Lab Exercise 8: Building a VPC with Terraform in AWS

Objective:

Learn how to use Terraform to create a basic Virtual Private Cloud (VPC) within Amazon Web Services (AWS).

- .

Steps:

1. Setting Up the Terraform Project:

- Create a directory for your Terraform project:

mkdir terraform-vpc

cd terraform-vpc

- Initialize the Terraform directory:

Bash

terraform init

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.34.0

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.
```

2. Configuring the VPC and Subnets:

- Create a file named main.tf and paste the following configuration:

Welcome | .terraform.lock.hcl | main.tf | vpc.tf

main.tf > resource "aws_subnet" "my_subnet" > cidr_block

```
provider "aws" {
 region = "ap-south-1"
access_key = "AKIAZW6RGWG6KYNVNEET"
secret_key = "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"
}
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.${count.index + 1}.0/24"
 availability_zone = "ap-south-1a"
 map_public_ip_on_launch = true
 tags = {
 Name = "MySubnet-${count.index + 1}"
 }
}
```

3. Creating the VPC and Subnets:

- Apply the Terraform configuration:

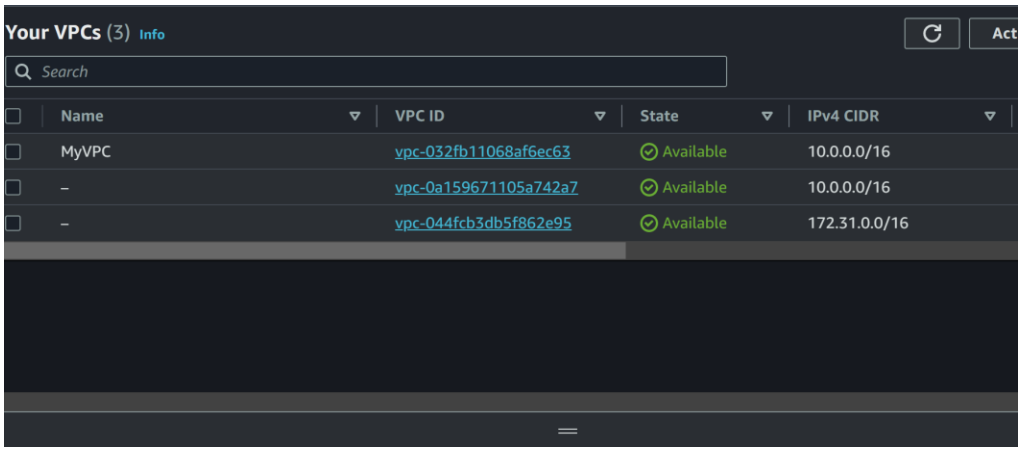
- ```
  vpc.tf > resource "aws_vpc" "gfg-vpc"
  resource "aws_vpc" "gfg-vpc" {
    cidr_block = "10.0.0.0/16"
  }

  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"
    }
  }
  ```

- Confirm the creation of the VPC and subnets by following the Terraform prompts.

- ```
  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_subnet.my_subnet[0]: Creating...
  aws_subnet.my_subnet[1]: Creating...
  aws_subnet.my_subnet[0]: Still creating... [10s elapsed]
  aws_subnet.my_subnet[1]: Still creating... [10s elapsed]
  aws_subnet.my_subnet[0]: Creation complete after 12s [id=subnet-044ab31637
  aws_subnet.my_subnet[1]: Creation complete after 12s [id=subnet-00ee6fb5a2

  Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
  PS E:\terraform-variables\terraformexp8\terraform-vpc>
  ```
-

4. Verifying Resources in the AWS Console:

- Log in to the AWS Management Console and navigate to the VPC service.
- Verify that the VPC named "MyVPC" and two subnets named "MySubnet-1" and "MySubnet-2" have been created with the specified configuration.



6. Cleaning Up:

- When finished, destroy the VPC and subnets:

Bash

terraform destroy

```
aws_subnet.my_subnet[1]: Destruction complete after 1s
aws_vpc.my_vpc: Destroying... [id=vpc-032fb11068af6ec63]
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 0s
aws_internet_gateway.gfg-gw: Destroying... [id=igw-0fa6b97b1a59a4981]
aws_vpc.my_vpc: Destruction complete after 0s
aws_internet_gateway.gfg-gw: Destruction complete after 1s
aws_vpc.gfg-vpc: Destroying... [id=vpc-0a159671105a742a7]
aws_vpc.gfg-vpc: Destruction complete after 1s

Destroy complete! Resources: 9 destroyed.
PS E:\terraform-variables\terraformexp8\terraform-vpc>
```

- Confirm the destruction by typing "yes" at the Terraform prompt.

Conclusion:

This lab exercise demonstrates the basic process of creating a VPC with subnets in AWS using Terraform. Feel free to experiment with different CIDR blocks, configuration options, and additional AWS resources to personalize your VPC infrastructure.