

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.

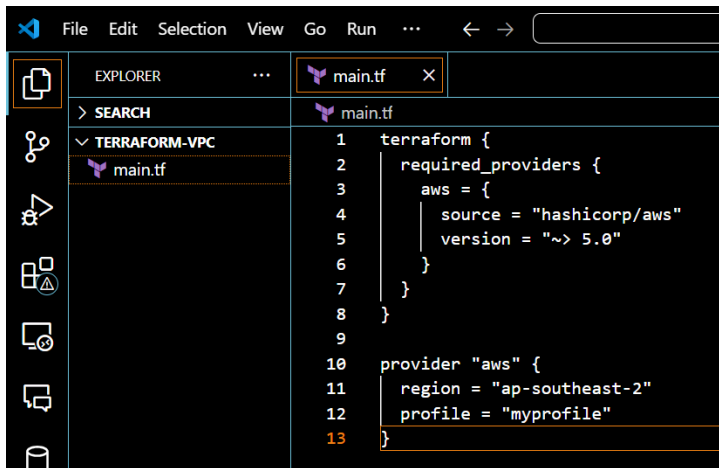
Steps:

1. Create a Terraform Directory:

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

```
D:\College\Sem-6\System Provisioning\Lab>mkdir terraform-vpc  
D:\College\Sem-6\System Provisioning\Lab>cd terraform-vpc  
D:\College\Sem-6\System Provisioning\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"
  }
}

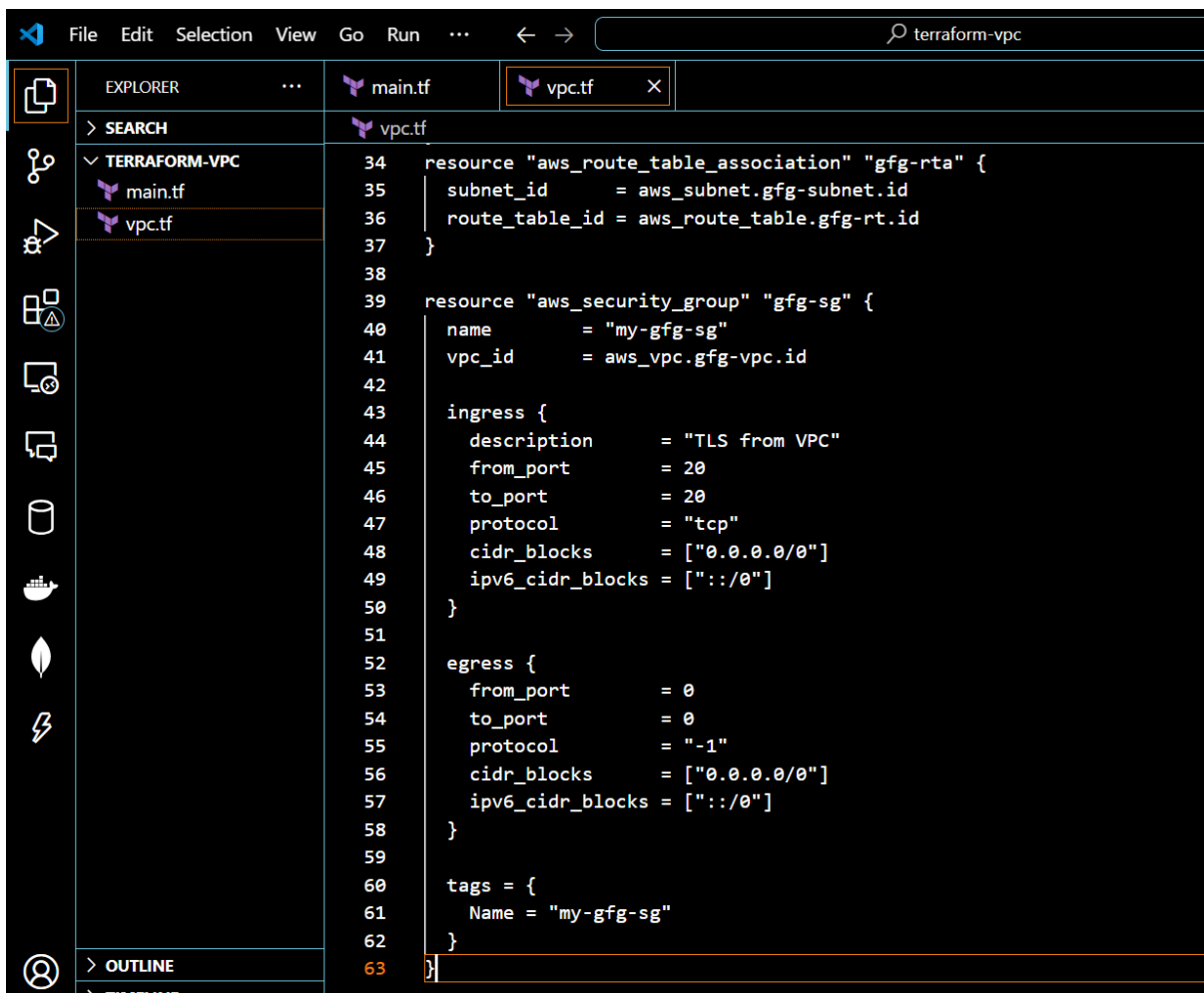
resource "aws_route_table_association" "gfg-rta" {
  subnet_id    = aws_subnet.gfg-subnet.id
  route_table_id = aws_route_table.gfg-rt.id
}

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"
}
}
```



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays a project named 'TERRAFORM-VPC' with two files: 'main.tf' and 'vpc.tf'. The 'vpc.tf' file is selected and its content is displayed in the main editor area. The code defines an 'aws_route_table_association' resource, an 'aws_security_group' resource with an 'ingress' rule, and an 'egress' rule. The 'egress' rule has the same configuration as the one shown in the first block of the document. The status bar at the bottom indicates the current line is 63.

```
34 resource "aws_route_table_association" "gfg-rta" {
35   subnet_id      = aws_subnet.gfg-subnet.id
36   route_table_id = aws_route_table.gfg-rt.id
37 }
38
39 resource "aws_security_group" "gfg-sg" {
40   name      = "my-gfg-sg"
41   vpc_id    = aws_vpc.gfg-vpc.id
42
43   ingress {
44     description      = "TLS from VPC"
45     from_port        = 20
46     to_port          = 20
47     protocol         = "tcp"
48     cidr_blocks      = ["0.0.0.0/0"]
49     ipv6_cidr_blocks = [ "::/0" ]
50   }
51
52   egress {
53     from_port    = 0
54     to_port      = 0
55     protocol     = "-1"
56     cidr_blocks  = ["0.0.0.0/0"]
57     ipv6_cidr_blocks = [ "::/0" ]
58   }
59
60   tags = {
61     Name = "my-gfg-sg"
62   }
63 }
```

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 D:\College\Sem-6\System Provisioning\Lab\terraform-vpc> C:\terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.88.0...
- Installed hashicorp/aws v5.88.0 (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.
```

terraform apply

```
Enter a value: yes

aws_vpc.gfg-vpc: Creating...
aws_vpc.gfg-vpc: Creation complete after 3s [id=vpc-0e306420f401dd4d6]
aws_internet_gateway.gfg-gw: Creating...
aws_subnet.gfg-subnet: Creating...
aws_security_group.gfg-sg: Creating...
aws_subnet.gfg-subnet: Creation complete after 1s [id=subnet-0efb5e947d9ed8914]
aws_internet_gateway.gfg-gw: Creation complete after 1s [id=igw-017bed7693c92bd69]
aws_route_table.gfg-rt: Creating...
aws_route_table.gfg-rt: Creation complete after 2s [id=rtb-01aa3ca8d06074501]
aws_route_table_association.gfg-rta: Creating...
aws_route_table_association.gfg-rta: Creation complete after 1s [id=rtbassoc-00be26489e935b812]
aws_security_group.gfg-sg: Creation complete after 4s [id=sg-093cf584489d3f9da]

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:

Subnets (4) [Info](#) Last updated less than a minute ago

<input type="checkbox"/>	Name	Subnet ID	State	VPC
<input type="checkbox"/>	-	subnet-008f6e249503c6ca3	Available	vpc-03575c7a95621f7a0
<input type="checkbox"/>	-	subnet-039a568c8483f1b65	Available	vpc-03575c7a95621f7a0
<input type="checkbox"/>	-	subnet-0fdb4875e4b03d6f6	Available	vpc-03575c7a95621f7a0
<input type="checkbox"/>	gfg-subnet	subnet-0c87b8d42a73042f3	Available	vpc-04c675dd44521bf87

Your VPCs (2) [Info](#) Last updated less than a minute ago

<input type="checkbox"/>	Name	VPC ID	State	Block Public...	IPv4 CIDR
<input type="checkbox"/>	-	vpc-03575c7a95621f7a0	Available	Off	172.31.0.0/16
<input type="checkbox"/>	-	vpc-04c675dd44521bf87	Available	Off	10.0.0.0/16

Route tables (3) [Info](#) Last updated 1 minute ago

<input type="checkbox"/>	Name	Route table ID	Explicit subnet associ...	Edge associations
<input type="checkbox"/>	GFG-Route-Table	rtb-0d1449dae8b2c1c8c	subnet-0c87b8d42a7304...	-
<input type="checkbox"/>	-	rtb-026fa23ef18918d4f	-	-
<input type="checkbox"/>	-	rtb-0695fc895ab3026bb	-	-

Internet gateways (2) [Info](#) Actions

<input type="checkbox"/>	Name	Internet gateway ID	State	VPC ID
<input type="checkbox"/>	-	igw-0182e8a55d84a1636	Attached	vpc-03575c7a95621f7a0
<input type="checkbox"/>	gfg-IG	igw-06877524cec80dc6	Attached	vpc-04c675dd44521bf87

- 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
```

```
aws_route_table_association.gfg-rta: Destroying... [id=rtbassoc-00be26489e935b812]
aws_security_group.gfg-sg: Destroying... [id=sg-093cf584489d3f9da]
aws_route_table_association.gfg-rta: Destruction complete after 1s
aws_subnet.gfg-subnet: Destroying... [id=subnet-0efb5e947d9ed8914]
aws_route_table.gfg-rt: Destroying... [id=rtb-01aa3ca8d06074501]
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-017bed7693c92bd69]
aws_internet_gateway.gfg-gw: Destruction complete after 1s
aws_vpc.gfg-vpc: Destroying... [id=vpc-0e306420f401dd4d6]
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.