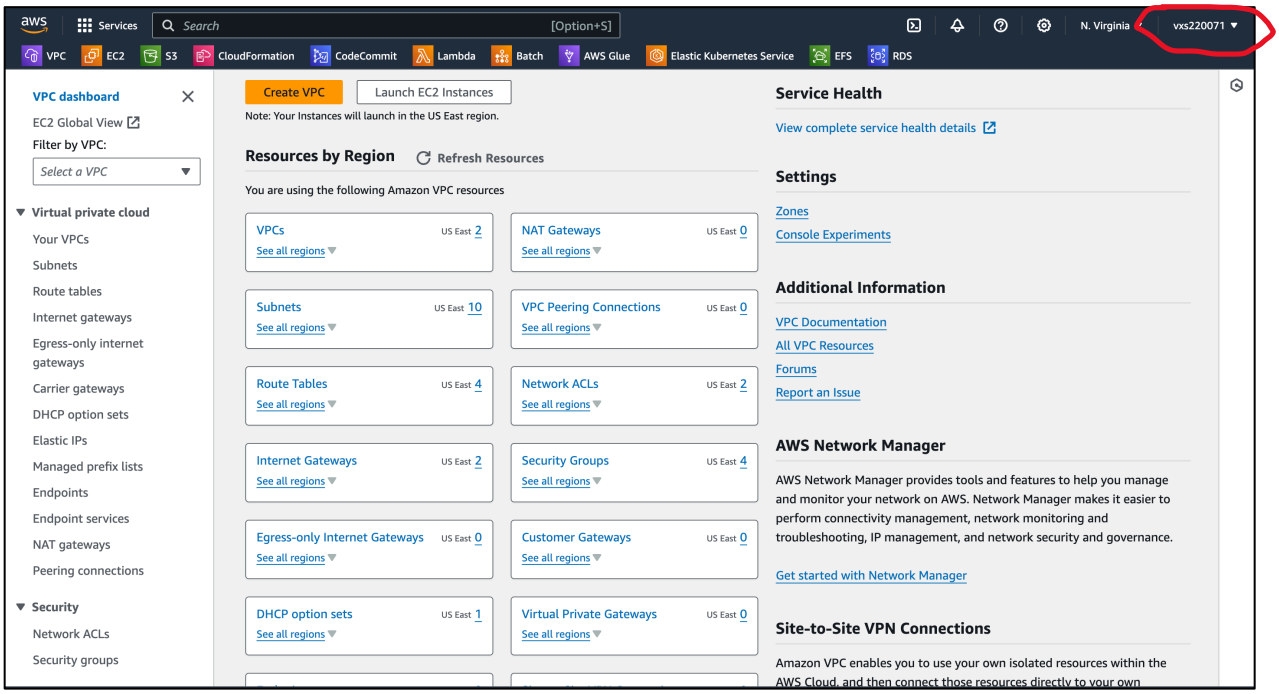
VPC Networking Lab

How do you document this lab?

1. Capture the full screenshot of the AWS console; this should contain your name on the screenshot.

Ex:



2. Make sure you do your labs in the North Virginia region; this is not mandatory but is advice. This will help troubleshoot a lot of things, including your billing.

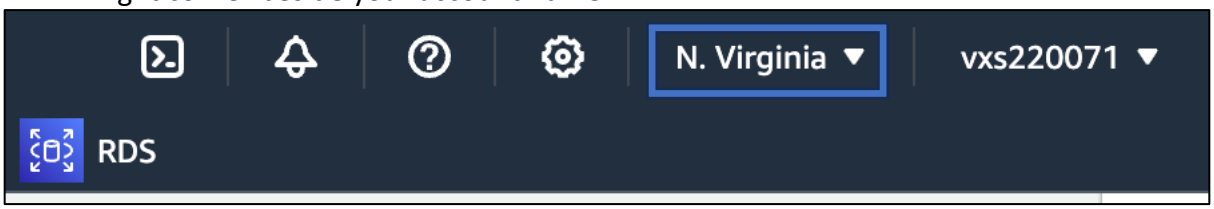
Deliverables: Submit the screenshots where it mentions "Note: This is a Deliverable." The screenshots should be submitted in the pdf format.

Step 5 – f
Step 6 – a
Step 7 – a
Step 9 – c and d

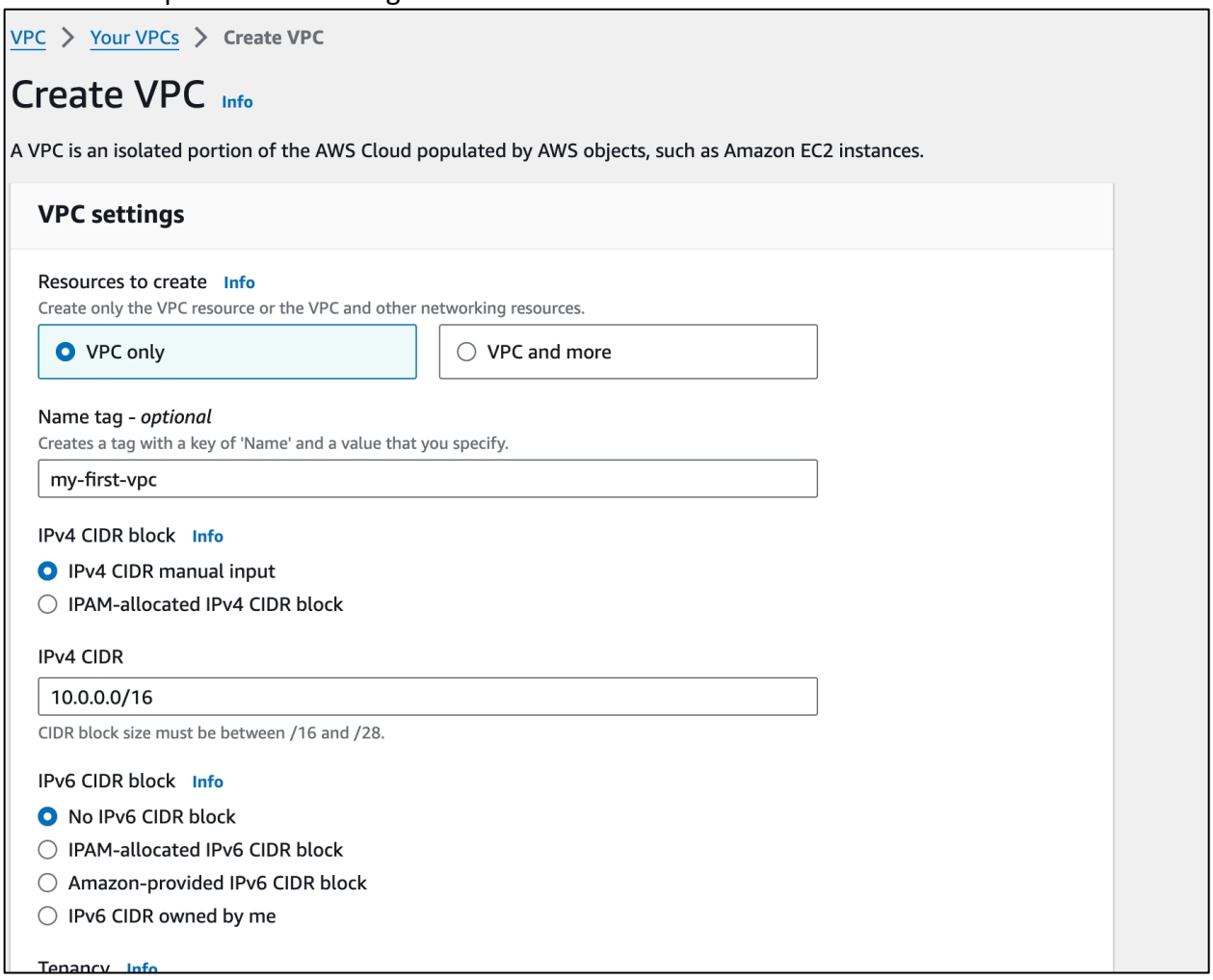
Table of Contents:

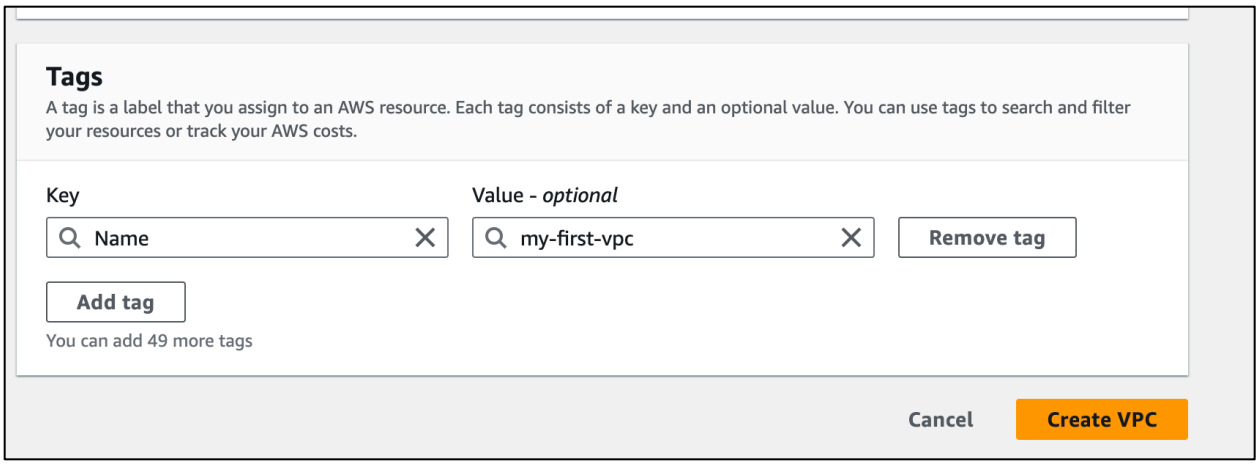
Step 1: Create VPC

a. Login to AWS Console and set the region to N.Virginia. This option is available at the top right corner beside your account name.



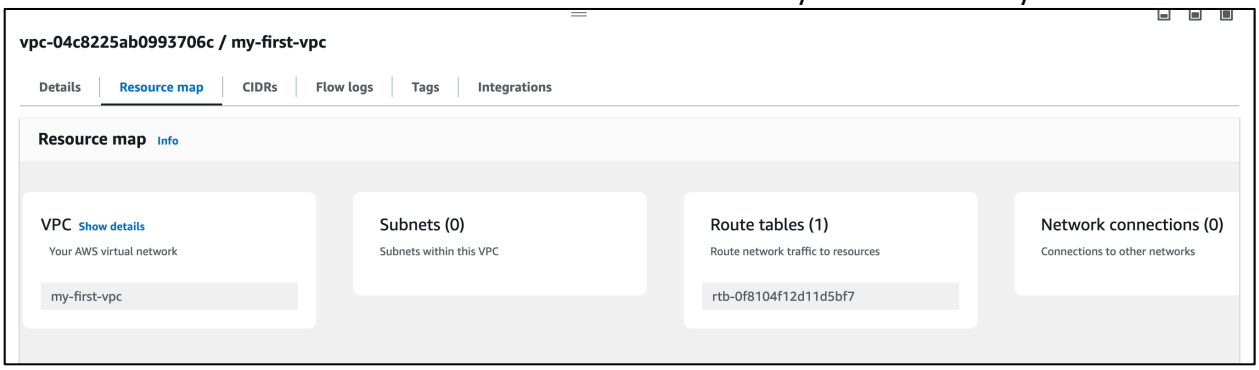
b. Now, search for VPC service and click on **Create VPC**. Choose/fill out the options as given below and Create the VPC. Note: Understand each column in detail. Understand that VPC spans the entire region.



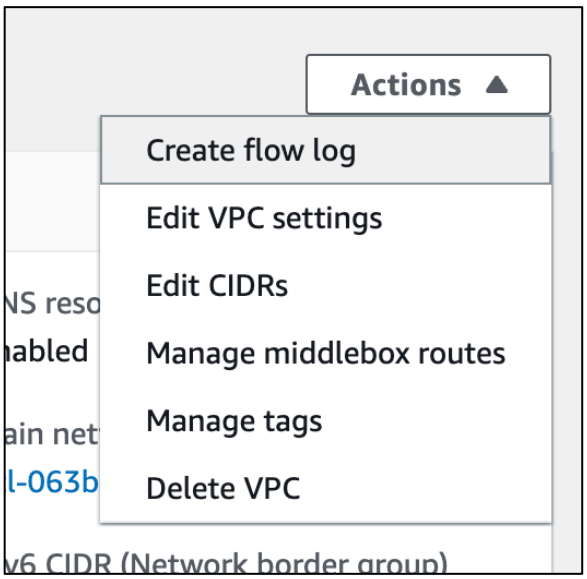


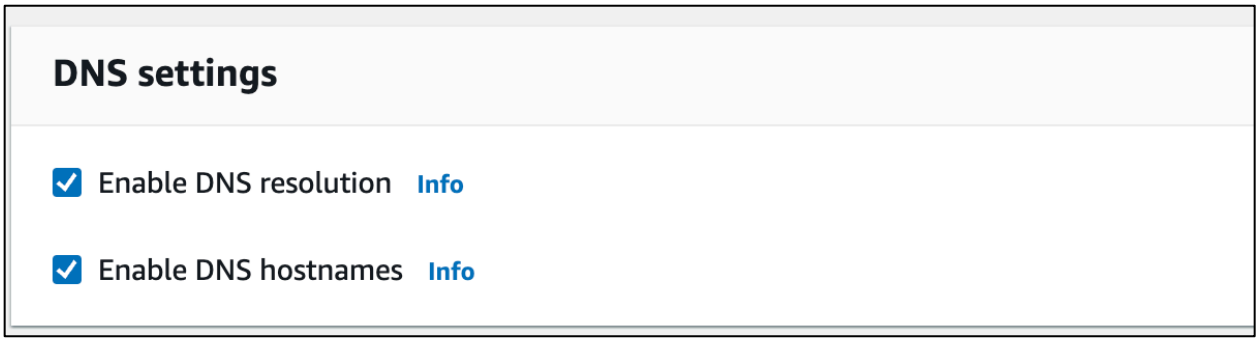
c. This is how it looks when you first create your VPC. Observe the resource map as we move forward while creating each resource in VPC.

You have to understand that a route table is created by default for every VPC created.



d. Enable DNS Hostnames: Click on **Edit VPC settings**, make the changes as shown below.



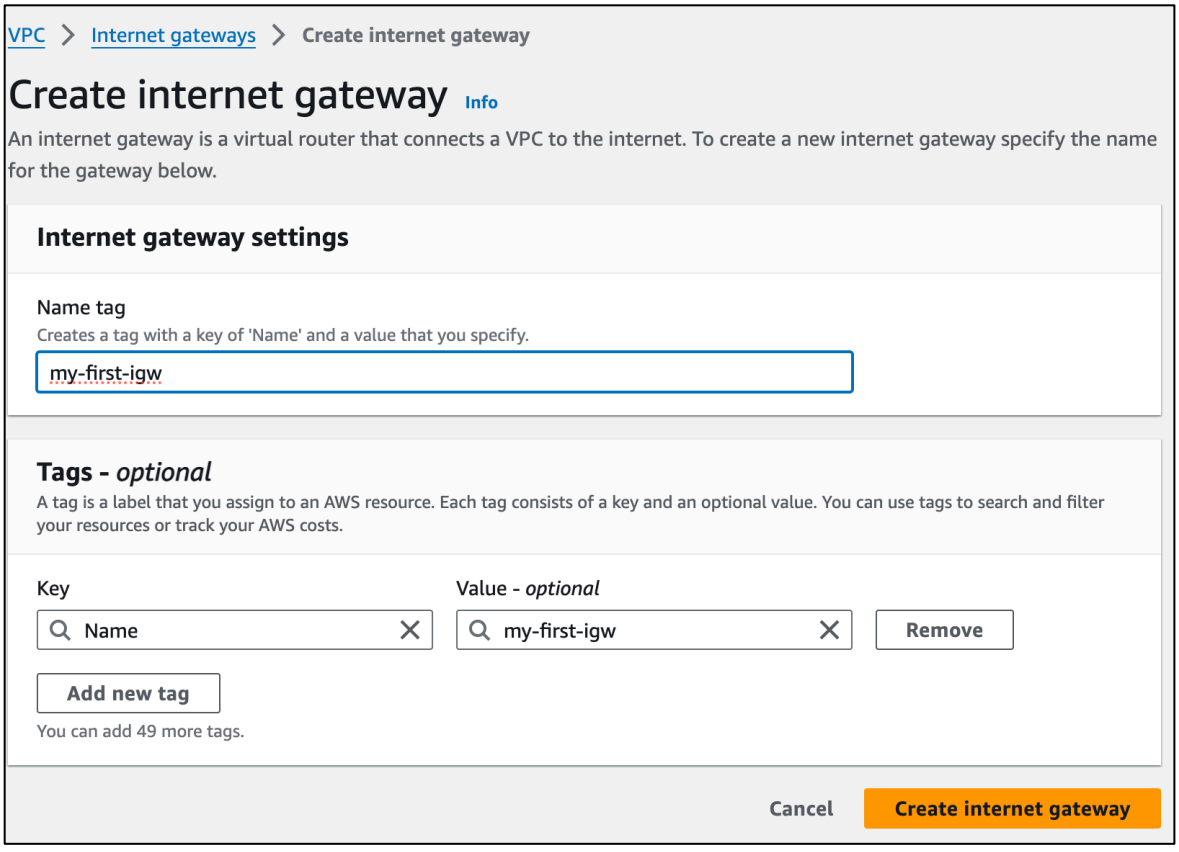


Step 2: Create an Internet Gateway and Attach it to VPC.

a. Now, Let's create an Internet Gateway.

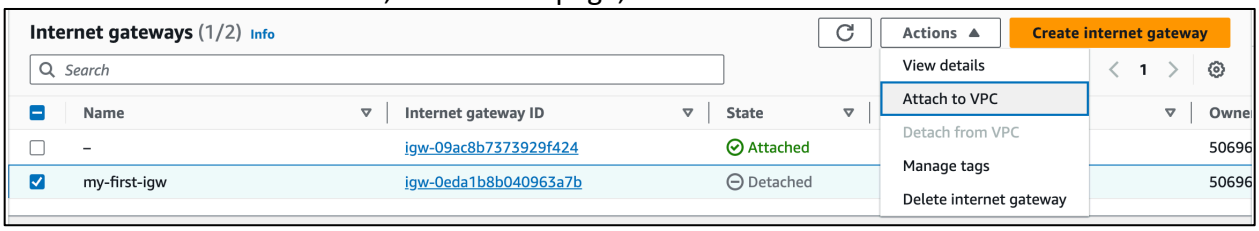
   What is an Internet Gateway? It's simply a virtual device to connect to the Internet. So, let's say we have a server created in a VPC. How do you connect to the internet from that server? That's where Internet Gateway comes into the picture. It's pretty similar to the Spectrum Wi-Fi, right? (Analogy)

b. Go to the VPC page and look out for Internet Gateway. Create One as shown below.

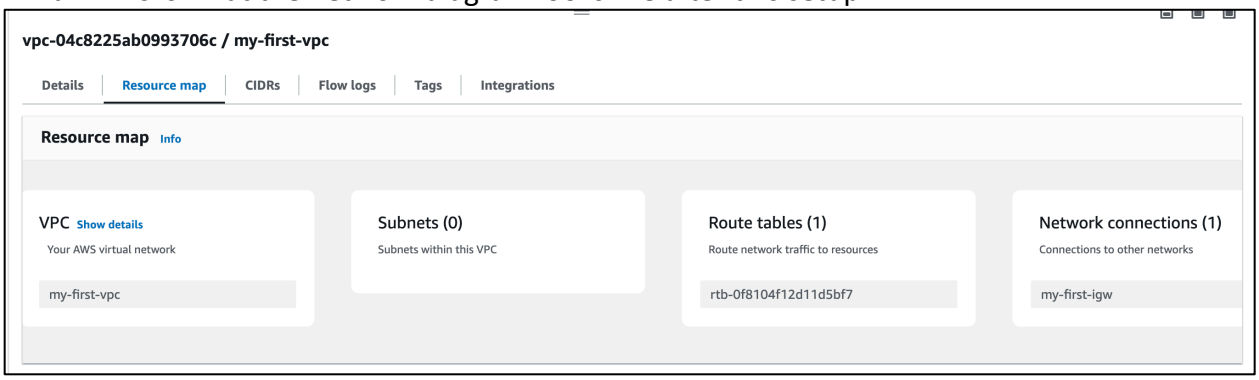


c. Now that we have created an IGW= Internet Gateway, we must attach it to the VPC, just like we link our router to an Internet Port at home.

Click on Attach to VPC; on the next page, select the VPC created and attach it.



d. This is what the network diagram looks like after this setup.

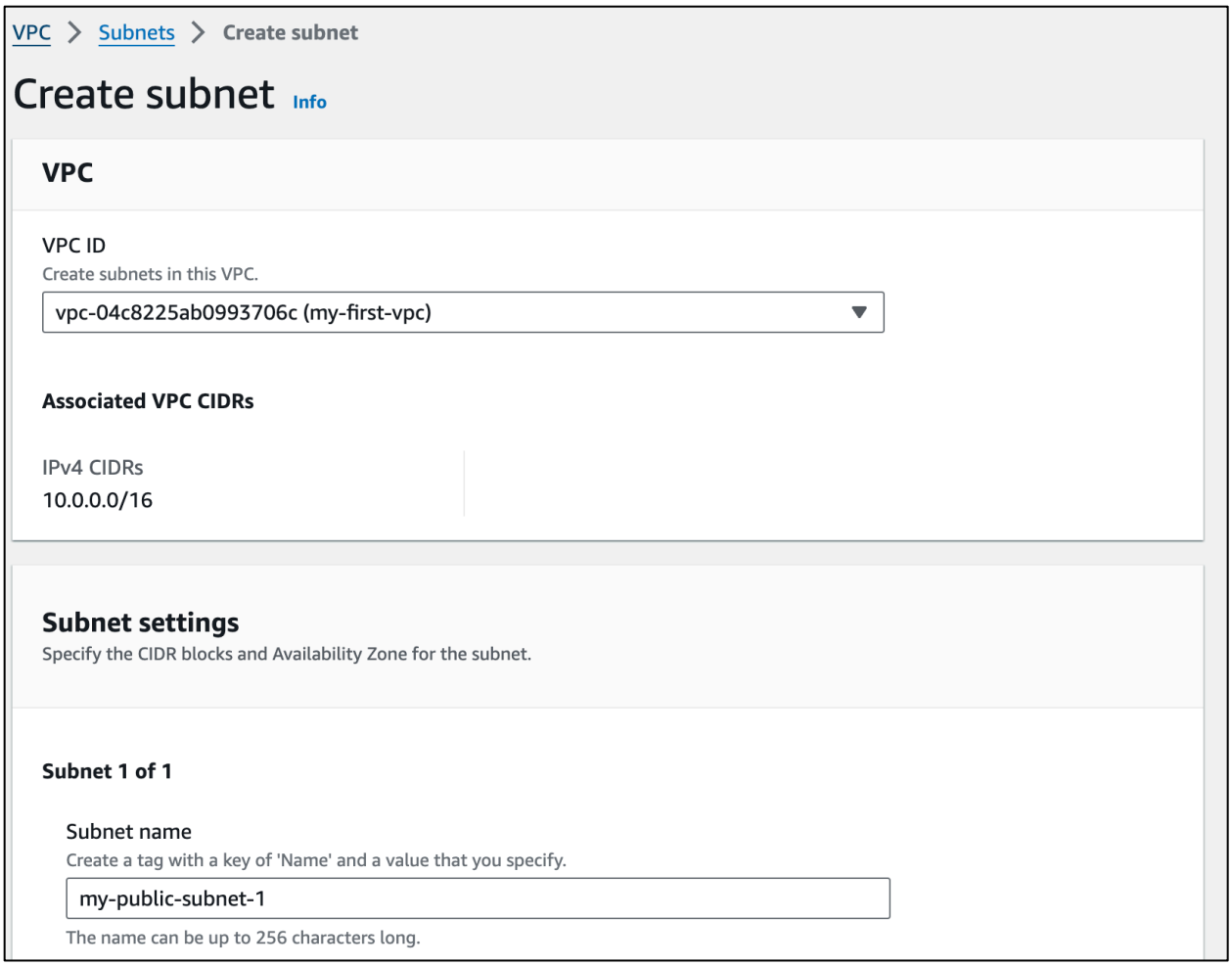


Step 3: Create Public and Private Subnets.

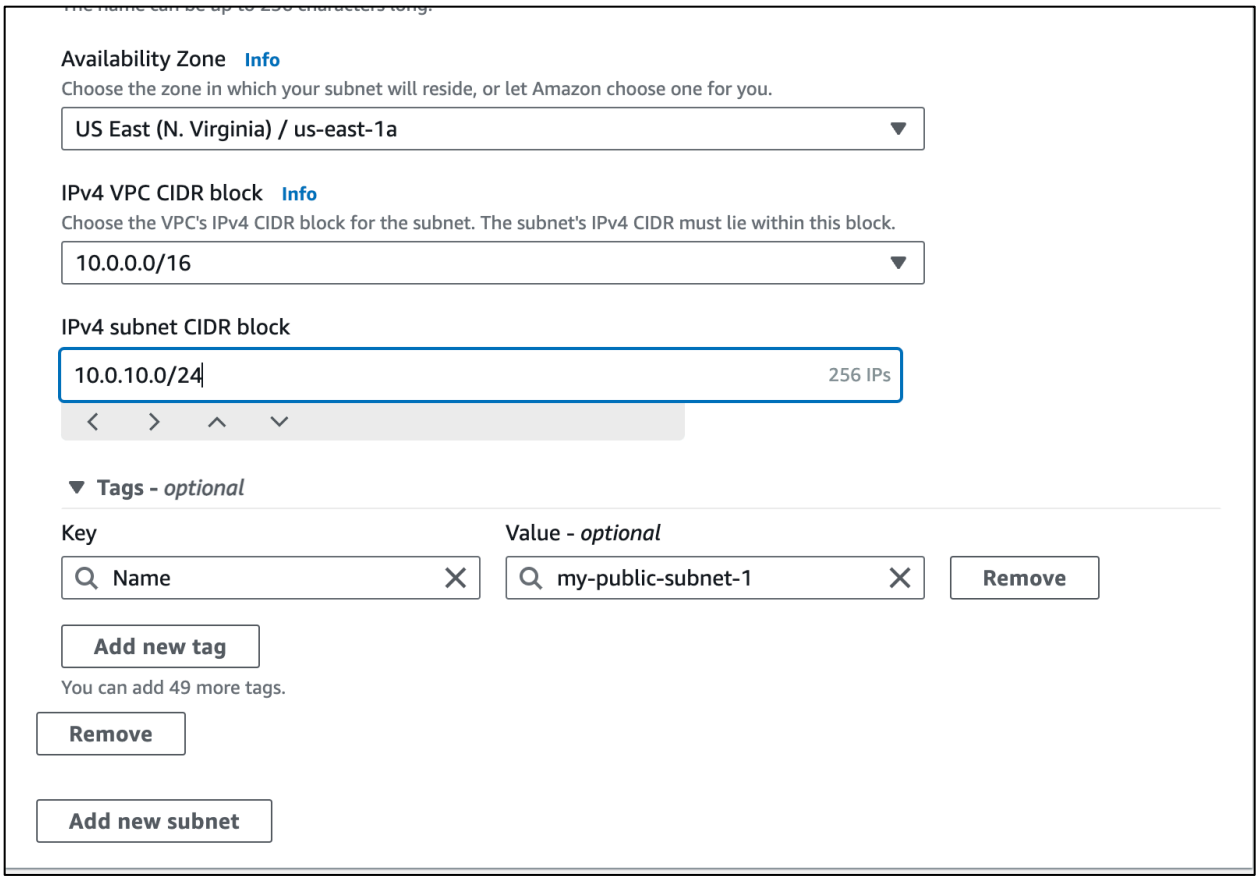
a. Let's divide our VPC network into public and private subnets.
Example: Think of an office building with different departments like HR, Finance, and IT, each occupying its section of the building. Similarly, subnets in AWS VPC segregate resources based on their function or purpose, ensuring that different parts of your cloud infrastructure remain isolated and secure.

b. We are creating two public and two private subnets.

c. Let's create Public Subnet 1

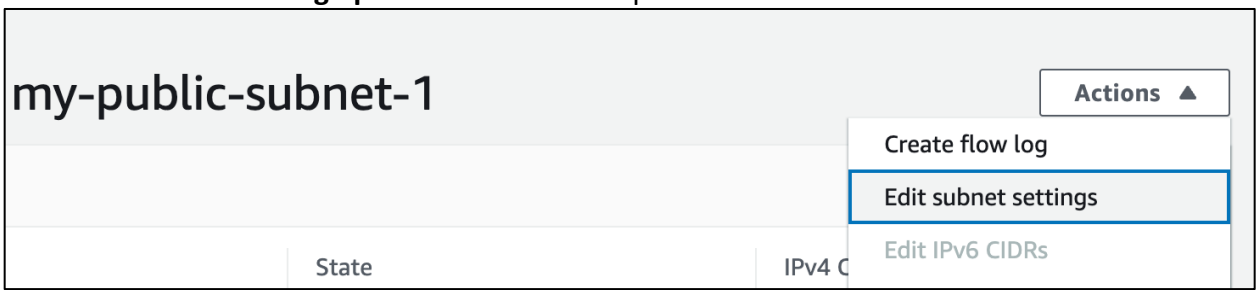


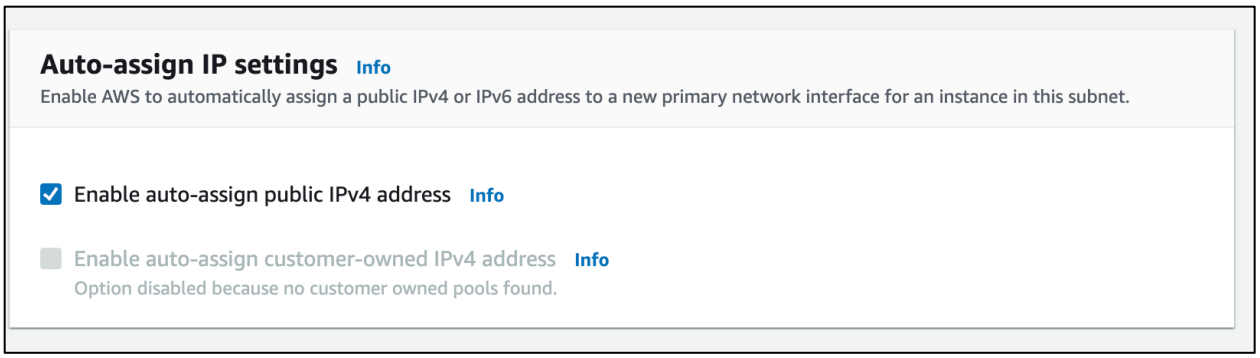
Beware of the availability zone chosen here.



d. Public Subnet 2: Your task is to create a public subnet with the name **my-public-subnet-2**, the CIDR block **10.0.2.0/24,** and the region is **us-east-1b.**

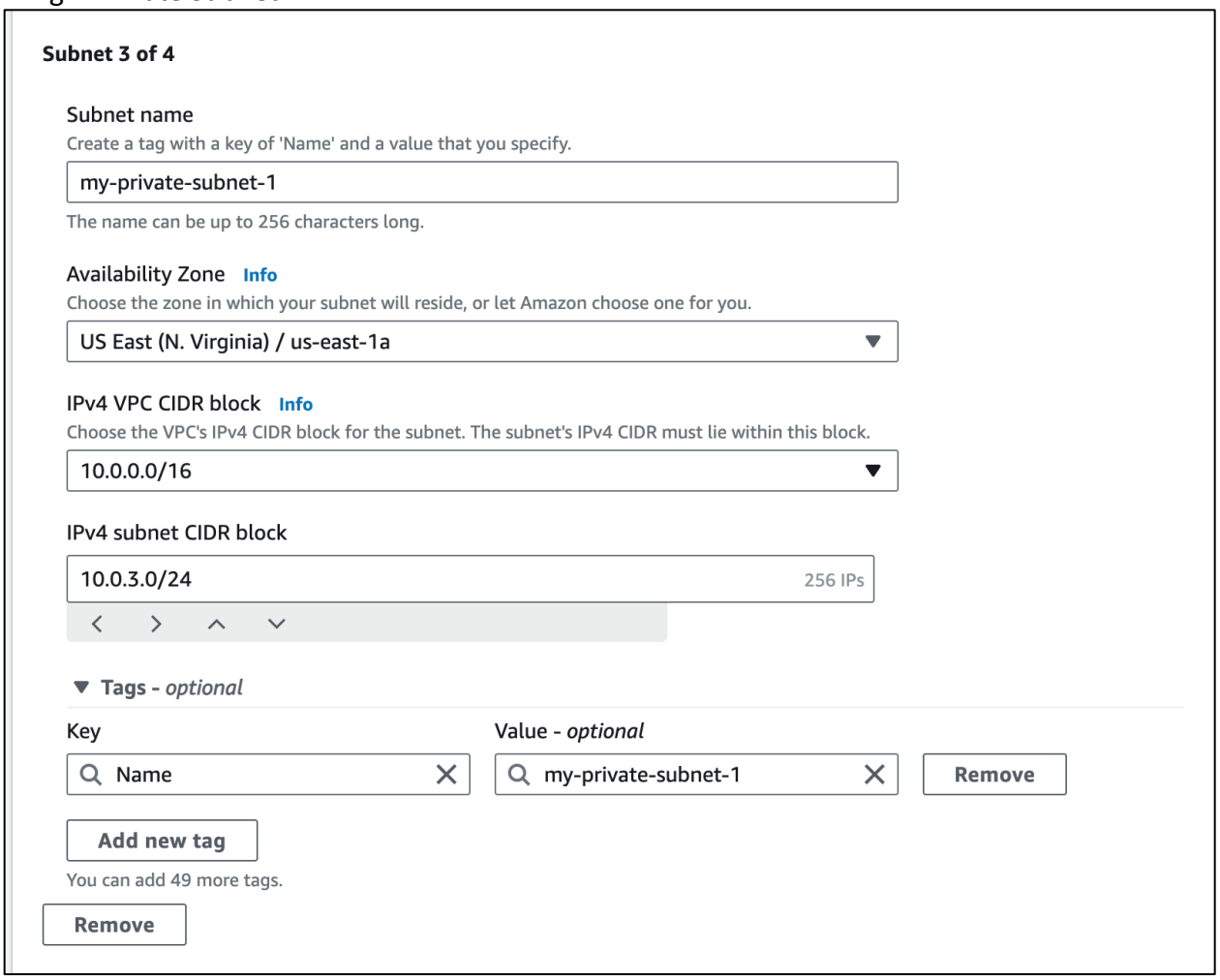
e. Enable **auto-assign public IPv4 address** option.





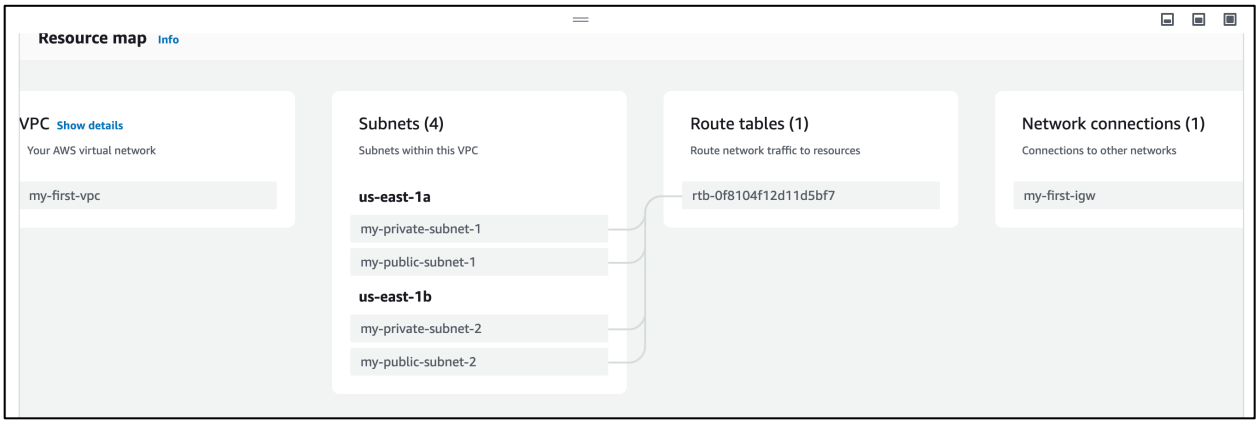
f. Do the above step for the second public subnet.

g. Private Subnet 1:



h. Private Subnet 2: Your task is to create a private subnet named **my-private-subnet-2**, the CIDR block **10.0.4.0/24,** and the region **is us-east-1b.**

i. After creating the subnets, my network map looks like this.

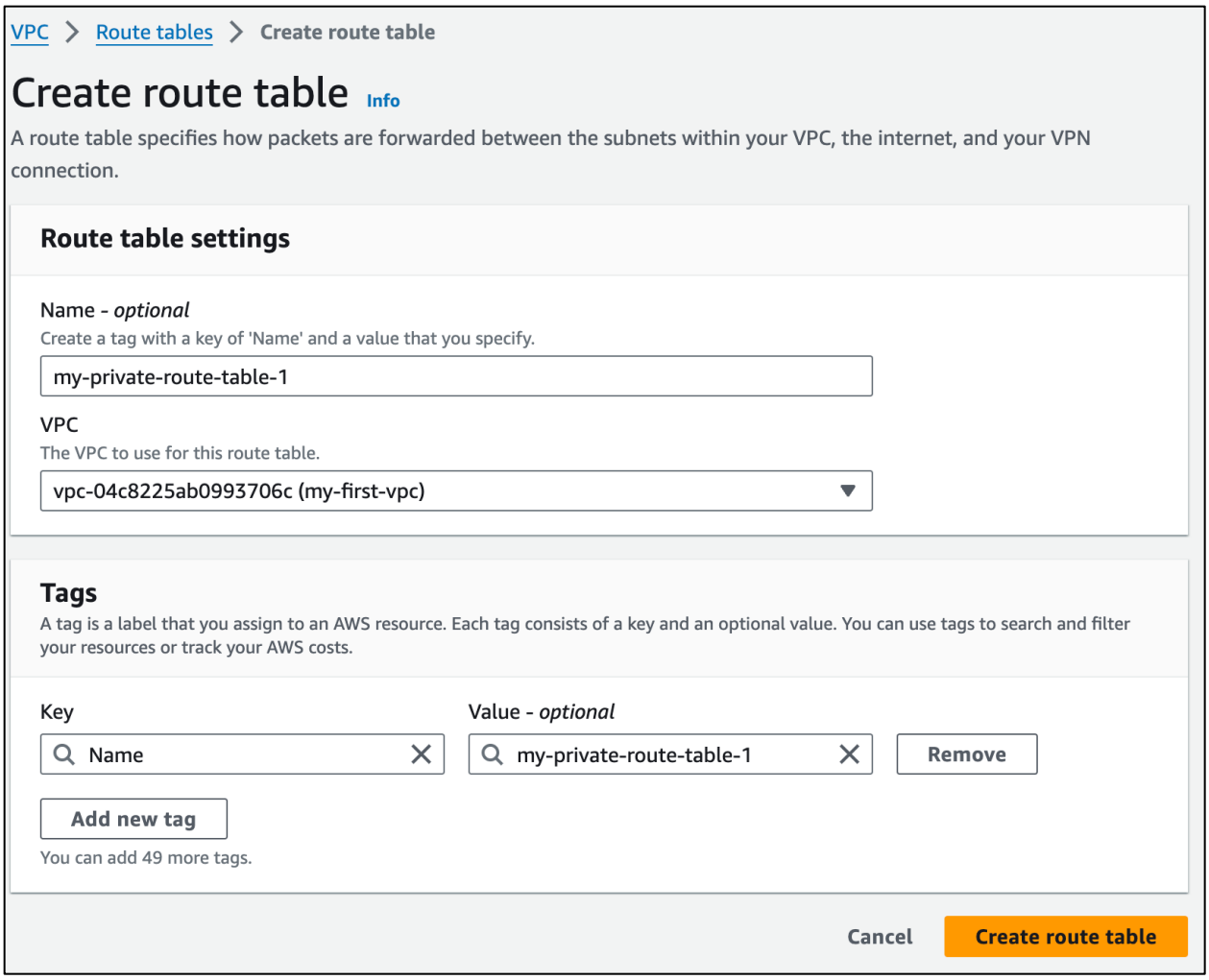


Step 4: Create Private Route Tables.

a. After creating the subnets, we should define the networks by making the Route Tables. What is a Route Table, and what is it used for?
Example: In a city, there are different roads leading to various neighborhoods and landmarks. Likewise, route tables in AWS VPC act as digital roadmaps, steering data traffic to other parts of your cloud infrastructure and ensuring efficient communication between services and resources.

Remember that a default route table was created when we created the VPC. We will use it for public communication purposes.

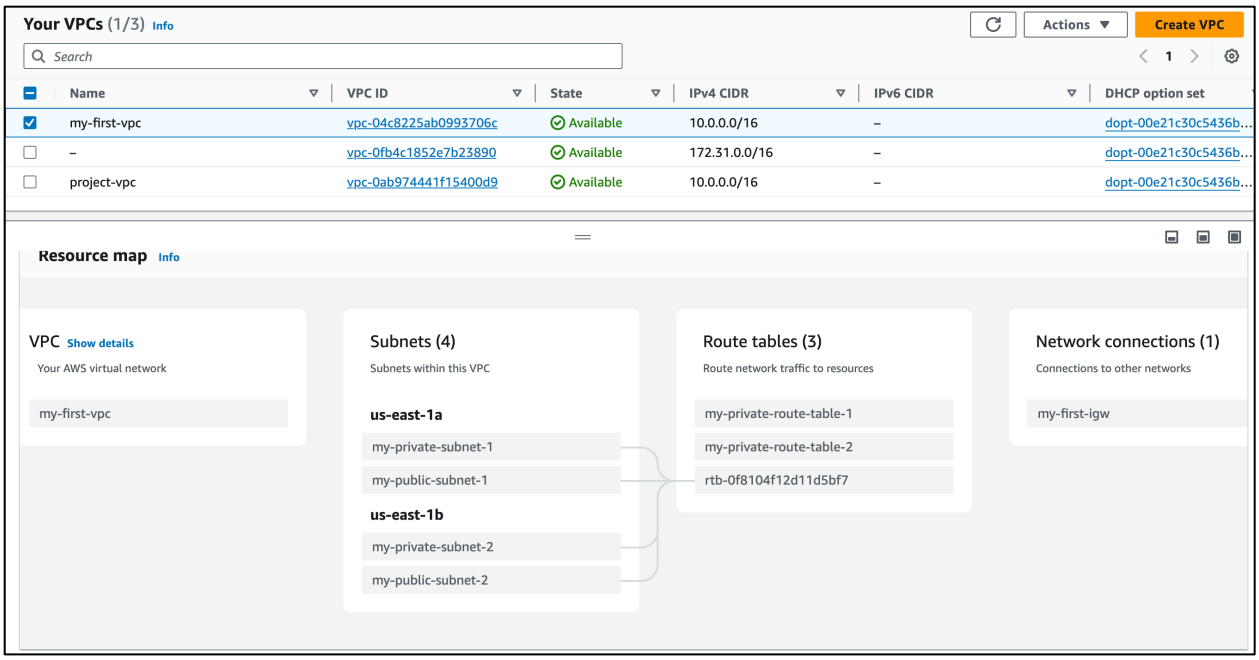
b. Let's create a Private Route Table.
This private route table allows communication for resources in a private subnet.



c. Your task is to create another Route Table named **my-private-route-table-2**.

d. Now, the updated resource map looks like it, as shown below. See the connection between subnets and route tables. The communication or transfer of data still occurs through the default route table. Now, we make the necessary changes to build the perfect network.

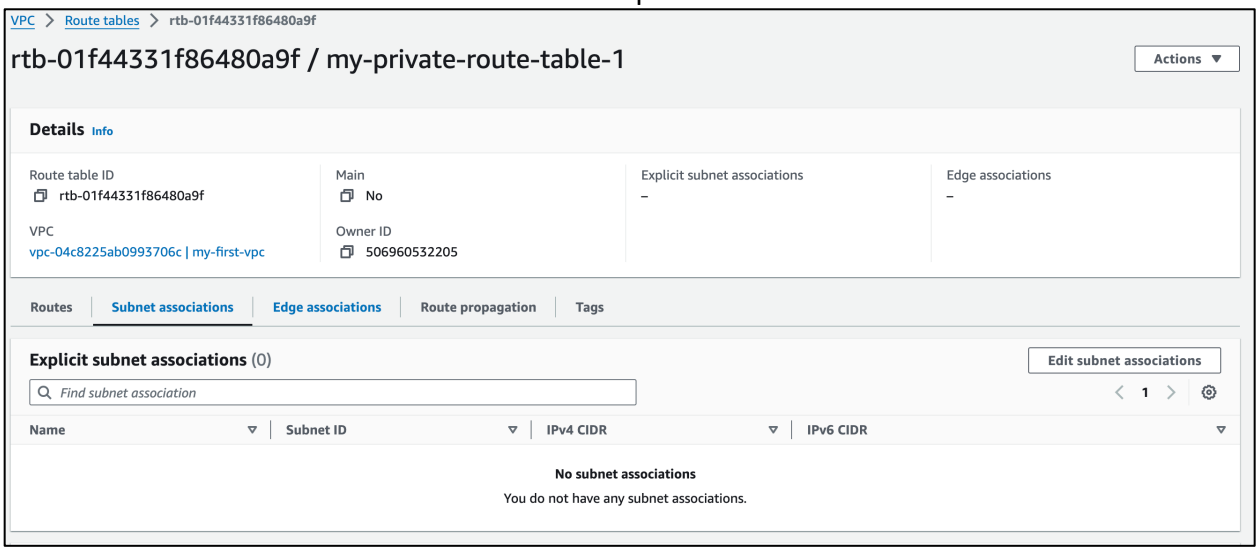
e. Let’s review the map after creating the route tables.

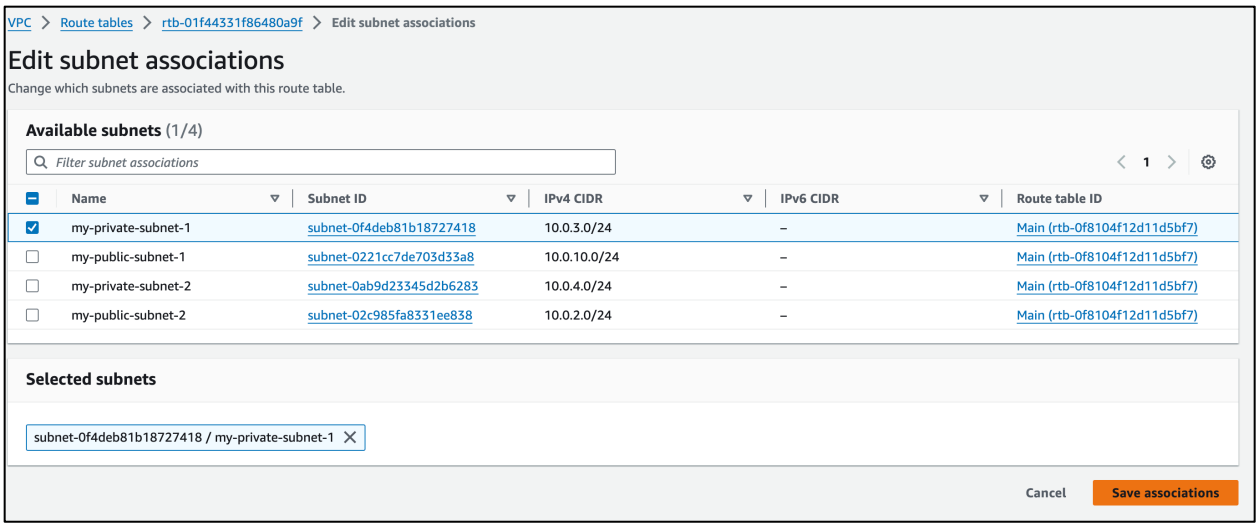


Step 5: Associate route tables to the subnets appropriately.

e. Since we have the default route table, we want to use it for public communication, and we will make changes in the private route tables alone.

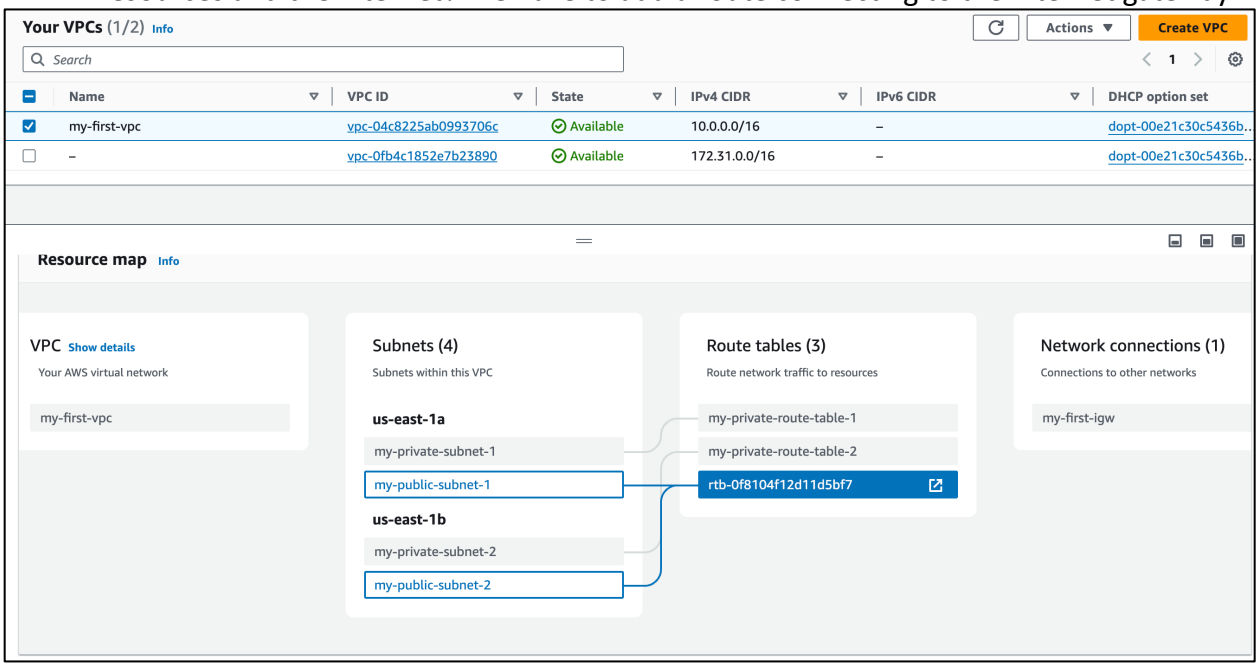
f. Let's edit the subnet association and add private subnet – 1 to this route table.



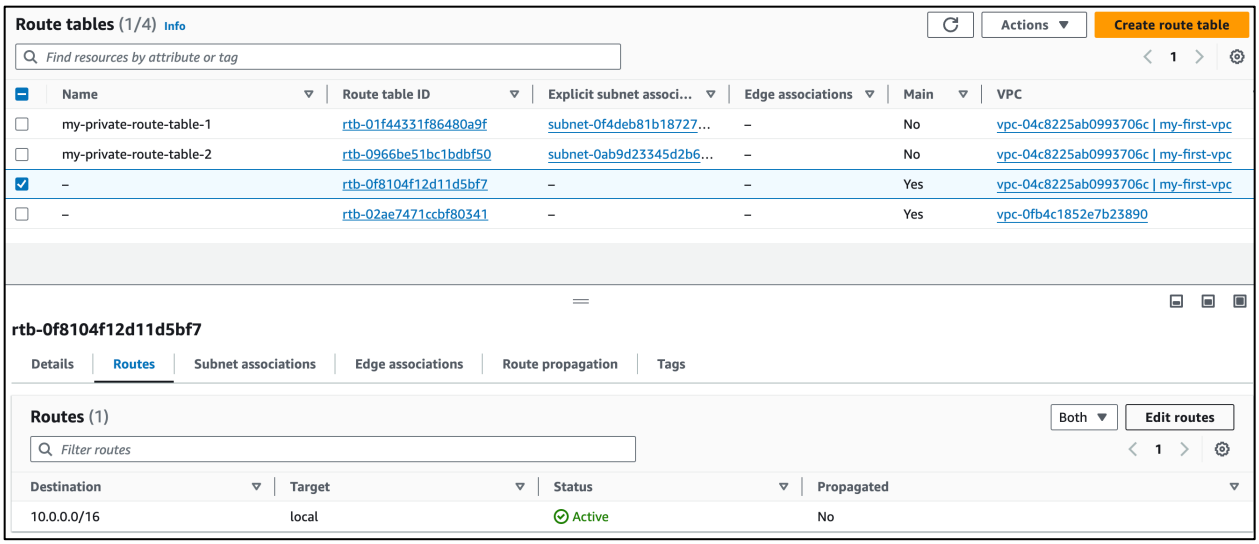


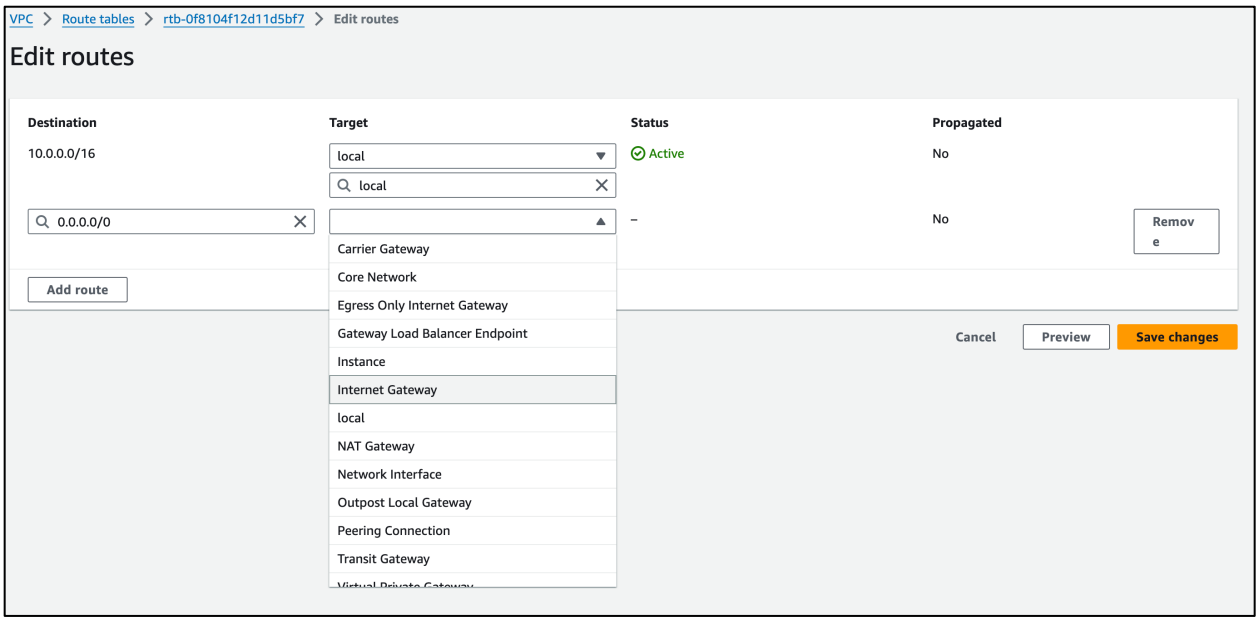
g. Your task is to associate the private route table 2 with **my-private-subnet-2**.

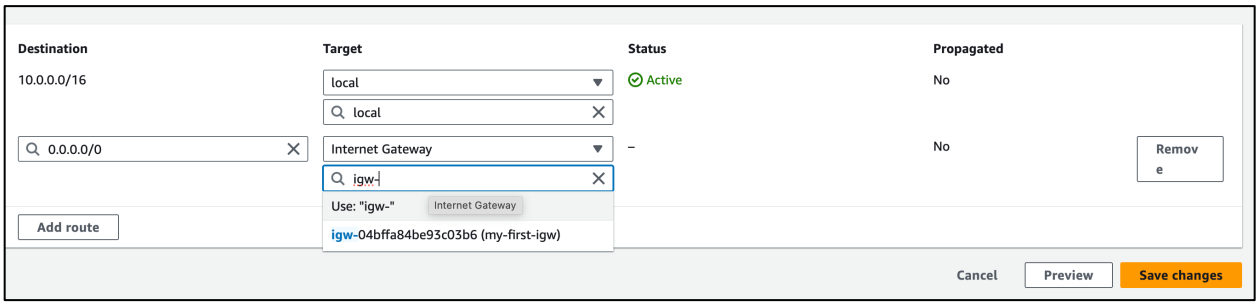
h. Let's review the Resource map now. This is what my map looks like right now. See, my public subnets are still associated with the default route table. So, we don't need to associate anything. But do you think about the communication between public resources and the internet? We have to add a route connecting to the internet gateway.



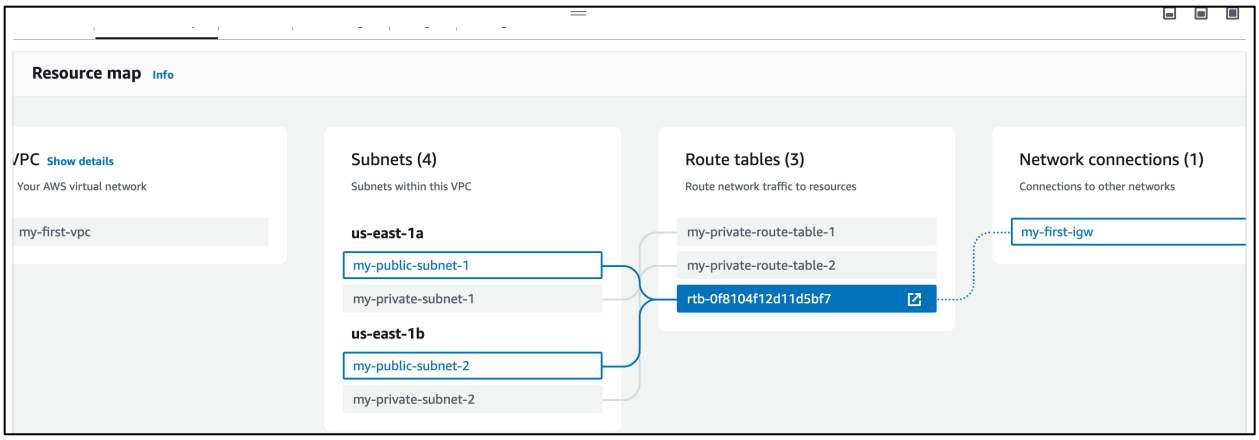
i. Let's edit the route option by adding the internet gateway to the default route table. Only after adding this route do our public servers placed in the public subnets communicate to the internet.







j. Let's review our Resource Map again.



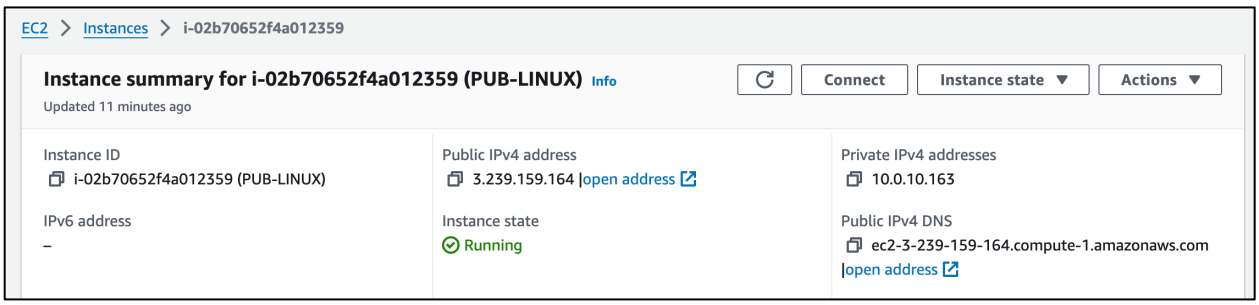
Note: This is a Deliverable.

Step 6: Test the Connectivity by Creating a Public EC2.

a. Create an ec2 server in the **my-public-subnet-1.**
Name: PUB-LINUX
KeyPair: Use the existing one or create one. Refer to folder **SSH_into_EC2.**
VPC: my-first-vpc
Subnet: my-public-subnet-1
Auto Assign public IP: Enable
SG: Create a new security group
Name: my-public-security-group
Description: Allow public communication
Type: SSH, Source: Anywhere
Type: All ICMP – IPV4, Source: Anywhere

The rest of the options are default.

Launch the instance.



Note: This is a Deliverable.

b. Login to the ec2 and see if public communication is allowed.

Observe that the public Linux allows communication from the Internet.
My laptop can reach the public server.

```
(base) venkatgirisasanapuri@Venkatgiris-MacBook-Air Downloads % ping 3.239.159.164
PING 3.239.159.164 (3.239.159.164): 56 data bytes
64 bytes from 3.239.159.164: icmp_seq=0 ttl=112 time=68.058 ms
64 bytes from 3.239.159.164: icmp_seq=1 ttl=112 time=43.676 ms
64 bytes from 3.239.159.164: icmp_seq=2 ttl=112 time=52.904 ms
^C
--- 3.239.159.164 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 43.676/54.879/68.058/10.051 ms
(base) venkatgirisasanapuri@Venkatgiris-MacBook-Air Downloads %
```

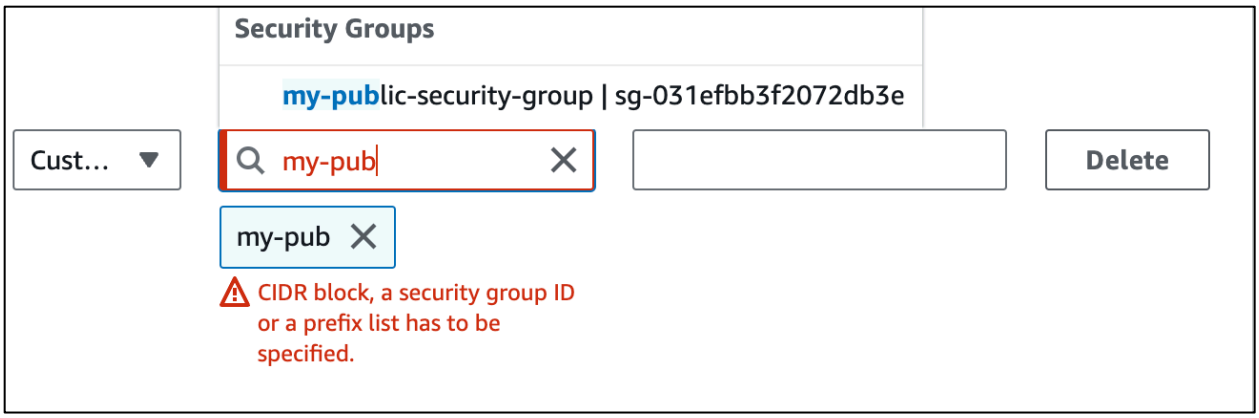
We can access the internet from the public server.

```
     ~~~         /
       ~~._.   _/
          _/ _/
        _/m/'
[ec2-user@ip-10-0-10-163 ~]$ ping google.com
PING google.com (172.253.115.102) 56(84) bytes of data.
64 bytes from bg-in-f102.1e100.net (172.253.115.102): icmp_seq=1 ttl=105 time=2.22 ms
64 bytes from bg-in-f102.1e100.net (172.253.115.102): icmp_seq=2 ttl=105 time=2.29 ms
64 bytes from bg-in-f102.1e100.net (172.253.115.102): icmp_seq=3 ttl=105 time=2.31 ms
^C
```

Step 7: Test the Connectivity by Creating a Private EC2.

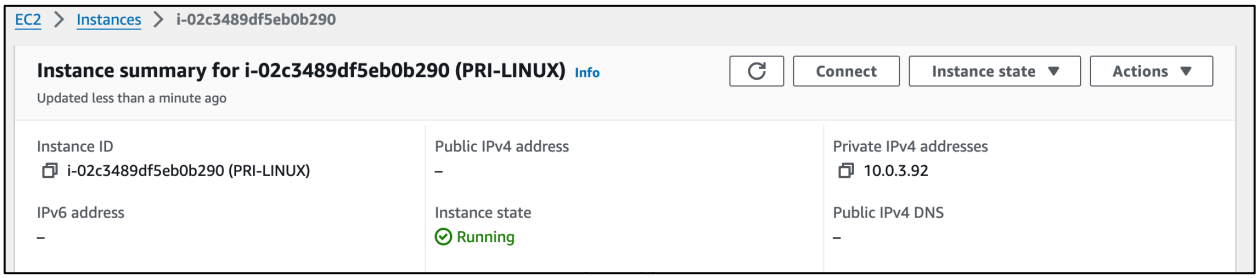
a. Create an ec2 server in the **my-public-subnet-1.**
Name: PRI-LINUX
KeyPair: Use the existing one or create one. Refer to folder **SSH_into_EC2.**
VPC: my-first-vpc
Subnet: my-private-subnet-1
Auto Assign public IP: Disable
SG: Create a new security group
Name: my-private-security-group
Description: Does not allow public communication
Type: SSH, Source: Anywhere
Type: All ICMP – IPV4, Source: Search for **my-public-security-group** created in the last step and select the SG created in the earlier step**.**

So, that means we are allowing communication from the servers in the public subnet only.



The rest of the options are default.

Launch the instance.



Note: This is a Deliverable.

b. Let's test the ping from our public ec2 server to private ec2.
ping <private-ip of PRI-LINUX>

```
[ec2-user@ip-10-0-10-163 ~]$ ping 10.0.3.92
PING 10.0.3.92 (10.0.3.92) 56(84) bytes of data.
64 bytes from 10.0.3.92: icmp_seq=1 ttl=127 time=0.483 ms
64 bytes from 10.0.3.92: icmp_seq=2 ttl=127 time=0.534 ms
64 bytes from 10.0.3.92: icmp_seq=3 ttl=127 time=0.443 ms
^C
--- 10.0.3.92 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2097ms
rtt min/avg/max/mdev = 0.443/0.486/0.534/0.037 ms
```

Step 8: How do you login to the Private server (i.e., ec2 in the private subnet)

a. Refer to the other files in the same assignment, which depends on your operating system.

b. After you log in, check if you can reach the internet. See that we are unable to reach the internet. That is where NAT comes into the picture.

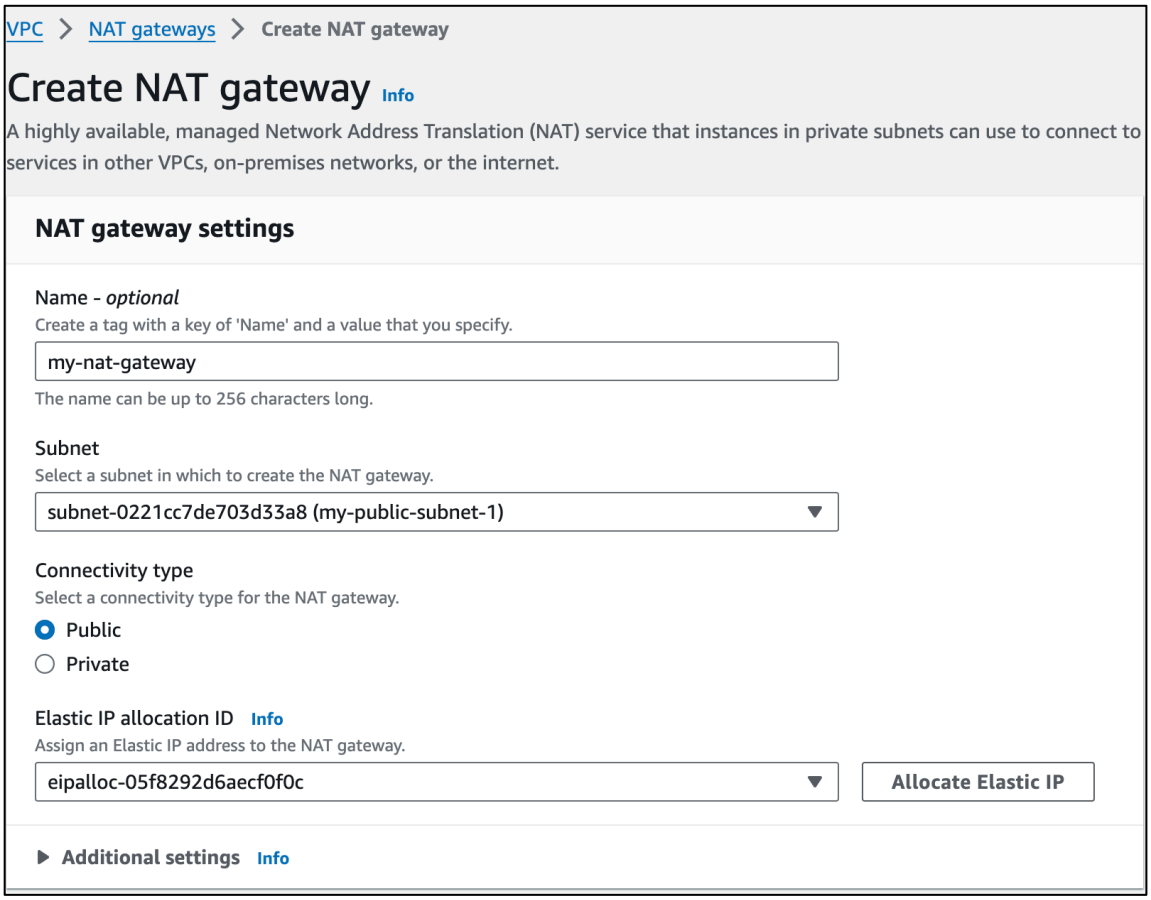
```
[ec2-user@ip-10-0-10-163 ~]$ ssh 10.0.3.92
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.139) 56(84) bytes of data.
^C
--- google.com ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4162ms

[ec2-user@ip-10-0-3-92 ~]$
```

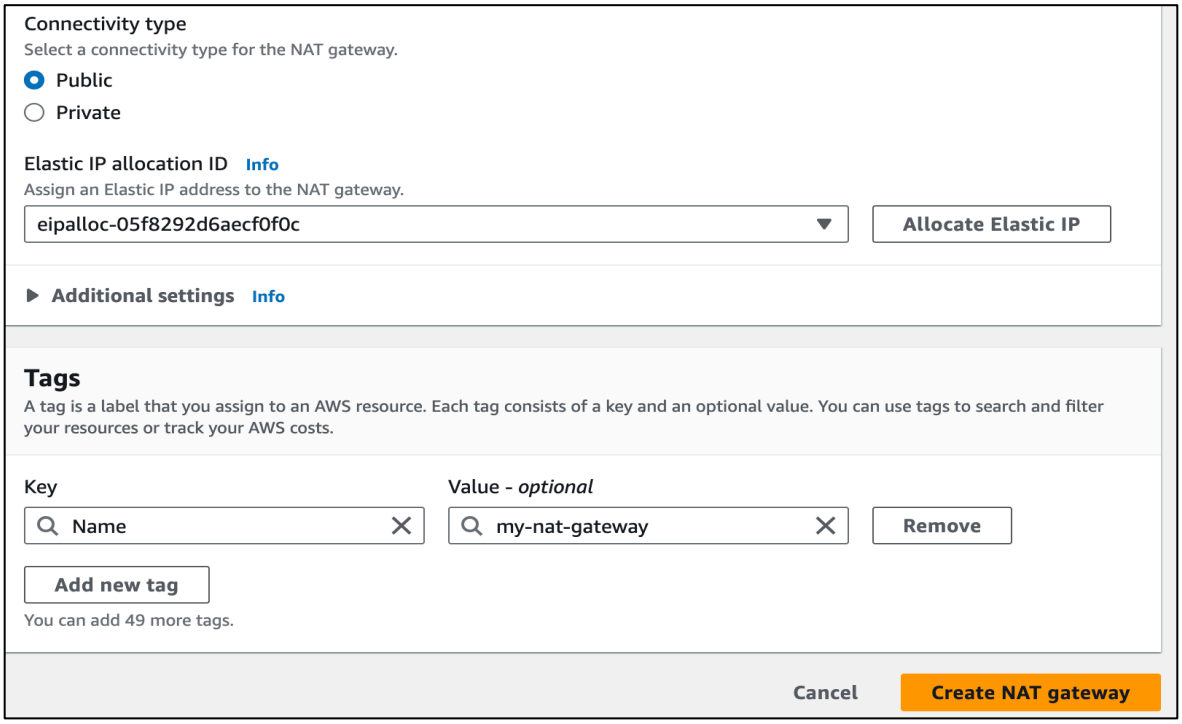
Step 9: NAT Gateway creation and testing internet connection from the private server.

a. Do you think about how the servers in the private subnet communicate to the internet when needed? NAT Gateways is the answer. We will place this Nat Gateway in the public subnet.

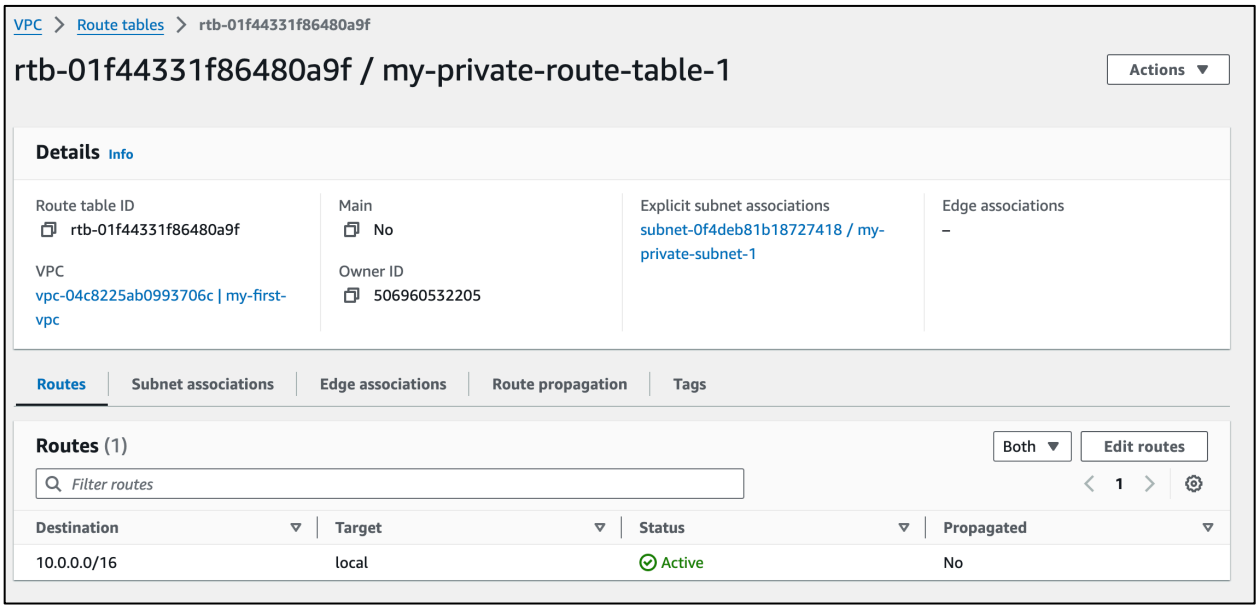
Let's create one.

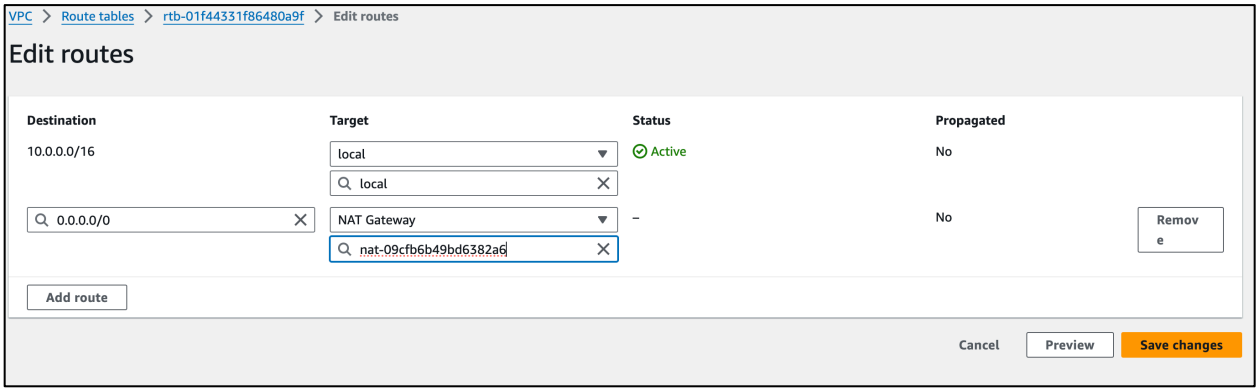
b. Click on **Allocate Elastic IP**

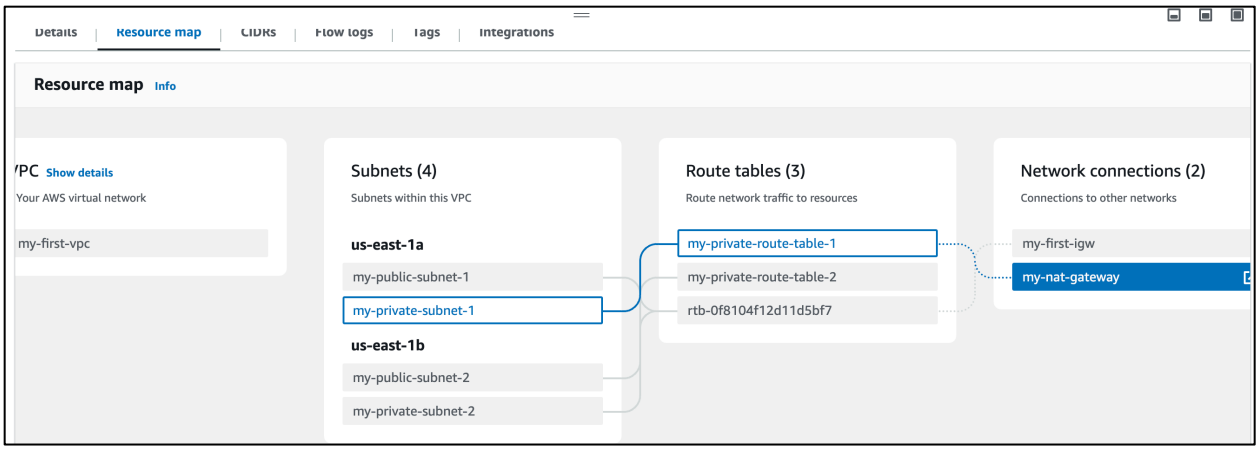




c. Remember, we have added a route from the Main(public) Route Table to the internet gateway. Similarly, we will add a route from the private route table to the Nat gateway.







Note: This is a Deliverable.

d. Once NAT is created, after two minutes, test the same command in the private server.

```
[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.139) 56(84) bytes of data.
^C
--- google.com ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4162ms

[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.101) 56(84) bytes of data.
^C
--- google.com ping statistics ---
19 packets transmitted, 0 received, 100% packet loss, time 18709ms

[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.100) 56(84) bytes of data.
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=1 ttl=107 time=3.09 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=2 ttl=107 time=2.54 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=3 ttl=107 time=2.48 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=4 ttl=107 time=2.49 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=5 ttl=107 time=2.46 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=6 ttl=107 time=2.63 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 2.464/2.616/3.085/0.217 ms
[ec2-user@ip-10-0-3-92 ~]$
```

Note: This is a Deliverable. (The private IP in the above screenshot should match the private ip of private ec2)

So, we use a NAT gateway for internet communications to a private server.

Deliverables: Submit the screenshots where it mentions "Note: This is a Deliverable." The screenshots should be submitted in the pdf format.
Step 5 – f
Step 6 – a
Step 7 – a
Step 9 – c and d

Note: Delete NAT Gateway and the two EC2 instances created in this lab

Done!!!