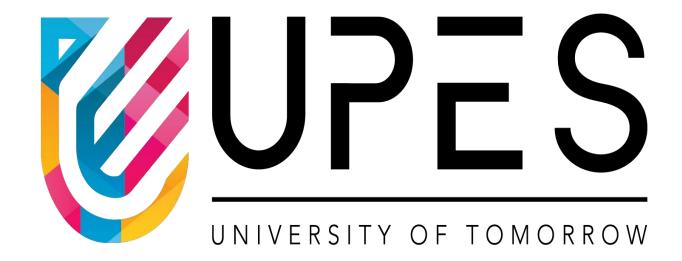
School of Computer Science

University of Petroleum and Energy Studies



System Provisioning & Configuration Management

Lab File (6th Sem)

Submitted By:-Akshat Pandey 500101788 R2142220306 DevOps B1 Submitted To:-Dr Hitesh Kumar Sharma

EXPERIMENT 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
```

```
C:\Users\aksha\Documents\SPCM_LAB>mkdir terraform-vpc
```

C:\Users\aksha\Documents\SPCM_LAB>cd terraform-vpc

C:\Users\aksha\Documents\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 = "o.o.o.o/o"
 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" {
          = "my-gfg-sg"
name
vpc_id = aws_vpc.gfg-vpc.id
ingress {
```

```
= "TLS from VPC"
 description
 from_port
              = 20
 to_port
             = 20
 protocol
             = "tcp"
 cidr\_blocks = ["o.o.o.o/o"]
 ipv6_cidr_blocks = ["::/o"]
egress {
 from_port
            = 0
 to_port
             = 0
             = "-1"
 protocol
 cidr\_blocks = ["o.o.o.o/o"]
 ipv6_cidr_blocks = ["::/o"]
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.

```
ypc.tf
                                                                                                                                                              × Y provider.tf
          .tf > 43 resource "aws_internet_gateway" "c
    resource "aws_vpc" "gfg-vpc" {
      cidr_block = "10.0.0.0/16"
               tags = {
   Name = "gfg-vpc"
                                                                                                                                                          subnet_id = aws_subnet.gfg-subnet.id
route_table_id = aws_route_table.gfg-rt.id
               esource "aws_subnet" "gfg-subnet" {

vpc_id = aws_vpc.gfg-vpc.id

cidr_block = "10.0.1.0/24"
                                                                                                                                                      resource "aws_security_group" "gfg-sg" {
    name = "my-gfg-sg"
    vpc_id = aws_vpc.gfg-vpc.id
               tags = {
  Name = "gfg-subnet"
                                                                                                                                                                                              = "TLS from VPC"
                                                                                                                                                              description
                                                                                                                                                              from port
               esource "aws_internet_gateway" "gfg-gw" {|
vpc_id = aws_vpc.gfg-vpc.id
                                                                                                                                                              to_port
                                                                                                                                                              protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
ipv6_cidr_blocks = ["::/0"]
               tags = {
Name = "gfg-IG"
                esource "aws_route_table" "gfg-rt" {
vpc_id = aws_vpc.gfg-vpc.id
                                                                                                                                                              from_port
                                                                                                                                                              to_port
               route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gfg-gw.id
                                                                                                                                                             protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
ipv6_cidr_blocks = ["::/0"]
                   tags = {
Name = "GFG-Route-Table"
                                                                                                                                                          tags = {
Name = "my-gfg-sg"
              esource "aws_route_table_association" "gfg-rta" {
    subnet_id = aws_subnet.gfg-subnet.id
```

2. Initialize and Apply:

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

terraform init terraform apply

```
\Documents\SPCM_LAB\terraform-vpc>terraform init
C:\Users\aksha\Documents\SPCM_LAB\terraform-vpc>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.96.0...
- Installed hashicorp/aws v5.96.0...
- Installed hashicorp/aws v5.96.0 (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
 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.
C:\Users\aksha\Documents\SPCM_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:
   }
+ tags_all = {
+ "Name" = "gfg-IG"
                 vpc_id = (known after apply)
   ] = {
tags = "GFG-Route-Table"
+ "Name" = "GFG-Route-Table"
                }
tags_all
                                                                                        = (known after apply)
= true
               = (known after

= true

= (known after apply)

= (known after apply)

= "default"

= (known after apply)

p (known after apply)

= (known after apply)

= (known after apply)

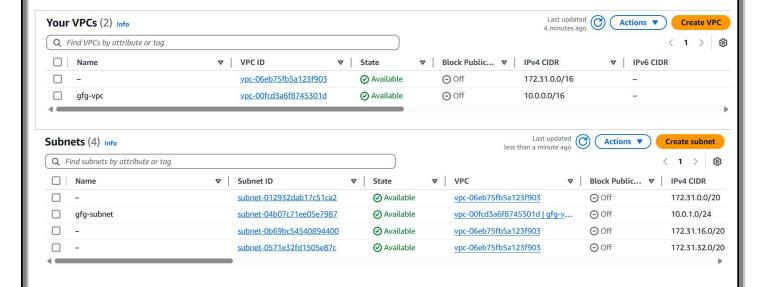
= (known after apply)

= (known after apply)
              tags
+ "Name" = "gfg-vpc"
               }
tags_all
+ "Name" = "gfg-vpc"
 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-00fcd3a6f8745301d]
aws_internet_gateway.gfg-gw: Creating...
aws_subnet.gfg-subnet: Creating...
aws_security_group.gfg-sg: Creating...
aws_security_group.gfg-sg: Creating...
aws_internet_gateway.gfg-gw: Creation complete after 0s [id=igw-0f24590595076ab70]
aws_route_table.gfg-rt: Creation.
aws_subnet.gfg-subnet: Creation complete after 0s [id=subnet-04b07c71ee05e7987]
aws_subnet.gfg-subnet: Creation complete after 1s [id=rtb-0a03a855b1f33f06e]
aws_route_table.gfg-pt: Creation complete after 0s [id=rtbassoc-0c0148fef7a957db5]
aws_route_table_association.gfg-rta: Creation complete after 0s [id=rtbassoc-0c0148fef7a957db5]
aws_security_group.gfg-sg: Creation complete after 2s [id=sg-09e2eb305d8aa2a9a]
```

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

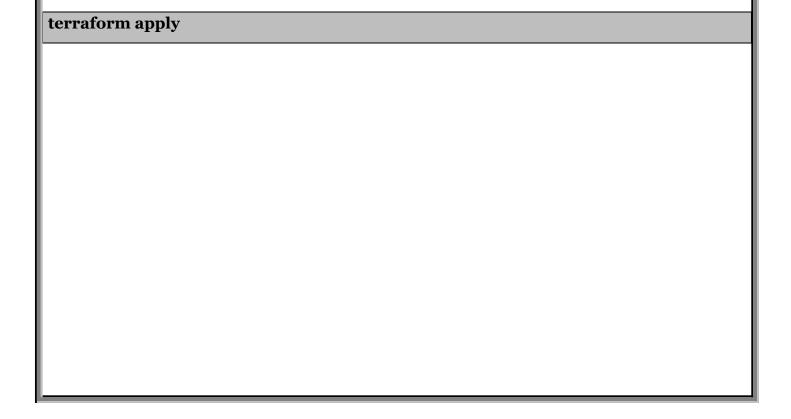
3. Verify Resources in AWS Console:

- 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:



```
C:\Users\aksha\Documents\SPCM_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
            "aws_internet_gateway" "gfg-gw" {
= (known after apply)
   resource
     + arn
       id = (known after apply)
owner_id = (known after apply)
     + id
      + tags
               = {
           "Name" = "gfg-IG"
       tags_all = {
           "Name" = "gfg-IG"
       vpc_id = (known after apply)
 + id
                        = (known after apply)
                       = (known after apply)
     + owner_id
       propagating_vgws = (known after apply)
     + route
                       = [
                                         = "0.0.0.0/0"
             + cidr_block
                                         = (known after apply)
             + gateway_id
               # (11 unchanged attributes hidden)
     + tags
           "Name" = "GFG-Route-Table"
     + tags_all
                       = {
      + enable_dns_hostnames
                                                  = (known after apply)
      + enable_dns_support
                                                  = true
      + enable_network_address_usage_metrics = (known after apply)
                                                  = (known after apply)
      + id
                                                  = "default"
      + instance_tenancy
       + ipv6_association_id
                                                  = (known after apply)
      + ipv6_cidr_block
                                                  = (known after apply)
       + ipv6_cidr_block_network_border_group = (known after apply)
                                                  = (known after apply)
      + main_route_table_id
       + owner_id
                                                  = (known after apply)
                                                     {
        tags
           + "Name" = "gfg-vpc"
       + tags_all
                                                  = {
           + "Name" = "gfg-vpc"
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-00fcd3a6f8745301d]
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-0f24590595076ab70] aws_route_table.gfg-rt: Creating...
aws_subnet.gfg-subnet: Creation complete after 0s [id=subnet-04b07c71ee05e7987]
aws_route_table.gfg-rt: Creation complete after 1s [id=rtb-0a03a855b1f33f06e]
aws_route_table_association.gfg-rta: Creating...
aws_route_table_association.gfg-rta: Creation complete after 0s [id=rtbassoc-0c0148fef7a957db5]
aws_security_group.gfg-sg: Creation complete after 2s [id=sg-09e2eb305d8aa2a9a]
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
```

5. Clean Up:

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

terraform destroy

```
ers\aksha\Documents\s\P\M_LLAB\terratorm-rop-reterratorm destroy
o.e_gf=yror. Refreshing state... [id=vpc-00fcd34687H59301d]
thernet_gatemay.gfg-gm: Refreshing state... [id=sigm-0f2H590595076ab70]
bubet.gfg=subnet: Refreshing state... [id=subnet-04b07C7lee05e7987]
scurity_group.gfg-gg: Refreshing state... [id=sp-09e2eb305d3ma2a9a]
bute_table_gfg-rt: Refreshing state... [id=sp-09e2eb305d3ma2a9a]
bute_table_gfg-rt: Refreshing state... [id=sp-09e2eb305d3ma2a9a]
bute_table_association.gfg-rta: Refreshing state... [id=sp-09e2eb305d3ma2a9a]
   erraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols
- destroy
  erraform 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-1:158878148841:internet-gateway/igw-0f24590595076ab70" -> null
    id = "igw-0f245995959676ab70" -> null
    owner_id = "158878148841" -> null
    tags = {
        "Name" = "gfg-IG"
    } - wlame" = "gfg-IG"
}
                tags_all = {
- "Name" = "gfg-IG"
                                 = "vpc-00fcd3a6f8745301d" -> null
               cidr_block = "0.0.0.0/0"
gateway.id = "igw-0f24590595076ab70"
# (11 unchanged attributes hidden)

        default_route_table_id
        = "rtb-0f7faa5ad07b17de8" -> null

        default_security_group_id
        = "sg-07d25005ee3cef497" -> null

        dhcp_options_id
        = "dopt-0f5aa075b3f4a8b77" -> null

                     dhcp_options_id
enable_dns_hostnames
                      enable_network_address_usage_metrics = false -> null
id = "vpc-00fcd3a6f8745301d" -> null
instance_tenancy = "default" -> null
                      instance_tenancy
ipv6_netmask_length
                                                                                                                           = 0 -> null
= "rtb-0f7faa5ad07b17de8" -> null
= "158878148841" -> null
= {
                      main_route_table_id
                      tags
- "Name" = "gfg-vpc"
                       tags_all
- "Name" = "gfg-vpc"
                       } -> null
# (4 unchanged attributes hidden)
 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-0c0148fef7a957db5] aws_security_group.gfg-sg: Destroying... [id=sg-09e2eb305d8aa2a9a] aws_route_table_association.gfg-rta: Destruction complete after 0s aws_subnet.gfg-subnet: Destroying... [id=subnet-04b07c7lee05e7987] aws_route_table.gfg-rt: Destroying... [id=rtb-0a03a855b1f33f06e] aws_subnet.gfg-subnet: Destruction complete after 1s aws_security_group.gfg-sg: Destruction complete after 1s aws_route_table.gfg-rt: Destruction complete after 1s aws_internet_gateway.gfg-gw: Destroying... [id=igw-0f24590595076ab70] aws_internet_gateway.gfg-gw: Destruction complete after 0s aws_vpc.gfg-vpc: Destruction complete after 1s
  Destroy complete! Resources: 6 destroyed.
 C:\Users\aksha\Documents\SPCM_LAB\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.