ANSHIKA SRIVASTAVA
ROLL NUMBER – R2142220907
SAP ID – 500107049

LAB EXERCISE 11

Lab Exercise 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

```
PS C:\D content backup\Academics\SPCM Lab\terraform-iam-users> cd ..
PS C:\D content backup\Academics\SPCM Lab> mkdir terraform-vpc


    Directory: C:\D content backup\Academics\SPCM Lab


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        28-02-2025     22:35                terraform-vpc


PS C:\D content backup\Academics\SPCM Lab> cd terraform-vpc
PS C:\D content backup\Academics\SPCM Lab\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"
```

```
}
}
```

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"
  }
}
```

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:

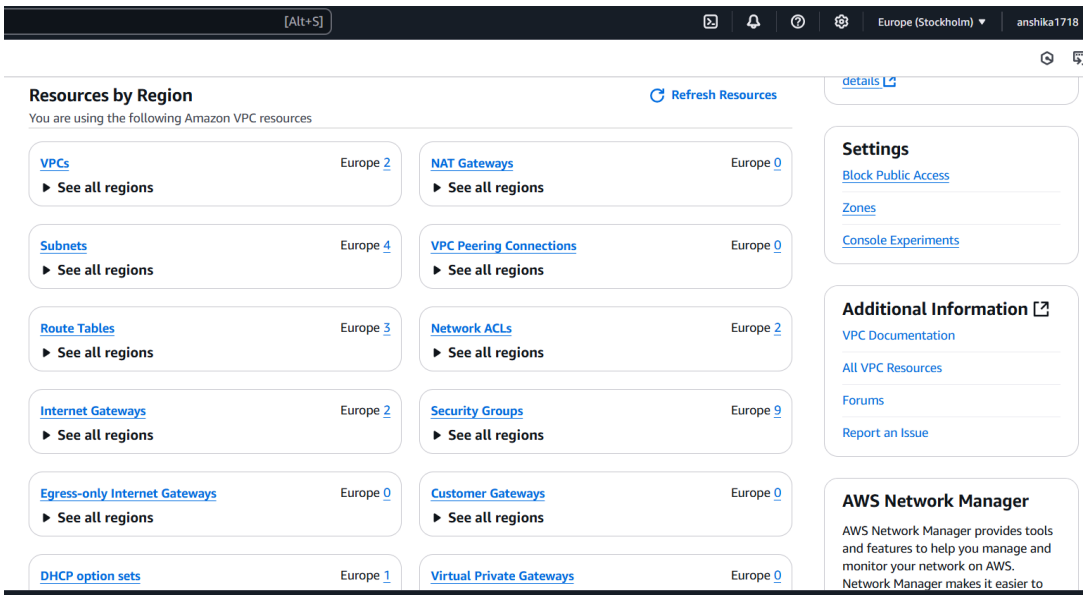
```
terraform init
terraform apply
```

```
OUTPUT   PROBLEMS   DEBUG CONSOLE   TERMINAL   PORTS

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_vpc.gfg-vpc: Creating...
aws_vpc.gfg-vpc: Creation complete after 4s [id=vpc-02ba51f85d2aa5b5c]
aws_internet_gateway.gfg-gw: Creating...
aws_subnet.gfg-subnet: Creating...
aws_security_group.gfg-sg: Creating...
aws_internet_gateway.gfg-gw: Creation complete after 1s [id=igw-088bd54d7f218cef7]
aws_route_table.gfg-rt: Creating...
aws_subnet.gfg-subnet: Creation complete after 1s [id=subnet-04ba3556ea9499e0c]
aws_route_table.gfg-rt: Creation complete after 2s [id=rtb-0e1aa382fba5c79c5]
aws_route_table_association.gfg-rta: Creating...
aws_route_table_association.gfg-rta: Creation complete after 1s [id=rtbassoc-082a863957cdcc8ba]
aws_security_group.gfg-sg: Creation complete after 5s [id=sg-03b0f68adb0a9b27d]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
PS C:\D content backup\Academics\SPCM Lab\terraform-vpc>
```

- 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:

5. Clean Up:

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

terraform 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_route_table_association.gfg-rta: Destroying... [id=rtbassoc-082a863957cdc
aws_security_group.gfg-sg: Destroying... [id=sg-03b0f68adb0a9b27d]
aws_route_table_association.gfg-rta: Destruction complete after 1s
aws_subnet.gfg-subnet: Destroying... [id=subnet-04ba3556ea9499e0c]
aws_route_table.gfg-rt: Destroying... [id=rtb-0e1aa382fba5c79c5]
aws_security_group.gfg-sg: Destruction complete after 2s
aws_route_table.gfg-rt: Destruction complete after 1s
aws_internet_gateway.gfg-gw: Destroying... [id=igw-088bd54d7f218cef7]
aws_subnet.gfg-subnet: Destruction complete after 1s
aws_internet_gateway.gfg-gw: Destruction complete after 1s
aws_vpc.gfg-vpc: Destroying... [id=vpc-02ba51f85d2aa5b5c]
aws_vpc.gfg-vpc: Destruction complete after 1s

Destroy complete! Resources: 6 destroyed.
PS C:\D content backup\Academics\SPCM Lab\terraform-vpc>
```

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.