Lesson 11 Demo 04

Working with Collections and Structure Types

Objective: To utilize Terraform collections and structure types for enhanced configuration flexibility and readability

Tools required: Terraform, AWS, and Visual Studio Code

Prerequisites: Refer to the **Demo 01** of **Lesson 11** for creating access and secret key

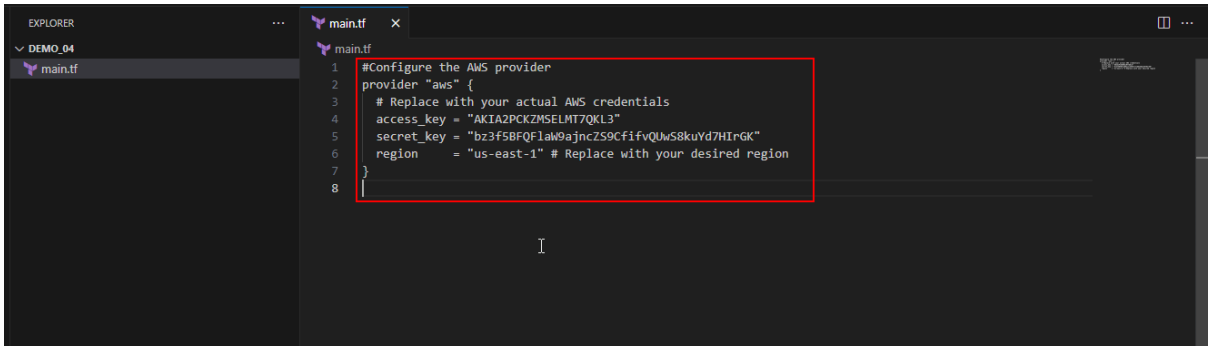
Steps to be followed:

1. Create and reference a new list variable
2. Add a new map variable to replace static values
3. Iterate over a map to create multiple resources
4. Utilize a complex map variable to simplify configuration readability

Step 1: Create and reference a new list variable

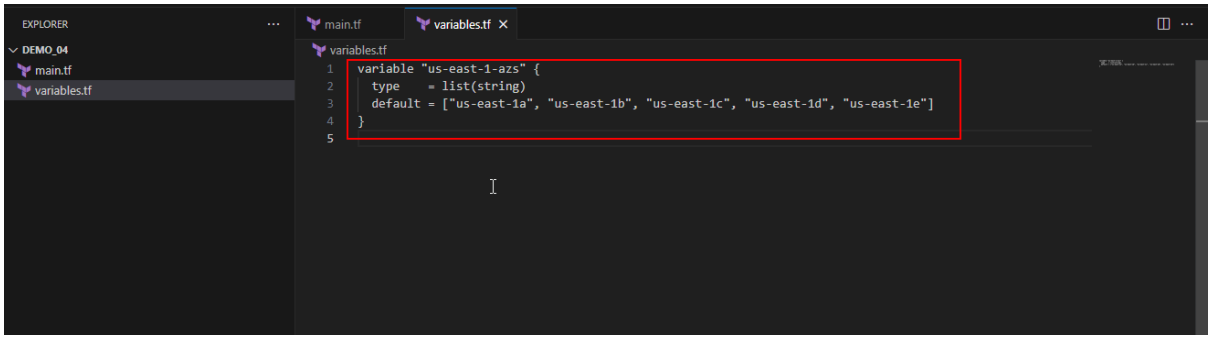
1.1 Open your Terraform configuration environment, create a file named **main.tf**, and add the following configuration block as shown in the screenshot below:

```
#Configure the AWS provider
provider "aws" {
  # Replace with your actual AWS credentials
  access_key = "YOUR_ACCESS_KEY"
  secret_key = "YOUR_SECRET_KEY"
  region     = "us-east-1" # Replace with your desired region
}
```



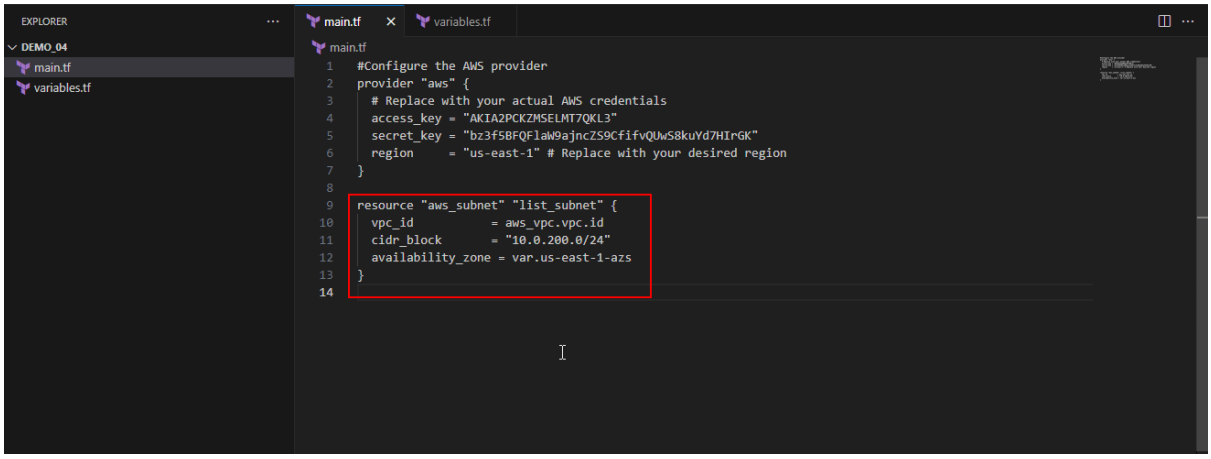
1.2 Create a file named **variables.tf**, and add the following variable to define a list of AWS availability zones:

```
variable "us-east-1-azs" {
  type    = list(string)
  default = ["us-east-1a", "us-east-1b", "us-east-1c", "us-east-1d", "us-east-1e"]
}
```

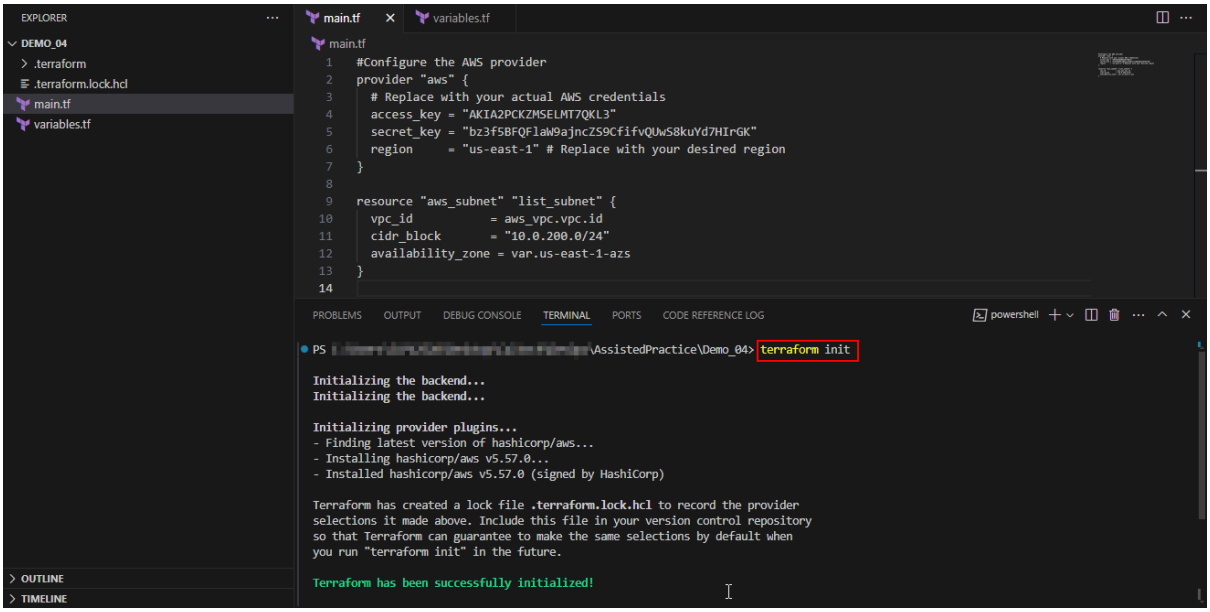


1.3 Utilize the new list variable within a resource definition in the **main.tf** file as shown in the screenshot below:

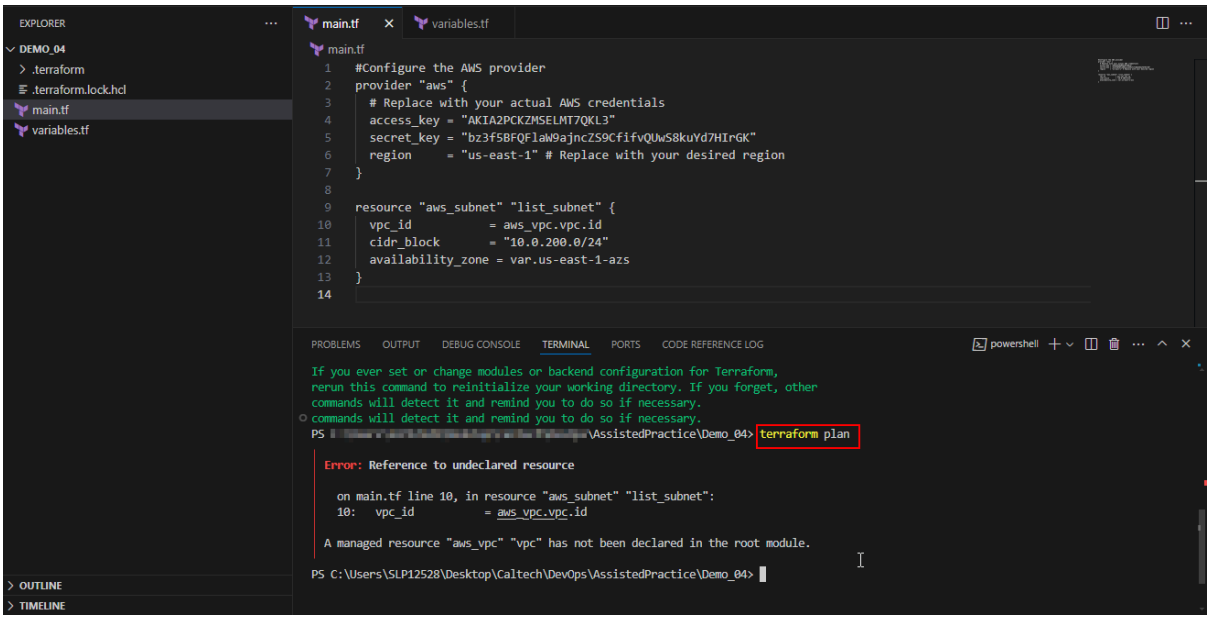
```
resource "aws_subnet" "list_subnet" {
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = "10.0.200.0/24"
  availability_zone = var.us-east-1-azs
}
```



1.4 Initialize the Terraform configuration using the following command:
terraform init

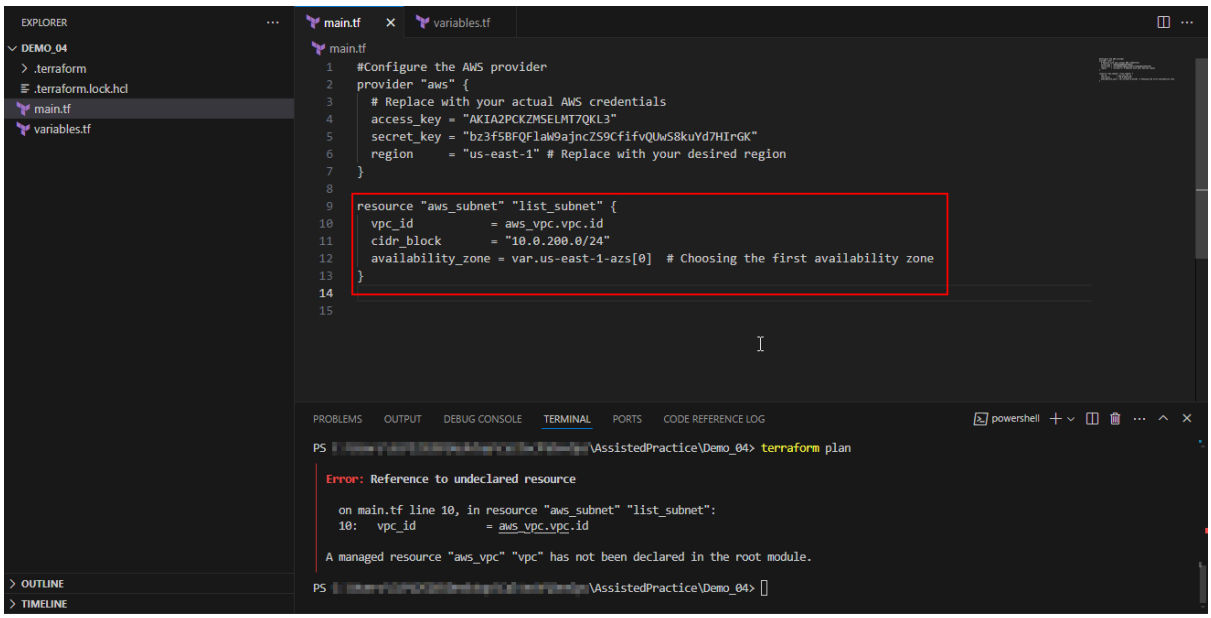


1.5 Execute **terraform plan** to identify errors in usage, noting that **availability_zone** expects a single string, not a list as shown in the screenshot below:



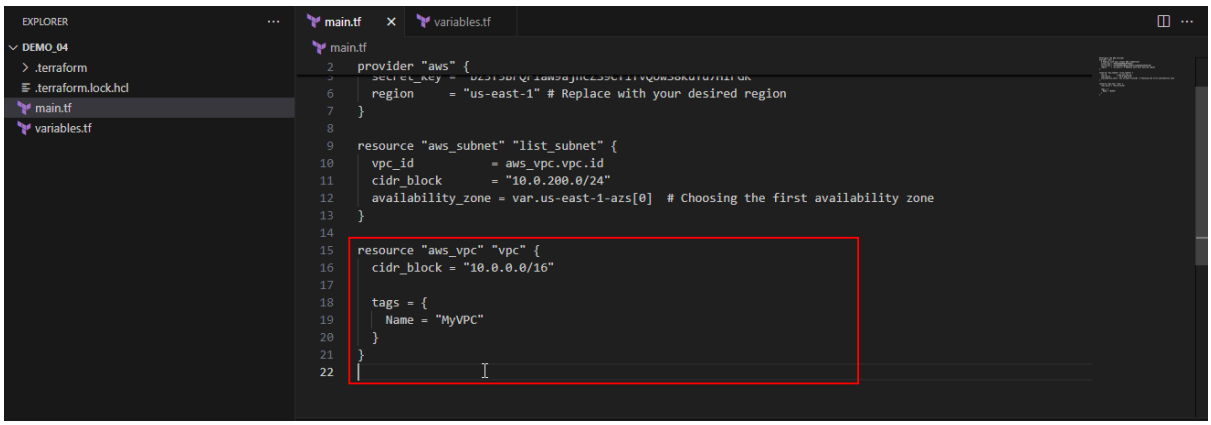
1.6 Correct the resource definition in **main.tf** to specify an element from the list by its index as shown in the screenshot below:

```
resource "aws_subnet" "list_subnet" {
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = "10.0.200.0/24"
  availability_zone = var.us-east-1-azs[0]  # Choosing the first availability zone
}
```

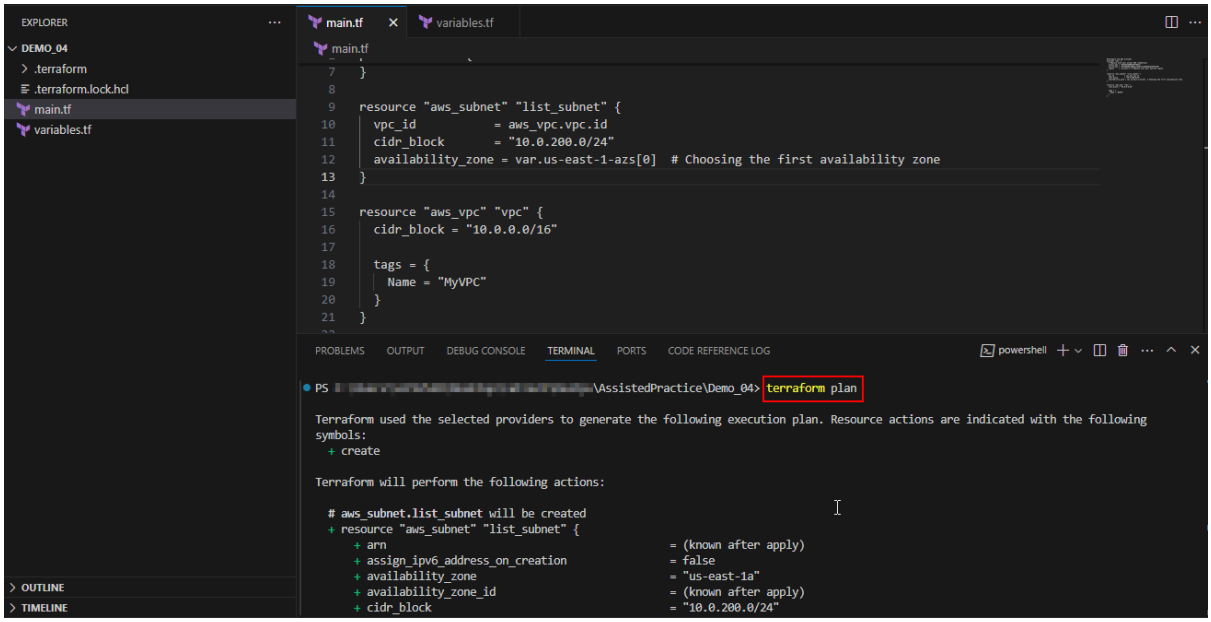


1.7 Add a declaration for the AWS VPC before defining subnets as shown in the screenshot below:

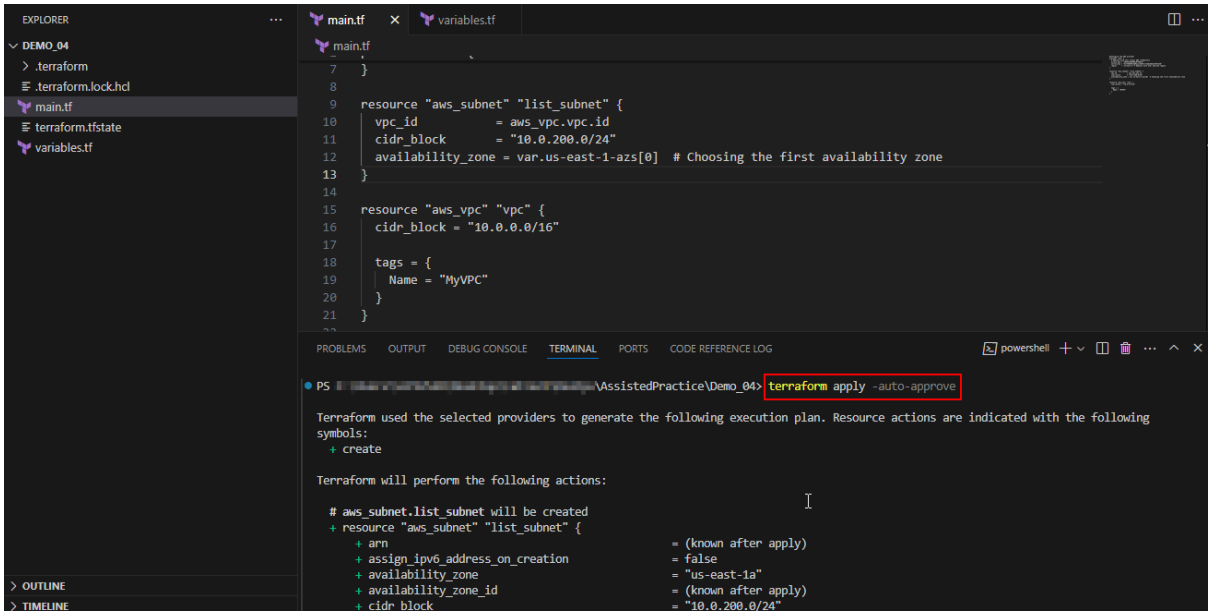
```
resource "aws_vpc" "vpc" {
  cidr_block = "10.0.0.0/16"

  tags = {
    Name = "MyVPC"
  }
}
```



1.8 Plan the deployment using the following command to see the proposed changes:
terraform plan



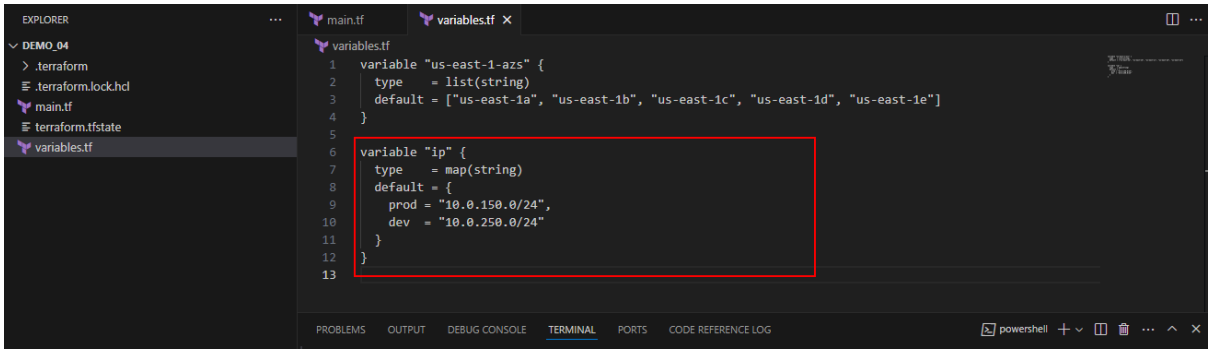
1.9 Apply the configuration using the following command to deploy the changes as shown in the screenshot below:
terraform apply -auto-approve



Step 2: Add a new map variable to replace static values

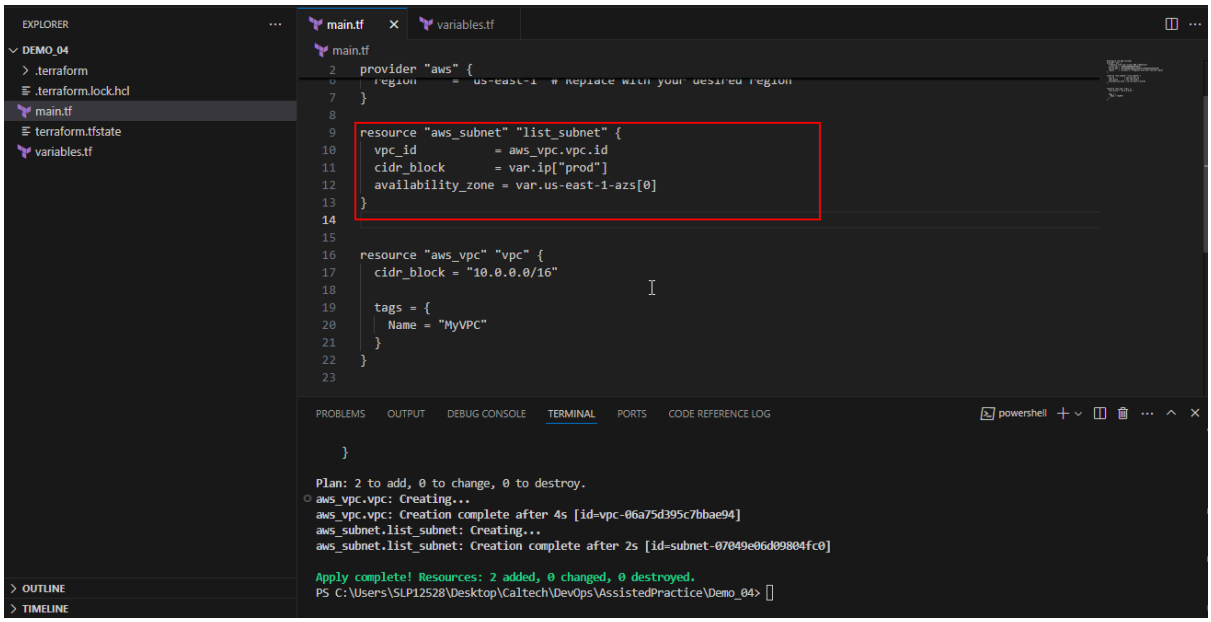
2.1 Update **variables.tf** to include a new map variable for CIDR blocks

```
variable "ip" {
  type    = map(string)
  default = {
    prod = "10.0.150.0/24",
    dev  = "10.0.250.0/24"
  }
}
```

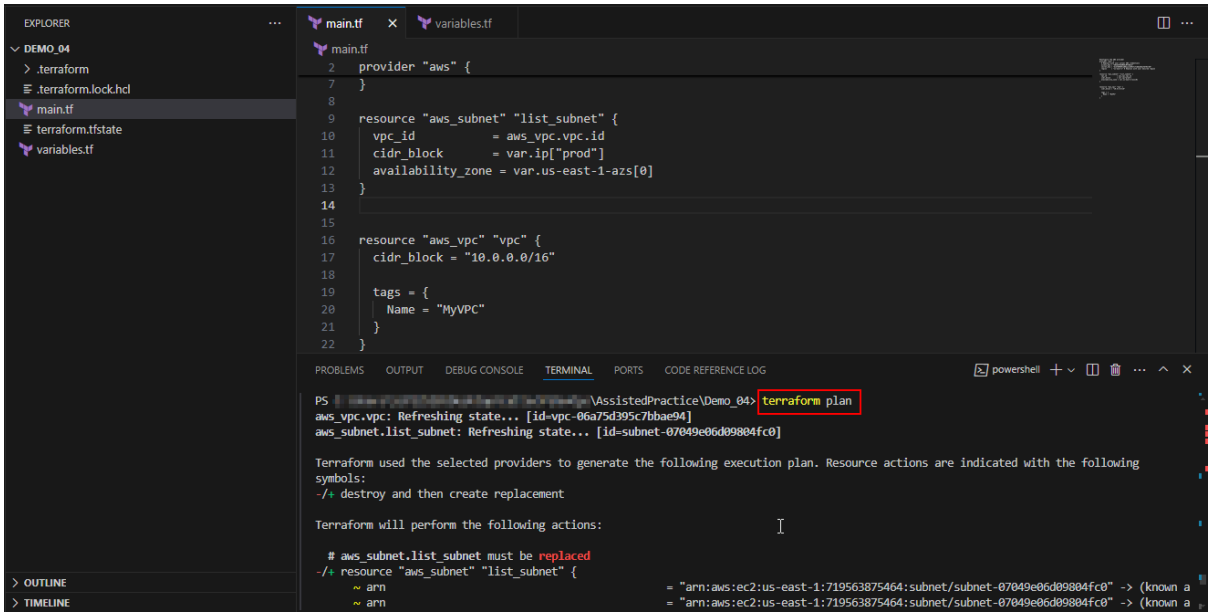


2.2 Modify the subnet resource in **main.tf** to use this map variable:

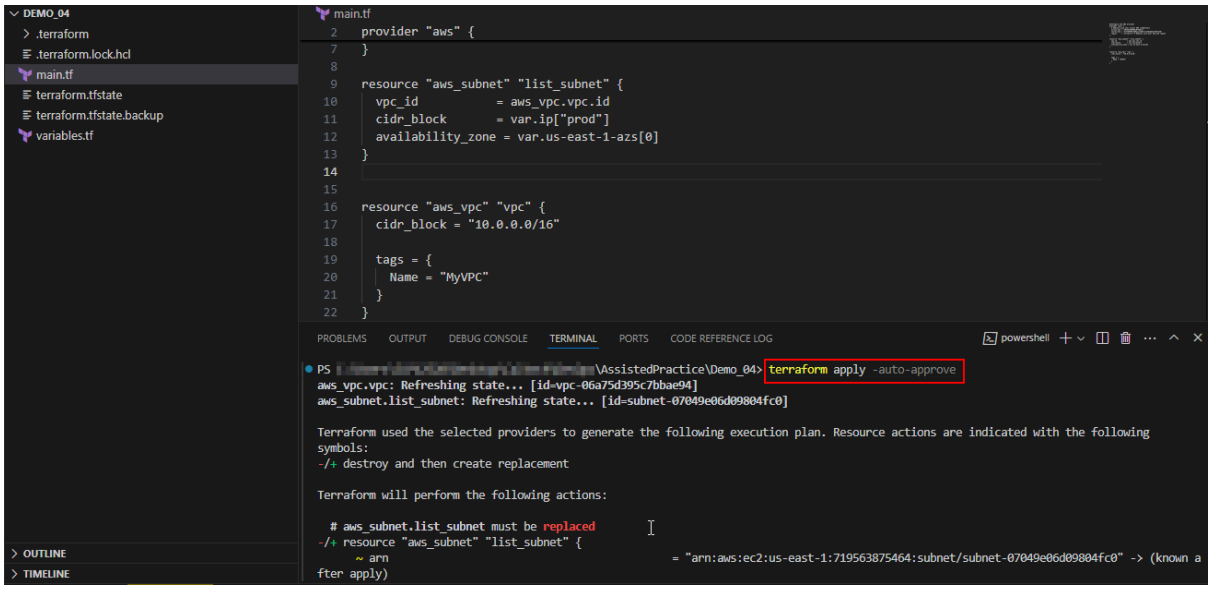
```
resource "aws_subnet" "list_subnet" {
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = var.ip["prod"]
  availability_zone = var.us-east-1-azs[0]
}
```



2.3 Plan the deployment using the following command to see the proposed changes:
terraform plan



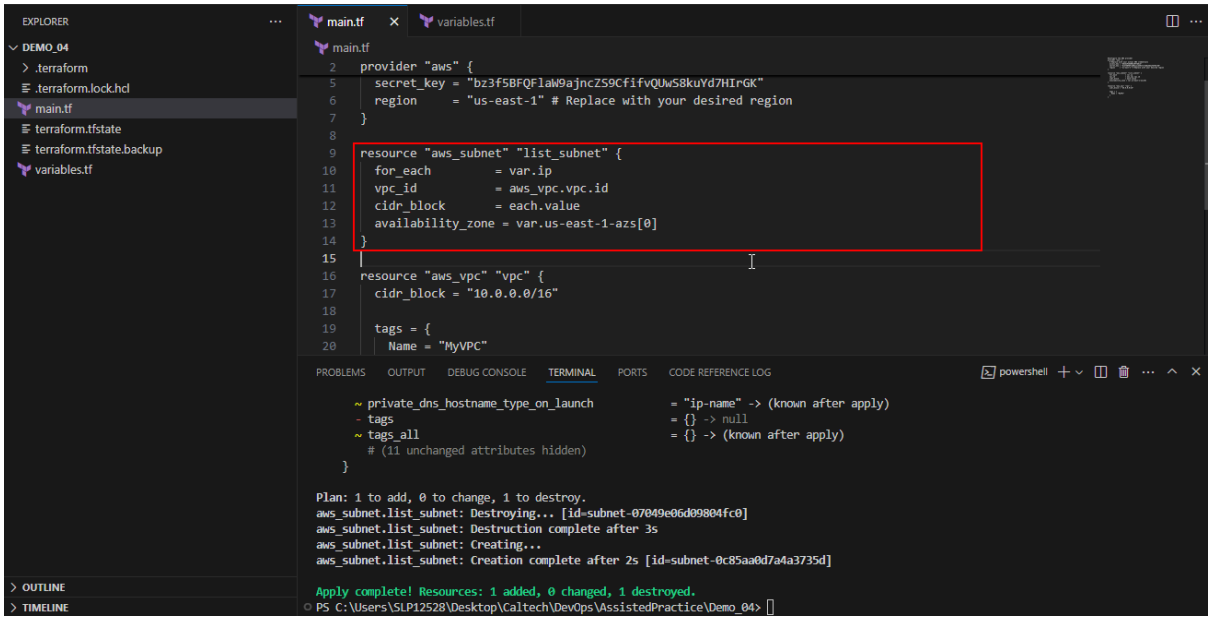
2.4 Apply the configuration using the following command to deploy the changes as shown in the screenshot below:
terraform apply -auto-approve



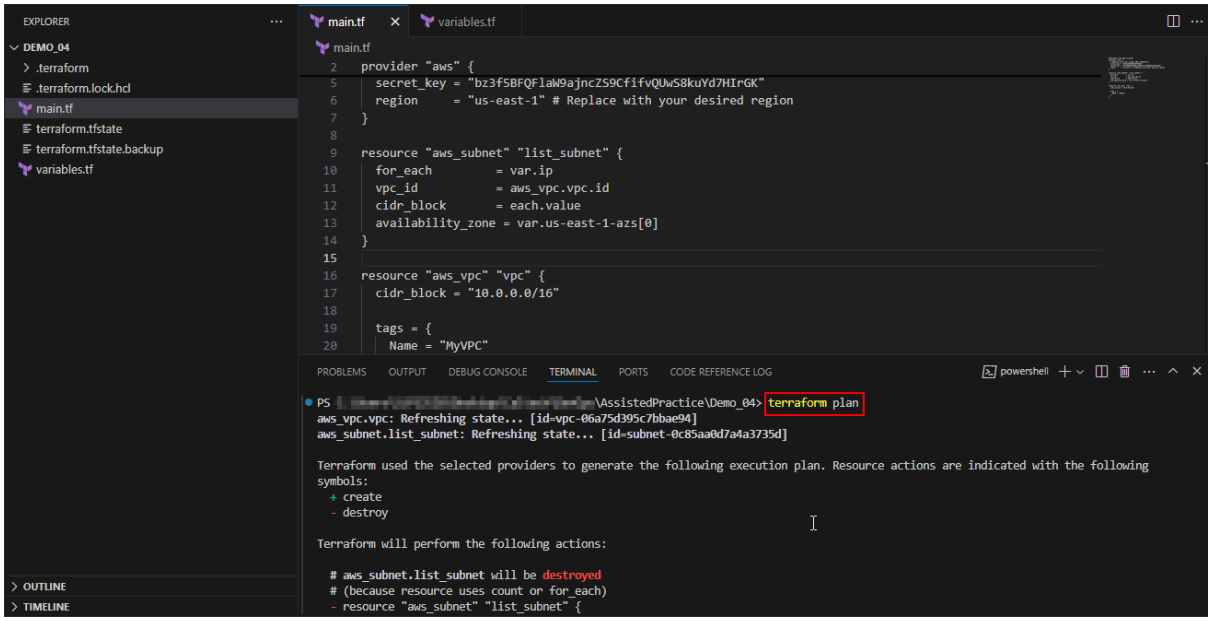
Step 3: Iterate over a map to create multiple resources

3.1 Adjust the subnet resource in **main.tf** to iterate over the map using **for_each:**

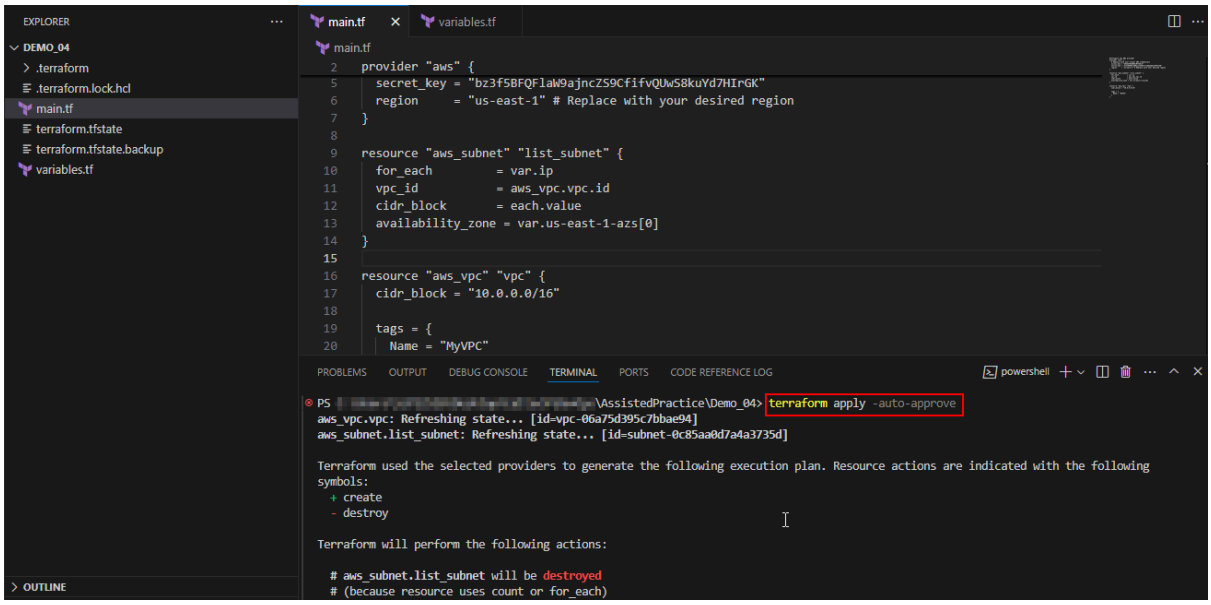
```
resource "aws_subnet" "list_subnet" {
  for_each        = var.ip
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = each.value
  availability_zone = var.us-east-1-azs[0]
}
```



3.2 Plan the deployment using the following command to see the proposed changes:
terraform plan



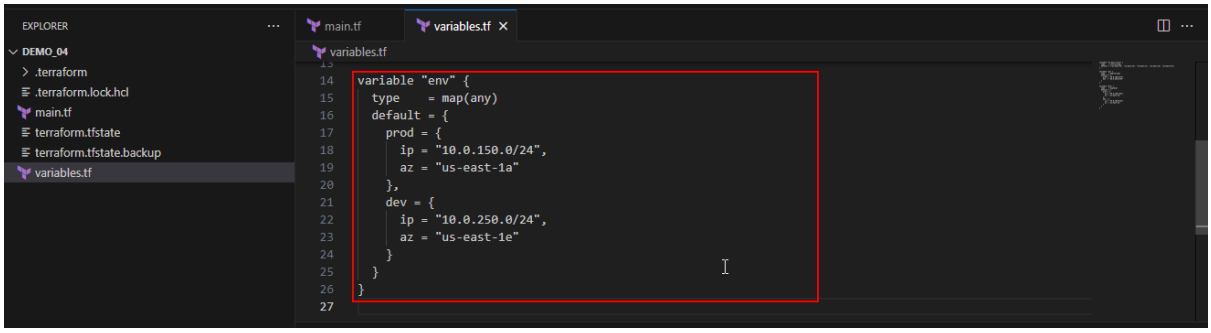
3.3 Apply the configuration using the following command to deploy the changes as shown in the screenshot below:
terraform apply -auto-approve



Step 4: Utilize a complex map variable to simplify configuration readability

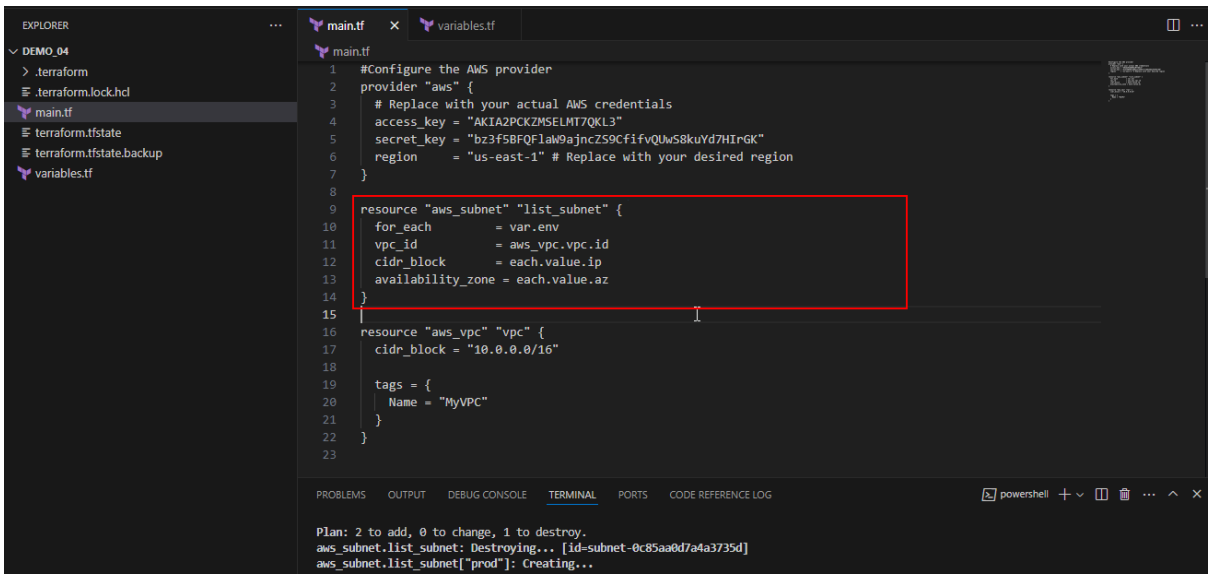
4.1 Add a complex map of maps in **variables.tf**:

```
variable "env" {
  type    = map(any)
  default = {
    prod = {
      ip = "10.0.150.0/24",
      az = "us-east-1a"
    },
    dev = {
      ip = "10.0.250.0/24",
      az = "us-east-1e"
    }
  }
}
```

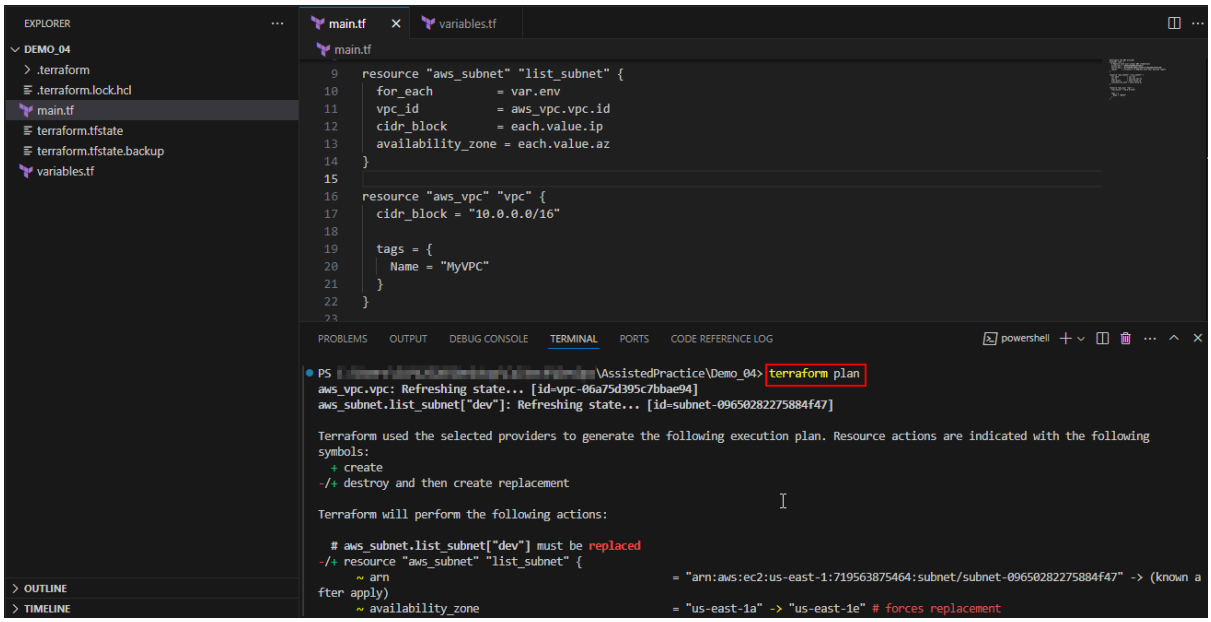


4.2 Modify the subnet definition in **main.tf** to use the new structure

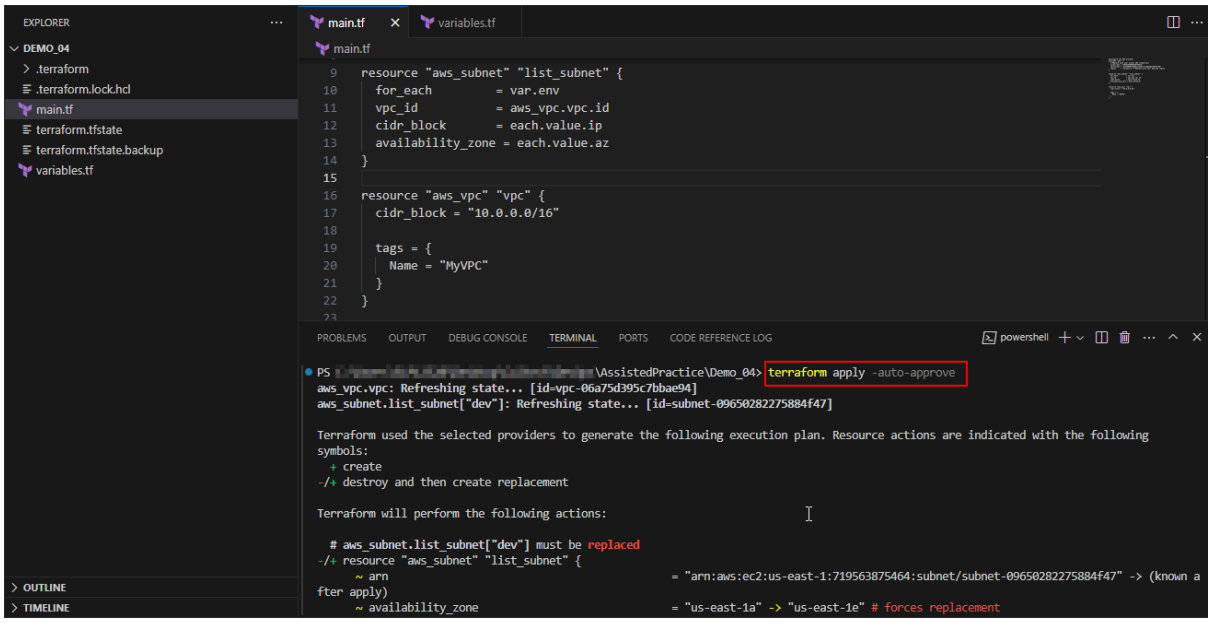
```
resource "aws_subnet" "list_subnet" {
  for_each        = var.env
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = each.value.ip
  availability_zone = each.value.az
}
```



4.3 Plan the deployment using the following command to see the proposed changes:
terraform plan



4.4 Apply the configuration using the following command to deploy the changes as shown in the screenshot below:
terraform apply -auto-approve



By following these steps, you have successfully utilized Terraform collections and structure types to enhance your configuration's flexibility and readability.