**1) Create VPC**

terraform {

  required\_providers {

    aws = {

        source = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

variable "aws\_vpc" {

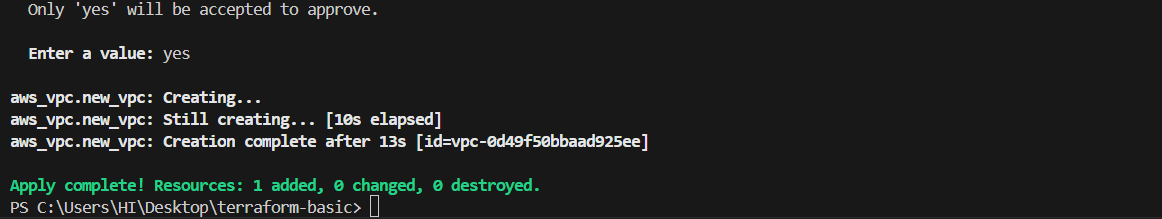
  default = "192.168.0.0/16"

}

variable "aws\_region" {

  default = "us-west-1"

}





1. Create Internet gateway

terraform {

  required\_providers {

    aws = {

        source = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

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

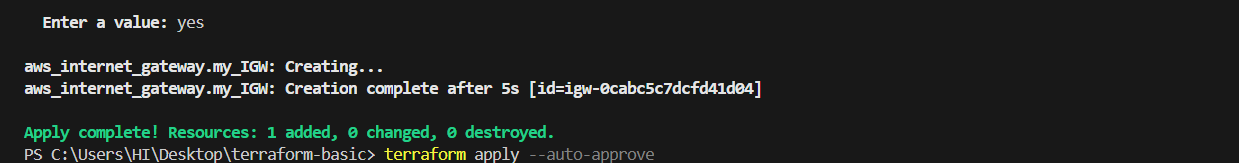
  vpc\_id = aws\_vpc.new\_vpc.id

  tags = {

    Name = "MY-IGW"

  }

}





1. Create Custom Route Table

terraform {

  required\_providers {

    aws = {

        source = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

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

  vpc\_id = aws\_vpc.new\_vpc.id

  tags = {

    Name = "MY-IGW"

  }

}

resource "aws\_route\_table" "new\_route\_table" {

  vpc\_id = aws\_vpc.new\_vpc.id

}

resource "aws\_route" "main" {

  route\_table\_id = aws\_route\_table.new\_route\_table.id

  destination\_cidr\_block = var.destination\_cidr\_block

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

}

variable "aws\_vpc" {

  default = "192.168.0.0/16"

}

variable "aws\_region" {

  default = "us-west-1"

}

variable "route\_table\_id" {

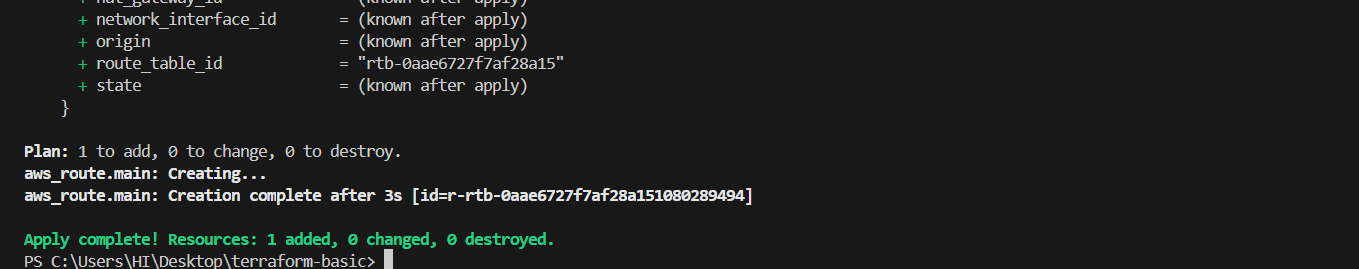
  default = "192.168.0.0/16"

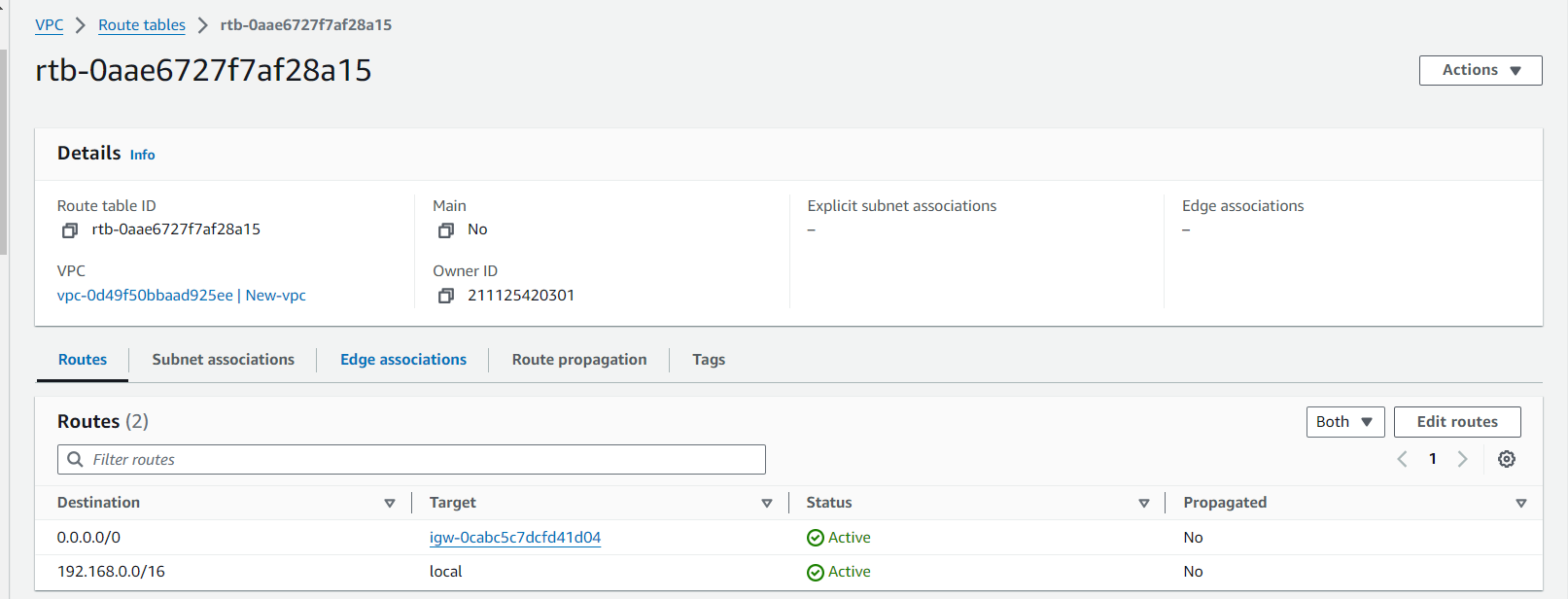
}

variable "destination\_cidr\_block" {

  default = "0.0.0.0/0"

}





1. **Create Subnet**

terraform {

  required\_providers {

    aws = {

        source = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

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

  vpc\_id = aws\_vpc.new\_vpc.id

  tags = {

    Name = "MY-IGW"

  }

}

resource "aws\_route\_table" "new\_route\_table" {

  vpc\_id = aws\_vpc.new\_vpc.id

}

resource "aws\_route" "main" {

  route\_table\_id = aws\_route\_table.new\_route\_table.id

  destination\_cidr\_block = var.destination\_cidr\_block

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

}

resource "aws\_subnet" "first\_subnet" {

  vpc\_id = aws\_vpc.new\_vpc.id

  cidr\_block = var.cidr\_block

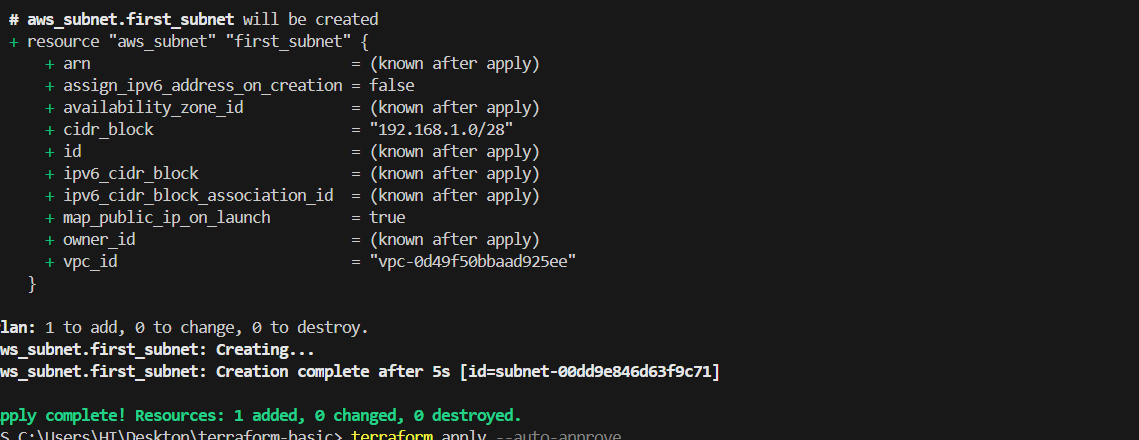
  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "pub-subnet"

  }

}



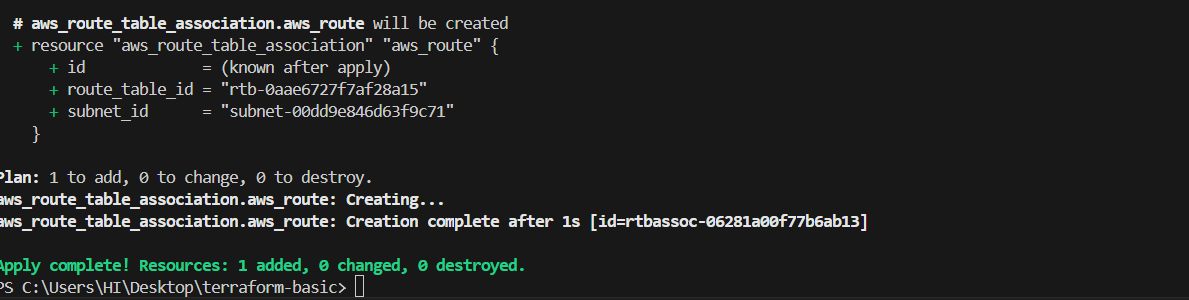
1. Associate subnet with Route Table

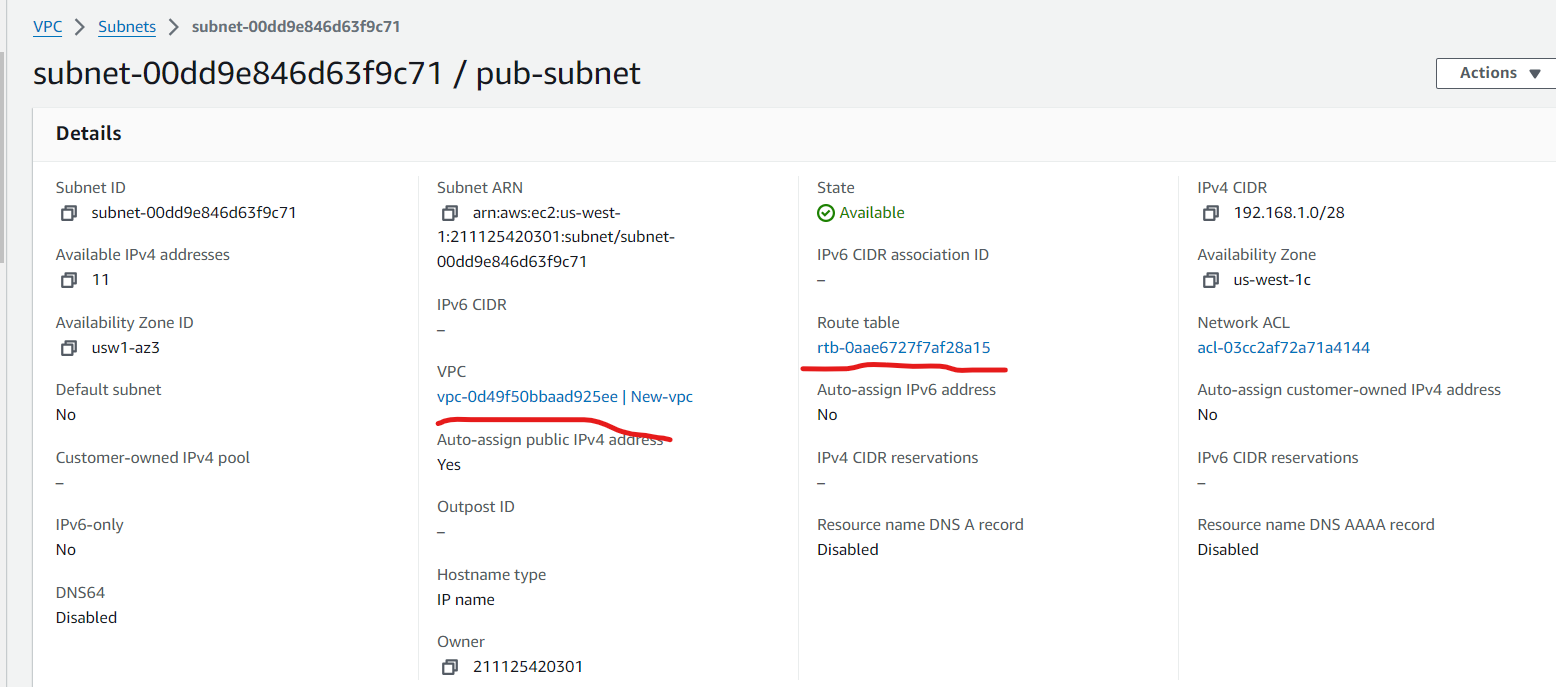
resource "aws\_route\_table\_association" "aws\_route" {

  subnet\_id = aws\_subnet.first\_subnet.id

  route\_table\_id = aws\_route\_table.new\_route\_table.id

}





1. **Create Security Group to allow port 22.80,443**

terraform {

  required\_providers {

    aws = {

        source = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

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

  vpc\_id = aws\_vpc.new\_vpc.id

  tags = {

    Name = "MY-IGW"

  }

}

resource "aws\_route\_table" "new\_route\_table" {

  vpc\_id = aws\_vpc.new\_vpc.id

}

resource "aws\_route" "main" {

  route\_table\_id = aws\_route\_table.new\_route\_table.id

  destination\_cidr\_block = var.destination\_cidr\_block

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

}

resource "aws\_subnet" "first\_subnet" {

  vpc\_id = aws\_vpc.new\_vpc.id

  cidr\_block = var.cidr\_block

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "pub-subnet"

  }

}

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

  subnet\_id = aws\_subnet.first\_subnet.id

  route\_table\_id = aws\_route\_table.new\_route\_table.id

}

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

  name = "new-security"

  vpc\_id = aws\_vpc.new\_vpc.id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 443

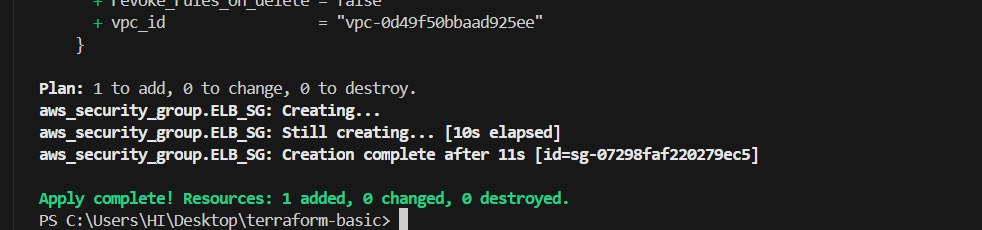
    to\_port     = 443

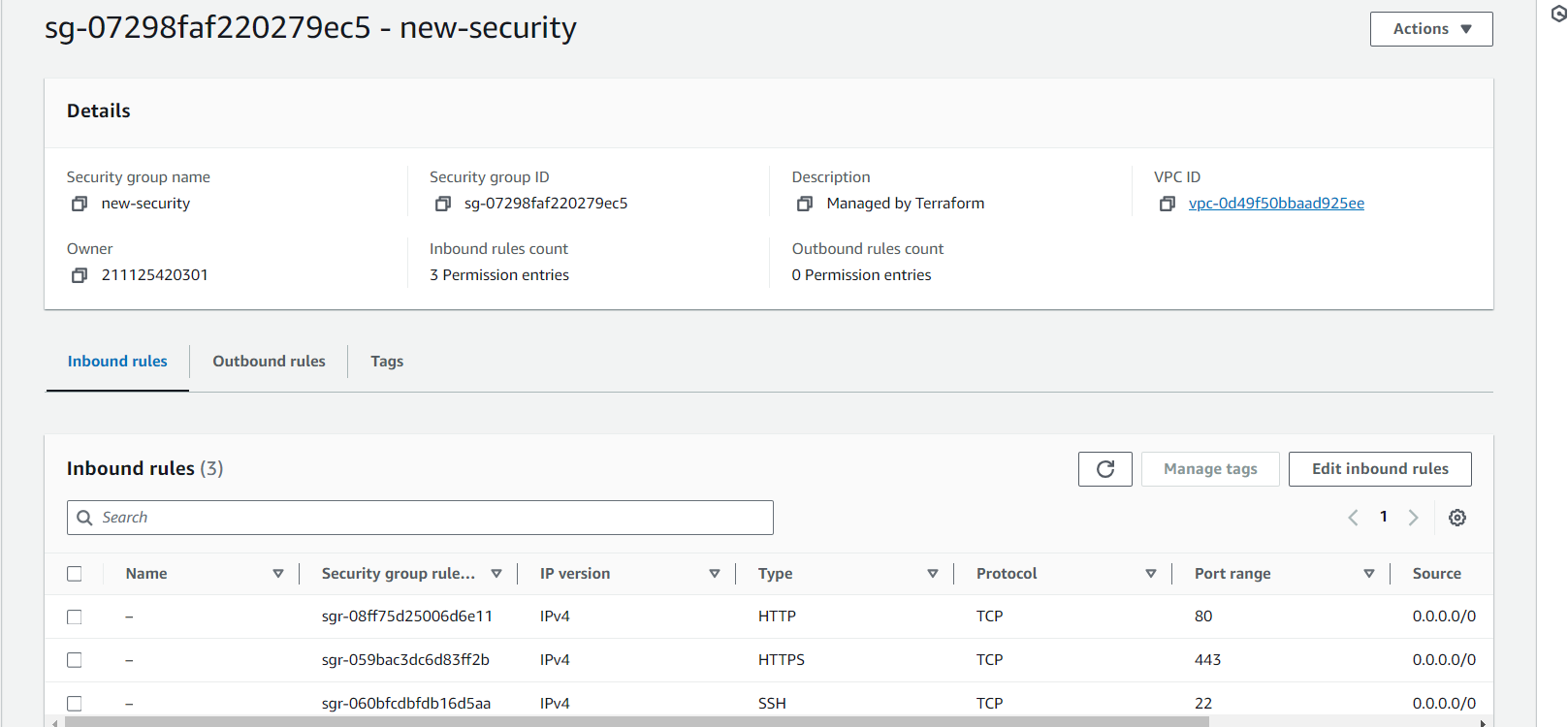
    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

}





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

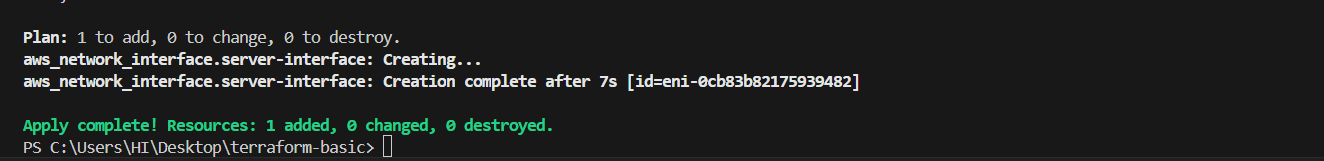
resource "aws\_network\_interface" "server-interface" {

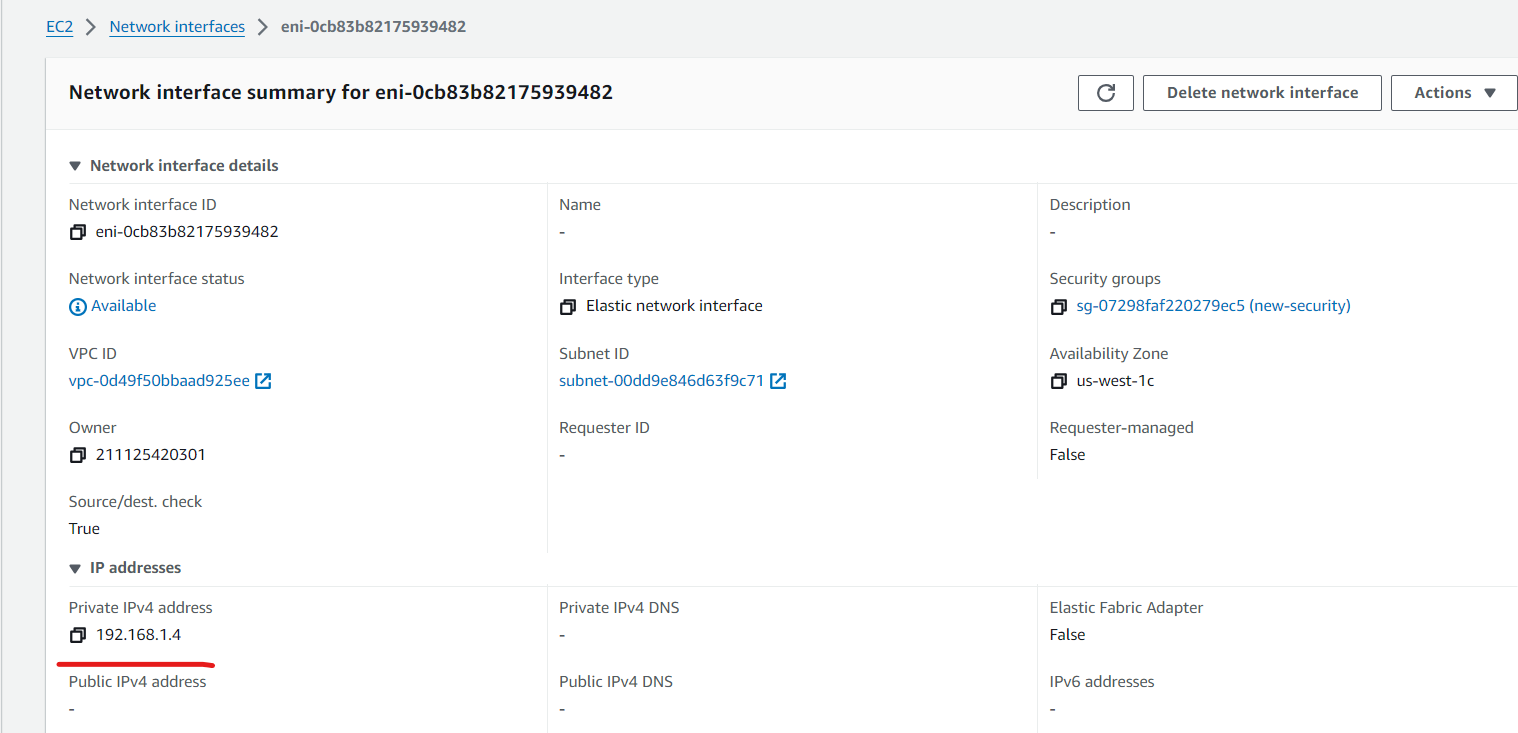
  subnet\_id = aws\_subnet.first\_subnet.id

  private\_ips = ["192.168.1.4"]

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

}





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

terraform {

  required\_providers {

    aws = {

        source = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

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

  vpc\_id = aws\_vpc.new\_vpc.id

  tags = {

    Name = "MY-IGW"

  }

}

resource "aws\_route\_table" "new\_route\_table" {

  vpc\_id = aws\_vpc.new\_vpc.id

}

resource "aws\_route" "main" {

  route\_table\_id = aws\_route\_table.new\_route\_table.id

  destination\_cidr\_block = var.destination\_cidr\_block

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

}

resource "aws\_subnet" "first\_subnet" {

  vpc\_id = aws\_vpc.new\_vpc.id

  cidr\_block = var.cidr\_block

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "pub-subnet"

  }

}

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

  subnet\_id = aws\_subnet.first\_subnet.id

  route\_table\_id = aws\_route\_table.new\_route\_table.id

}

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

  name = "new-security"

  vpc\_id = aws\_vpc.new\_vpc.id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 443

    to\_port     = 443

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

resource "aws\_network\_interface" "server-interface" {

  subnet\_id = aws\_subnet.first\_subnet.id

  private\_ips = ["192.168.1.4"]

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

}

resource "aws\_eip" "elastic-ip" {

  vpc = true

  network\_interface = aws\_network\_interface.server-interface.id

  associate\_with\_private\_ip = "192.168.1.4"

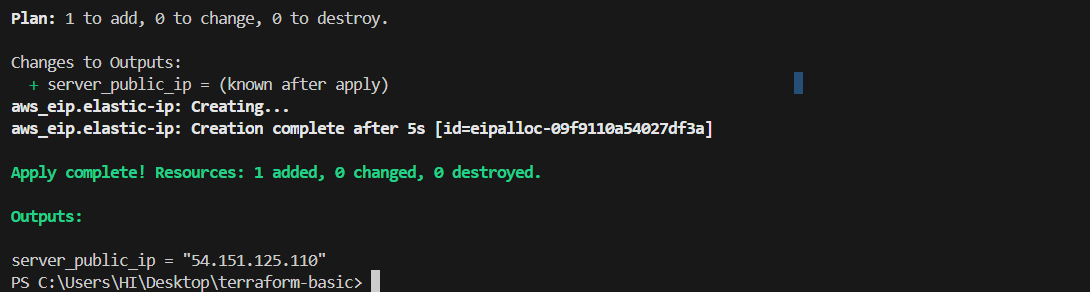
  depends\_on = [ aws\_internet\_gateway.my\_IGW ]

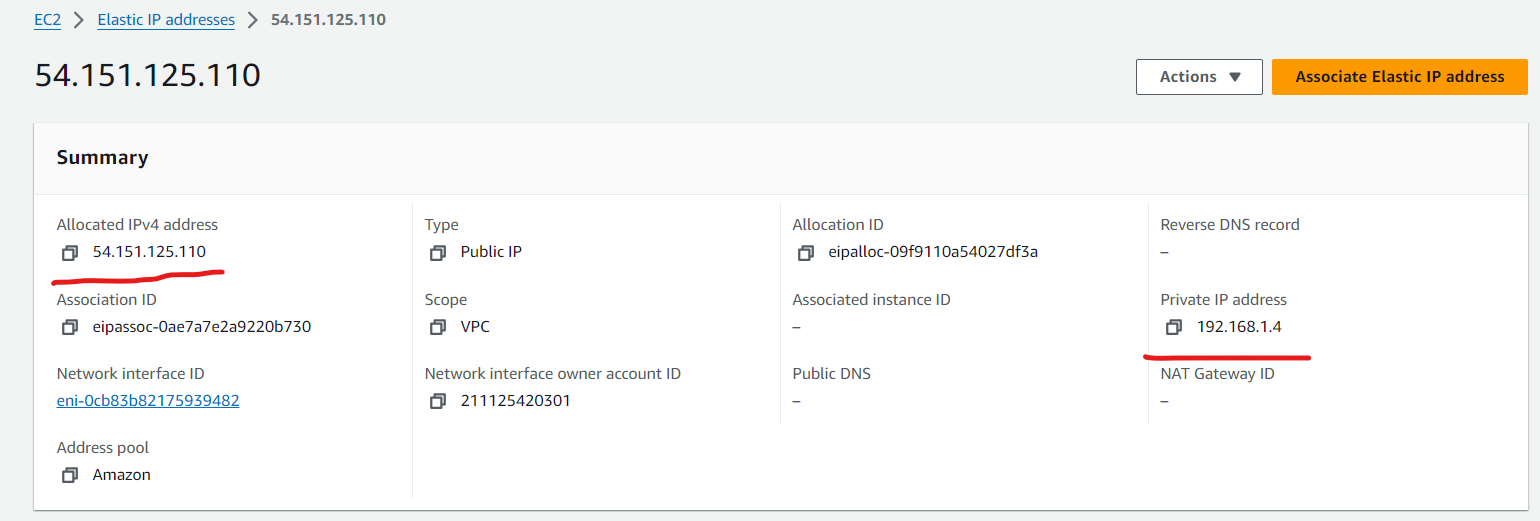
}

output "server\_public\_ip" {

  value = aws\_eip.elastic-ip.public\_ip

}





1. **Create Ubuntu server and install/enable apache2**

terraform {

  required\_providers {

    aws = {

        source = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

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

  vpc\_id = aws\_vpc.new\_vpc.id

  tags = {

    Name = "MY-IGW"

  }

}

resource "aws\_route\_table" "new\_route\_table" {

  vpc\_id = aws\_vpc.new\_vpc.id

}

resource "aws\_route" "main" {

  route\_table\_id = aws\_route\_table.new\_route\_table.id

  destination\_cidr\_block = var.destination\_cidr\_block

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

}

resource "aws\_subnet" "first\_subnet" {

  vpc\_id = aws\_vpc.new\_vpc.id

  cidr\_block = var.cidr\_block

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "pub-subnet"

  }

}

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

  subnet\_id = aws\_subnet.first\_subnet.id

  route\_table\_id = aws\_route\_table.new\_route\_table.id

}

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

  name = "new-security"

  vpc\_id = aws\_vpc.new\_vpc.id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 443

    to\_port     = 443

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

resource "aws\_network\_interface" "server-interface" {

  subnet\_id = aws\_subnet.first\_subnet.id

  private\_ips = ["192.168.1.4"]

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

}

resource "aws\_eip" "elastic-ip" {

  vpc = true

  network\_interface = aws\_network\_interface.server-interface.id

  associate\_with\_private\_ip = "192.168.1.4"

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

}

output "server\_public\_ip" {

  value = aws\_eip.elastic-ip.public\_ip

}

resource "aws\_instance" "own-server" {

  ami = var.ami

  instance\_type = "t2.micro"

  key\_name = "america"

  availability\_zone = "us-west-1c"

  user\_data = <<-EOF

  #!/bin/bash

 sudo apt-get install apache2 -y",

      "sudo systemctl start apache2"

  EOF

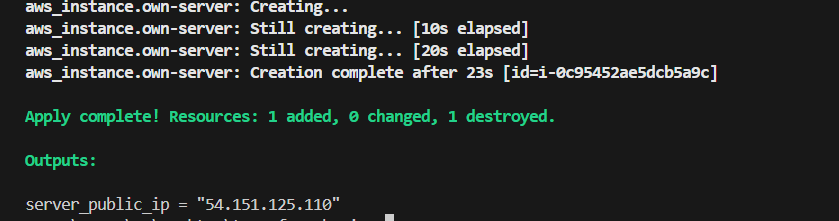
network\_interface {

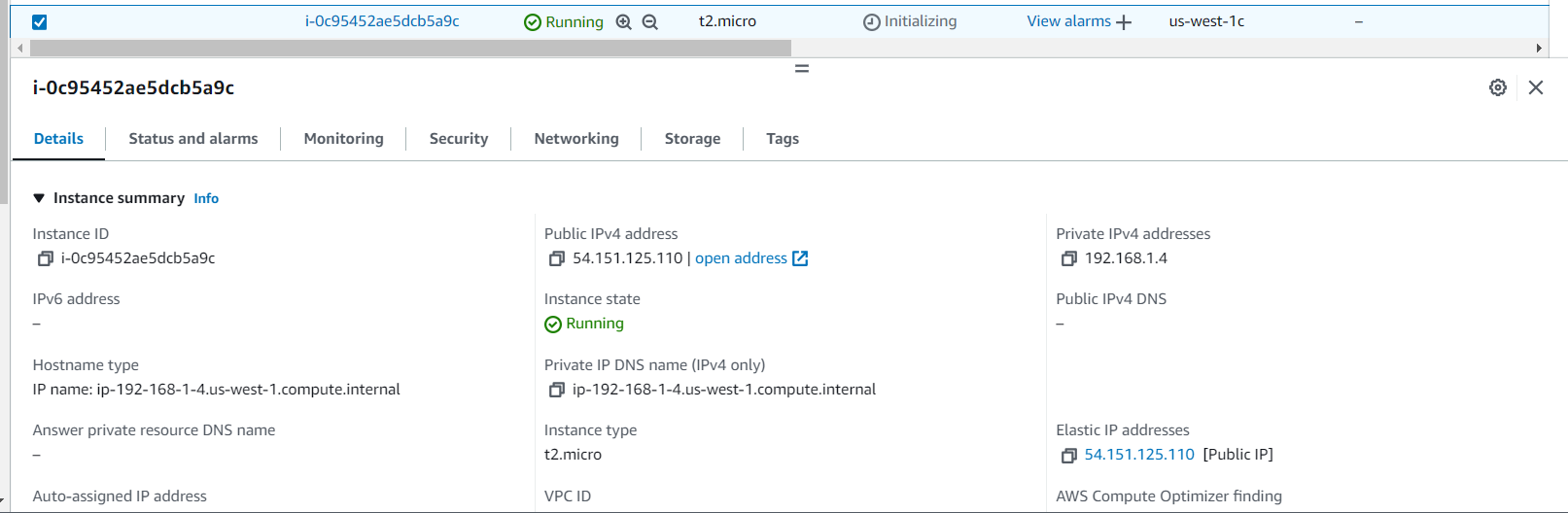
  device\_index = 0

  network\_interface\_id = aws\_network\_interface.server-interface.id

}

}





terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

        version = "~>2.70"

    }

  }

}

provider "aws" {

  region = var.aws\_region

}

resource "aws\_vpc" "new\_vpc" {

  cidr\_block = var.aws\_vpc

tags = {

  Name = "New-vpc"

}

}

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

  vpc\_id = aws\_vpc.new\_vpc.id

  tags = {

    Name = "MY-IGW"

  }

}

resource "aws\_route\_table" "new\_route\_table" {

  vpc\_id = aws\_vpc.new\_vpc.id

}

resource "aws\_route" "main" {

  route\_table\_id = aws\_route\_table.new\_route\_table.id

  destination\_cidr\_block = var.destination\_cidr\_block

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

}

resource "aws\_subnet" "first\_subnet" {

  vpc\_id = aws\_vpc.new\_vpc.id

  cidr\_block = var.cidr\_block

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "pub-subnet"

  }

}

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

  subnet\_id = aws\_subnet.first\_subnet.id

  route\_table\_id = aws\_route\_table.new\_route\_table.id

}

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

  name = "new-security"

  vpc\_id = aws\_vpc.new\_vpc.id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 443

    to\_port     = 443

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

resource "aws\_network\_interface" "server-interface" {

  subnet\_id = aws\_subnet.first\_subnet.id

  private\_ips = ["192.168.1.4"]

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

}

resource "aws\_eip" "elastic-ip" {

  vpc = true

  network\_interface = aws\_network\_interface.server-interface.id

  associate\_with\_private\_ip = "192.168.1.4"

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

}

output "server\_public\_ip" {

  value = aws\_eip.elastic-ip.public\_ip

}

resource "aws\_instance" "own-server" {

  ami = var.ami

  instance\_type = "t2.micro"

  key\_name = "america"

  availability\_zone = "us-west-1c"

  user\_data = <<-EOF

  #!/bin/bash

  sudo apt update

sudo apt install apache2 -y

sudo systemctl status apache2

sudo systemctl start apache2

sudo systemctl enable apache2

  EOF

network\_interface {

  device\_index = 0

  network\_interface\_id = aws\_network\_interface.server-interface.id

}

}

resource "aws\_dynamodb\_table" "dynamodb-route\_table\_id" {

  name = "dbdynamo-terraform-state"

  hash\_key = "LockID"

  read\_capacity = 20

  write\_capacity = 20

  attribute {

    name = "LockID"

    type = "S"

  }

}

terraform {

  backend "s3" {

    bucket = "techie050936"

    dynamodb\_table = "dbdynamo-terraform-state"

    key = "terraform.tfstate"

    region = "us-west-1"

  }

}

