System Provisioning and Configuration Management

Submitted by:

NAME: Tanishqua Tellakula
SAP ID: 500084269

Roll number: R120220007

Batch: B1(Hons)

EXPERIMENT 11

Creating a VPC in Terraform

Objective

Objective:

Learn how to use Terraform to create a basic Virtual Private Cloud (VPC) in AWS.

Prerequisites:

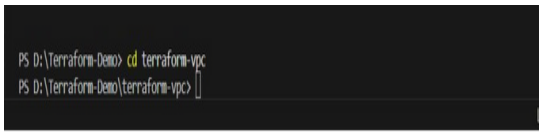
· *Terraform installed on your machine.*

· *AWS CLI configured with the necessary credentials.*

Steps:

1. Create a Terraform Directory:

```
mkdir terraform-vpc
cd terraform-vpc
```



· *Create Terraform Configuration Files:*

· *Create a file named main.tf:*

 # vpc.tf

```
resource "aws_vpc" "gfg-vpc" {
 cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "gfg-subnet" {
 vpc_id = aws_vpc.gfg-vpc.id
 cidr_block = "10.0.1.0/24"
```

```
 tags = {
Name = "gfg-subnet"
}
}

resource "aws_internet_gateway" "gfg-gw" {
vpc_id = aws_vpc.gfg-vpc.id

 tags = {
Name = "gfg-IG"
}
}

resource "aws_route_table" "gfg-rt" {
vpc_id = aws_vpc.gfg-vpc.id

 route {
cidr_block = "0.0.0.0/0"
gateway_id = aws_internet_gateway.gfg-gw.id
}

   tags = {
Name = "GFG-Route-Table"
}
}

resource "aws_route_table_association" "gfg-rta" {
subnet_id      = aws_subnet.gfg-subnet.id
route_table_id = aws_route_table.gfg-rt.id
```

```
}

resource "aws_security_group" "gfg-sg" {
name        = "my-gfg-sg"
 vpc_id      = aws_vpc.gfg-vpc.id

 ingress {
description = "TLS from VPC"
from_port         = 20
to_port           = 20
protocol          = "tcp"
cidr_blocks       = ["0.0.0.0/0"]
ipv6_cidr_blocks = ["::/0"]
}

 egress {
from_port         = 0
to_port           = 0
protocol          = "-1"
cidr_blocks       = ["0.0.0.0/0"]
ipv6_cidr_blocks = ["::/0"]
}

 tags = {
Name = "my-gfg-sg"
}
}
```

In this configuration, we define an AWS provider, a VPC with a specified CIDR block, and two subnets within the VPC.

```
terraform-vpc > vpc.tf > ...
resource "aws_vpc" "gfg-vpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "gfg-subnet" {
  vpc_id     = aws_vpc.gfg-vpc.id
  cidr_block = "10.0.1.0/24"

  tags = {
    Name = "gfg-subnet"
  }
}

resource "aws_internet_gateway" "gfg-gw" {
  vpc_id = aws_vpc.gfg-vpc.id

  tags = {
    Name = "gfg-IG"
  }
}

resource "aws_route_table" "gfg-rt" {
  vpc_id = aws_vpc.gfg-vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gfg-gw.id
  }
```

2. Initialize and Apply:

· Run the following Terraform commands to initialize and apply the configuration:

terraform init

terraform apply

· Terraform will prompt you to confirm the creation of the VPC and subnets. Type yes and press Enter.

```
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.
```

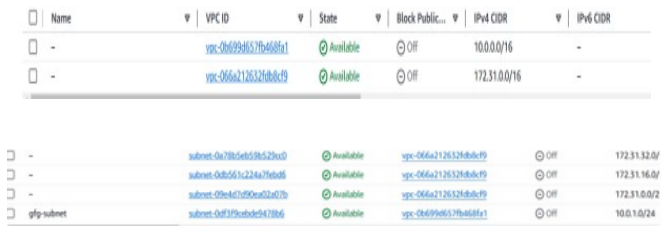
```
  Enter a value: yes

aws_vpc.gfg-vpc: Creating...
aws_vpc.gfg-vpc: Creation complete after 1s [id=vp
aws_internet_gateway.gfg-gw: Creating...
aws_subnet.gfg-subnet: Creating...
aws_security_group.gfg-sg: Creating...
aws_internet_gateway.gfg-gw: Creation complete aft
aws_route_table.gfg-rt: Creating...
aws_subnet.gfg-subnet: Creation complete after 1s
```

3. Verify Resources in AWS Console:

· Log in to the AWS Management Console and navigate to the VPC service.

· *Verify that the VPC and subnets with the specified names and settings have been created.*



4. Update VPC Configuration:

· *If you want to modify the VPC configuration, update the main.tf file with the desired changes.*

· *Rerun the terraform apply command to apply the changes:*

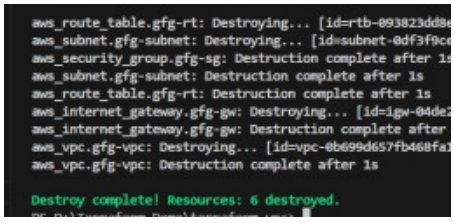
```
terraform apply
```

5. Clean Up:

After testing, you can clean up the VPC and subnets:

```
terraform destroy
```

Confirm the destruction by typing yes.



6. Conclusion:

This lab exercise demonstrates how to create a basic Virtual Private Cloud (VPC) with subnets in AWS using Terraform. The example includes

a simple VPC configuration with two subnets. Experiment with different CIDR blocks, settings, and additional AWS resources to customize your VPC.