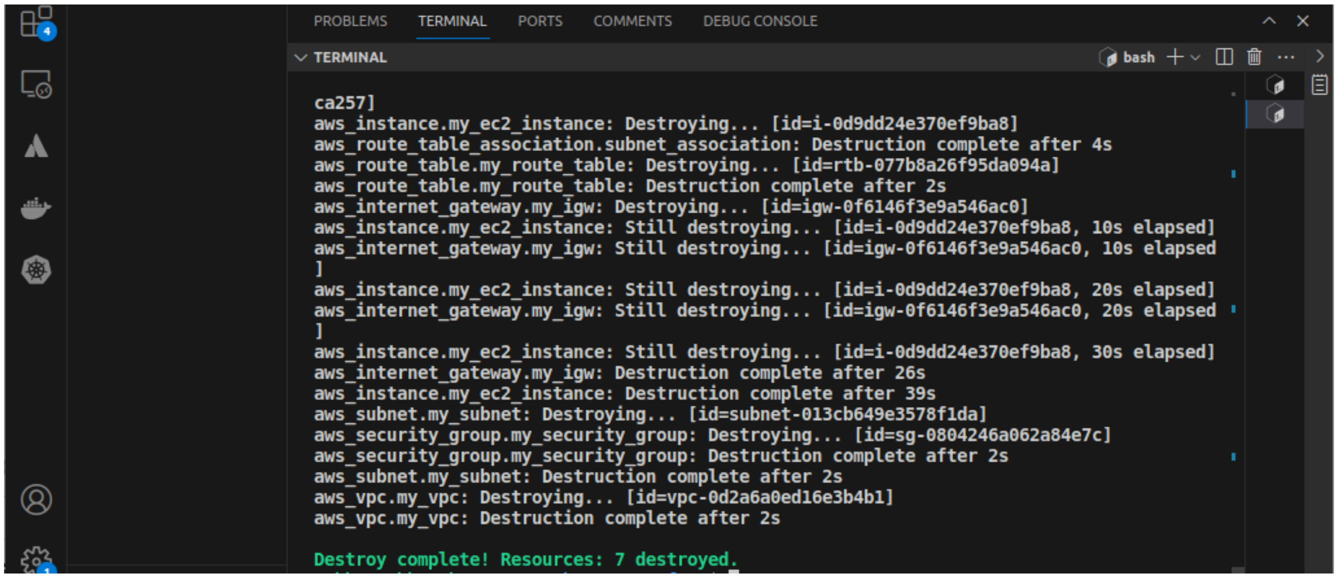
1. Destroy the previous deployments
2. Create a script to install Apache2
3. Run this script on a newly created EC2 instance
4. Print the IP address of the instance in a file on the local once deployed



```
ca257]
aws_instance.my_ec2_instance: Destroying... [id=i-0d9dd24e370ef9ba8]
aws_route_table_association.subnet_association: Destruction complete after 4s
aws_route_table.my_route_table: Destroying... [id=rtb-077b8a26f95da094a]
aws_route_table.my_route_table: Destruction complete after 2s
aws_internet_gateway.my_igw: Destroying... [id=igw-0f6146f3e9a546ac0]
aws_instance.my_ec2_instance: Still destroying... [id=i-0d9dd24e370ef9ba8, 10s elapsed]
aws_internet_gateway.my_igw: Still destroying... [id=igw-0f6146f3e9a546ac0, 10s elapsed
]
aws_instance.my_ec2_instance: Still destroying... [id=i-0d9dd24e370ef9ba8, 20s elapsed]
aws_internet_gateway.my_igw: Still destroying... [id=igw-0f6146f3e9a546ac0, 20s elapsed
]
aws_instance.my_ec2_instance: Still destroying... [id=i-0d9dd24e370ef9ba8, 30s elapsed]
aws_internet_gateway.my_igw: Destruction complete after 26s
aws_instance.my_ec2_instance: Destruction complete after 39s
aws_subnet.my_subnet: Destroying... [id=subnet-013cb649e3578f1da]
aws_security_group.my_security_group: Destroying... [id=sg-0804246a062a84e7c]
aws_security_group.my_security_group: Destruction complete after 2s
aws_subnet.my_subnet: Destruction complete after 2s
aws_vpc.my_vpc: Destroying... [id=vpc-0d2a6a0ed16e3b4b1]
aws_vpc.my_vpc: Destruction complete after 2s

Destroy complete! Resources: 7 destroyed.
```

No: 2

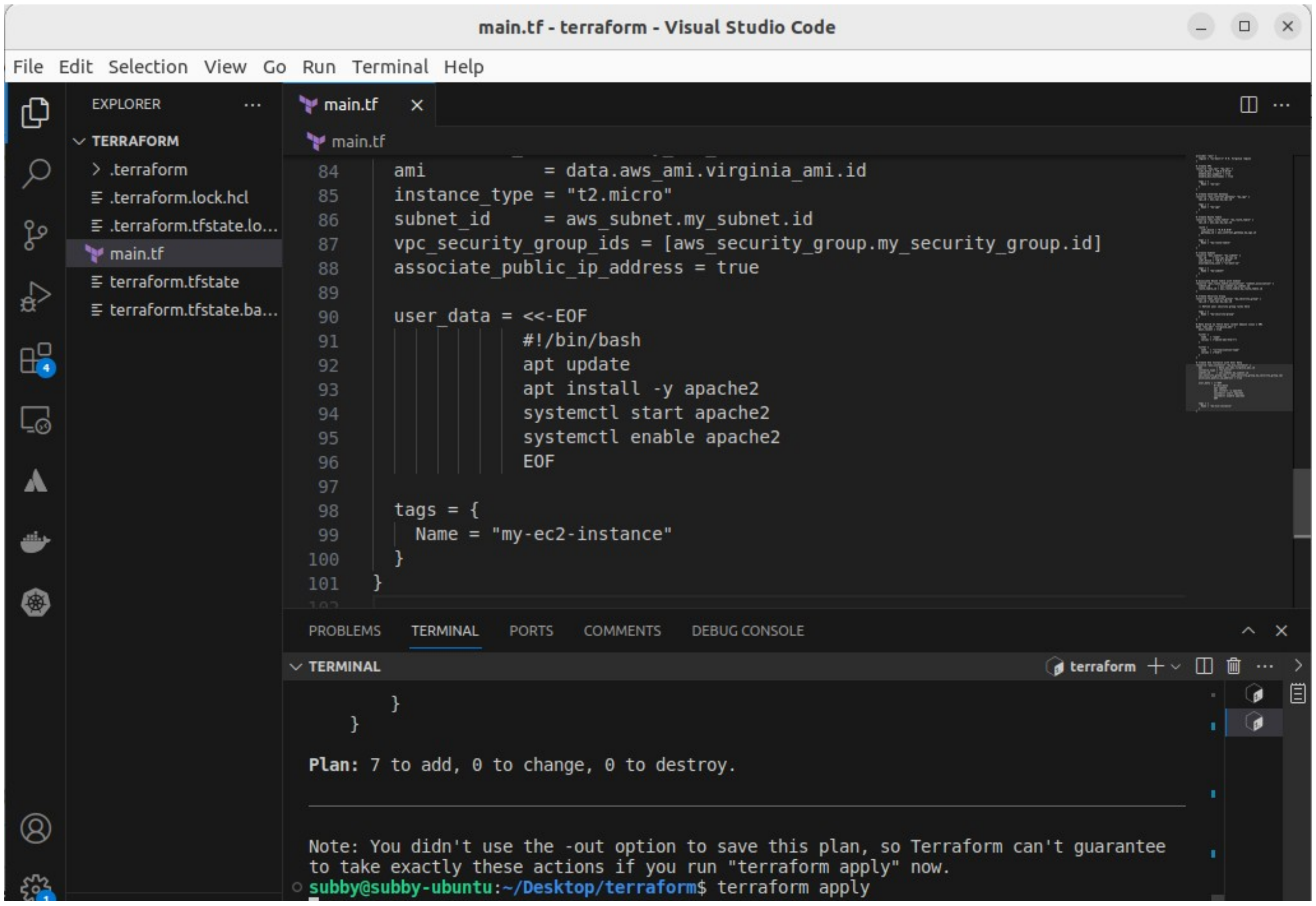
```
#!/bin/bash
apt update
apt install -y apache2
systemctl start apache2
systemctl enable apache2
```

No: 3

```
provider "aws" {
  region = "us-east-1" # N. Virginia region
}

# Create VPC
resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true

  tags = {
    Name = "my-vpc"
  }
}

# Create Internet Gateway
```

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

  tags = {
    Name = "my-igw"
  }
}

# Create Route Table
resource "aws_route_table" "my_route_table" {
  vpc_id = aws_vpc.my_vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.my_igw.id
  }

  tags = {
    Name = "my-route-table"
  }
}

# Create Subnet
resource "aws_subnet" "my_subnet" {
  vpc_id     = aws_vpc.my_vpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "us-east-1a"

  tags = {
    Name = "my-subnet"
  }
}

# Associate Route Table with Subnet
resource "aws_route_table_association" "subnet_association" {
  subnet_id      = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.my_route_table.id
}

# Create Security Group
resource "aws_security_group" "my_security_group" {
  vpc_id = aws_vpc.my_vpc.id

  // Define your security group rules here

  tags = {
    Name = "my-security-group"
  }
}
```

```
# Data block to fetch most recent Amazon Linux 2 AMI
data "aws_ami" "virginia_ami" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

# Create EC2 Instance with User Data
resource "aws_instance" "my_ec2_instance" {
  ami           = data.aws_ami.virginia_ami.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.my_subnet.id
  vpc_security_group_ids = [aws_security_group.my_security_group.id]
  associate_public_ip_address = true

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

  tags = {
    Name = "my-ec2-instance"
  }
}
```



```
  ami           = data.aws_ami.virginia_ami.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.my_subnet.id
  vpc_security_group_ids = [aws_security_group.my_security_group.id]
  associate_public_ip_address = true

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

  tags = {
    Name = "my-ec2-instance"
  }
}
```

```
        }
    }

Plan: 7 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee
to take exactly these actions if you run "terraform apply" now.
subby@subby-ubuntu:~/Desktop/terraform$ terraform apply
```

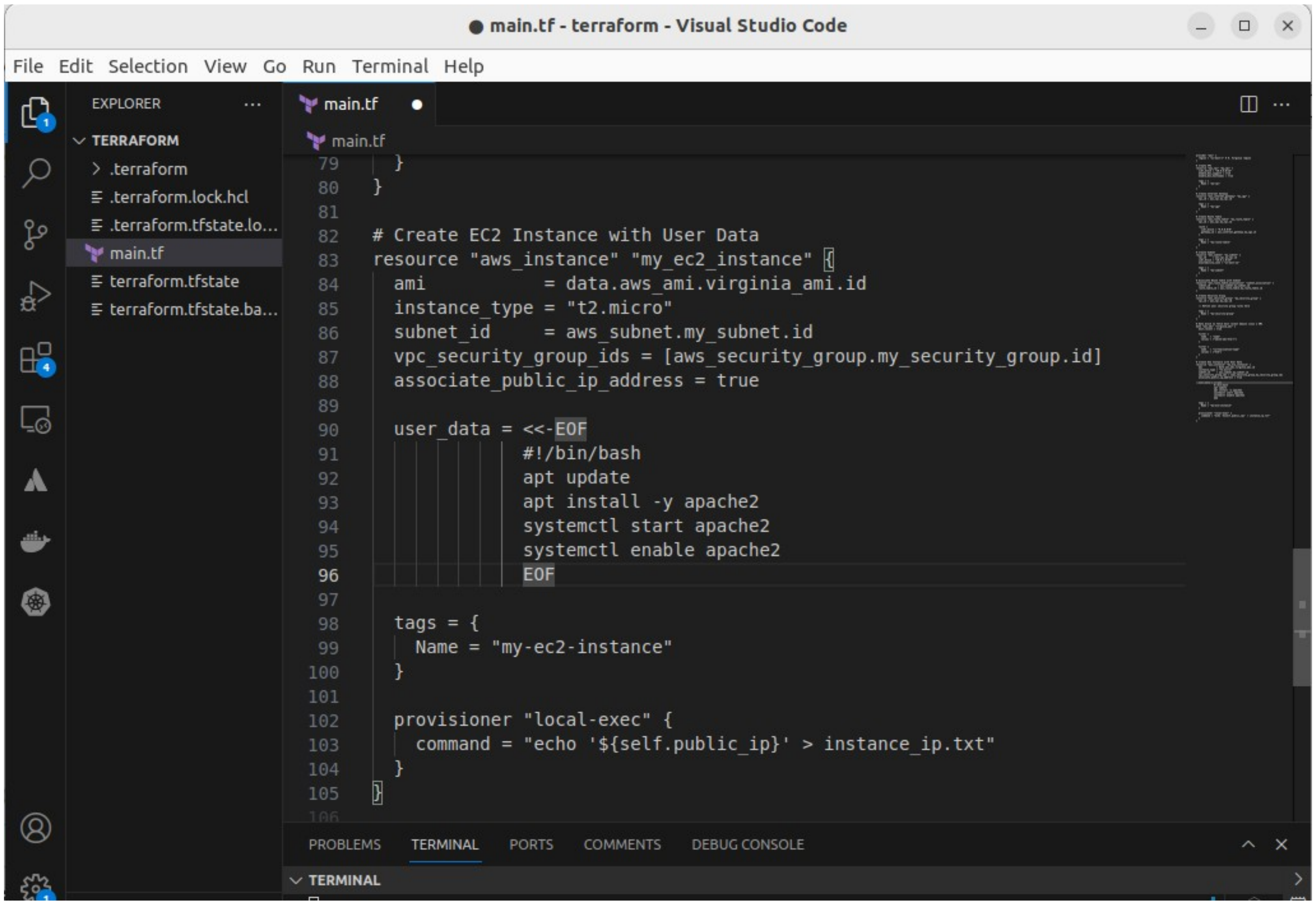
No: 4

Print the IP address of the instance in a file on the local once deployed:

```
  provisioner "local-exec" {
    command = "echo '${self.public_ip}' > instance_ip.txt"
  }
```

```
provider "aws" {
  region = "us-east-1" # N. Virginia region
}

# Create VPC
resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true

  tags = {
    Name = "my-vpc"
  }
}

# Create Internet Gateway
resource "aws_internet_gateway" "my_igw" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "my-igw"
  }
}

# Create Route Table
resource "aws_route_table" "my_route_table" {
  vpc_id = aws_vpc.my_vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.my_igw.id
  }

  tags = {
    Name = "my-route-table"
  }
```

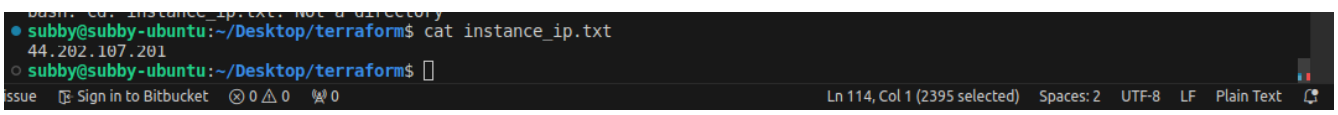
```
}

# Create Subnet
resource "aws_subnet" "my_subnet" {
  vpc_id     = aws_vpc.my_vpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "us-east-1a"

  tags = {
    Name = "my-subnet"
  }
}

# Associate Route Table with Subnet
resource "aws_route_table_association" "subnet_association" {
  subnet_id      = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.my_route_table.id
}

# Create Security Group
resource "aws_security_group" "my_security_group" {
  vpc_id = aws_vpc.my_vpc.id

  // Define your security group rules here

  tags = {
    Name = "my-security-group"
  }
}

# Data block to fetch most recent Amazon Linux 2 AMI
data "aws_ami" "virginia_ami" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

# Create EC2 Instance with User Data
resource "aws_instance" "my_ec2_instance" {
  ami           = data.aws_ami.virginia_ami.id
  instance_type = "t2.micro"
  subnet_id      = aws_subnet.my_subnet.id
  vpc_security_group_ids = [aws_security_group.my_security_group.id]
  associate_public_ip_address = true

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

  tags = {
    Name = "my-ec2-instance"
  }

  provisioner "local-exec" {
    command = "echo '${self.public_ip}' > instance_ip.txt"
  }
}
```



```
  }
}

# Create EC2 Instance with User Data
resource "aws_instance" "my_ec2_instance" {
  ami           = data.aws_ami.virginia_ami.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.my_subnet.id
  vpc_security_group_ids = [aws_security_group.my_security_group.id]
  associate_public_ip_address = true

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

  tags = {
    Name = "my-ec2-instance"
  }

  provisioner "local-exec" {
    command = "echo '${self.public_ip}' > instance_ip.txt"
  }
}
```

```
provider "aws" {
  region = "us-east-1" # N. Virginia region
}

# Create VPC
resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true

  tags = {
    Name = "my-vpc"
  }
}

# Create Internet Gateway
resource "aws_internet_gateway" "my_igw" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "my-igw"
  }
}

# Create Route Table
resource "aws_route_table" "my_route_table" {
  vpc_id = aws_vpc.my_vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.my_igw.id
  }

  tags = {
    Name = "my-route-table"
  }
}

# Create Subnet
resource "aws_subnet" "my_subnet" {
  vpc_id     = aws_vpc.my_vpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "us-east-1a"

  tags = {
    Name = "my-subnet"
  }
}
```

```
# Associate Route Table with Subnet
resource "aws_route_table_association" "subnet_association" {
  subnet_id      = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.my_route_table.id
}

# Create Security Group
resource "aws_security_group" "my_security_group" {
  vpc_id = aws_vpc.my_vpc.id

  // Define your security group rules here
  // Allow inbound traffic from everywhere (0.0.0.0/0) on all ports and protocols
  ingress {
    from_port   = 0  # All ports
    to_port     = 0  # All ports
    protocol    = "-1"  # All protocols
    cidr_blocks = ["0.0.0.0/0"]  # Allow from anywhere
  }

  tags = {
    Name = "my-security-group"
  }
}

# Data block to fetch most recent Amazon Linux 2 AMI
data "aws_ami" "virginia_ami" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

# Create EC2 Instance with User Data
resource "aws_instance" "my_ec2_instance" {
  ami           = data.aws_ami.virginia_ami.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.my_subnet.id
  associate_public_ip_address = true
  security_groups = [aws_security_group.my_security_group.id] # Use the created security group
  user_data = <<-EOF
              #!/bin/bash
              sudo yum update -y
              sudo yum install -y httpd
              sudo systemctl start httpd
              sudo systemctl enable httpd
              EOF

  tags = {
    Name = "my-ec2-instance"
  }

  provisioner "local-exec" {
    command = "echo '${self.public_ip}' > instance_ip.txt"
  }
}
```



```
bash: cd: instance_ip.txt: Not a directory
subby@subby-ubuntu:~/Desktop/terraform$ cat instance_ip.txt
44.202.107.201
subby@subby-ubuntu:~/Desktop/terraform$
```