

# Lesson 11 Demo 07

## Working with Graphs

**Objective:** To create and visualize Terraform resource dependency graphs for better infrastructure management

**Tools required:** Terraform, AWS, and Visual Studio Code

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

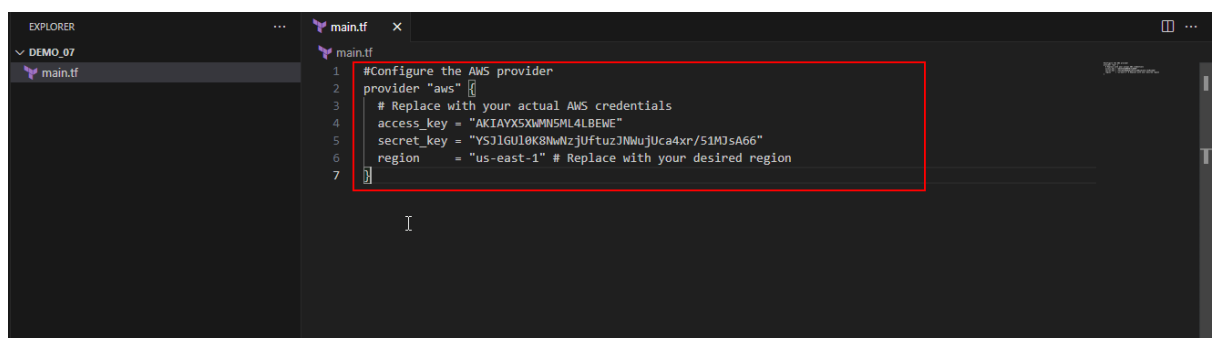
Steps to be followed:

1. Set up infrastructure with dependencies
2. Generate and visualize the resource graph

### Step 1: Set up infrastructure with dependencies

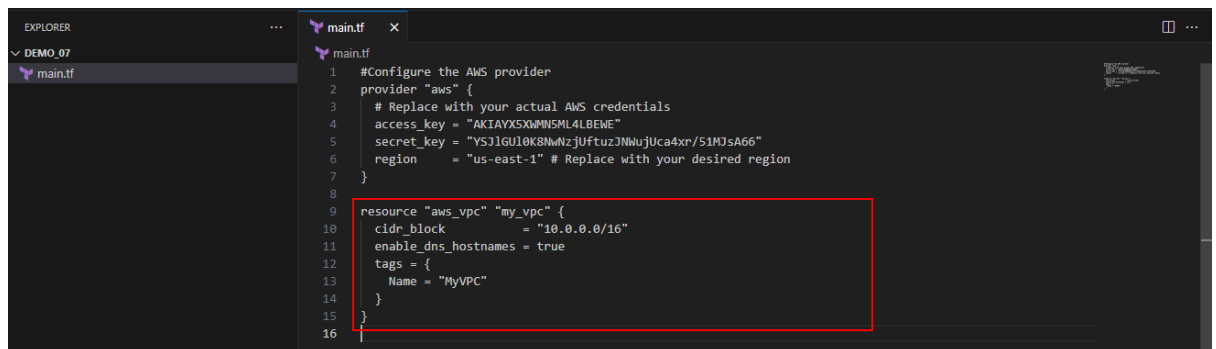
- 1.1 Open your Terraform configuration environment and create a file named **main.tf**, and add the following configuration block as shown in the screenshot below:

```
#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
}
```



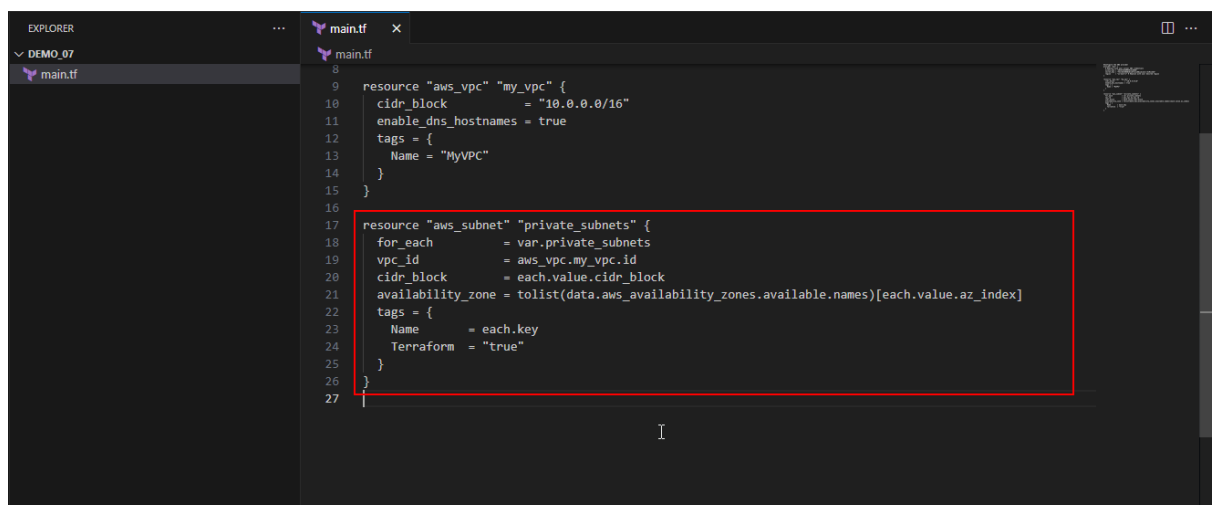
- 1.2 Add the following block to define a VPC with a CIDR block and enable DNS hostnames as shown in the screenshot below:

```
resource "aws_vpc" "my_vpc" {
  cidr_block      = "10.0.0.0/16"
  enable_dns_hostnames = true
  tags = {
    Name = "MyVPC"
  }
}
```



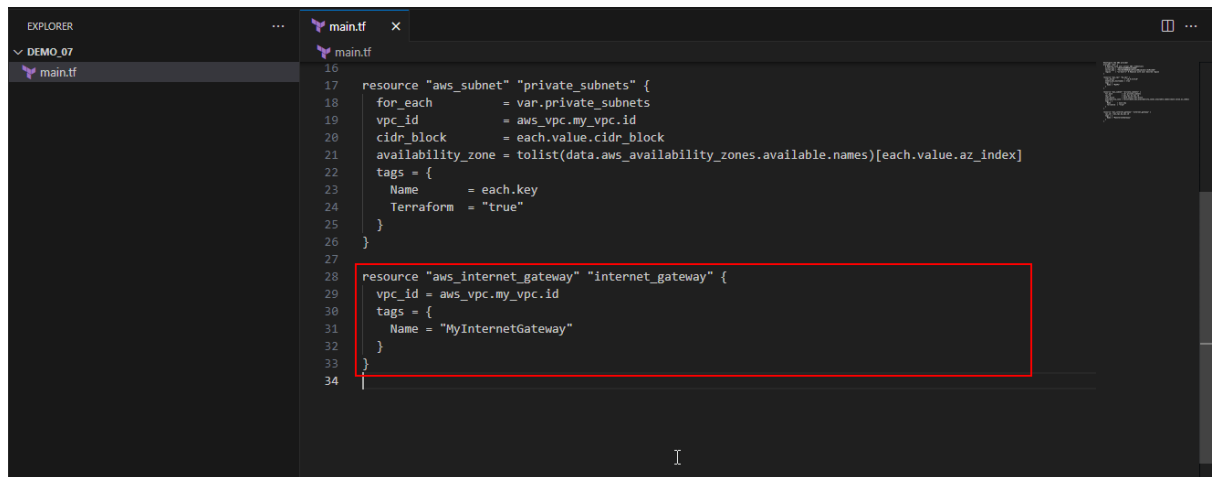
- 1.3 Add the following block to declare subnets within the VPC:

```
resource "aws_subnet" "private_subnets" {
  for_each      = var.private_subnets
  vpc_id        = aws_vpc.my_vpc.id
  cidr_block    = each.value.cidr_block
  availability_zone =
  tolist(data.aws_availability_zones.available.names)[each.value.az_index]
  tags = {
    Name      = each.key
    Terraform = "true"
  }
}
```

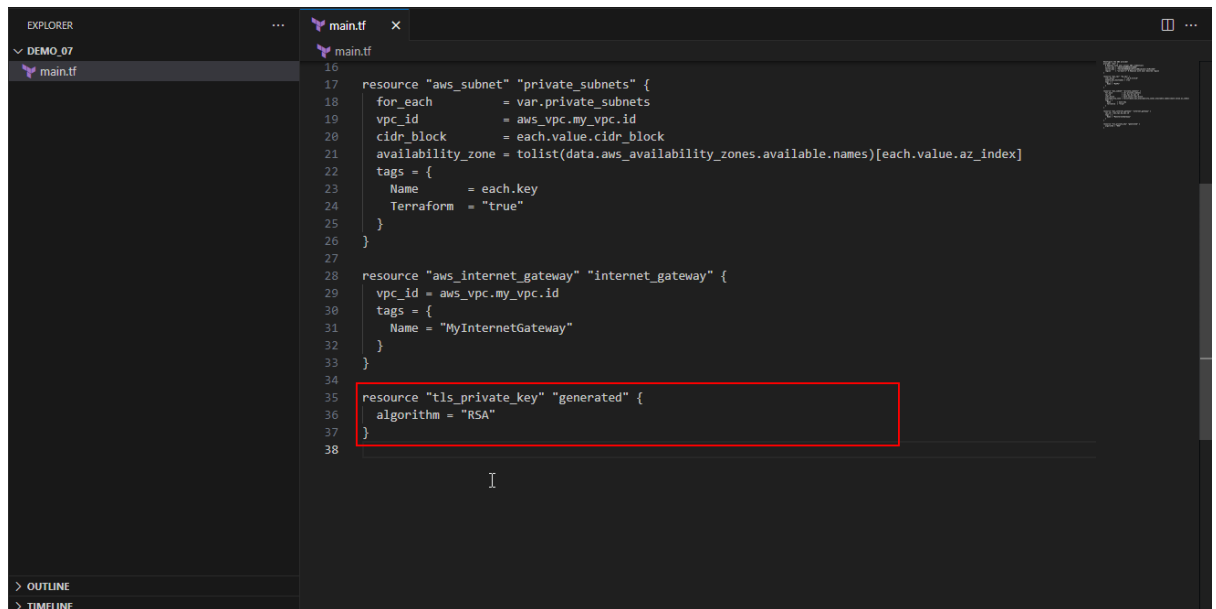


1.4 Add the following block and set up an internet gateway as shown in the screenshot below:

```
resource "aws_internet_gateway" "internet_gateway" {  
  vpc_id = aws_vpc.my_vpc.id  
  tags = {  
    Name = "MyInternetGateway"  
  }  
}
```

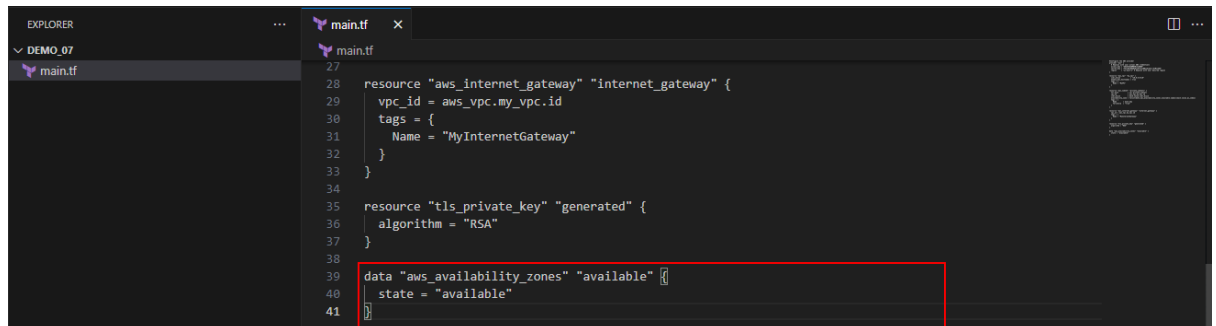


1.5 Add the following block to generate a TLS private key as shown in the screenshot below:



1.6 Add the following data block to retrieve available AWS availability zones:

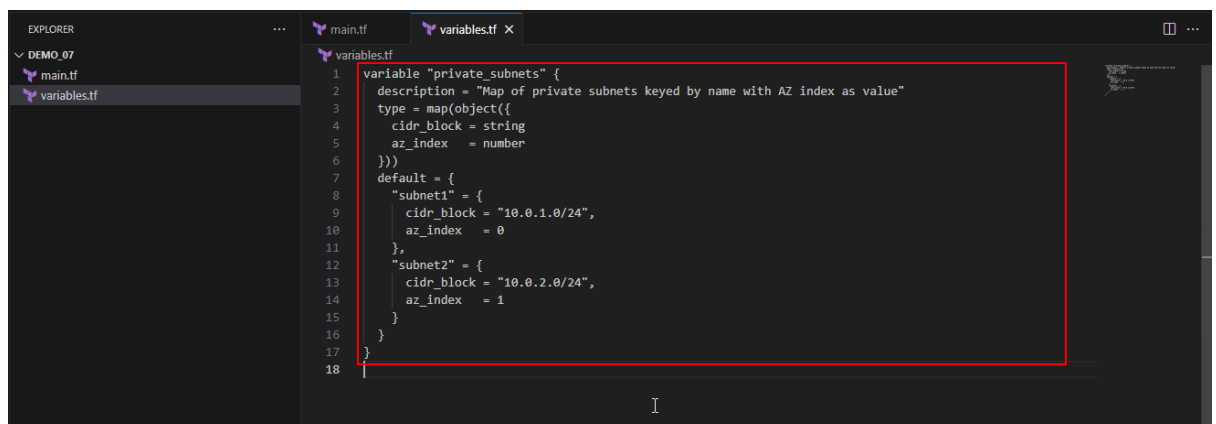
```
data "aws_availability_zones" "available" {
  state = "available"
}
```



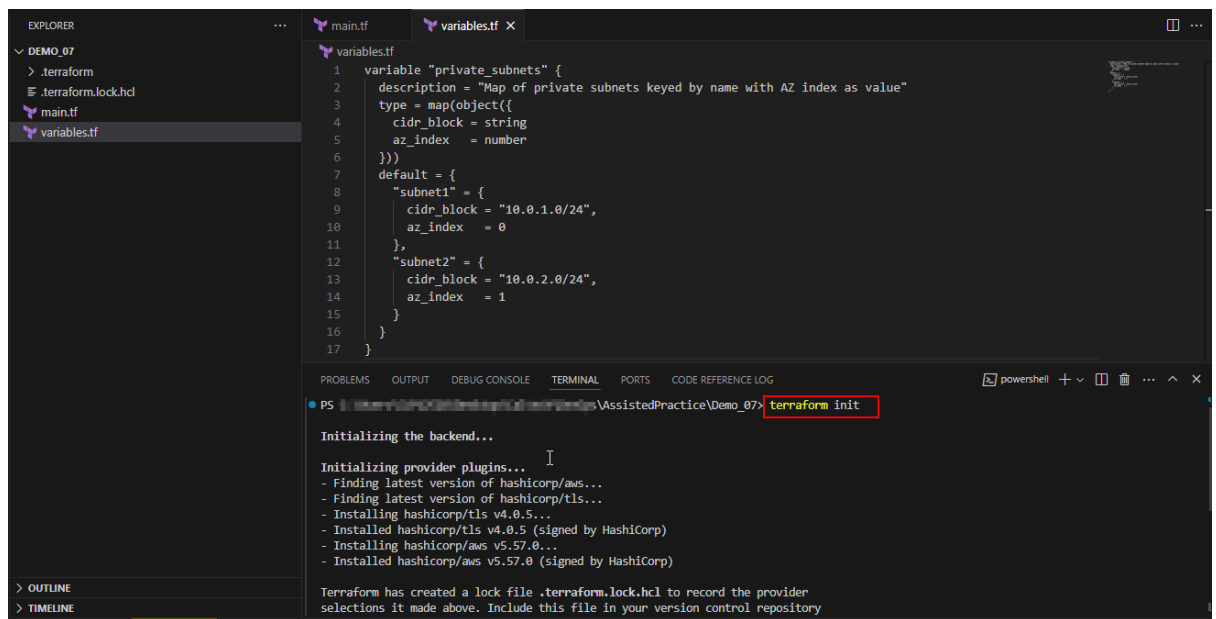
1.7 Create a file named **variables.tf** and add the following block to declare the

**private\_subnets** variable in your Terraform configuration:

```
variable "private_subnets" {
  description = "Map of private subnets keyed by name with AZ index as value"
  type = map(object({
    cidr_block = string
    az_index   = number
  }))
  default = {
    "subnet1" = {
      cidr_block = "10.0.1.0/24",
      az_index   = 0
    },
    "subnet2" = {
      cidr_block = "10.0.2.0/24",
      az_index   = 1
    }
  }
}
```



## 1.8 Run **terraform init** to initialize the Terraform configuration



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a terminal window at the bottom. The file explorer shows a project named 'DEMO\_07' with files '.terraform', '.terraform.lock.hcl', 'main.tf', and 'variables.tf'. The 'variables.tf' file is open in the editor, showing a Terraform variable definition for 'private\_subnets'. The terminal window shows the command 'terraform init' being executed, which initializes the backend and provider plugins. The output indicates that the latest versions of the AWS, TLS, and Terraform providers are being installed.

```
variable "private_subnets" {
  description = "Map of private subnets keyed by name with AZ index as value"
  type = map(object({
    cidr_block = string
    az_index   = number
  }))
  default = {
    "subnet1" = {
      cidr_block = "10.0.1.0/24",
      az_index   = 0
    },
    "subnet2" = {
      cidr_block = "10.0.2.0/24",
      az_index   = 1
    }
  }
}
```

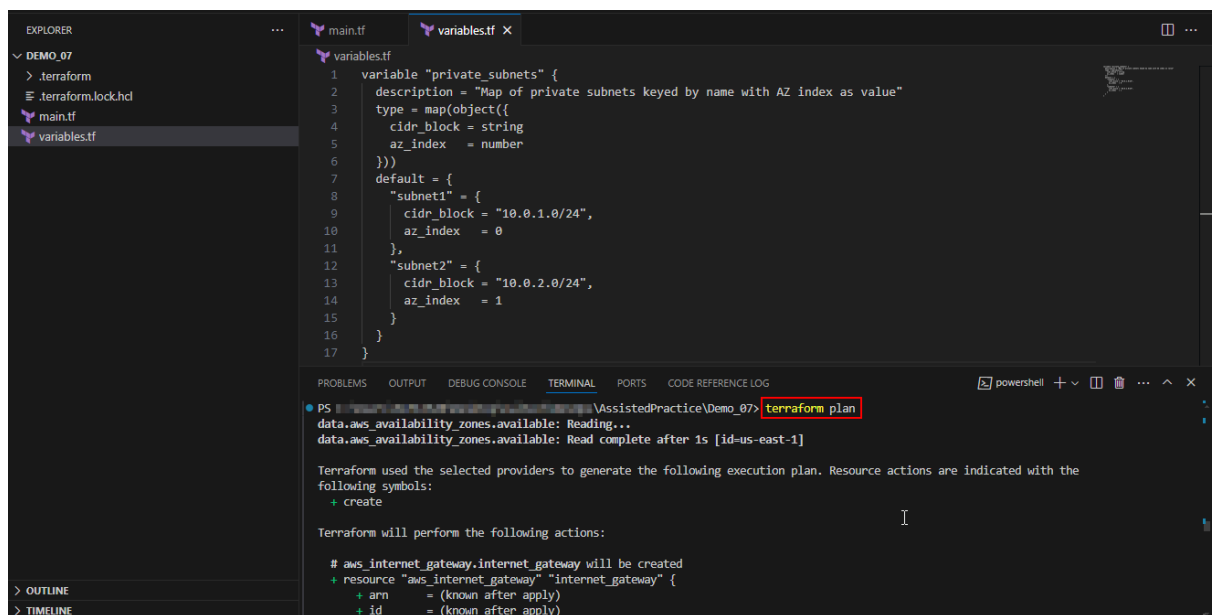
```
PS > terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/tls...
- Installing hashicorp/tls v4.0.5...
- Installing hashicorp/tls v4.0.5 (signed by HashiCorp)
- Installing hashicorp/aws v5.57.0...
- Installing 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
```

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



The screenshot shows the Visual Studio Code interface with the same file explorer and editor as the previous screenshot. The terminal window now shows the command 'terraform plan' being executed. The output indicates that the AWS availability zones are being read and that Terraform will perform the following actions: create the 'aws\_internet\_gateway' resource. The output also shows the resource definition for 'aws\_internet\_gateway'.

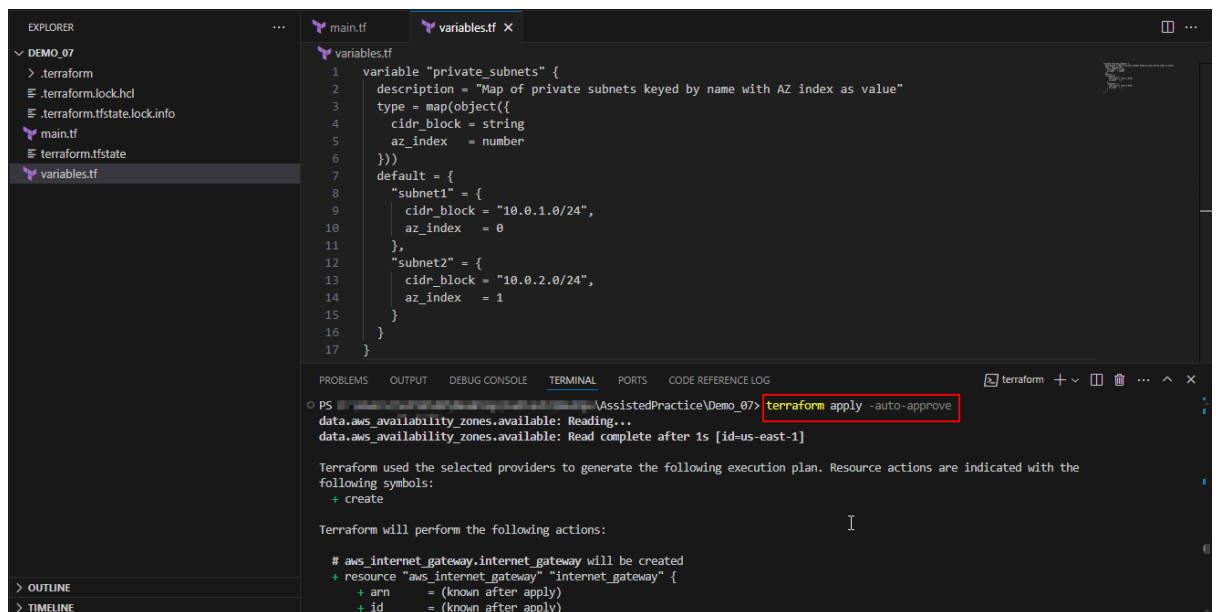
```
data.aws_availability_zones.available: Reading...
data.aws_availability_zones.available: Read complete after 1s [id-us-east-1]

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.internet_gateway will be created
+ resource "aws_internet_gateway" "internet_gateway" {
  + arn      = (known after apply)
  + id      = (known after apply)
```

## 1.10 Execute `terraform apply -auto-approve` to create the infrastructure as defined



The screenshot shows the VS Code interface with the `variables.tf` file open in the editor. The file defines a variable `private_subnets` as a map of private subnets keyed by name with AZ index as value. The default values for `subnet1` and `subnet2` are defined with their respective CIDR blocks and AZ indices.

```
1 variable "private_subnets" {
2   description = "Map of private subnets keyed by name with AZ index as value"
3   type = map(object({
4     cidr_block = string
5     az_index   = number
6   }))
7   default = {
8     "subnet1" = {
9       cidr_block = "10.0.1.0/24",
10      az_index   = 0
11    },
12    "subnet2" = {
13      cidr_block = "10.0.2.0/24",
14      az_index   = 1
15    }
16  }
17 }
```

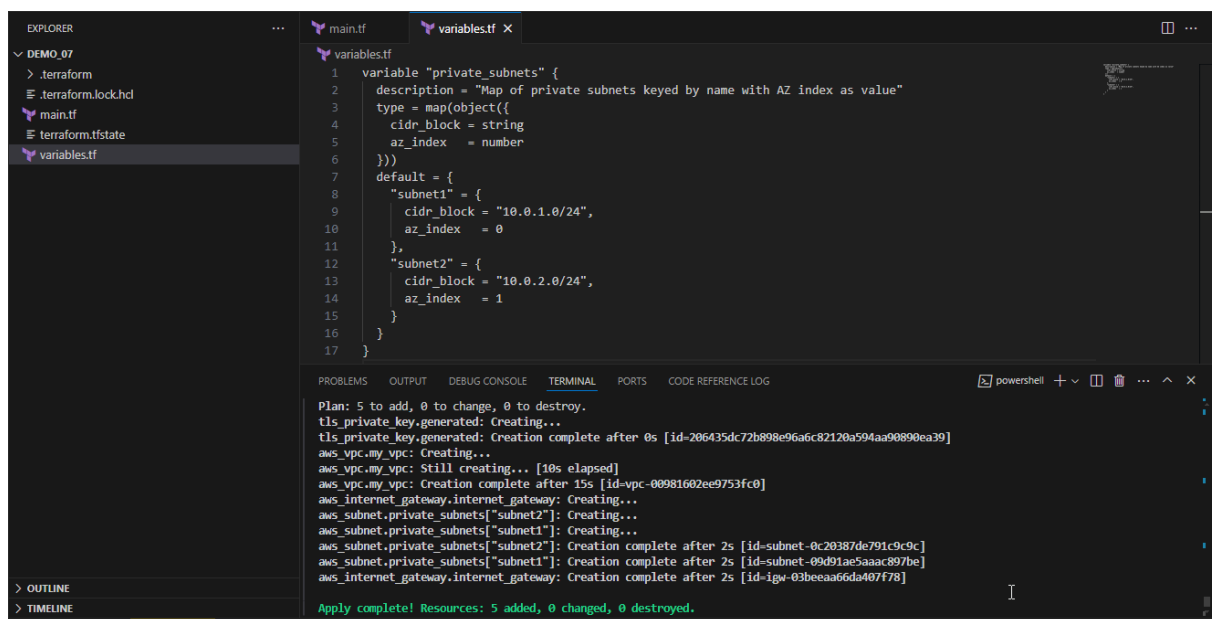
The terminal window shows the command `terraform apply -auto-approve` being executed. The output indicates that the data sources `data.aws_availability_zones.available` are being read. The execution plan shows that the `aws_internet_gateway.internet_gateway` resource will be created.

```
PS C:\AssistedPractice\Demo_07> terraform apply -auto-approve
data.aws_availability_zones.available: Reading...
data.aws_availability_zones.available: Read complete after 1s [id-us-east-1]

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.internet_gateway will be created
+ resource "aws_internet_gateway" "internet_gateway" {
+   arm      = (known after apply)
+   id       = (known after apply)
}
```



The screenshot shows the VS Code interface with the `variables.tf` file open in the editor. The terminal window shows the output of the `terraform apply -auto-approve` command, indicating that the infrastructure has been successfully created.

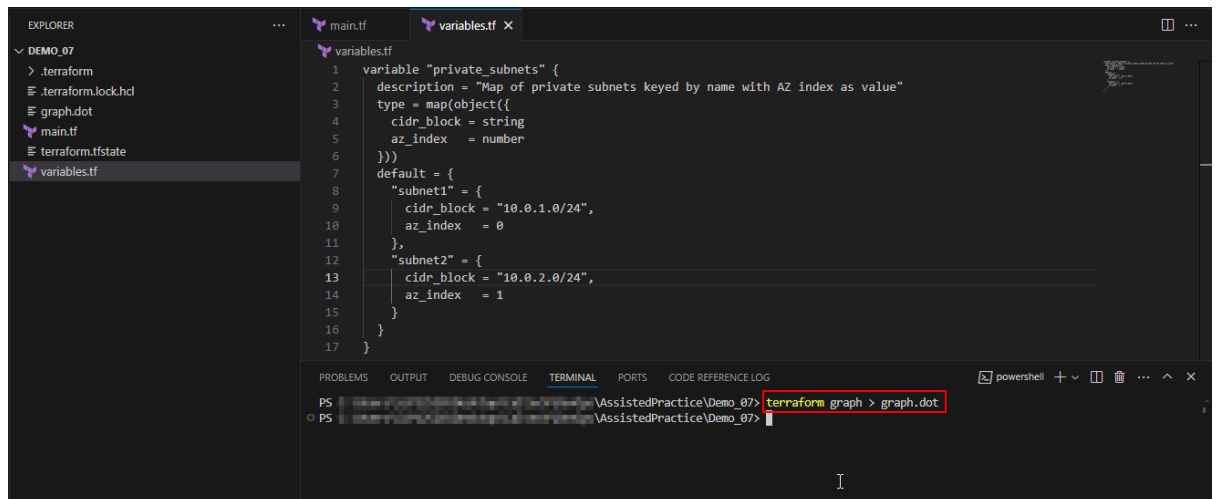
```
Plan: 5 to add, 0 to change, 0 to destroy.
tls_private_key.generated: Creating...
tls_private_key.generated: Creation complete after 0s [id-206435dc72b898e96a6c82120a594aa90890ea39]
aws_vpc.my_vpc: Creating...
aws_vpc.my_vpc: Still creating... [10s elapsed]
aws_vpc.my_vpc: Creation complete after 15s [id-vpc-00981602ee9753fc0]
aws_internet_gateway.internet_gateway: Creating...
aws_subnet.private_subnets["subnet2"]: Creating...
aws_subnet.private_subnets["subnet1"]: Creating...
aws_subnet.private_subnets["subnet2"]: Creation complete after 2s [id-subnet-0c20387ds791c0c9c]
aws_subnet.private_subnets["subnet1"]: Creation complete after 2s [id-subnet-09d91ae5aaac897be]
aws_internet_gateway.internet_gateway: Creation complete after 2s [id-igw-03beaa66da407f78]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```

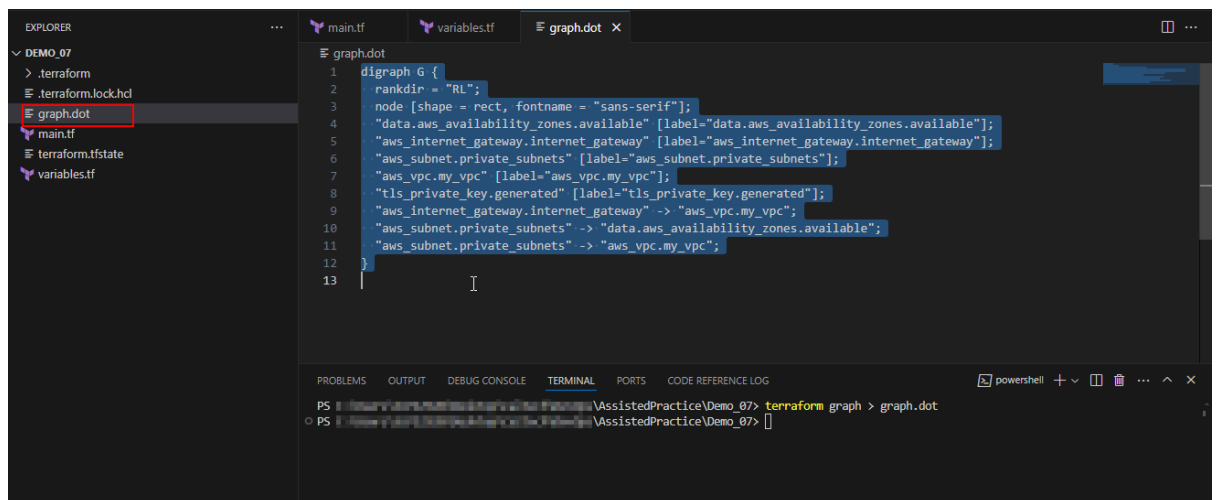
## Step 2: Generate and visualize the resource graph

2.1 Use the following command to generate a dependency graph of the Terraform configuration:

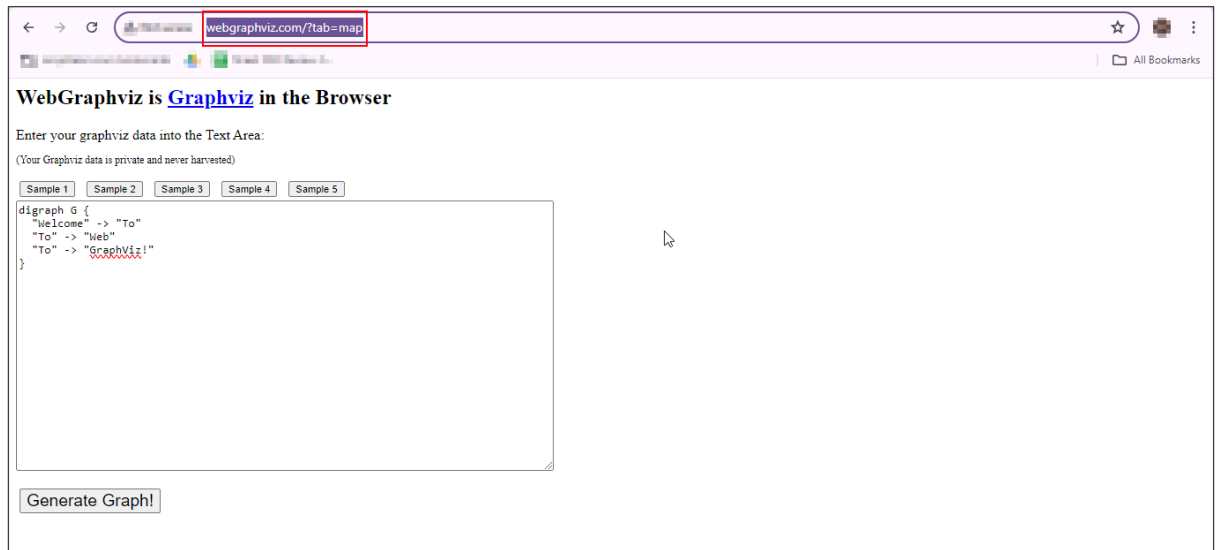
**terraform graph > graph.dot**



2.2 Click on the **graph.dot** file and copy the diagram as shown in the screenshot below:



2.3 Navigate to <http://www.webgraphviz.com/?tab=map> as shown in the screenshot below:

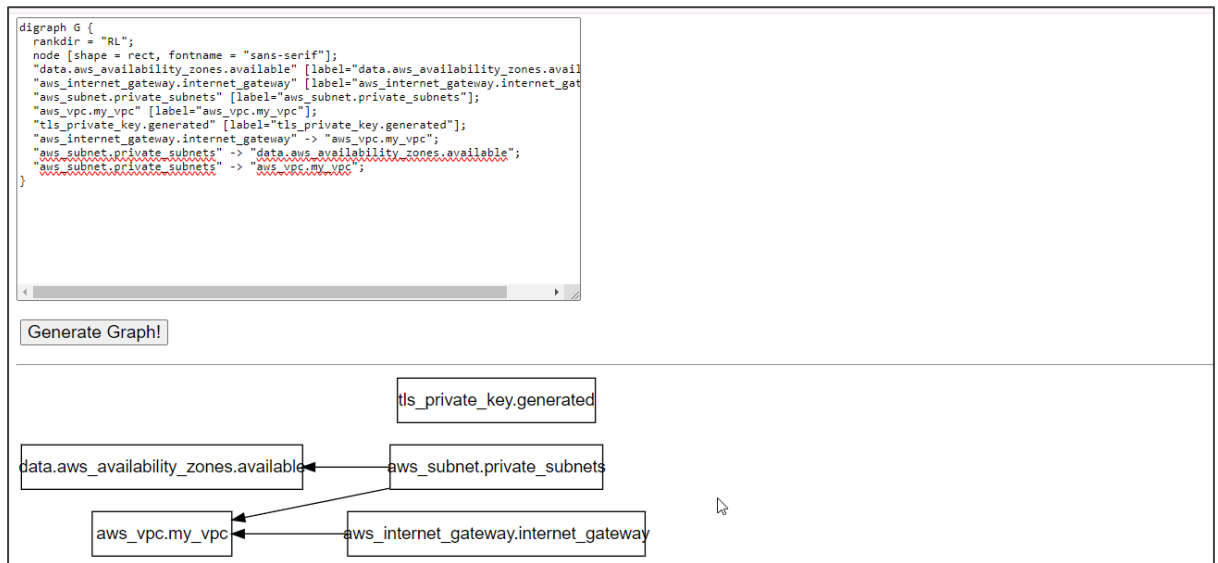


2.4 Paste the copied data in the input box and click on the **Generate Graph!** button as shown in the screenshot below:





You will see the generated graph as shown in the screenshot below:



By following these steps, you have successfully created and visualized Terraform resource dependency graphs for better infrastructure management.