Module 8 – Terraform

Assignment-5

1. Destroy the previous deployments

2. Create a script to install Apache2

3. Run this script on a newly created EC2 instance

4. Print the IP address of the instance in a file on the local once deployed

Destoyed the previous deployment with terraform destroy command

```
ubuntu@ip-172-31-35-160:~/assignmnet4$ terraform destroy
aws_vpc.main: Refreshing state... [id=vpc-07ed484825eb50b4d]
aws_subnet.main: Refreshing state... [id=subnet-04a34b5ba015dfc8e]
aws_instance.this: Refreshing state... [id=i-06059bf1f3b9e514a]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.this will be destroyed
  - resource "aws_instance" "this" {
      - ami                                  = "ami-05fb0b8c1424f266b" -> null
      - arn                                  = "arn:aws:ec2:us-east-2:571055632388:instance/i-06059bf1f3b9e514a" -> null
```

```
Plan: 0 to add, 0 to change, 3 to destroy.

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.this: Destroying... [id=i-06059bf1f3b9e514a]
aws_instance.this: Still destroying... [id=i-06059bf1f3b9e514a, 10s elapsed]
aws_instance.this: Still destroying... [id=i-06059bf1f3b9e514a, 20s elapsed]
aws_instance.this: Still destroying... [id=i-06059bf1f3b9e514a, 30s elapsed]
aws_instance.this: Destruction complete after 30s
aws_subnet.main: Destroying... [id=subnet-04a34b5ba015dfc8e]
aws_subnet.main: Destruction complete after 0s
aws_vpc.main: Destroying... [id=vpc-07ed484825eb50b4d]
aws_vpc.main: Destruction complete after 1s

Destroy complete! Resources: 3 destroyed.
```

Then created a new direcoty named assignment and written a main.tf with a hcl script.

```
ubuntu@ip-172-31-35-160:~/assignmnet4$ cd
ubuntu@ip-172-31-35-160:~$ mkdir assignment5
ubuntu@ip-172-31-35-160:~$ cd assignment5
ubuntu@ip-172-31-35-160:~/assignment5$ ls
ubuntu@ip-172-31-35-160:~/assignment5$ vi main.tf
```

```
provider "aws" {
        secret_key = "Pr7tQthXHgkb3dgT+S1010wfwkQPaDrusDdD8Rf8"
        access_key = "AKIAYJ5MROACCA2BOW5E"
        region = "us-east-2"
}
resource "aws_instance" "this" {
        ami = "ami-05fb0b8c1424f266b"
        instance_type = "t2.micro"
        key_name = "venkatohio"
        user_data = "${file("install-apache.sh")}"
        tags = {
                Name = "Assignment1"
        }
}
output "ipv4" {
      value =  aws_instance.this.public_ip
}
~
~
~
"main.tf" 17L, 473B
```

Created shell script

```
ubuntu@ip-172-31-35-160:~/assignment5$ vi install-apache.sh
```

```
#!bin/bash
sudo apt update
sudo apt install apache2 -y
~
~
~
~
```

Initialised terraform

```
ubuntu@ip-172-31-35-160:~/assignment5$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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.
```

Performed terraform plan

```
ubuntu@ip-172-31-35-160:~/assignment5$ terraform plan

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

Terraform will perform the following actions:

  # aws_instance.this will be created
  + resource "aws_instance" "this" {
      + ami                                  = "ami-05fb0b8c1424f266b"
      + 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)
```

Applied changes with terraform aplly command

```
ubuntu@ip-172-31-35-160:~/assignment5$ terraform apply

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

Terraform will perform the following actions:

  # aws_instance.this will be created
  + resource "aws_instance" "this" {
      + ami                                  = "ami-05fb0b8c1424f266b"
      + 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)
```

```
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.this: Creating...
aws_instance.this: Still creating... [10s elapsed]
aws_instance.this: Still creating... [20s elapsed]
aws_instance.this: Still creating... [30s elapsed]
aws_instance.this: Creation complete after 31s [id=i-0940961b621400d23]

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

Outputs:

ipv4 = "18.118.195.146"
ubuntu@ip-172-31-35-160:~/assignment5$
```