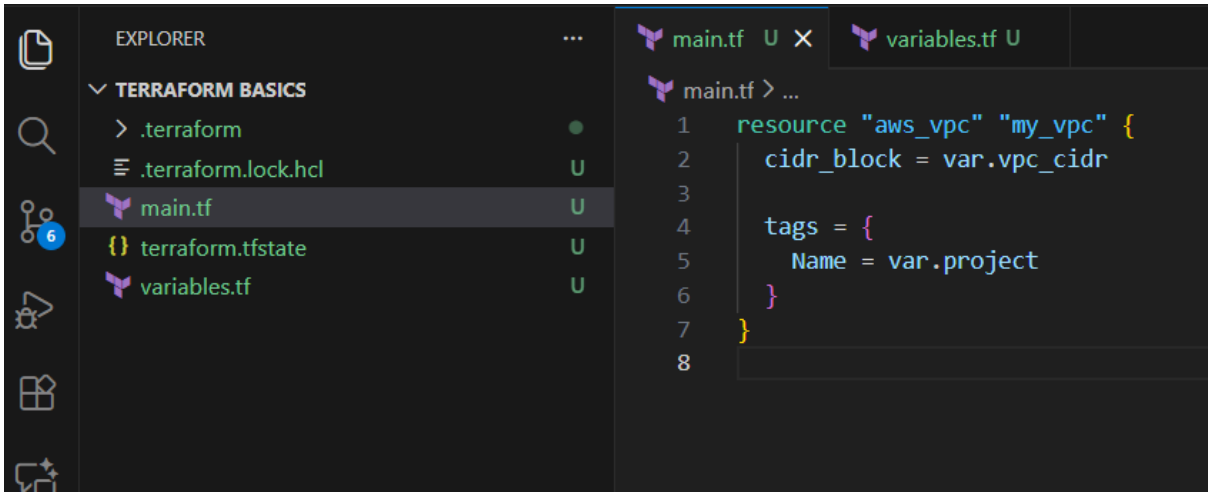
1. Create VPC

Give main.tf as this code.

Main.tf:

```
variable "project" {
  default = "demo-vpc"
}

variable "vpc_cidr" {
  default = "10.0.0.0/16"
}
```



Write the code in variables.tf as

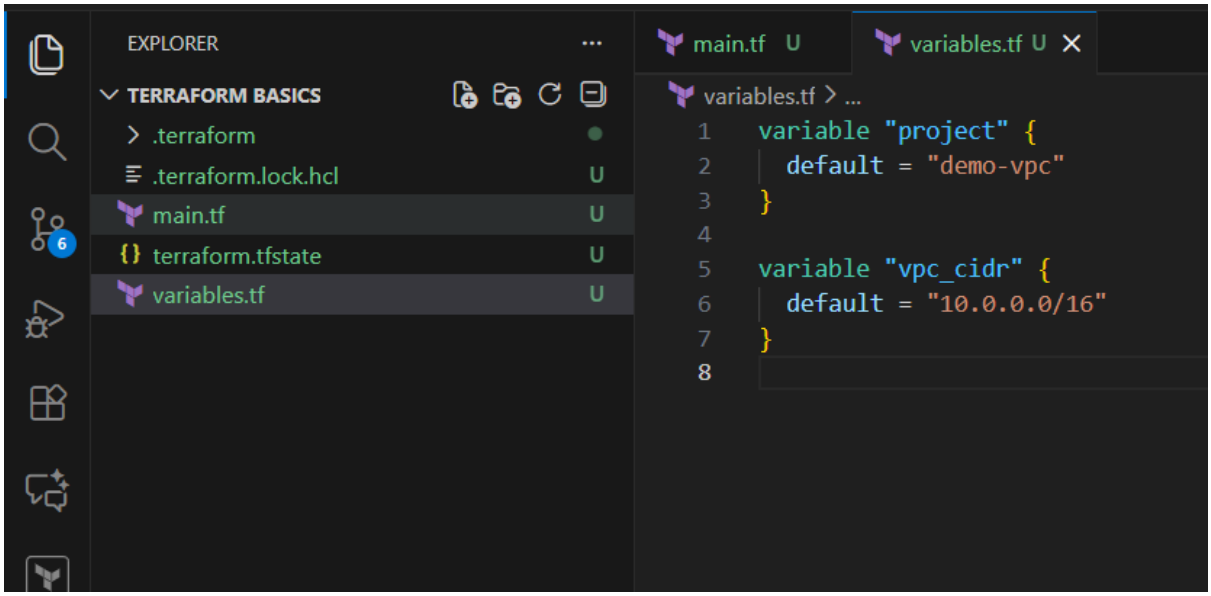
Variables.tf:

```
variable "project" {
  default = "demo-vpc"
}
```

variable "vpc_cidr" {

default = "10.0.0.0/16"

}



- terraform init
- terraform apply

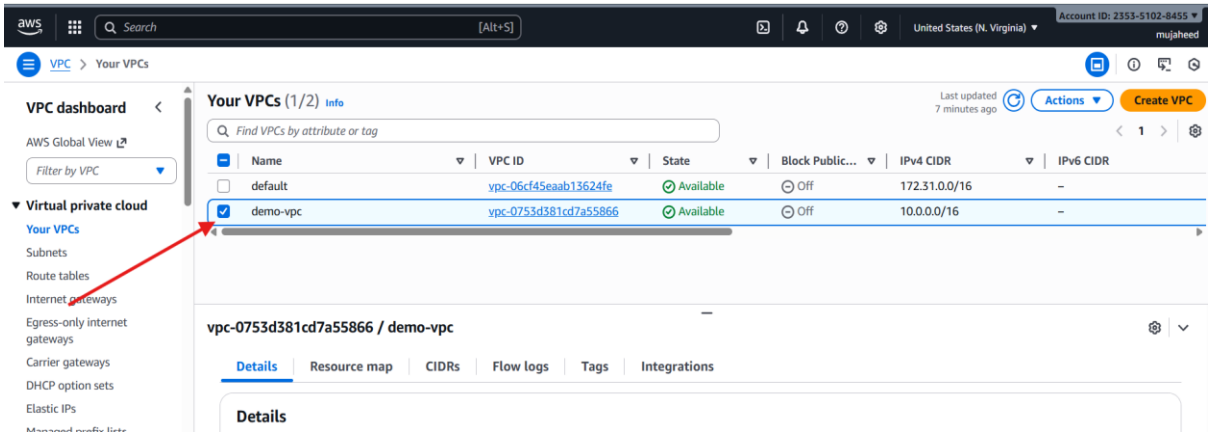
```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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!

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,
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

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

Terraform will perform the following actions:

  # aws_vpc.my_vpc will be created
  + resource "aws_vpc" "my_vpc" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
          + "Name" = "demo-vpc"
        }
      + tags_all                               = {
          + "Name" = "demo-vpc"
        }
    }

Plan: 1 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: Creation complete after 5s [id=vpc-0753d381cd7a55866]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

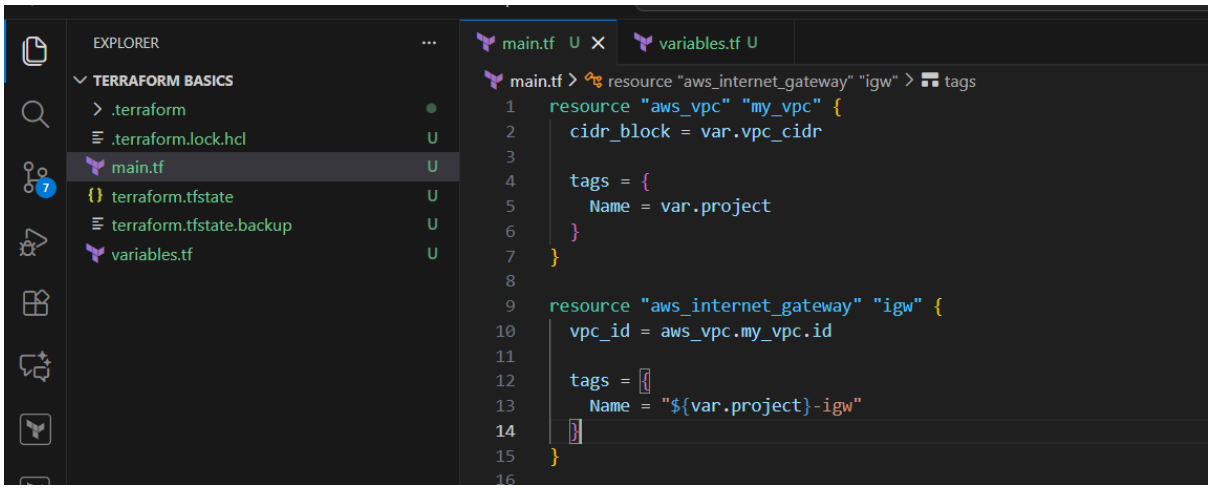
A vpc has been created.



2. Create Internet gateway

Give the resource in main as

```
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.my_vpc.id


  tags = {
    Name = "${var.project}-igw"
  }
}
```



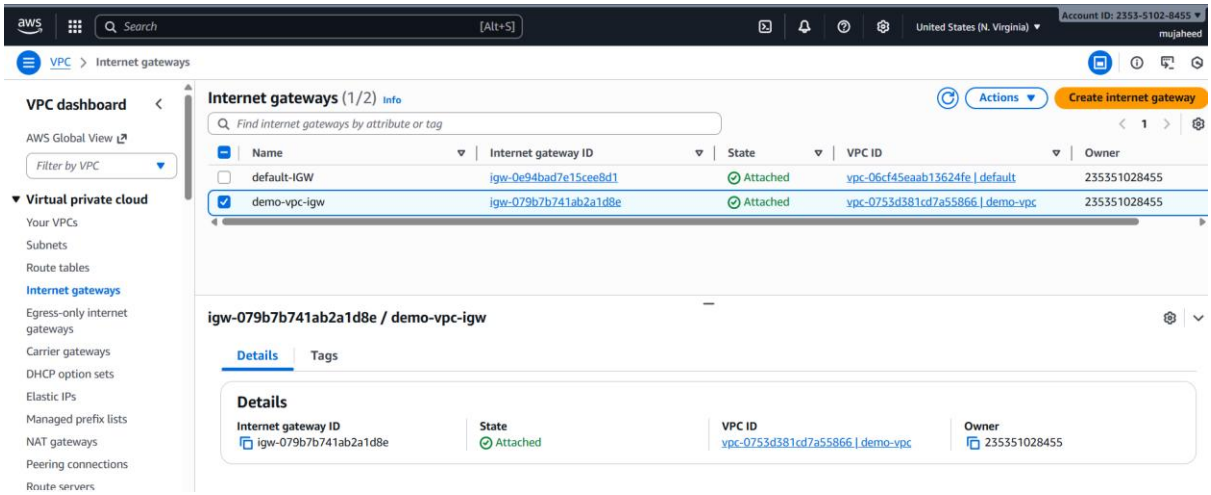
Keep the vaiables.tf as same

Give the commands terraform init,terraform apply.

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]

Terraform used the selected providers to generate the following execution plan. Resou
  + create
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

        }
      + tags_all = {
          + "Name" = "demo-vpc-igw"
        }
      + vpc_id   = "vpc-0753d381cd7a55866"
    }

Plan: 1 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_internet_gateway.igw: Creating...
aws_internet_gateway.igw: Creation complete after 2s [id=igw-079b7b741ab2a1d8e]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

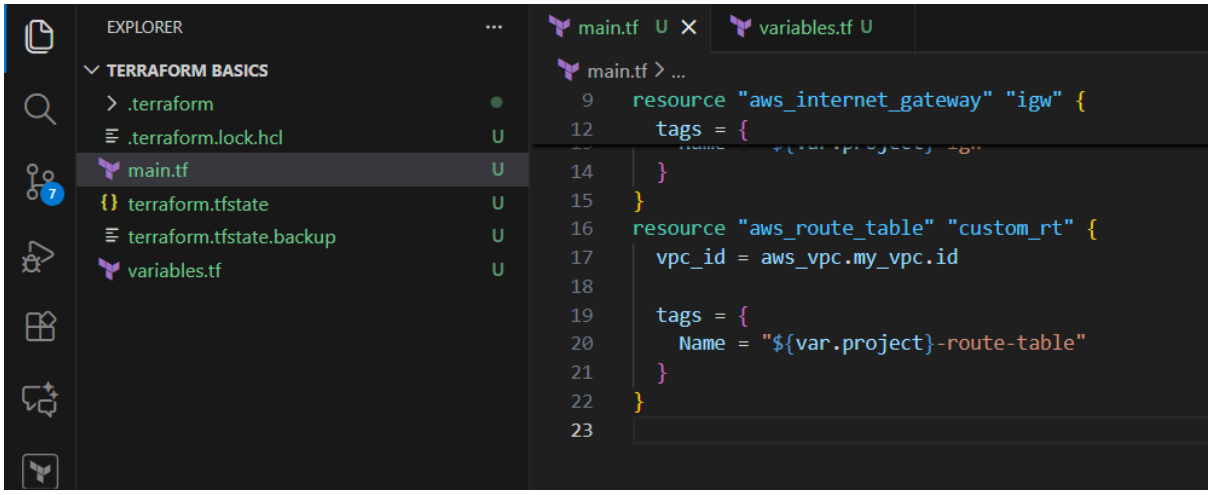
A internet gateway has been created and attached to your created vpc.



3. Create Custom Route Table

Add resource in the main.tf

```
resource "aws_route_table" "custom_rt" {
  vpc_id = aws_vpc.my_vpc.id


  tags = {
    Name = "${var.project}-route-table"
  }
}
```

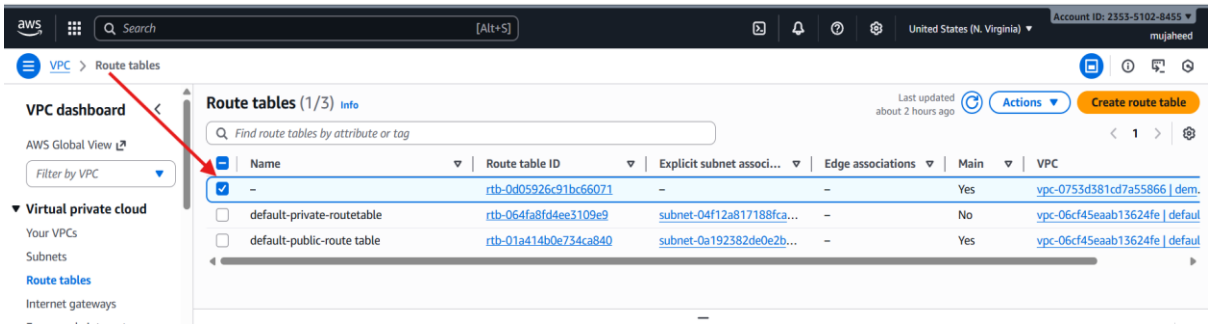



```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]

Terraform used the selected providers to generate the following execution plan. Resourc
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
      }
    + tags_all         = {
        + "Name" = "demo-vpc-route-table"
      }
    + vpc_id           = "vpc-0753d381cd7a55866"
  }

Plan: 1 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_route_table.custom_rt: Creating...
aws_route_table.custom_rt: Creation complete after 2s [id=rtb-0c60d4a3229f73727]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

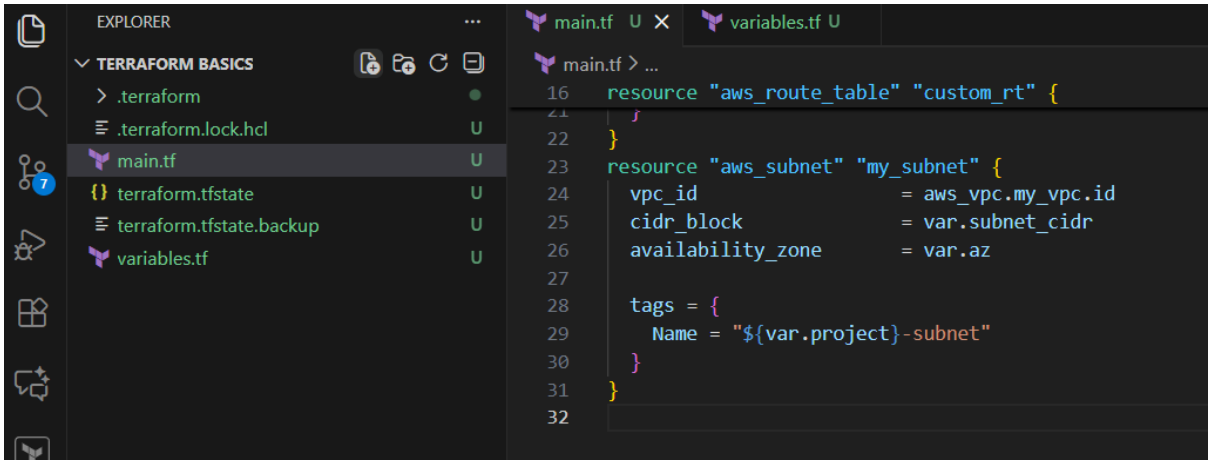
A routetable has been created for your vpc



4. Create Subnet

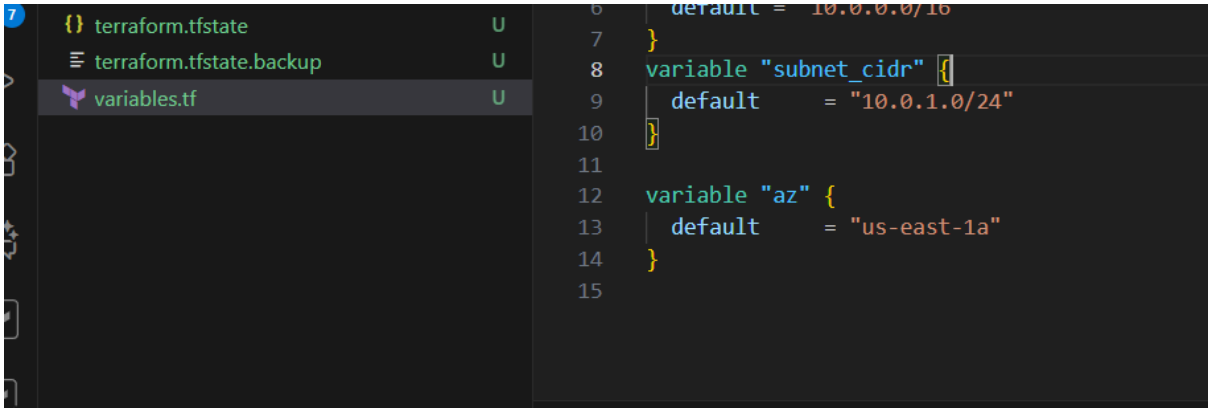
Add this resource block in main.tf

```
resource "aws_subnet" "my_subnet" {
  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = var.subnet_cidr
  availability_zone       = var.az

  tags = {
    Name = "${var.project}-subnet"
  }
}
```



Add the data in the variables.tf

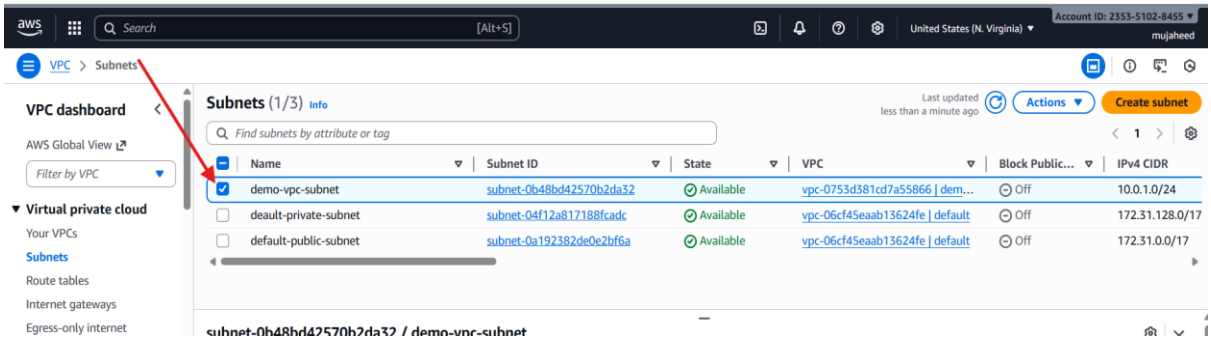
```
variable "subnet_cidr" {
  default     = "10.0.1.0/24"
}

variable "az" {
  default     = "us-east-1a"
}
```



```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

      }
    + tags_all                                = {
        + "Name" = "demo-vpc-subnet"
      }
    + vpc_id                                  = "vpc-0753d381cd7a55866"
  }

Plan: 1 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_subnet.my_subnet: Creating...
aws_subnet.my_subnet: Creation complete after 3s [id=subnet-0b48bd42570b2da32]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

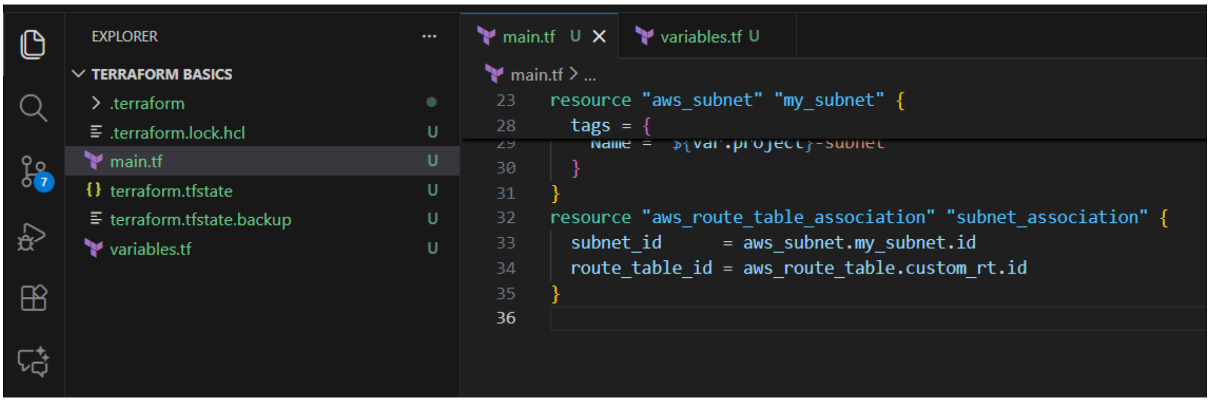
A subnet has been created for our vpc.

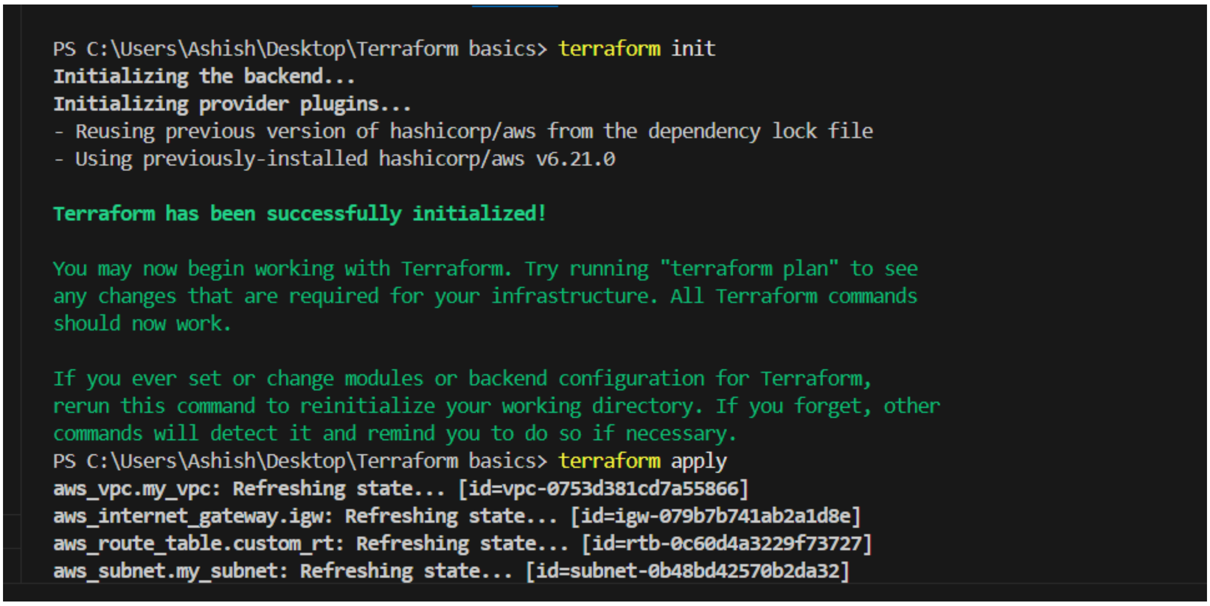


5. Associate subnet with Route Table

Add the resource in main.tf to associate subnet with route table.

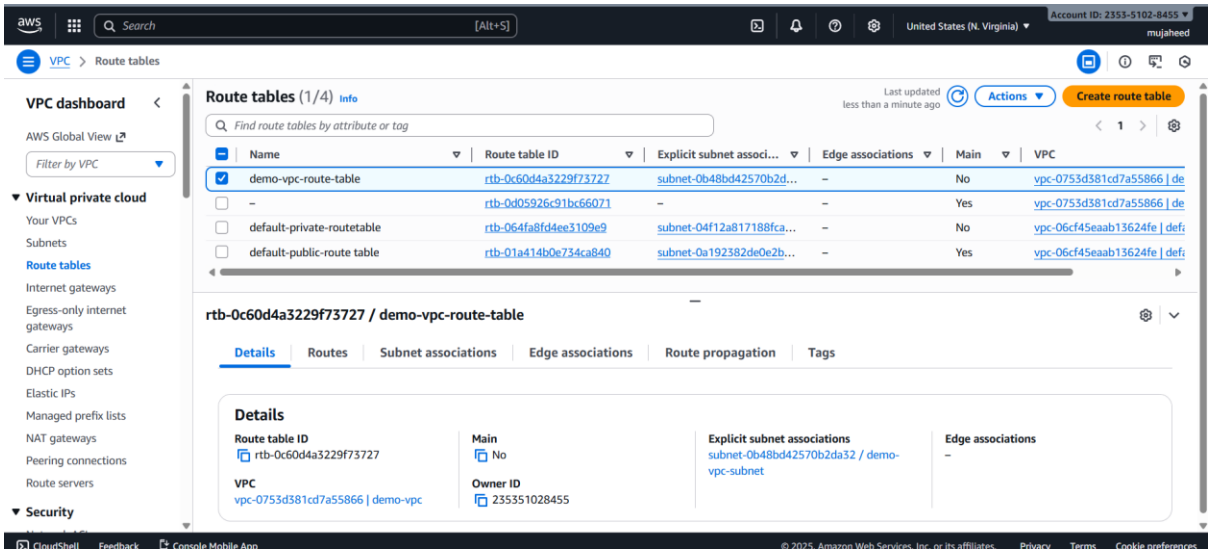
```
resource "aws_route_table_association" "subnet_association" {
  subnet_id      = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.custom_rt.id
}
```





```
  + resource "aws_route_table_association" "subnet_association" {
      + id             = (known after apply)
      + region         = "us-east-1"
      + route_table_id = "rtb-0c60d4a3229f73727"
      + subnet_id      = "subnet-0b48bd42570b2da32"
    }

Plan: 1 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_route_table_association.subnet_association: Creating...
aws_route_table_association.subnet_association: Creation complete after 7s [id=rtbassoc-0d5fb1390a1079cb7]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

An instance has been created with the routetable subnet association.



6. Create Security Group to allow port 22,80,443

Give the script in main.tf as

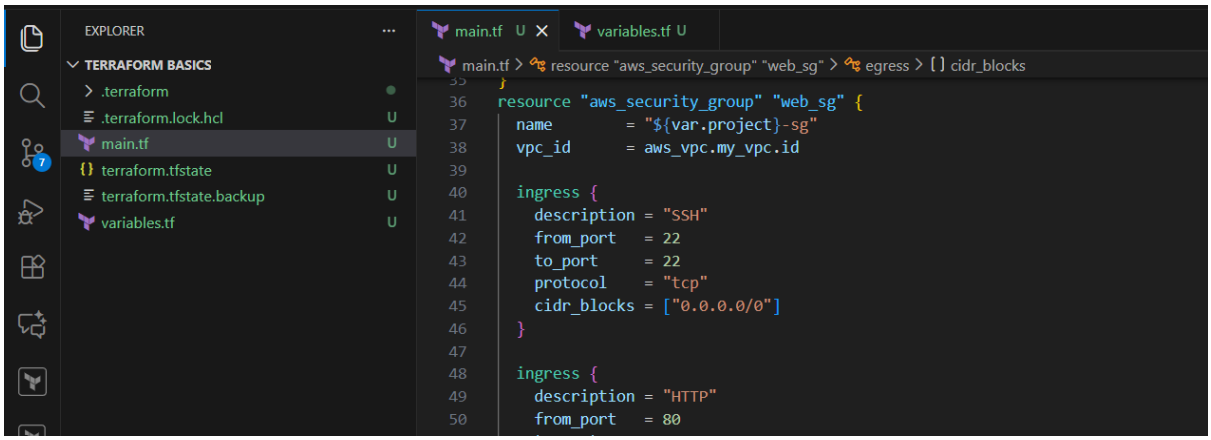
resource "aws_security_group" "web_sg" {

name = "${var.project}-sg"

vpc_id = aws_vpc.my_vpc.id

```
ingress {
  description = "SSH"
  from_port   = 22
  to_port     = 22
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  description = "HTTP"
  from_port   = 80
  to_port     = 80
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  description = "HTTPS"
  from_port   = 443
  to_port     = 443
```

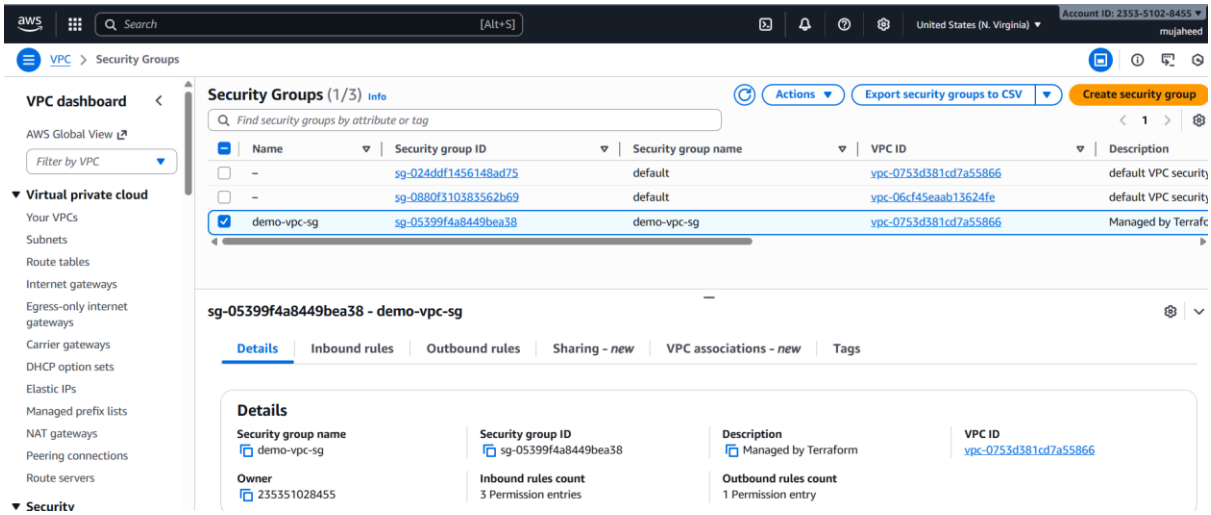
```
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
 }

 egress {
  description = "Allow all outbound"
  from_port   = 0
  to_port     = 0
  protocol    = "-1"
  cidr_blocks = ["0.0.0.0/0"]
 }

 tags = {
  Name = "${var.project}-sg"
 }
}
```



```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
aws_subnet.my_subnet: Refreshing state... [id=subnet-0b48bd42570b2da32]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

      + tags_all                = {
          + "Name" = "demo-vpc-sg"
        }
      + vpc_id                  = "vpc-0753d381cd7a55866"
    }

Plan: 1 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_security_group.web_sg: Creating...
aws_security_group.web_sg: Still creating... [00m10s elapsed]
aws_security_group.web_sg: Creation complete after 18s [id=sg-05399f4a8449bea38]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

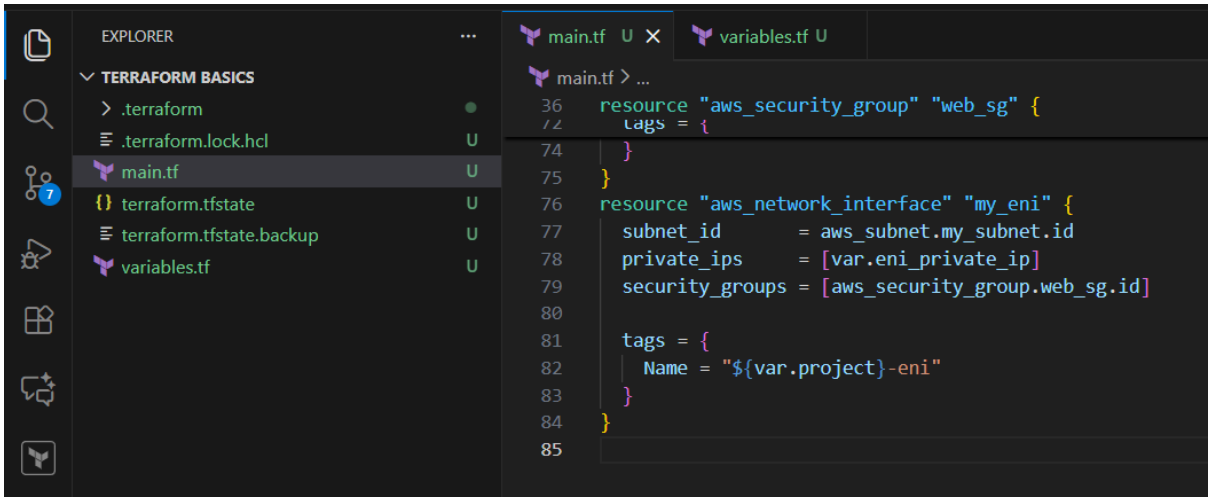
A security group has been created with the 3 protocols.



7. Create a network interface with an ip in the subnet that was created in step 4

Give the script in main.tf as

```
resource "aws_network_interface" "my_eni" {
  subnet_id       = aws_subnet.my_subnet.id
  private_ips     = [var.eni_private_ip]
  security_groups = [aws_security_group.web_sg.id]


  tags = {
    Name = "${var.project}-eni"
  }
}
```

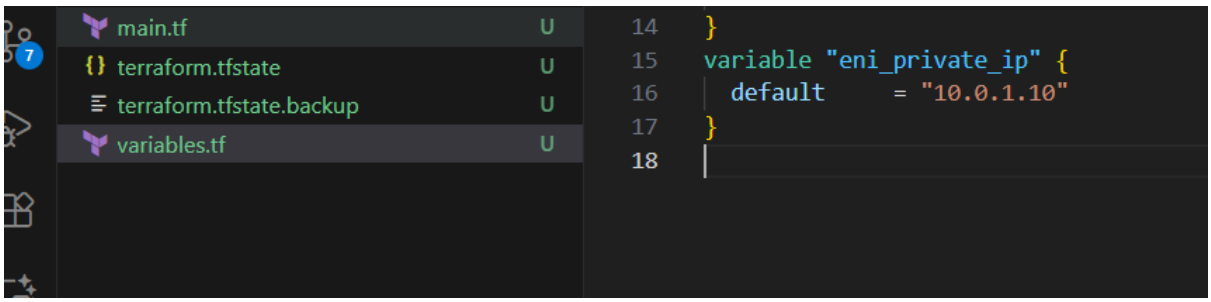


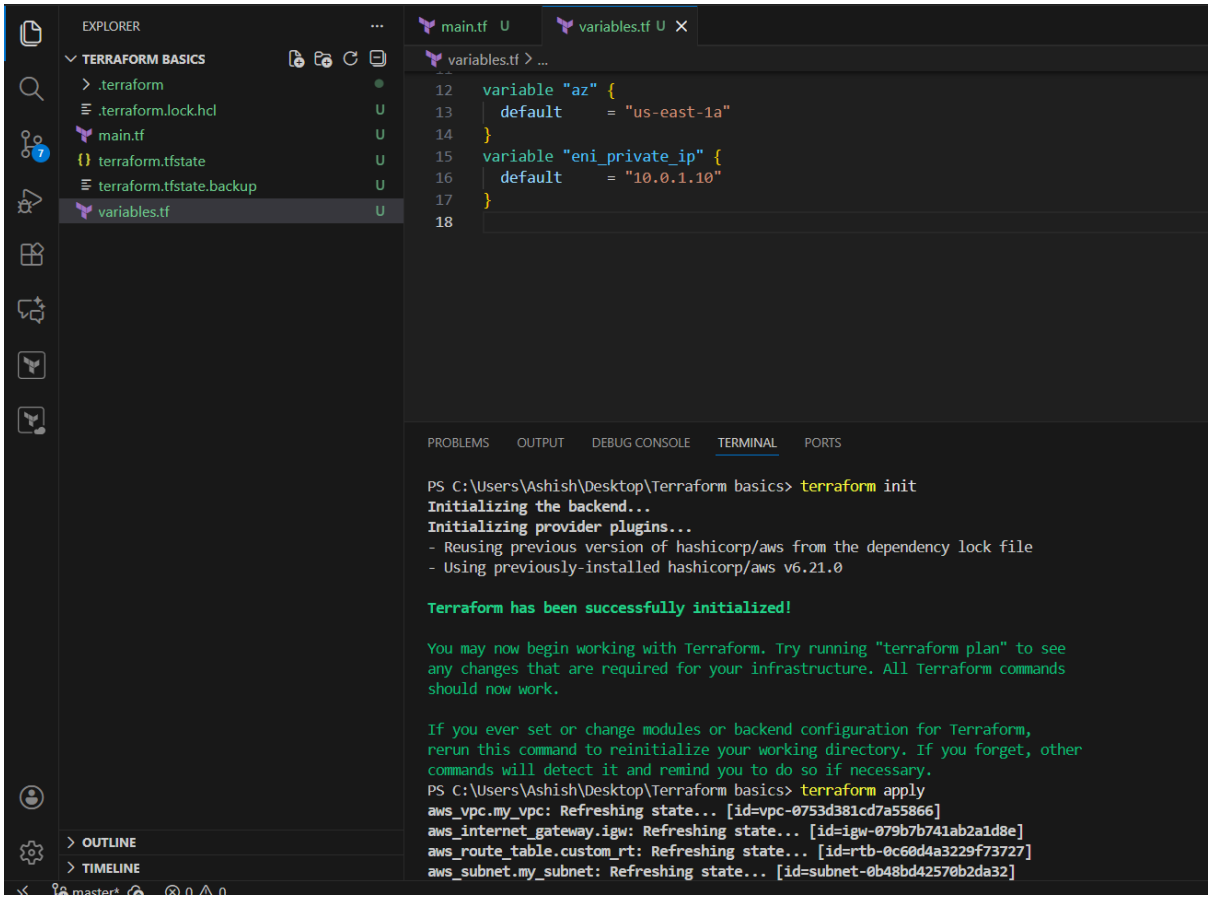
Give the ip in variables.tf

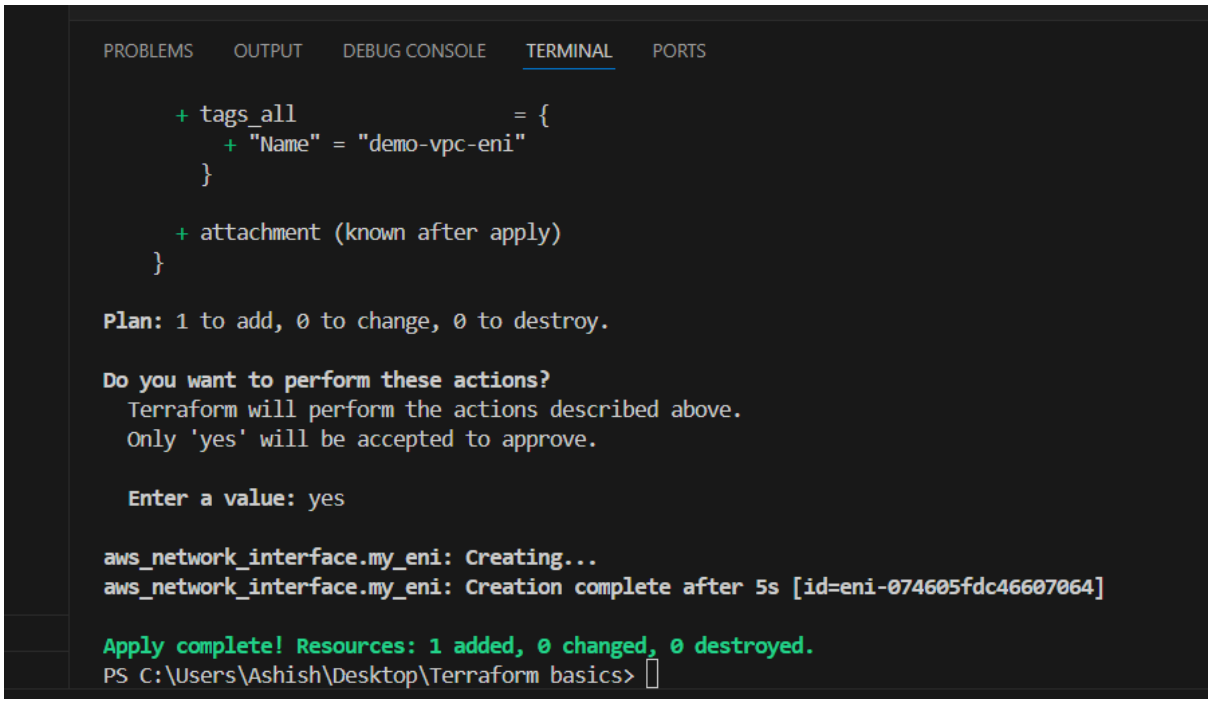
variable "eni_private_ip" {

default = "10.0.1.10"

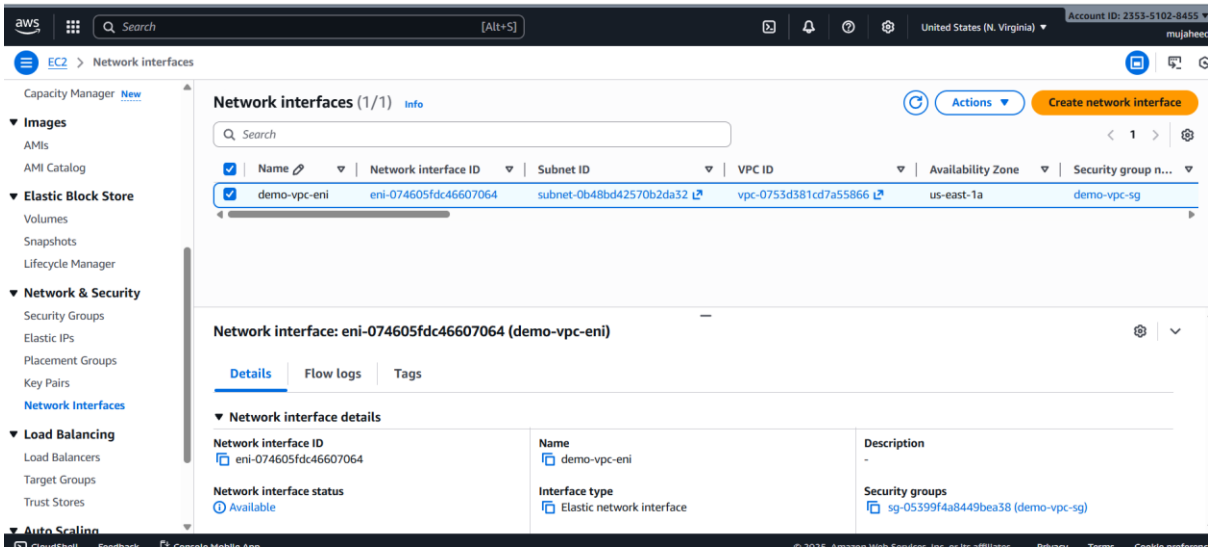
}







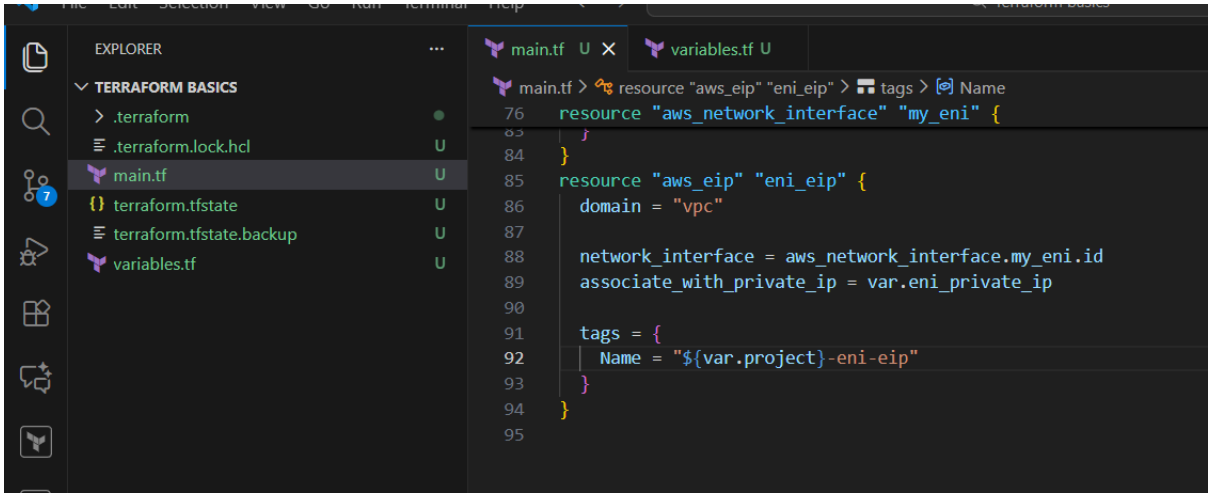
An network interface will be created with this.



8. Assign an elastic IP to the network interface created in step 7

Add this script in resource block in main.tf

```
resource "aws_eip" "eni_eip" {
  domain = "vpc"

  network_interface = aws_network_interface.my_eni.id
  associate_with_private_ip = var.eni_private_ip

  tags = {
    Name = "${var.project}-eni-eip"
  }
}
```

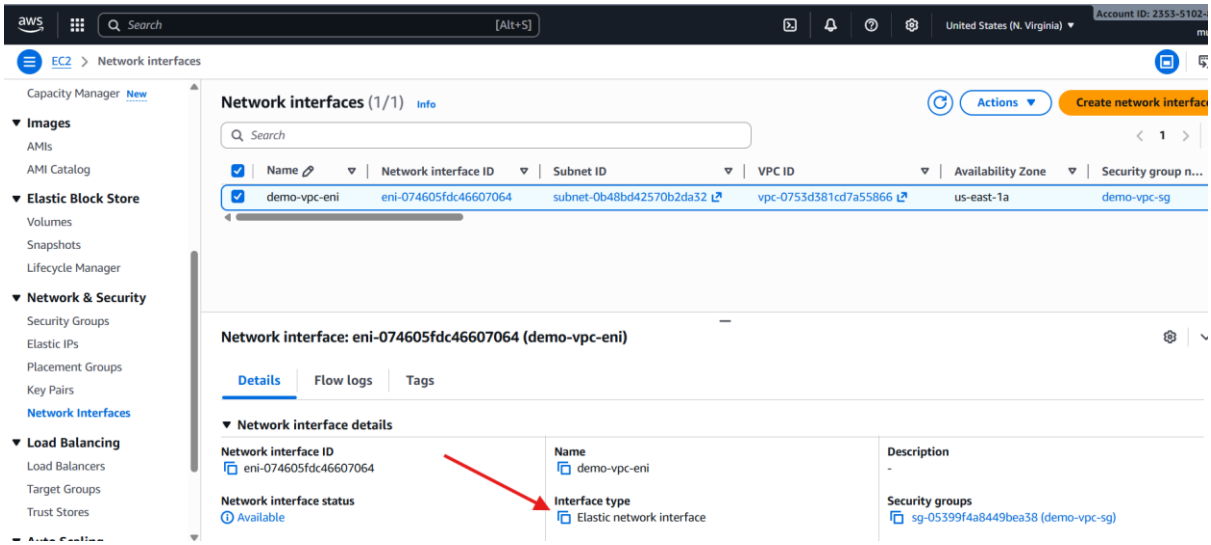


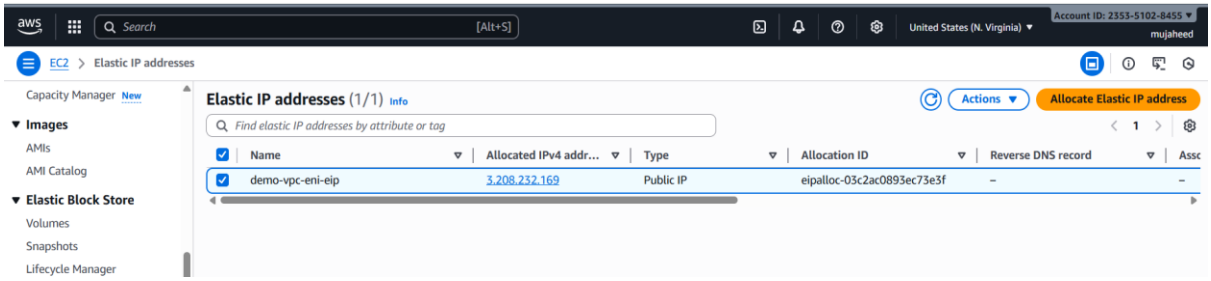

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
aws_subnet.my_subnet: Refreshing state... [id=subnet-0b48bd42570b2da32]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
      + "Name" = "demo-vpc-eni-eip"
    }
  + tags_all                    = {
      + "Name" = "demo-vpc-eni-eip"
    }
}

Plan: 1 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_eip.eni_eip: Creating...
aws_eip.eni_eip: Creation complete after 4s [id=eipalloc-03c2ac0893ec73e3f]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

Elastic ip has been assigned to your network interface.



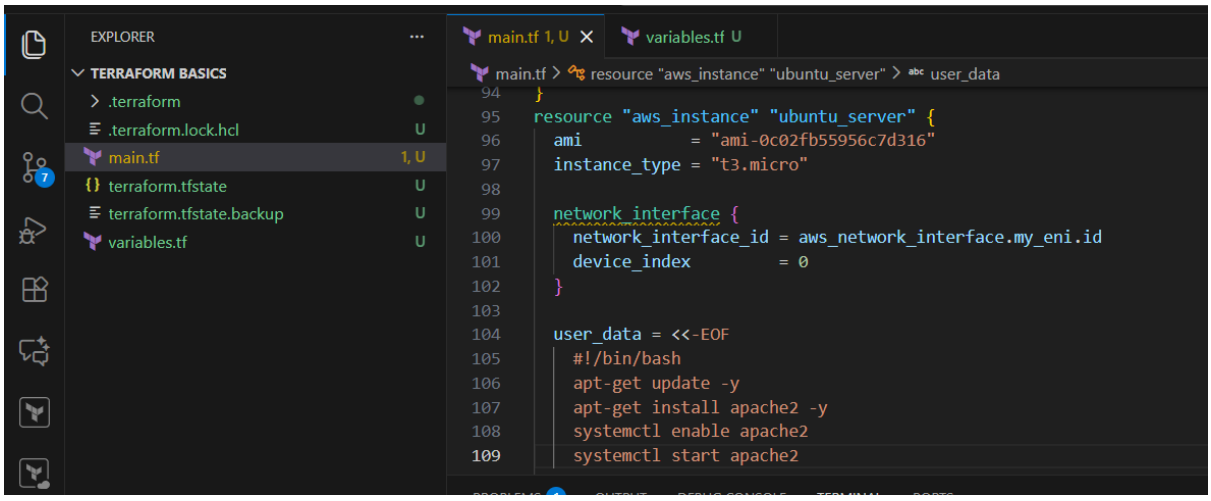


9. Create Ubuntu server and install/enable apache2

Add the script in resource block in main.tf

```
resource "aws_instance" "ubuntu_server" {
  ami           = "ami-0c02fb55956c7d316"
  instance_type = "t3.micro"
key_name = “red”

  network_interface {
    network_interface_id = aws_network_interface.my_eni.id
```

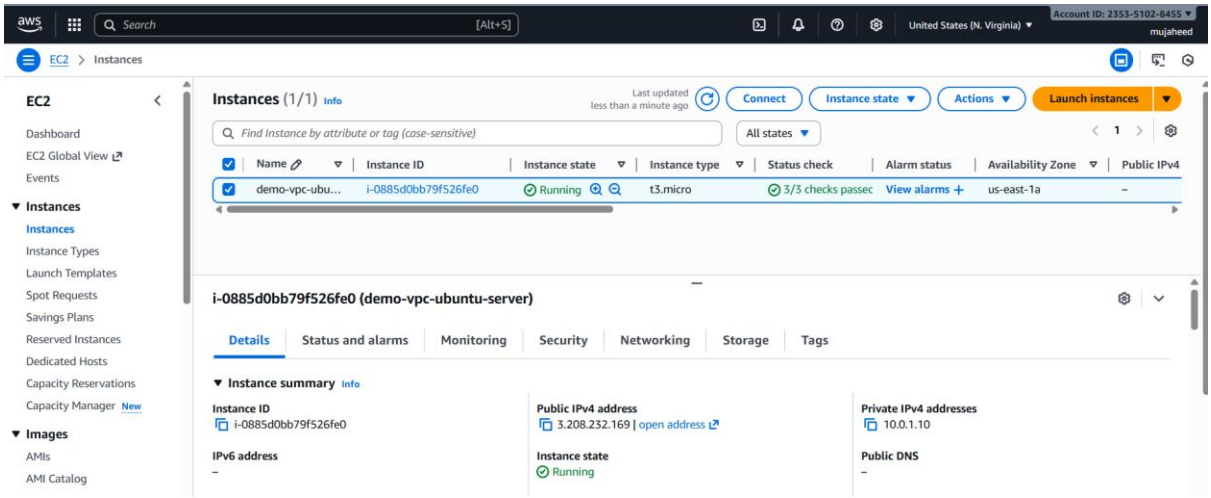
```
  device_index        = 0
 }

 user_data = <<-EOF
  #!/bin/bash
  apt-get update -y
  apt-get install apache2 -y
  systemctl enable apache2
  systemctl start apache2
 EOF

 tags = {
  Name = "${var.project}-ubuntu-server"
 }
}
```



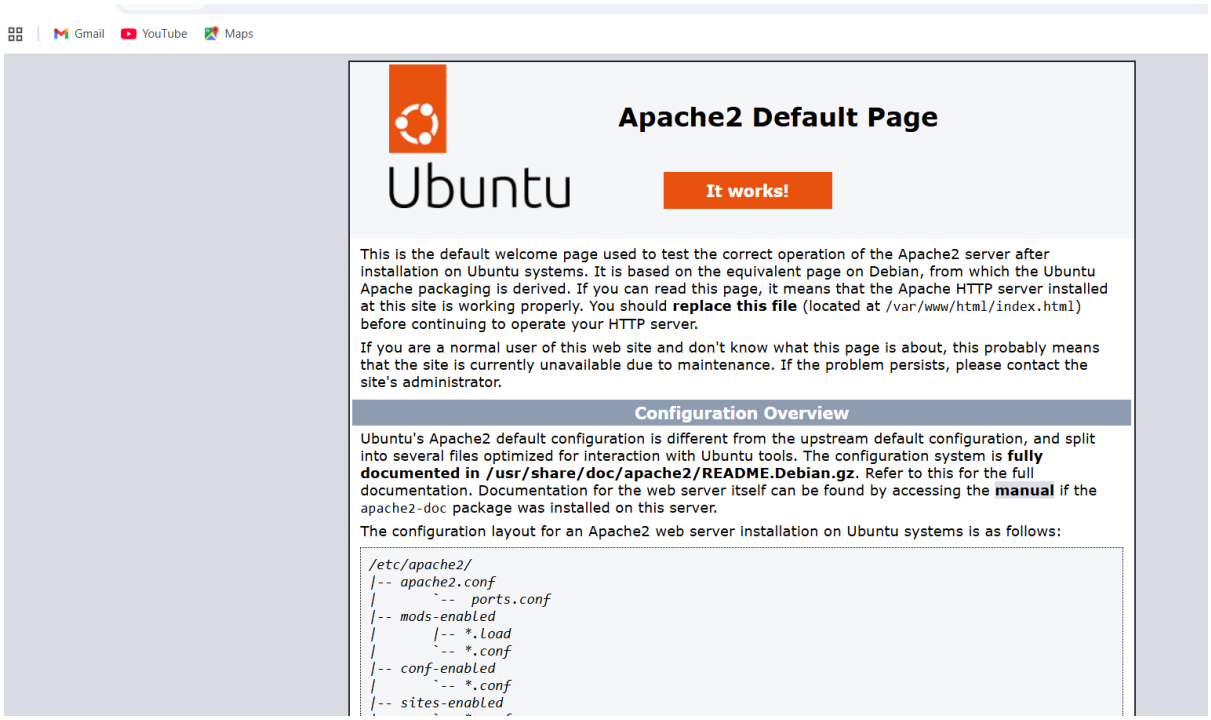
```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
aws_subnet.my_subnet: Refreshing state... [id=subnet-0b48bd42570b2da32]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
```

```
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.ubuntu_server: Creating...
aws_instance.ubuntu_server: Still creating... [00m10s elapsed]
aws_instance.ubuntu_server: Creation complete after 18s [id=i-0885d0bb79f526fe0]

Warning: Argument is deprecated

  with aws_instance.ubuntu_server,
  on main.tf line 95, in resource "aws_instance" "ubuntu_server":
  95: resource "aws_instance" "ubuntu_server" {

network_interface is deprecated. To specify the primary network interface, use primary_network_interface instead. To attach additional
network interfaces, use the aws_network_interface_attachment resource.

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

An ubuntu instance has been created.



If you copy your ip and search in browser with 80 port number you will see like.



main.tf:

```
resource "aws_vpc" "my_vpc" {
  cidr_block = var.vpc_cidr


  tags = {
    Name = var.project
  }
}


resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.my_vpc.id
```

```
  tags = {
    Name = "${var.project}-igw"
  }
}
resource "aws_route_table" "custom_rt" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "${var.project}-route-table"
  }
}

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.custom_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_subnet" "my_subnet" {
  vpc_id                  = aws_vpc.my_vpc.id
```

```
  cidr_block              = var.subnet_cidr
  availability_zone       = var.az

  tags = {
    Name = "${var.project}-subnet"
  }
}
resource "aws_route_table_association"
"subnet_association" {
  subnet_id      = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.custom_rt.id
}
resource "aws_security_group" "web_sg" {
  name        = "${var.project}-sg"
  vpc_id      = aws_vpc.my_vpc.id

  ingress {
    description = "SSH"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
```

```
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  description = "HTTP"
  from_port   = 80
  to_port     = 80
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  description = "HTTPS"
  from_port   = 443
  to_port     = 443
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

egress {
  description = "Allow all outbound"
```

```
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "${var.project}-sg"
  }
}
resource "aws_network_interface" "my_eni" {
  subnet_id       = aws_subnet.my_subnet.id
  private_ips     = [var.eni_private_ip]
  security_groups = [aws_security_group.web_sg.id]

  tags = {
    Name = "${var.project}-eni"
  }
}
resource "aws_eip" "eni_eip" {
  domain = "vpc"
```

```
  network_interface = aws_network_interface.my_eni.id
  associate_with_private_ip = var.eni_private_ip

  tags = {
    Name = "${var.project}-eni-eip"
  }
}
resource "aws_instance" "ubuntu_server" {
  ami           = "ami-0c02fb55956c7d316"
  instance_type = "t3.micro"
  key_name = "red"

  network_interface {
    network_interface_id = aws_network_interface.my_eni.id
    device_index         = 0
  }

  user_data = <<-EOF
    #!/bin/bash
    apt-get update -y
```

```
    apt-get install apache2 -y
    systemctl enable apache2
    systemctl start apache2
  EOF

  tags = {
    Name = "${var.project}-ubuntu-server"
  }
}
```

Variables.tf:

```
variable "project" {
  default = "demo-vpc"
}

variable "vpc_cidr" {
  default = "10.0.0.0/16"
}
variable "subnet_cidr" {
  default     = "10.0.1.0/24"
```

```
}


variable "az" {
  default     = "us-east-1a"
}
variable "eni_private_ip" {
  default     = "10.0.1.10"
}
```