Module 8 – Terraform

Assignment-2

1. Destroy the previous deployment

2. Create a new EC2 instance with an Elastic IP

- Destoyed the previous deployment with terraform destroy command

```
ubuntu@ip-172-31-35-160:~/assignment1$ terraform destroy
aws_instance.this: Refreshing state... [id=i-0def6e8048b000bdc]

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-0def6e8048b000bdc" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "us-east-2a" -> 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
      - get_password_data                    = false -> null

Plan: 0 to add, 0 to change, 1 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-0def6e8048b000bdc]
aws_instance.this: Still destroying... [id=i-0def6e8048b000bdc, 10s elapsed]
aws_instance.this: Still destroying... [id=i-0def6e8048b000bdc, 20s elapsed]
aws_instance.this: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
ubuntu@ip-172-31-35-160:~/assignment1$
```

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

```
ubuntu@ip-172-31-35-160:~$ mkdir assignment2
ubuntu@ip-172-31-35-160:~$ ls
assign1  assignment1  assignment2
ubuntu@ip-172-31-35-160:~$ cd assignment2
ubuntu@ip-172-31-35-160:~/assignment2$ ls
ubuntu@ip-172-31-35-160:~/assignment2$ vi main.tf
```

- In the hcl script added aws_eip block to create a elastic Ip and aws_eip_association block to attach same to the Instance to be created

```
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"
        tags = {
                Name = "Assignment1"
        }
}
resource "aws_eip" "elastic-ip" {
        domain   = "vpc"
}
resource "aws_eip_association" "eip_assoc" {
  instance_id   = aws_instance.this.id
  allocation_id = aws_eip.elastic-ip.id
}
"main.tf" 20L, 548B
```

Initialised terraform

```
ubuntu@ip-172-31-35-160:~/assignment2$ 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.

If you ever set or change modules or backend configuration for Terraform,
```

Performed terraform plan

```
ubuntu@ip-172-31-35-160:~/assignment2$ 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_eip.elastic-ip will be created
  + resource "aws_eip" "elastic-ip" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = "vpc"
      + id                   = (known after apply)
      + instance             = (known after apply)
```

Applied changes with terraform aplly command

```
ubuntu@ip-172-31-35-160:~/assignment2$ 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_eip.elastic-ip will be created
  + resource "aws_eip" "elastic-ip" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = "vpc"
      + id                   = (known after apply)
```

```
Plan: 3 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.this: Creating...
aws_eip.elastic-ip: Creating...
aws_eip.elastic-ip: Creation complete after 1s [id=eipalloc-0405367f703365ec6]
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 32s [id=i-0ea293436d0485e24]
aws_eip_association.eip_assoc: Creating...
aws_eip_association.eip_assoc: Creation complete after 1s [id=eipassoc-01034a7083bfe5e45]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-35-160:~/assignment2$
```