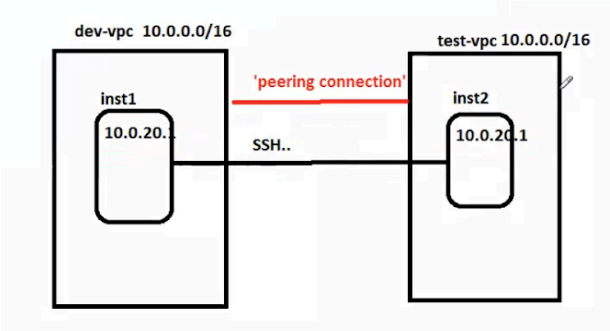
AWS Peering Connection

The diagram depicts two VPCs, namely 'dev-vpc' and 'test-vpc.' Each VPC contains a respective instance: 'dev-vpc' has an instance named 'inst1,' and 'test-vpc' has an instance named 'inst2.' The objective is to establish a connection between 'inst1' and 'inst2.'



But we have limitation

if we perform ssh connection between these two instances, it won't be possible.

We can not change the CIDR, once the VPCs are created

So the solution for this problem is **peering connection.**

Peering connections in the context of AWS (Amazon Web Services) refer to the establishment of network connections between two separate VPCs (Virtual Private Clouds) or between a VPC and an on-premises network. This connectivity is achieved through the AWS Peering feature, and it serves several important needs:

1. ****Inter-VPC Communication:**** Peering connections enable secure and direct communication between different VPCs within the same or different AWS regions. This is useful when you have multiple VPCs for different workloads or teams, and you need them to communicate with each other.

2. ****Resource Sharing:**** Peering allows resources in one VPC to access resources in another VPC. This is particularly beneficial when you want to share data, services, or applications across different VPCs while maintaining isolation and security.

3. ****Cost Savings:**** Instead of using the public internet or dedicated connections, peering connections provide a cost-effective way for VPCs to communicate with each other. Data transfer between peered VPCs is typically faster and doesn't incur additional data transfer costs.

4. ****Simplified Network Architecture:**** Peering connections simplify the network architecture by allowing VPCs to communicate directly without the need for complex routing configurations. This ease of communication can be crucial in scenarios where services in different VPCs need to interact seamlessly.

5. **Hybrid Cloud Connectivity:** Peering connections can be used to establish connectivity between an AWS VPC and an on-premises data centre. This is especially useful in hybrid cloud architectures where certain workloads or services reside on-premises while others are hosted in the AWS cloud.

6. **Improved Security and Control:** Peering connections can be configured with route tables and security groups to control and secure traffic between the peered VPCs. This helps in implementing fine-grained access controls and ensures that only authorised communication occurs between VPCs.

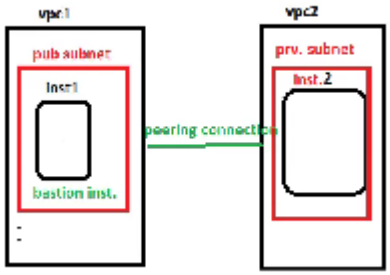
7. **Global Reach:** AWS peering connections can be established not only within the same region but also across different AWS regions, allowing for global communication between VPCs.

In summary, peering connections in AWS facilitate efficient, secure, and cost-effective communication between different VPCs or between a VPC and an on-premises network, contributing to a more flexible and scalable cloud infrastructure.

==========

Questions

Q1 How we can connect private instances to public instances



We have 2 virtual private clouds. vpc1 and vpc2 .

vpc1 has a public subnet and inside it, there is instance1, vpc2 has a private subnet and inside it, there is instance2. We want to connect vpc1 and vpc2.

->Using a peering connection.

Q2 What is a peering connection?

-> A peering connection in AWS is like a virtual bridge that connects two separate virtual networks. It allows these networks to communicate with each other as if they were part of the same network, even though they are distinct and separate entities. This connection is private, which means that the data exchanged between the networks doesn't travel over the public internet, ensuring security and better performance.

Q3 Bastion instance?

-> A Bastion instance, also known as a Bastion host or a Jump box, acts as an intermediary server between your local machine and the private instances within your VPC. It provides a secure point of entry into the private network. You can connect to the Bastion instance over SSH (Secure Shell) from your local machine, and from there, you can use the Bastion instance to access other instances within the private network.

simple example

Bastion Host (or Bastion Instance):

A Bastion host is a server that acts as an intermediary or a secure gateway between clients (such as system administrators) and private servers or resources within a network. The Bastion host is usually placed in a DMZ (Demilitarized Zone), a network segment that is isolated from the internal network and the internet. Its primary purpose is to provide a controlled and monitored entry point for accessing internal resources.

Use Case Example: Secure Remote Access to Private Instances in AWS:

Let's consider an example within the context of AWS:

1. **Network Setup:**

   - You have a Virtual Private Cloud (VPC) in AWS, and within this VPC, there are private instances (e.g., databases, application servers) that should not be directly accessible from the internet.

   - You also have a Bastion instance placed in a public subnet in the same VPC. This Bastion instance is the only server exposed to the internet.

2. **Accessing Private Instances:**

- System administrators or authorised users connect to the Bastion instance using SSH or another secure protocol.

- Once connected to the Bastion instance, they can then use it as a jumping-off point to access the private instances within the VPC.

3. **Security Measures:**

- The Bastion host is hardened and configured with strong security measures, including strict access controls, monitoring, and logging.

- Access to the Bastion host may be restricted using key pairs, multi-factor authentication, or other secure authentication methods.

4. **Bastion as a Secure Gateway:**

- The Bastion instance acts as a secure gateway, allowing administrators to access internal resources without exposing them directly to the internet.

- After connecting to the Bastion instance, administrators can use it to SSH or RDP into the private instances.

Benefits:

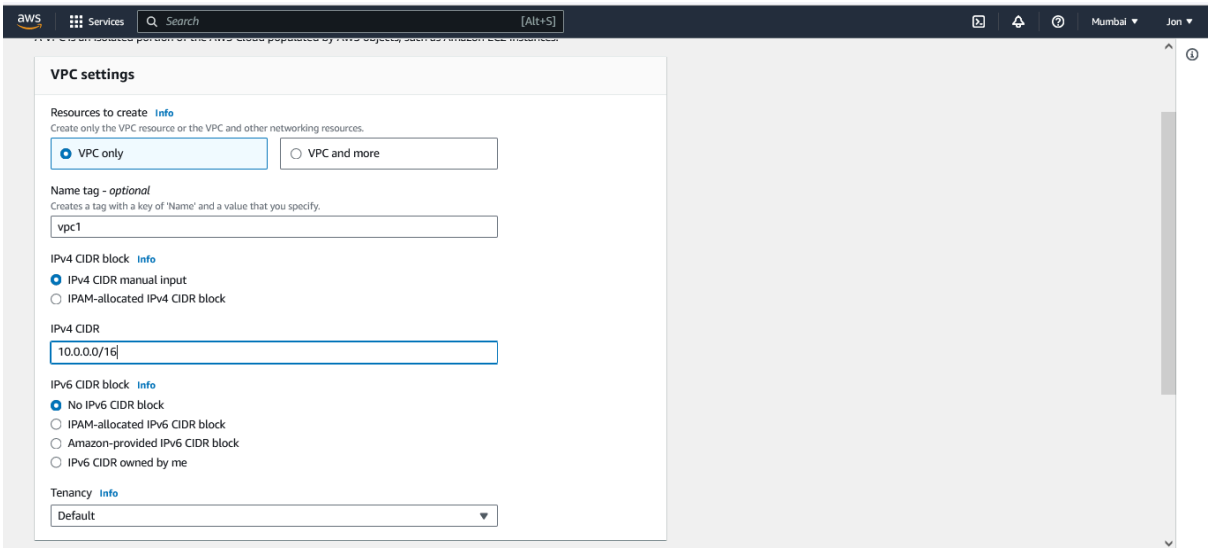
- Enhanced security: Private instances are not directly exposed to the internet, reducing the attack surface.

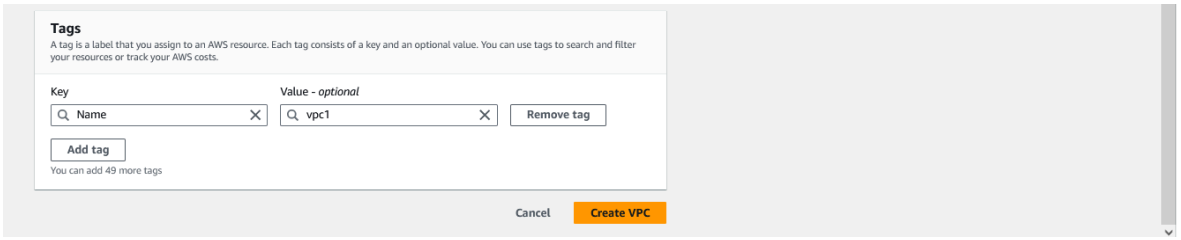
- Centralised access control: Access to internal resources can be tightly controlled and monitored through the Bastion host.

- Auditing and logging: All access to internal resources can be logged and audited through the Bastion instance.

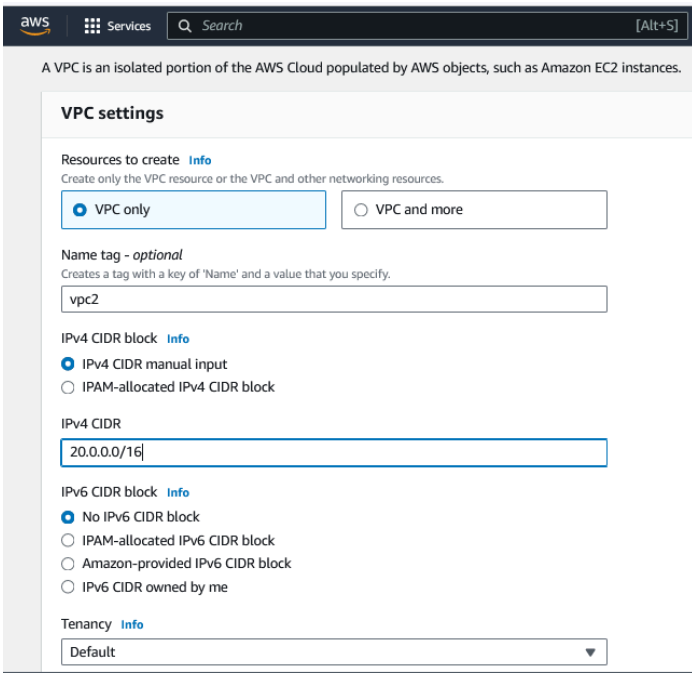
In summary, a Bastion instance serves as a secure entry point for accessing private resources within a network, and it helps maintain a more secure and controlled environment.

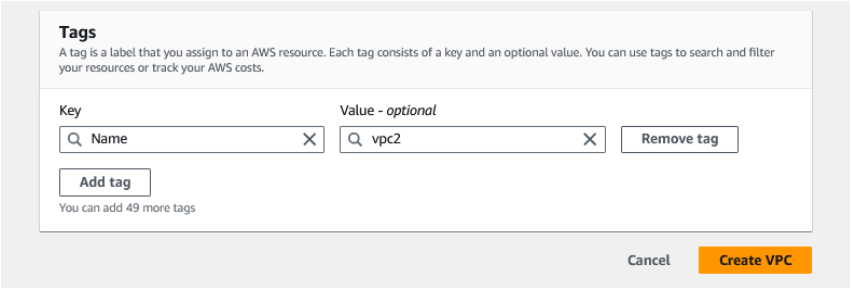
Step 1:-Create vpc1





Step 2:- Create vpc2

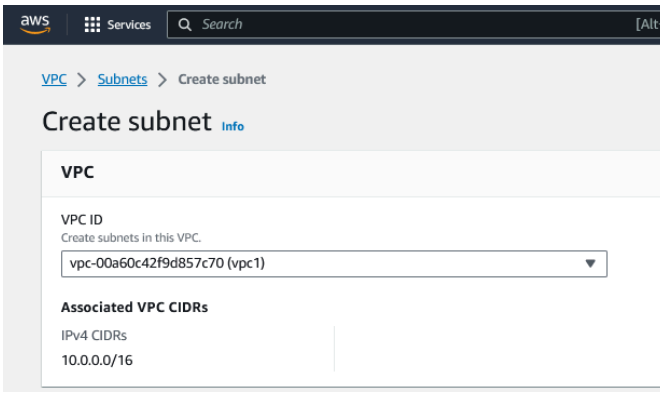


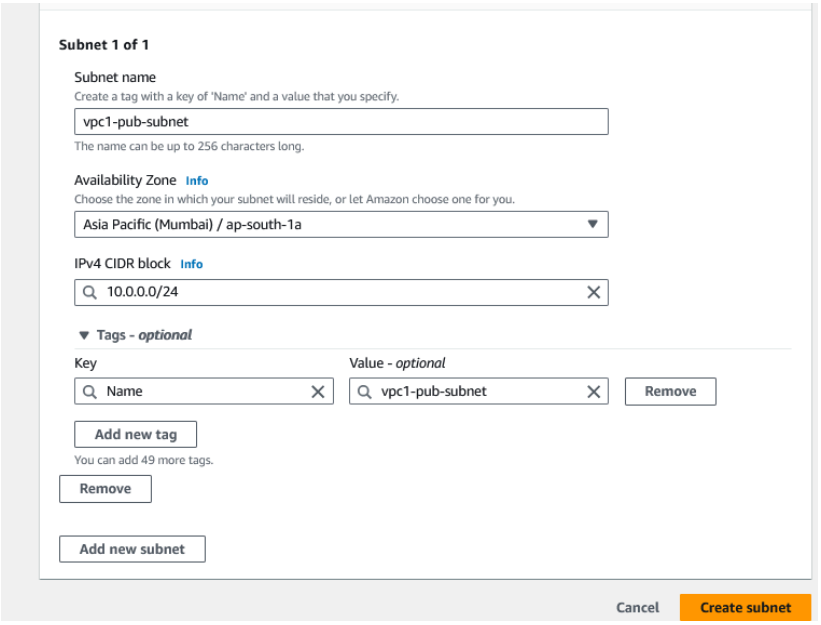


—----------

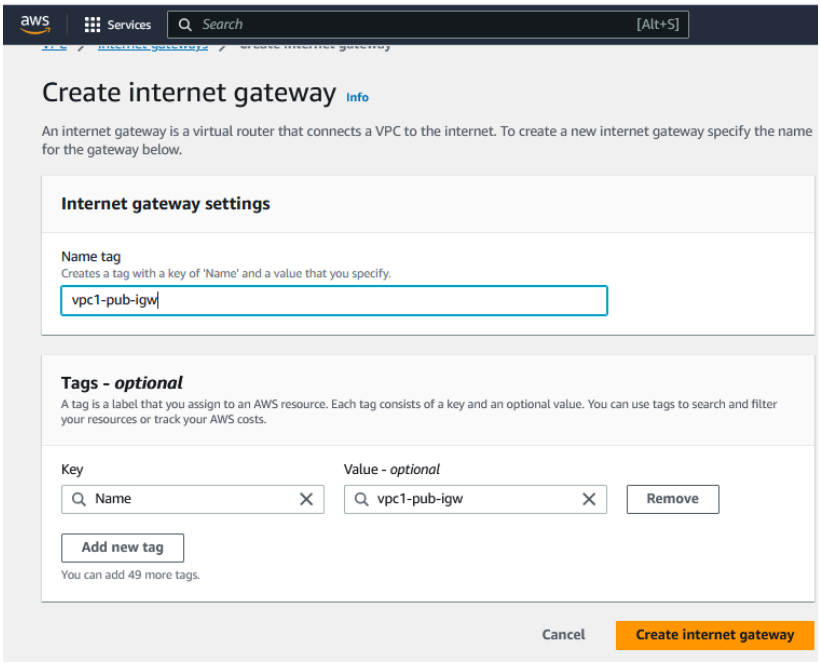
For VPC1

Step 3:- Create subnet for vpc1



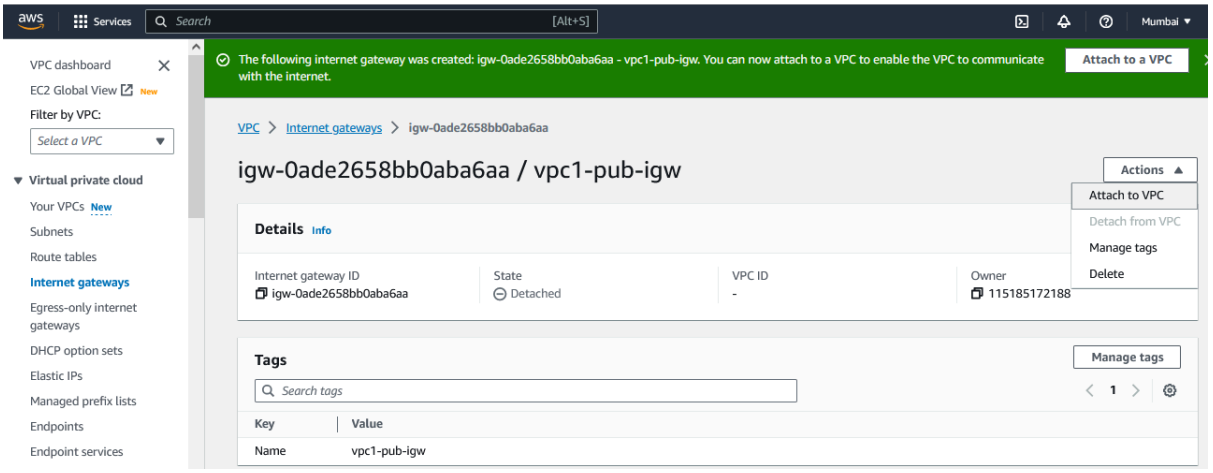


Step 4 :- Create a internet gateway for vpc1 as it is a public vpc

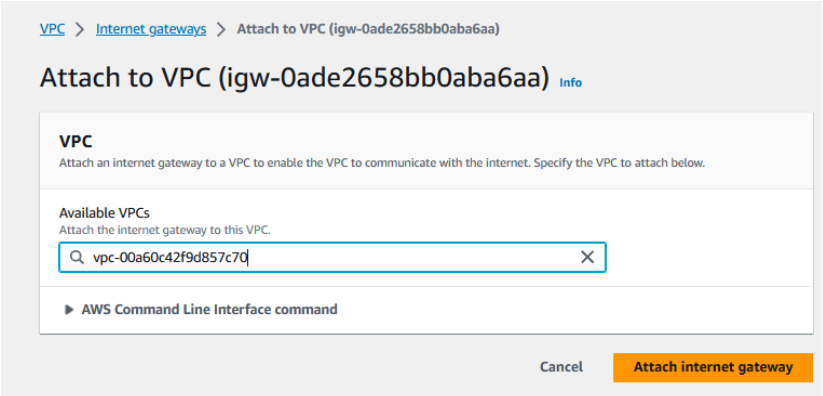


Step 5:- Attach internet gateway to vpc1

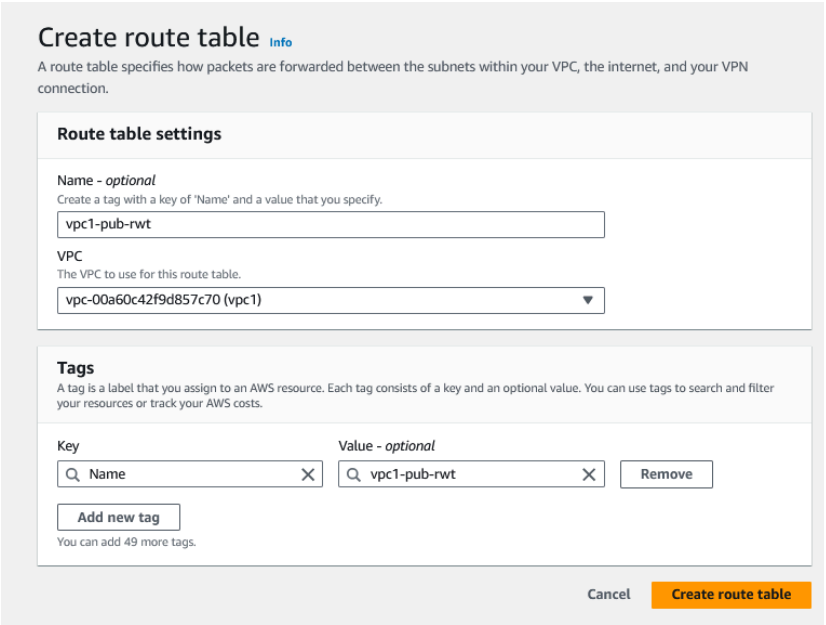
Actions -> Attach to VPC



Select vpc1

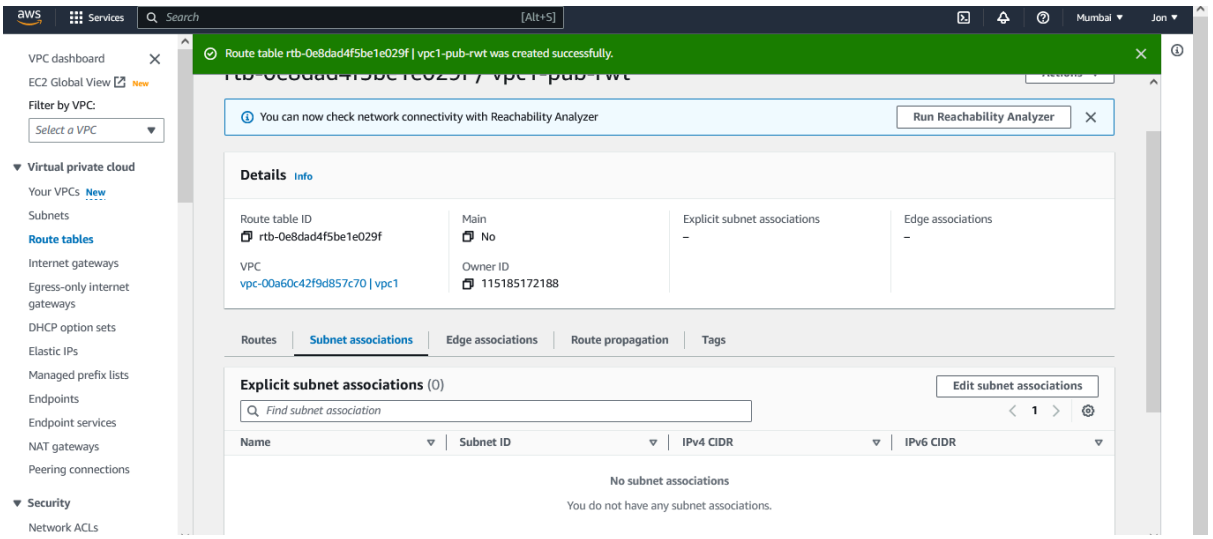


Step 6:- Create route table for vpc1

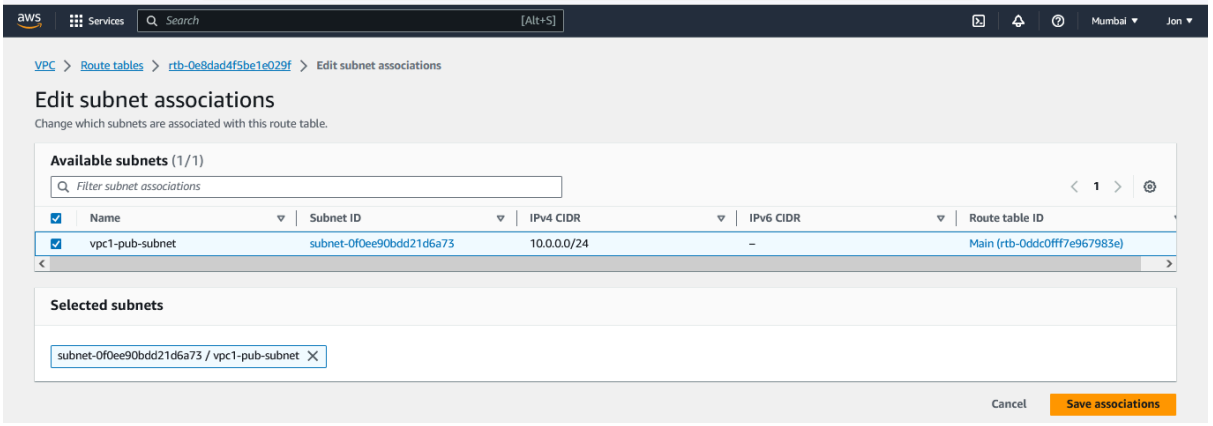


Step 7:- vpc1 subnet association

Subnet association -> Edit subnet association

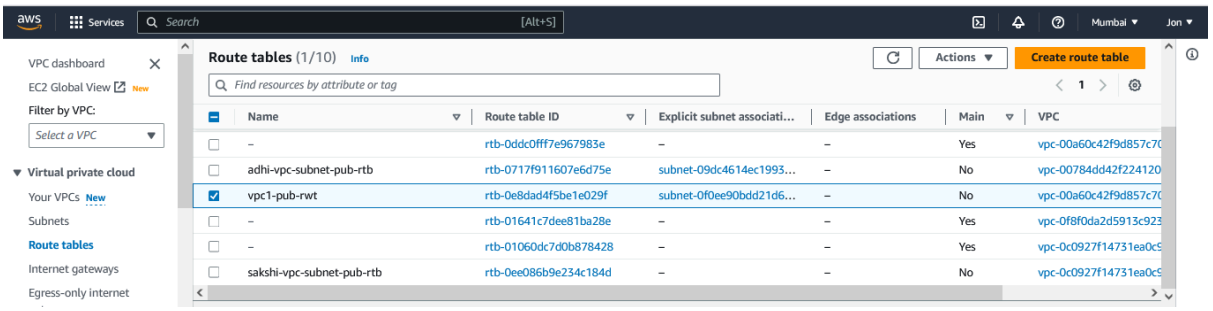


select vpc1-pub-subnet -> Click Save association

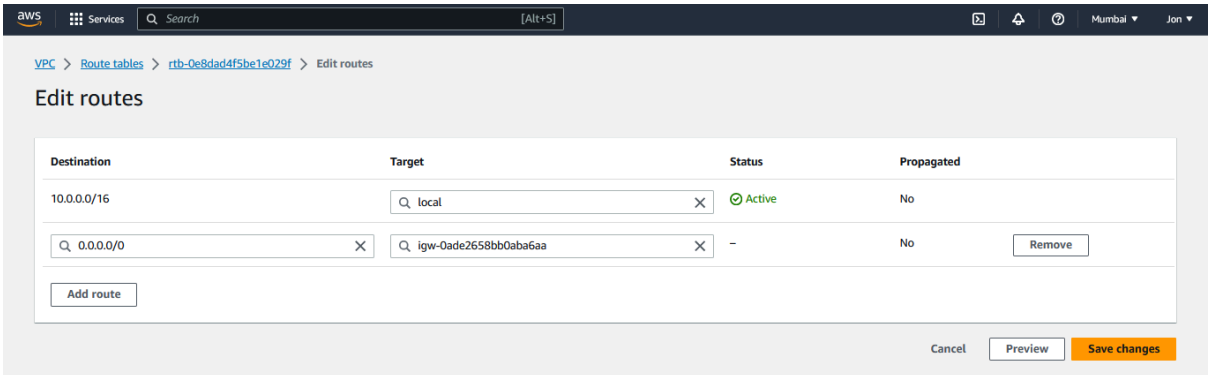


Step 8 :- Edit routes

Route tables ->Select vpc1->Routes->Edit routes

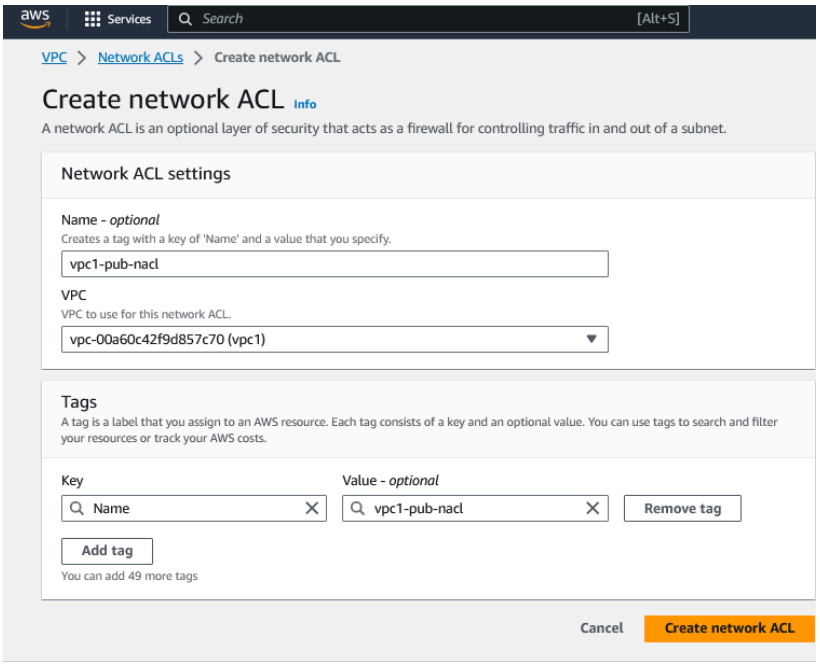


Target -> Internet Gateway



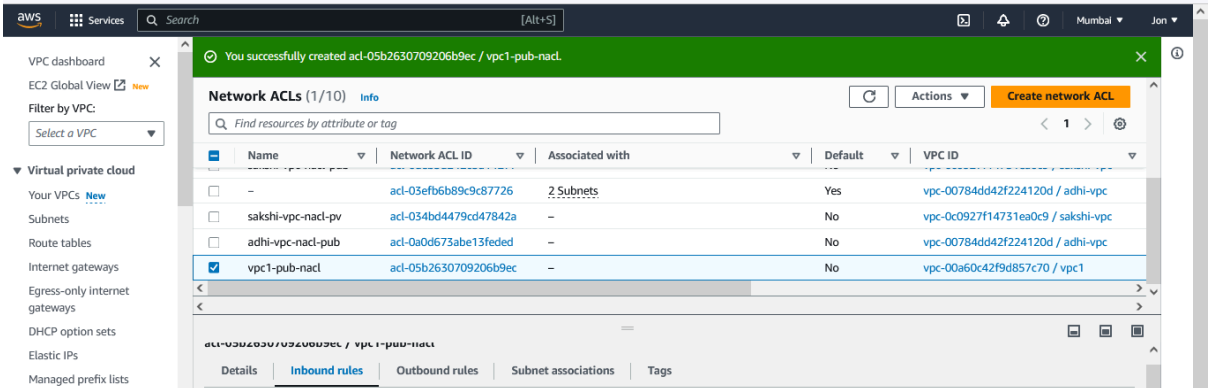
Step 9:- Nacl for vpc1

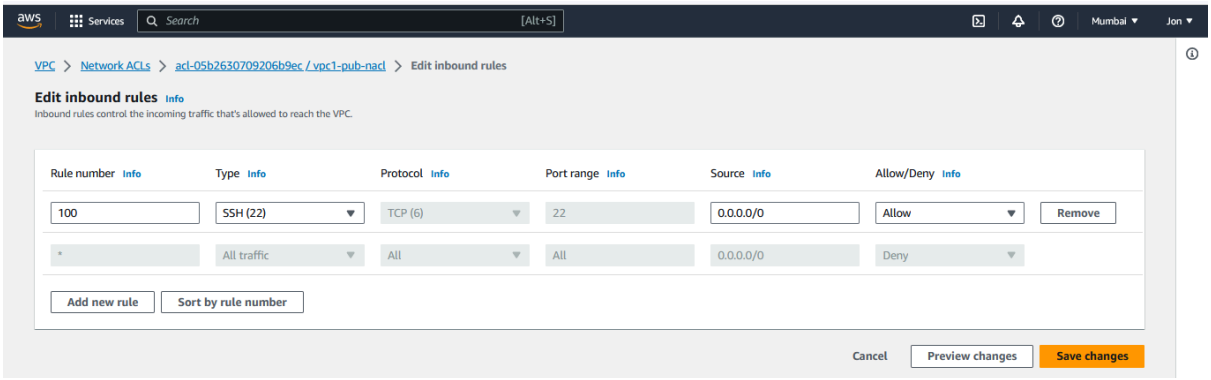
Security -> nacl -> Create nacl



Step 10 :- Edit inbound rules for nacl of vpc1

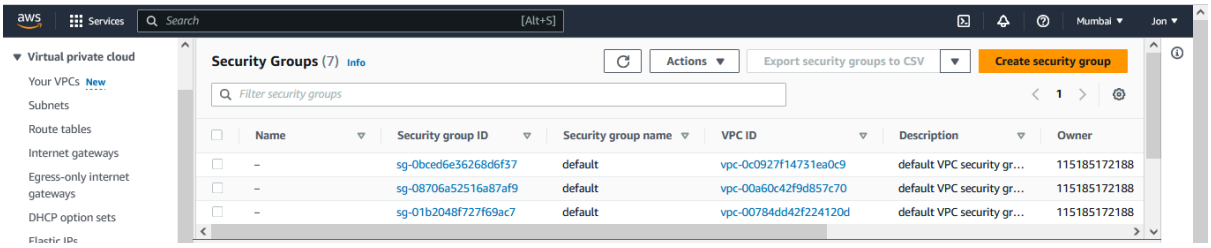
Security -> nacl -> Select vpc1 -> Inbound rules -> Edit inbound rules

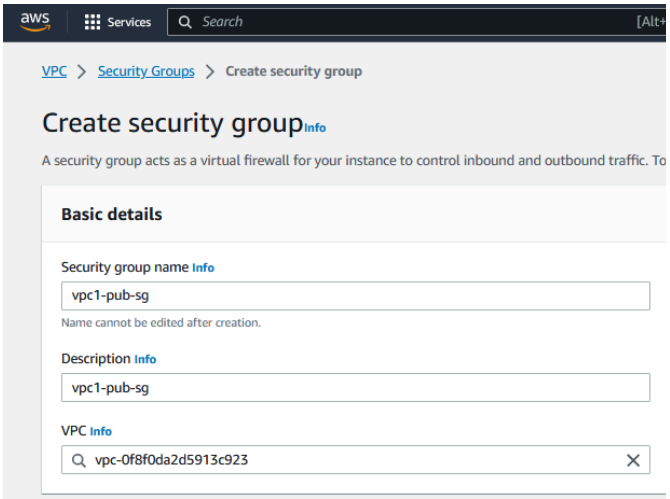


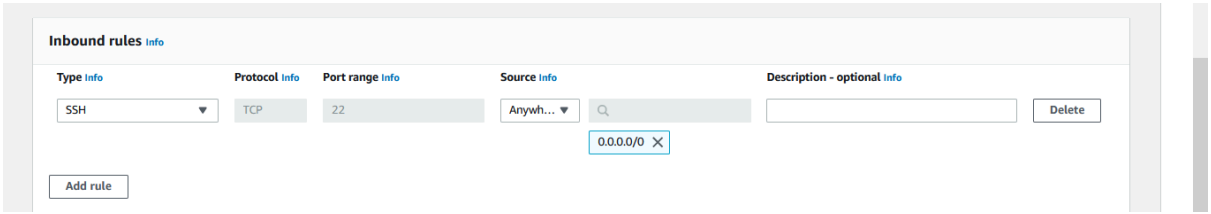


Step 11 :- Create a security group for vpc1

Security -> Security groups -> Create security group ->



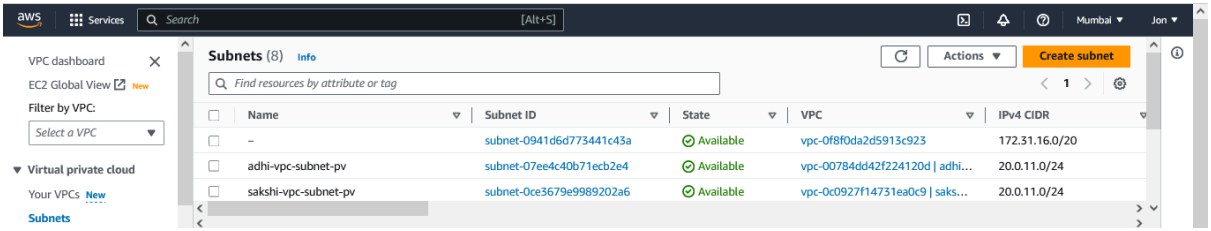


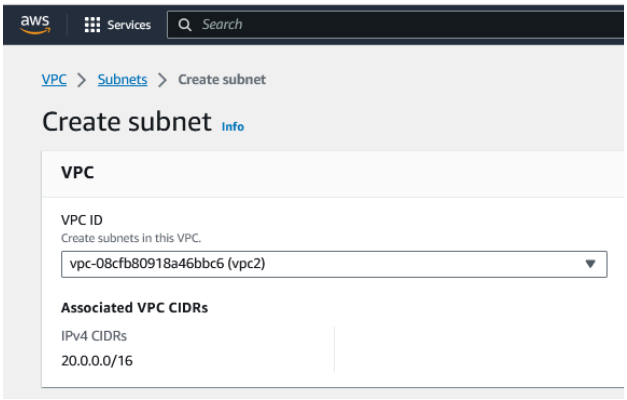


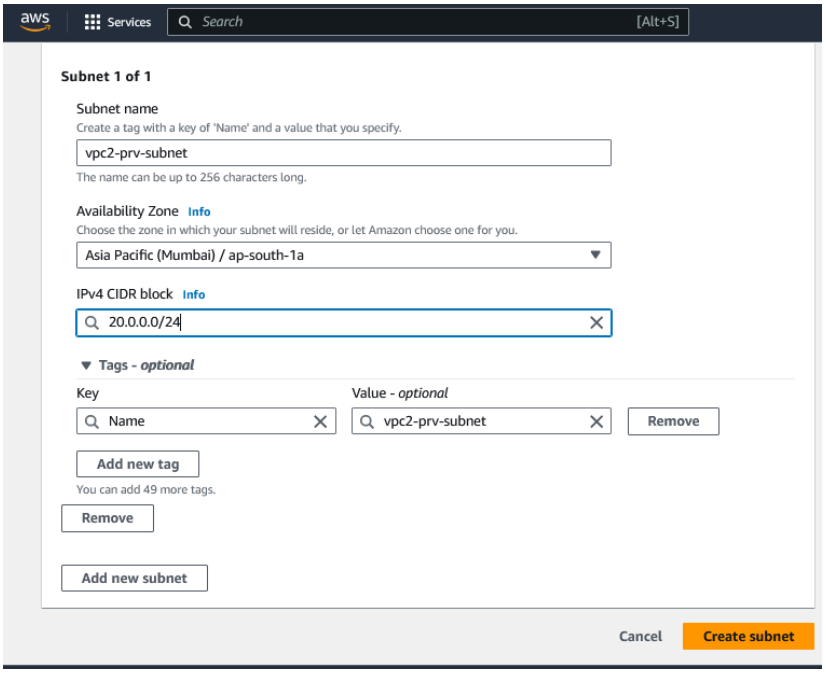
—----------

For VPC2

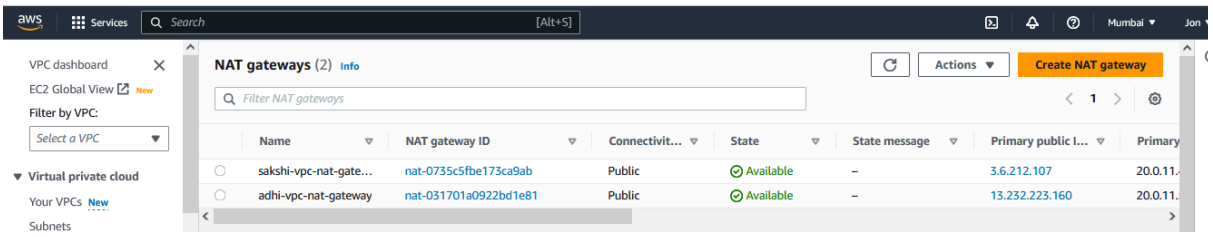
Step 1:- Create subnet for vpc2 (Private subnet)

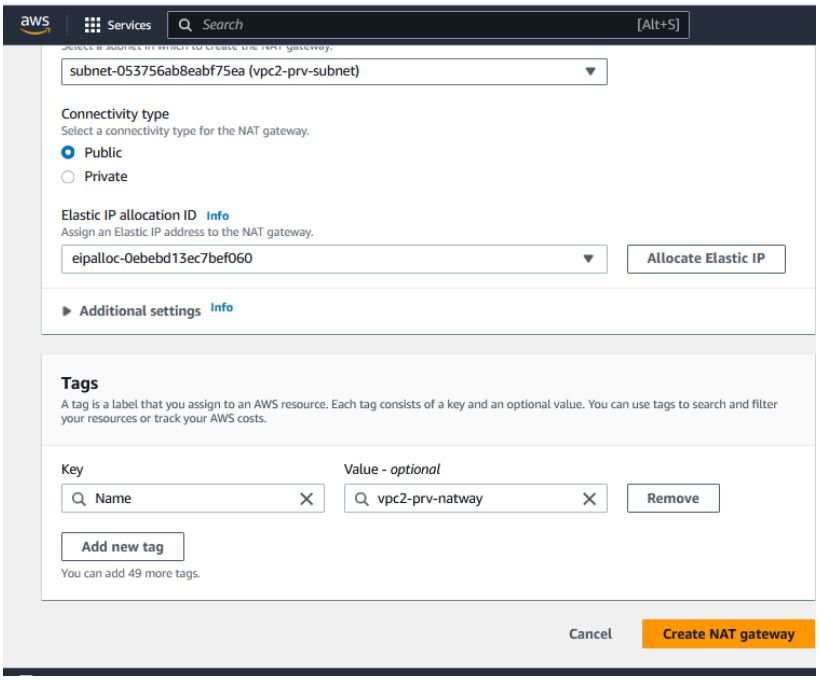




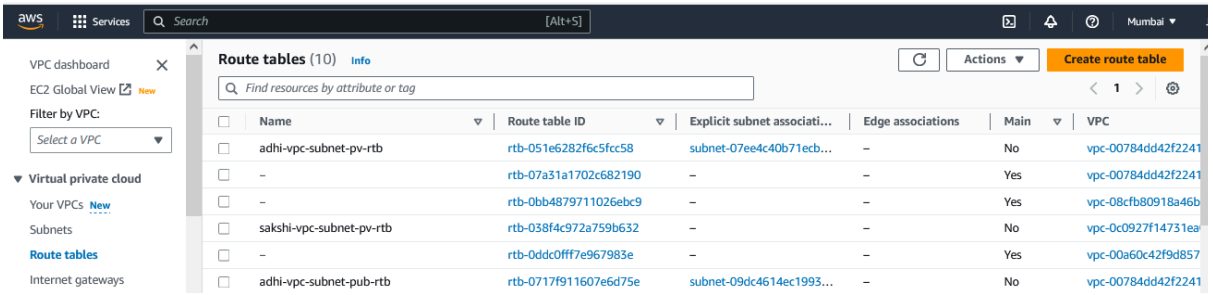


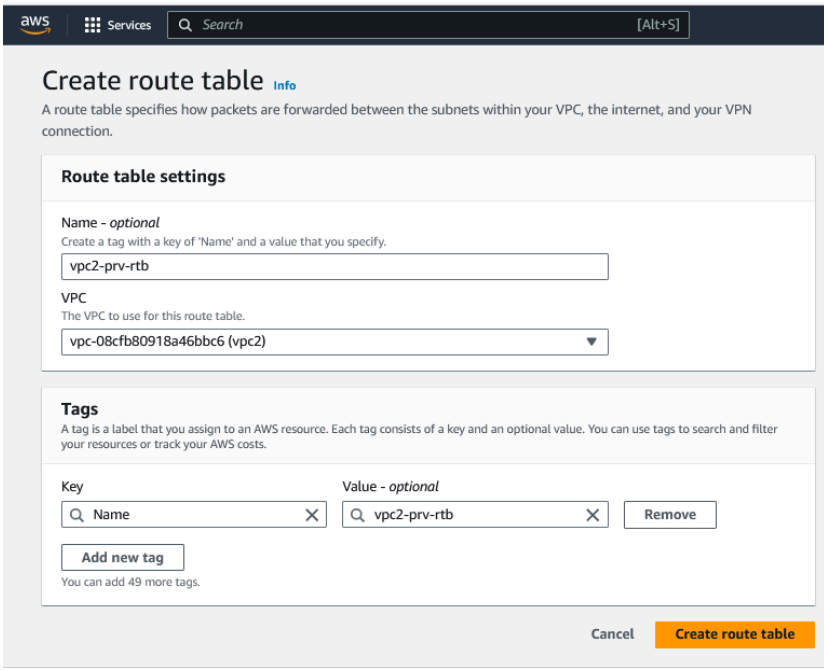
Step 2:- NAT gateway





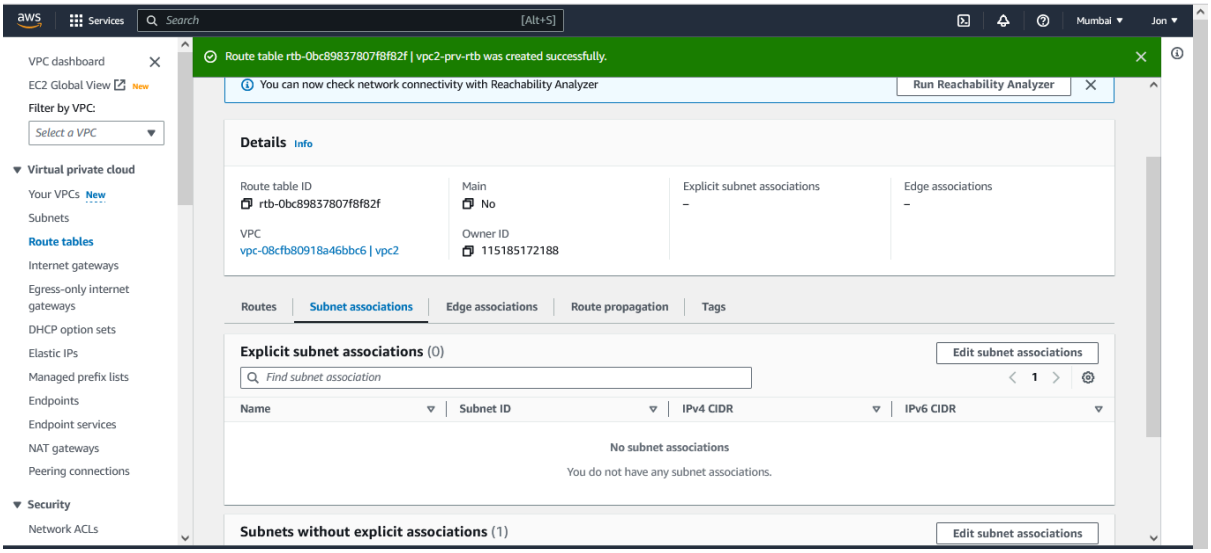
Step 3:- Route table

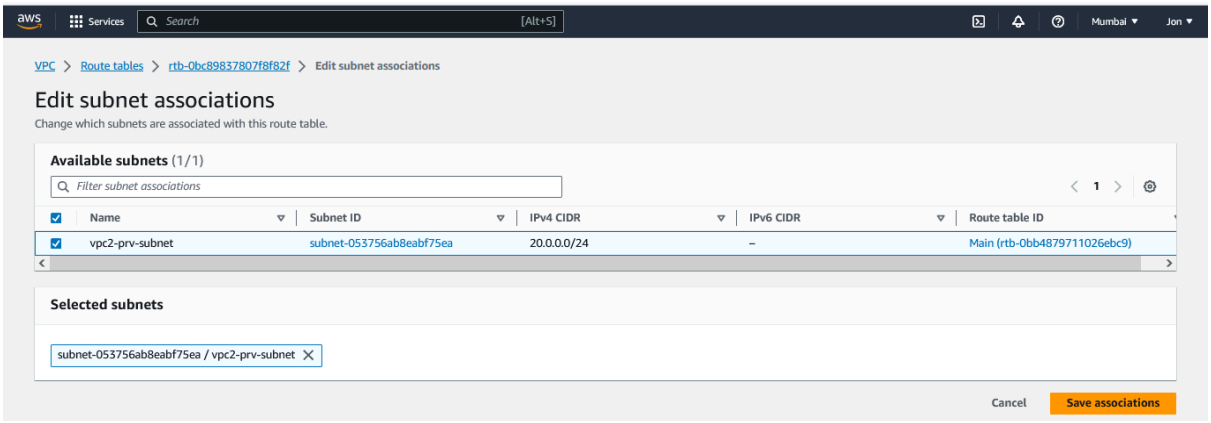




Step 4:- Subnet Associations

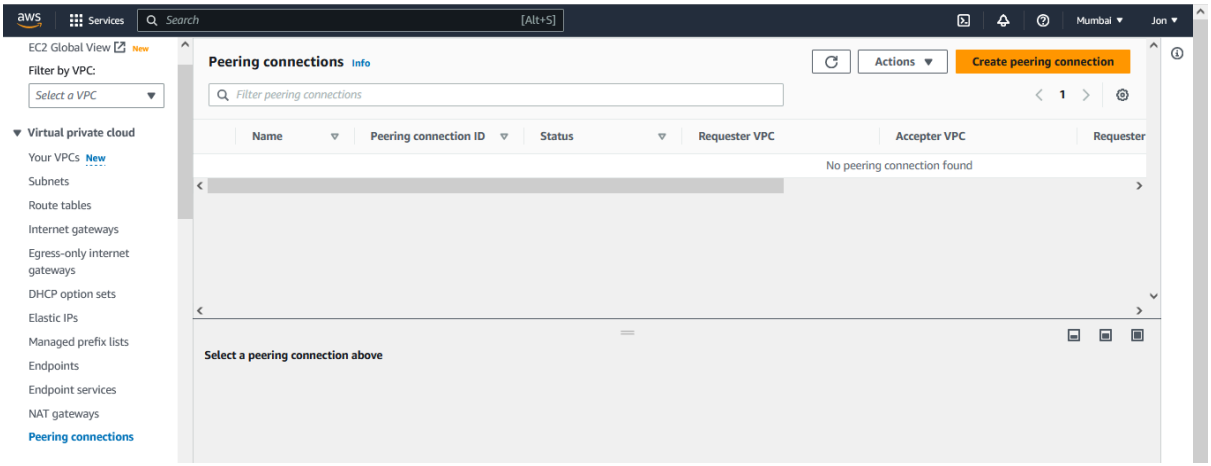
Edit subnet associations





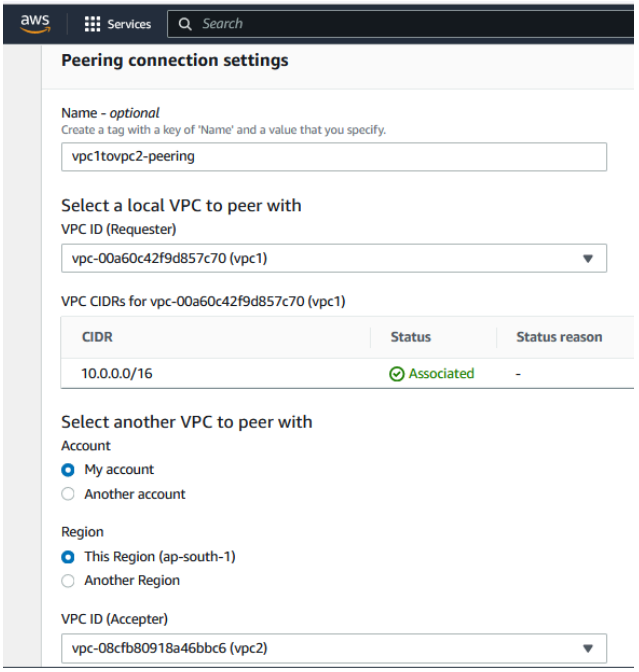
Step 5:- Peering connection

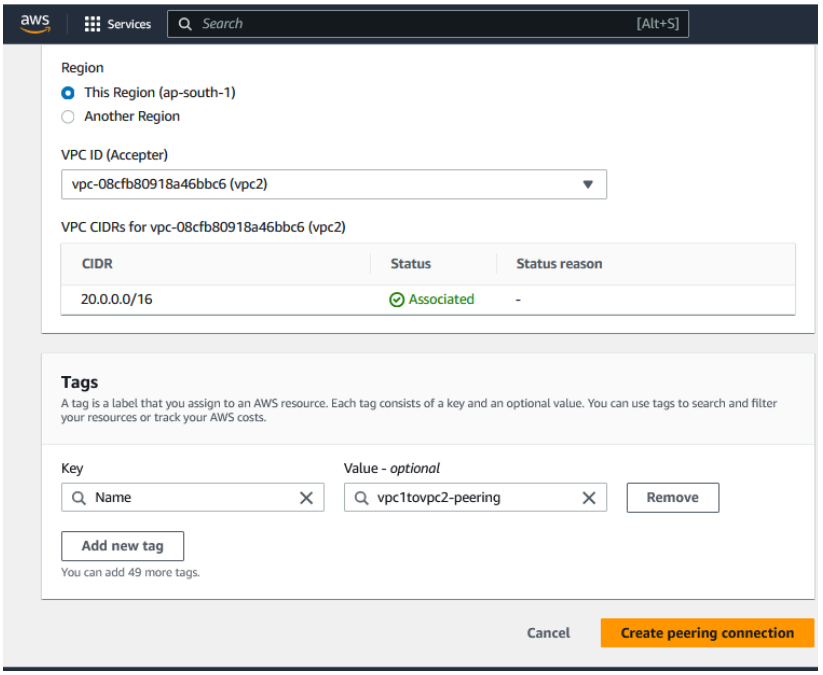
Peering connection -> Create peering connection



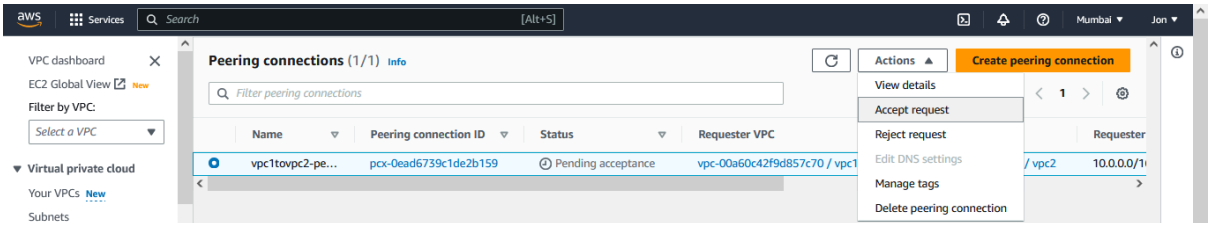
Requestor -> vpc1

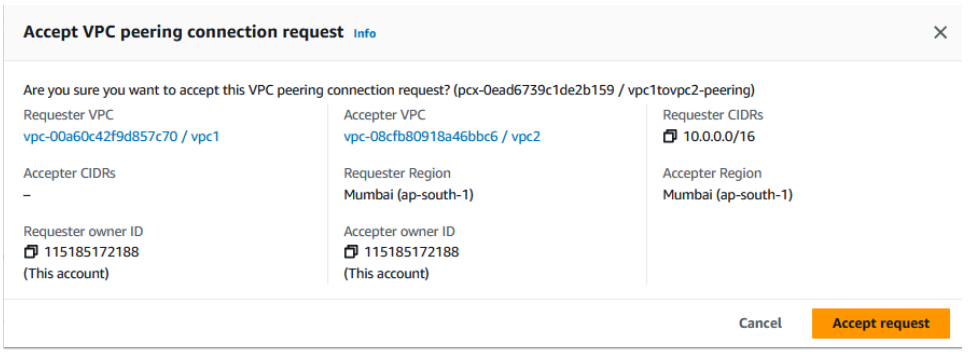
Acceptor -> vpc2





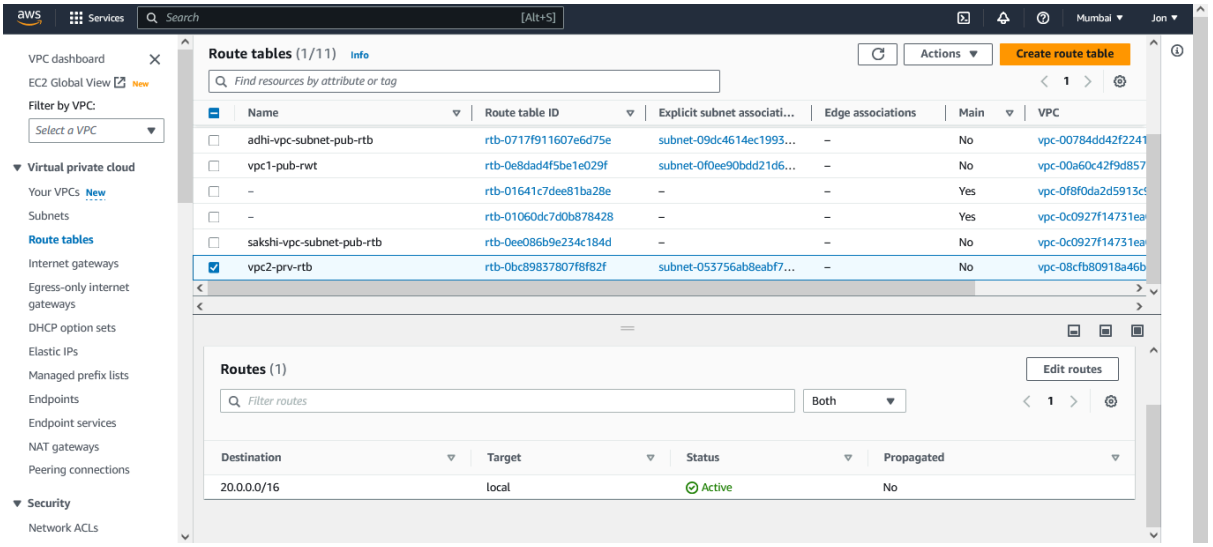
Accept

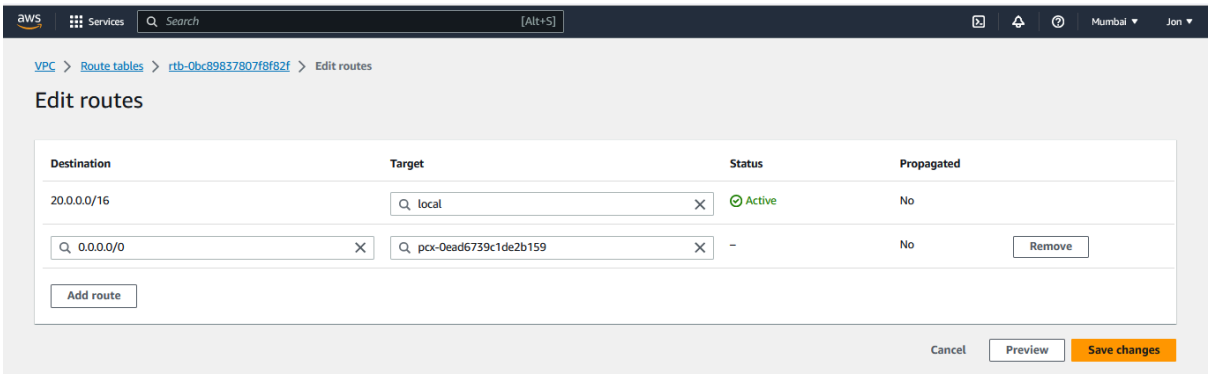




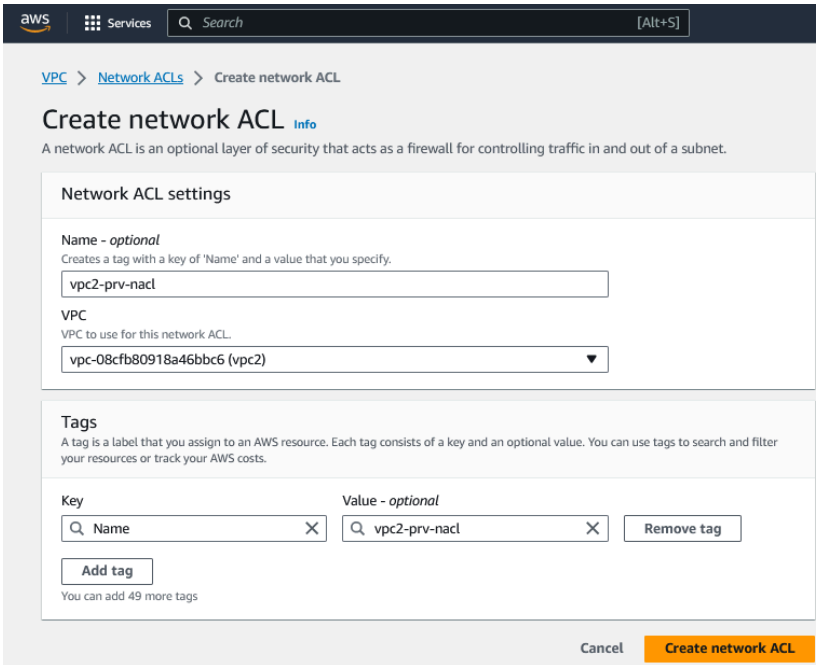
Step 5:- Route table association

Edit routes

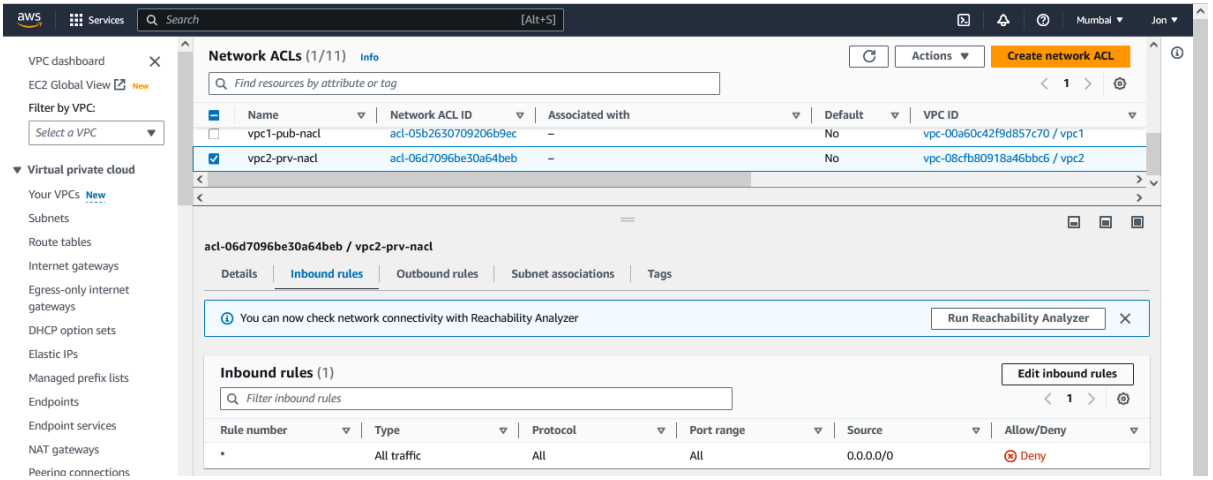


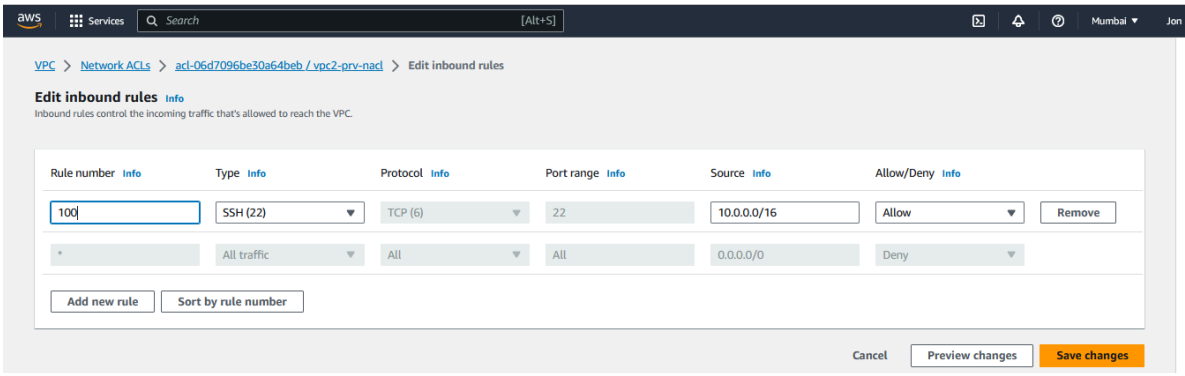


Step 6:- Create nacl

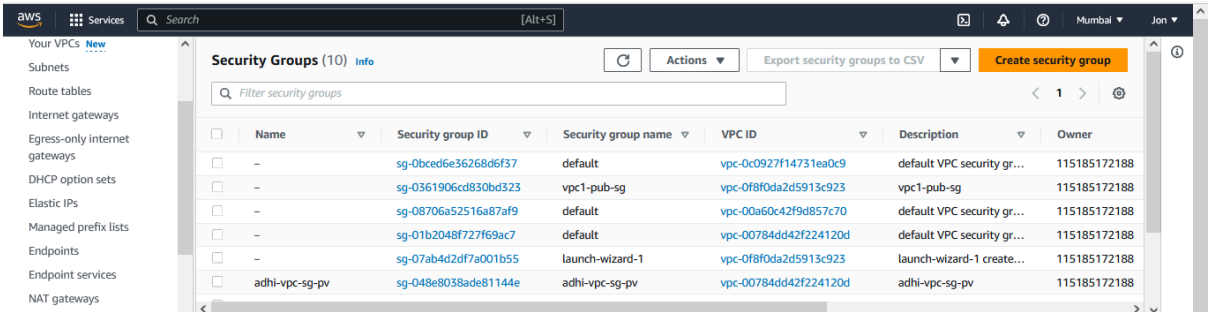


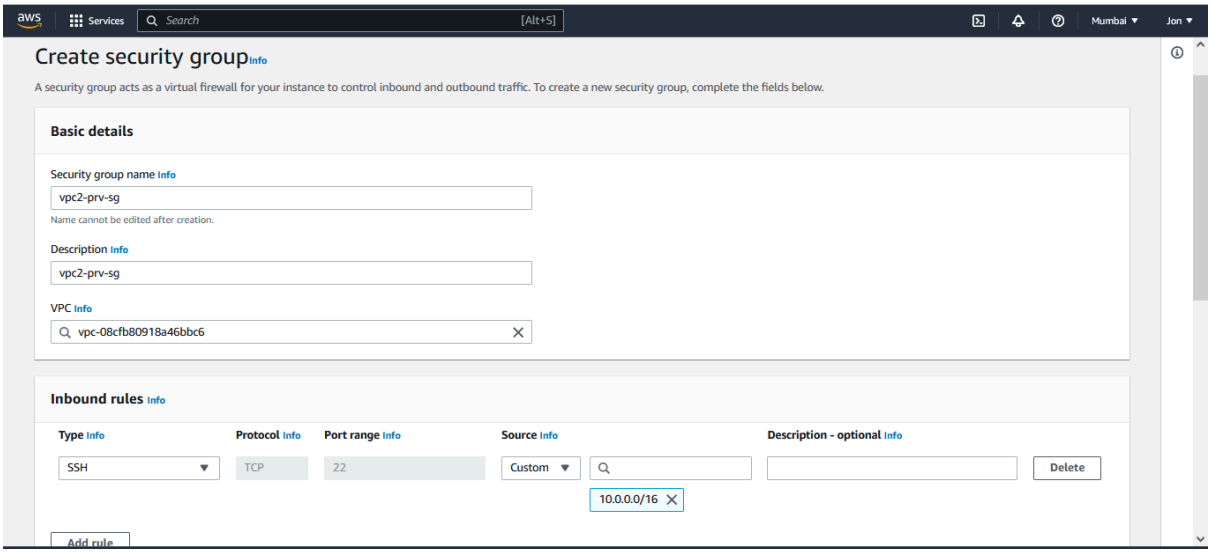
Edit nacl inbound rules

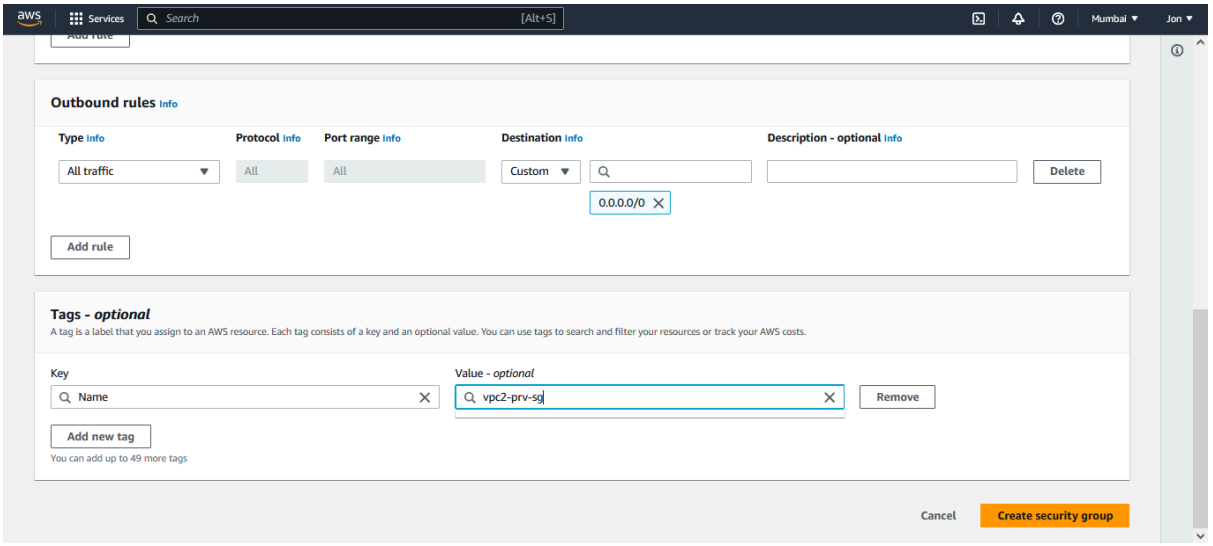




Step 7:- Create a security group







—------

Creating instances

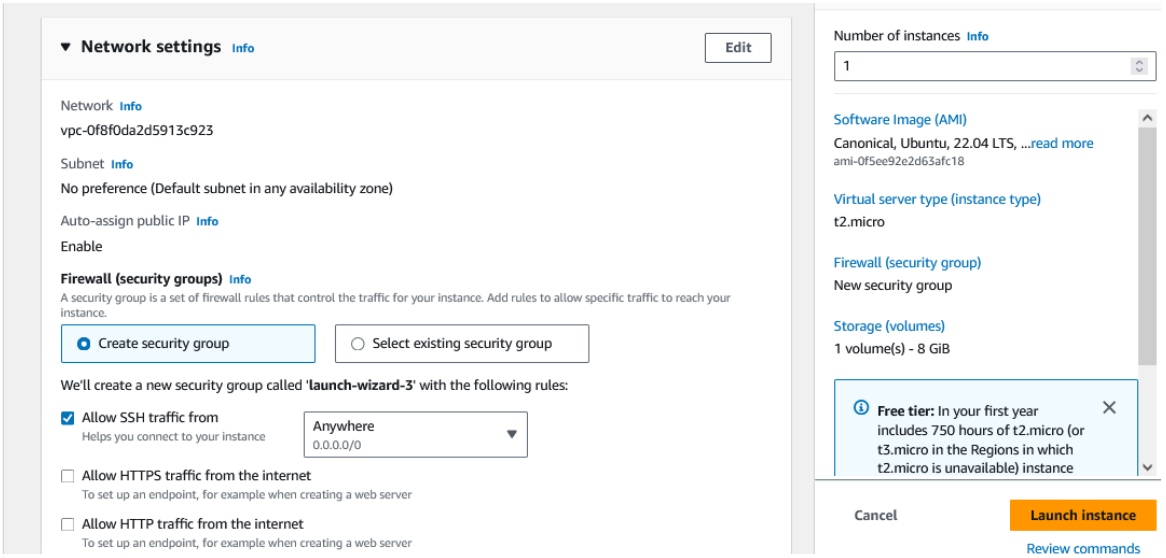
Public Instance

Launch instance

Step 1:-Now giving name as :- vpc1-public-instance

Step to Launch instance are same , only the difference is in Network settings

Ec2-> Launch instance -> Network settings ->Edit

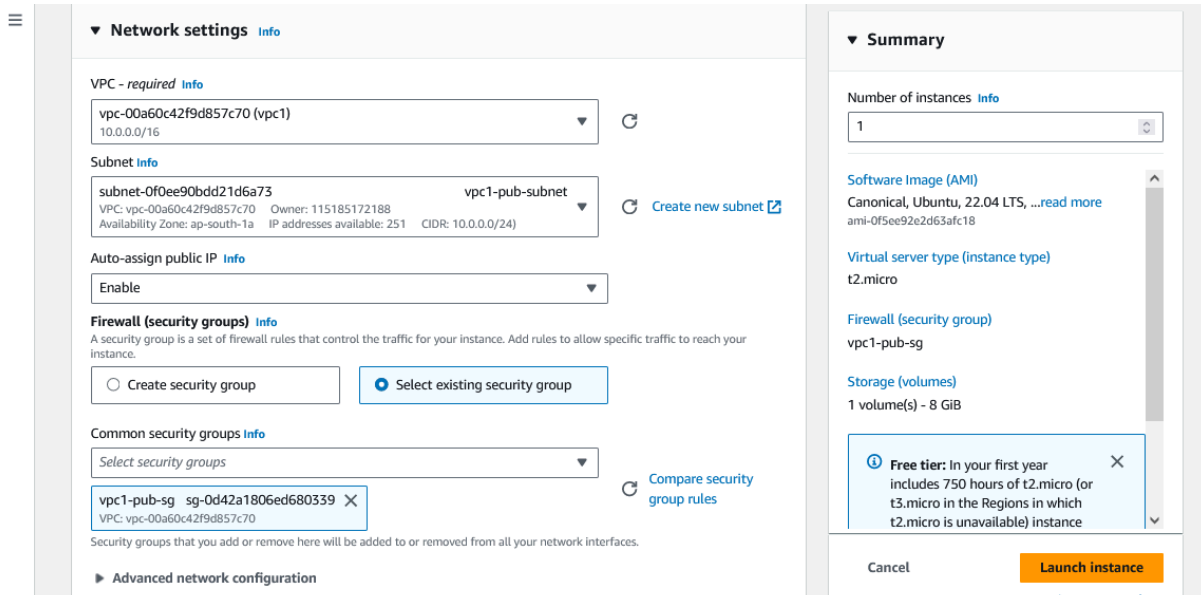


VPC -> select vpc1

Subnet -> select subnet1

Auto assign public IP -> Enable

Firewall:- select existing sg -> vpc 1



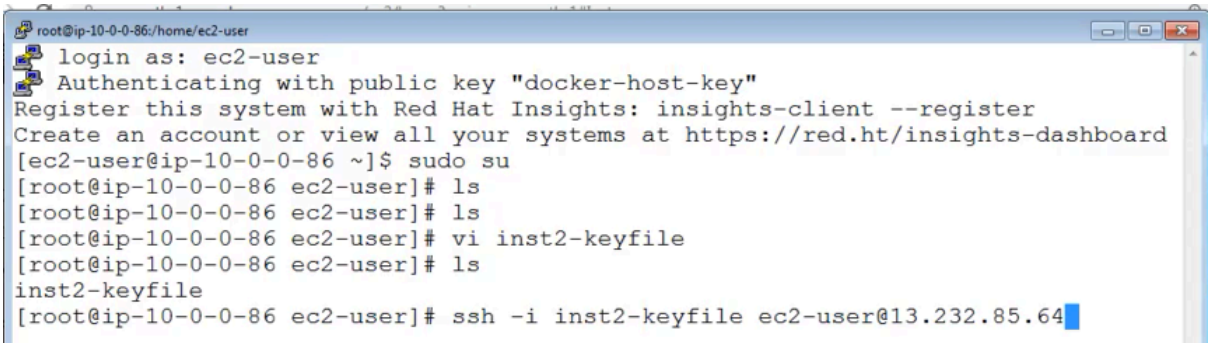
Follow the same process and create private instance (Select vpc2)

Step 2:-Now giving name as:- vpc2-private-instance

Follow the same process and create private instance (Select vpc2)

Step 3:- Do ssh means putty related configuration of vpc1-pub-instance

Now you have to connect from instance 1 to instance 2 .i.e. from vpc1 instance we want to connect to vpc2 instance



(Public ip of vpc2)

Run the above commands on vpc1-public-terminal