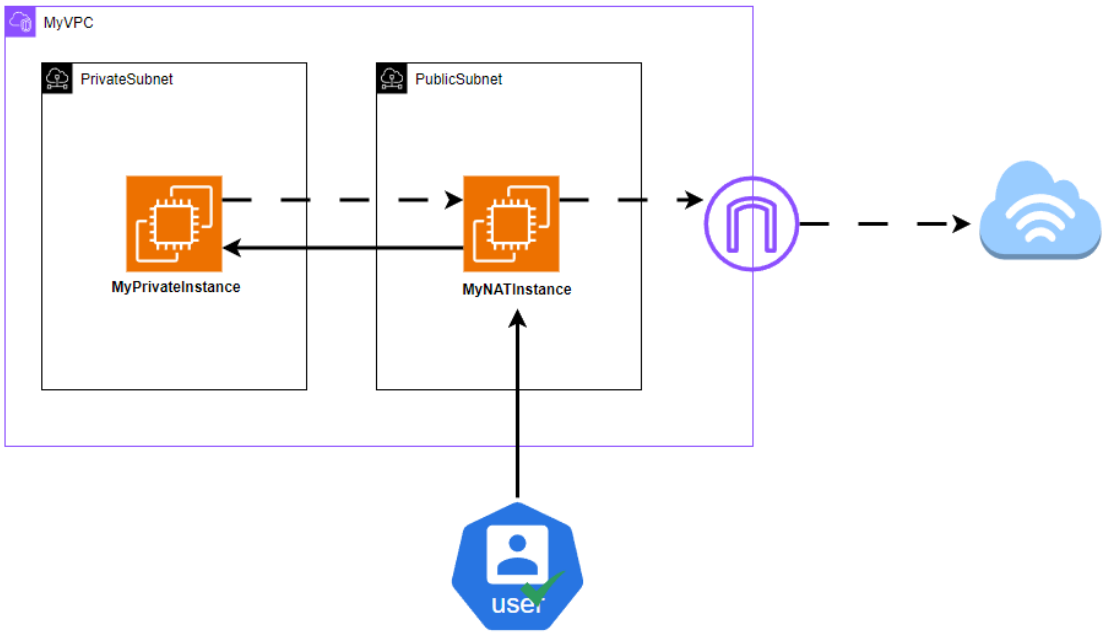
Guided Lab: Configuring a NAT Instance for Secure Connectivity

Description

In modern cloud architectures, ensuring secure and efficient connectivity for instances within private subnets is crucial. Network Address Translation (NAT) instances play a pivotal role in achieving this by allowing instances in private subnets to communicate with the internet for outbound traffic (e.g., software updates) without exposing them to inbound connections. This guided lab will demonstrate the setup and configuration of a NAT instance within an Amazon VPC, simulating a scenario where instances in a private subnet need to connect securely to a NAT instance.



Prerequisites

This labs assume you have basic knowledge of Amazon EC2 Instance, VPCs & basic network configuration in AWS Cloud.

If you find any gaps in your knowledge, consider taking the following lab:

- **Creating an Amazon EC2 instance (Linux)**
- **Creating a Custom Virtual Private Cloud (VPC) from scratch**

Objectives

By the end of this lab, participants will be able to:

- **Set up a VPC with public and private subnets.**
- **Launch and configure EC2 instances in both public and private subnets.**
- **Create and configure a NAT instance to enable secure communication for instances in the private subnet.**

- **Modify route tables to direct traffic appropriately.**
- **Establish a secure connection between a private instance and a NAT instance.**

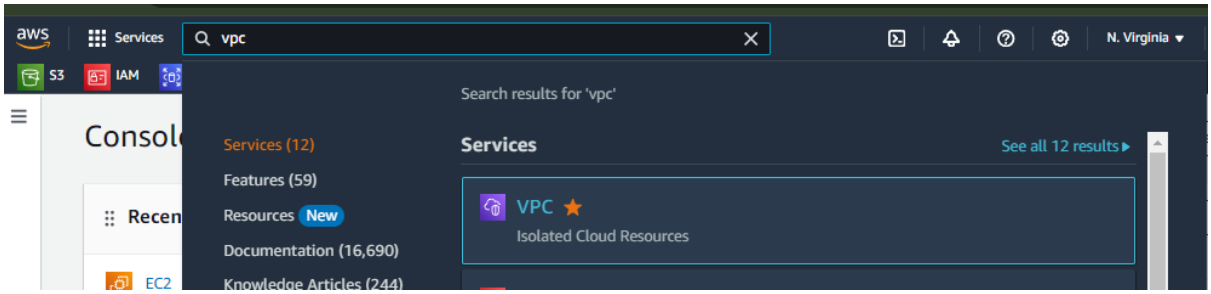
Subscribe to access AWS PlayCloud Labs

Lab Steps

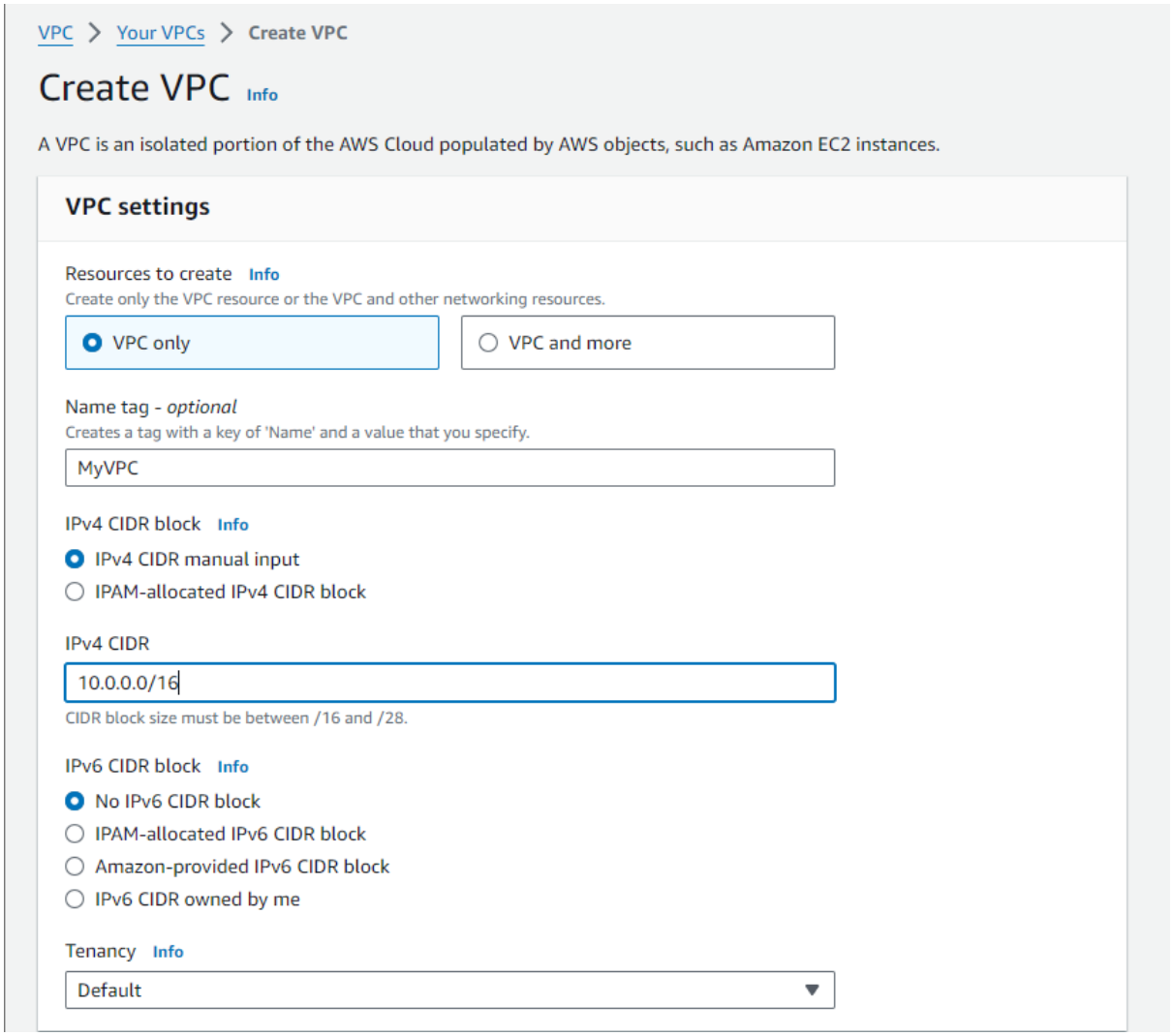
Setup Network Configurations

1. Create a VPC:

- **Go to the VPC dashboard in the AWS Management Console.**

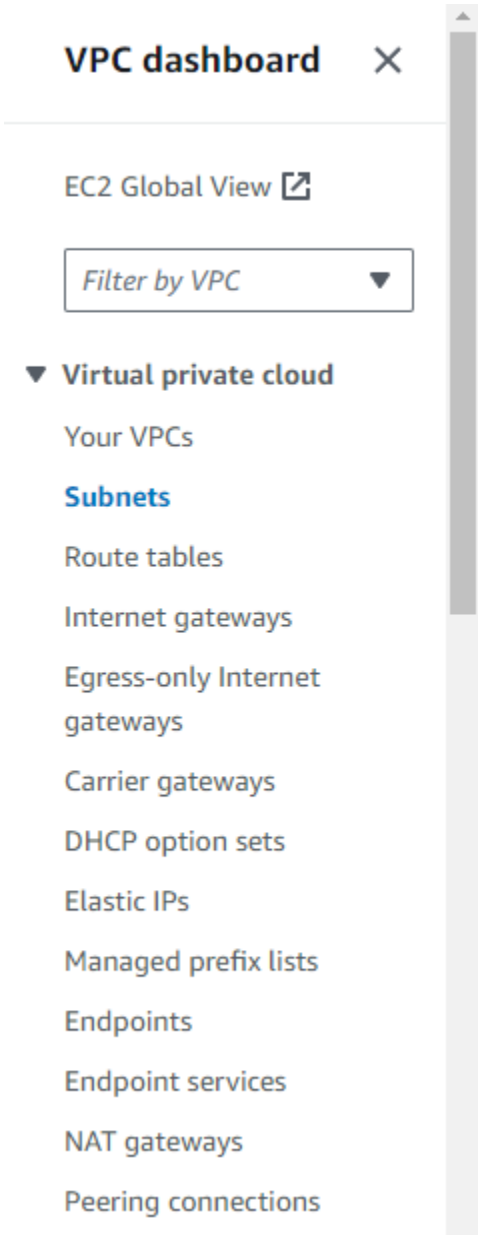


- **Click on Create VPC.**
- **Resource to create: VPC only**
- **Name: MyVPC**
- **CIDR block: 10.0.0.0/16**
- **Tenancy: Default**
- **Click Create.**

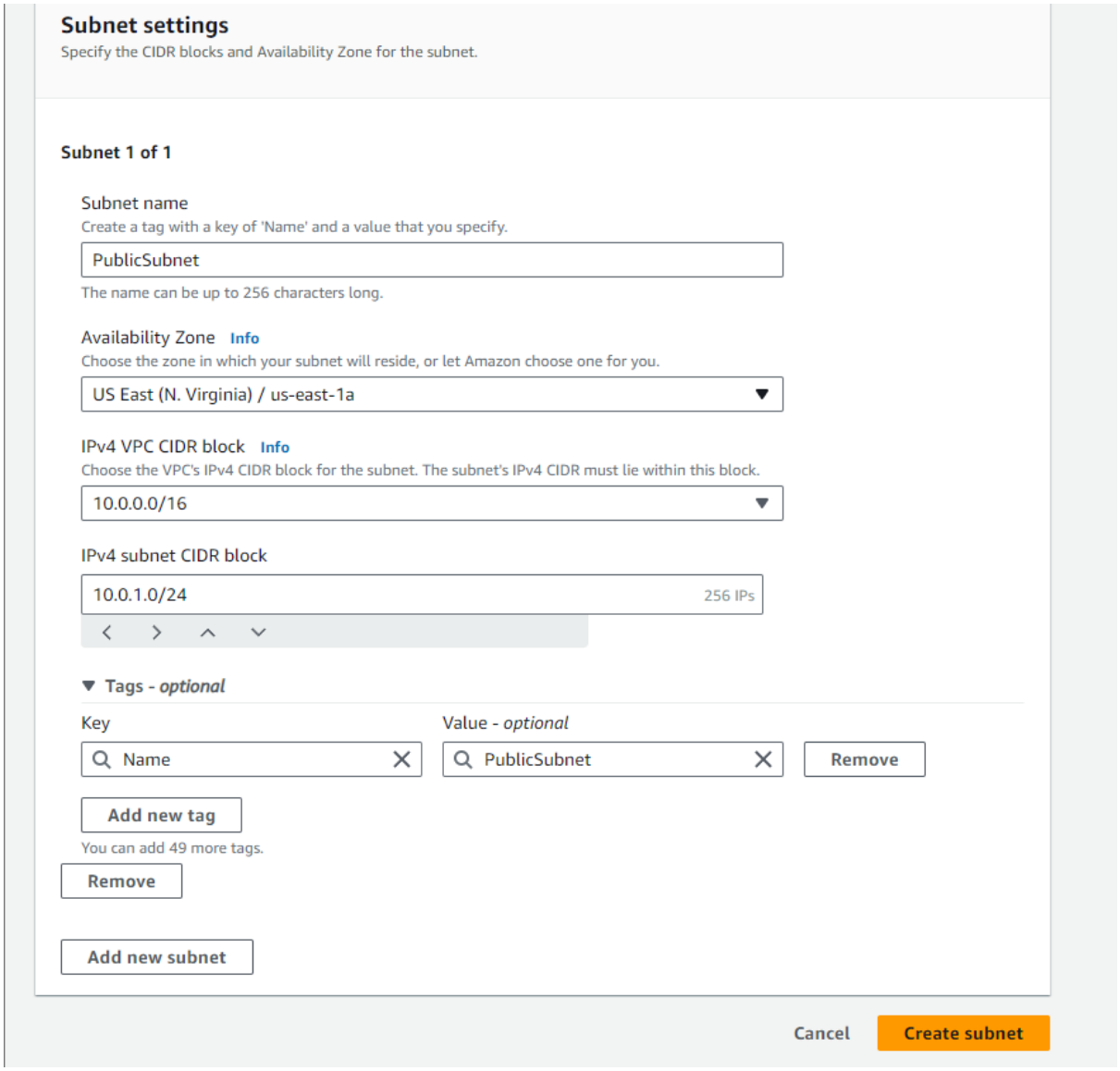


2. Create Subnets:

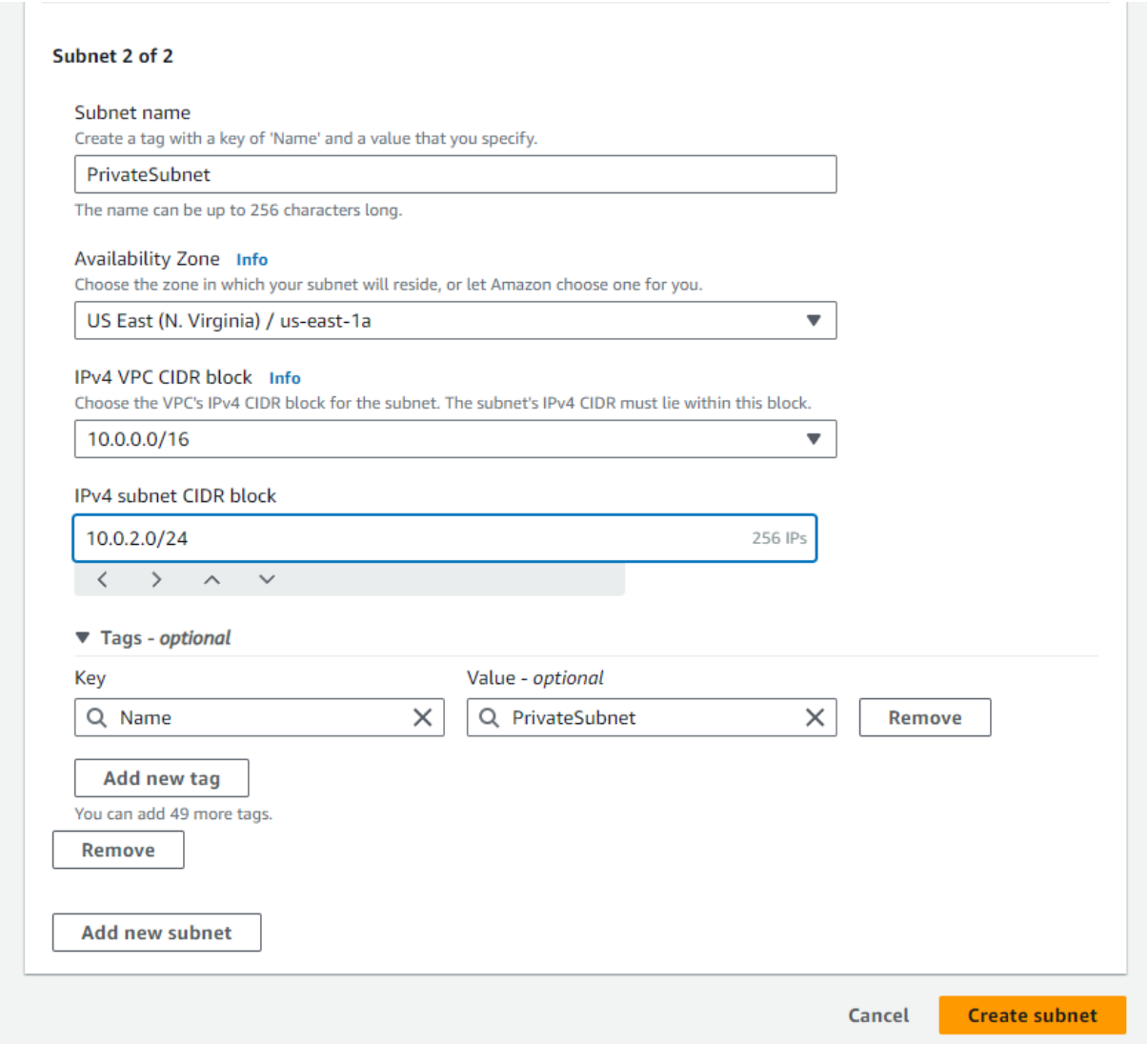
- **Navigate to Subnets in th VPC dashboard.**



- **Create two subnets with the following configurations:**
    - **Public Subnet:**
        - **VPC ID: MyVPC**
        - **Subnet name: PublicSubnet**
        - **Availability Zone: Select one from the list (e.g. us-east-1a)**
        - **IPv4 subnet CIDR block: 10.0.1.0/24**

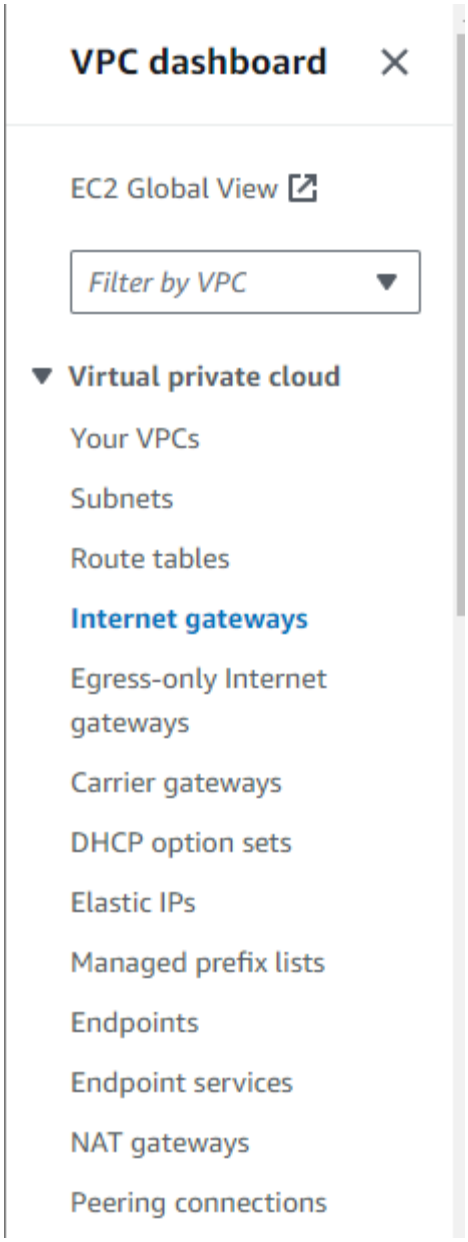


-
    -
        - **Add new subnet**
- **Private Subnet: (Click on Add new subnet)**
    - **Name: PrivateSubnet**
    - **IPv4 subnet CIDR block: 10.0.2.0/24**
    - **Availability Zone: Same as the public subnet**
    - **Click Create.**

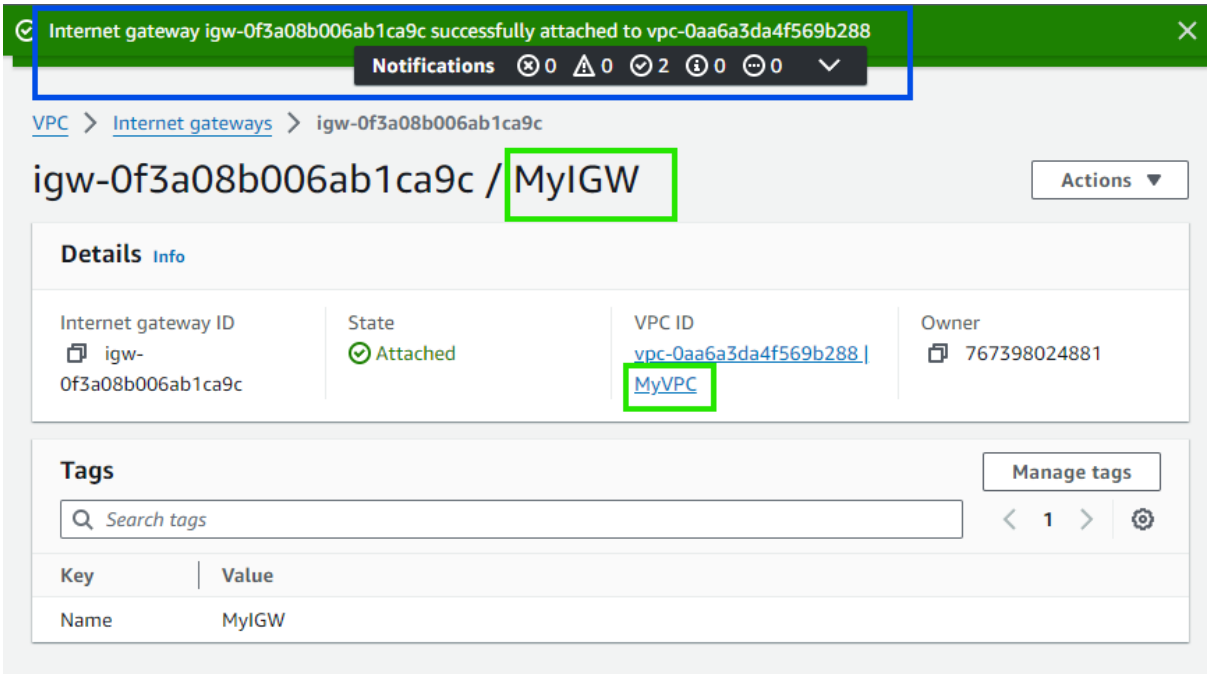


3. Create an Internet Gateway:

- **Go to Internet Gateways in the VPC dashboard.**



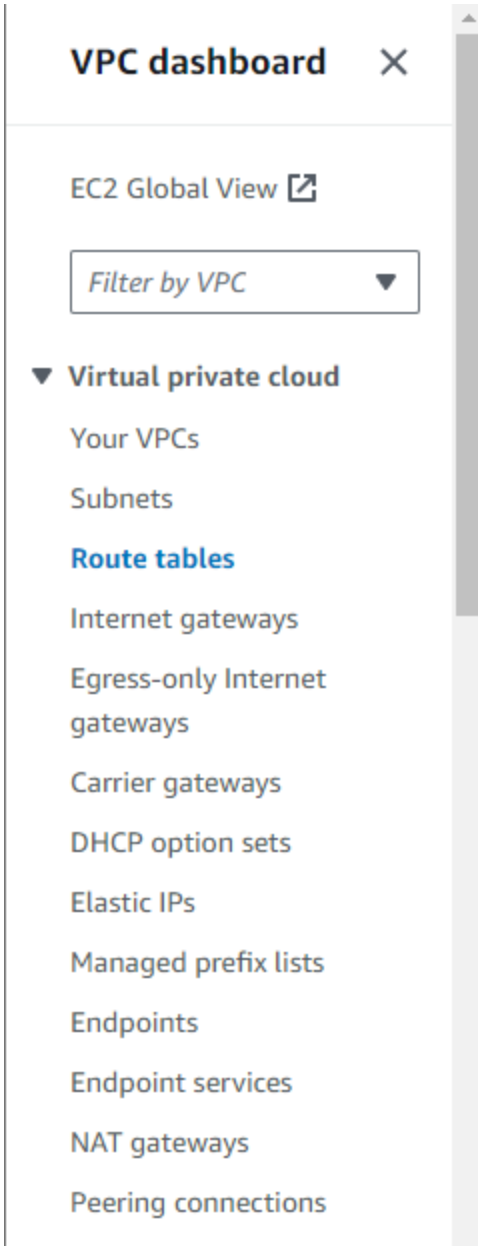
- **Click Create internet gateway.**
- **Name: MyIGW**
- **Click Create.**
- **Attach the Internet Gateway to MyVPC.**



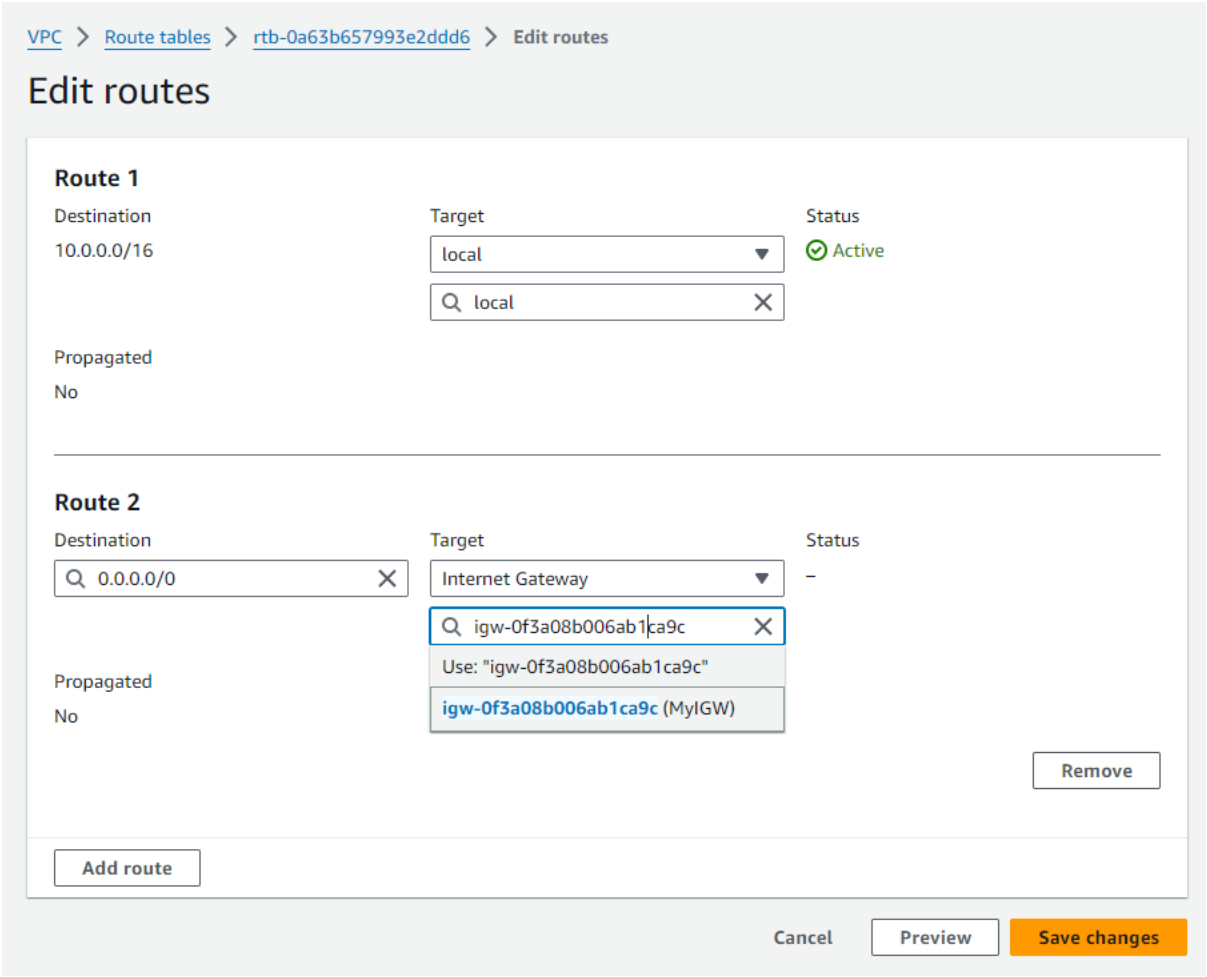
4. Create a new Route Table for the Public Subnet

Take note that there should be a new Route Table created after creating a new VPC.

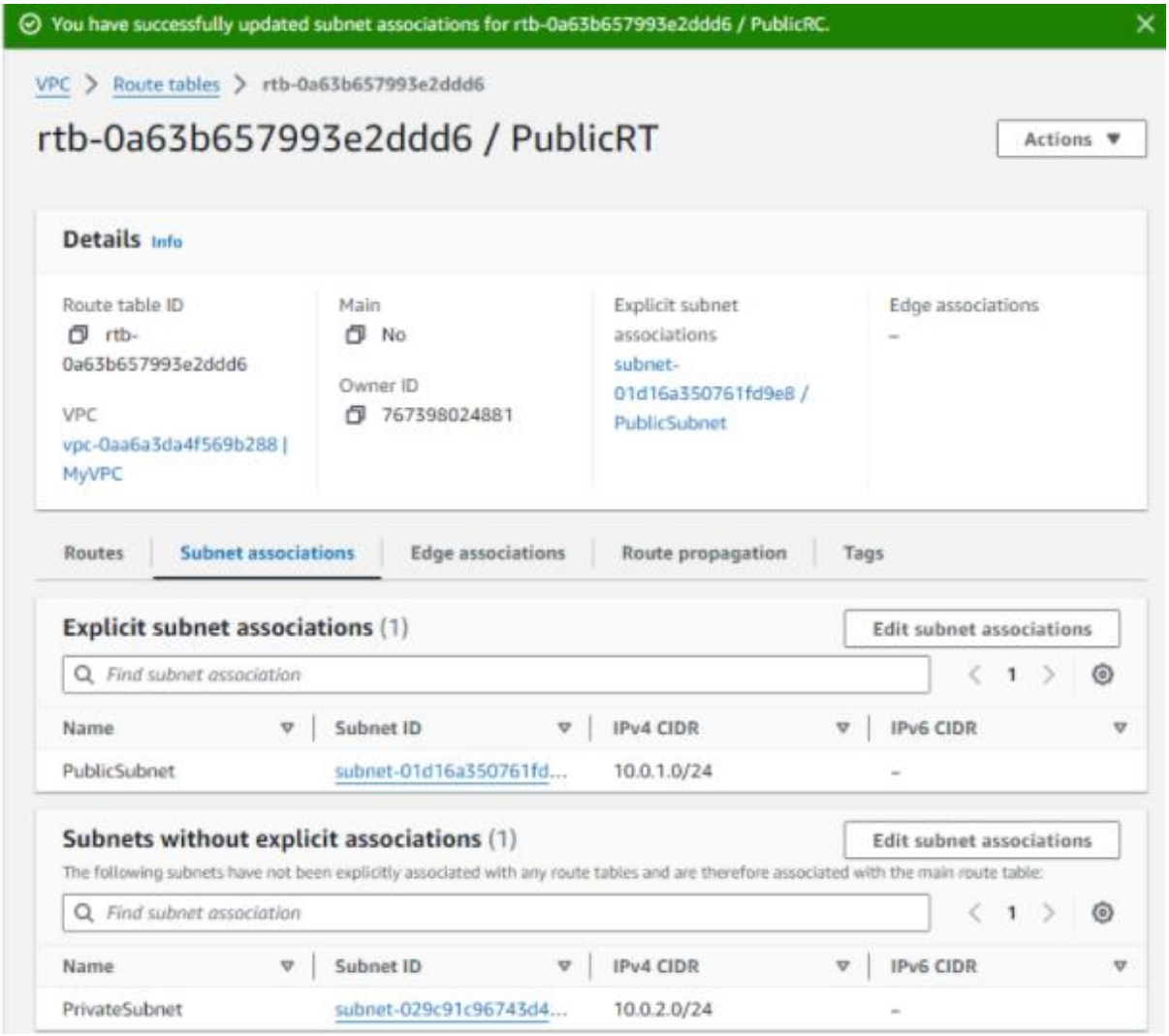
- **Go to the Route Tables section in the VPC dashboard**



- **Click Create route table**
- **Name: PublicRT**
- **VPC: MyVPC**
- **Click Create route table**
- **Edit routes: Add route 0.0.0.0/0 to target MyIGW.**
- **Save changes.**

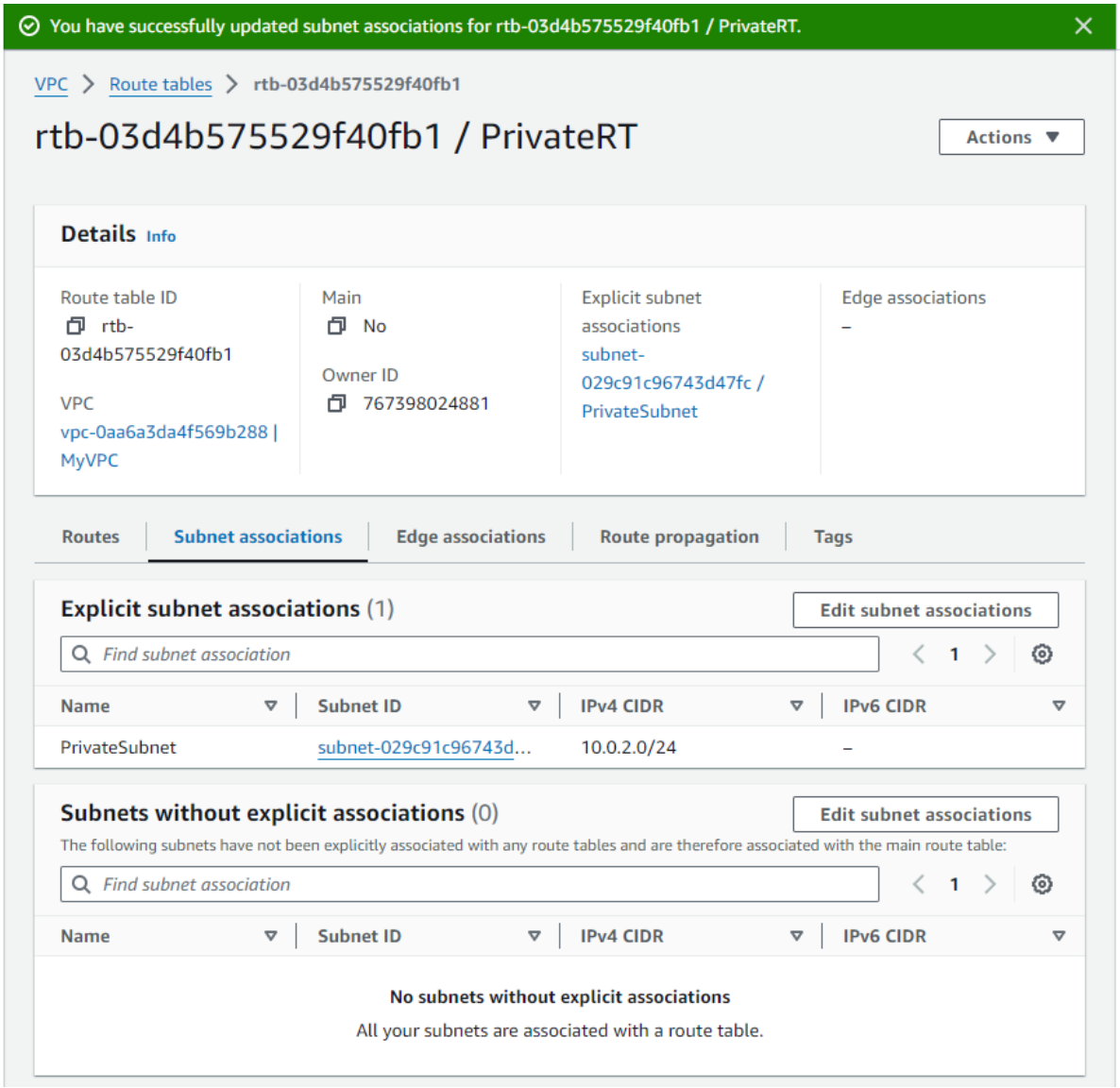


- **Edit Explicit subnet associations in the Subnet association tab: Associate PublicSubnet.**



5. Create another Route Table for Private Subnet

- **Create another route table: Name: PrivateRouteTable, VPC: MyVPC.**
- **Edit Explicit subnet associations in the Subnet association tab: Associate PrivateSubnet.**

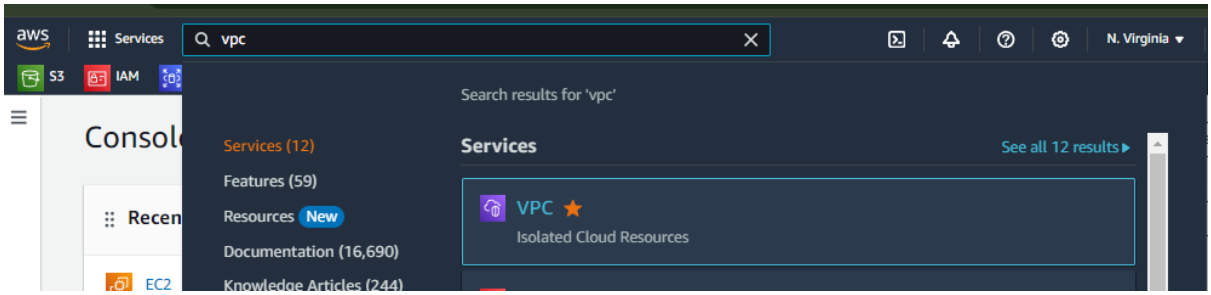


Lab Steps

Setup Network Configurations

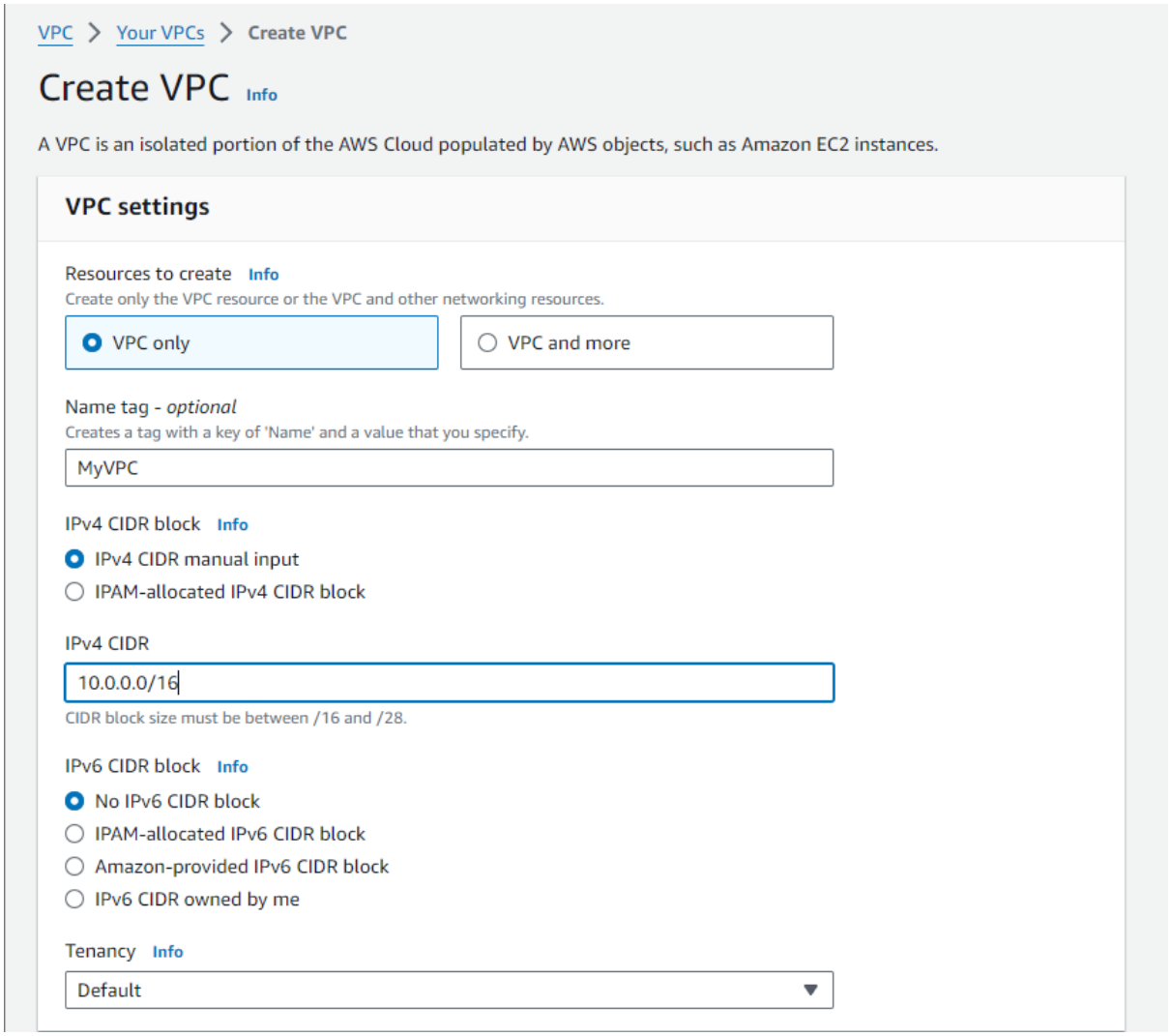
1. **Create a VPC**:

- Go to the VPC dashboard in the AWS Management Console.



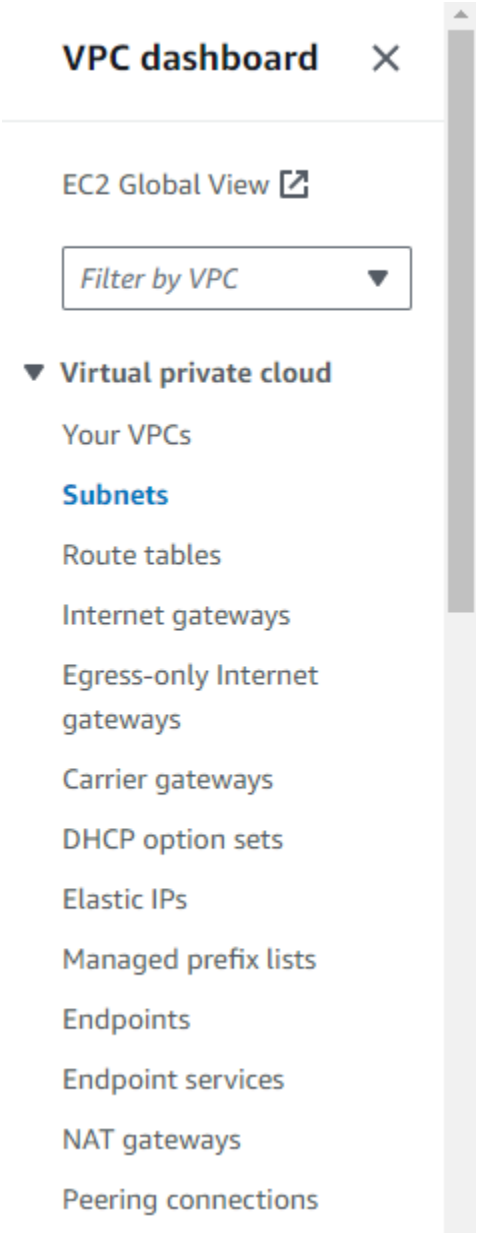
- Click on Create VPC.

- Resource to create: VPC only
- Name: MyVPC
- CIDR block: 10.0.0.0/16
- Tenancy: Default
- Click Create.

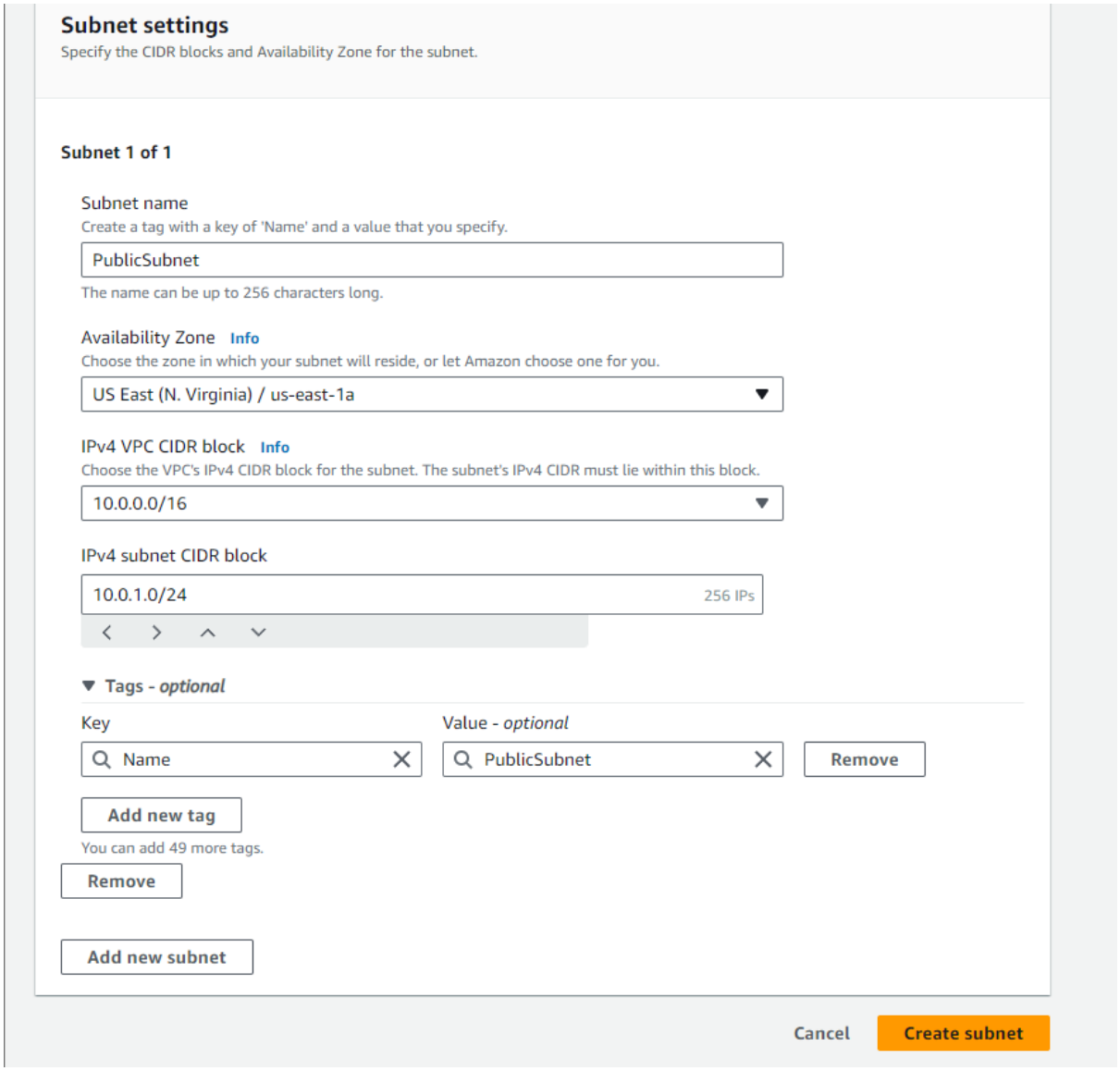


2. **Create Subnets**:

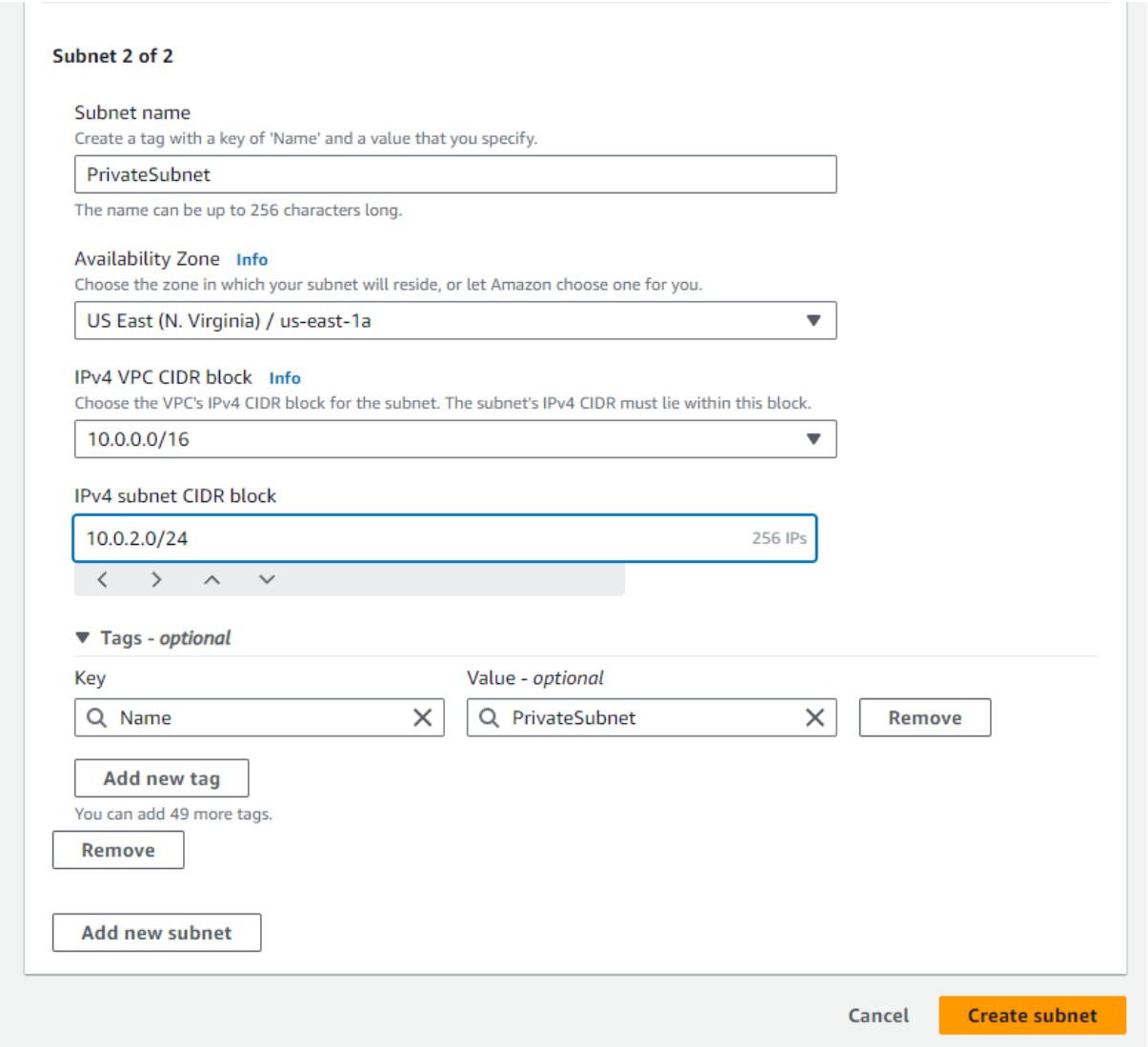
- Navigate to Subnets in th VPC dashboard.



- Create two subnets with the following configurations:
    - **Public Subnet**:
        - VPC ID: MyVPC
        - Subnet name: PublicSubnet
        - Availability Zone: Select one from the list (e.g. us-east-1a)
        - IPv4 subnet CIDR block: 10.0.1.0/24

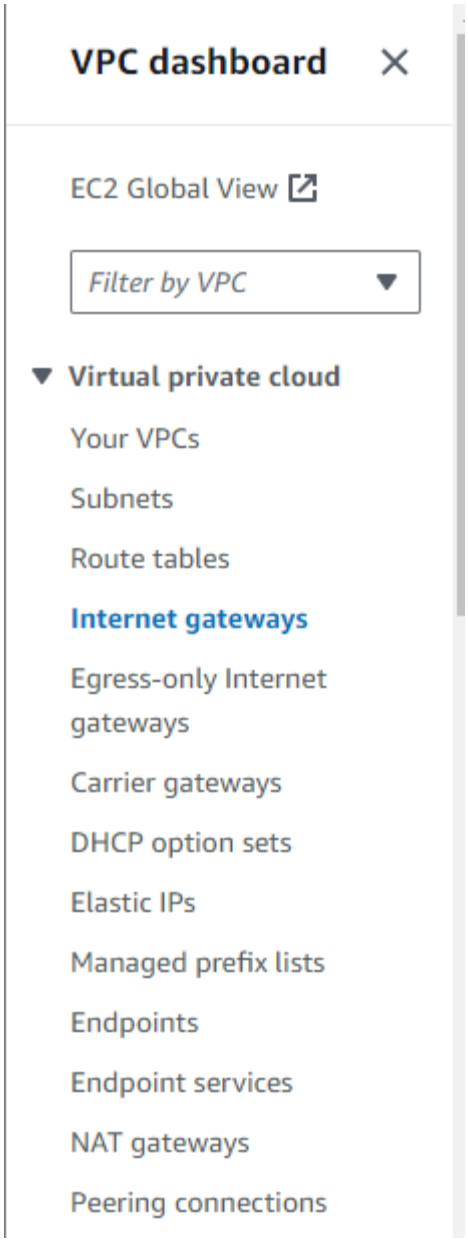


-
    -
        - **Add new subnet**
- **Private Subnet**: (Click on **Add new subnet**)
    - Name: PrivateSubnet
    - IPv4 subnet CIDR block: 10.0.2.0/24
    - Availability Zone: Same as the public subnet
    - Click Create.

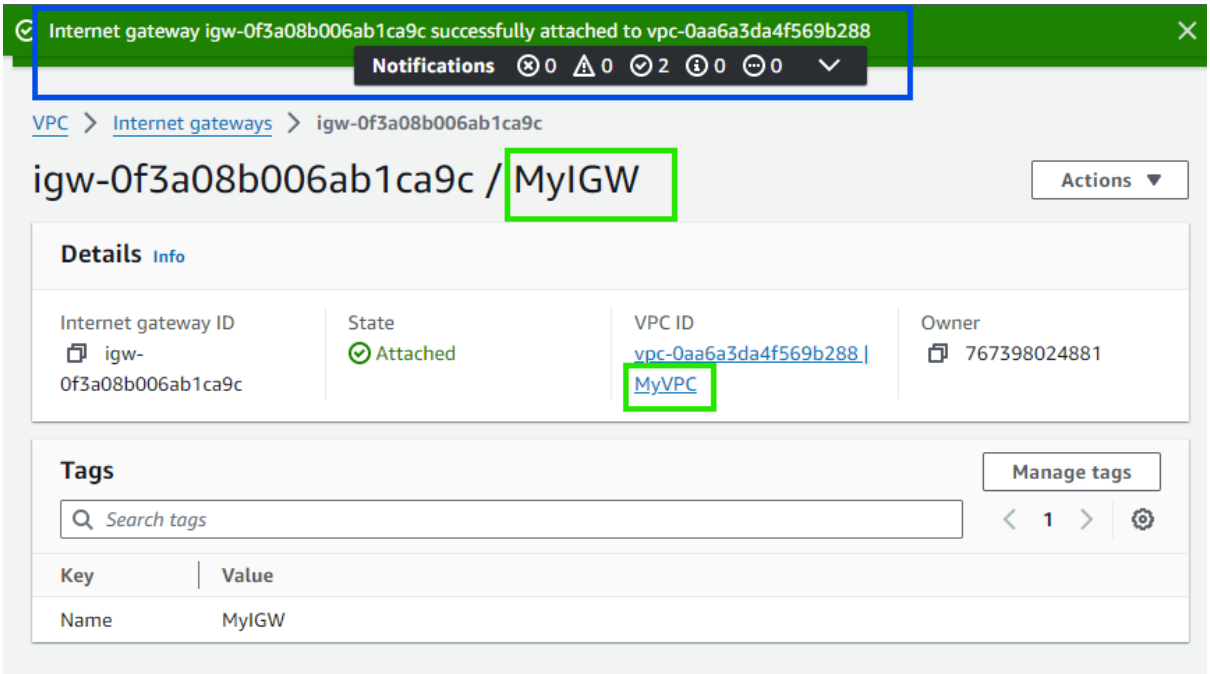


3. **Create an Internet Gateway**:

- Go to Internet Gateways in the VPC dashboard.



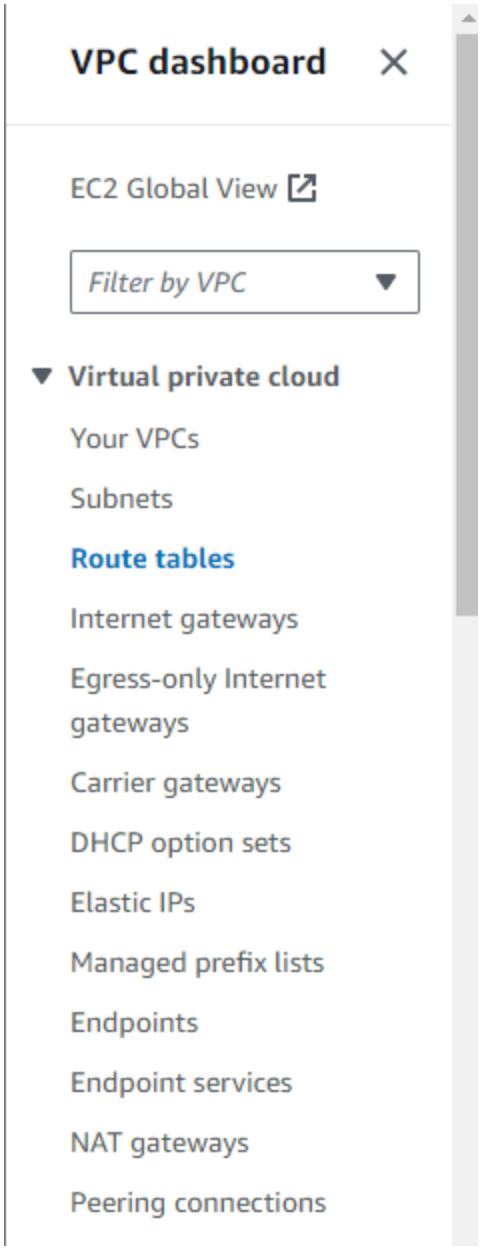
- Click **Create internet gateway**.
- Name: MyIGW
- Click Create.
- Attach the Internet Gateway to MyVPC.



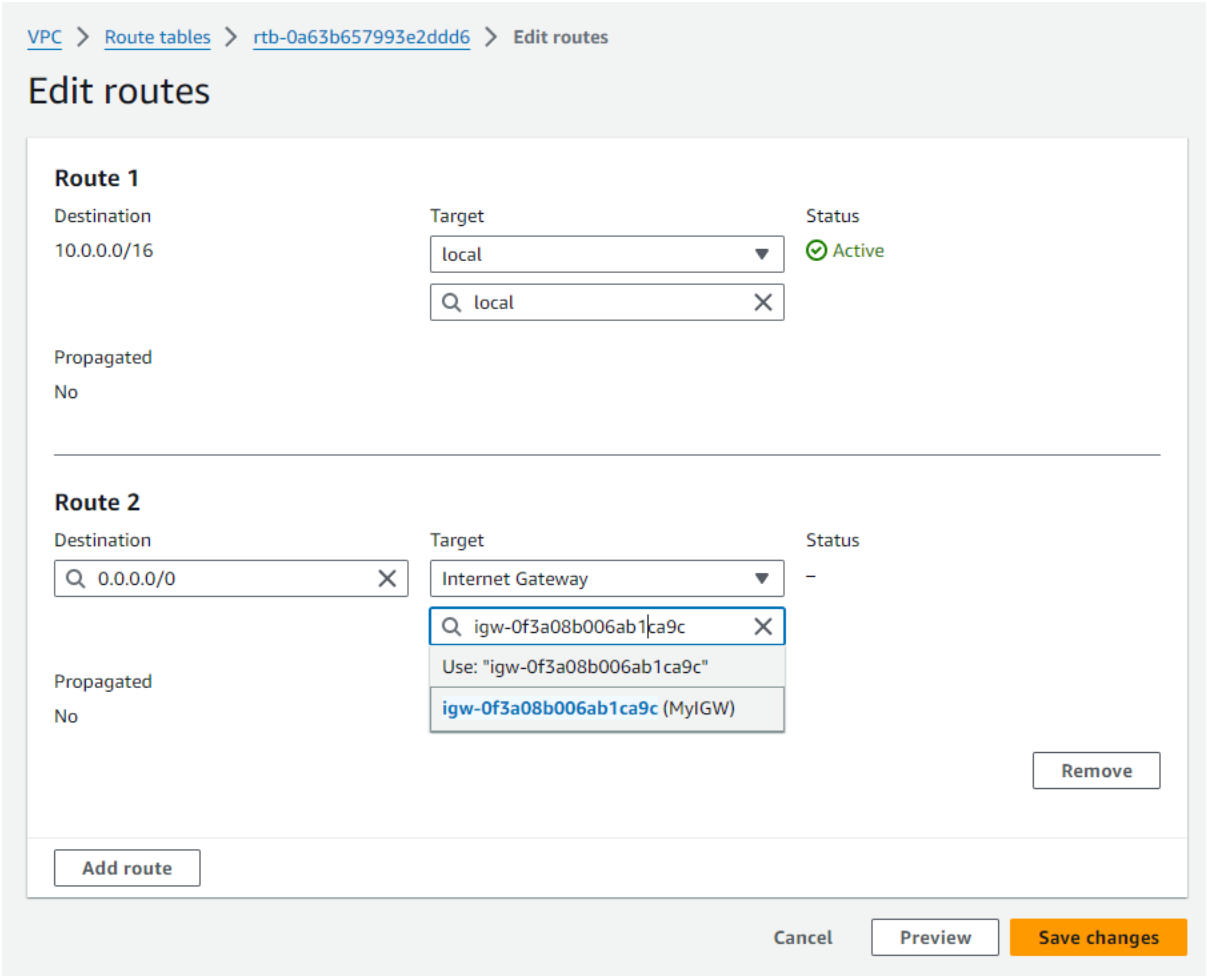
4. **Create a new Route Table for the Public Subnet**

Take note that there should be a new Route Table created after creating a new VPC.

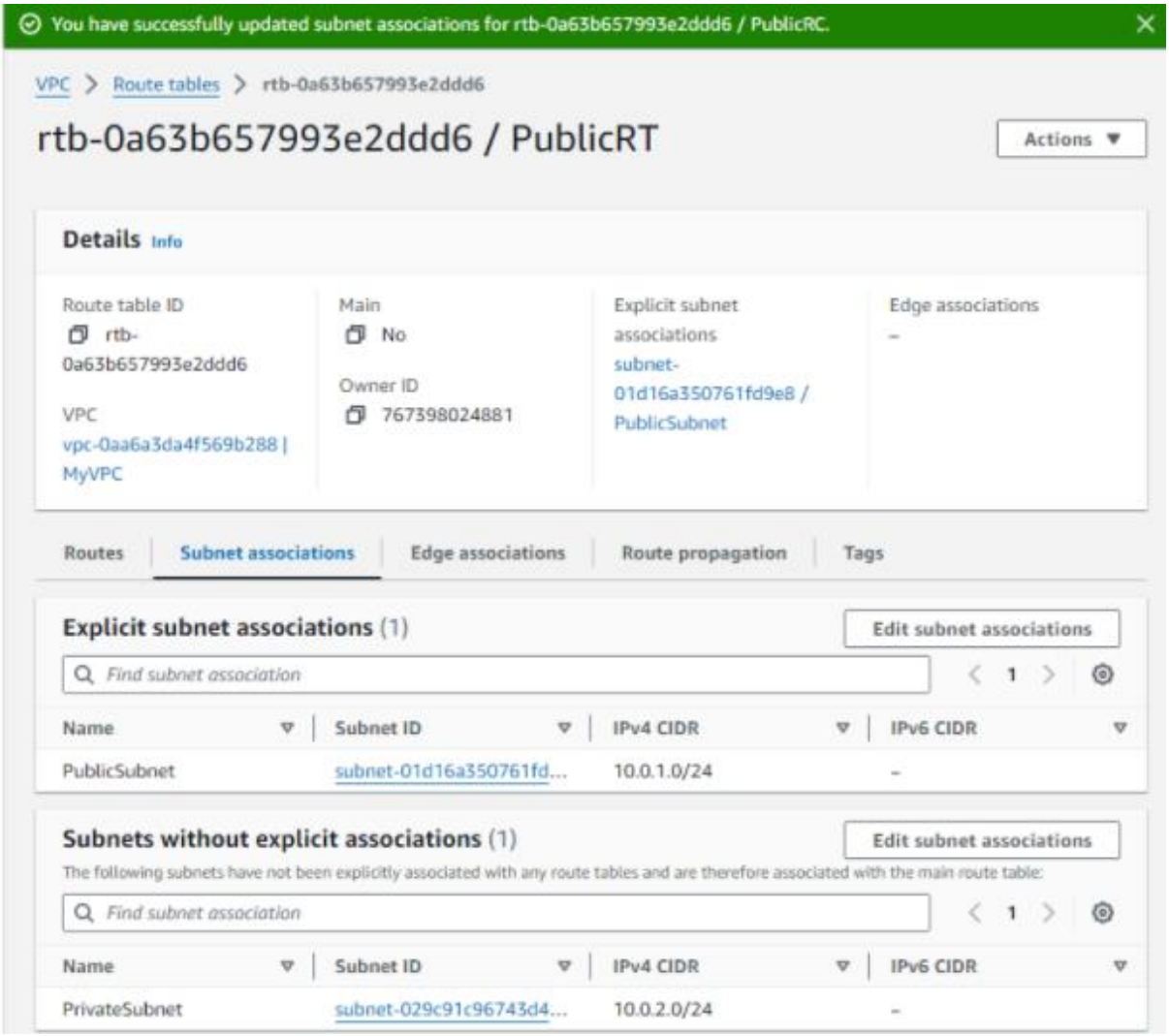
- Go to the Route Tables section in the VPC dashboard



- Click **Create route table**
- Name: PublicRT
- VPC: MyVPC
- Click **Create route table**
- Edit routes: Add route 0.0.0.0/0 to target MyIGW.
- Save changes.

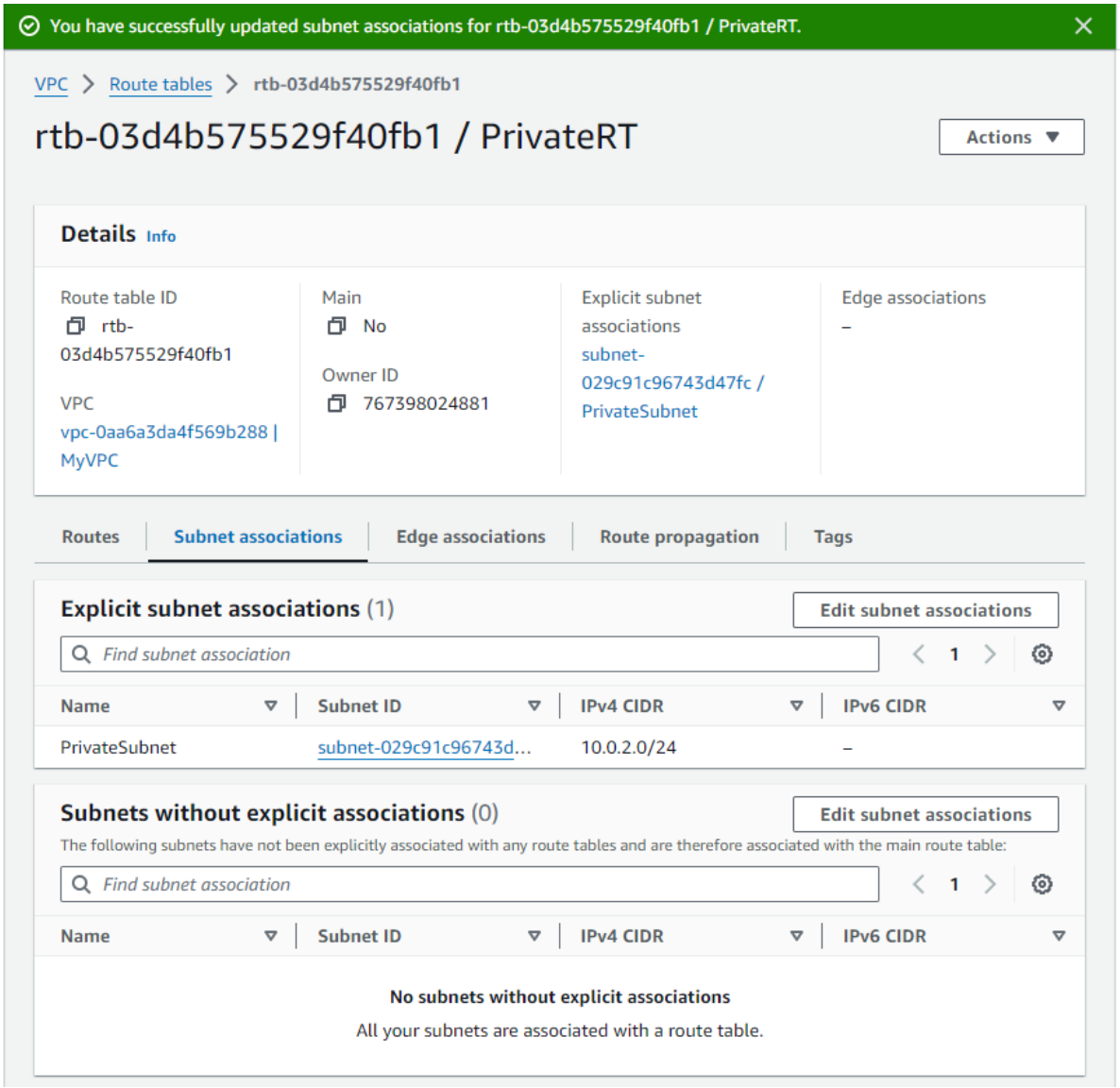


- Edit Explicit subnet associations in the **Subnet association tab**: Associate PublicSubnet.



5. **Create another Route Table for Private Subnet**

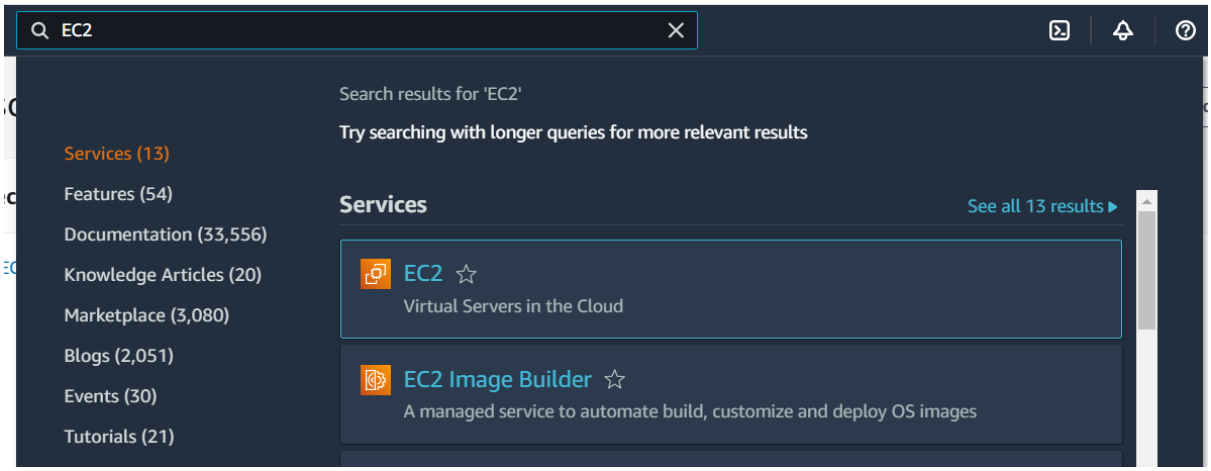
- Create another route table: Name: PrivateRouteTable, VPC: MyVPC.
- Edit Explicit subnet associations in the **Subnet association tab**: Associate PrivateSubnet.



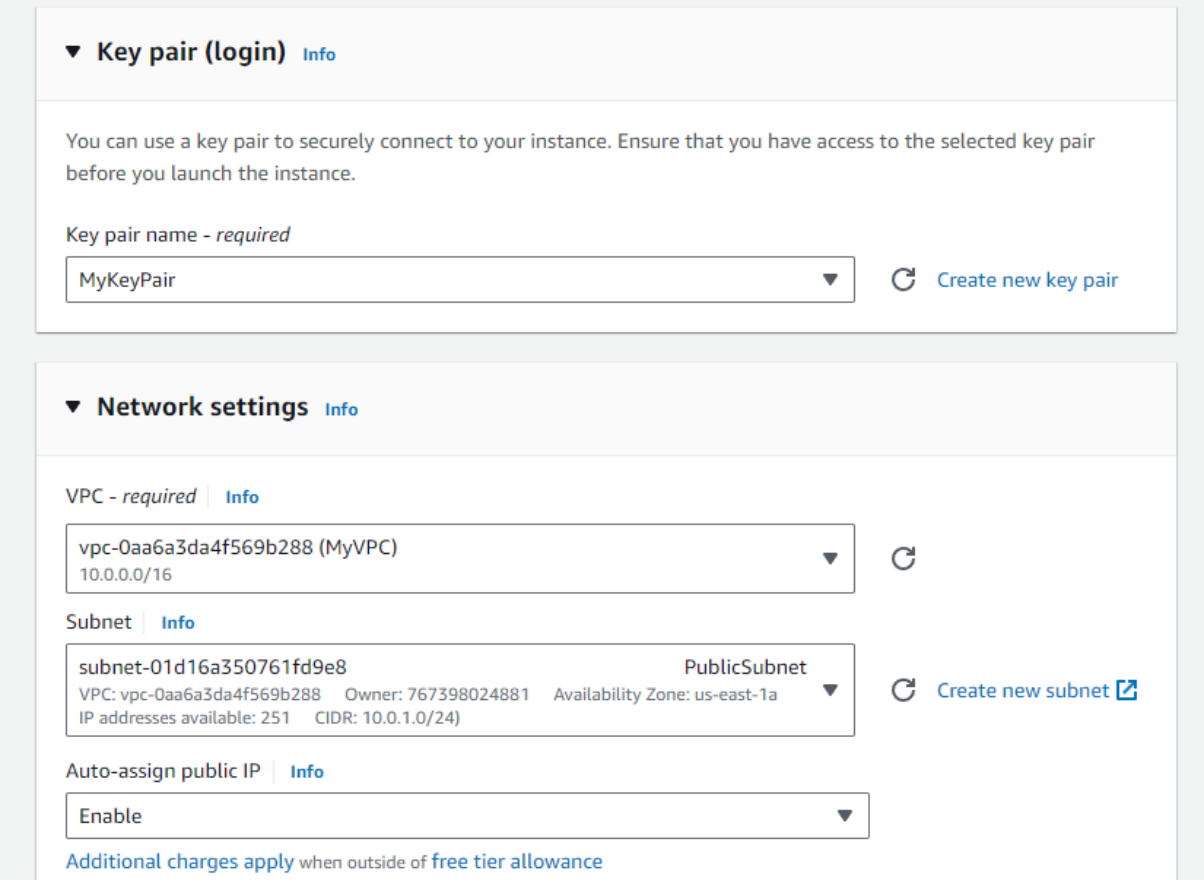
Launch Instances

1. **Launch an EC2 Instance in the Public Subnet**:

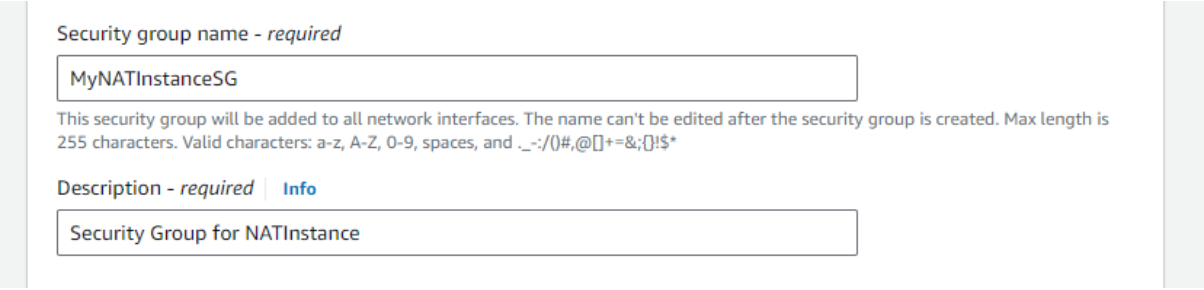
- Go to the EC2 dashboard.



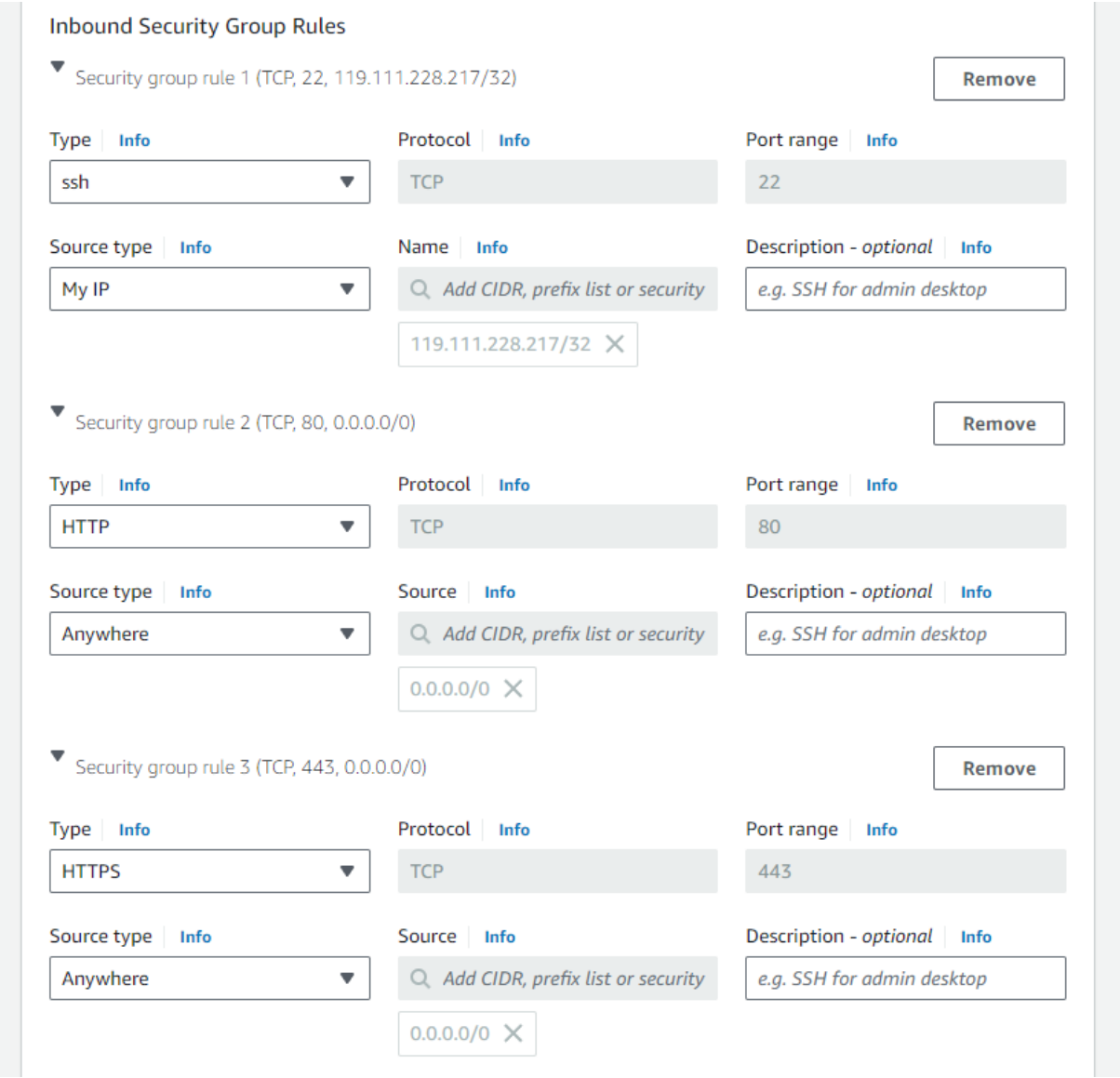
- Click Launch Instance.
- Name: MyNATInstance
- AMI: **Amazon Linux 2023 AMI**
- Instance Type: t2.micro
- Create key pair: MyKeyPair
- Network Settings:
    - VPC: MyVPC,
    - Subnet: PublicSubnet
- Enable Auto-assign Public IP.



- Firewall (security groups): Select **Create security group**
    - Security group name: MyNATInstanceSG
    - Description – *required*: **Security Group for NATInstance**



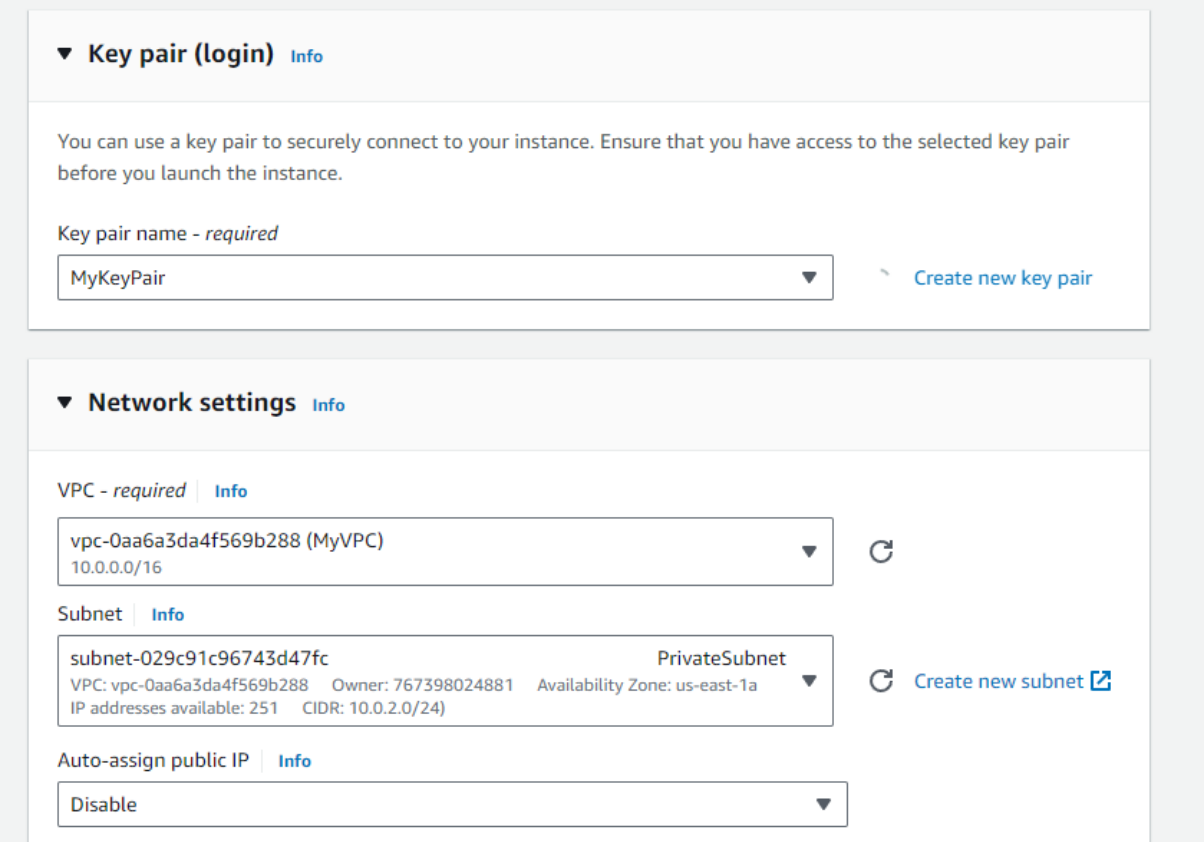
- Configure Security Group rules: Allow SSH (port **22**) from your IP and HTTP/HTTPS (port **80**/port **443**) from anywhere.



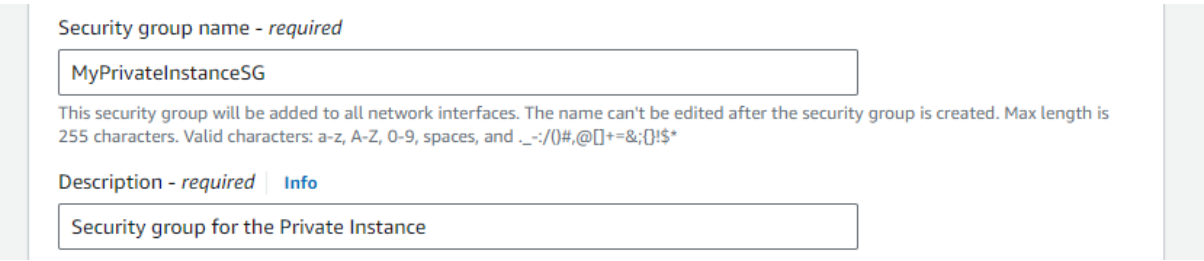
- Review and Launch.

2. **Launch another EC2 Instance in the Private Subnet**:

- Name: MyPrivateInstance
- Follow the same steps, but
    - select PrivateSubnet
    - Key pair: use the same key pair MyKeyPair
    - Network Settings:
        - VPC: MyVPC
        - Subnet: PrivateSubnet
    - Disable Auto-assign Public IP.

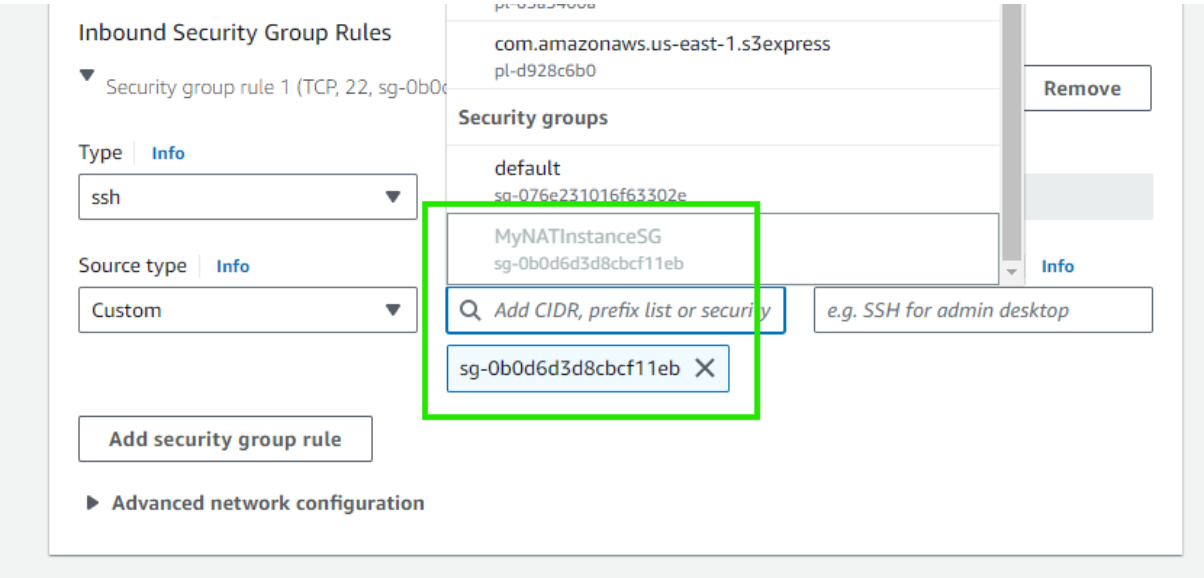


- Firewall (security groups): Select **Create security group**
    - Security group name: MyPrivateInstanceSG
    - Description – *required*: **Security group for Private Instance**



- Inbound Security Group Rules: Allow SSH (port 22) from MyNATInstanceSG.

You can also enter the IP subnet CIDR block of the PublicSubnet as an alternative here

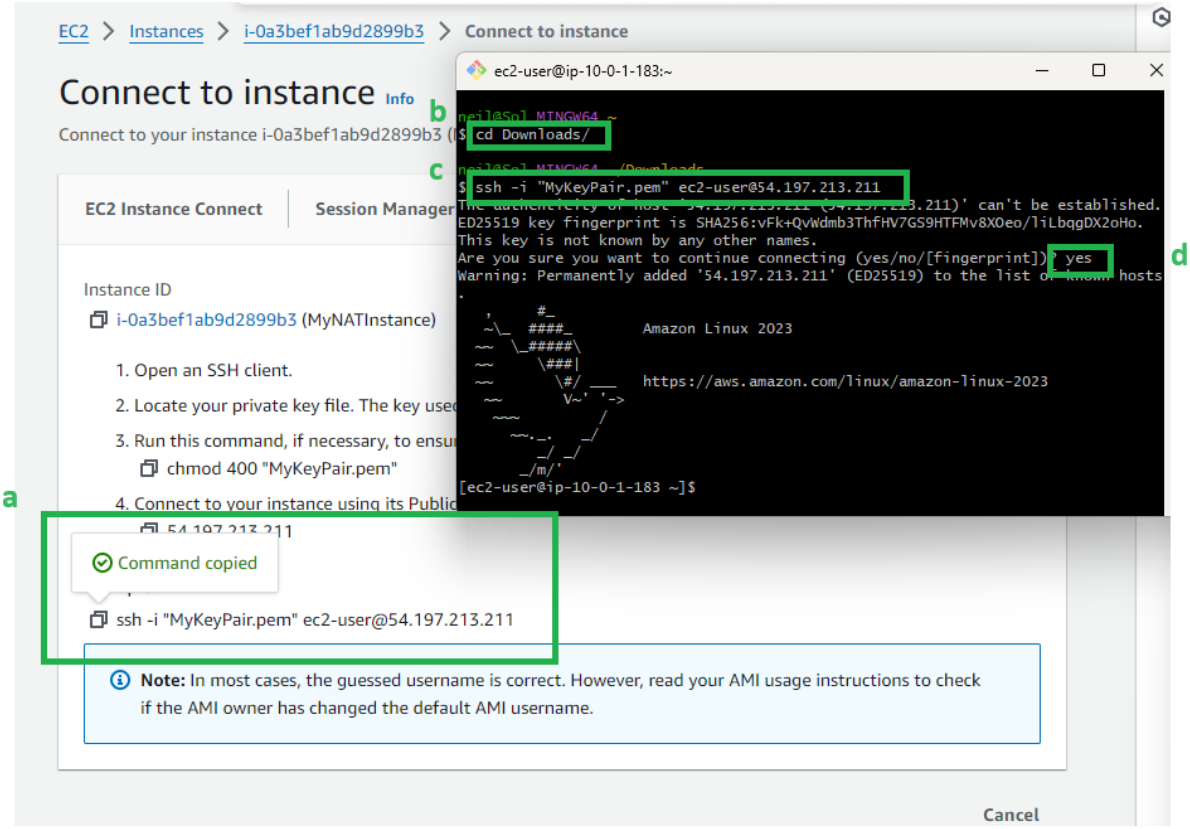


- Review and Launch.

Configure the NAT Instance

1. SSH into the MyNATInstance using your terminal (e.g., Git Bash, PuTTY).

- In AWS console copy the SSH command from EC2 DASHBOARD > select the MyNATInstance > Click on **Connect**.
- Go to your terminal, navigate to your directory where your .pem key is located using the cd command. (For example, in the Downloads/ folder)
- Paste the command into your terminal and press ENTER.
- A prompt confirmation will pop to the terminal, answer yes



- Run the following commands to enable IP forwarding and configure NAT:
    - Create a file using vi or vim:

sudo vi /etc/sysctl.d/custom-ip-forwarding.conf

-
    - Next, paste the following:

net.ipv4.ip_forward=1

-
    - Save and exit using the :wq! command!
    - Then, run the command:

sudo sysctl -p /etc/sysctl.d/custom-ip-forwarding.conf

```
[ec2-user@ip-10-0-1-183 ~]$ sudo sysctl -p /etc/sysctl.d/custom-ip-forwarding.conf
net.ipv4.ip_forward = 1
```

-
    - Next, install the iptables using the command:

sudo yum install iptables-services -y

```
[ec2-user@ip-10-0-1-183 ~]$ sudo yum install iptables-services -y
Last metadata expiration check: 0:27:01 ago on Tue Aug  6 02:18:11 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                 Repository     Size
================================================================================
Installing:
 iptables-services       noarch   1.8.8-3.amzn2023.0.2    amazonlinux    18 k
Installing dependencies:
 iptables-libs           x86_64   1.8.8-3.amzn2023.0.2    amazonlinux   401 k
 iptables-nft            x86_64   1.8.8-3.amzn2023.0.2    amazonlinux   183 k
 iptables-utils          x86_64   1.8.8-3.amzn2023.0.2    amazonlinux    43 k
 libnetfilter_conntrack  x86_64   1.0.8-2.amzn2023.0.2    amazonlinux    58 k
 libnfnetlink            x86_64   1.0.1-19.amzn2023.0.2   amazonlinux    30 k
 libnftnl                x86_64   1.2.2-2.amzn2023.0.2    amazonlinux    84 k

Transaction Summary
================================================================================
Install  7 Packages

Total download size: 816 k
Installed size: 2.9 M
Downloading Packages:
(1/7): iptables-libs-1.8.8-3.amzn2023.0.2.x86_6 5.7 MB/s | 401 kB     00:00
(2/7): iptables-services-1.8.8-3.amzn2023.0.2.n 254 kB/s |  18 kB     00:00
(3/7): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 2.2 MB/s | 183 kB     00:00
(4/7): iptables-utils-1.8.8-3.amzn2023.0.2.x86_ 2.4 MB/s |  43 kB     00:00
(5/7): libnetfilter_conntrack-1.0.8-2.amzn2023. 2.2 MB/s |  58 kB     00:00
(6/7): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_6 1.3 MB/s |  30 kB     00:00
(7/7): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm 3.7 MB/s |  84 kB     00:00
--------------------------------------------------------------------------------
Total                                           4.7 MB/s | 816 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64              1/7
  Installing       : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64     2/7
  Installing       : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64              3/7
  Installing       : libnftnl-1.2.2-2.amzn2023.0.2.x86_64                   4/7
  Installing       : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64               5/7
  Running scriptlet: iptables-nft-1.8.8-3.amzn2023.0.2.x86_64               5/7
  Installing       : iptables-utils-1.8.8-3.amzn2023.0.2.x86_64             6/7
  Installing       : iptables-services-1.8.8-3.amzn2023.0.2.noarch          7/7
  Running scriptlet: iptables-services-1.8.8-3.amzn2023.0.2.noarch          7/7
  Verifying        : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64              1/7
```

-
    - Check the name of the primary network interface uisng the command:

netstat -i

```
[ec2-user@ip-10-0-1-183 ~]$ netstat -i
Kernel Interface table
Iface             MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
enX0             9001     3032      0      0 0          3033      0      0      0 BMRU
lo              65536       12      0      0 0            12      0      0      0 LRU
```

- Take note of this network Interface and use the command following command:

Do not forget to change the <primary_network_interface> placeholder

sudo iptables -t nat -A POSTROUTING -o enX0 -j MASQUERADE

sudo iptables -A FORWARD -i eth0 -o enX0 -m state --state RELATED,ESTABLISHED -j ACCEPT

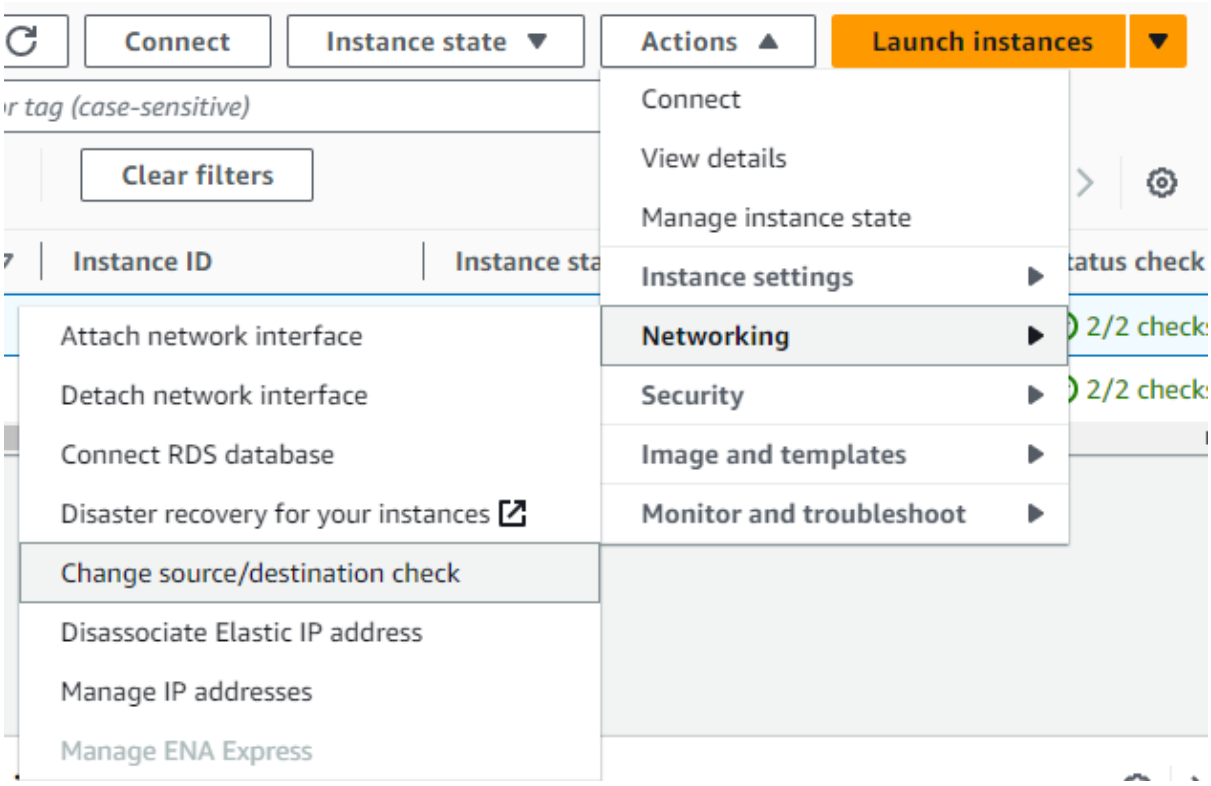
sudo iptables -A FORWARD -i eth0 -o enX0 -j ACCEPT

sudo service iptables save

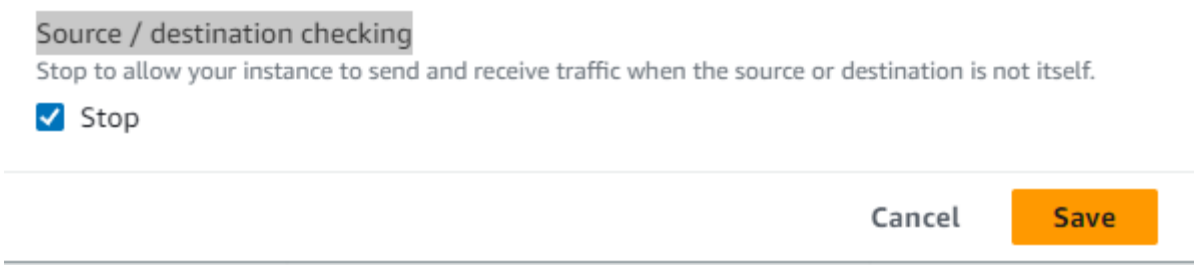
sudo service iptables restart

```
[ec2-user@ip-10-0-1-183 ~]$ sudo /sbin/iptables -t nat -A POSTROUTING -o enX0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o enX0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o enX0 -j ACCEPT
sudo service iptables save
sudo service iptables restart
iptables: Saving firewall rules to /etc/sysconfig/iptables: [  OK  ]
Redirecting to /bin/systemctl restart iptables.service
```

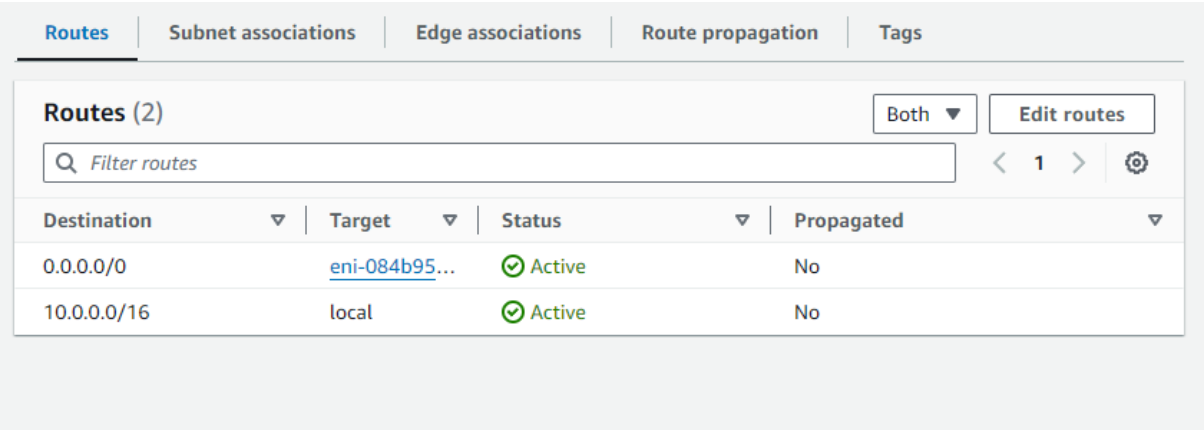
2. In the EC2 Dashboard, navigate to your NAT Instance. Click **Actions > Networking > Change source/destination check**



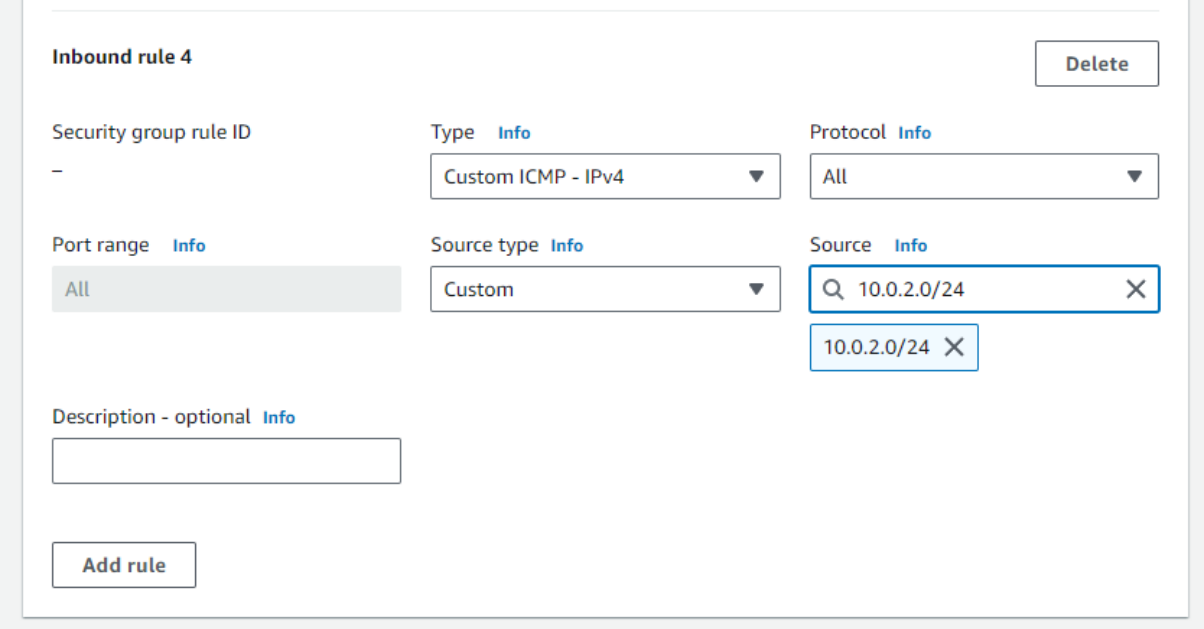
3. Check the checkbox Stop in the **Source / destination checking**



4. Update the Route Table of the Private Subnet, Destination: 0.0.0.0/0 and the Target: Instance (MyNATInstance)



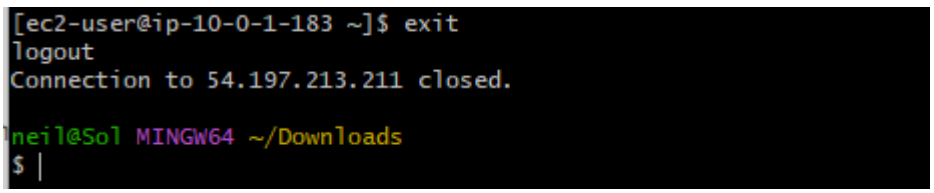
5. Update also the Security for the NAT Instance, adding the Type: ICMP - IPv4, Source: 10.0.2.0/24 (The IPv4 subnet CIDR range)



Access the PrivateInstnace

1. Firsts, we need to copy the Private Key (MyKeyPair.pem) to the NAT Instance in our local machine by:

- Exit from your current ssh connection with the NAT Intsance uisng the exit command and go to directory of your .pem key first



- Copy, edit, and paste the following command accordingly:
  (Since we used the same key pair in both intance)

Do not forget to change the placeholder

scp -i <your_NAT_Instance_Key_Pair> <Your_Private_Instance_Key_Pair> ec2-user@<public_IP_address_of_NAT_Instance:/home/ec2-user/

```
neil@Sol MINGW64 ~/Downloads
$ scp -i MyKeyPair.pem MyKeyPair.pem ec2-user@54.197.213.211:/home/ec2-user/
MyKeyPair.pem                                 100% 1678     7.1KB/s   00:00

neil@Sol MINGW64 ~/Downloads
$
```

2. SSH to the NATInstance again.

Do not forget to change the placeholder

ssh -i your-key-pair.pem ec2-user@<NATInstance-Public-IP>

```
neil@Sol MINGW64 ~/Downloads
$ ssh -i "MyKeyPair.pem" ec2-user@54.197.213.211
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Tue Aug  6 02:59:07 2024 from 119.111.228.217
[ec2-user@ip-10-0-1-183 ~]$
```

3. From the NAT instance, Change the permission of the MyKeyPair.pem:

- Let's check first if teh .pem key was successfully copied using the ls or ls -la command:

```
[ec2-user@ip-10-0-1-183 ~]$ ls
MyKeyPair.pem
[ec2-user@ip-10-0-1-183 ~]$
```

- Use the following command to add permission to the key:

sudo chmod 400 MyKeyPair.pem

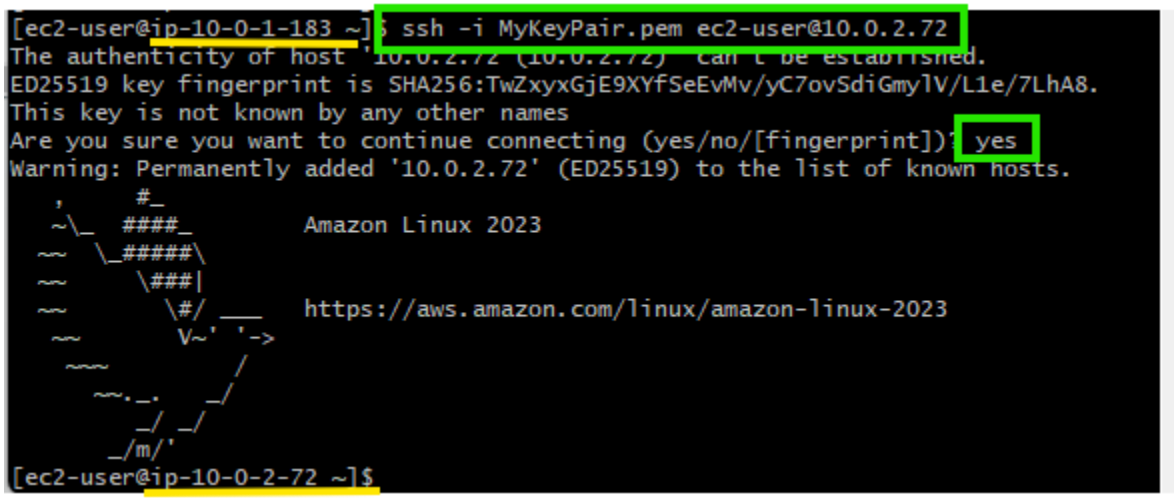
```
[ec2-user@ip-10-0-1-183 ~]$ sudo chmod 400 MyKeyPair.pem
```

4. SSH into the Private instance:

Do not forget to change the placeholder

ssh -i MyKeyPair.pem ec2-user@<PrivateInstance-Private-IP>

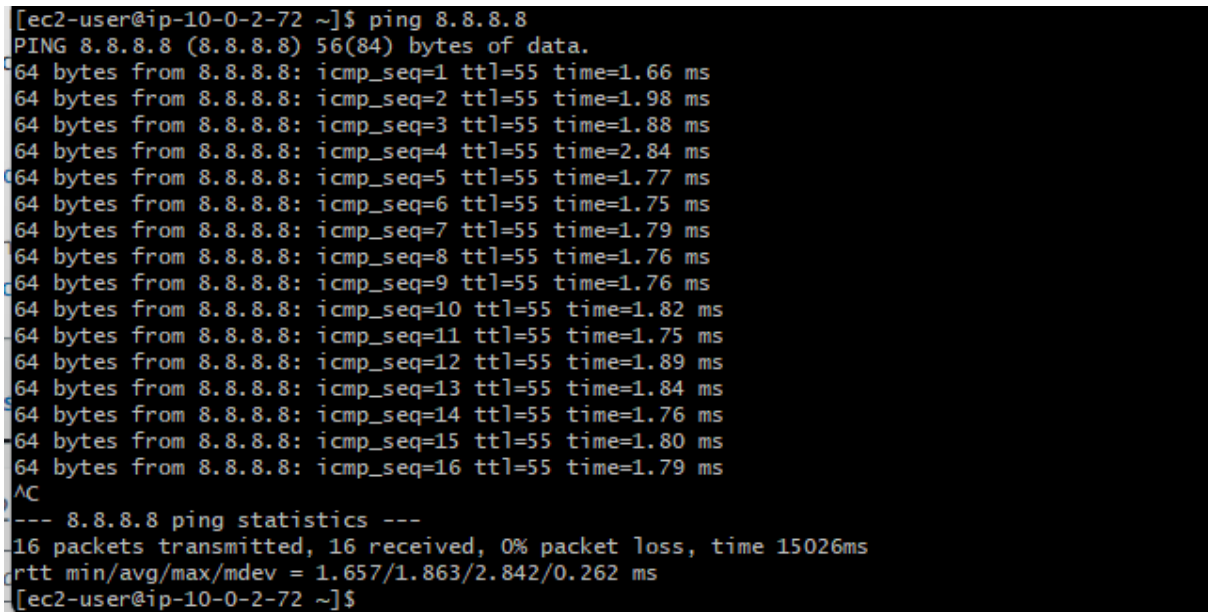
A prompt confirmation will pop to the terminal, answer yes



Notice the ec2-user@<IP_Address> has changed.

5. Now, ping google or 8.8.8.8

ping 8.8.8.8



That's it! Congratulations! You have experienced setting up and learning the importance of a NAT instance in providing secure connectivity for instances in a private subnet. By enabling communication with a NAT instance, the private instance remains protected from direct inbound internet traffic while still being able to perform necessary updates and communications through the NAT instance. Understanding and configuring NAT instances is essential for managing secure and efficient cloud architectures, ensuring that private instances can securely access external resources when needed.