Lab Exercise 11– Creating a VPC in Terraform

Aditya Tomar

500106015

R2142221060

Batch-2(DevOps)

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
```

```
[adityatomar@Adityas-MacBook-Air-3 terraform-iam-users % cd
[adityatomar@Adityas-MacBook-Air-3 ~ % mkdir terraform-vpc

[adityatomar@Adityas-MacBook-Air-3 ~ % cd terraform-vpc

 adityatomar@Adityas-MacBook-Air-3 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"
 }
}
```

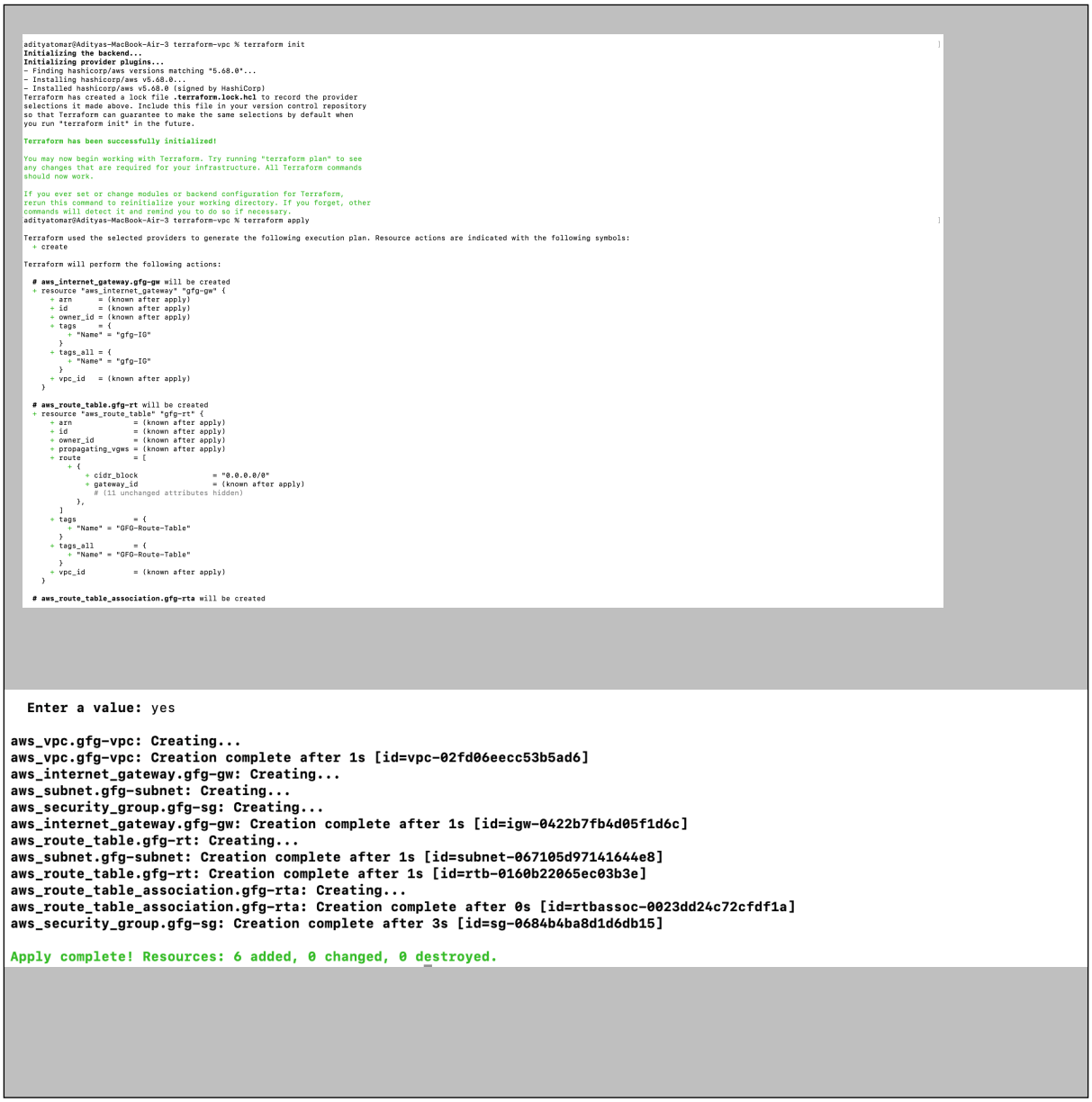
```
resource "aws_vpc" "gfg-vpc" {
  cidr_block = "10.0.0.0/16"
}

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"]
  }
```

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

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.

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
```

```
  Enter a value: yes

aws_route_table_association.gfg-rta: Destroying... [id=rtbassoc-0023dd24c72cfdf1a]
aws_security_group.gfg-sg: Destroying... [id=sg-0684b4ba8d1d6db15]
aws_route_table_association.gfg-rta: Destruction complete after 0s
aws_subnet.gfg-subnet: Destroying... [id=subnet-067105d97141644e8]
aws_route_table.gfg-rt: Destroying... [id=rtb-0160b22065ec03b3e]
aws_security_group.gfg-sg: Destruction complete after 1s
aws_subnet.gfg-subnet: Destruction complete after 1s
aws_route_table.gfg-rt: Destruction complete after 1s
aws_internet_gateway.gfg-gw: Destroying... [id=igw-0422b7fb4d05f1d6c]
aws_internet_gateway.gfg-gw: Destruction complete after 1s
aws_vpc.gfg-vpc: Destroying... [id=vpc-02fd06eecc53b5ad6]
aws_vpc.gfg-vpc: Destruction complete after 0s

Destroy complete! Resources: 6 destroyed.
```

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.