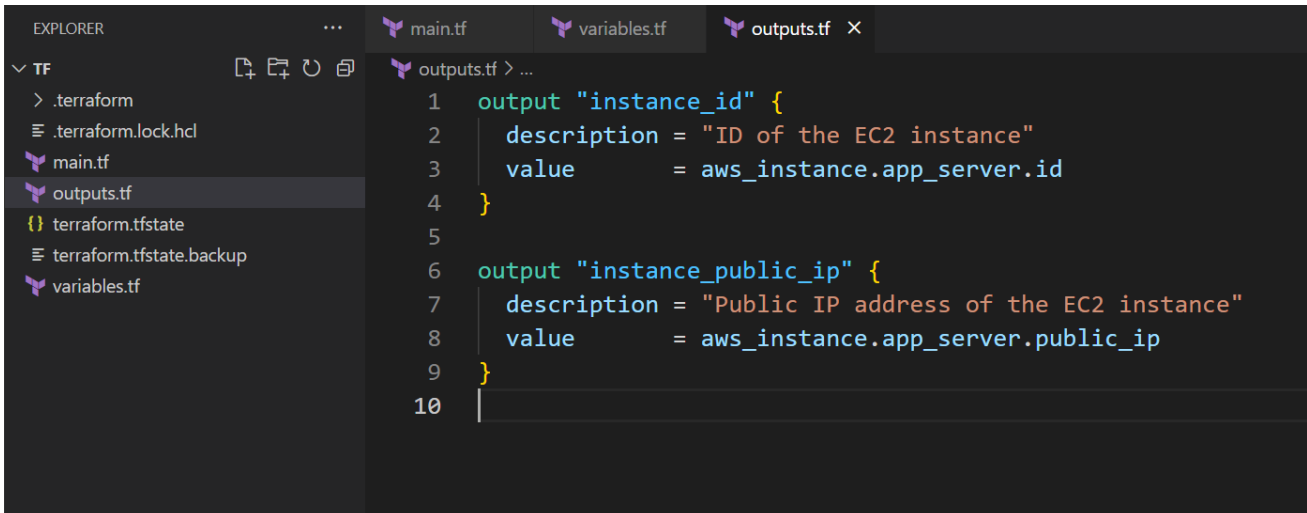
✚ terraform apply -var "instance_name=Terraform-Instance"

This command will change the instance name by editing the name(It will not delete and create new instance)

Terraform configurations can include variables to make your configuration more dynamic and flexible.

Instances (1) Info							Refresh	Connect	Instance state ▾	Actions
<input type="text" value="Find instance by attribute or tag (case-sensitive)"/>										
Instance state = running ✕ Clear filters										
<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status				
<input type="checkbox"/>	Instance-TF	i-0972cfd2c03ddf33	Running 🔍	t2.micro	2/2 checks passed	No alarms				

 Query Data with Outputs: We will use output values to present useful information to the Terraform user.



```
1 output "instance_id" {
2     description = "ID of the EC2 instance"
3     value       = aws_instance.app_server.id
4 }
5
6 output "instance_public_ip" {
7     description = "Public IP address of the EC2 instance"
8     value       = aws_instance.app_server.public_ip
9 }
10
```

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

Changes to Outputs:

```
+ instance_id           = "i-0972cfcd2c03ddf33"
+ instance_public_ip    = "3.108.51.231"
```

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.app_server: Modifying... [id=i-0972cfcd2c03ddf33]




aws_instance.app_server: Modifications complete after 1s [id=i-0972cfcd2c03ddf33]

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

Outputs:

```
instance_id = "i-0972cfcd2c03ddf33"
instance_public_ip = "3.108.51.231"
```

```
D:\TF>terraform output
instance_id = "i-0972cfcd2c03ddf33"
instance_public_ip = "3.108.51.231"
```

<input type="checkbox"/>	Name	Instance ID	Instance state	
<input type="checkbox"/>	Terraform-Instance	i-0972cfcd2c03ddf33	 Running	 


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

Changes to Outputs:

```
- instance_id          = "i-0972cfcd2c03ddf33" -> null
- instance_public_ip   = "3.108.51.231" -> null
```

Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

```
aws_instance.app_server: Destroying... [id=i-0972cfcd2c03ddf33]
aws_instance.app_server: Still destroying... [id=i-0972cfcd2c03ddf33, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-0972cfcd2c03ddf33, 20s elapsed]
aws_instance.app_server: Still destroying... [id=i-0972cfcd2c03ddf33, 30s elapsed]
aws_instance.app_server: Destruction complete after 40s
```

```
Destroy complete! Resources: 1 destroyed.
```

Conclusion: We have installed terraform and AWS CLI and learned how to use Terraform to create infrastructure as code, specifically an EC2 instance on AWS. We also learned how to use input and output variables to make our code more flexible and reusable. An EC2 Instance was deployed and also destroyed using CLI.