System Provisioning and Configuration Management Lab

LAB 11: Creating a VPC in Terraform

Under the Guidance of: **Dr. Hitesh Kumar Sharma**

Submitted by: Vibhav Khaneja

SAP: 500105662

Batch:DevOps-B1(N-H)

Roll No.: R2142220297

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
```

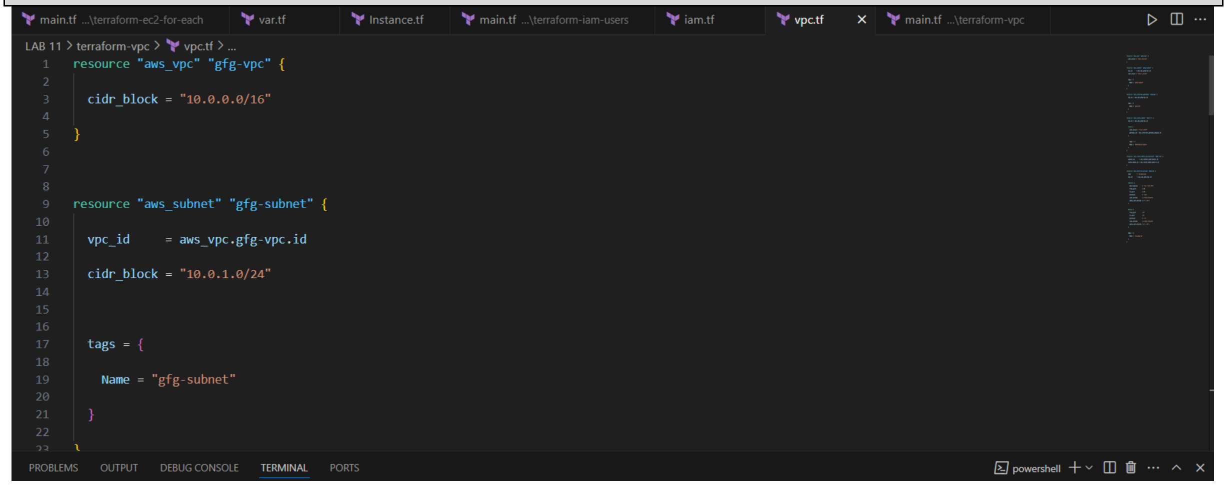
- Create Terraform Configuration Files:
- Create a file named main.tf:

# 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"
```

```
 }
}
```



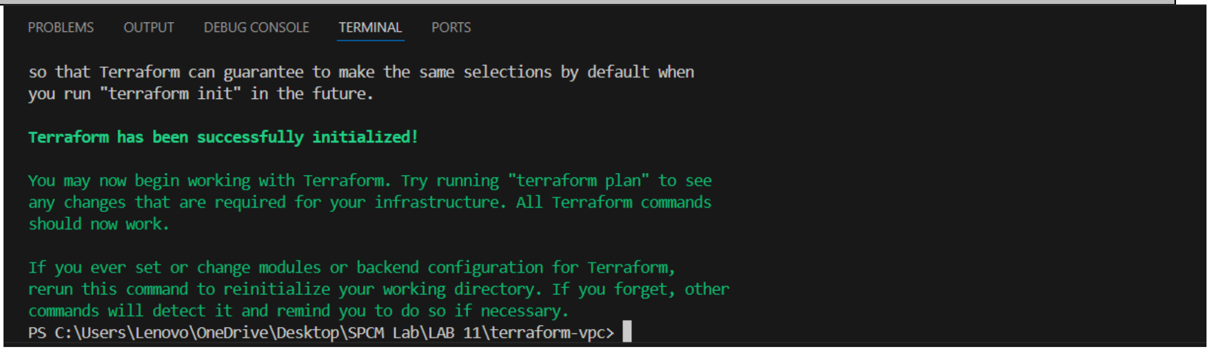
```
resource "aws_vpc" "gfg-vpc" {

  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"

  }

}
```

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
```



```
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\SPCM Lab\LAB 11\terraform-vpc>
```

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

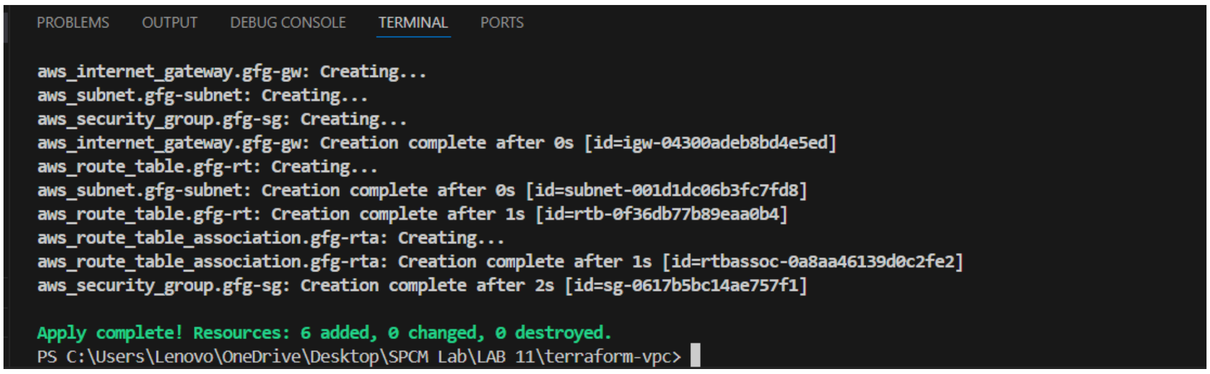
3. Verify Resources in AWS Console:

- 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:

```
terraform apply
```



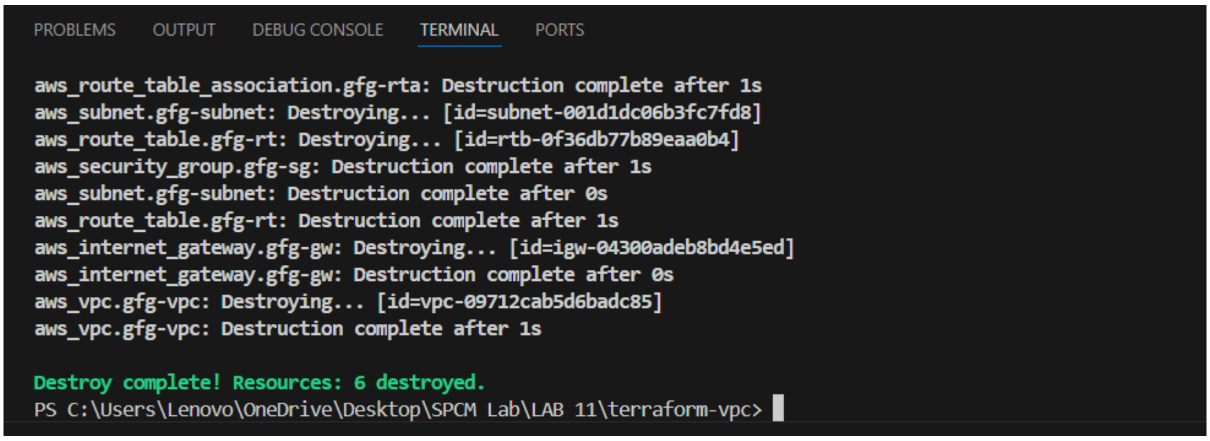
```
aws_internet_gateway.gfg-gw: Creating...
aws_subnet.gfg-subnet: Creating...
aws_security_group.gfg-sg: Creating...
aws_internet_gateway.gfg-gw: Creation complete after 0s [id=igw-04300adeb8bd4e5ed]
aws_route_table.gfg-rt: Creating...
aws_subnet.gfg-subnet: Creation complete after 0s [id=subnet-001d1dc06b3fc7fd8]
aws_route_table.gfg-rt: Creation complete after 1s [id=rtb-0f36db77b89eaa0b4]
aws_route_table_association.gfg-rta: Creating...
aws_route_table_association.gfg-rta: Creation complete after 1s [id=rtbassoc-0a8aa46139d0c2fe2]
aws_security_group.gfg-sg: Creation complete after 2s [id=sg-0617b5bc14ae757f1]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 11\terraform-vpc>
```

5. Clean Up:

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

```
terraform destroy
```

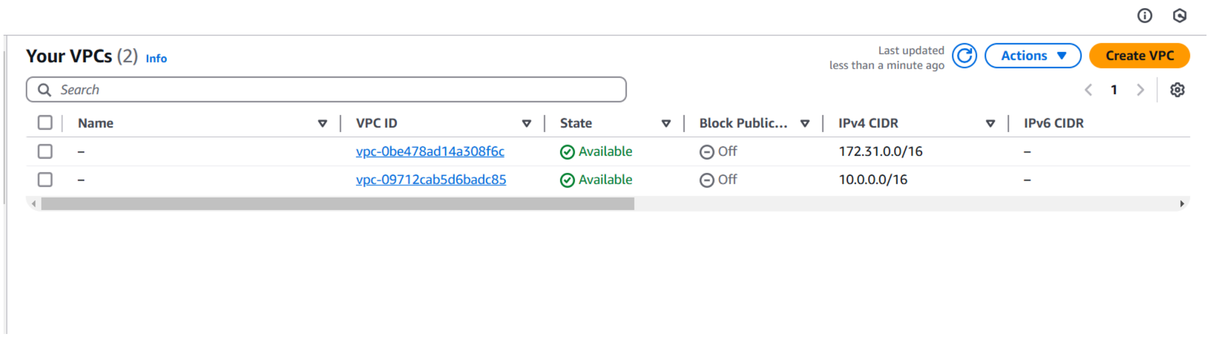
Confirm the destruction by typing yes.



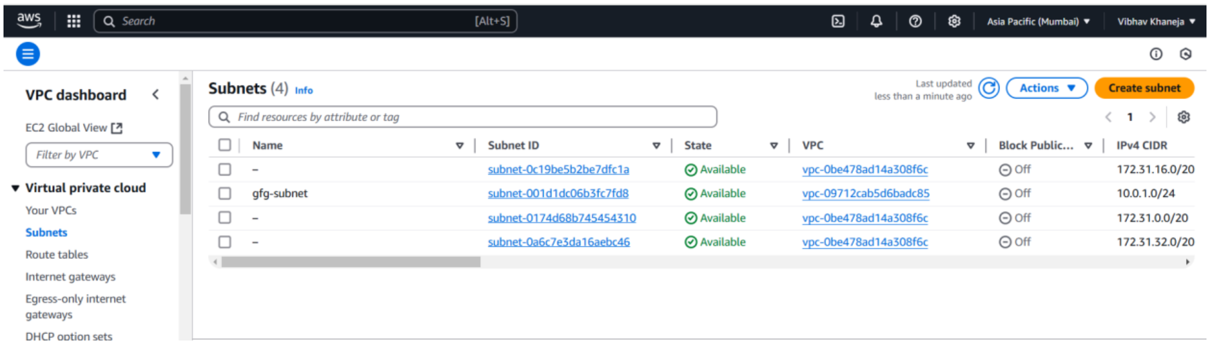
```
aws_route_table_association.gfg-rta: Destruction complete after 1s
aws_subnet.gfg-subnet: Destroying... [id=subnet-001d1dc06b3fc7fd8]
aws_route_table.gfg-rt: Destroying... [id=rtb-0f36db77b89eaa0b4]
aws_security_group.gfg-sg: Destruction complete after 1s
aws_subnet.gfg-subnet: Destruction complete after 0s
aws_route_table.gfg-rt: Destruction complete after 1s
aws_internet_gateway.gfg-gw: Destroying... [id=igw-04300adeb8bd4e5ed]
aws_internet_gateway.gfg-gw: Destruction complete after 0s
aws_vpc.gfg-vpc: Destroying... [id=vpc-09712cab5d6badc85]
aws_vpc.gfg-vpc: Destruction complete after 1s

Destroy complete! Resources: 6 destroyed.
PS C:\Users\Lenovo\OneDrive\Desktop\SPCM Lab\LAB 11\terraform-vpc>
```

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.



Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
–	vpc-0be478ad14a308f6c	Available	Off	172.31.0.0/16	–
–	vpc-09712cab5d6badc85	Available	Off	10.0.0.0/16	–



Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
–	subnet-0c19be5b2be7dfc1a	Available	vpc-0be478ad14a308f6c	Off	172.31.16.0/20
gfg-subnet	subnet-001d1dc06b3fc7fd8	Available	vpc-09712cab5d6badc85	Off	10.0.1.0/24
–	subnet-0174d68b745454310	Available	vpc-0be478ad14a308f6c	Off	172.31.0.0/20
–	subnet-0a6c7e3da16aebc46	Available	vpc-0be478ad14a308f6c	Off	172.31.32.0/20