

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

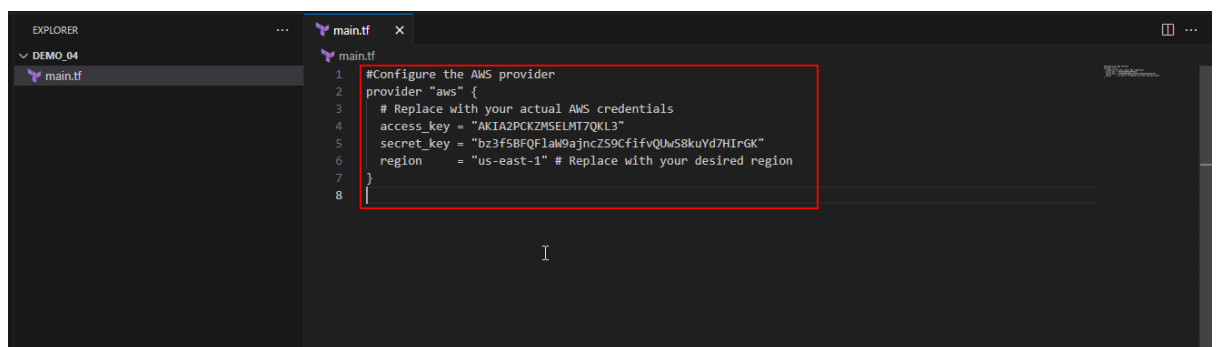
Steps to be followed:

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

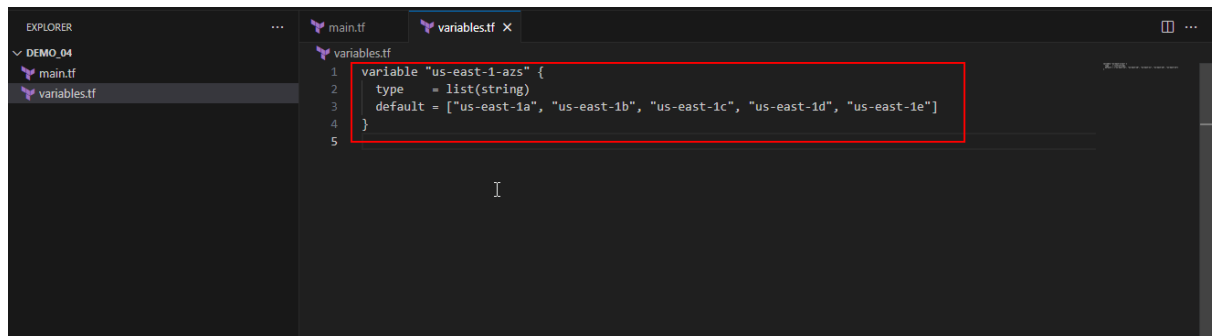
- 1.1 Open your Terraform configuration environment, create a file named **main.tf**, and add the following configuration block as shown in the screenshot below:

```
#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
}
```



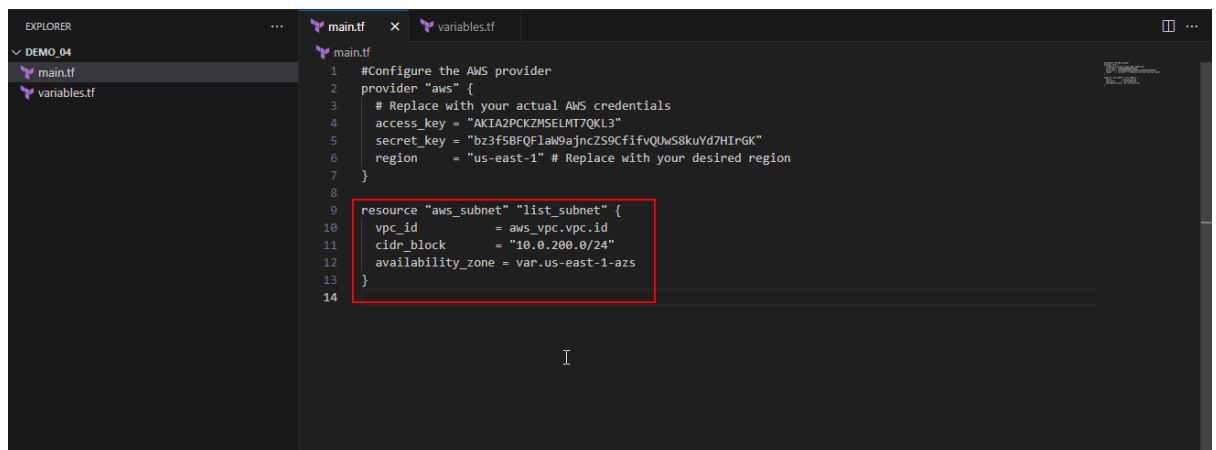
- 1.2 Create a file named **variables.tf**, and add the following variable to define a list of AWS availability zones:

```
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**

The screenshot shows the Visual Studio Code interface with a file explorer on the left showing a project named 'DEMO_04' containing '.terraform', '.terraform.lock.hcl', 'main.tf', and 'variables.tf'. The 'main.tf' file is open in the editor, showing Terraform configuration for the AWS provider and an 'aws_subnet' resource. The terminal at the bottom shows the command 'terraform init' being executed, with output indicating the backend and provider plugins are being initialized. The output concludes with 'Terraform has been successfully initialized!'.

```
1 #Configure the AWS provider
2 provider "aws" {
3   # Replace with your actual AWS credentials
4   access_key = "AKIA2PCKZMSELM7QKL3"
5   secret_key = "bz3f5BFQFlaw9ajncZS9CfifvQUwS8kuVd7HirGK"
6   region     = "us-east-1" # Replace with your desired region
7 }
8
9 resource "aws_subnet" "list_subnet" {
10  vpc_id       = aws_vpc.vpc.id
11  cidr_block   = "10.0.200.0/24"
12  availability_zone = var.us-east-1-azs
13 }
14
```

```
PS C:\Users\SLP12528\Desktop\Caltech\DevOps\AssistedPractice\Demo_04> terraform init

Initializing the backend...
Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.57.0...
- Installed hashicorp/aws v5.57.0 (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!
```

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:

The screenshot shows the Visual Studio Code interface with the same project and files as the previous screenshot. The 'main.tf' file is open, showing the same configuration. The terminal at the bottom shows the command 'terraform plan' being executed. The output displays an error message: 'Error: Reference to undeclared resource' on line 10 of 'main.tf', stating that the resource 'aws_vpc.vpc.id' has not been declared in the root module.

```
1 #Configure the AWS provider
2 provider "aws" {
3   # Replace with your actual AWS credentials
4   access_key = "AKIA2PCKZMSELM7QKL3"
5   secret_key = "bz3f5BFQFlaw9ajncZS9CfifvQUwS8kuVd7HirGK"
6   region     = "us-east-1" # Replace with your desired region
7 }
8
9 resource "aws_subnet" "list_subnet" {
10  vpc_id       = aws_vpc.vpc.id
11  cidr_block   = "10.0.200.0/24"
12  availability_zone = var.us-east-1-azs
13 }
14
```

```
PS C:\Users\SLP12528\Desktop\Caltech\DevOps\AssistedPractice\Demo_04> terraform plan

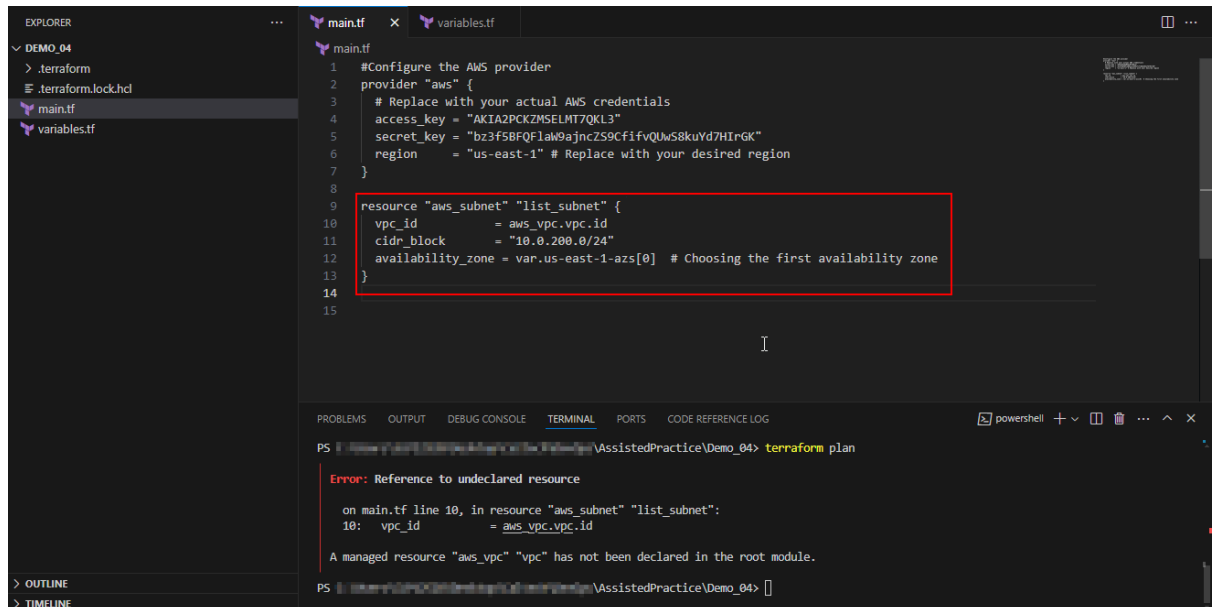
Error: Reference to undeclared resource

   on main.tf line 10, in resource "aws_subnet" "list_subnet":
    10:   vpc_id       = aws_vpc.vpc.id

A managed resource "aws_vpc" "vpc" has not been declared in the root module.
```

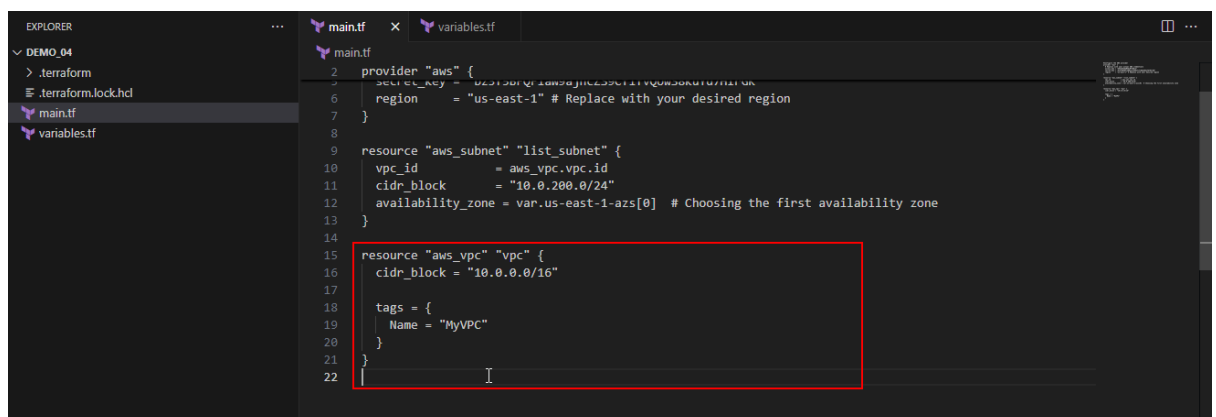
- 1.6 Correct the resource definition in **main.tf** to specify an element from the list by its index as shown in the screenshot below:

```
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  
}
```

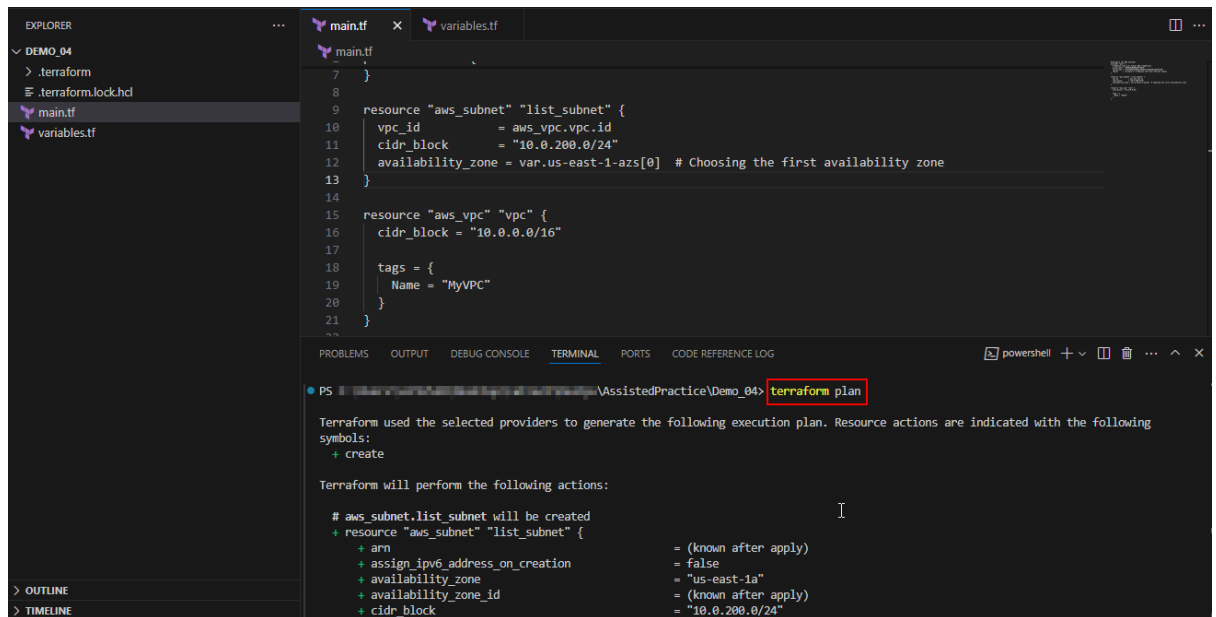


- 1.7 Add a declaration for the AWS VPC before defining subnets as shown in the screenshot below:

```
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



The screenshot shows a Visual Studio Code editor with two files open: `main.tf` and `variables.tf`. The `main.tf` file contains Terraform configuration for an AWS VPC and a subnet. The terminal window shows the command `terraform plan` being executed. The output indicates that Terraform will create an `aws_subnet` resource named `list_subnet` and an `aws_vpc` resource named `vpc`. The output also shows the values for various attributes, including `arn`, `assign_ipv6_address_on_creation`, `availability_zone`, `availability_zone_id`, and `cidr_block`.

```
7 }
8
9 resource "aws_subnet" "list_subnet" {
10   vpc_id      = aws_vpc.vpc.id
11   cidr_block   = "10.0.200.0/24"
12   availability_zone = var.us-east-1-azs[0] # Choosing the first availability zone
13 }
14
15 resource "aws_vpc" "vpc" {
16   cidr_block = "10.0.0.0/16"
17
18   tags = {
19     Name = "MyVPC"
20   }
21 }
```

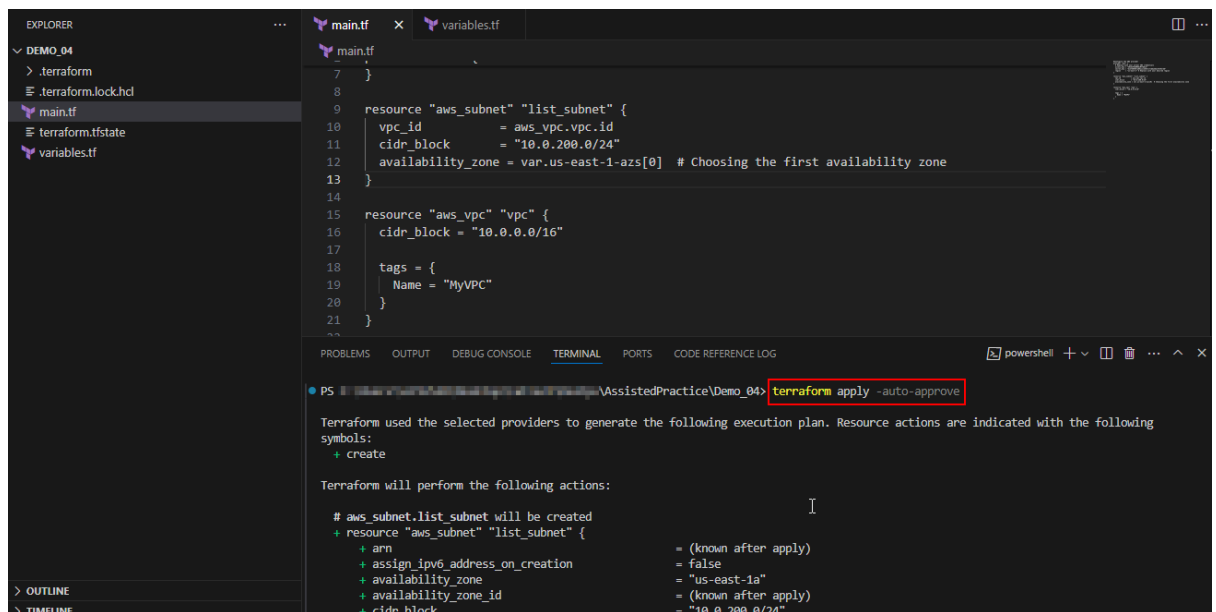
```
PS C:\AssistedPractice\Demo_04> terraform plan

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.list_subnet will be created
+ resource "aws_subnet" "list_subnet" {
+   arn                                     = (known after apply)
+   assign_ipv6_address_on_creation         = false
+   availability_zone                       = "us-east-1a"
+   availability_zone_id                   = (known after apply)
+   cidr_block                             = "10.0.200.0/24"
```

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



The screenshot shows the same Visual Studio Code editor with the `main.tf` and `variables.tf` files. The terminal window now shows the command `terraform apply -auto-approve` being executed. The output is identical to the previous screenshot, showing the plan for creating the `aws_subnet` and `aws_vpc` resources.

```
PS C:\AssistedPractice\Demo_04> terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
+ create

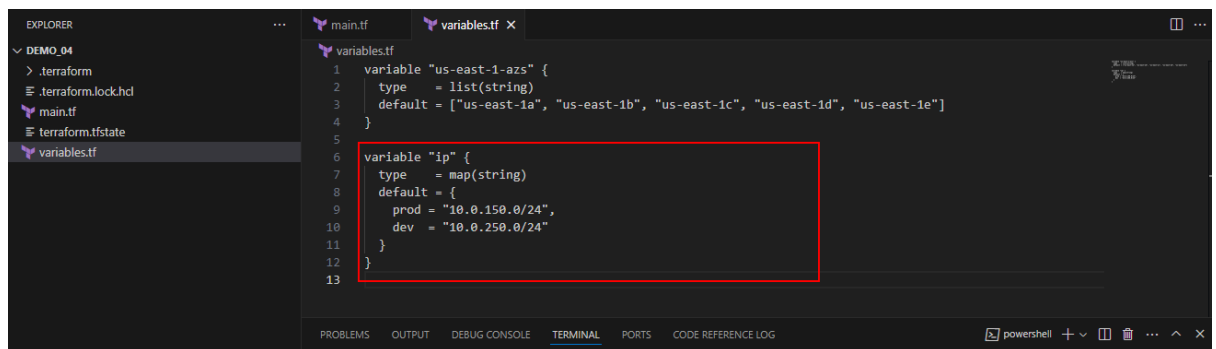
Terraform will perform the following actions:

# aws_subnet.list_subnet will be created
+ resource "aws_subnet" "list_subnet" {
+   arn                                     = (known after apply)
+   assign_ipv6_address_on_creation         = false
+   availability_zone                       = "us-east-1a"
+   availability_zone_id                   = (known after apply)
+   cidr_block                             = "10.0.200.0/24"
```

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

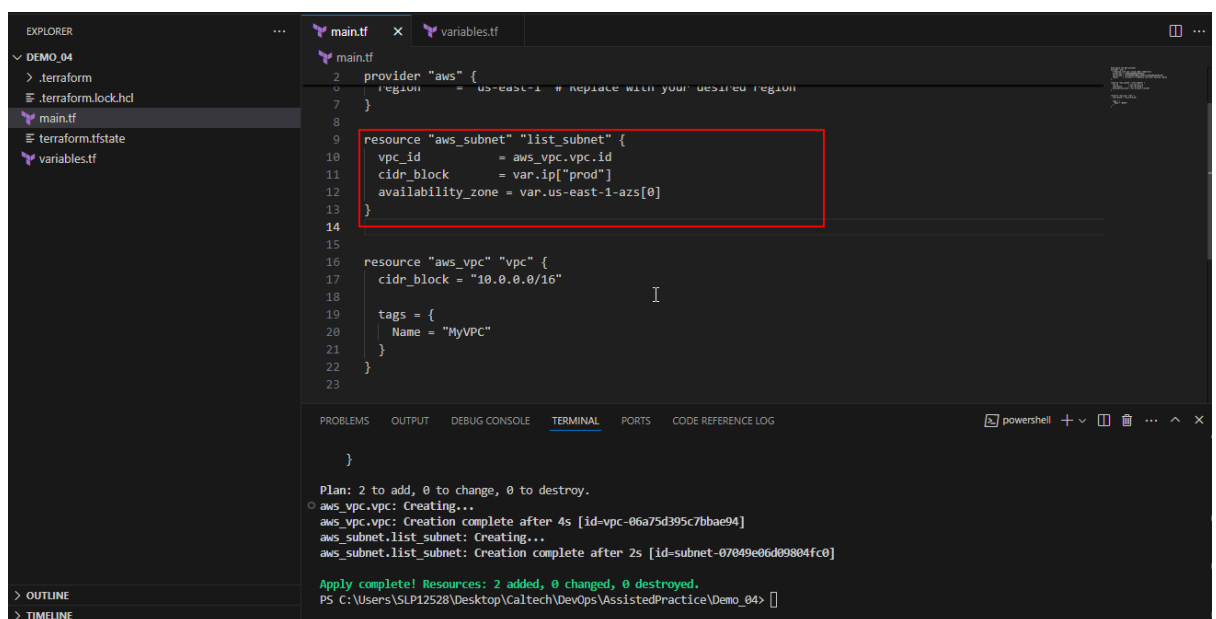
2.1 Update **variables.tf** to include a new map variable for CIDR blocks

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

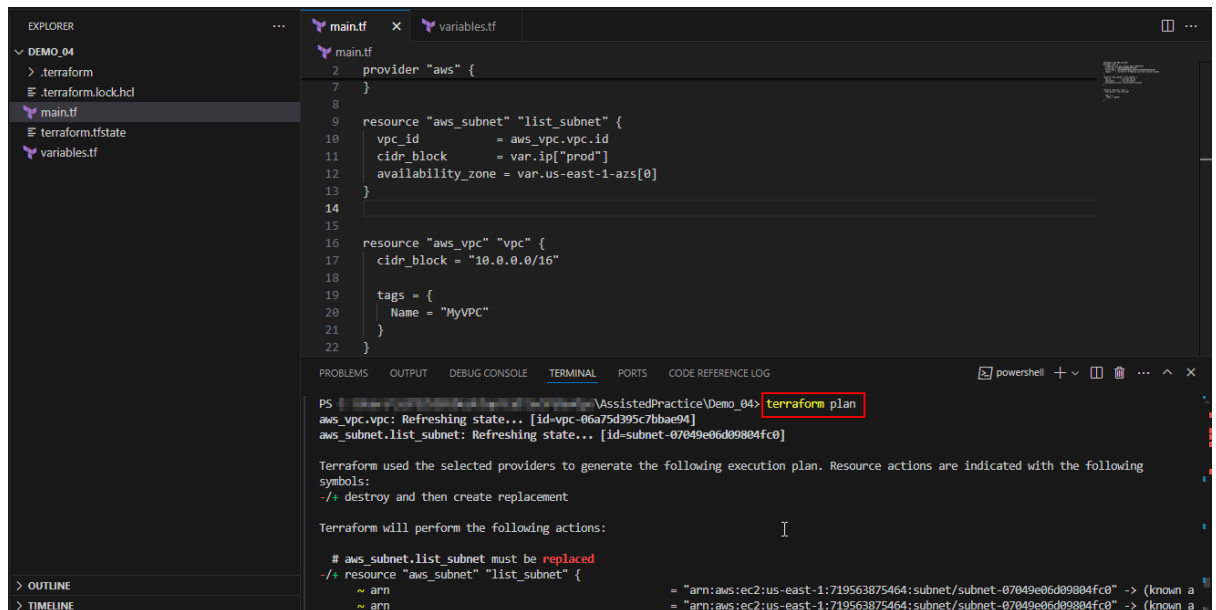


2.2 Modify the subnet resource in **main.tf** to use this map variable:

```
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**



The screenshot shows a Visual Studio Code editor with a Terraform configuration file named `main.tf`. The configuration defines an AWS VPC and a subnet. The terminal window displays the output of the `terraform plan` command, which shows the state of the resources and the actions Terraform will perform.

```
main.tf
2 provider "aws" {
7 }
8
9 resource "aws_subnet" "list_subnet" {
10   vpc_id      = aws_vpc.vpc.id
11   cidr_block  = var.ip["prod"]
12   availability_zone = var.us-east-1-azs[0]
13 }
14
15
16 resource "aws_vpc" "vpc" {
17   cidr_block = "10.0.0.0/16"
18
19   tags = {
20     Name = "MyVPC"
21   }
22 }
```

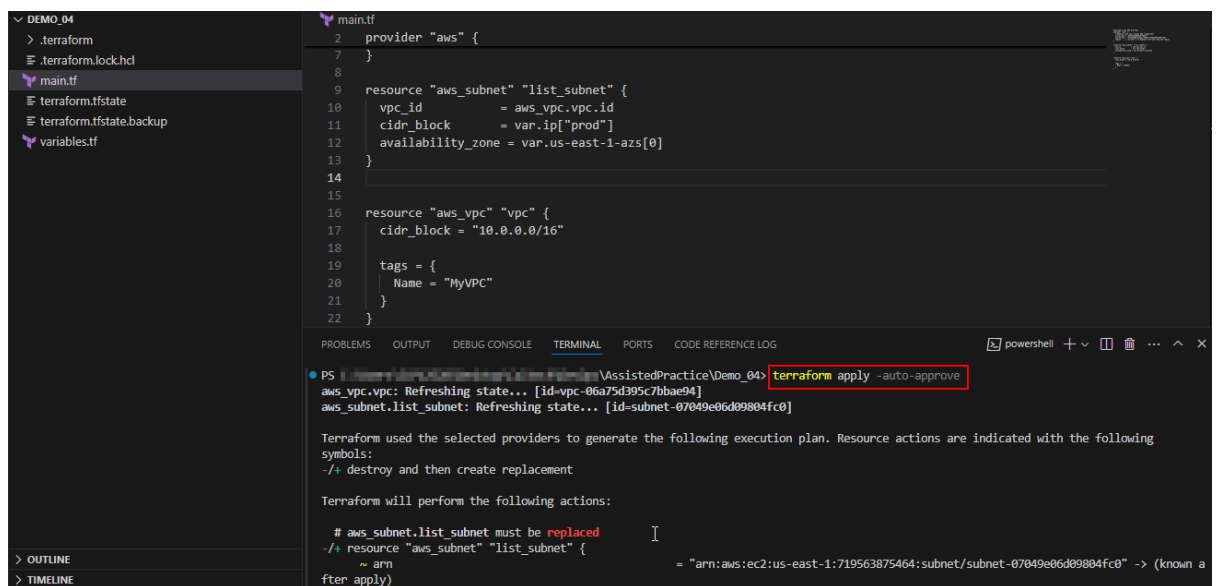
```
PS C:\AssistedPractice\Demo_04> terraform plan
aws_vpc.vpc: Refreshing state... [id=vpc-06a75d395c7bbae94]
aws_subnet.list_subnet: Refreshing state... [id=subnet-07049e06d09804fc0]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

# aws_subnet.list_subnet must be replaced
/+ resource "aws_subnet" "list_subnet" {
  ~ arn              = "arn:aws:ec2:us-east-1:719563875464:subnet/subnet-07049e06d09804fc0" -> (known a
  ~ arn              = "arn:aws:ec2:us-east-1:719563875464:subnet/subnet-07049e06d09804fc0" -> (known a
```

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



The screenshot shows the same Visual Studio Code editor with the `main.tf` file. The terminal window displays the output of the `terraform apply -auto-approve` command, which shows the state of the resources and the actions Terraform will perform, including the creation of the VPC and subnet.

```
main.tf
2 provider "aws" {
7 }
8
9 resource "aws_subnet" "list_subnet" {
10   vpc_id      = aws_vpc.vpc.id
11   cidr_block  = var.ip["prod"]
12   availability_zone = var.us-east-1-azs[0]
13 }
14
15
16 resource "aws_vpc" "vpc" {
17   cidr_block = "10.0.0.0/16"
18
19   tags = {
20     Name = "MyVPC"
21   }
22 }
```

```
PS C:\AssistedPractice\Demo_04> terraform apply -auto-approve
aws_vpc.vpc: Refreshing state... [id=vpc-06a75d395c7bbae94]
aws_subnet.list_subnet: Refreshing state... [id=subnet-07049e06d09804fc0]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

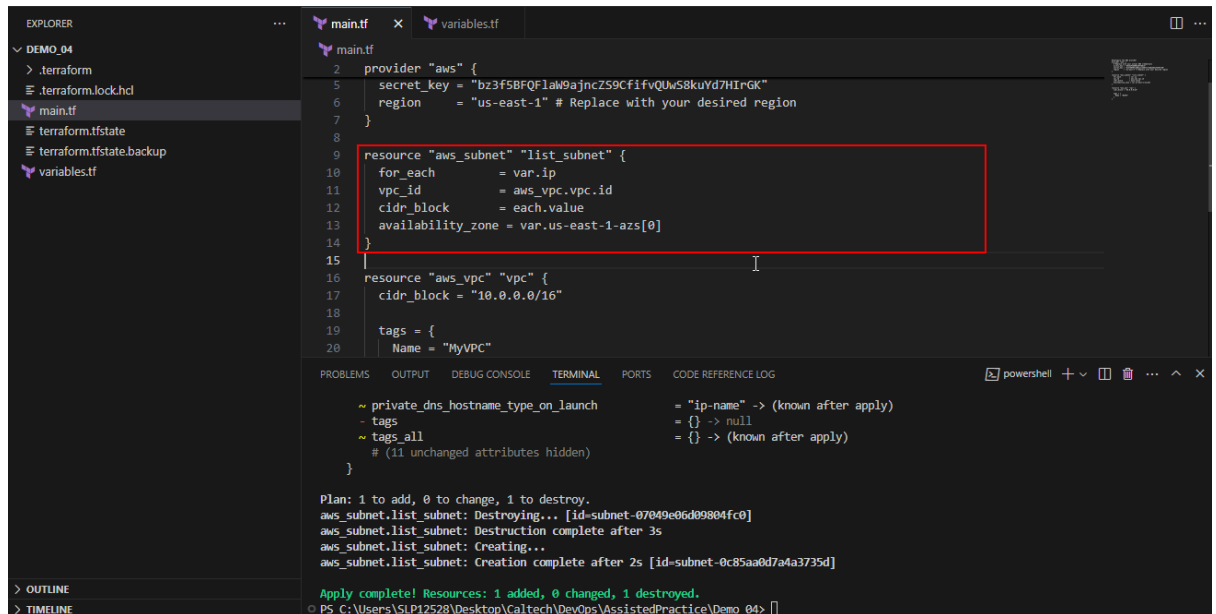
Terraform will perform the following actions:

# aws_subnet.list_subnet must be replaced
/+ resource "aws_subnet" "list_subnet" {
  ~ arn              = "arn:aws:ec2:us-east-1:719563875464:subnet/subnet-07049e06d09804fc0" -> (known a
  ~ arn              = "arn:aws:ec2:us-east-1:719563875464:subnet/subnet-07049e06d09804fc0" -> (known a
  after apply)
```

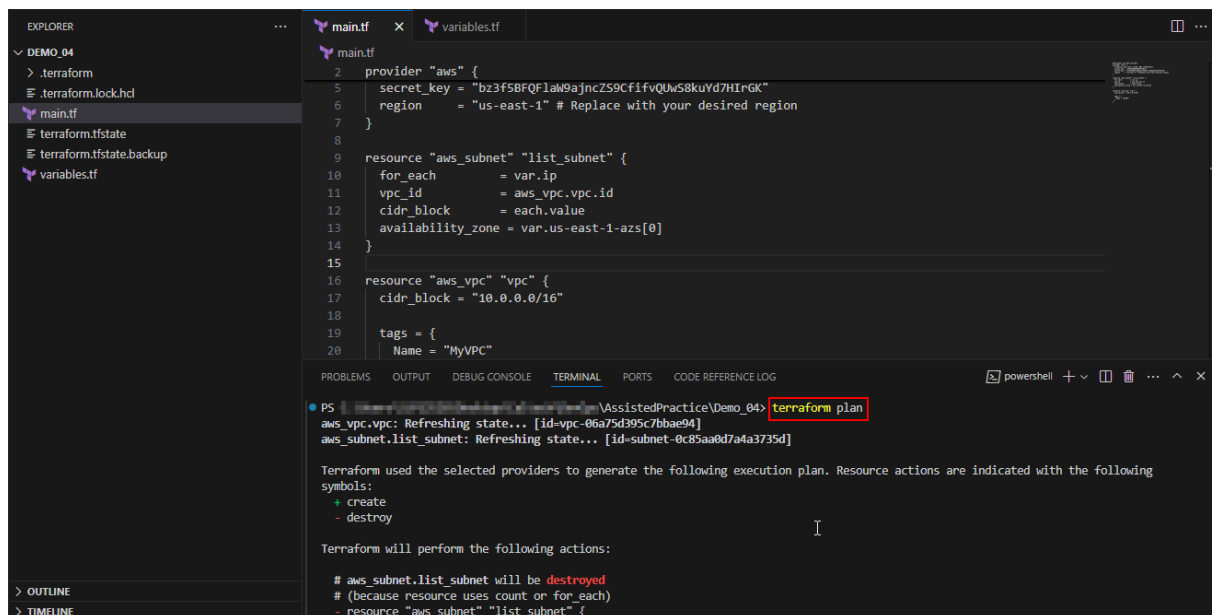
Step 3: Iterate over a map to create multiple resources

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

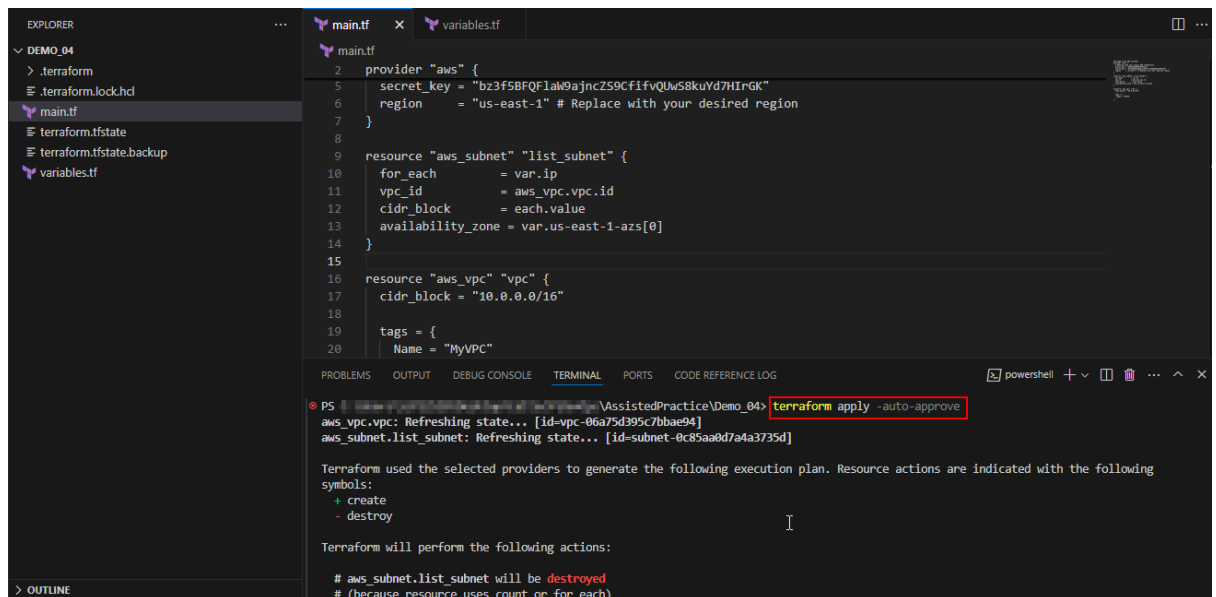
```
resource "aws_subnet" "list_subnet" {  
  for_each      = var.ip  
  vpc_id        = aws_vpc.vpc.id  
  cidr_block    = each.value  
  availability_zone = var.us-east-1-azs[0]  
}
```



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



The screenshot shows a VS Code editor with two files open: `main.tf` and `variables.tf`. The `main.tf` file contains Terraform configuration for an AWS provider, a VPC, and a list of subnets. The terminal window at the bottom shows the command `terraform apply -auto-approve` being executed. The output indicates that the state is being refreshed for the VPC and subnets, and then Terraform will perform the following actions: create the VPC and destroy the subnets (because they use count or for_each).

```
main.tf
2 provider "aws" {
3   secret_key = "bz3f5BFQFlaw9ajncZ59CfifvQUw58kuYd7HirGK"
4   region    = "us-east-1" # Replace with your desired region
5 }
6
7 resource "aws_subnet" "list_subnet" {
8   for_each = var.ip
9   vpc_id   = aws_vpc.vpc.id
10  cidr_block = each.value
11  availability_zone = var.us-east-1-azs[0]
12 }
13
14 resource "aws_vpc" "vpc" {
15   cidr_block = "10.0.0.0/16"
16
17   tags = {
18     Name = "MyVPC"
19   }
20 }
```

```
variables.tf
14 variable "env" {
15   type = map(any)
16   default = {
17     prod = {
18       ip = "10.0.150.0/24",
19       az = "us-east-1a"
20     },
21     dev = {
22       ip = "10.0.250.0/24",
23       az = "us-east-1e"
24     }
25   }
26 }
```

```
terminal
PS C:\AssistedPractice\Demo_04> terraform apply -auto-approve
aws_vpc.vpc: Refreshing state... [id=vpc-06a75d395c7bbae94]
aws_subnet.list_subnet: Refreshing state... [id=subnet-0c85aa8d7a4a3735d]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
+ create
- destroy

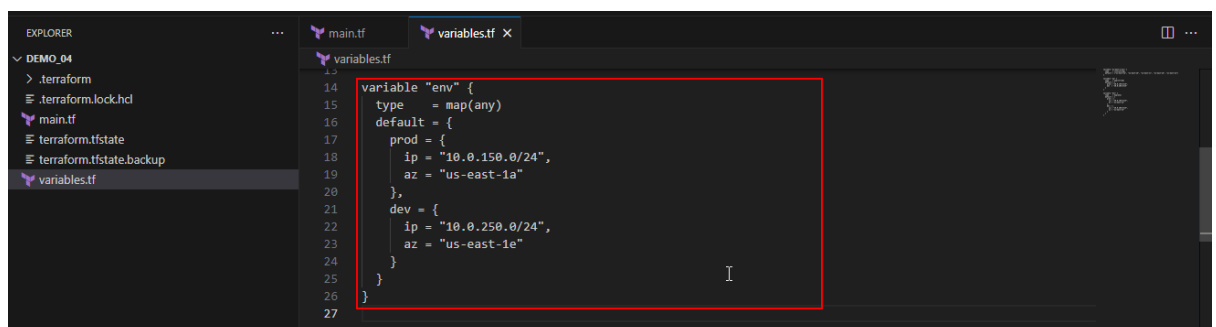
Terraform will perform the following actions:

# aws_subnet.list_subnet will be destroyed
# (because resource uses count or for_each)
```

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"
    }
  }
}
```

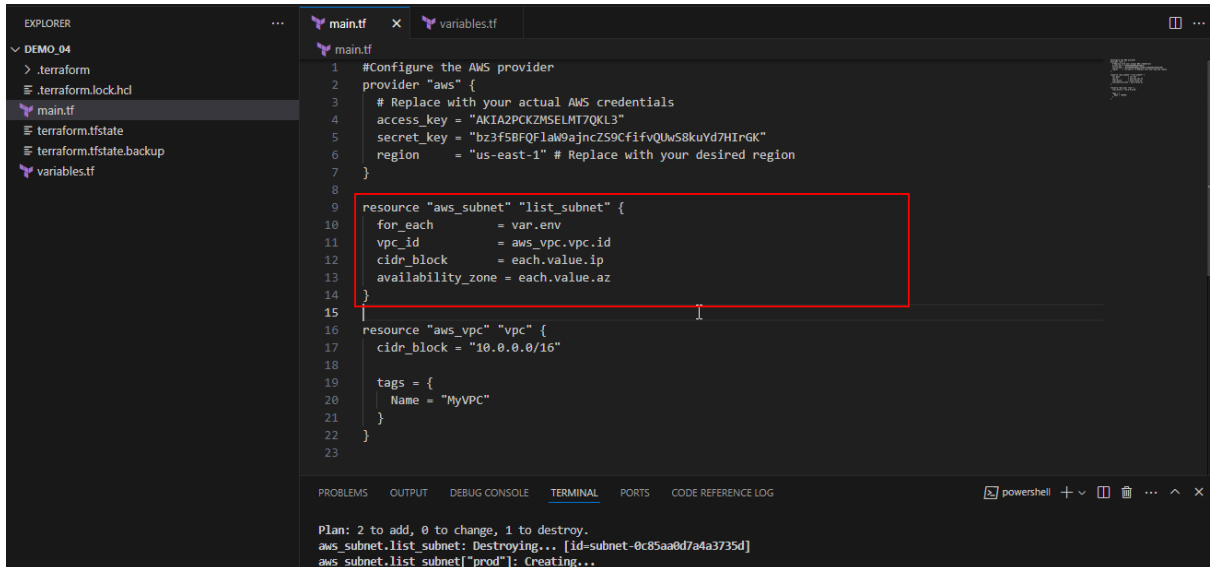


The screenshot shows the `variables.tf` file in the VS Code editor. The file contains a complex map variable named `env` with two entries: `prod` and `dev`. Each entry is a map with `ip` and `az` attributes.

```
variables.tf
14 variable "env" {
15   type = map(any)
16   default = {
17     prod = {
18       ip = "10.0.150.0/24",
19       az = "us-east-1a"
20     },
21     dev = {
22       ip = "10.0.250.0/24",
23       az = "us-east-1e"
24     }
25   }
26 }
```

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  
}
```



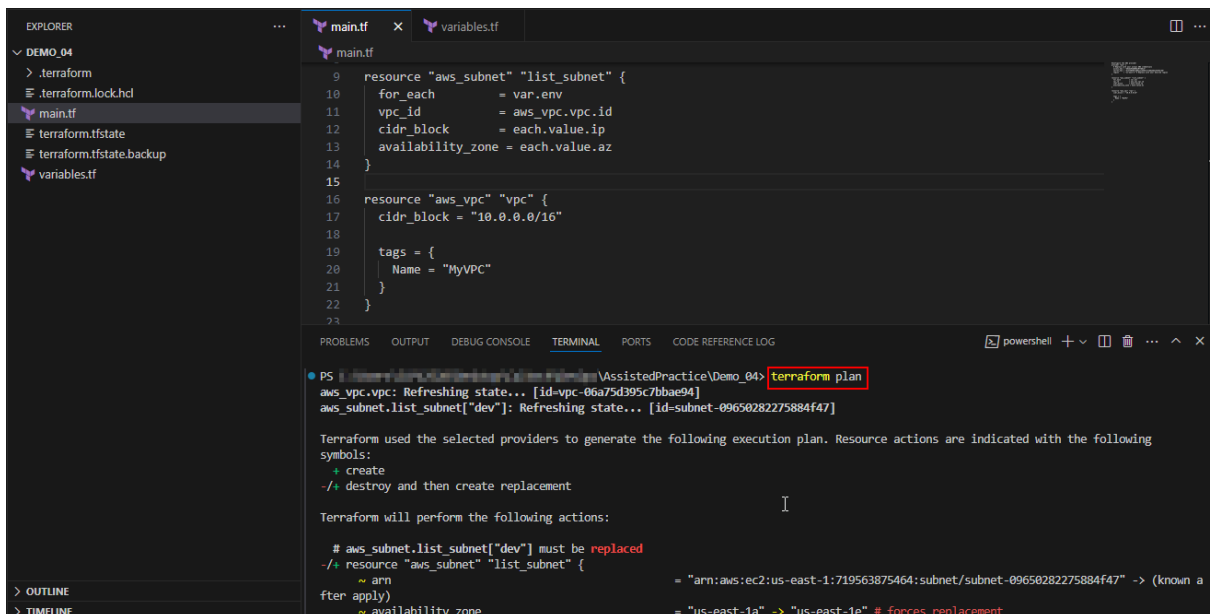
The screenshot shows the Visual Studio Code editor with the **main.tf** file open. The Explorer panel on the left shows the project structure for **DEMO_04**, including **.terraform**, **.terraform.lock.hcl**, **main.tf**, **terraform.tfstate**, **terraform.tfstate.backup**, and **variables.tf**. The **main.tf** file contains the following code:

```
1 #Configure the AWS provider  
2 provider "aws" {  
3   # Replace with your actual AWS credentials  
4   access_key = "AKIA2PCKZMSELM7QKL3"  
5   secret_key = "bz3f5BFQFlak9ajncZS9CflfvQUwS8kuYd7HirGK"  
6   region    = "us-east-1" # Replace with your desired region  
7 }  
8  
9 resource "aws_subnet" "list_subnet" {  
10   for_each      = var.env  
11   vpc_id        = aws_vpc.vpc.id  
12   cidr_block    = each.value.ip  
13   availability_zone = each.value.az  
14 }  
15  
16 resource "aws_vpc" "vpc" {  
17   cidr_block = "10.0.0.0/16"  
18  
19   tags = {  
20     Name = "MyVPC"  
21   }  
22 }  
23
```

The **aws_subnet** resource definition (lines 9-14) is highlighted with a red box. The terminal at the bottom shows the output of the **terraform plan** command:

```
Plan: 2 to add, 0 to change, 1 to destroy.  
aws_subnet.list_subnet: Destroying... [id=subnet-0c85aa8d7a4a3735d]  
aws_subnet.list_subnet["prod"]: Creating...
```

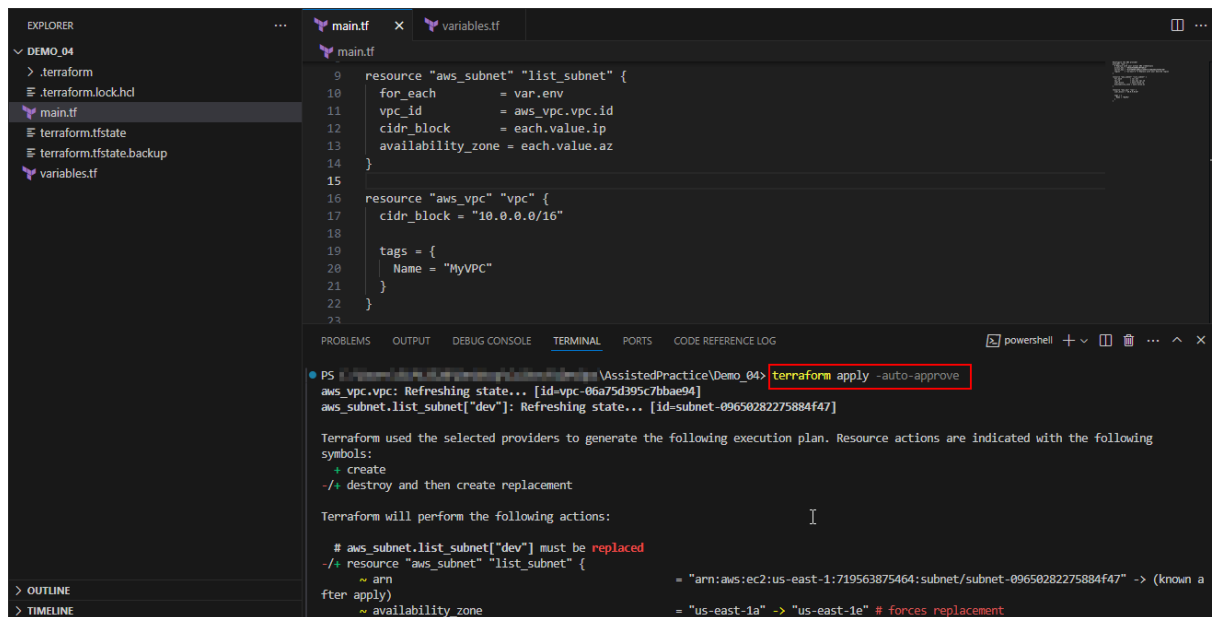
4.3 Plan the deployment using the following command to see the proposed changes: **terraform plan**



The screenshot shows the Visual Studio Code editor with the **main.tf** file open. The terminal at the bottom shows the output of the **terraform plan** command:

```
PS C:\AssistedPractice\Demo_04> terraform plan  
aws_vpc.vpc: Refreshing state... [id=vpc-06a75d395c7bbae94]  
aws_subnet.list_subnet["dev"]: Refreshing state... [id=subnet-09650282275884f47]  
  
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:  
+ create  
-/+ destroy and then create replacement  
  
Terraform will perform the following actions:  
  
# aws_subnet.list_subnet["dev"] must be replaced  
-/+ resource "aws_subnet" "list_subnet" {  
  ~ arn = "arn:aws:ec2:us-east-1:719563875464:subnet/subnet-09650282275884f47" -> (known a  
  ~ after apply)  
  ~ availability_zone = "us-east-1a" -> "us-east-1e" # forces replacement
```

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



The screenshot shows a VS Code editor with two files open: `main.tf` and `variables.tf`. The `main.tf` file contains Terraform configuration for an AWS VPC and a list of subnets. The terminal window shows the command `terraform apply -auto-approve` being executed. The output indicates that the VPC and subnets are being refreshed and then replaced. The terminal also shows the execution plan, indicating that the subnets will be replaced.

```
9 resource "aws_subnet" "list_subnet" {
10   for_each = var.env
11   vpc_id   = aws_vpc.vpc.id
12   cidr_block = each.value.ip
13   availability_zone = each.value.az
14 }
15
16 resource "aws_vpc" "vpc" {
17   cidr_block = "10.0.0.0/16"
18
19   tags = {
20     Name = "MyVPC"
21   }
22 }
```

```
PS C:\AssistedPractice\Demo_04> terraform apply -auto-approve
aws_vpc.vpc: Refreshing state... [id=vpc-06a75d395c7bbae94]
aws_subnet.list_subnet["dev"]: Refreshing state... [id=subnet-09650282275884f47]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create
  -/+ destroy and then create replacement

Terraform will perform the following actions:

  # aws_subnet.list_subnet["dev"] must be replaced
  -/+ resource "aws_subnet" "list_subnet" {
    ~ arn                                = "arn:aws:ec2:us-east-1:719563875464:subnet/subnet-09650282275884f47" -> (known a
    ~ after apply)
    ~ availability_zone                  = "us-east-1a" -> "us-east-1e" # forces replacement
```

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