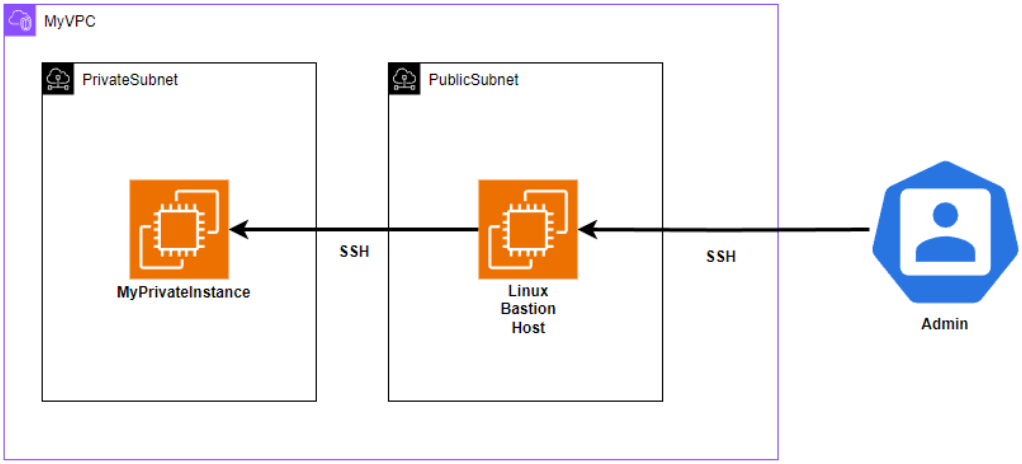
Guided Lab: Setting Up a Linux Bastion Host on AWS

Description

A Bastion host plays a vital role in securely accessing instances within private subnets, serving as a gateway for controlling resource management.

This guided lab will walk you through setting up a Bastion host using a Linux instance on AWS. By leveraging a Bastion host, administrators can securely manage private instances without exposing them to the internet, significantly enhancing the security posture of your AWS environment.



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:

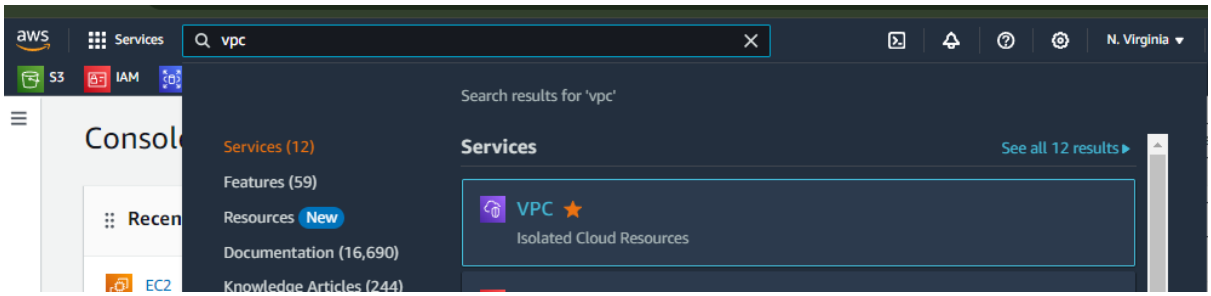
- Understand the concept and purpose of a Bastion host.
- Launch and configure a Linux Bastion host in the public subnet
- Access instances in the private subnet securely through the Bastion host.

Lab Steps

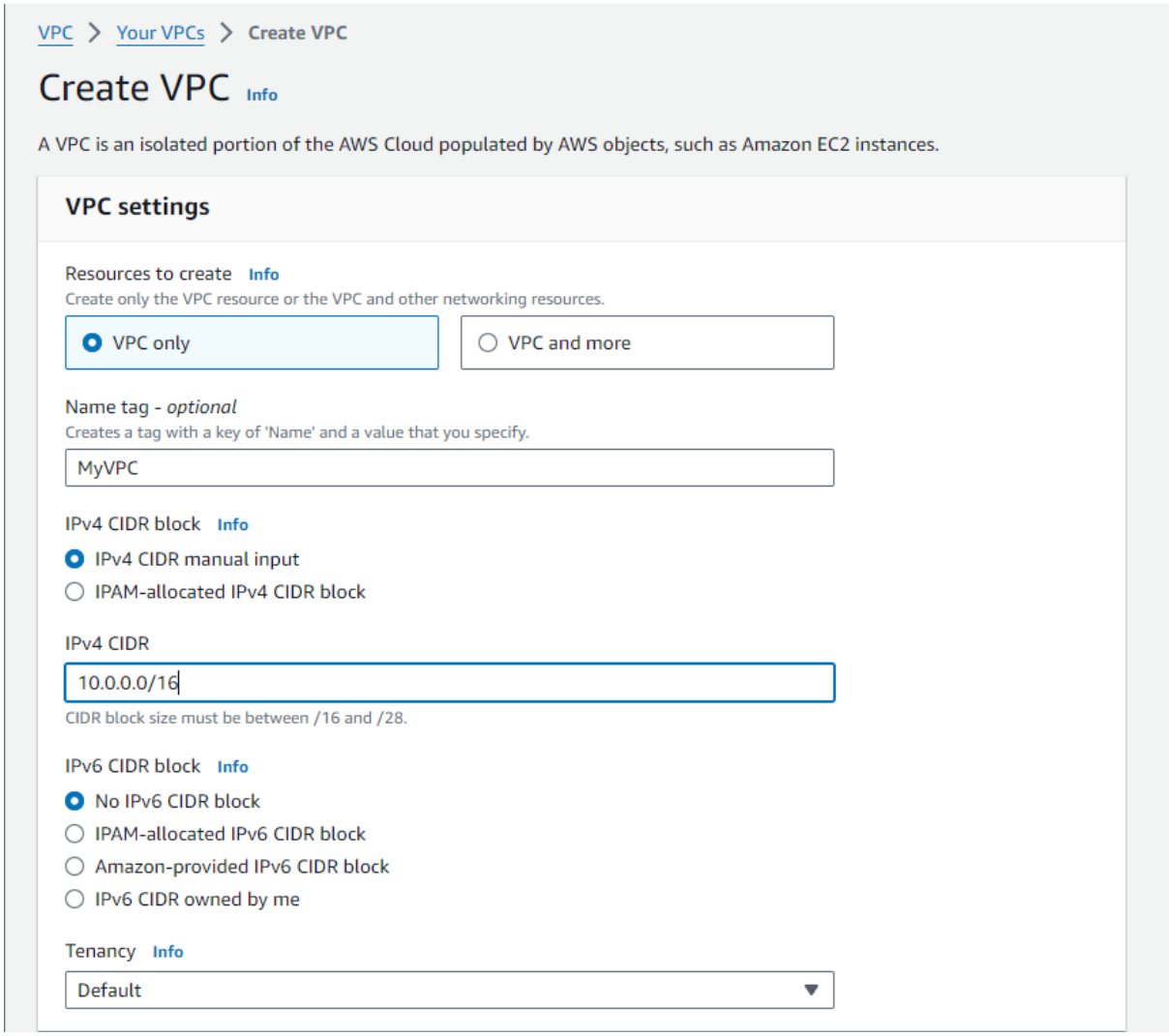
Setup Network Confgurations

1. **Create a VPC**:

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



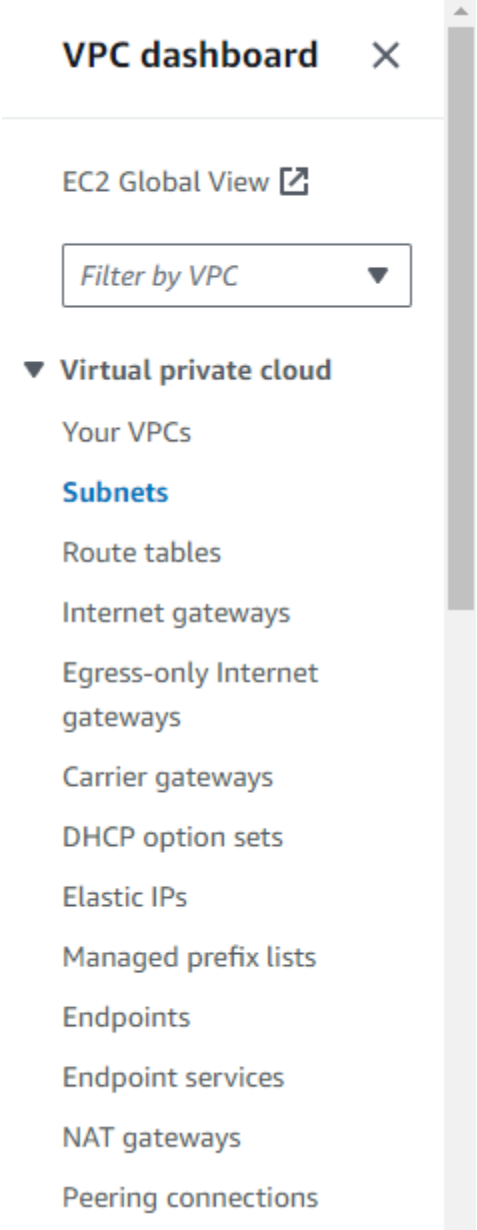
- Click on Create VPC.
- Resource to create: VPC only
- Name: MyVPC
- CIDR block: 10.0.0.0/16
- Tenancy: Default
- Click Create.



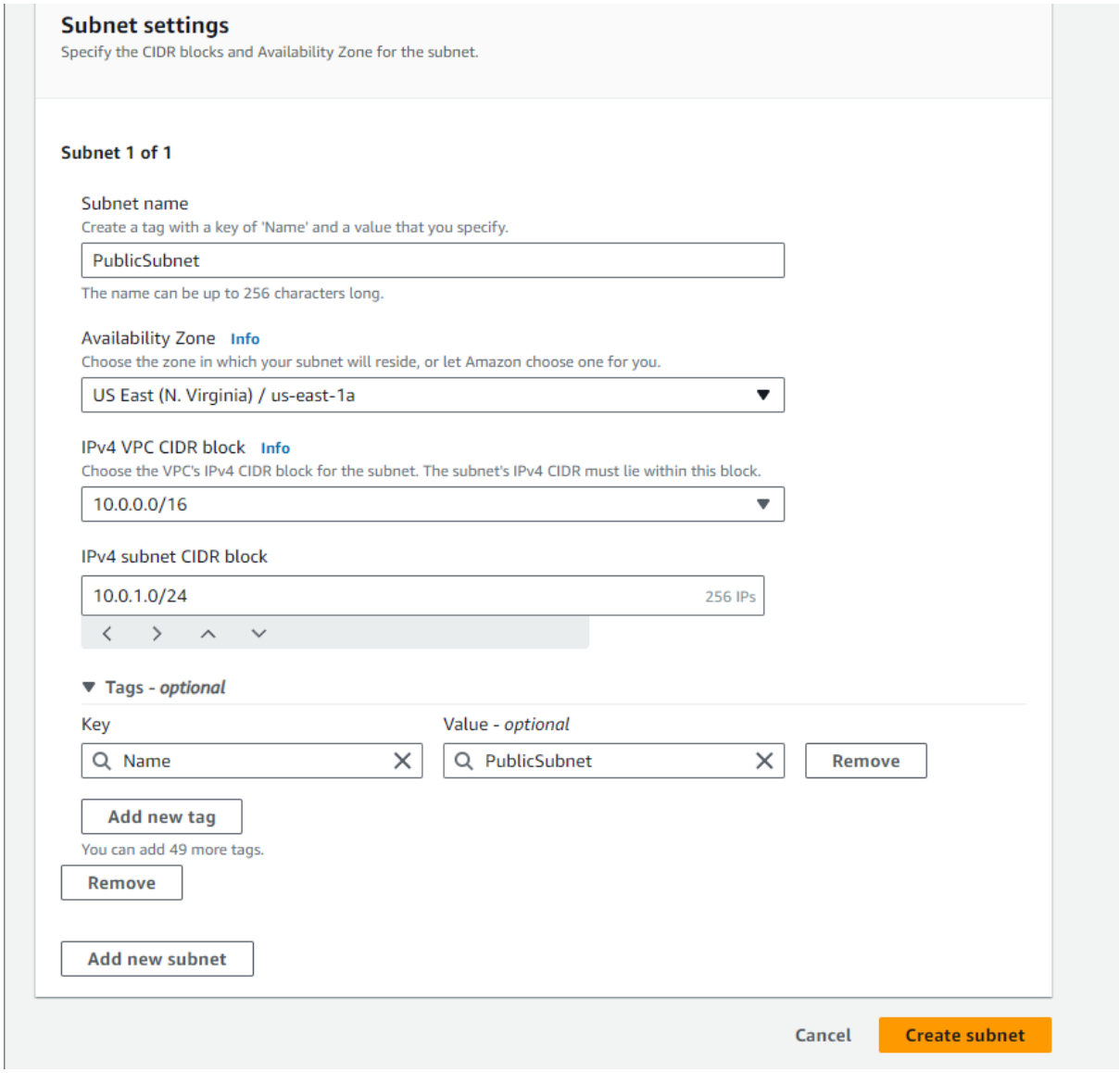
Creating a VPC establishes an isolated network within AWS where you can launch AWS resources.

2. **Create Subnets**:

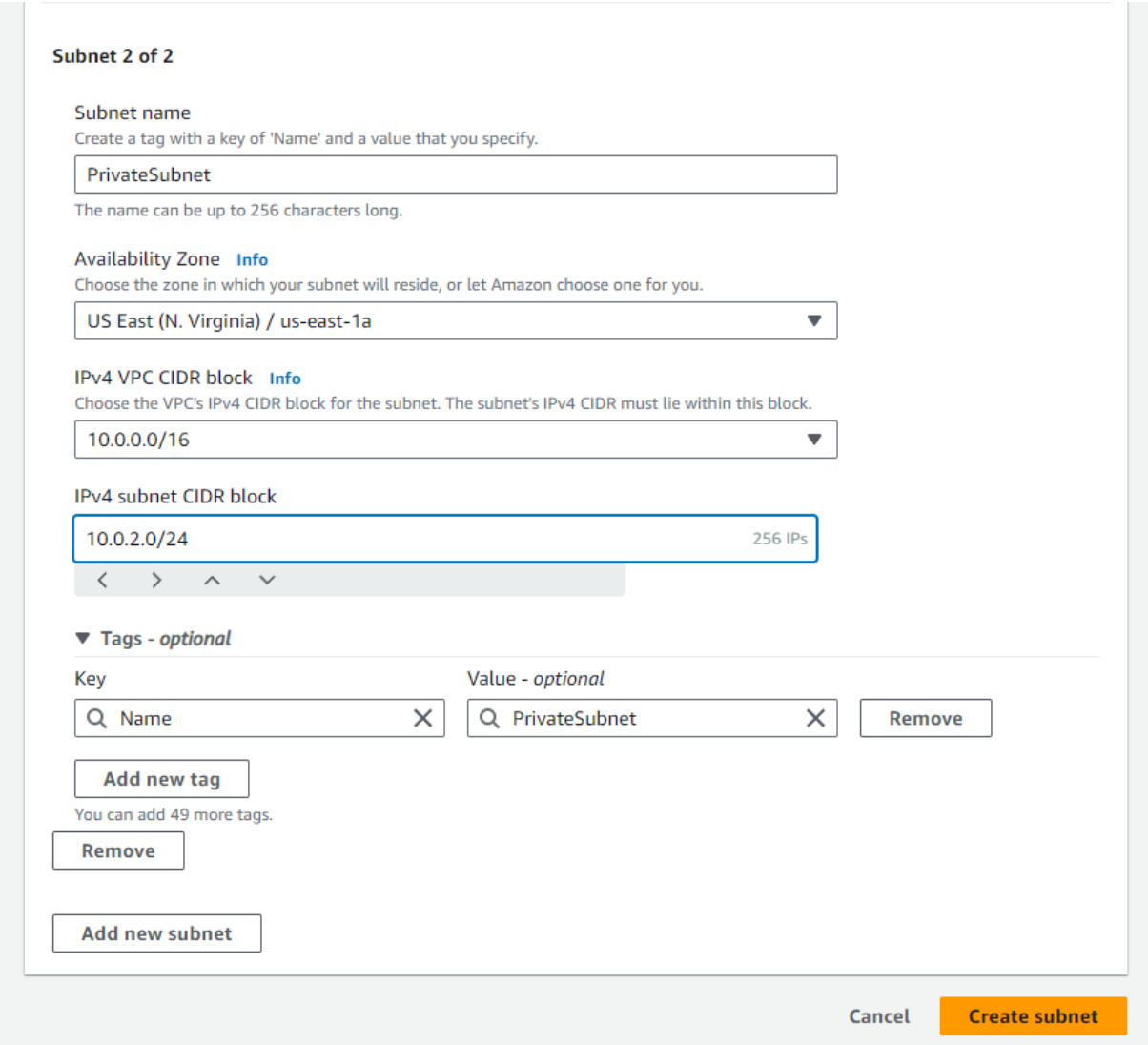
- Navigate to Subnets in th VPC dashboard.



- 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



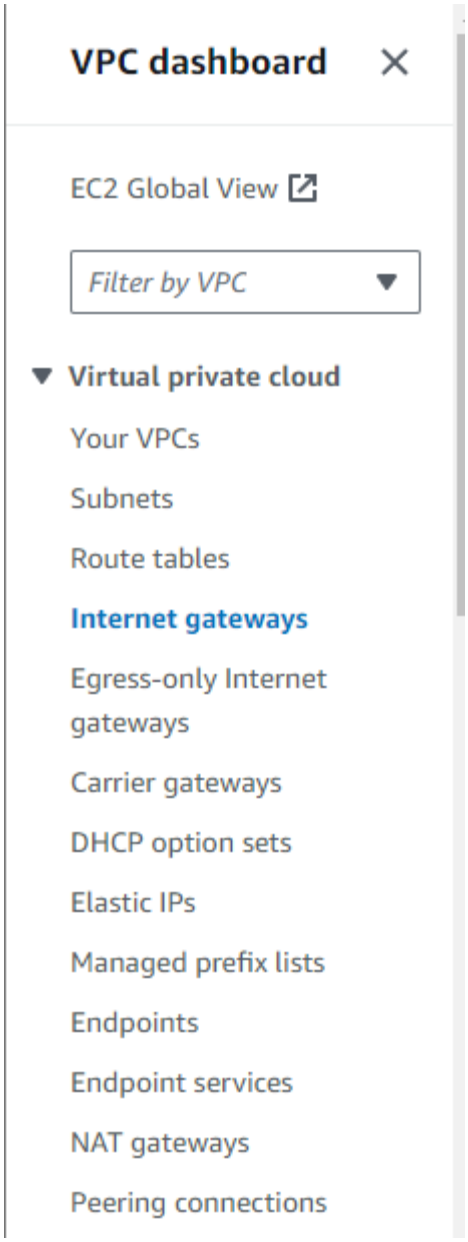
-
    -
        - **Add new subnet**
- **Private Subnet**: (Click on **Add new subnet**)
    - Name: PrivateSubnet
    - IPv4 subnet CIDR block: 10.0.2.0/24
    - Availability Zone: Same as the public subnet
    - Click Create.



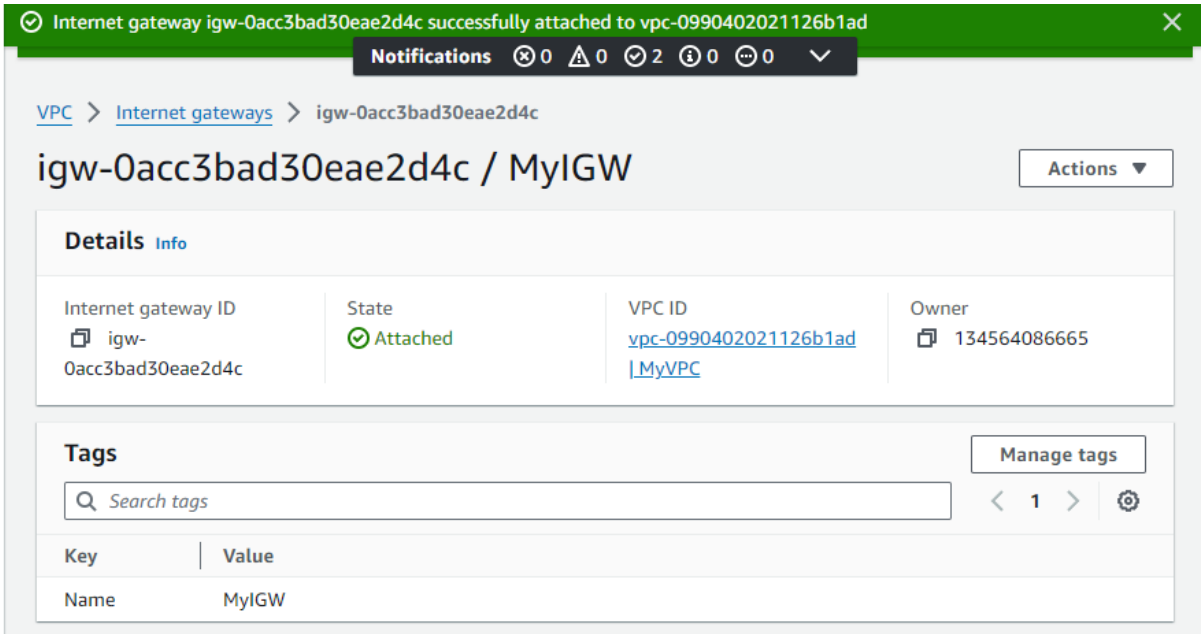
Subnets segment the VPC network. Public subnets allow direct access to the internet, while private subnets do not.

3. **Create an Internet Gateway**:

- Go to Internet Gateways in the VPC dashboard.



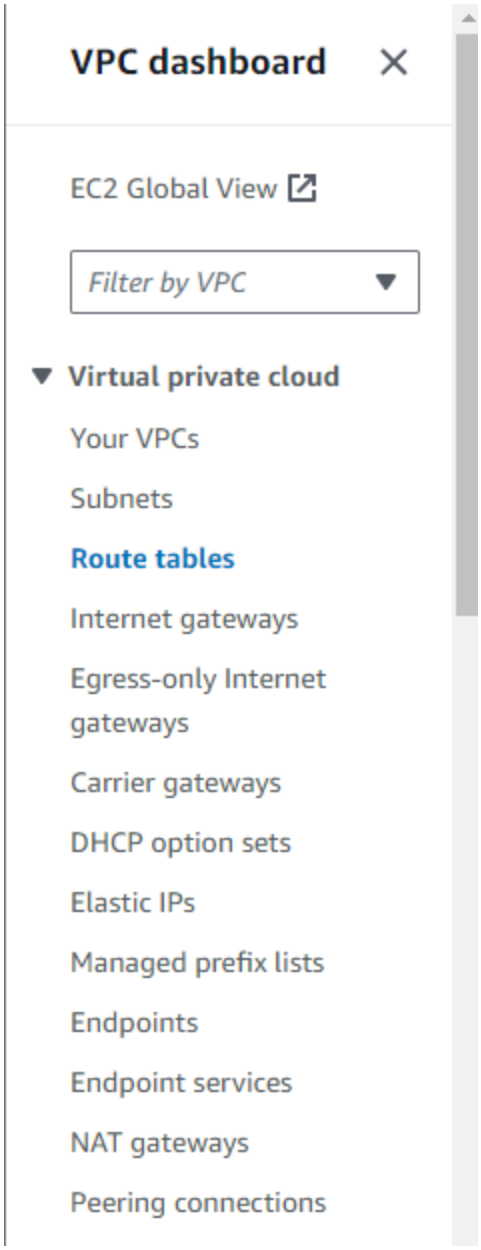
- Click **Create internet gateway**.
- Name: MyIGW
- Click Create.
- Attach the Internet Gateway to MyVPC.



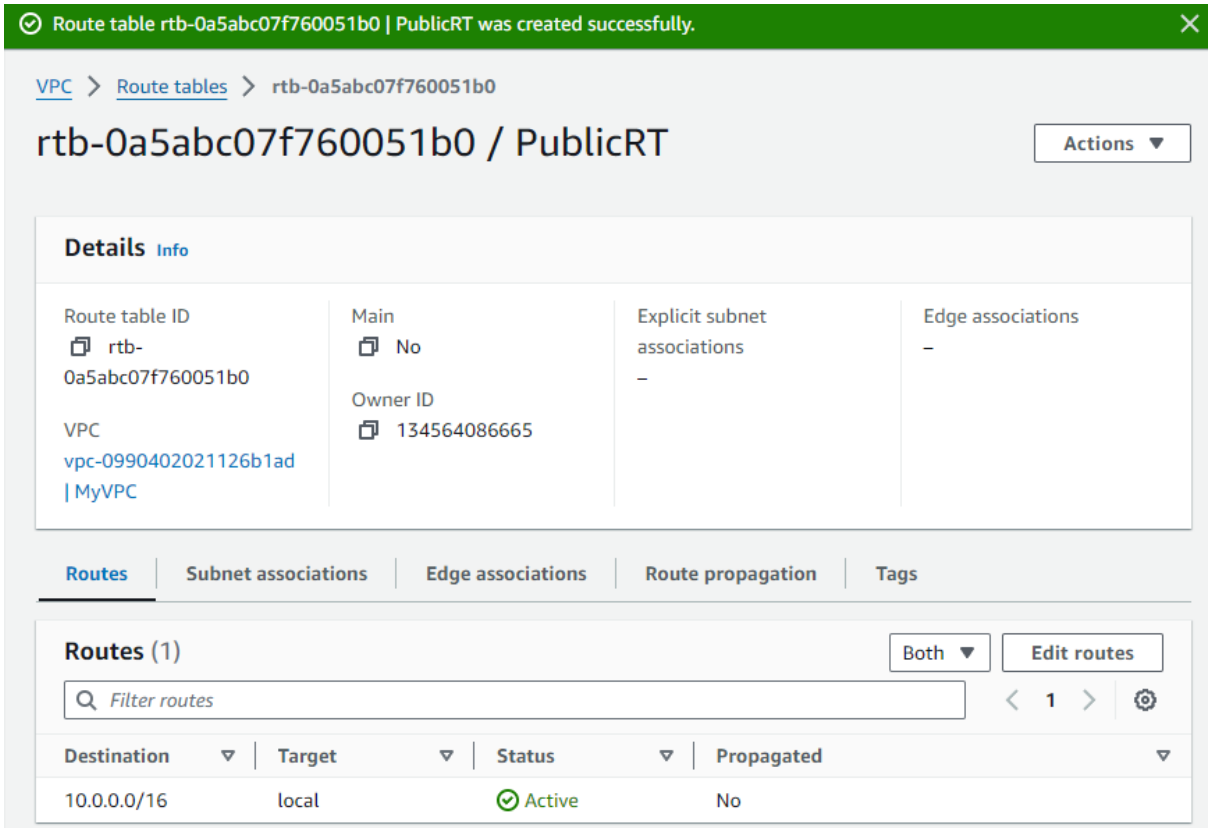
4. **Create a new Route Table for the Public Subnet**

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

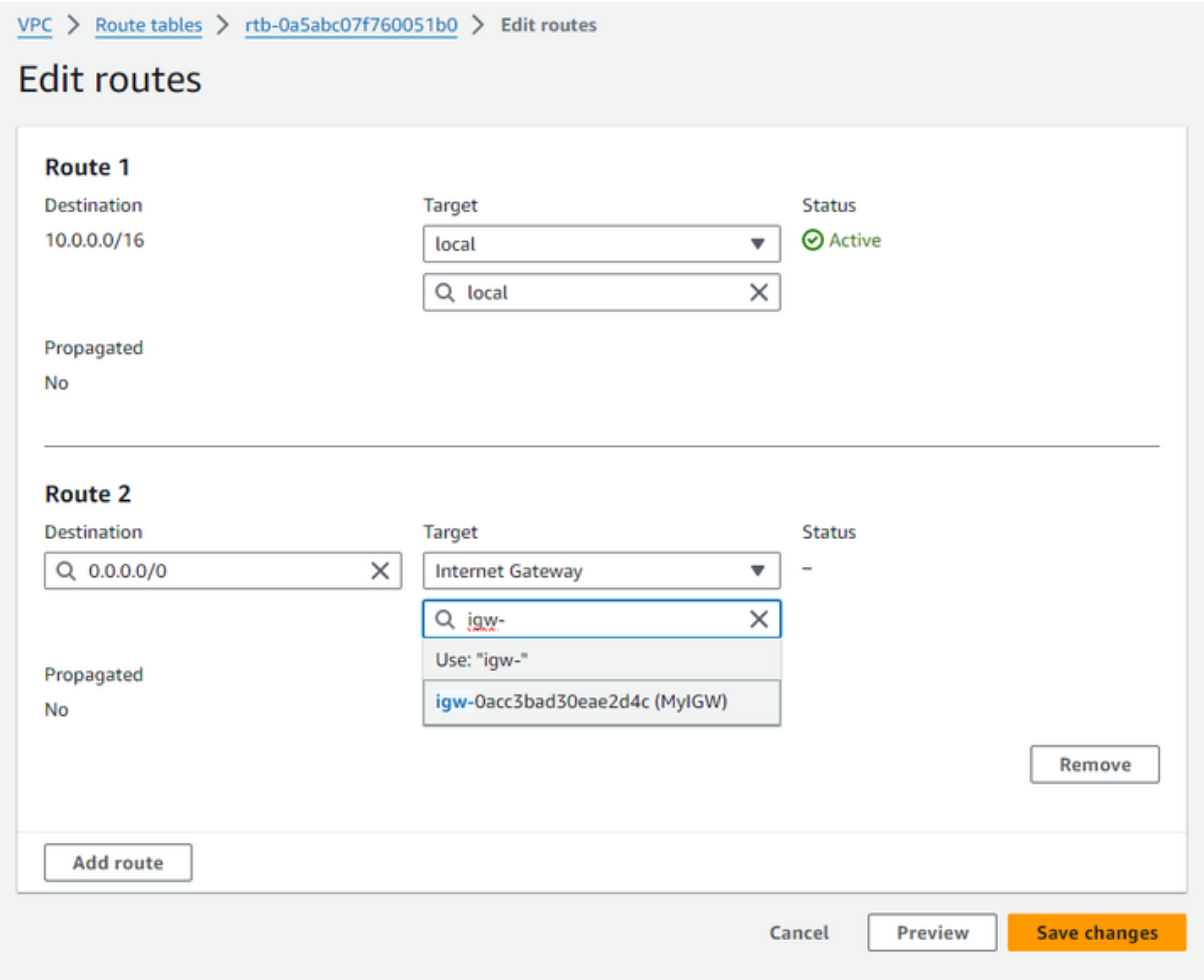
- Go to the Route Tables section in the VPC dashboard



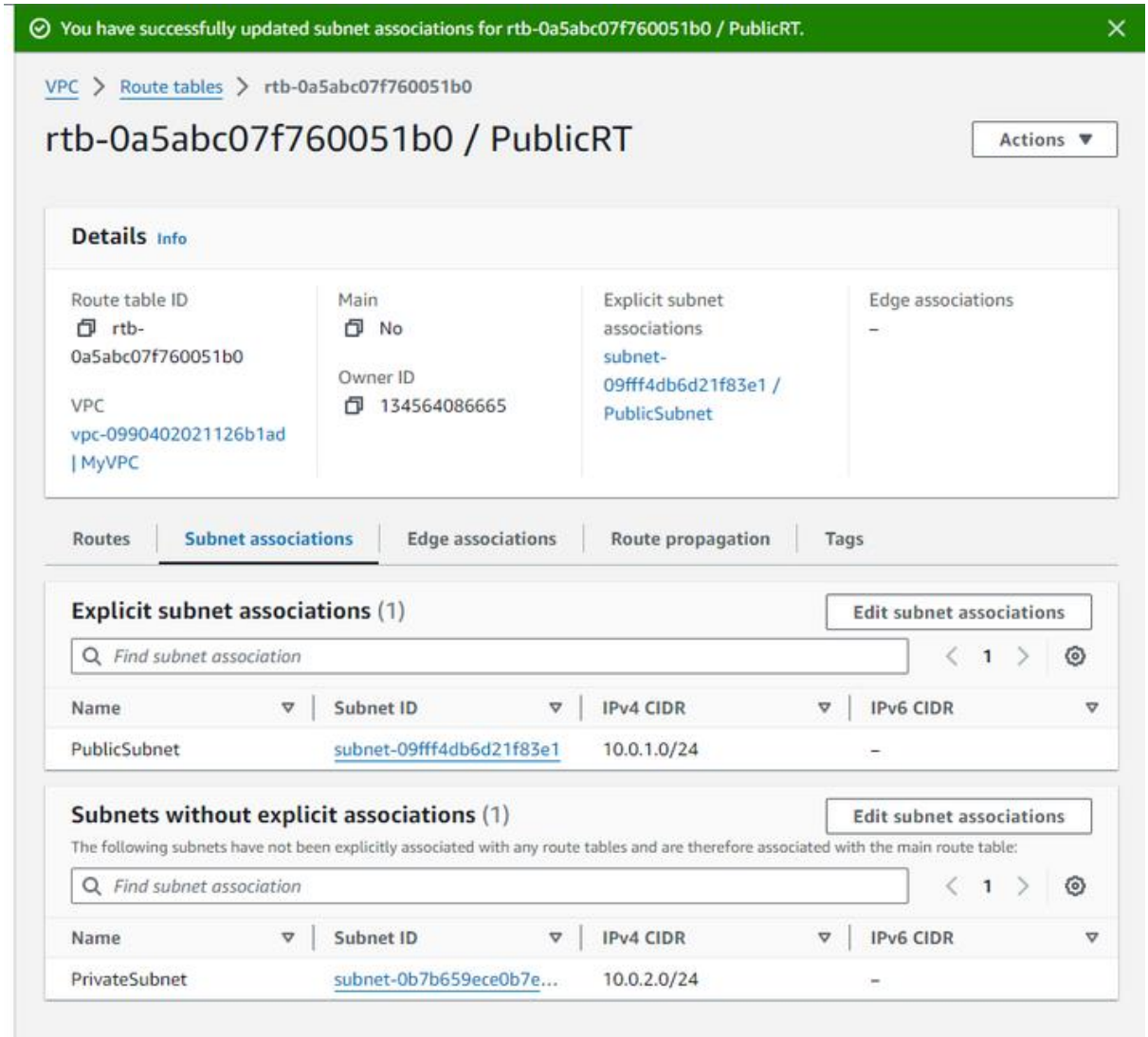
- Click **Create route table**
    - Name: PublicRT
    - VPC: MyVPC
- Click **Create route table**



- Edit routes:
    - Add route 0.0.0.0/0 to target MyIGW.

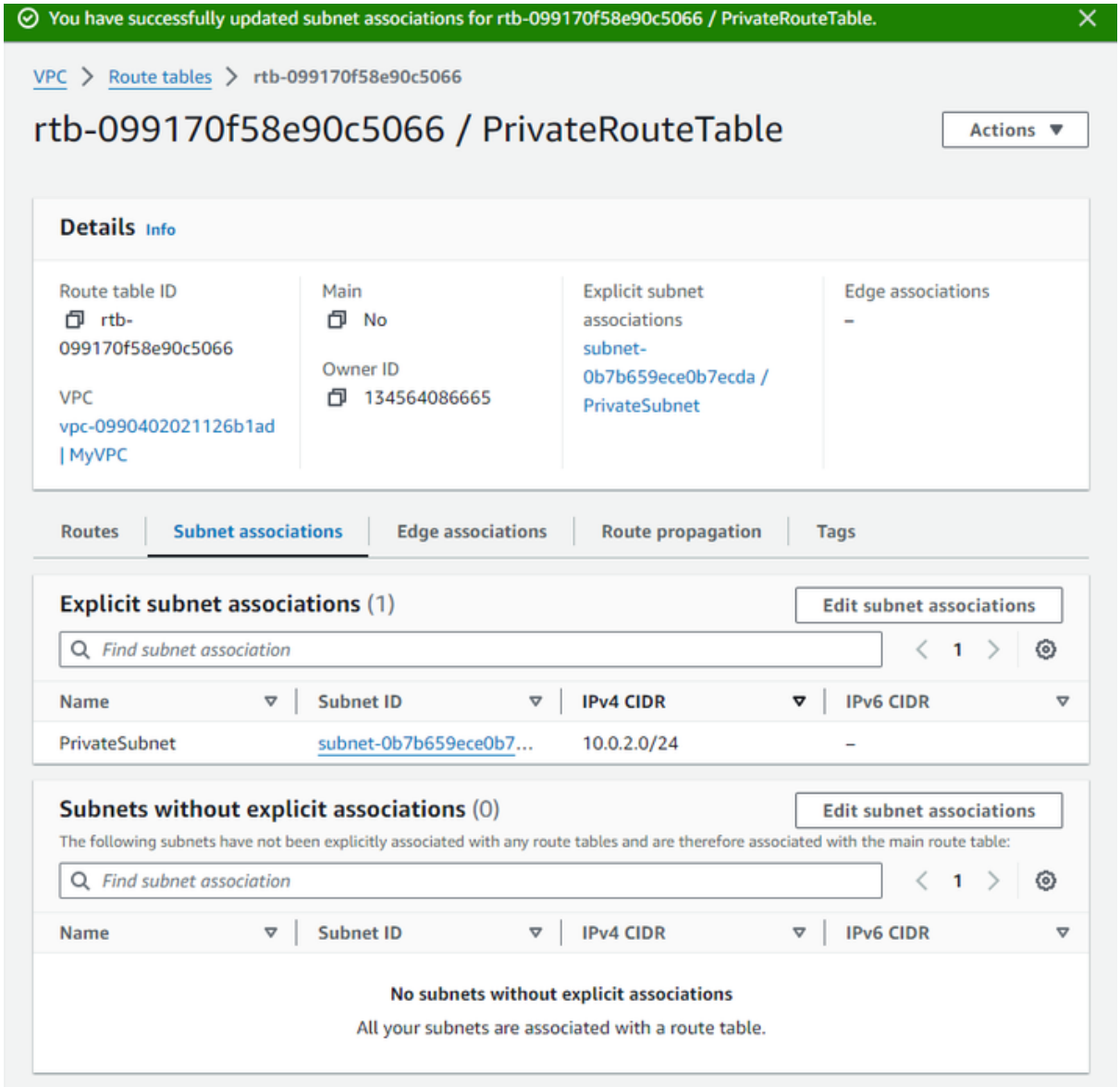


- Save changes.
- Edit **Explicit subnet associations** in the **Subnet associations tab**: Associate PublicSubnet.



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

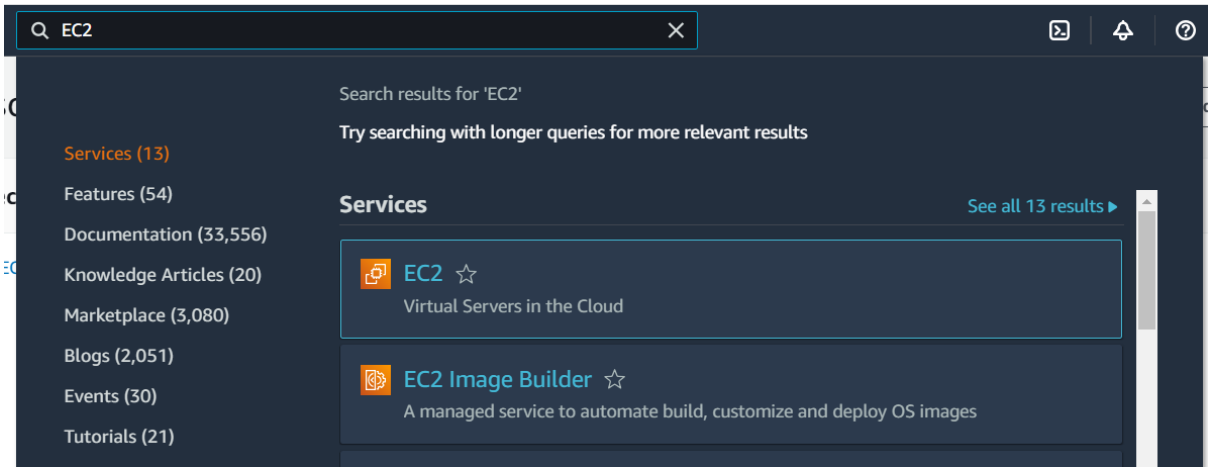
- Create another route table: Name: PrivateRouteTable, VPC: MyVPC.
- Edit Explicit subnet associations in the **Subnet associations tab**: Associate PrivateSubnet.



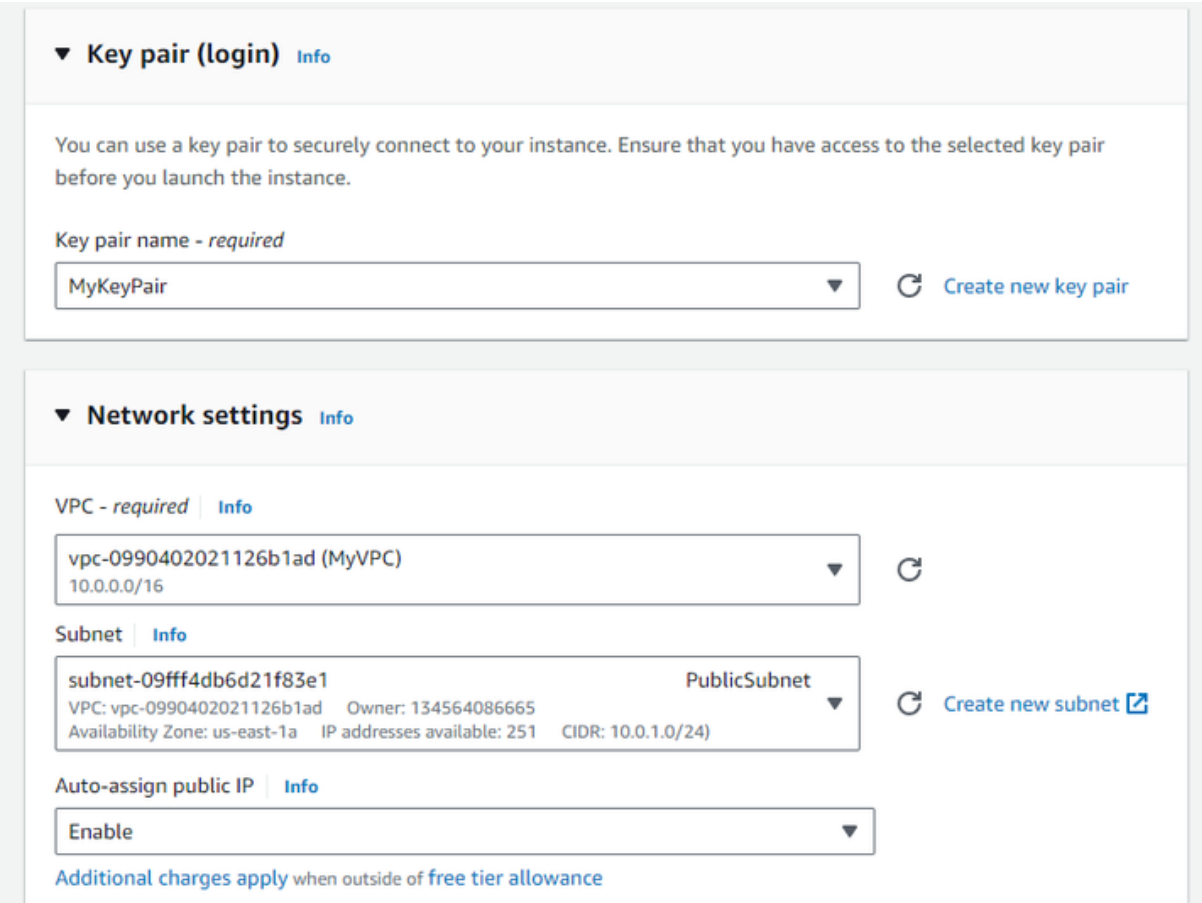
Launch Instances

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

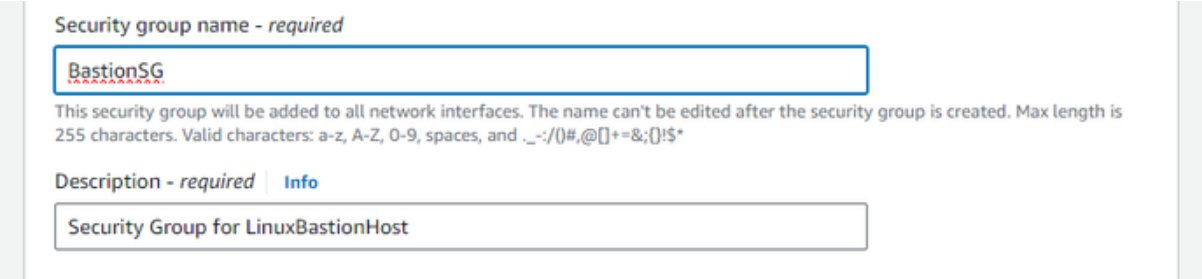
- Go to the EC2 dashboard.



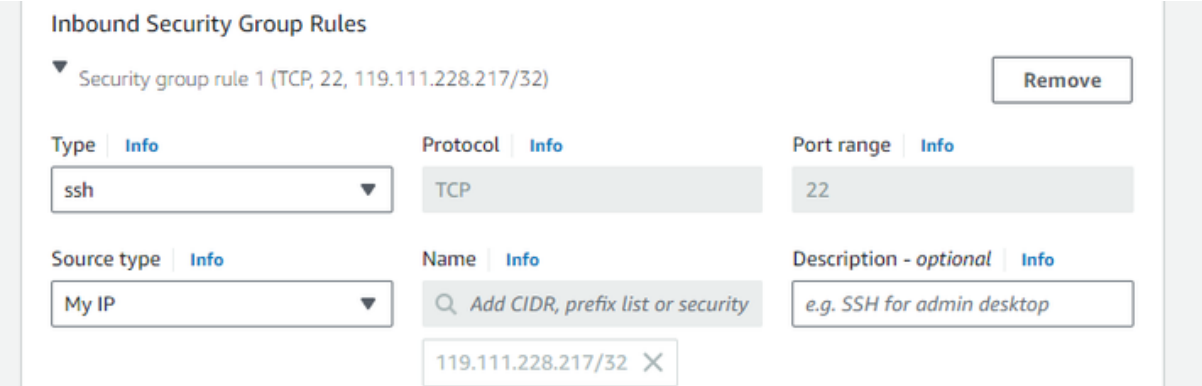
- Click Launch Instance.
- Name: LinuxBastionHost
- AMI: Amazon Linux 2023 AMI
- Instance Type: t2.micro
- Create key pair: MyKeyPair
- Network Settings:
    - VPC: MyVPC,
    - Subnet: PublicSubnet
- Enable Auto-assign Public IP.



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



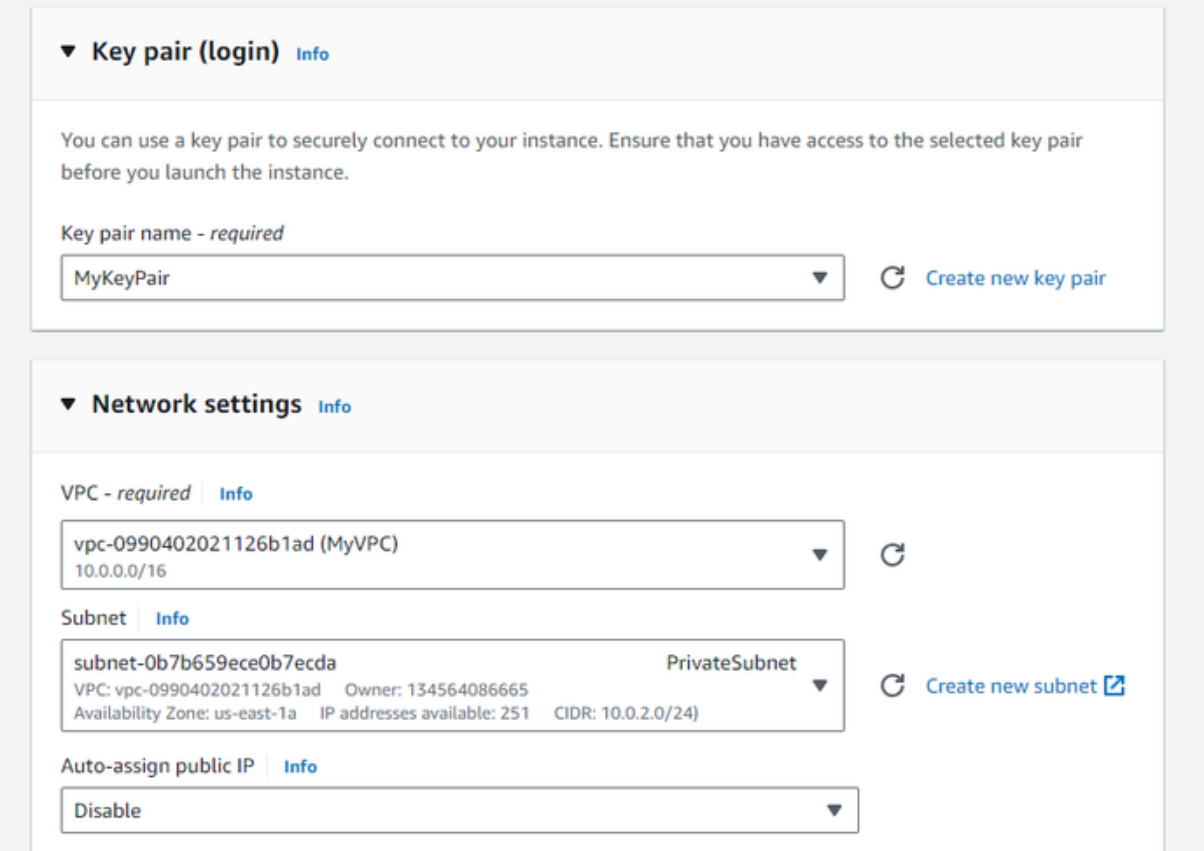
- Configure Inbound Security Group rules: Allow SSH (port **22**) from your IP



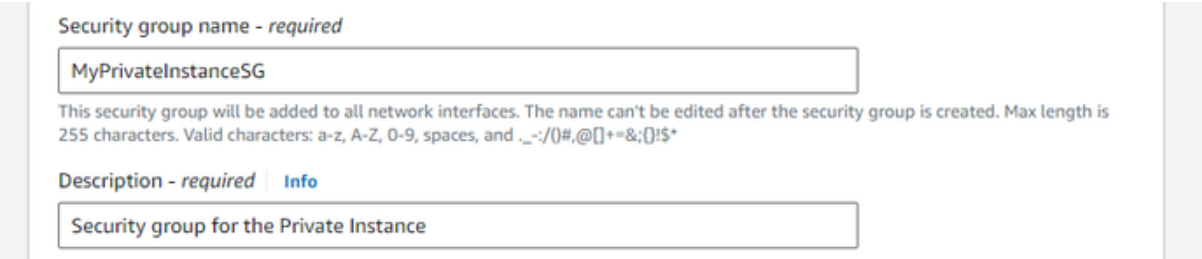
- Review and Launch.

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

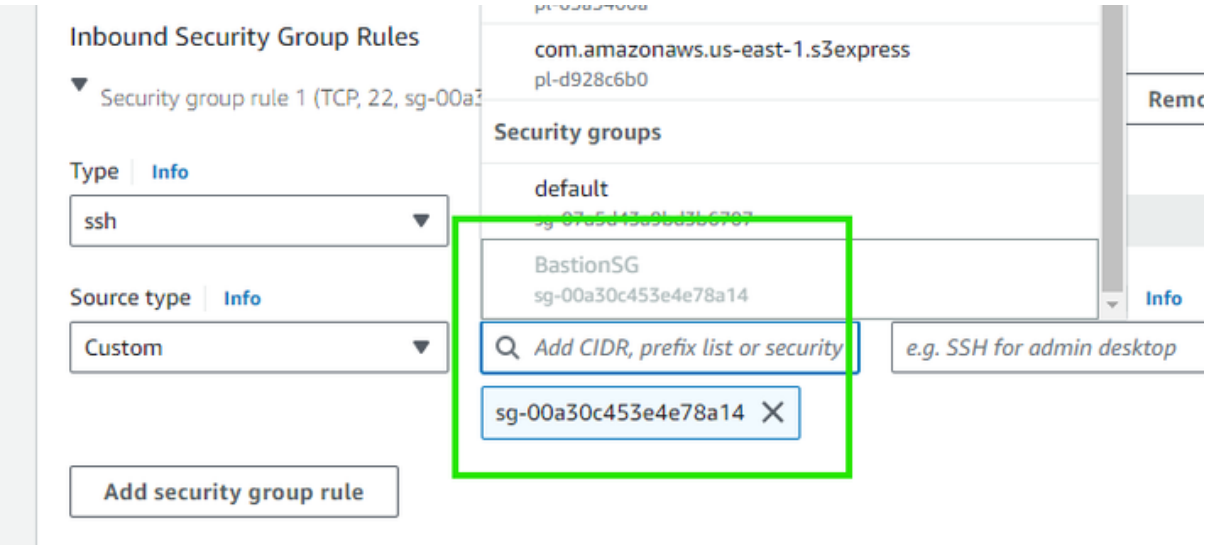
- Name: MyPrivateInstance
- Follow the same steps, but
    - select PrivateSubnet
    - Key pair: use the same key pair MyKeyPair
    - Network Settings:
        - VPC: MyVPC
        - Subnet: PrivateSubnet
    - Disable Auto-assign Public IP.



- 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 BastionSG.



- Review and Launch.

Security groups control the inbound and outbound traffic to instances. The Bastion host must have SSH access to the private instance.

Access the PrivateInstnace

1. Open up your terminal (e.g., Git Bash, PuTTY).

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

- Going to directory of your .pem key

- Copy, edit, and paste the following command accordingly:

Do not forgert to change the <your_NAT_Instance_Key_Pair>, <Your_Private_Instance_Key_Pair>, and <public_IP_address_of_NAT_Instance> placholders with the right value
If confirmation pops up in your terminal, type yes and ENTER

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@44.200.61.173:/home/ec2-user/
The authenticity of host '44.200.61.173 (44.200.61.173)' can't be established.
ED25519 key fingerprint is SHA256:uC8naihPonhF5NiA9Te4VMfix69Z5xP9bxipruzwsF8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '44.200.61.173' (ED25519) to the list of known hosts.
MyKeyPair.pem                                 100% 1674     7.6KB/s   00:00

neil@Sol MINGW64 ~/Downloads
$
```

3. SSH to the NATInstance

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

```
neil@Sol MINGW64 ~/Downloads
$ ssh -i "MyKeyPair.pem" ec2-user@44.200.61.173
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-235 ~]$
```

4. Let's check first if the .pem key was successfully copied using the ls -l command.

```
[ec2-user@ip-10-0-1-235 ~]$ ls -l
total 4
-rw-r--r--. 1 ec2-user ec2-user 1674 Aug  6 06:22 MyKeyPair.pem
[ec2-user@ip-10-0-1-235 ~]$
```

5. Use the following command to add permission to the key

sudo chmod 400 MyKeyPair.pem

```
[ec2-user@ip-10-0-1-235 ~]$ sudo chmod 400 MyKeyPair.pem
[ec2-user@ip-10-0-1-235 ~]$
```

6. SSH into the Private instance:

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

A prompt confirmation will pop to the terminal, answer yes

```
[ec2-user@ip-10-0-1-235 ~]$ ssh -i MyKeyPair.pem ec2-user@10.0.2.171
The authenticity of host '10.0.2.171 (10.0.2.171)' can't be established.
ED25519 key fingerprint is SHA256:zDCs8685EraV1C6yiOQ8E2Qw9IgOmaHQ8toQG12sBiY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.171' (ED25519) to the list of known hosts.
       #_
       ~\_  ####_        Amazon Linux 2023
      ~~  \_#####\
      ~~     \###|
      ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
       ~~       V~' '->
        ~~~         /
          ~~._.   _/
             _/ _/
           _/m/'
[ec2-user@ip-10-0-2-171 ~]$
```

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

Using the Bastion host as an intermediate jump box, you can securely access instances in the private subnet.

That's it! Congratulations on successfully setting up a Linux Bastion host to securely access instances in a private subnet.

By completing this lab, you have learned how to establish a secure access point for managing instances within a private subnet. This ensures that your private instances remain protected from direct internet exposure while still being accessible for necessary management and maintenance tasks. Configuring a Bastion host is an essential skill for maintaining secure and efficient network operations in AWS.