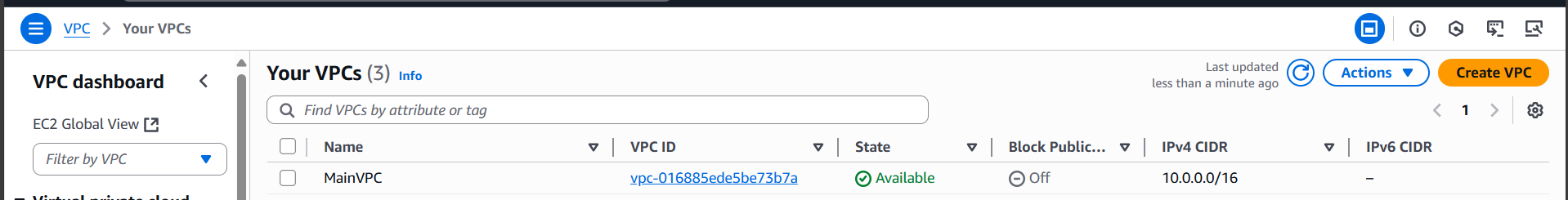
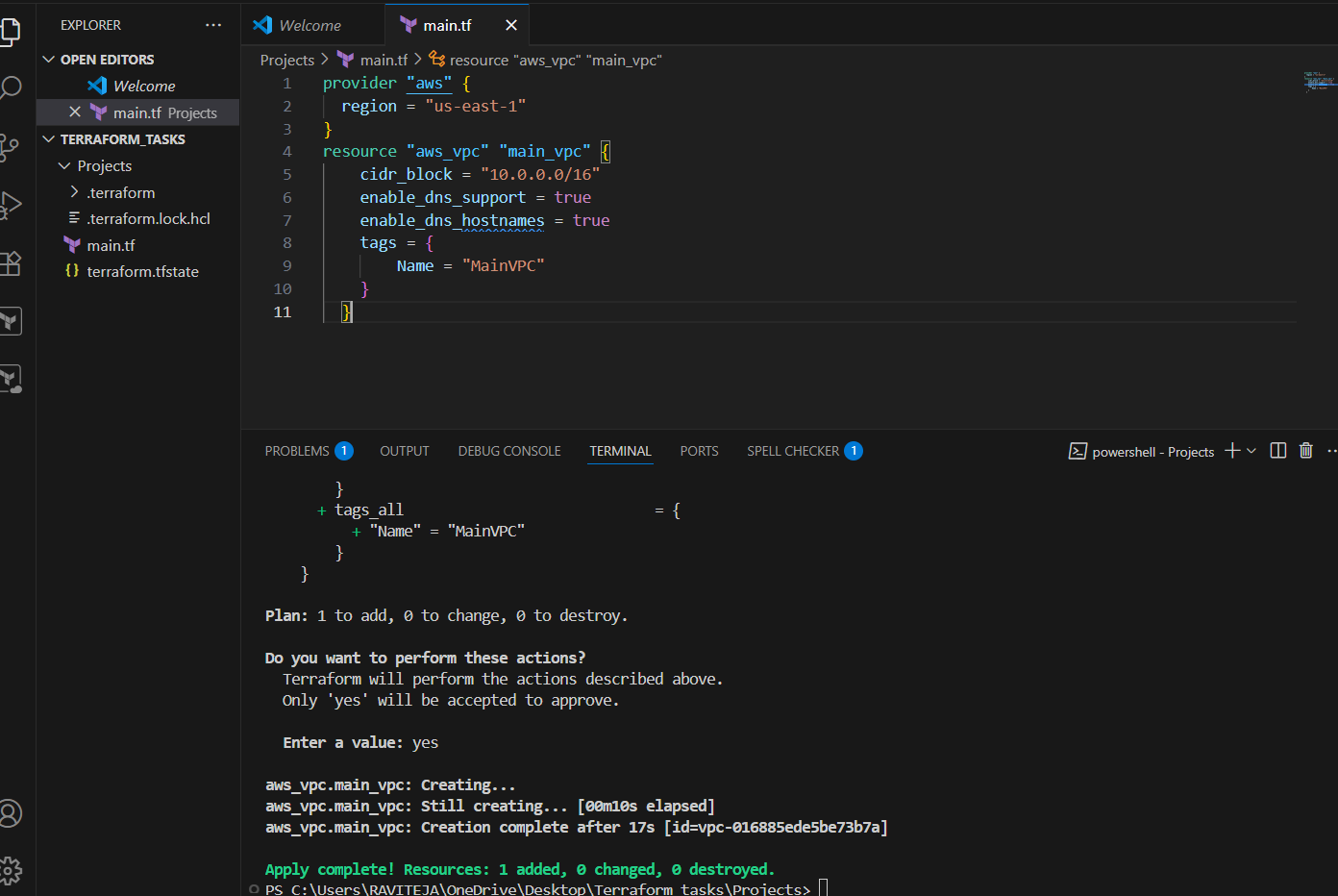
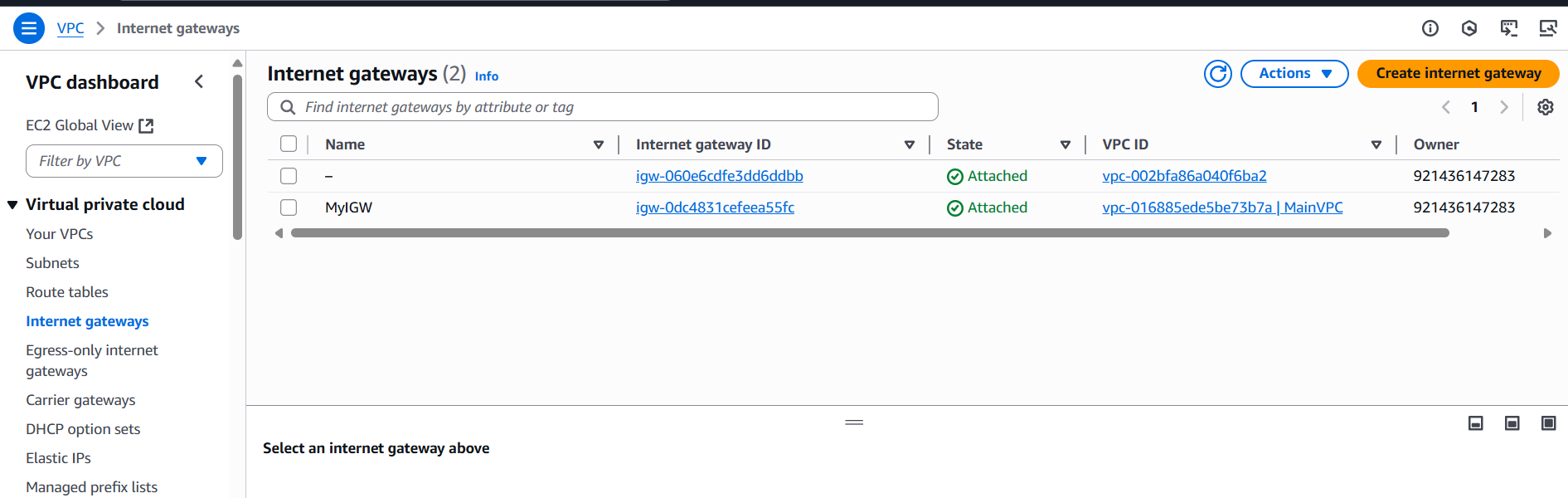
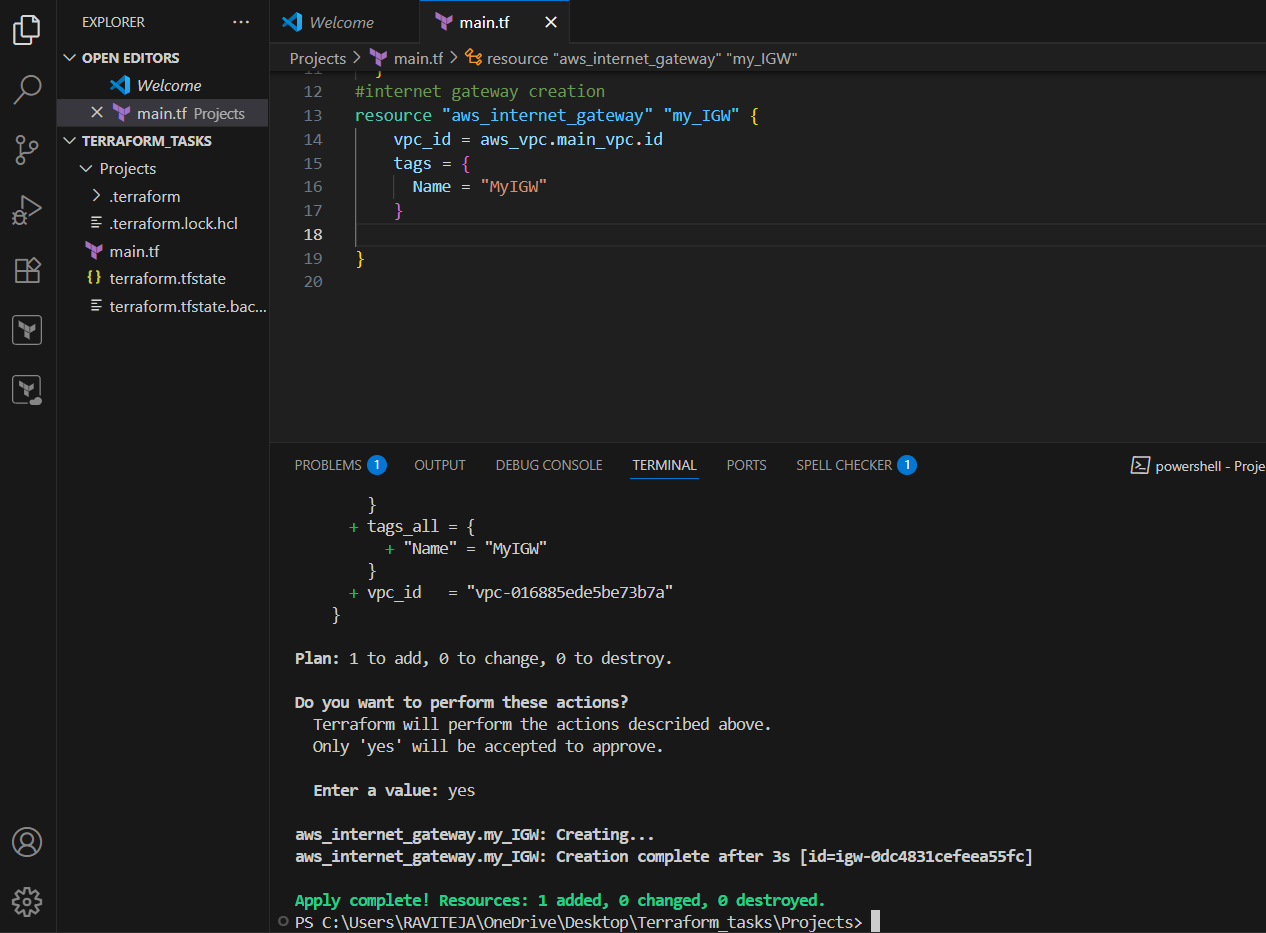
1. Create VPC

Created VPC Name: MainVPC



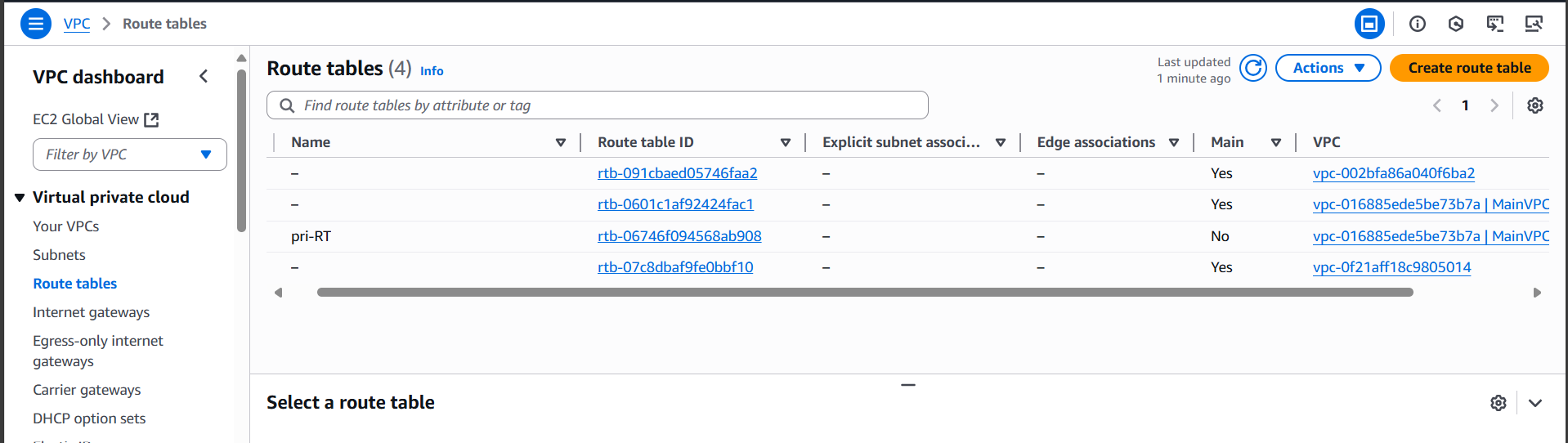
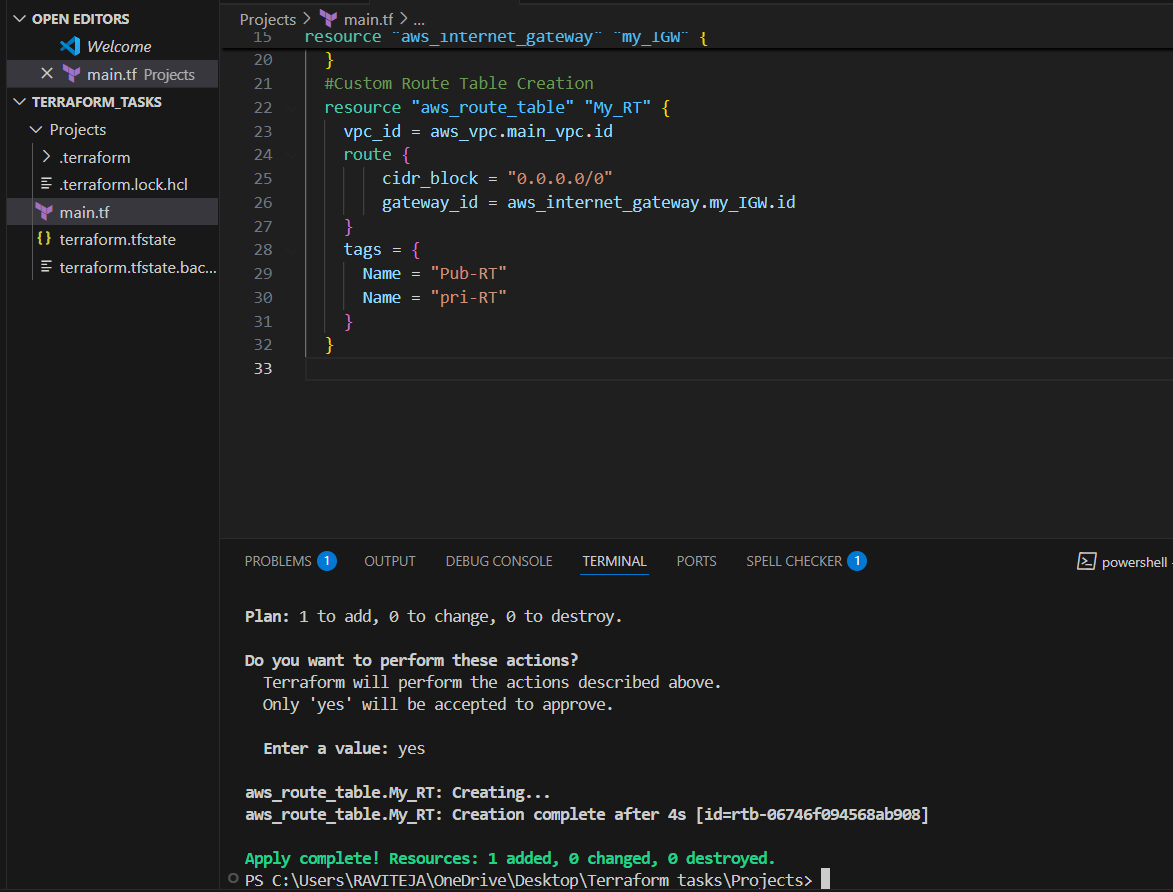
1. Create Internet gateway

Created IGW Name:MyIGW

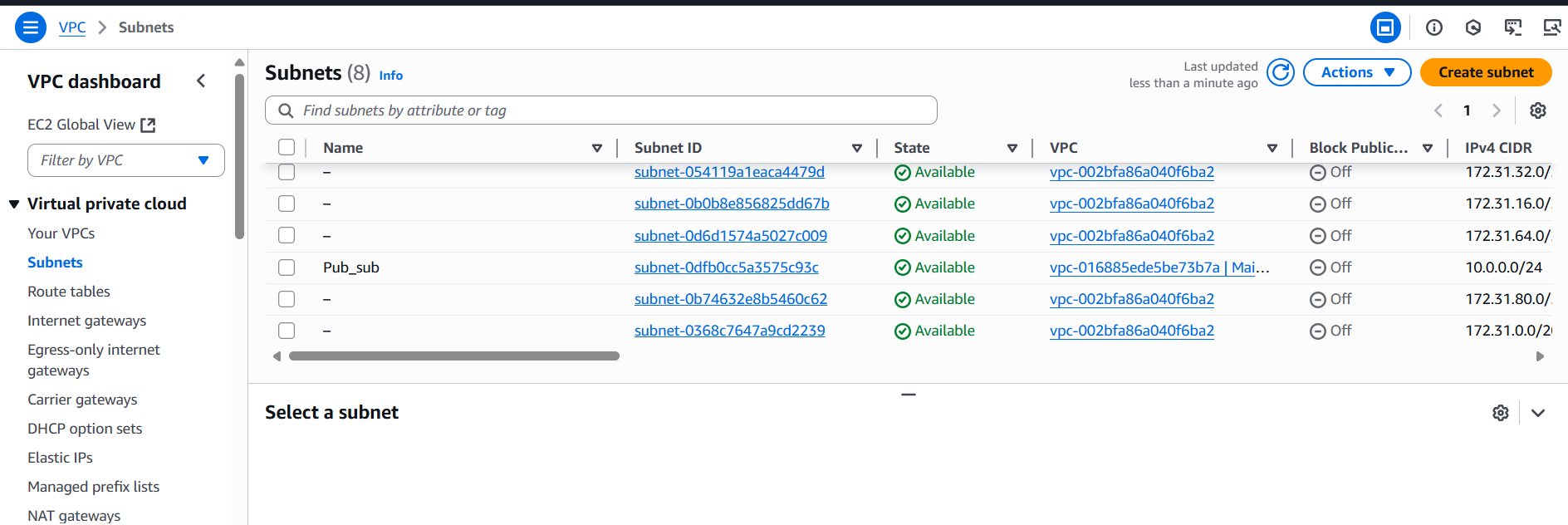
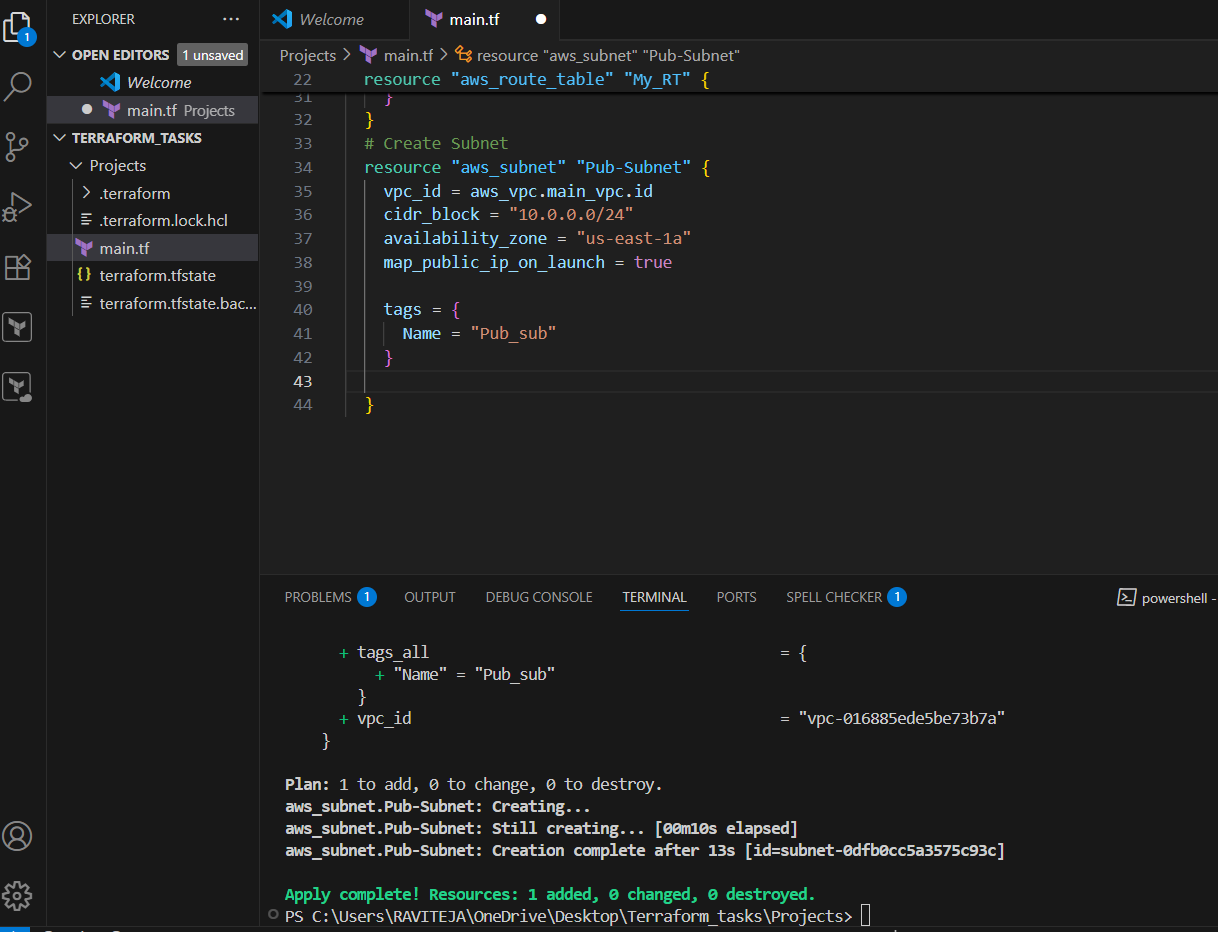


1. Create Custom Route Table

Custom route table Created

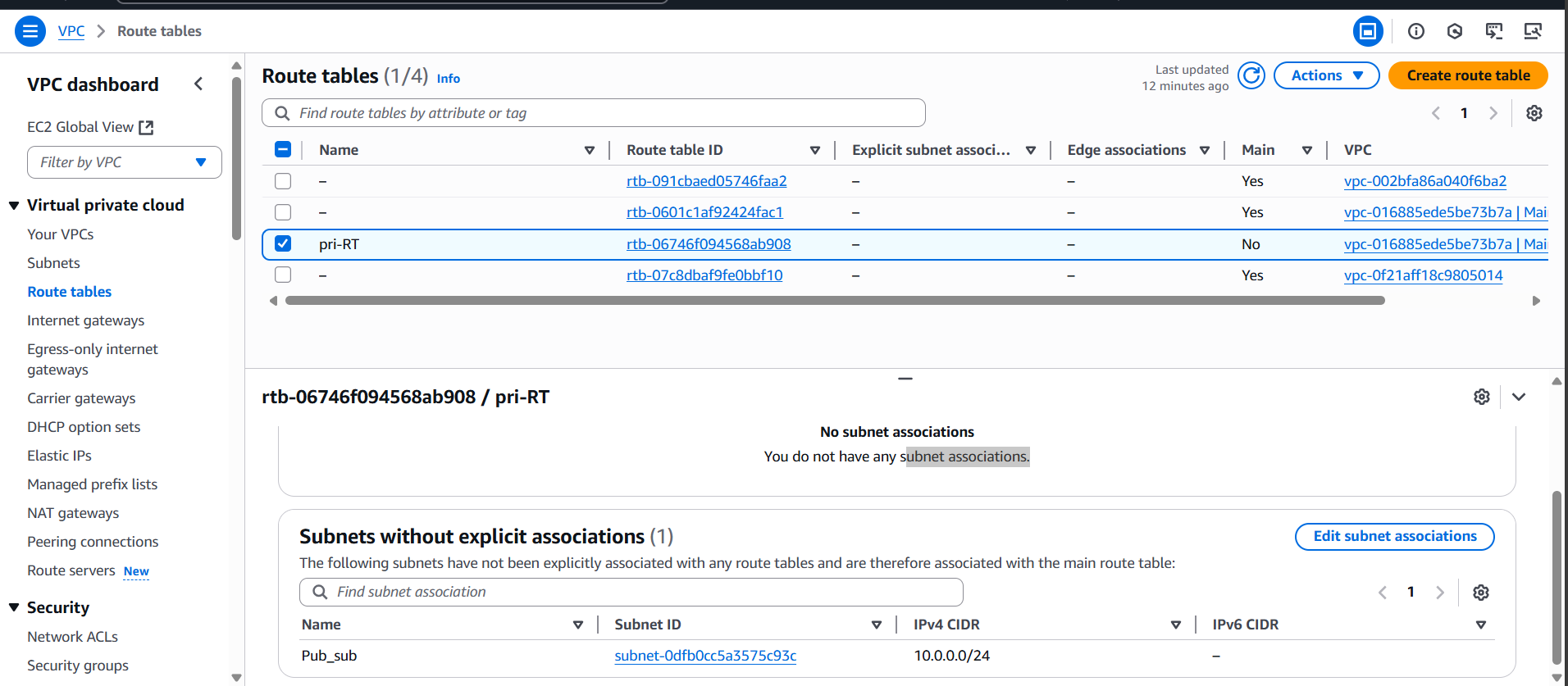
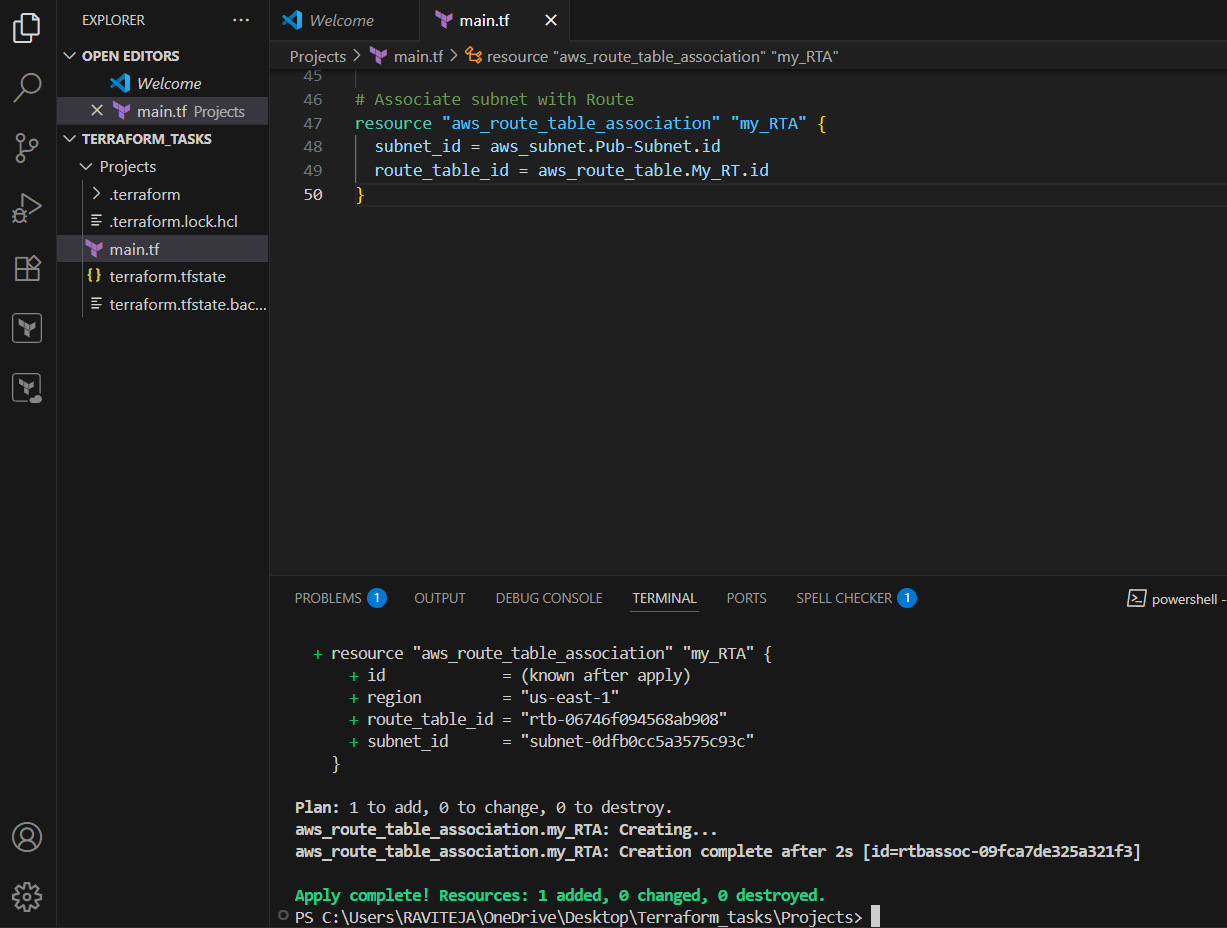


1. Create Subnet



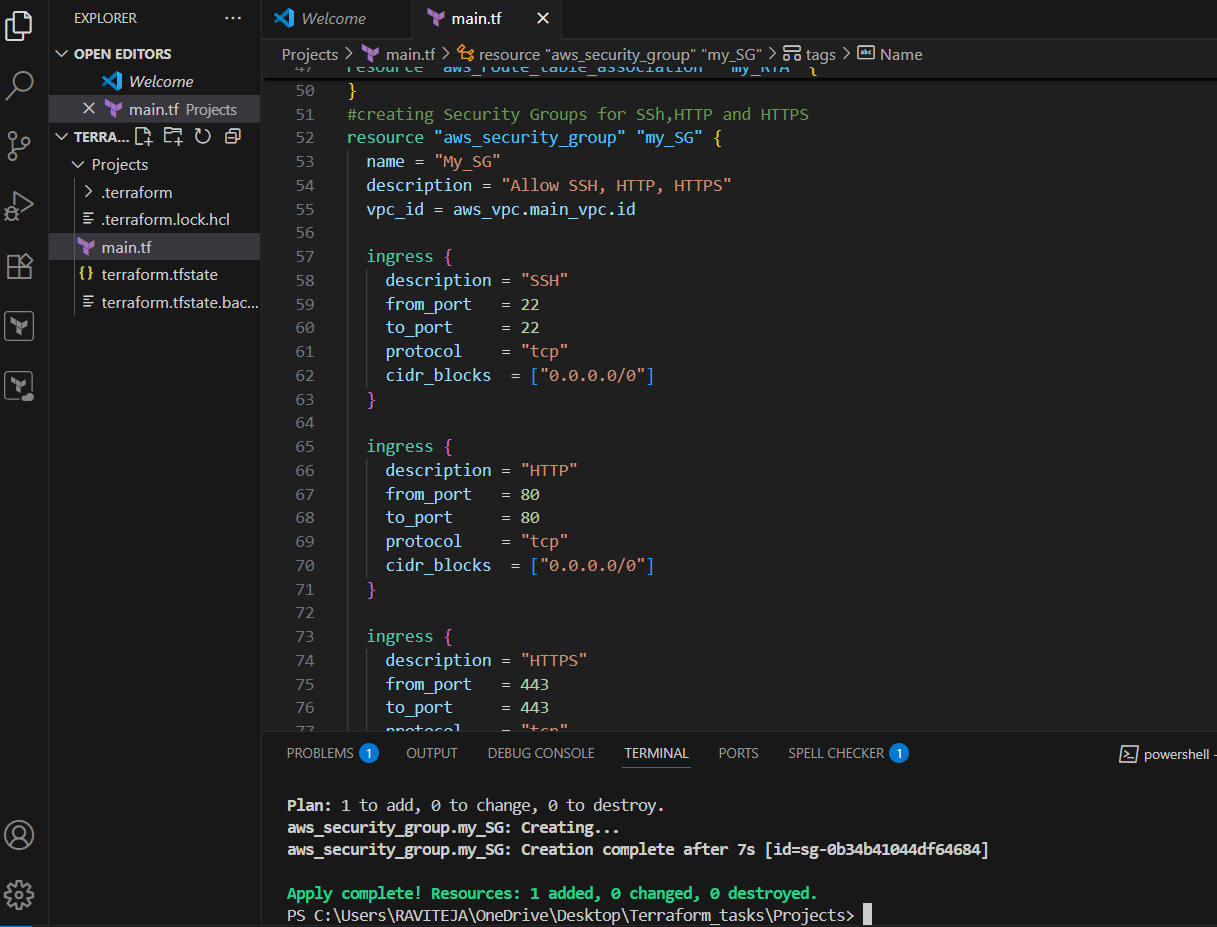
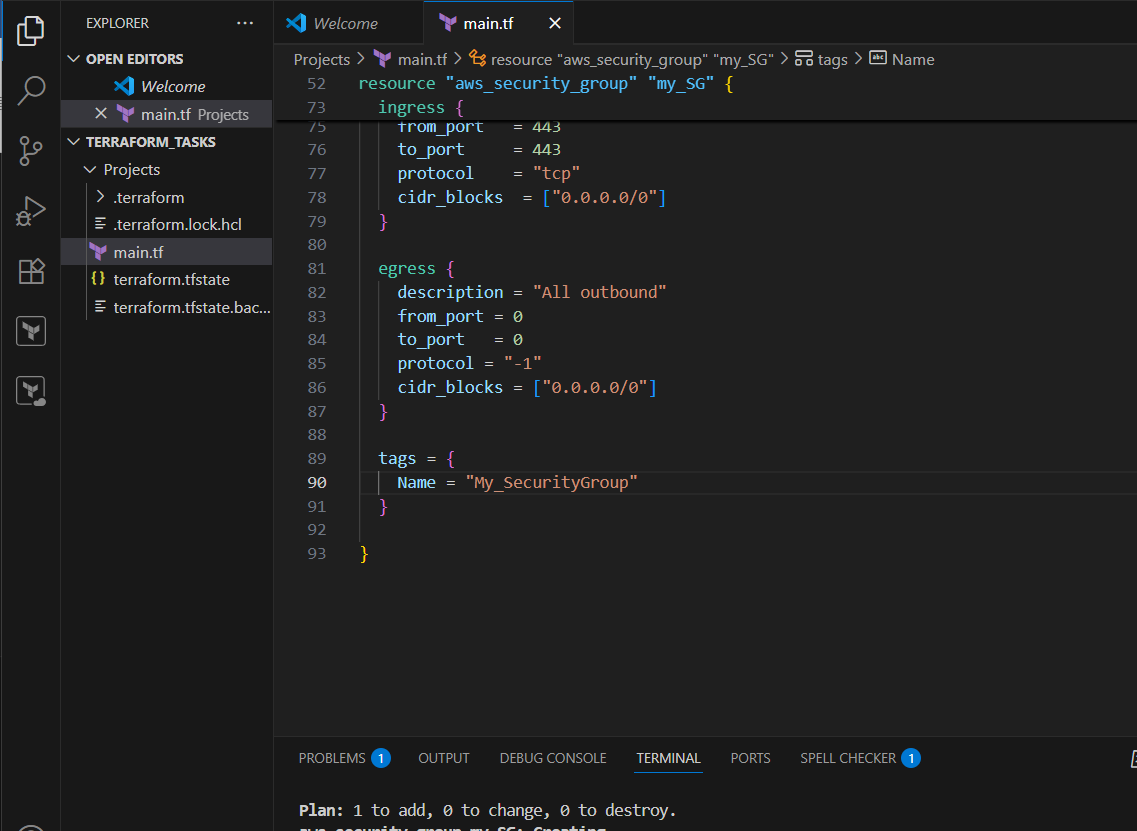
5) Associate subnet with Route Table

Route\_table\_subnet\_association Created

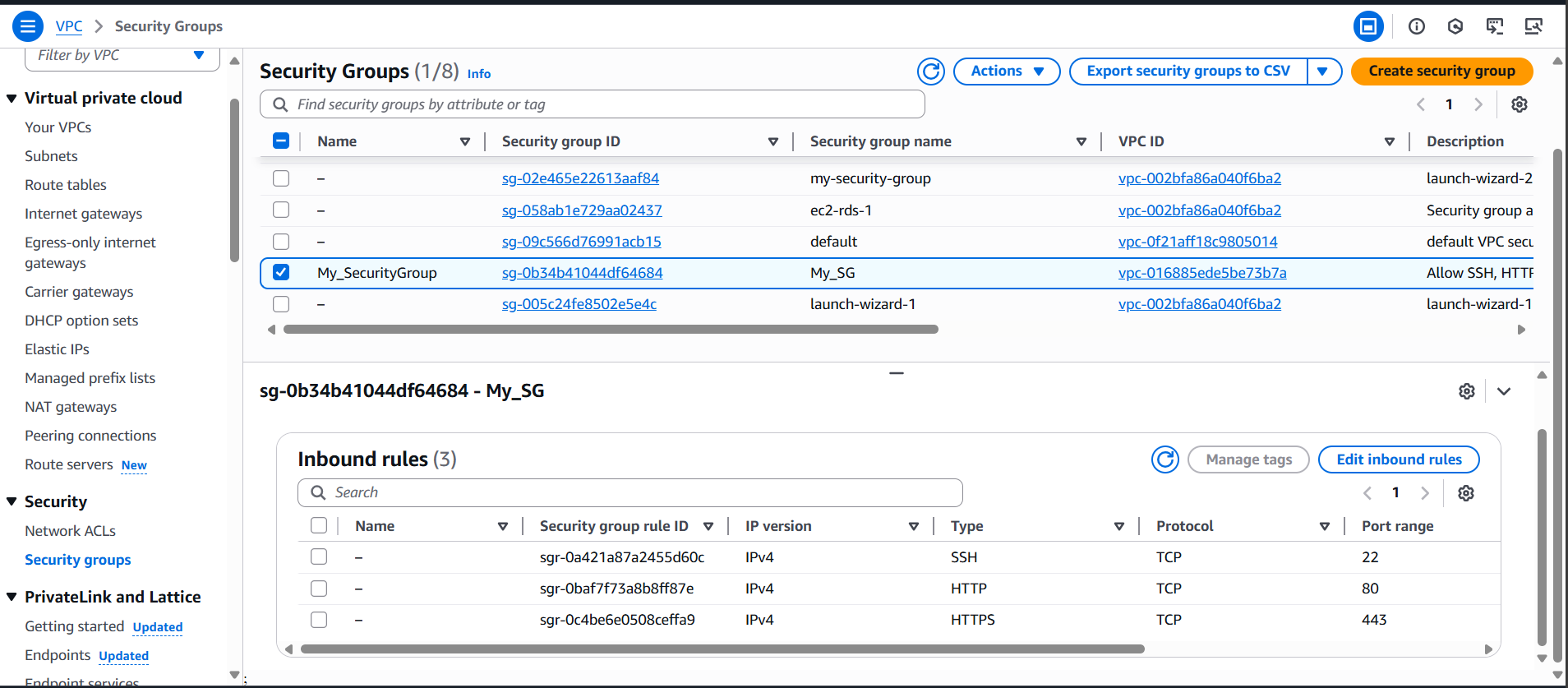


1. Create Security Group to allow port 22.80,443

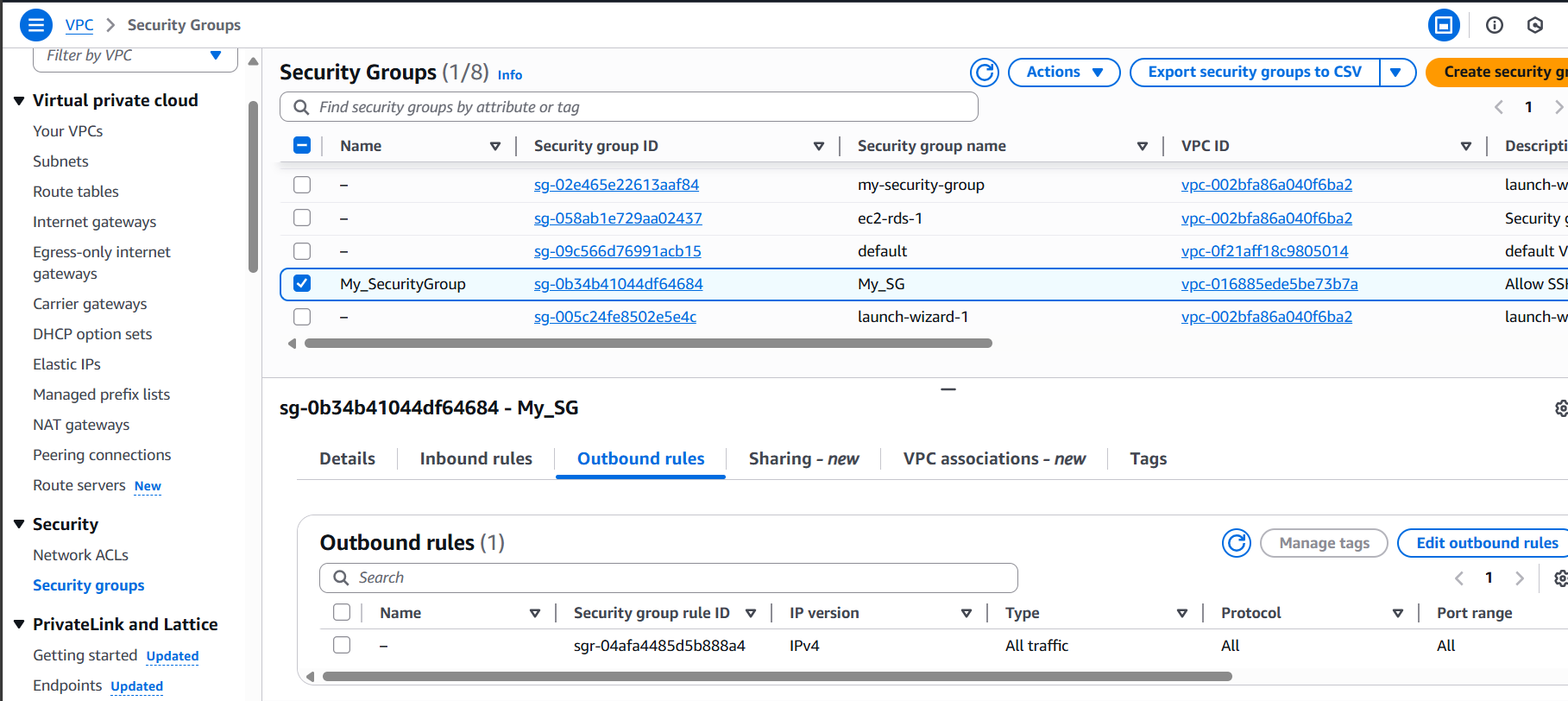
Created Security Group to allow port 22.80 and 443

Inbound rules

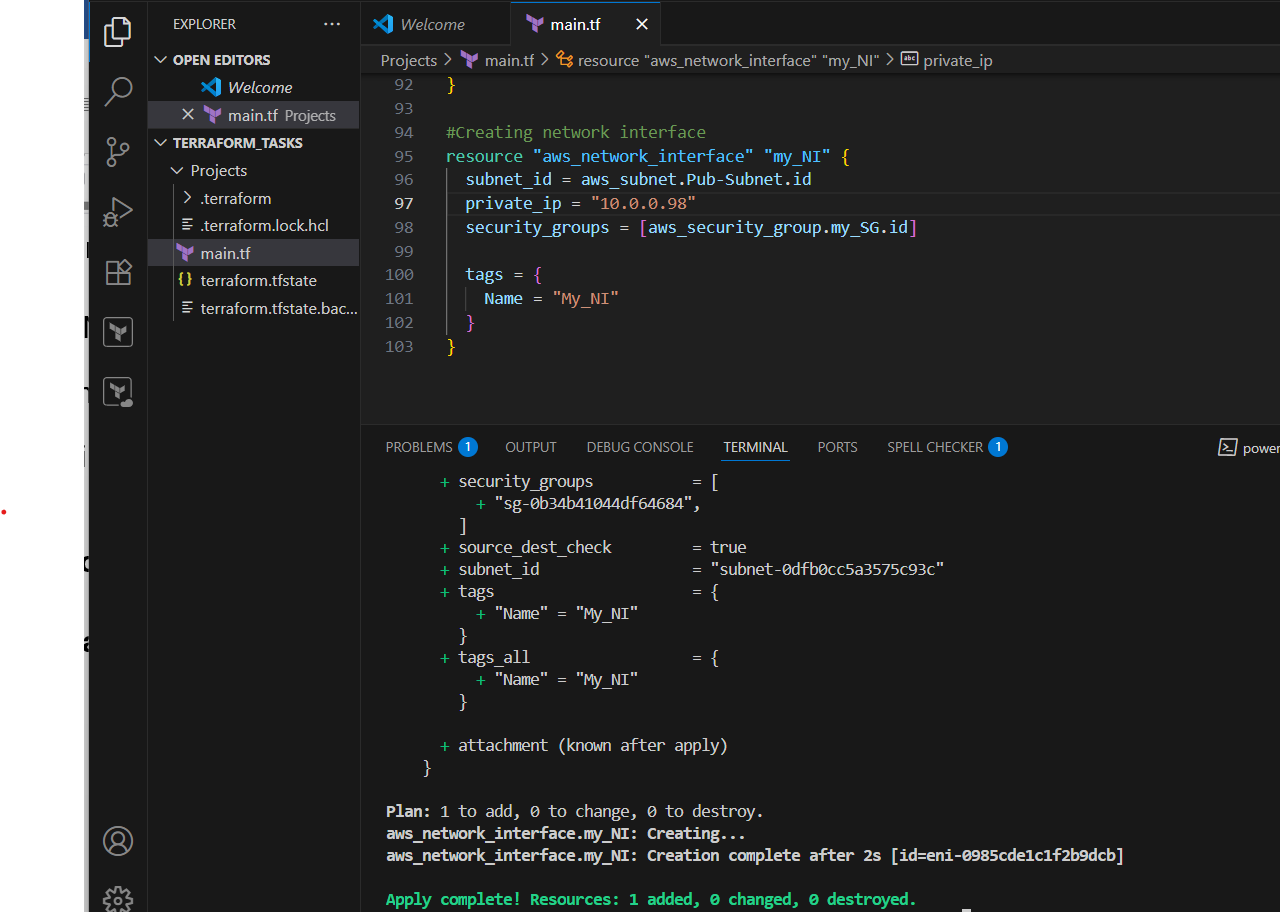


Outbound rules

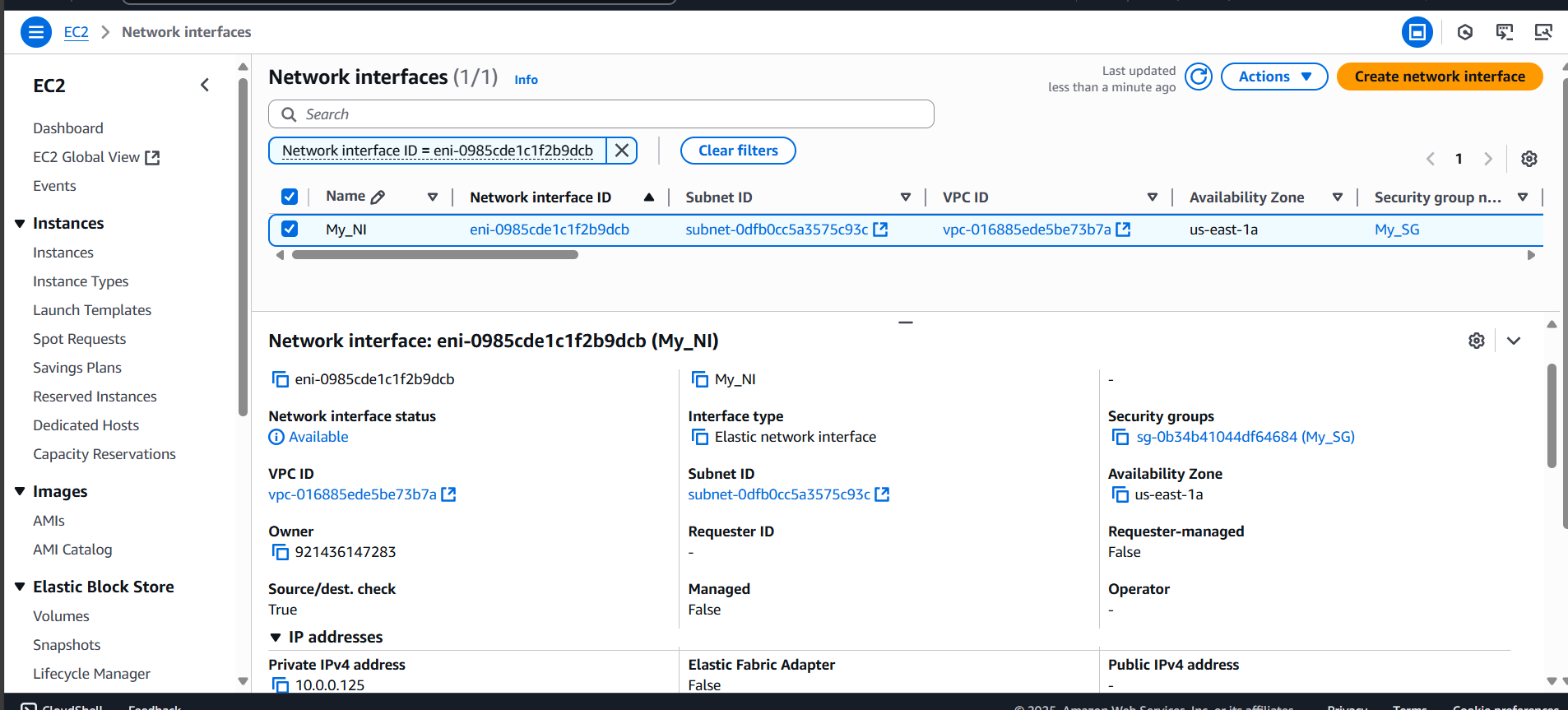


1. Create a network interface with an ip in the subnet that was created in step 4

Created “NETWORK INTERFACE WITH AN PRIVATE IP IN THE SUBNET”

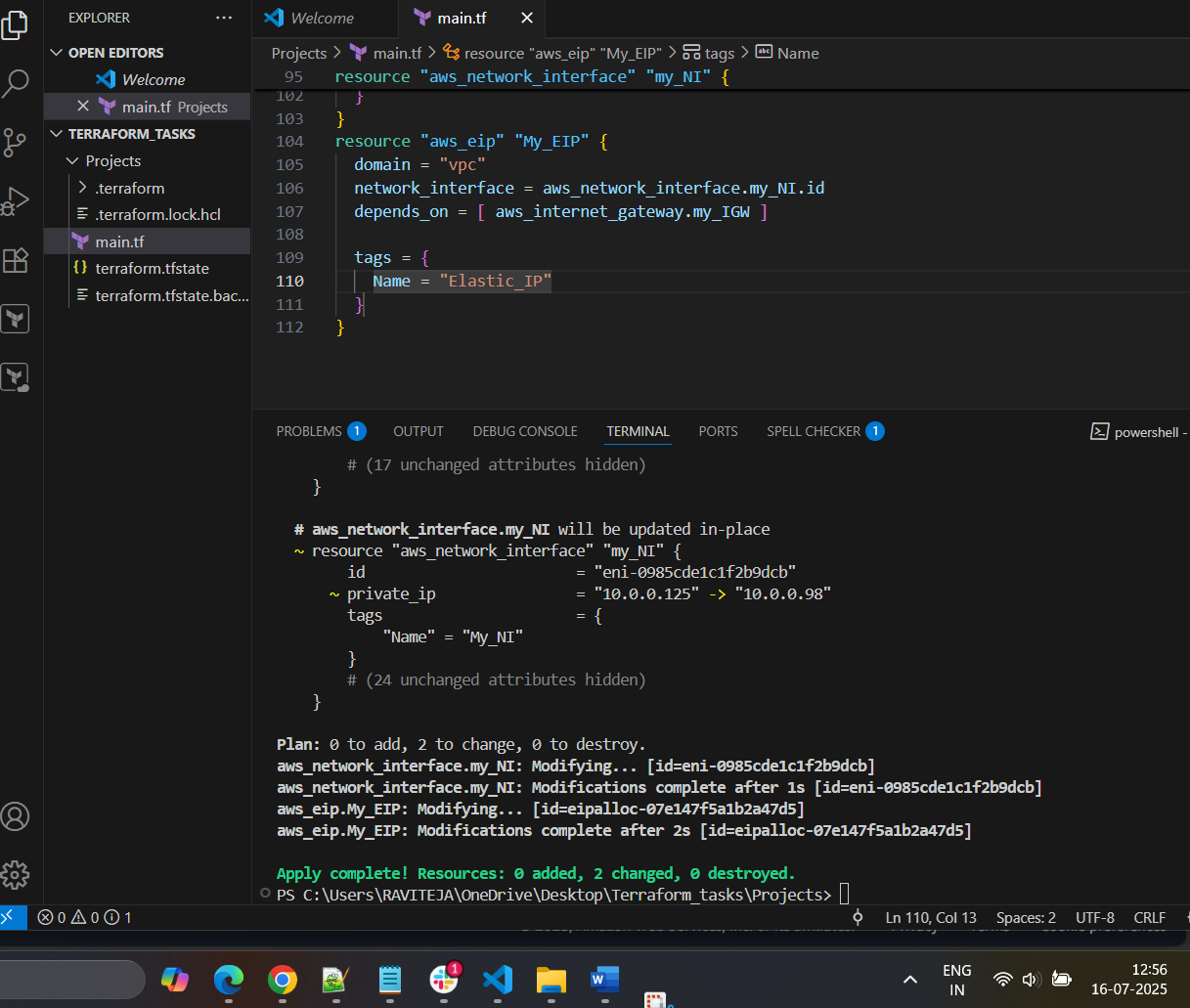
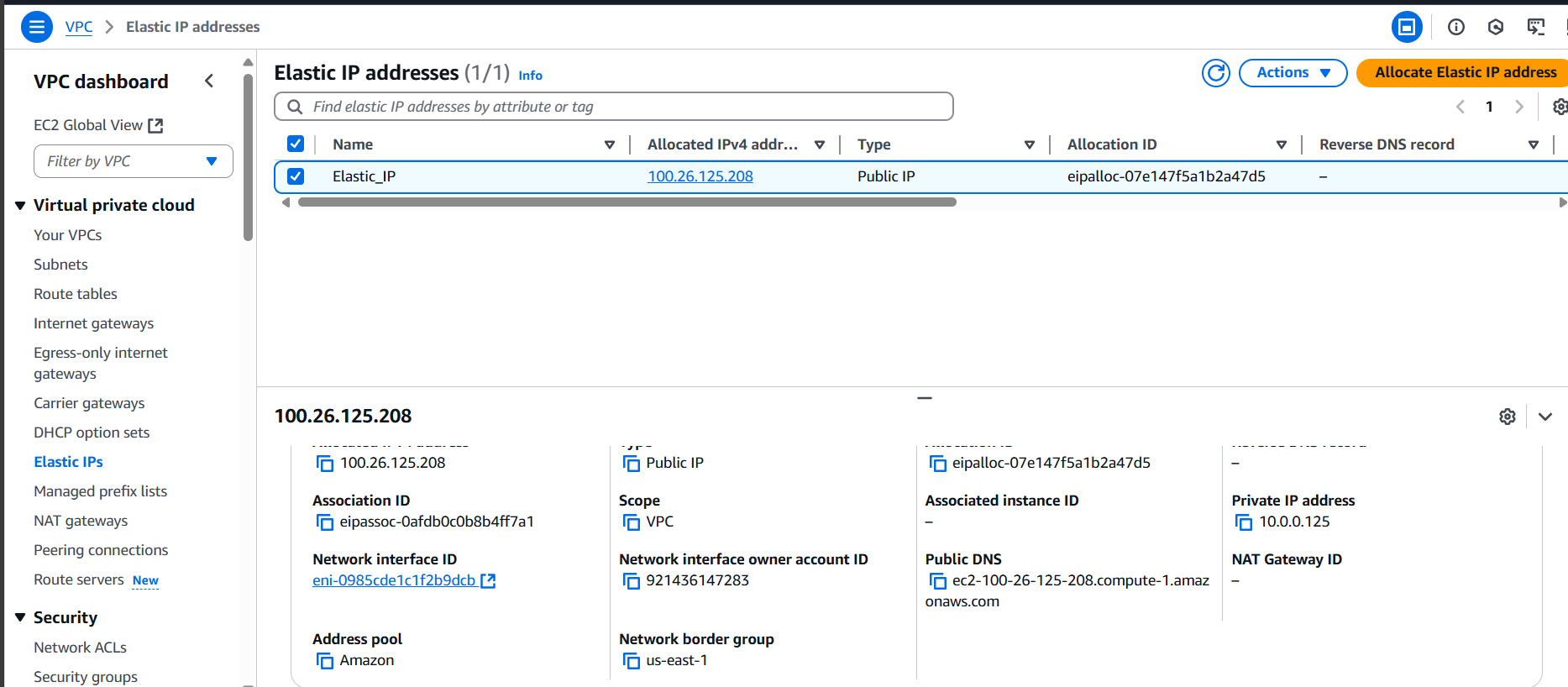


Network interface:



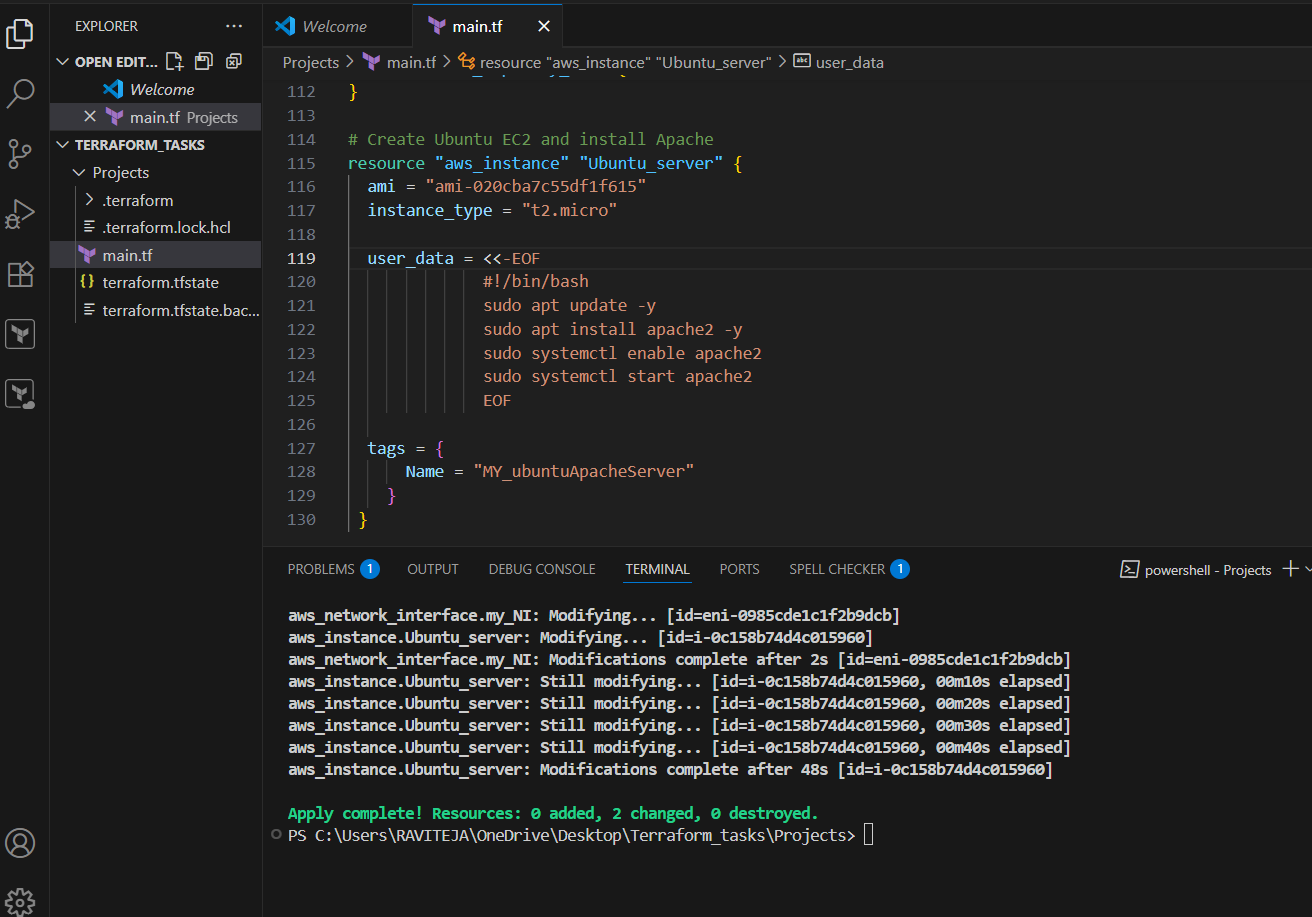
1. Assign an elastic IP to the network interface created in step 7

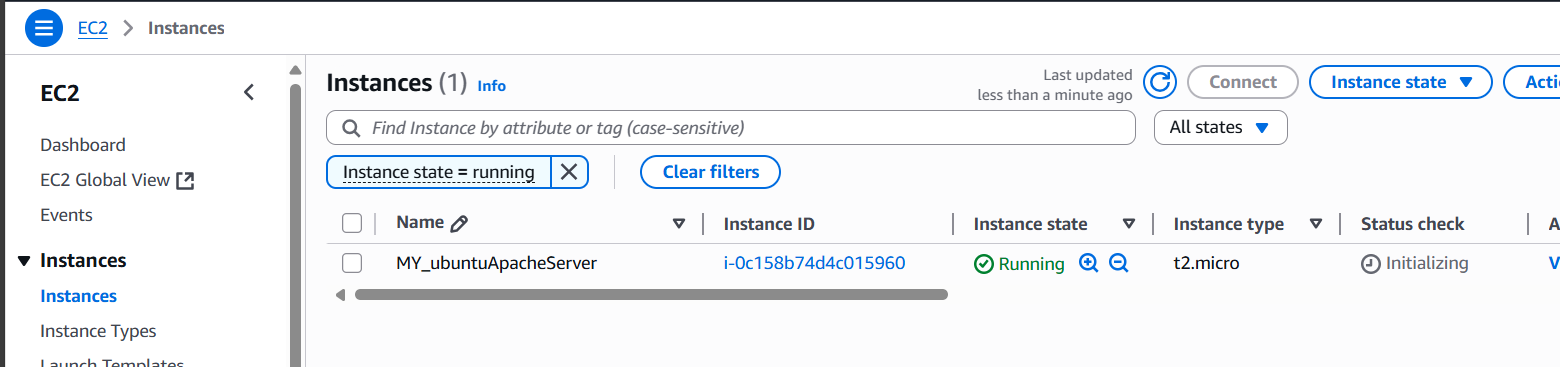
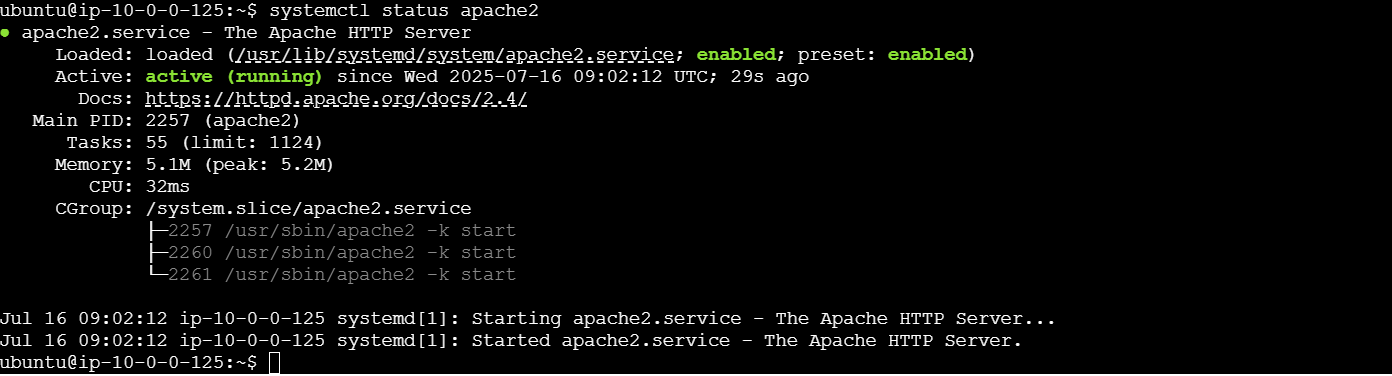
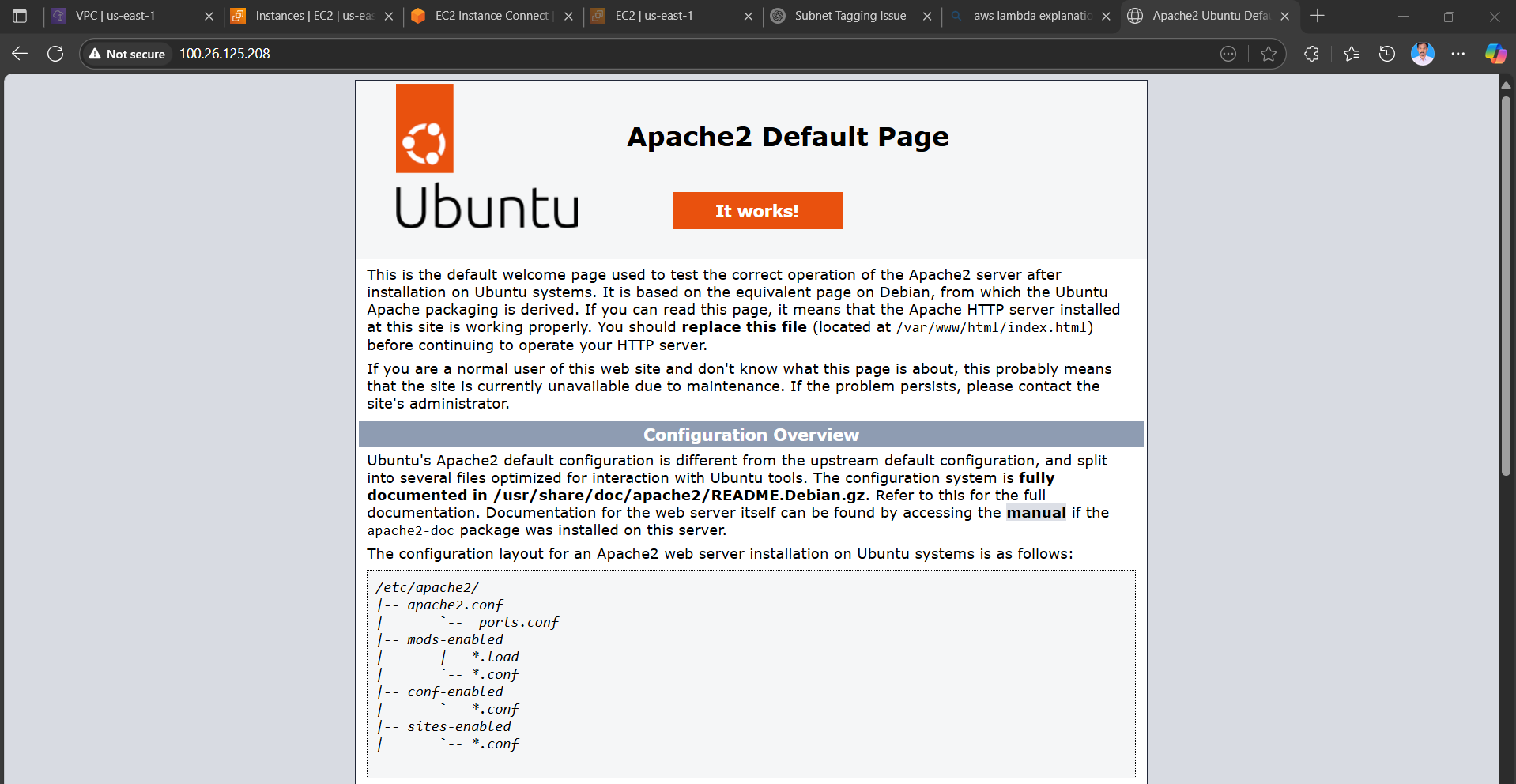
Assigned an elastic IP to network interfaces

1. Create Ubuntu server and install/enable apache2

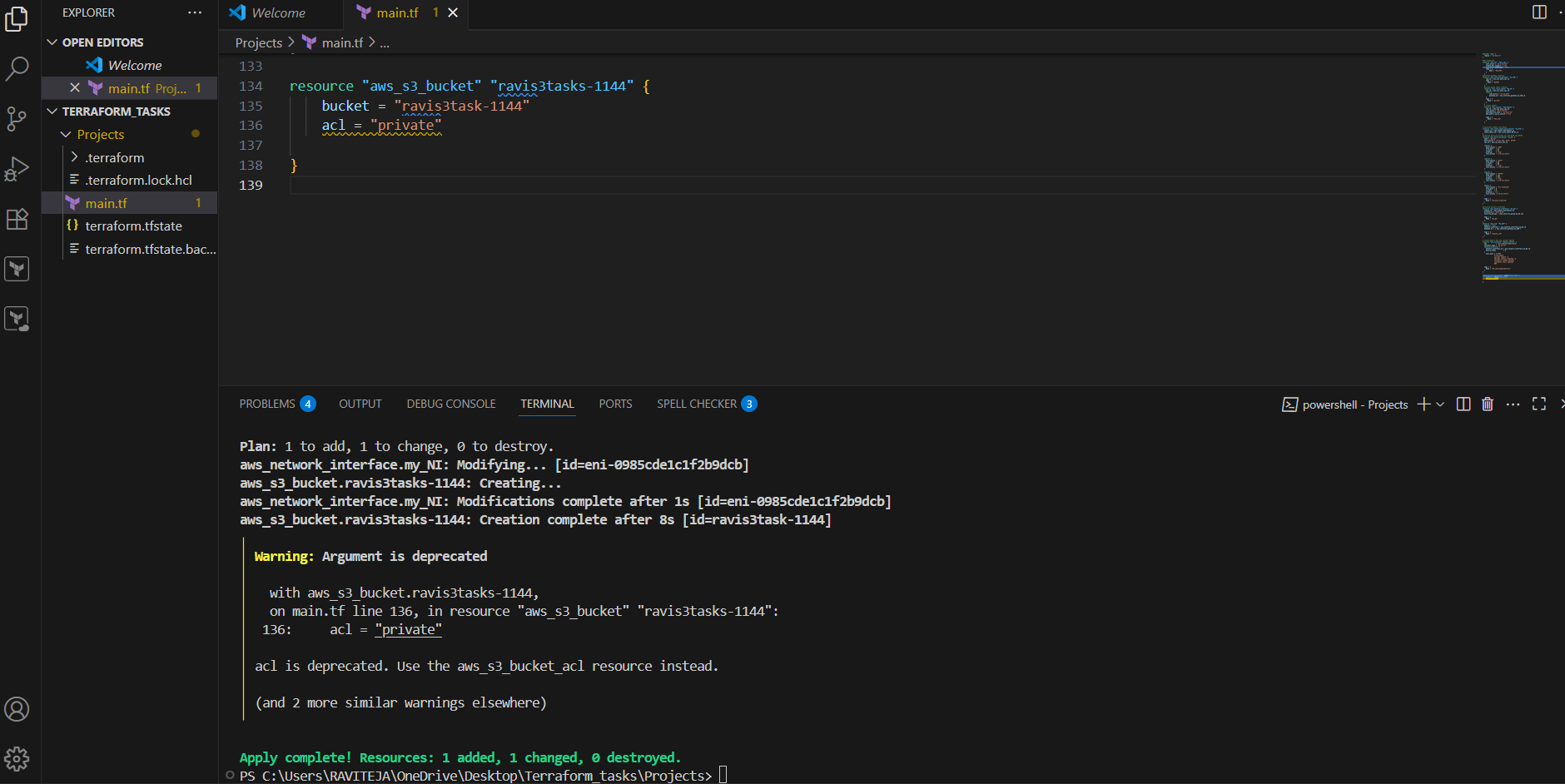
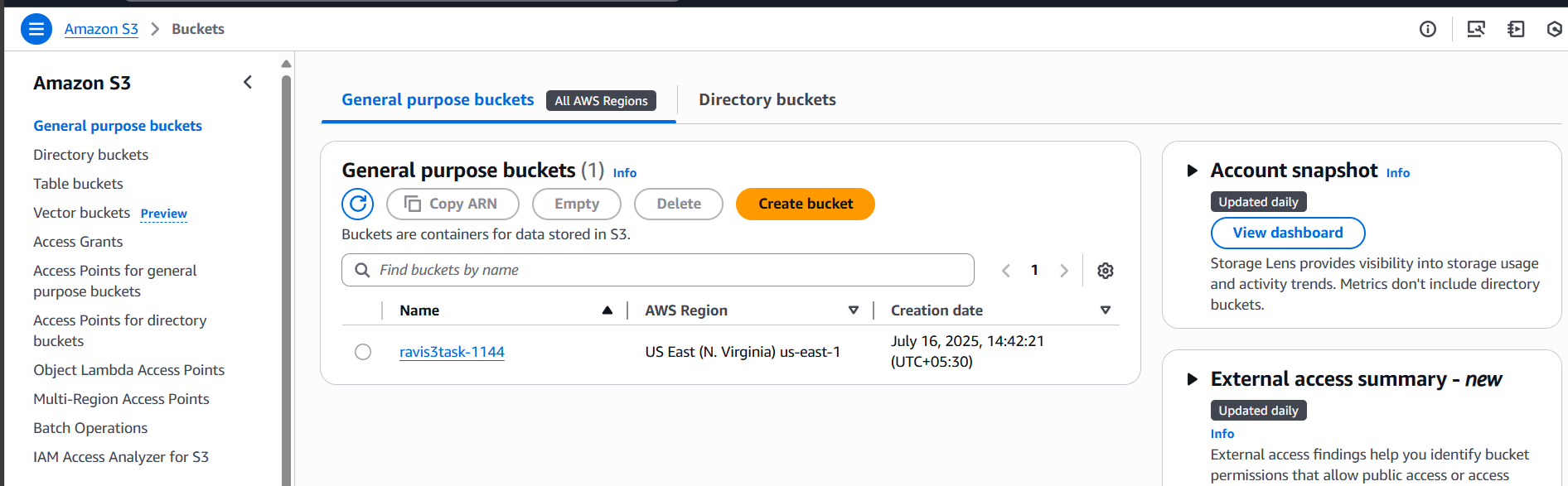
Created ec2 instace with Ubuntu server in apache2

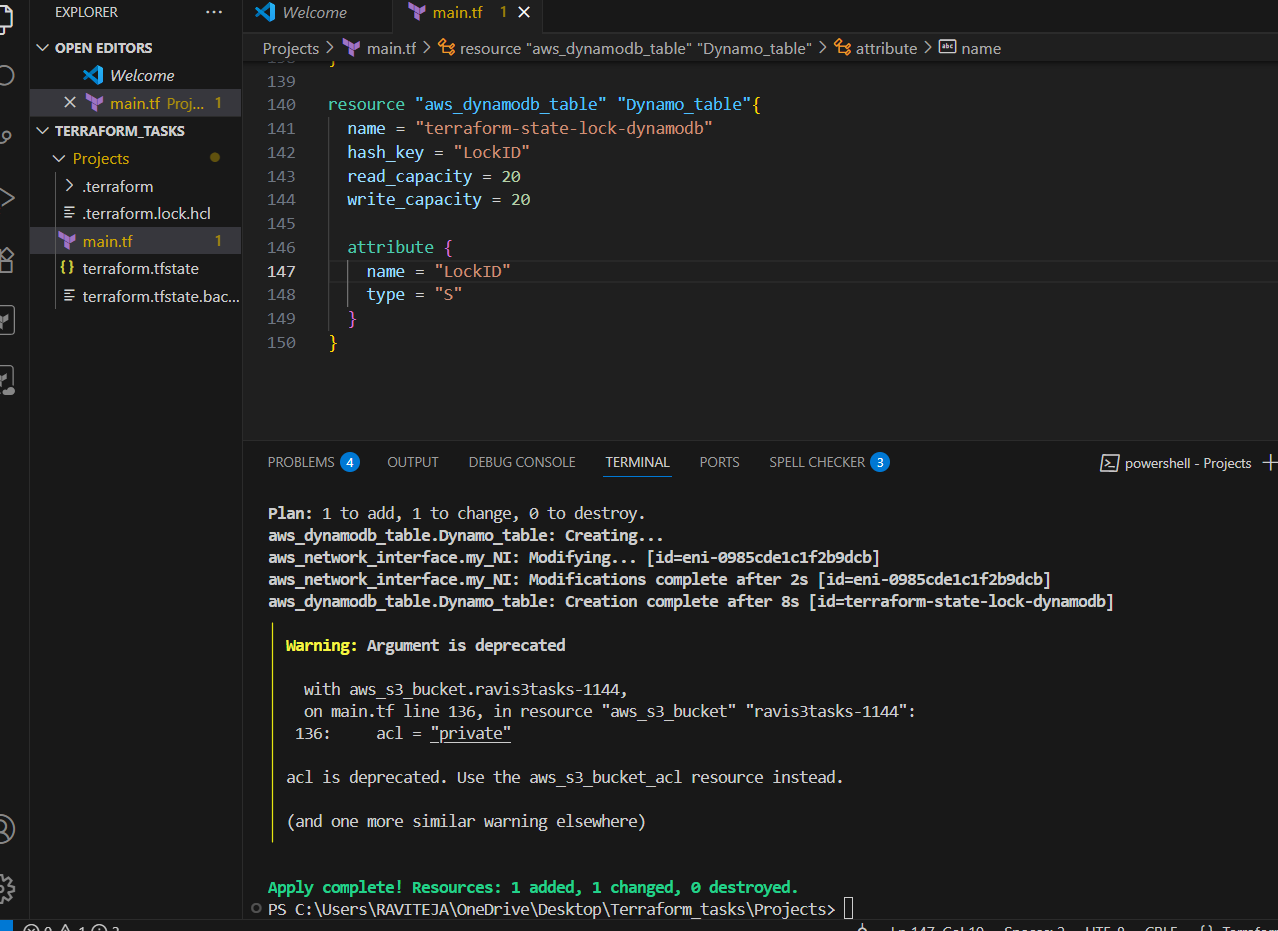
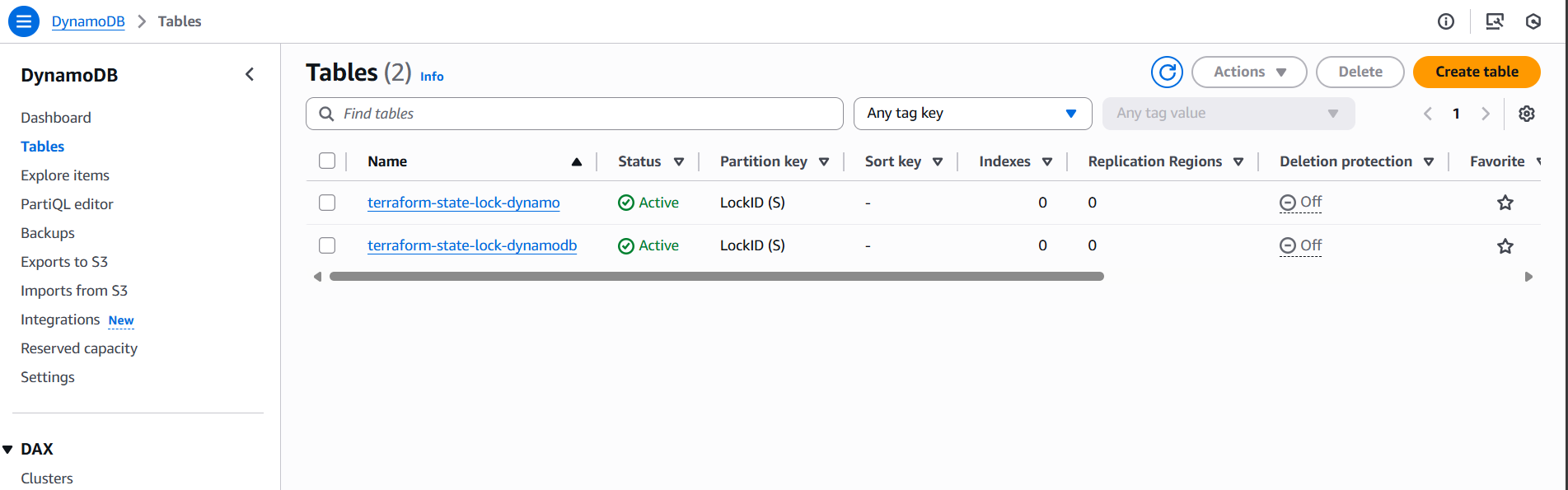


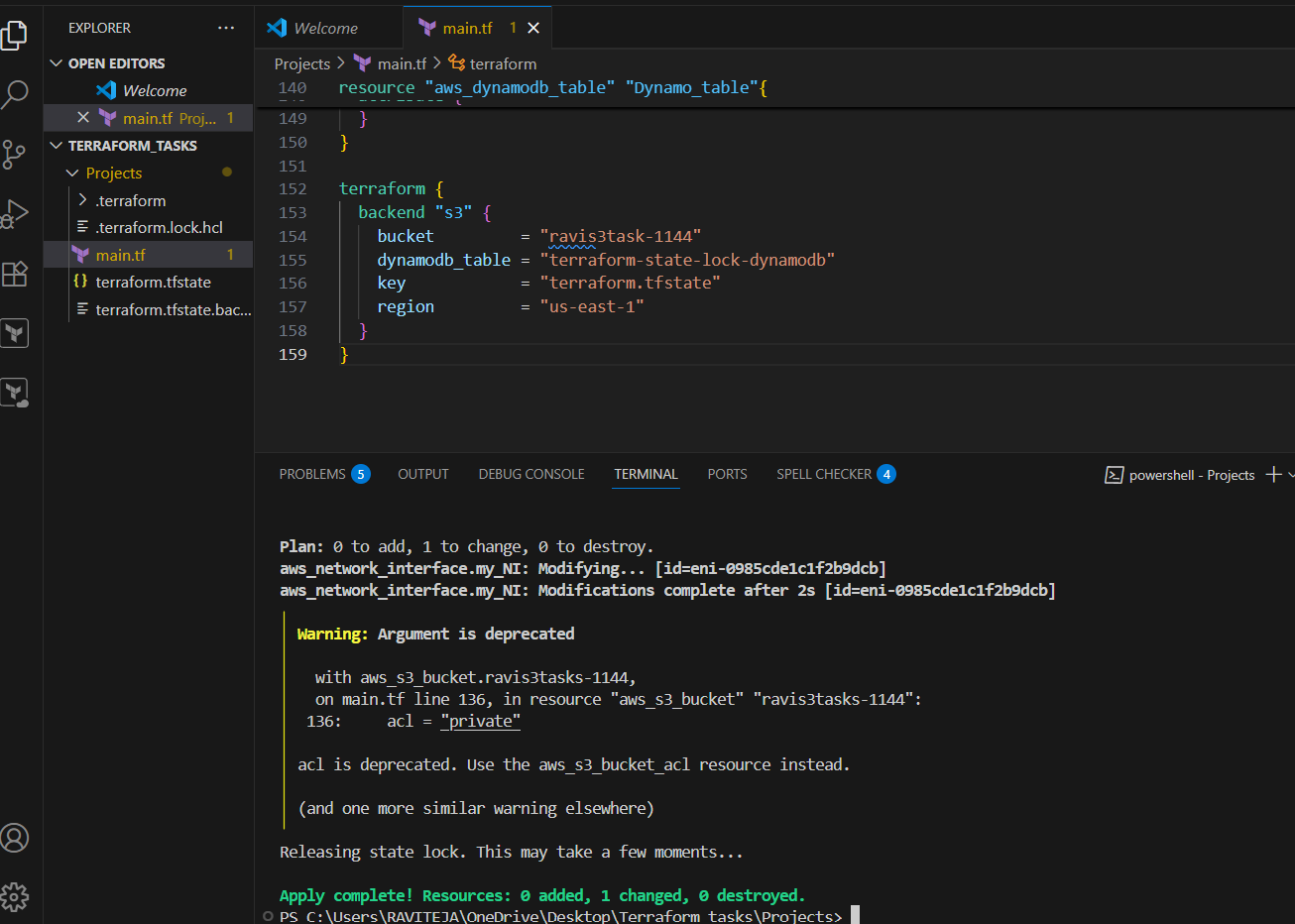
**Note:**  
**1) Create single main.tf which will be created the above resources and do not hardcode the id's.**

**2) Configure s3 as backend and dynamo db locking for multi user execution.**

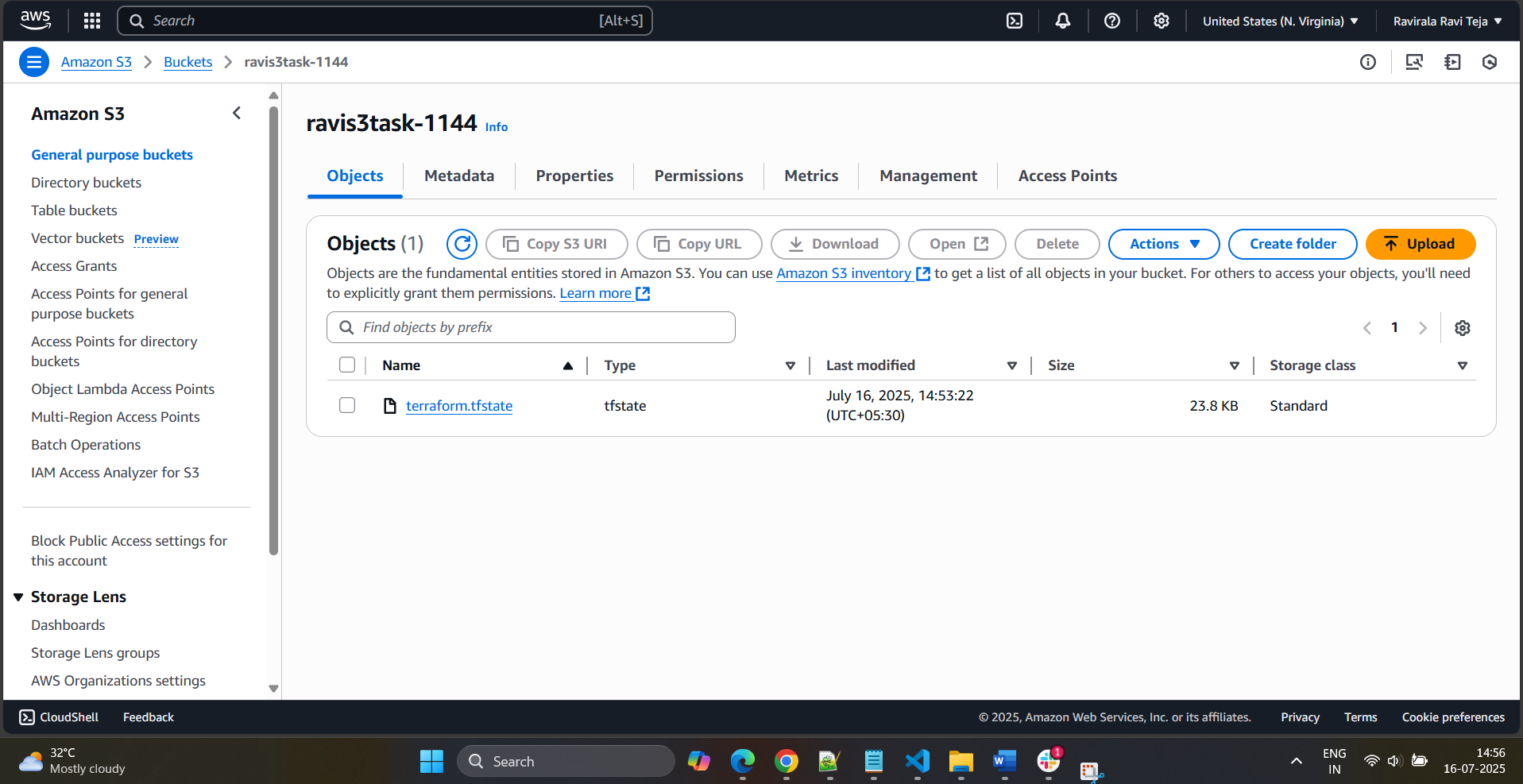
**Created S3 bucket for store state lock** ****

Created DynamoDB Table for Locking  
 

Created backed S3 to store the terraform.tfstate file



Stored in s3 bucket:



Terraform code for vpc,IGW,route tables,subnets,subnet association,network interface,elastic IP,created instance using ubuntu, s3 bucket,dynamodb table, terraform backend locked file to store s3

provider "aws" {

  region = "us-east-1"

}

#VPC CreationS

resource "aws\_vpc" "main\_vpc" {

  cidr\_block           = "10.0.0.0/16"

  enable\_dns\_support   = true

  enable\_dns\_hostnames = true

  tags = {

    Name = "MainVPC"

  }

}

#internet gateway creation

resource "aws\_internet\_gateway" "my\_IGW" {

  vpc\_id = aws\_vpc.main\_vpc.id

  tags = {

    Name = "MyIGW"

  }

}

#Custom Route Table Creation

resource "aws\_route\_table" "My\_RT" {

  vpc\_id = aws\_vpc.main\_vpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.my\_IGW.id

  }

  tags = {

    Name = "pri-RT"

  }

}

# Create Subnet

resource "aws\_subnet" "Pub-Subnet" {

  vpc\_id                  = aws\_vpc.main\_vpc.id

  cidr\_block              = "10.0.0.0/24"

  availability\_zone       = "us-east-1a"

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "Pub\_sub"

  }

}

# Associate subnet with Route

resource "aws\_route\_table\_association" "my\_RTA" {

  subnet\_id      = aws\_subnet.Pub-Subnet.id

  route\_table\_id = aws\_route\_table.My\_RT.id

}

#creating Security Groups for SSh,HTTP and HTTPS

resource "aws\_security\_group" "my\_SG" {

  name        = "My\_SG"

  description = "Allow SSH, HTTP, HTTPS"

  vpc\_id      = aws\_vpc.main\_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 = "All outbound"

    from\_port   = 0

    to\_port     = 0

    protocol    = "-1"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  tags = {

    Name = "My\_SecurityGroup"

  }

}

#Creating network interface

resource "aws\_network\_interface" "my\_NI" {

  subnet\_id       = aws\_subnet.Pub-Subnet.id

  private\_ip      = "10.0.0.98"

  security\_groups = [aws\_security\_group.my\_SG.id]

  tags = {

    Name = "My\_NI"

  }

}

resource "aws\_eip" "My\_EIP" {

  domain            = "vpc"

  network\_interface = aws\_network\_interface.my\_NI.id

  depends\_on        = [aws\_internet\_gateway.my\_IGW]

  tags = {

    Name = "Elastic\_IP"

  }

}

# Create Ubuntu EC2 and install Apache

resource "aws\_instance" "Ubuntu\_server" {

  ami           = "ami-020cba7c55df1f615"

  instance\_type = "t2.micro"

  network\_interface {

    network\_interface\_id = aws\_network\_interface.my\_NI.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 = "MY\_ubuntuApacheServer"

  }

}

resource "aws\_s3\_bucket" "ravis3tasks-1144" {

  bucket = "ravis3task-1144"

  acl    = "private"

}

resource "aws\_dynamodb\_table" "Dynamo\_table" {

  name           = "terraform-state-lock-dynamodb"

  hash\_key       = "LockID"

  read\_capacity  = 20

  write\_capacity = 20

  attribute {

    name = "LockID"

    type = "S"

  }

}

terraform {

  backend "s3" {

    bucket         = "ravis3task-1144"

    dynamodb\_table = "terraform-state-lock-dynamodb"

    key            = "terraform.tfstate"

    region         = "us-east-1"

  }

}