SCHOOL OF COMPUTER SCIENCE

System Provisionig and Configuration Management Lab File

Submitted by

Name	**Kashish**
Branch	**BTech CSE(DevOps)B-1(NH)**
Semester	**6**
SAPID	**500107137**
Roll no	**R2142220335**

Lab Exercise 11– Creating a VPC in Terraform

Objective:

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

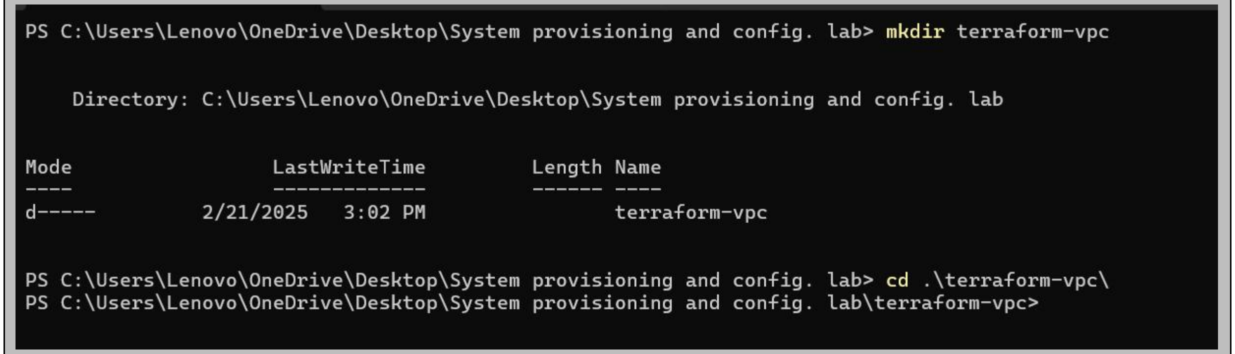
Prerequisites:

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

Steps:

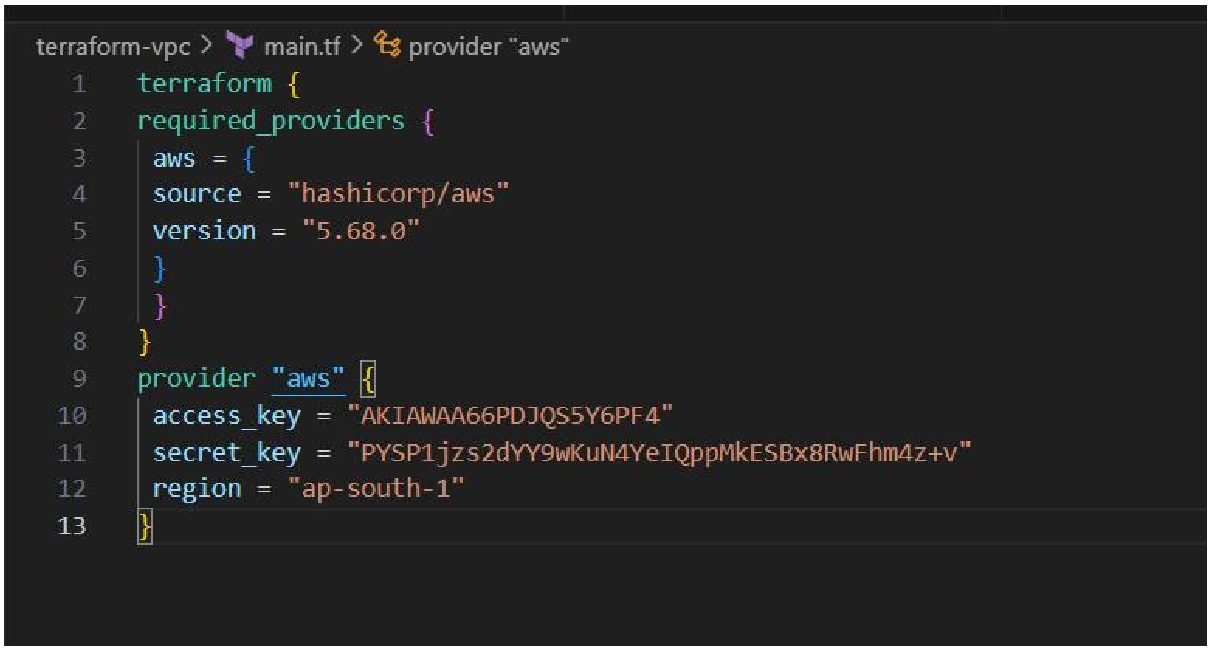
1. Create a Terraform Directory:

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



```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab> mkdir terraform-vpc


    Directory: C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2/21/2025   3:02 PM                terraform-vpc


PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab> cd .\terraform-vpc\
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\terraform-vpc>
```

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

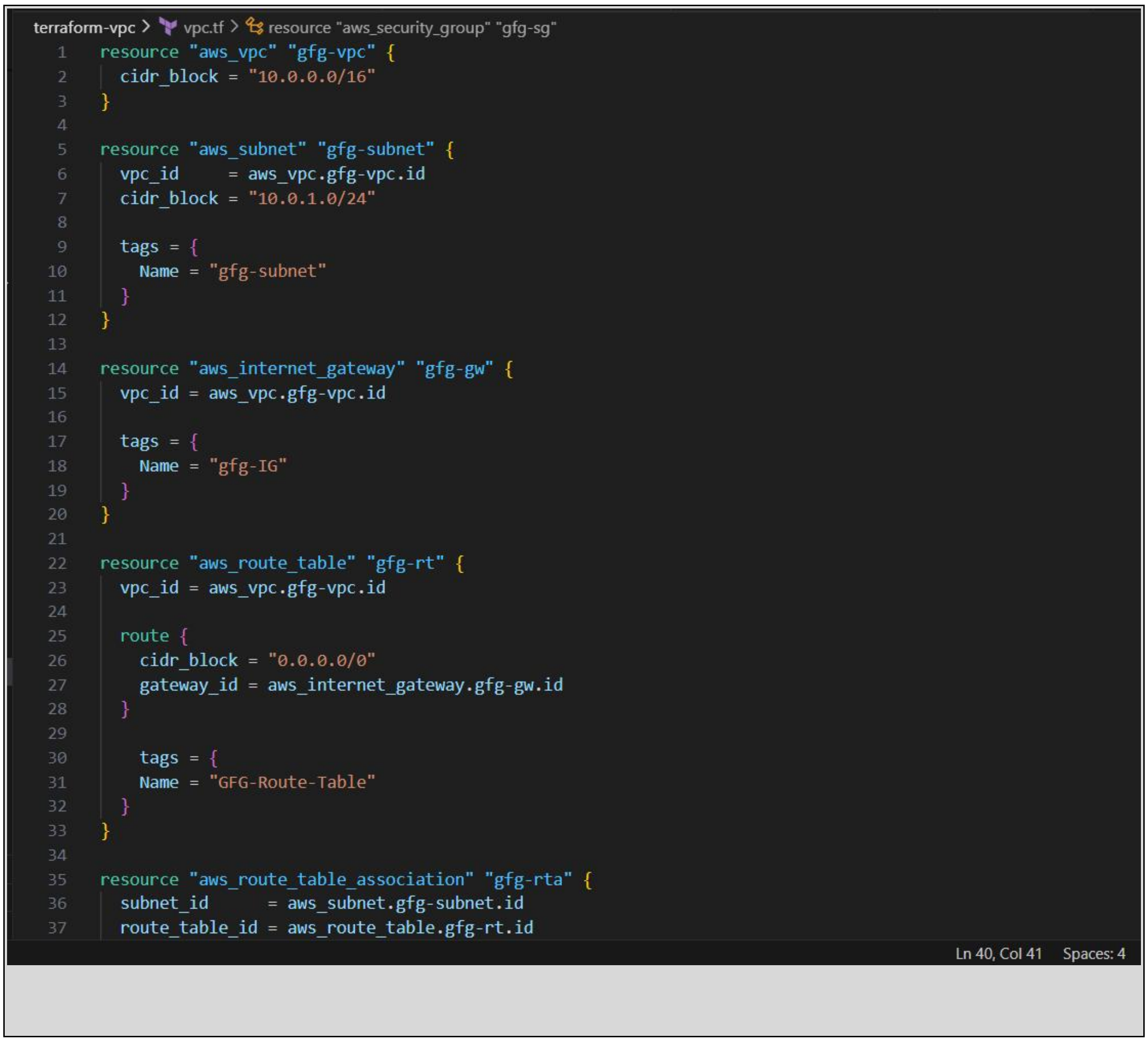


terraform-vpc > main.tf > provider "aws"

```
terraform {
required_providers {
 aws = {
 source = "hashicorp/aws"
 version = "5.68.0"
 }
 }
}
provider "aws" {
 access_key = "AKIAWAA66PDJQS5Y6PF4"
 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"
  }
}
```



terraform-vpc > vpc.tf > resource "aws_security_group" "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
```

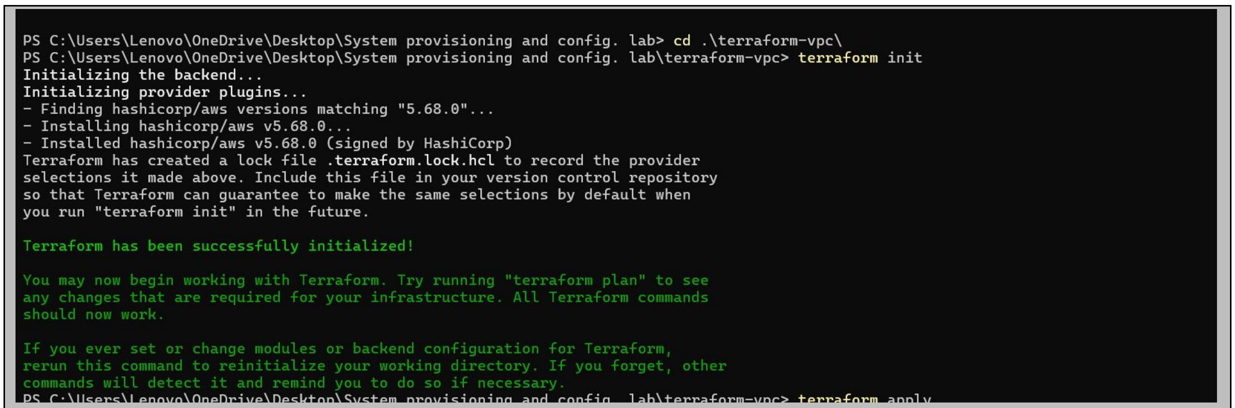
Ln 40, Col 41   Spaces: 4

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:

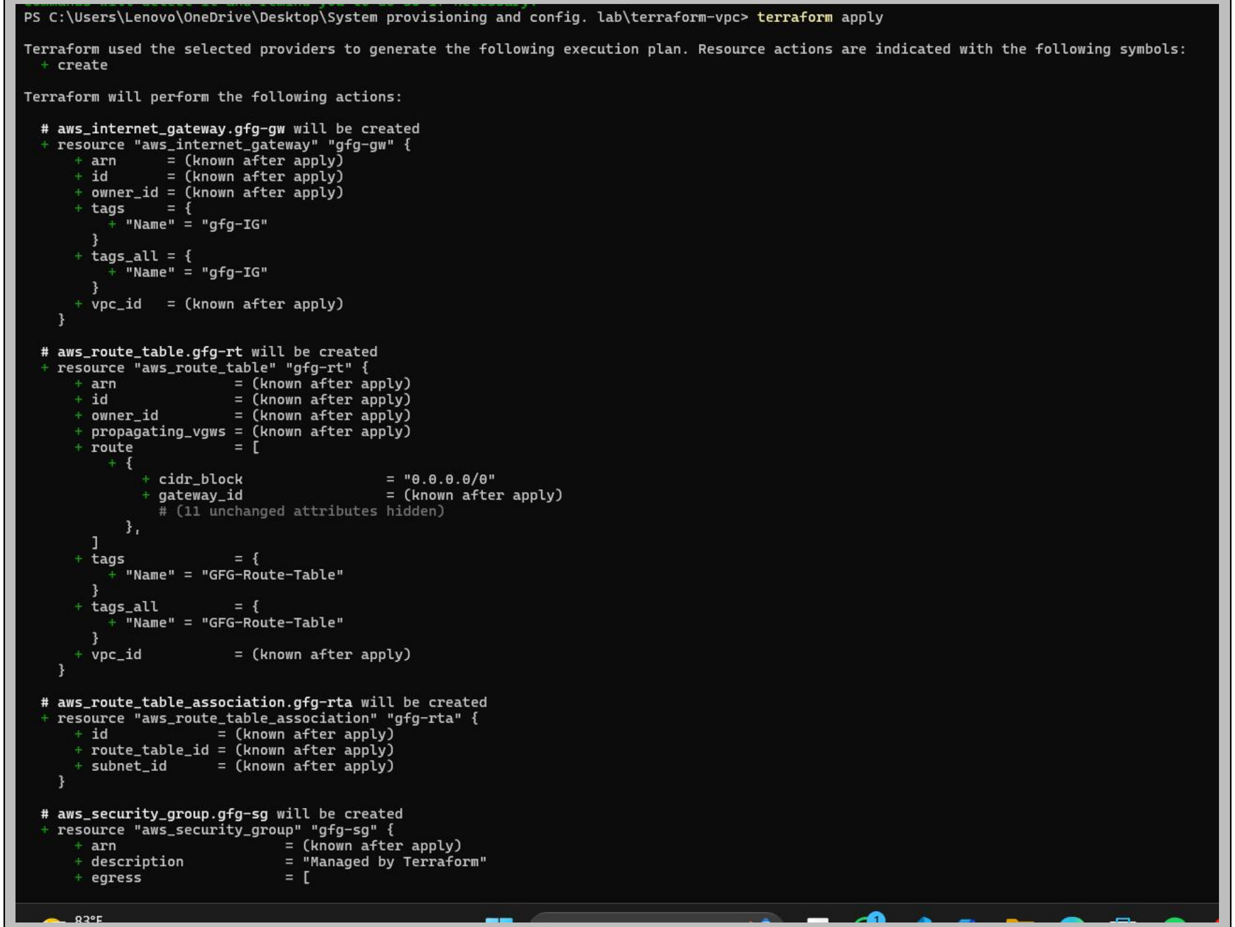
- Run the following Terraform commands to initialize and apply the configuration:

```
terraform init
```

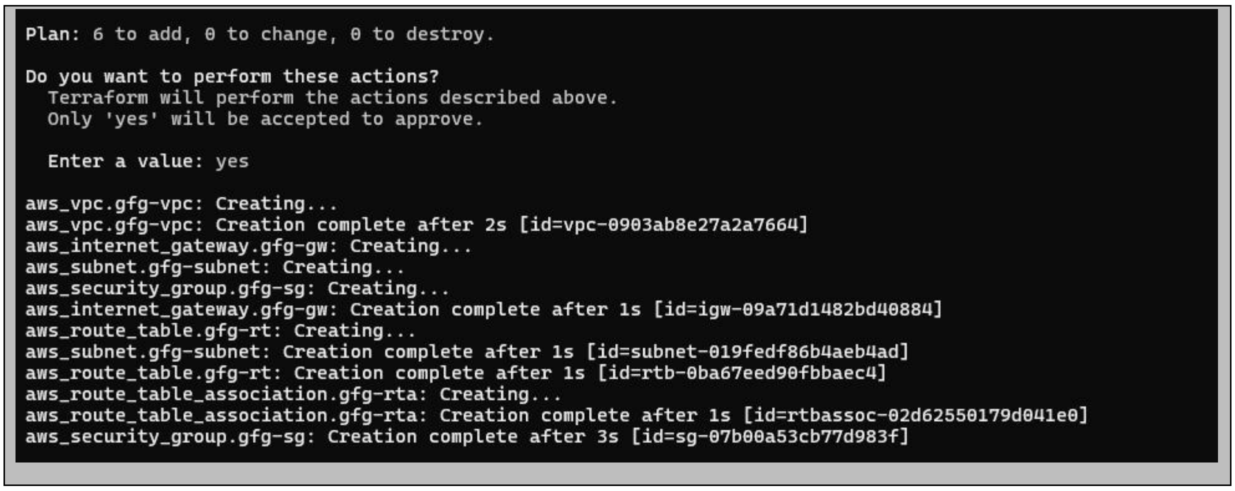


```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab> cd .\terraform-vpc\
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\terraform-vpc> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "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 C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\terraform-vpc> terraform apply
```

terraform apply



```
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\terraform-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)
                # (11 unchanged attributes hidden)
            },
        ]
      + tags             = {
          + "Name" = "GFG-Route-Table"
        }
      + tags_all         = {
          + "Name" = "GFG-Route-Table"
        }
      + vpc_id           = (known after apply)
    }

  # aws_route_table_association.gfg-rta will be created
  + resource "aws_route_table_association" "gfg-rta" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = (known after apply)
    }

  # aws_security_group.gfg-sg will be created
  + resource "aws_security_group" "gfg-sg" {
      + arn                    = (known after apply)
      + description            = "Managed by Terraform"
      + egress                 = [
```

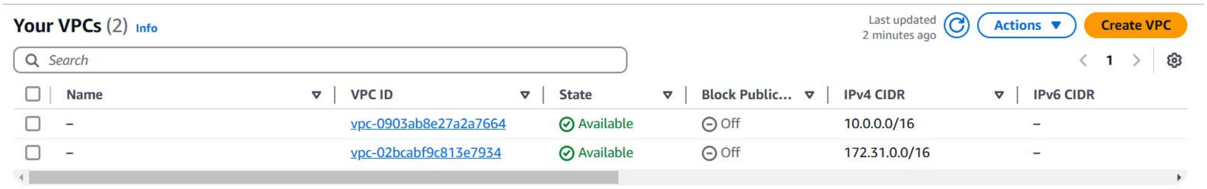


```
Plan: 6 to add, 0 to change, 0 to destroy.

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 2s [id=vpc-0903ab8e27a2a7664]
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-09a71d1482bd40884]
aws_route_table.gfg-rt: Creating...
aws_subnet.gfg-subnet: Creation complete after 1s [id=subnet-019fedf86b4aeb4ad]
aws_route_table.gfg-rt: Creation complete after 1s [id=rtb-0ba67eed90fbbaec4]
aws_route_table_association.gfg-rta: Creating...
aws_route_table_association.gfg-rta: Creation complete after 1s [id=rtbassoc-02d62550179d041e0]
aws_security_group.gfg-sg: Creation complete after 3s [id=sg-07b00a53cb77d983f]
```

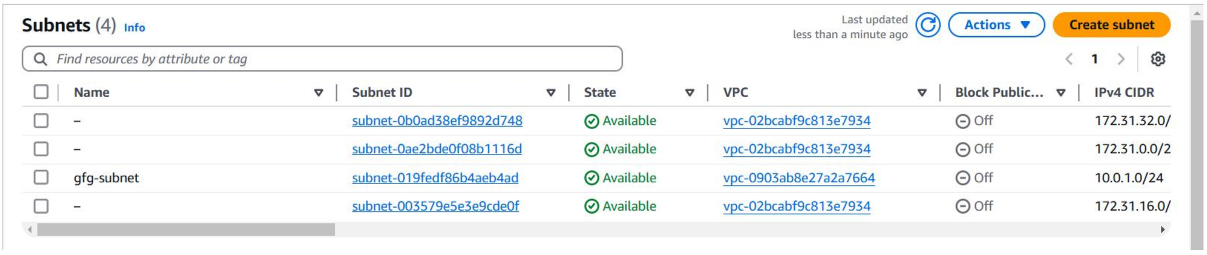
- Terraform will prompt you to confirm the creation of the VPC and subnets. Type yes and press Enter.

3. Verify Resources in AWS Console:

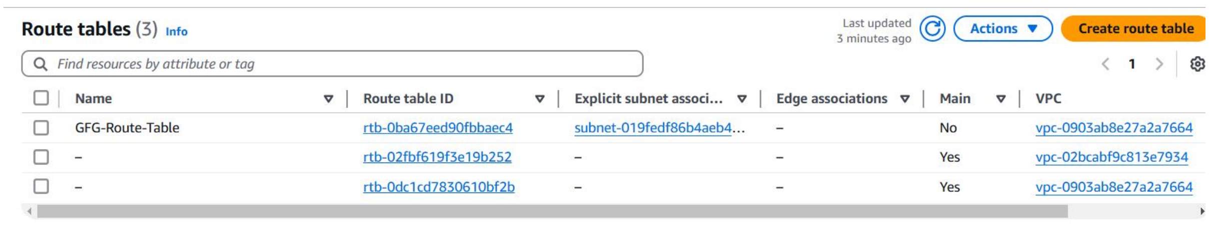
- Log in to the AWS Management Console and navigate to the VPC service.
- Verify that the VPC and subnets with the specified names and settings have been created.



Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
–	vpc-0903ab8e27a2a7664	Available	Off	10.0.0.0/16	–
–	vpc-02bcabf9c813e7934	Available	Off	172.31.0.0/16	–

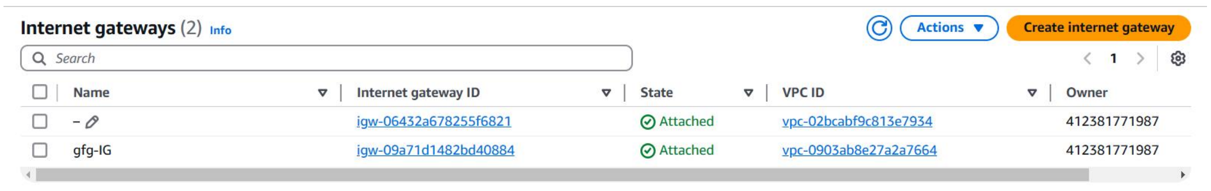

Subnets (4) Info
Last updated less than a minute ago
Actions
Create subnet
Find resources by attribute or tag

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
–	subnet-0b0ad38ef9892d748	Available	vpc-02bcabf9c813e7934	Off	172.31.32.0/
–	subnet-0ae2bde0f08b1116d	Available	vpc-02bcabf9c813e7934	Off	172.31.0.0/2
gfg-subnet	subnet-019fedf86b4aeb4ad	Available	vpc-0903ab8e27a2a7664	Off	10.0.1.0/24
–	subnet-003579e5e3e9cde0f	Available	vpc-02bcabf9c813e7934	Off	172.31.16.0/

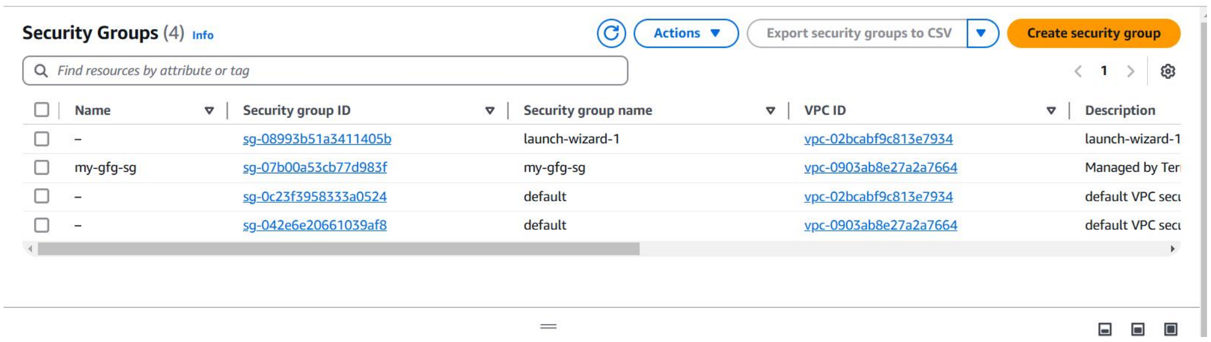

Route tables (3) Info

Last updated 3 minutes ago

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
GFG-Route-Table	rtb-0ba67eed90fbbaec4	subnet-019fedf86b4aeb4...	–	No	vpc-0903ab8e27a2a7664
–	rtb-02fbf619f3e19b252	–	–	Yes	vpc-02bcabf9c813e7934
–	rtb-0dc1cd7830610bf2b	–	–	Yes	vpc-0903ab8e27a2a7664


Internet gateways (2) Info

Name	Internet gateway ID	State	VPC ID	Owner
–	igw-06432a678255f6821	Attached	vpc-02bcabf9c813e7934	412381771987
gfg-IG	igw-09a71d1482bd40884	Attached	vpc-0903ab8e27a2a7664	412381771987


Security Groups (4) Info

Name	Security group ID	Security group name	VPC ID	Description
–	sg-08993b51a3411405b	launch-wizard-1	vpc-02bcabf9c813e7934	launch-wizard-1
my-gfg-sg	sg-07b00a53cb77d983f	my-gfg-sg	vpc-0903ab8e27a2a7664	Managed by Ter
–	sg-0c23f3958333a0524	default	vpc-02bcabf9c813e7934	default VPC secu
–	sg-042e6e20661039af8	default	vpc-0903ab8e27a2a7664	default VPC secu

Actions
Export security groups to CSV
Create security group
Find resources by attribute or tag

4. Update VPC Configuration:

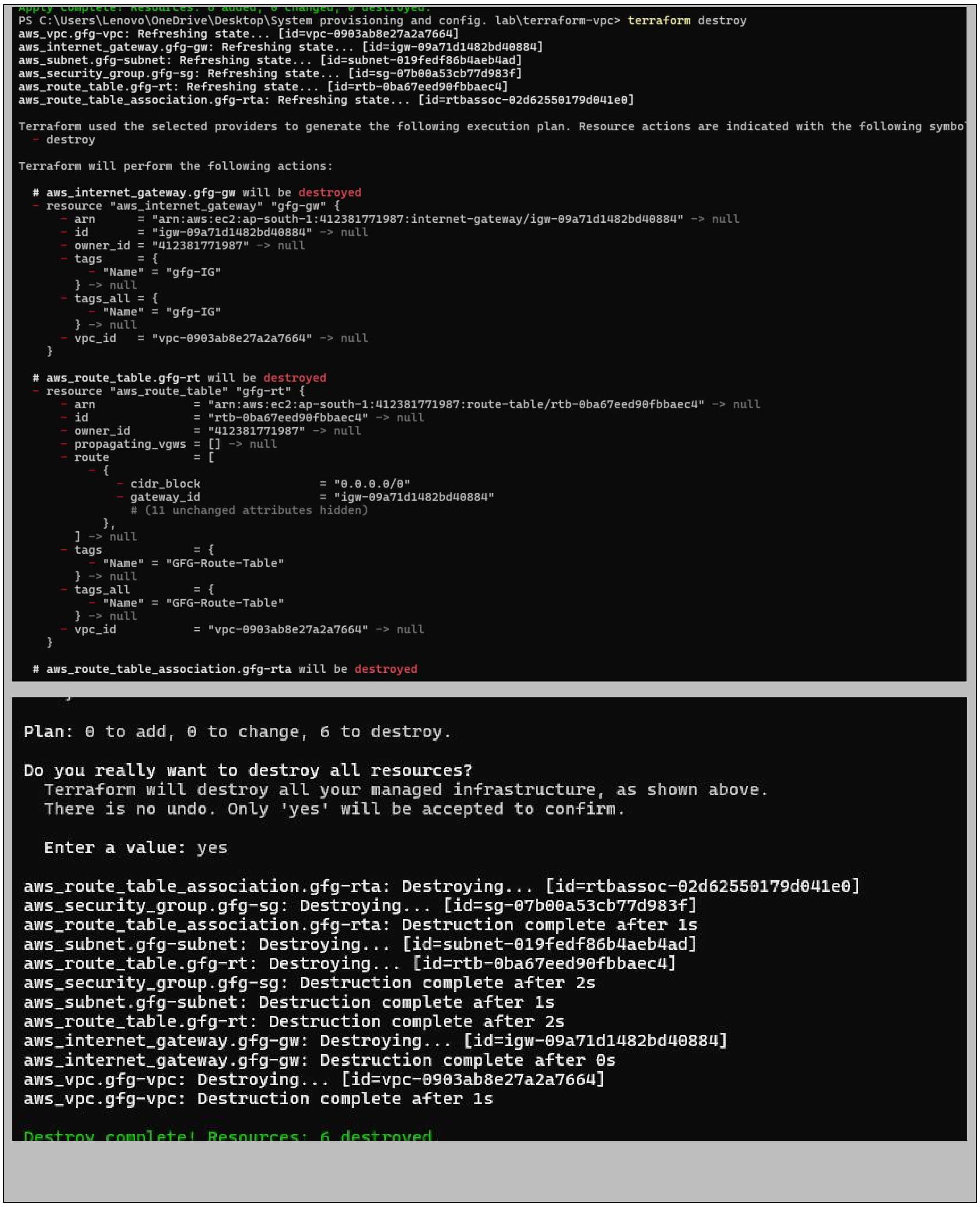
- If you want to modify the VPC configuration, update the main.tf file with the desired changes.
- Rerun the terraform apply command to apply the changes:

```
terraform apply
```

5. Clean Up:

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

```
terraform destroy
```



```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
PS C:\Users\Lenovo\OneDrive\Desktop\System provisioning and config. lab\terraform-vpc> terraform destroy
aws_vpc.gfg-vpc: Refreshing state... [id=vpc-0903ab8e27a2a7664]
aws_internet_gateway.gfg-gw: Refreshing state... [id=igw-09a71d1482bd40884]
aws_subnet.gfg-subnet: Refreshing state... [id=subnet-019fedf86b4aeb4ad]
aws_security_group.gfg-sg: Refreshing state... [id=sg-07b00a53cb77d983f]
aws_route_table.gfg-rt: Refreshing state... [id=rtb-0ba67eed90fbbaec4]
aws_route_table_association.gfg-rta: Refreshing state... [id=rtbassoc-02d62550179d041e0]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbo
  - 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-1:412381771987:internet-gateway/igw-09a71d1482bd40884" -> null
      - id       = "igw-09a71d1482bd40884" -> null
      - owner_id = "412381771987" -> null
      - tags     = {
          - "Name" = "gfg-IG"
        } -> null
      - tags_all = {
          - "Name" = "gfg-IG"
        } -> null
      - vpc_id   = "vpc-0903ab8e27a2a7664" -> null
    }

  # aws_route_table.gfg-rt will be destroyed
  - resource "aws_route_table" "gfg-rt" {
      - arn              = "arn:aws:ec2:ap-south-1:412381771987:route-table/rtb-0ba67eed90fbbaec4" -> null
      - id               = "rtb-0ba67eed90fbbaec4" -> null
      - owner_id         = "412381771987" -> null
      - propagating_vgws = [] -> null
      - route            = [
          - {
              - cidr_block                 = "0.0.0.0/0"
              - gateway_id                 = "igw-09a71d1482bd40884"
                # (11 unchanged attributes hidden)
            },
        ] -> null
      - tags             = {
          - "Name" = "GFG-Route-Table"
        } -> null
      - tags_all         = {
          - "Name" = "GFG-Route-Table"
        } -> null
      - vpc_id           = "vpc-0903ab8e27a2a7664" -> null
    }

  # aws_route_table_association.gfg-rta will be destroyed
```

```
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-02d62550179d041e0]
aws_security_group.gfg-sg: Destroying... [id=sg-07b00a53cb77d983f]
aws_route_table_association.gfg-rta: Destruction complete after 1s
aws_subnet.gfg-subnet: Destroying... [id=subnet-019fedf86b4aeb4ad]
aws_route_table.gfg-rt: Destroying... [id=rtb-0ba67eed90fbbaec4]
aws_security_group.gfg-sg: Destruction complete after 2s
aws_subnet.gfg-subnet: Destruction complete after 1s
aws_route_table.gfg-rt: Destruction complete after 2s
aws_internet_gateway.gfg-gw: Destroying... [id=igw-09a71d1482bd40884]
aws_internet_gateway.gfg-gw: Destruction complete after 0s
aws_vpc.gfg-vpc: Destroying... [id=vpc-0903ab8e27a2a7664]
aws_vpc.gfg-vpc: Destruction complete after 1s

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.