Building AWS VPC with EC2 Using Terraform: A Step-by-Step Guide

Terraform simplifies infrastructure setup with code. This document guides you through the step-by-step process of creating an AWS VPC and launching an EC2 instance, including security configurations, ensuring scalability and efficiency. 🚀

→

Introduction: Outline the purpose of the Terraform script, which is to create an AWS VPC, subnet, security group, and deploy an EC2 instance with Internet connectivity.

Provider.tf

The **provider.tf** file in Terraform is crucial because:

It defines the **cloud provider** (e.g., AWS, Azure, GCP) for managing resources.

It tells Terraform where to interact, like which API or service to use.

It includes configurations like regions or authentication details for the provider.

Without it, Terraform won't know how to connect to the provider's resources.

It's the foundation for any Terraform project to deploy resources.

EC2 & VPC Creation

```
count
ec2
variables
VPC_WITH_EC2
.terraform
.terraform.lock.... U
.terraform.tfsta... U
provider.tf U
terraform.tfstate U
terraform.tfstat... U
VPC_with_EC2.tf U
```

```terraform
# Create instance
resource "aws_instance" "Terraform" {
  ami           = "ami-06b21ccaeff8cd686"
  instance_type = "t2.micro"
  subnet_id = aws_subnet.demo_subnet.id
  vpc_security_group_ids = [aws_security_group.demo-vpc-sg.id]

 tags = {
    Name = "Terrafom-ec2"
  }

}

# Create VPC

resource "aws_vpc" "demo-vpc" {
  cidr_block = "10.10.0.0/16"

  tags = {
    Name = "demo-vpc"
  }

}

# Create subnet

resource "aws_subnet" "demo_subnet" {
  vpc_id     = aws_vpc.demo-vpc.id
  cidr_block = "10.10.1.0/24"

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

NOTE: We will define the subnets and security groups ID`S in ec2 resources after creating them

Define the EC2 Resource:

- Use the aws_instance resource and name it "Terraform".

Specify the AMI:

- Set the Amazon Machine Image to "ami-06b21caeff8cd686"

Choose the Instance Type

- Specify the instance type as "t2.micro" for a free-tier eligible instance.

Link the Subnet:

- Associate the EC2 instance with the created subnet using subnet_id = aws_subnet.demo_subnet.id.

Attach the Security Group:

- Add the security group using vpc_security_group_ids = [aws_security_group.demo-vpc-sg.id].

Create a VPC:

- Define the aws_vpc resource with a CIDR block of 10.10.0.0/16
- Add a tag Name: demo-vpc for identification.

Create a Subnet:

- Use the aws_subnet resource, linking it to the created VPC (vpc_id)
- Assign a CIDR block of 10.10.1.0/24 and tag it as demo-subnet

Create an Internet Gateway:

- Define the aws_internet_gateway resource and associate it with the VPC.

Create a Route Table:

- Configure the aws_route_table resource, associating it with the VPC and adding a default route to enable Internet access via the Internet Gateway

```
resource "aws_internet_gateway" "demo-igw" {
  vpc_id = aws_vpc.demo-vpc.id

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

#Create Route Table

resource "aws_route_table" "demo-rt" {
  vpc_id = aws_vpc.demo-vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.demo-igw.id
  }

  tags = {
    Name = "demo-rt"
  }
}

# Create Subnet association

resource "aws_route_table_association" "demo-rt-association" {
  subnet_id      = aws_subnet.demo_subnet.id
  route_table_id = aws_route_table.demo-rt.id
}
```

Associate the Route Table:

- Link the route table to the subnet using the aws_route_table_association resource.

```
# Create Security group

resource "aws_security_group" "demo-vpc-sg" {
    name = "demo-vpc-sg"
    vpc_id = aws_vpc.demo-vpc.id

    ingress {
        from_port        = 22
        to_port          = 22
        protocol         = "tcp"
        cidr_blocks      = ["0.0.0.0/0"]
        ipv6_cidr_blocks = ["::/0"]
    }

     egress {
        from_port        = 0
        to_port          = 0
        protocol         = "-1"
        cidr_blocks      = ["0.0.0.0/0"]
        ipv6_cidr_blocks = ["::/0"]
    }


    tags = {
        Name = "demo-vpc-sg"
    }
}
```

Define the Security Group Resource:

- Use the aws_security_group resource with the name demo-vpc-sg

Associate with VPC:

- Link the security group to the created VPC using the vpc_id

Configure Ingress Rules:

- Allow SSH (port 22) access using the from_port and to_port values set to 22
- Use the tcp protocol and allow all incoming IPs by setting cidr_blocks = ["0.0.0.0/0"]
- Include IPv6 access by adding ipv6_cidr_blocks = ["::/0"]

Configure Egress Rules:

- Allow all outbound traffic by setting from_port and to_port to 0.
- Use the -1 protocol, which means all protocols are allowed
- Enable access to all IPs with cidr_blocks = ["0.0.0.0/0"] and ipv6_cidr_blocks = ["::/0"].

terraform init: Initialize the project.

```
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.72.1"...
- Installing hashicorp/aws v5.72.1...
- Installed hashicorp/aws v5.72.1 (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,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

chall@Uma_Mahi MINGW64 /c/devops/repos/Terraform/VPC_WITH_EC2 (main)
$ terraform validate
Success! The configuration is valid.
```

terraform plan: Review the infrastructure plan.

```
$ terraform plan

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_instance.Terraform will be created
  + resource "aws_instance" "Terraform" {
      + ami                                  = "ami-06b21ccaeff8cd686"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "Nikhil06"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
```

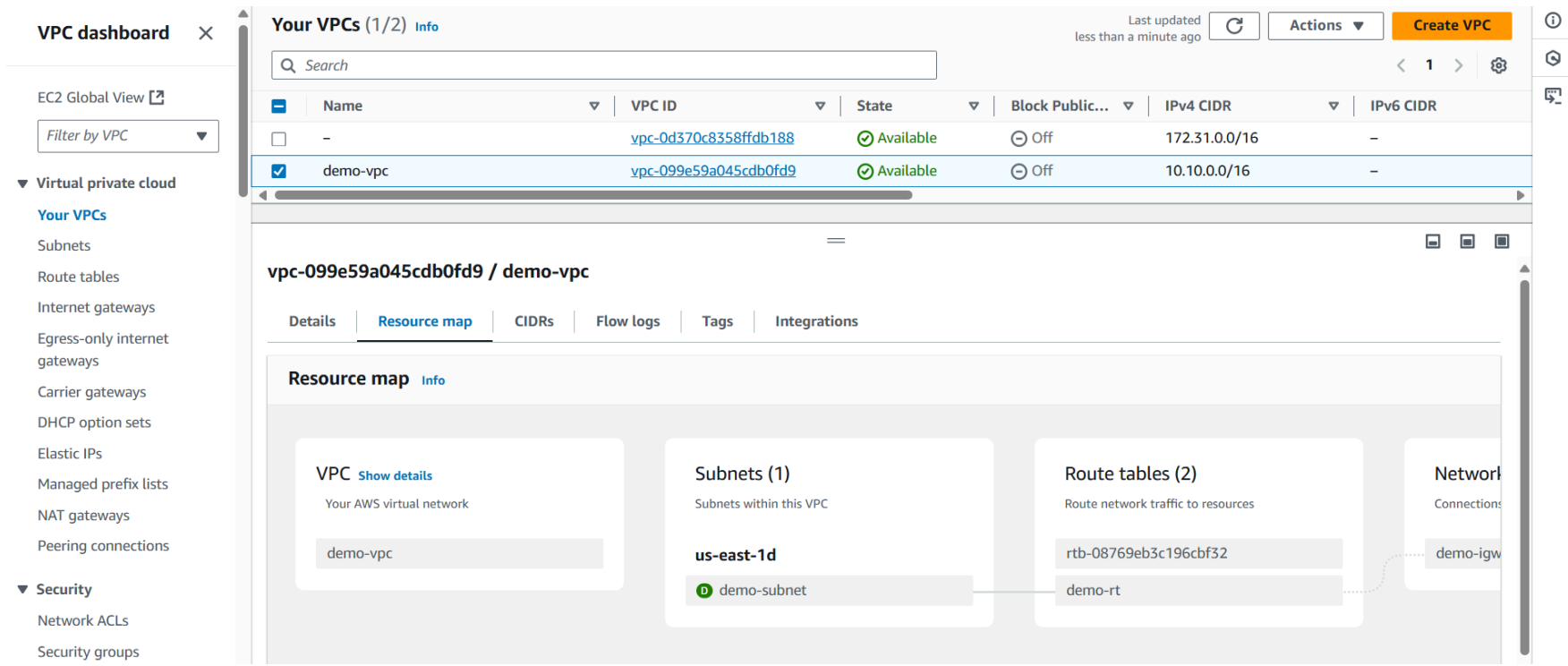
terraform apply: Deploy the resources.

```
Plan: 7 to add, 0 to change, 0 to destroy.
aws_vpc.demo-vpc: Creating...
aws_vpc.demo-vpc: Creation complete after 6s [id=vpc-099e59a045cdb0fd9]
aws_internet_gateway.demo-igw: Creating...
aws_subnet.demo_subnet: Creating...
aws_security_group.demo-vpc-sg: Creating...
aws_subnet.demo_subnet: Creation complete after 2s [id=subnet-0fffa9df6ec99b1a8]
aws_internet_gateway.demo-igw: Creation complete after 3s [id=igw-0137ff198ffa1307b]
aws_route_table.demo-rt: Creating...
aws_route_table.demo-rt: Creation complete after 3s [id=rtb-04ac53b1e75263474]
aws_route_table_association.demo-rt-association: Creating...
aws_security_group.demo-vpc-sg: Creation complete after 6s [id=sg-03d4edc724ac53c12]
aws_instance.Terraform: Creating...
aws_route_table_association.demo-rt-association: Creation complete after 1s [id=rtbassoc-087fec4fcf72f7837]
aws_instance.Terraform: Still creating... [10s elapsed]
aws_instance.Terraform: Creation complete after 17s [id=i-0588917f5649bdf2d]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
```

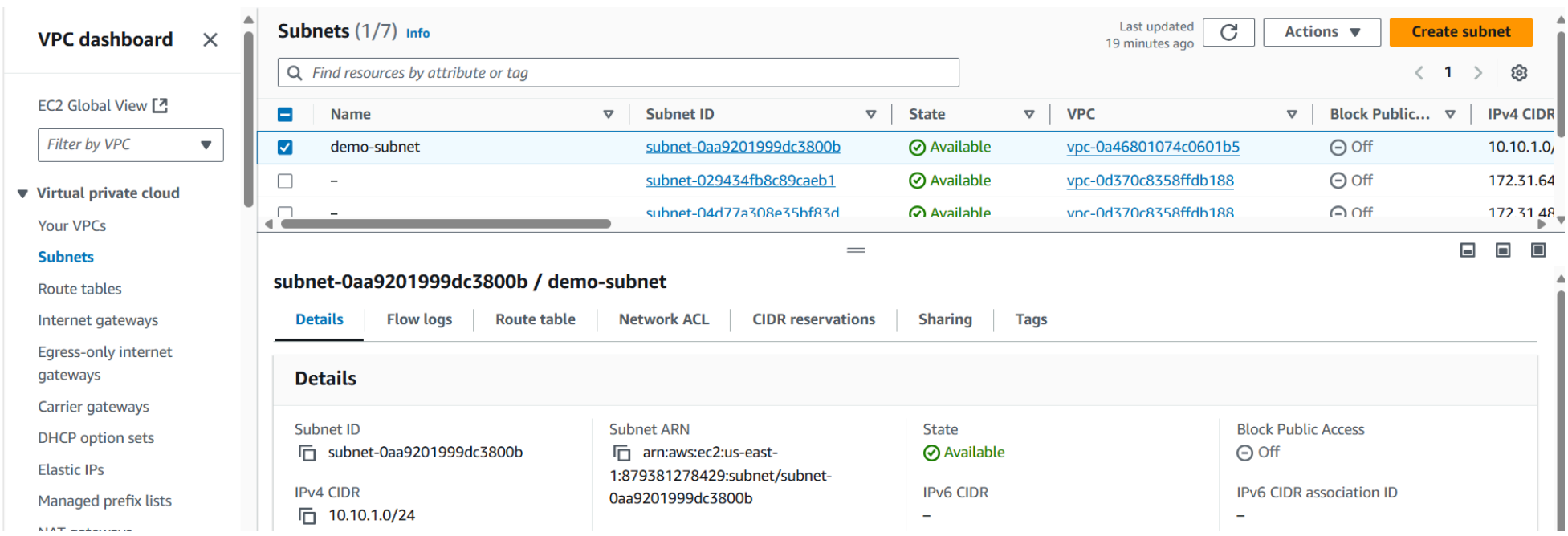
Verify the resources in the AWS Management Console.

VPC:

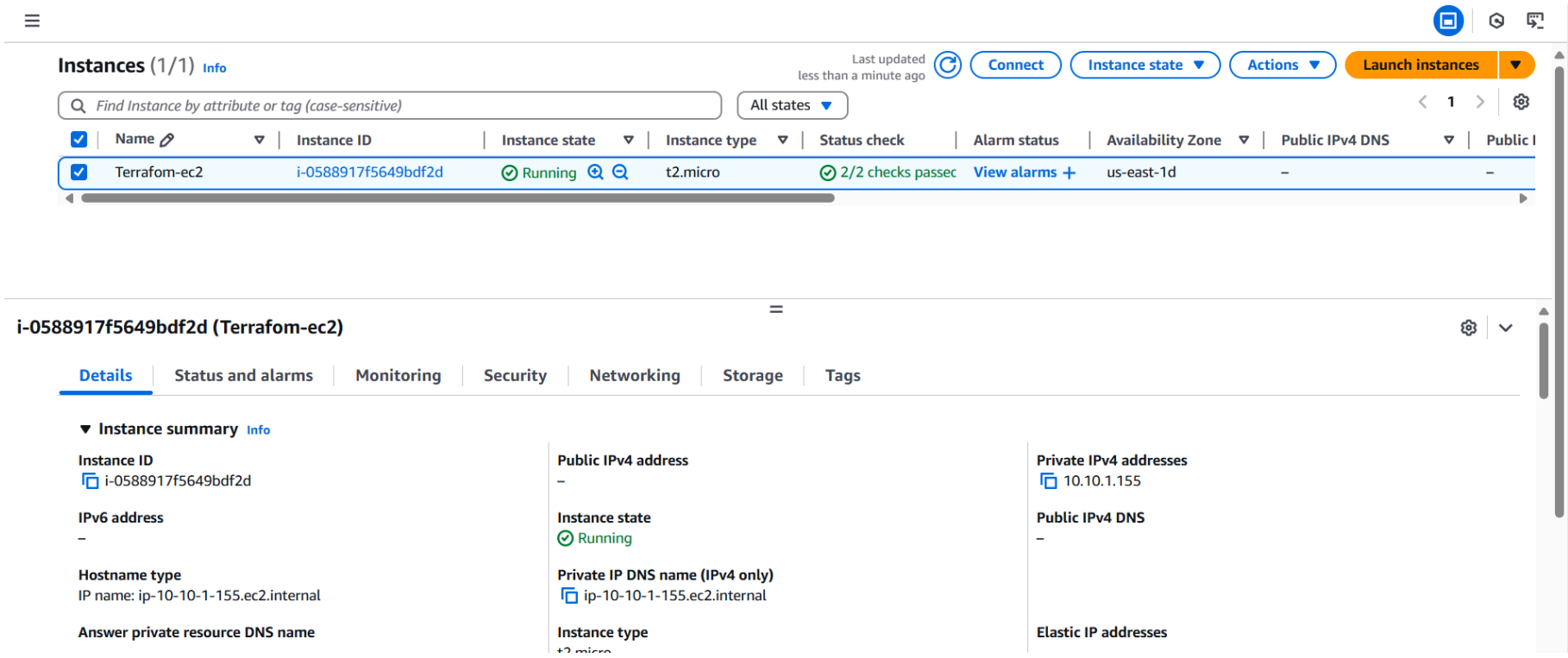


In the resource map you can observe all the resources created subnets route tables etc....

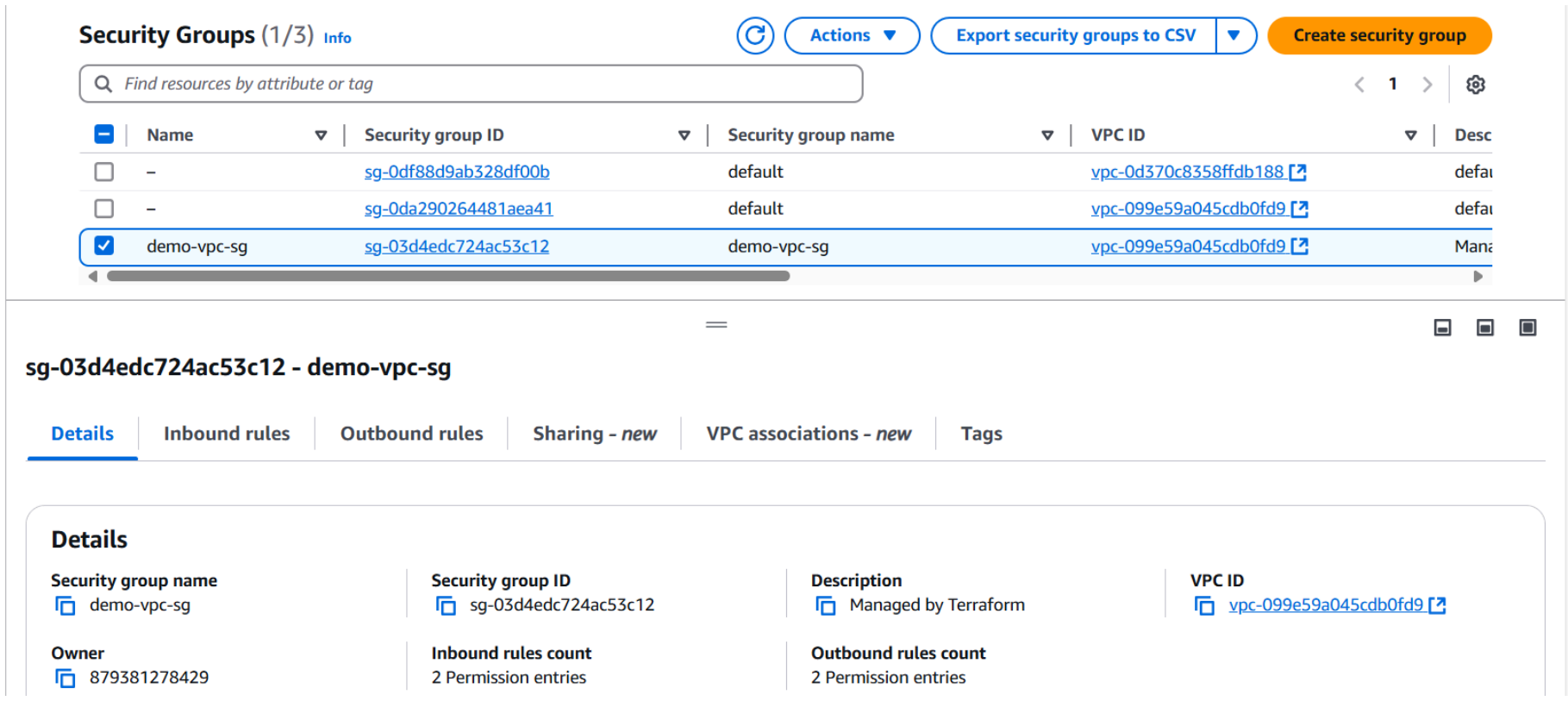
SUBNET:



EC2:



SECURITY GROUPS:



terraform destroy: Remove the resources

Follow for more **@nikhil-c-031444178**

Notes by **NIKHIL**