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:

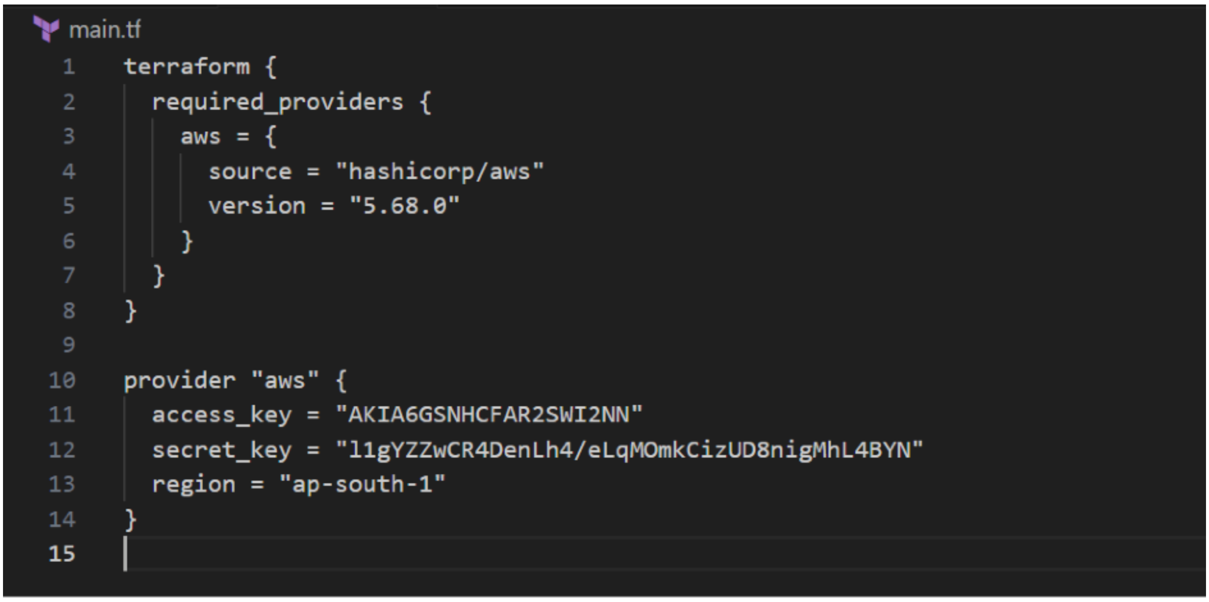
- Terraform installed on your machine.
- AWS CLI configured with the necessary credentials.

Steps:

1. Create a Terraform Directory:

```
mkdir exp11-vpc
cd exp11-vpc
```

- Create Terraform Configuration Files:
- Create a file named main.tf:



```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.68.0"
    }
  }
}

provider "aws" {
  access_key = "AKIA6GSNHCFAR2SWI2NN"
  secret_key = "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"
  region = "ap-south-1"
}
```

# 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"
```

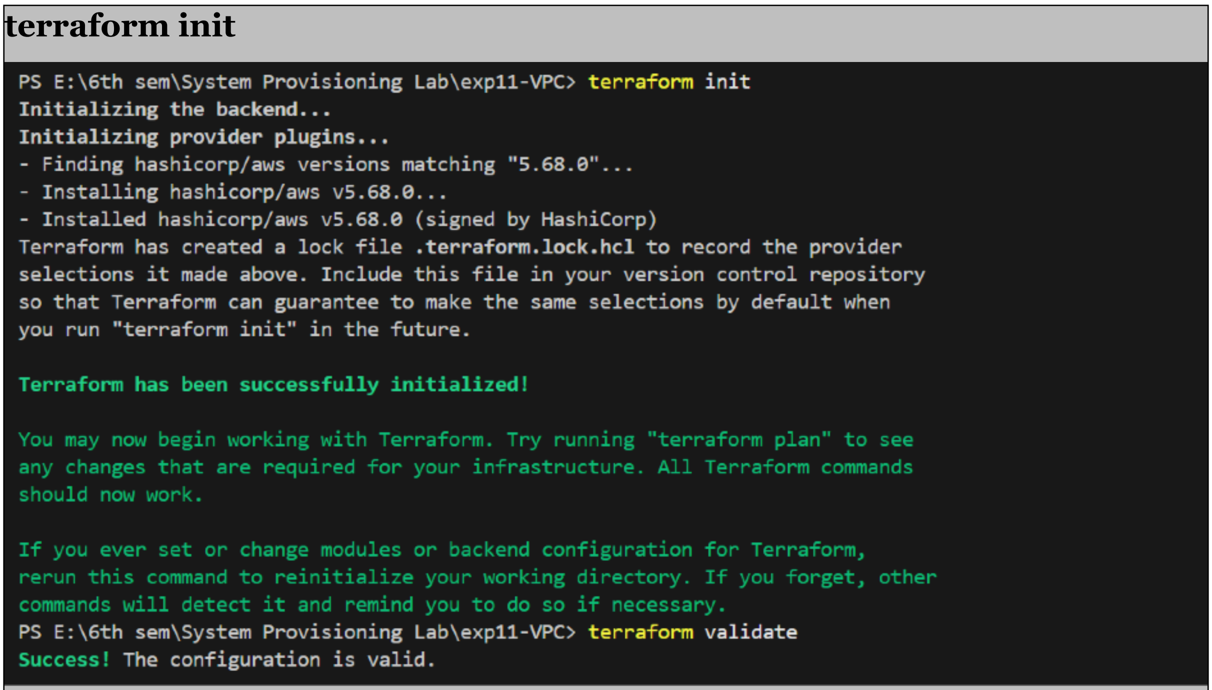
```
 }
}
```

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

2. Initialize and Apply:

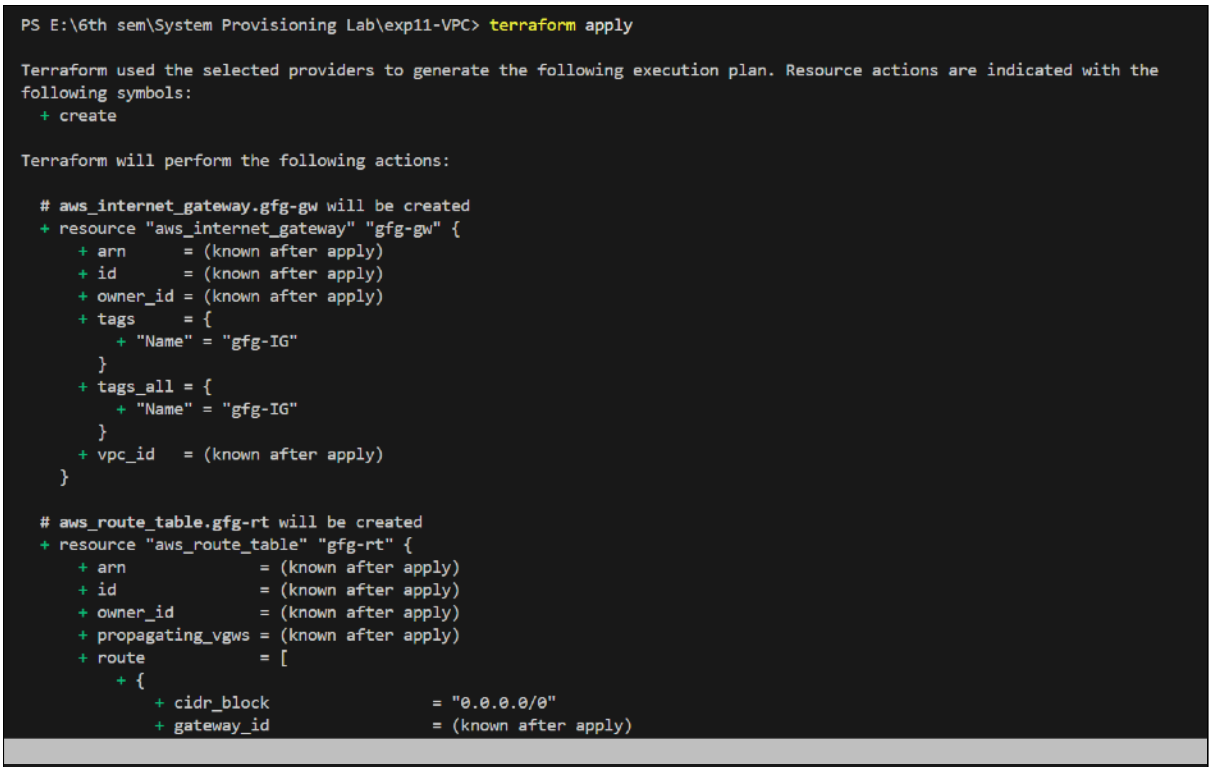
- Run the following Terraform commands to initialize and apply the configuration:

```
terraform init
```

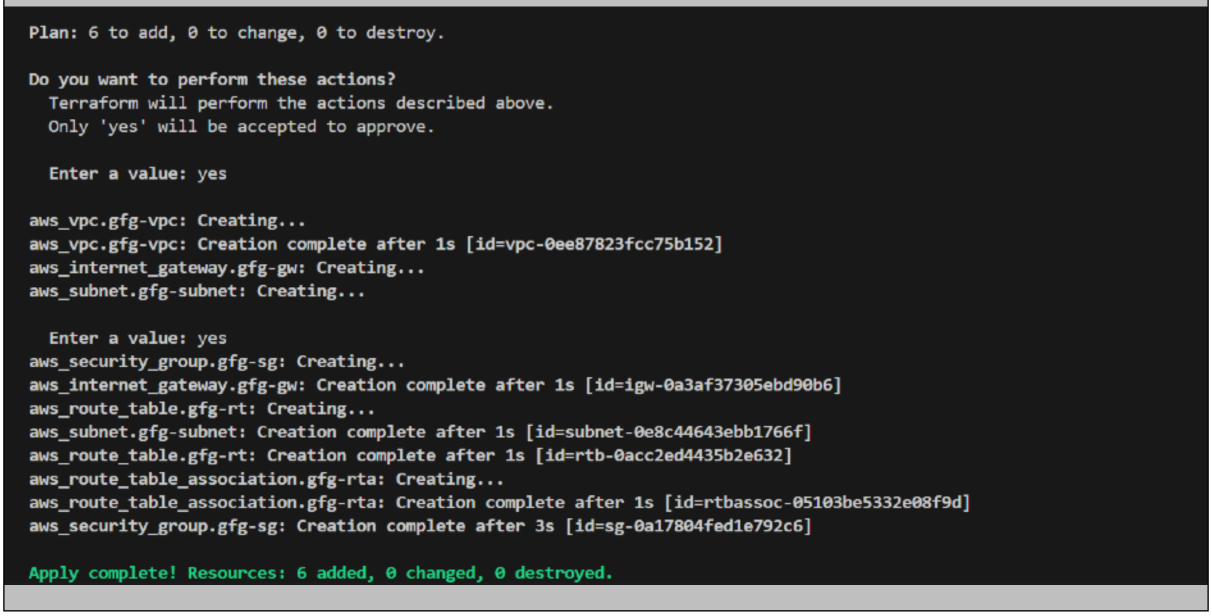


```
PS E:\6th sem\System Provisioning Lab\exp11-VPC> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.68.0"...
- Installing hashicorp/aws v5.68.0...
- Installed hashicorp/aws v5.68.0 (signed by HashiCorp)
Terraform has created a 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,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS E:\6th sem\System Provisioning Lab\exp11-VPC> terraform validate
Success! The configuration is valid.
```

```
terraform apply
```



```
PS E:\6th sem\System Provisioning Lab\exp11-VPC> 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_internet_gateway.gfg-gw will be created
  + resource "aws_internet_gateway" "gfg-gw" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "gfg-IG"
        }
      + tags_all = {
          + "Name" = "gfg-IG"
        }
      + vpc_id   = (known after apply)
    }

  # aws_route_table.gfg-rt will be created
  + resource "aws_route_table" "gfg-rt" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = [
          + {
              + cidr_block                 = "0.0.0.0/0"
              + gateway_id                 = (known after apply)
```



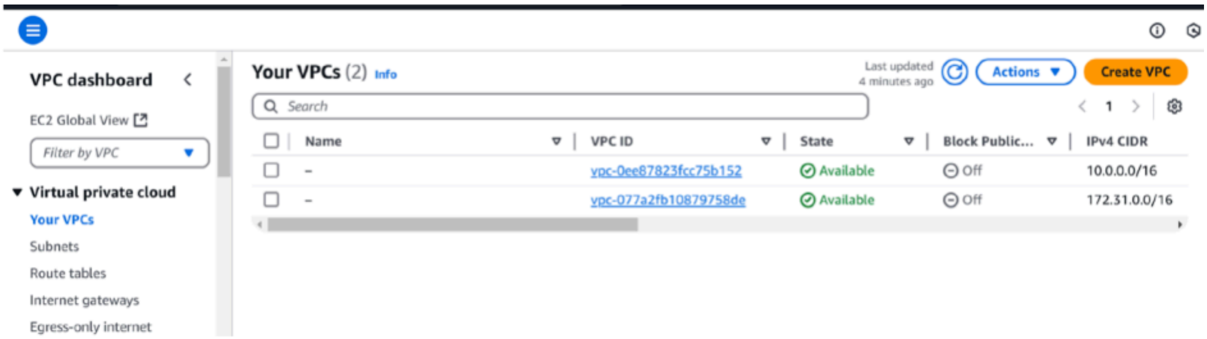
```
Plan: 6 to add, 0 to change, 0 to destroy.

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_vpc.gfg-vpc: Creating...
aws_vpc.gfg-vpc: Creation complete after 1s [id=vpc-0ee87823fcc75b152]
aws_internet_gateway.gfg-gw: Creating...
aws_subnet.gfg-subnet: Creating...

  Enter a value: yes
aws_security_group.gfg-sg: Creating...
aws_internet_gateway.gfg-gw: Creation complete after 1s [id=igw-0a3af37305ebd90b6]
aws_route_table.gfg-rt: Creating...
aws_subnet.gfg-subnet: Creation complete after 1s [id=subnet-0e8c44643ebb1766f]
aws_route_table.gfg-rt: Creation complete after 1s [id=rtb-0acc2ed4435b2e632]
aws_route_table_association.gfg-rta: Creating...
aws_route_table_association.gfg-rta: Creation complete after 1s [id=rtbassoc-05103be5332e08f9d]
aws_security_group.gfg-sg: Creation complete after 3s [id=sg-0a17804fed1e792c6]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
```

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

3. Verify Resources in AWS Console:

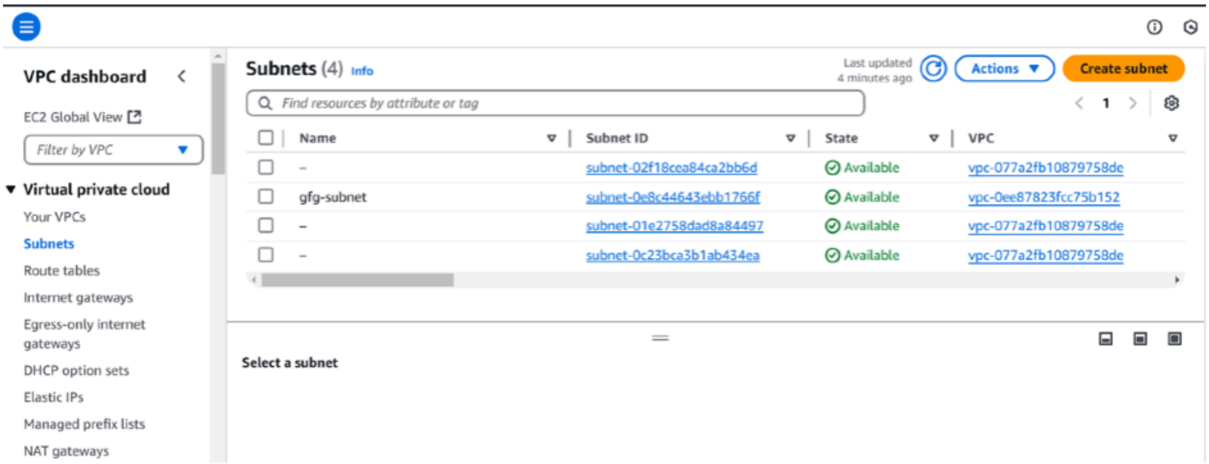
- Log in to the AWS Management Console and navigate to the VPC service.

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



Your VPCs (2) Info

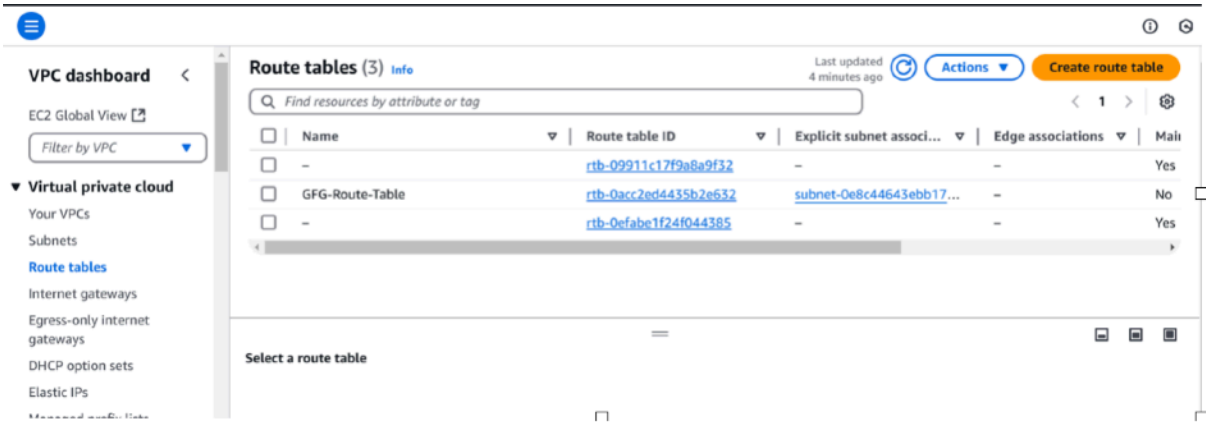
☐	Name	VPC ID	State	Block Public...	IPv4 CIDR
☐	-	vpc-0ee87823fcc75b152	Available	Off	10.0.0.0/16
☐	-	vpc-077a2fb10879758de	Available	Off	172.31.0.0/16



Subnets (4) Info

☐	Name	Subnet ID	State	VPC
☐	-	subnet-02f18cea84ca2bb6d	Available	vpc-077a2fb10879758de
☐	gfg-subnet	subnet-0e8c44643ebb1766f	Available	vpc-0ee87823fcc75b152
☐	-	subnet-01e2758dad8a84497	Available	vpc-077a2fb10879758de
☐	-	subnet-0c23bca3b1ab434ea	Available	vpc-077a2fb10879758de

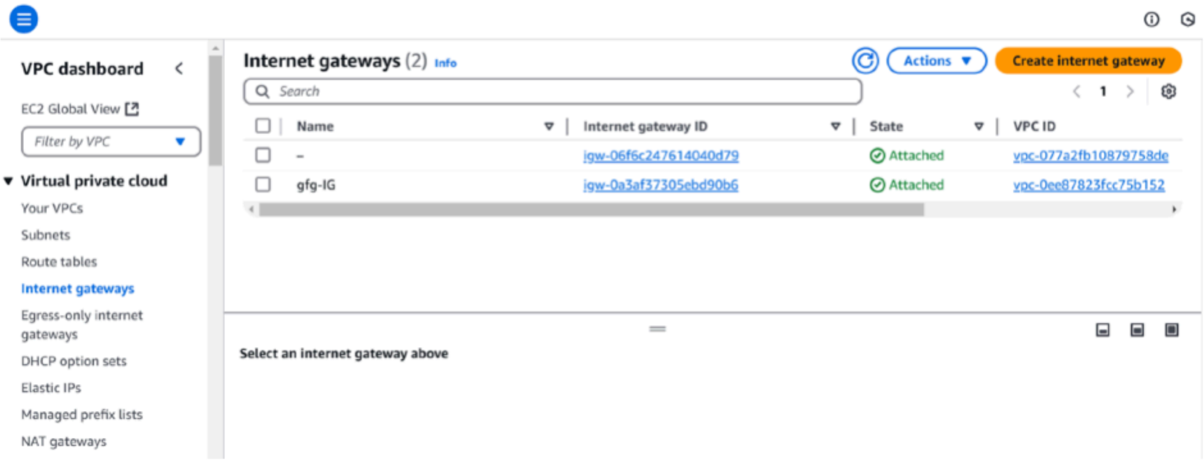
Select a subnet



Route tables (3) Info

☐	Name	Route table ID	Explicit subnet associ...	Edge associations	Mai
☐	-	rtb-09911c17f9a8a9f32	-	-	Yes
☐	GFG-Route-Table	rtb-0acc2ed4435b2e632	subnet-0e8c44643ebb17...	-	No
☐	-	rtb-0efabe1f24f044385	-	-	Yes

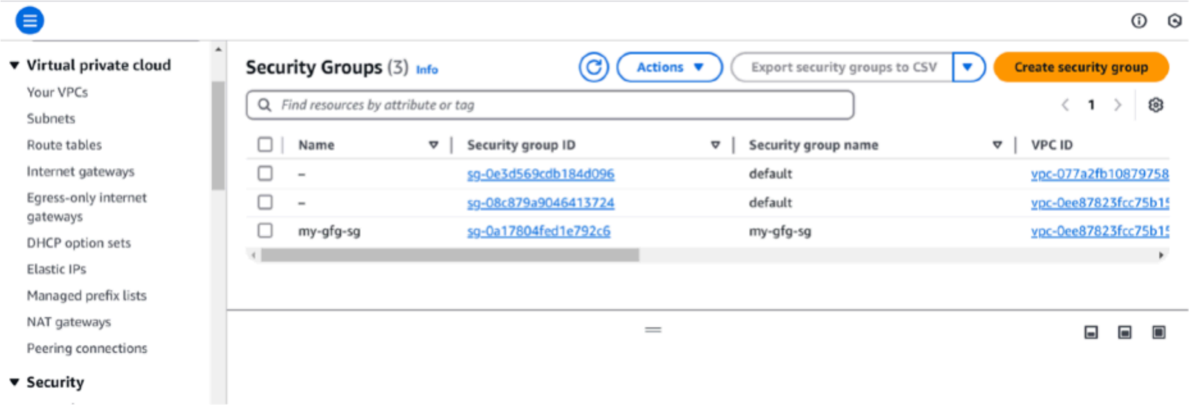
Select a route table



Internet gateways (2) Info

□	Name	Internet gateway ID	State	VPC ID
□	-	igw-06f6c247614040d79	Attached	vpc-077a2fb10879758de
□	gfg-IG	igw-0a3af37305ebd90b6	Attached	vpc-0ee87823fcc75b152

Select an internet gateway above



Security Groups (3) Info

□	Name	Security group ID	Security group name	VPC ID
□	-	sg-0e3d569cdb184d096	default	vpc-077a2fb10879758
□	-	sg-08c879a9046413724	default	vpc-0ee87823fcc75b1!
□	my-gfg-sg	sg-0a17804fed1e792c6	my-gfg-sg	vpc-0ee87823fcc75b1!

4. Update VPC Configuration:

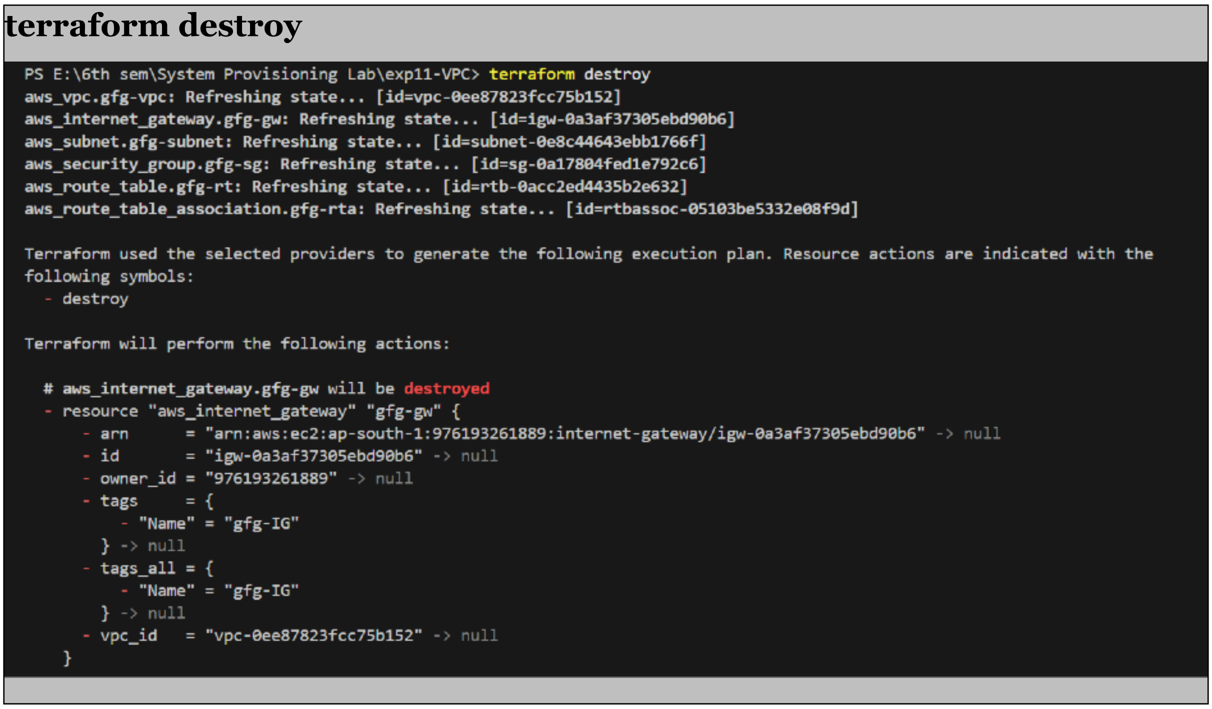
- If you want to modify the VPC configuration, update the main.tf file with the desired changes.
- Rerun the terraform apply command to apply the changes:

```
terraform apply
```

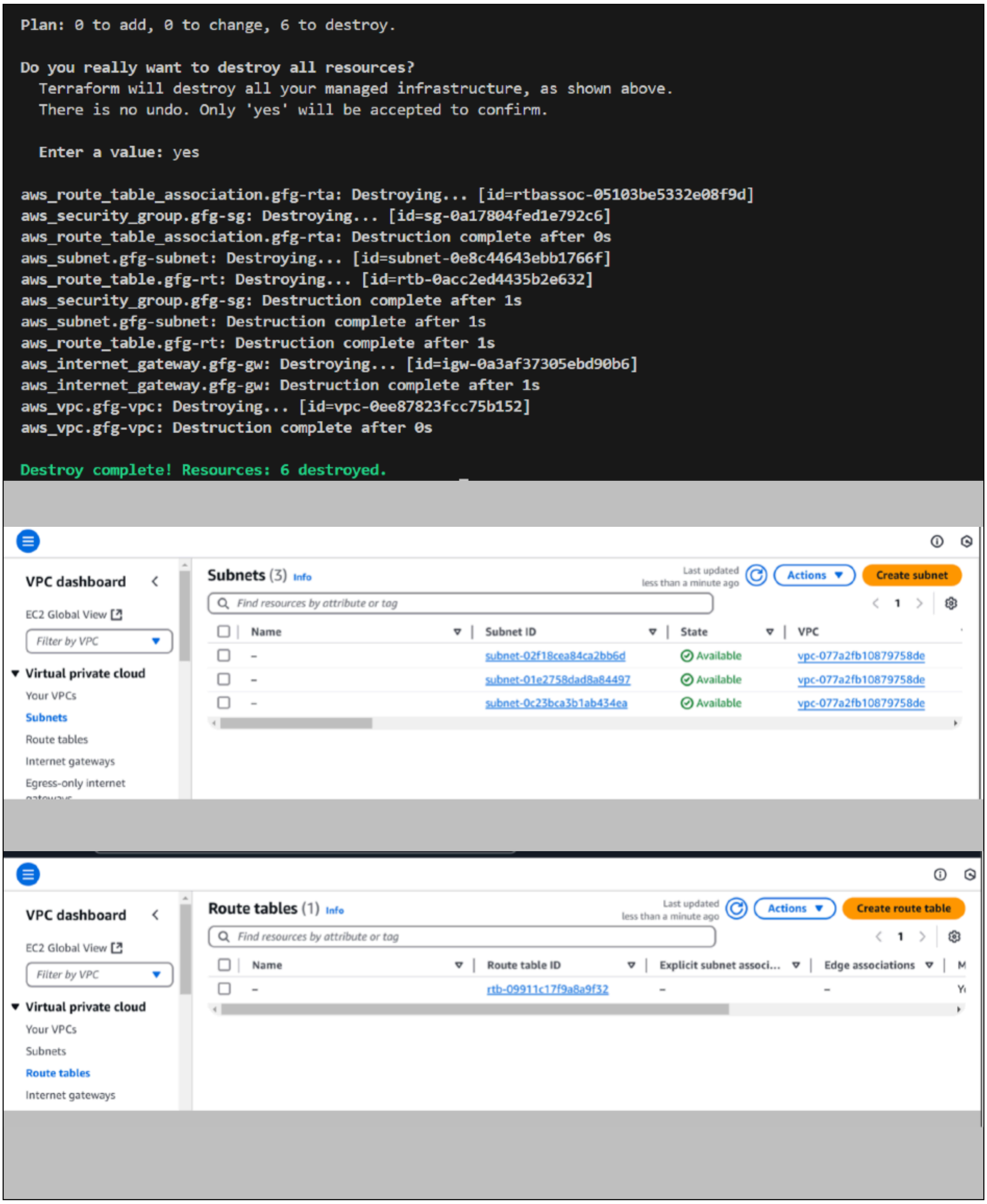
5. Clean Up:

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

```
terraform destroy
```



```
PS E:\6th sem\System Provisioning Lab\exp11-VPC> terraform destroy
aws_vpc.gfg-vpc: Refreshing state... [id=vpc-0ee87823fcc75b152]
aws_internet_gateway.gfg-gw: Refreshing state... [id=igw-0a3af37305ebd90b6]
aws_subnet.gfg-subnet: Refreshing state... [id=subnet-0e8c44643ebb1766f]
aws_security_group.gfg-sg: Refreshing state... [id=sg-0a17804fed1e792c6]
aws_route_table.gfg-rt: Refreshing state... [id=rtb-0acc2ed4435b2e632]
aws_route_table_association.gfg-rta: Refreshing state... [id=rtbassoc-05103be5332e08f9d]

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_internet_gateway.gfg-gw will be destroyed
  - resource "aws_internet_gateway" "gfg-gw" {
      - arn      = "arn:aws:ec2:ap-south-1:976193261889:internet-gateway/igw-0a3af37305ebd90b6" -> null
      - id       = "igw-0a3af37305ebd90b6" -> null
      - owner_id = "976193261889" -> null
      - tags     = {
          - "Name" = "gfg-IG"
        } -> null
      - tags_all = {
          - "Name" = "gfg-IG"
        } -> null
      - vpc_id   = "vpc-0ee87823fcc75b152" -> null
    }
```



```
Plan: 0 to add, 0 to change, 6 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_route_table_association.gfg-rta: Destroying... [id=rtbassoc-05103be5332e08f9d]
aws_security_group.gfg-sg: Destroying... [id=sg-0a17804fed1e792c6]
aws_route_table_association.gfg-rta: Destruction complete after 0s
aws_subnet.gfg-subnet: Destroying... [id=subnet-0e8c44643ebb1766f]
aws_route_table.gfg-rt: Destroying... [id=rtb-0acc2ed4435b2e632]
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-0a3af37305ebd90b6]
aws_internet_gateway.gfg-gw: Destruction complete after 1s
aws_vpc.gfg-vpc: Destroying... [id=vpc-0ee87823fcc75b152]
aws_vpc.gfg-vpc: Destruction complete after 0s

Destroy complete! Resources: 6 destroyed.
```

☐	Name	Subnet ID	State	VPC
☐	-	subnet-02f18cea84ca2bb6d	✓ Available	vpc-077a2fb10879758de
☐	-	subnet-01e2758dad8a84497	✓ Available	vpc-077a2fb10879758de
☐	-	subnet-0c23bca3b1ab434ea	✓ Available	vpc-077a2fb10879758de

☐	Name	Route table ID	Explicit subnet associ...	Edge associations	M
☐	-	rtb-09911c17f9a8a9f32	-	-	Y

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.