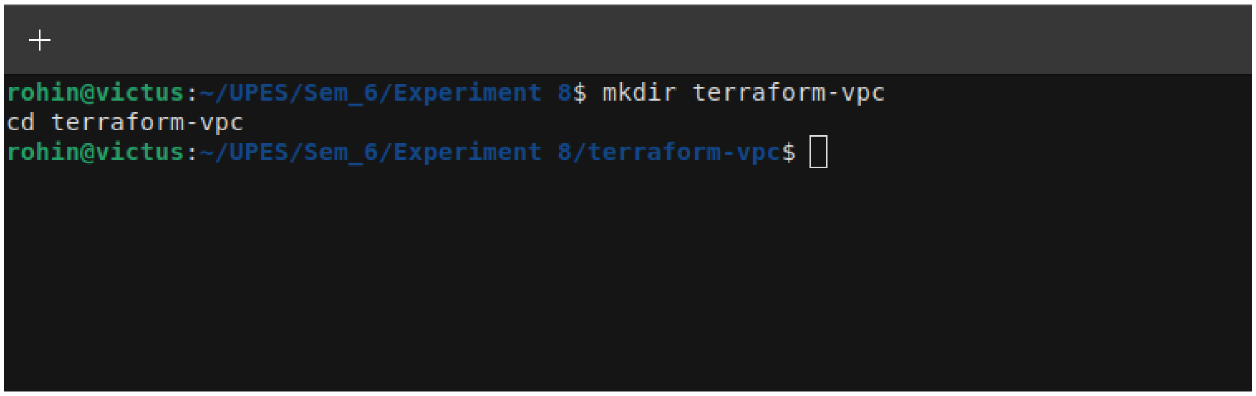
Lab Exercise 8– Creating a VPC in Terraform Objective:

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.

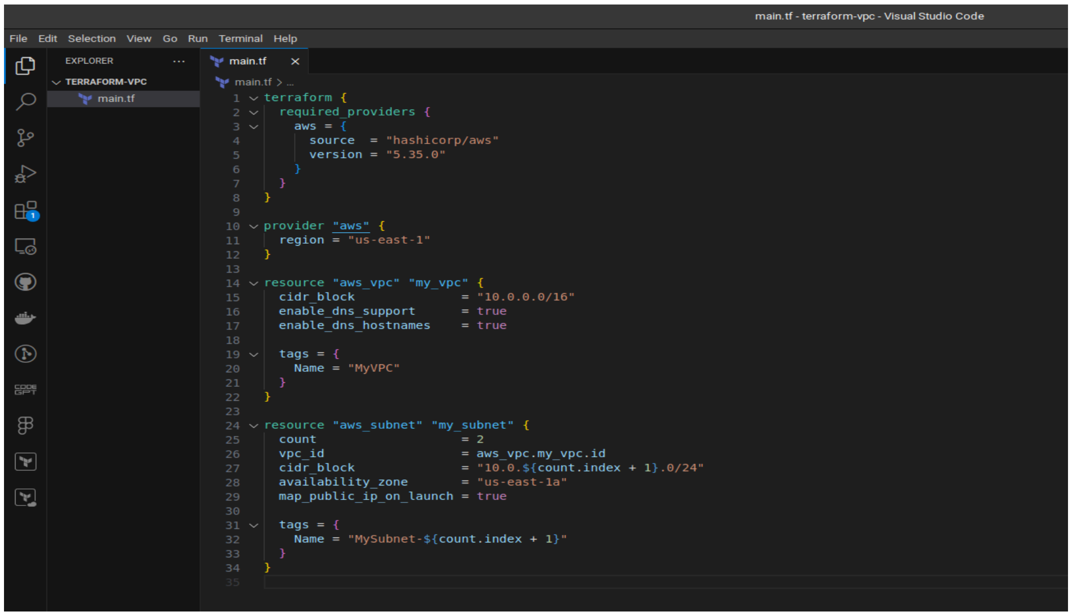
Steps:
1. Create a Terraform Directory:



```
rohin@victus:~/UPES/Sem_6/Experiment 8$ mkdir terraform-vpc
cd terraform-vpc
rohin@victus:~/UPES/Sem_6/Experiment 8/terraform-vpc$ 
```

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

# Main.tf



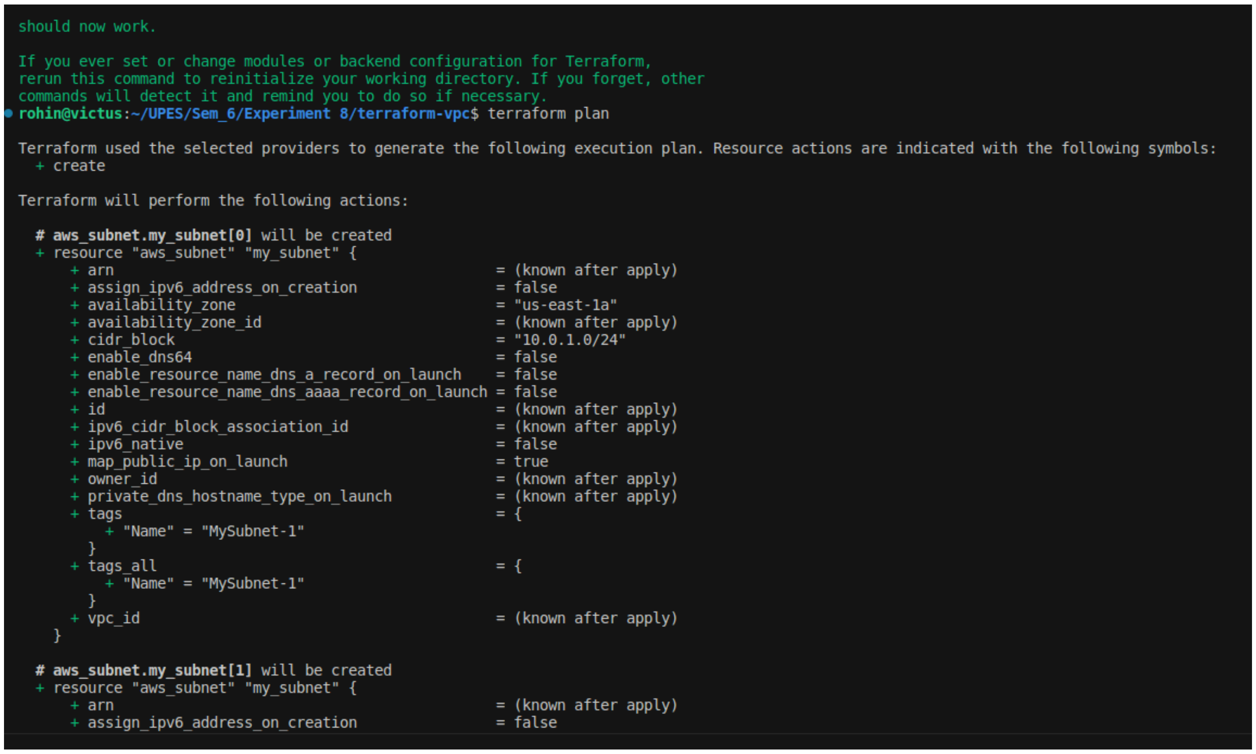
main.tf - terraform-vpc - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
TERRAFORM-VPC
main.tf
main.tf > ...

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.35.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "my_vpc" {
  cidr_block               = "10.0.0.0/16"
  enable_dns_support       = true
  enable_dns_hostnames     = true

  tags = {
    Name = "MyVPC"
  }
}

resource "aws_subnet" "my_subnet" {
  count                   = 2
  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = "10.0.${count.index + 1}.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "MySubnet-${count.index + 1}"
  }
}
```

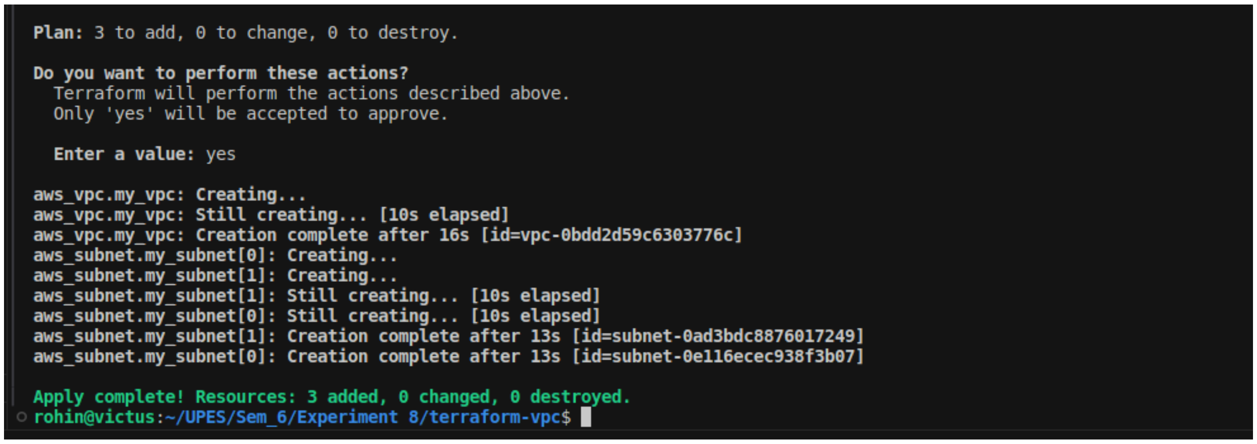
In this configuration, we define an AWS provider, a VPC with a specified CIDR block, and two subnets within the VPC.

2. Initialize and Apply:

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



```
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.
rohin@victus:~/UPES/Sem_6/Experiment 8/terraform-vpc$ terraform plan

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_subnet.my_subnet[0] will be created
  + resource "aws_subnet" "my_subnet" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.1.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = true
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = {
          + "Name" = "MySubnet-1"
        }
      + tags_all                                       = {
          + "Name" = "MySubnet-1"
        }
      + vpc_id                                         = (known after apply)
    }

  # aws_subnet.my_subnet[1] will be created
  + resource "aws_subnet" "my_subnet" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
```

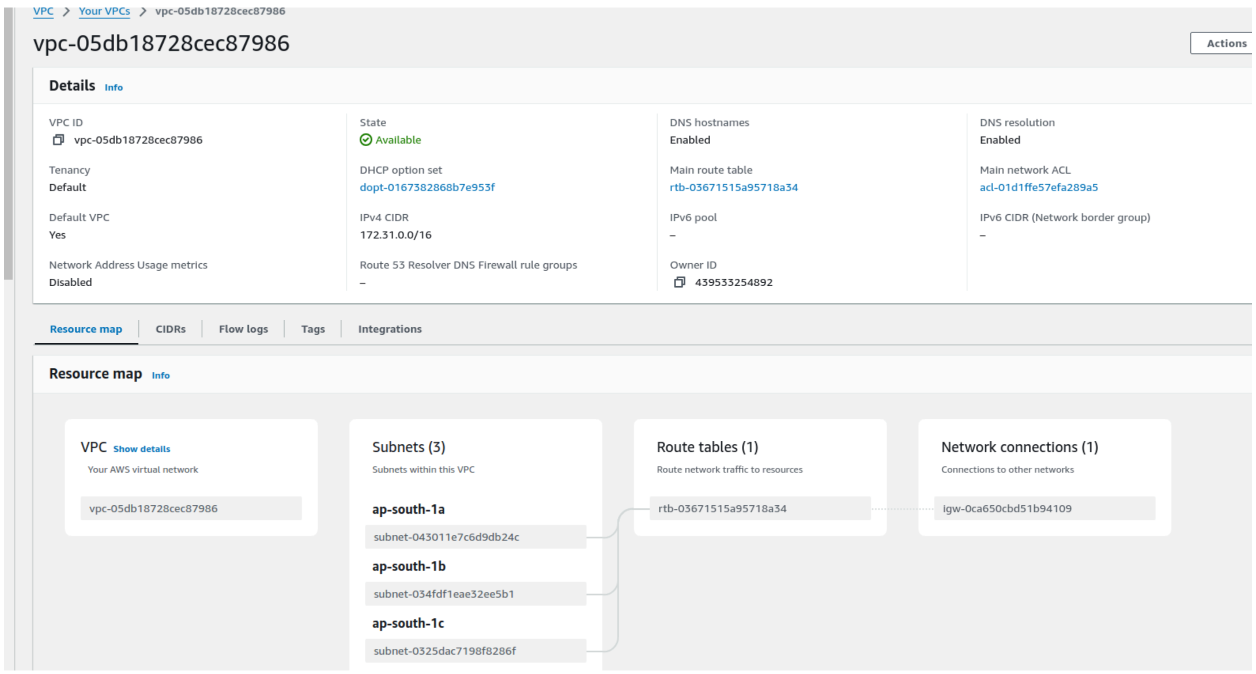


```
Plan: 3 to add, 0 to change, 0 to destroy.

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.my_vpc: Creating...
aws_vpc.my_vpc: Still creating... [10s elapsed]
aws_vpc.my_vpc: Creation complete after 16s [id=vpc-0bdd2d59c6303776c]
aws_subnet.my_subnet[0]: Creating...
aws_subnet.my_subnet[1]: Creating...
aws_subnet.my_subnet[1]: Still creating... [10s elapsed]
aws_subnet.my_subnet[0]: Still creating... [10s elapsed]
aws_subnet.my_subnet[1]: Creation complete after 13s [id=subnet-0ad3bdc8876017249]
aws_subnet.my_subnet[0]: Creation complete after 13s [id=subnet-0e116ecec938f3b07]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
rohin@victus:~/UPES/Sem_6/Experiment 8/terraform-vpc$
```

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



vpc-05db18728cec87986

Details

VPC ID vpc-05db18728cec87986	State Available	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0167382868b7e953f	Main route table rtb-03671515a95718a34	Main network ACL acl-01d1ffe57efa289a5
Default VPC Yes	IPv4 CIDR 172.31.0.0/16	IPv6 pool –	IPv6 CIDR (Network border group) –
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 439533254892	

Resource map

VPC
Your AWS virtual network
vpc-05db18728cec87986

Subnets (3)
Subnets within this VPC
ap-south-1a
subnet-043011e7c6d9db24c
ap-south-1b
subnet-034fdf1eae32ee5b1
ap-south-1c
subnet-0325dac7198f8286f

Route tables (1)
Route network traffic to resources
rtb-03671515a95718a34

Network connections (1)
Connections to other networks
igw-0ca650cbd51b94109

4. Update VPC Configuration:

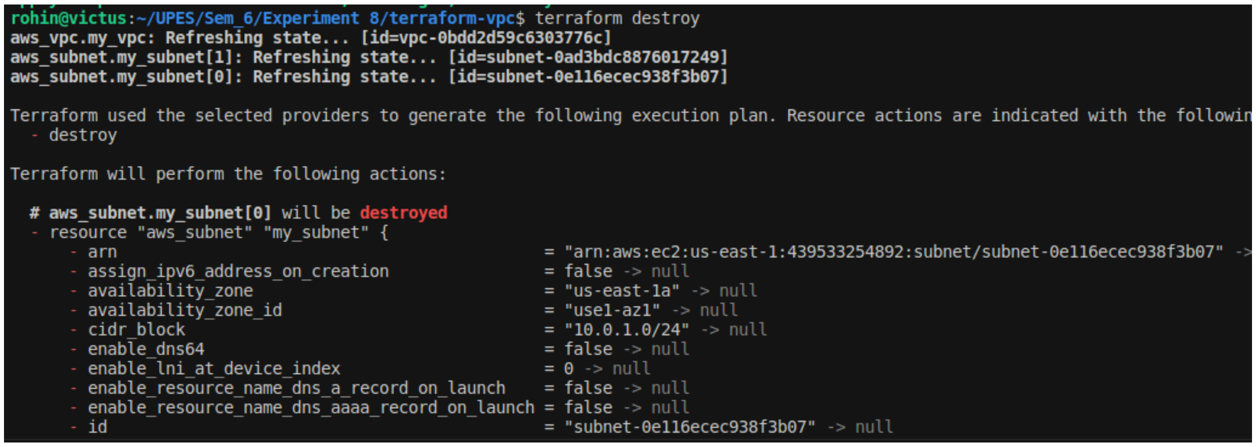
•If you want to modify the VPC configuration, update the main.tf file with the desired changes.

•Rerun the terraform apply command to apply the changes:

5. Clean Up:

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

`terraform destroy`



```
rohin@victus:~/UPES/Sem_6/Experiment 8/terraform-vpc$ terraform destroy
aws_vpc.my_vpc: Refreshing state... [id=vpc-0bdd2d59c6303776c]
aws_subnet.my_subnet[1]: Refreshing state... [id=subnet-0ad3bdc8876017249]
aws_subnet.my_subnet[0]: Refreshing state... [id=subnet-0e116ecec938f3b07]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the followin
  - destroy

Terraform will perform the following actions:

  # aws_subnet.my_subnet[0] will be destroyed
  - resource "aws_subnet" "my_subnet" {
      - arn                                            = "arn:aws:ec2:us-east-1:439533254892:subnet/subnet-0e116ecec938f3b07" ->
      - assign_ipv6_address_on_creation                = false -> null
      - availability_zone                              = "us-east-1a" -> null
      - availability_zone_id                           = "use1-az1" -> null
      - cidr_block                                     = "10.0.1.0/24" -> null
      - enable_dns64                                   = false -> null
      - enable_lni_at_device_index                     = 0 -> null
      - enable_resource_name_dns_a_record_on_launch    = false -> null
      - enable_resource_name_dns_aaaa_record_on_launch = false -> null
      - id                                             = "subnet-0e116ecec938f3b07" -> null
```

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.