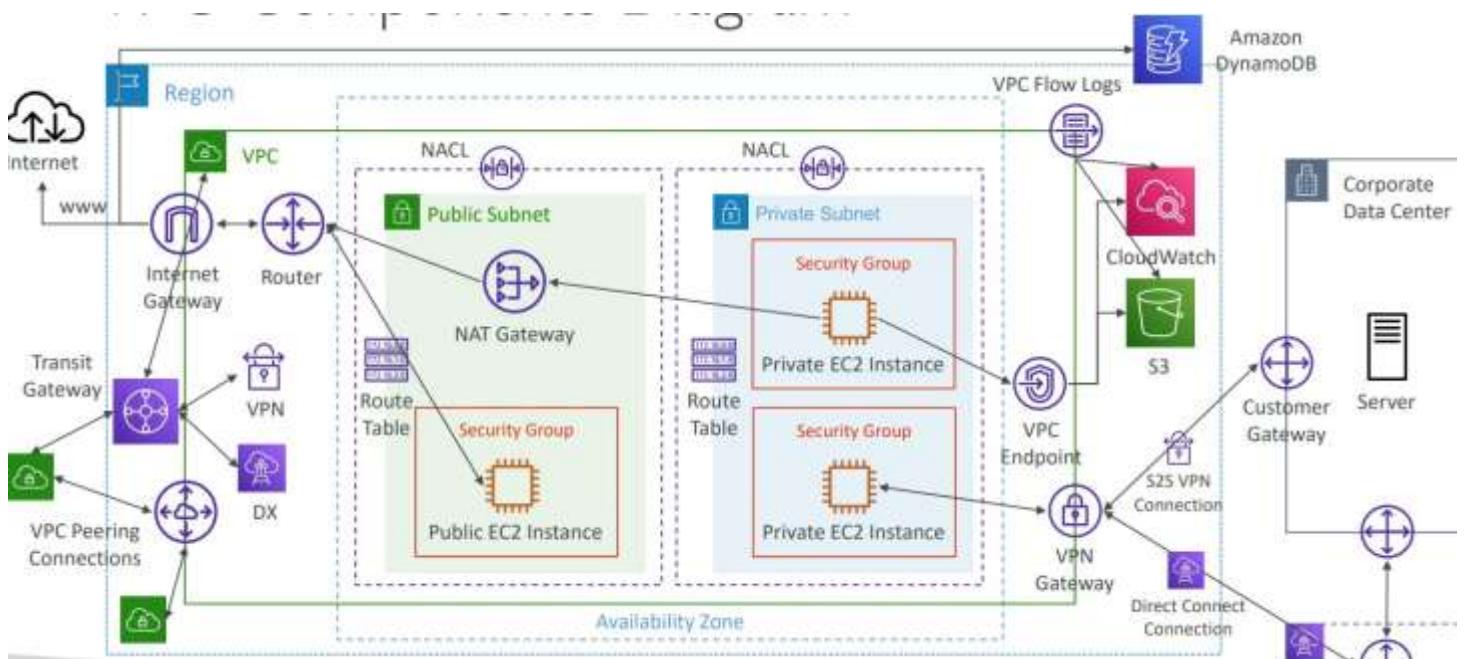


# Creating an AWS VPC from Scratch

## (ENTERPRISE GRADE)



This is the final VPC which I will construct step by step

# **VPC**

Stands for Virtual Private Cloud

In this guide, I'll create a custom VPC in AWS from scratch, including subnets, route tables, Internet connectivity, and essential components. This provides better control and security than using the default VPC.

This VPC **WILL BE** an Enterprise grade VPC with options and features which ensure

1. High Availability and scalability
2. Resilience
3. Security
4. High performance and traffic spiking workloads

Let's make it from scratch

## **Step1.**

Creating a VPC and establishing subnets in a **region**

VPC as stated is a virtual private cloud in short like an independent society with residents not dependent on the outside for any resources.

Imagine Subnets like the houses in the Neighborhood Some our public like movie halls and parks

Rest are private like residential Homes



Screenshot of the AWS VPC Subnets management interface. The top navigation bar includes "AWS", "Search", "Actions", "Create subnet", and "Region: Asia Pacific (Mumbai)". The left sidebar shows "VPC dashboard" and "Virtual private cloud" sections with various sub-options like "Subnets", "Route tables", "Internet gateways", etc. The main content area is titled "Subnets (6)" and contains a table with the following data:

Subnet ID	State	VPC	Block Public... (radio)	IPv4 CIDR	IPv6 CIDR
subnet-0f7e486d2b3fbcc5d	Available	vpc-05d1f5c9b8e0c12be   defa...	Off	10.0.1.0/24	-
subnet-0f5ef0a1c504b2b5d	Available	vpc-02edcb56c281e19e9   defa...	Off	172.31.0.0/20	-
subnet-096d3b467e2bc4d8	Available	vpc-02c0cb56c281e19e9   defa...	Off	172.31.16.0/20	-
subnet-09d288e1052159647	Available	vpc-05d1f5c9b8e0c12be   defa...	Off	10.0.0.0/24	-
subnet-07a84f294ac1ea07	Available	vpc-02c0cb56c281e19e9   defa...	Off	172.31.32.0/20	-
subnet-05548e7a5003d1ff6	Available	vpc-05d1f5c9b8e0c12be   defa...	Off	10.0.2.0/24	-

Screenshot of the AWS VPC Subnets interface for a specific VPC named "demo-vpc". The top navigation bar shows "VPC Show details" and "Your AWS virtual network". The main content area is titled "Subnets (3)" and lists three subnets under the heading "Subnets within this VPC":

- ap-south-1a**: newBabySubnet (marked with a blue square icon)
- ap-south-1b**: babysubnet2 (marked with a green circle icon)
- ap-south-1c**: newbabysubnet3 (marked with a blue square icon)

## STEP2

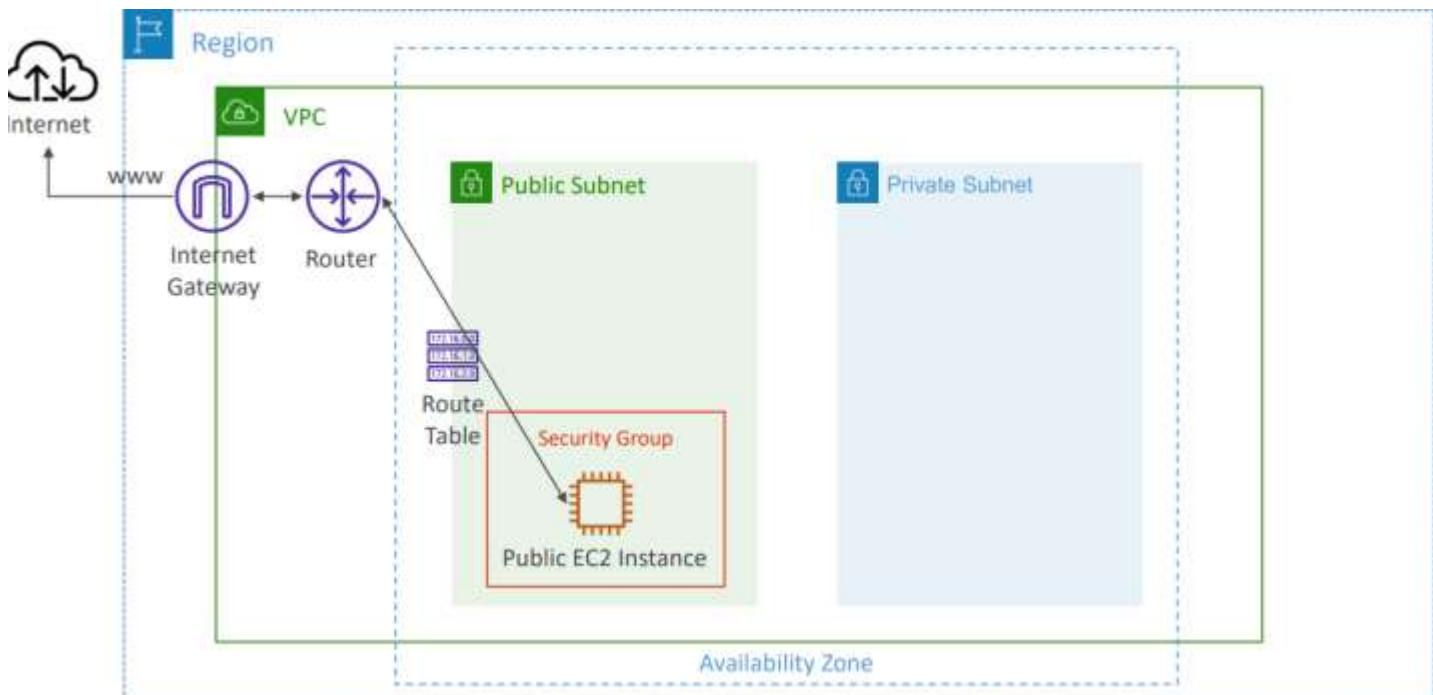
### **Internet GateWay**

Now that the neighborhoods created, we require a connection for import exports or in cloud terms Internet Access to the instances containing the app

I created an internet Gateway now the diagram is like this

Note:

- It scales horizontally and is highly available and redundant
- Must be created separately from a VPC
- One VPC can only be attached to one IGW and vice versa
- Internet Gateways on their own do not allow Internet access...
- Route tables must also be edited!



Below shows the attached Internet Gateway  
and the route table records being attached to the Internet Gateway

PUBLIC SUBNET	PRIVATE SUBNET
Its personal Route table is created pointing the Internet Gateway	Its personal Route table is created either pointing to a Bastion Host or private networks only

The screenshot shows the AWS VPC Internet Gateways console. The main pane displays the details of an Internet Gateway named "igw-Off212bc85a35b62e". Key information includes:

- Internet gateway ID:** igw-Off212bc85a35b62e
- State:** Attached
- VPC ID:** vpc-03d15a0bæ9c526e | demo-vpc
- Owner:** 254159011250

The "Tags" section shows a single tag: Name = demoInternetGateway.

Below the main pane, there are tabs for "Routes", "Subnet associations", "Edge associations", "Route propagation", and "Tags". The "Routes" tab is selected, showing three routes:

Destination	Target	Status	Propagated
0.0.0.0/0	igw-Off212bc85a35b62e	Active	No
10.0.0.0/16	local	Active	No
172.31.0.0/16	pxx-00ca29223482e24c7	Active	No

So now you might be having a doubt?

I want to keep my applications private so that they are visible only to me. For those, I will not give a connection to the Internet Gateway. So, from where will they get the network?"

# Bastion Hosts (Step-2.5)

These are the types of instances which are hosted on public subnets have access and are visible to the public and can operate resources to the private subnet instances simply but doing SSH into those instances

Here I have

1. Created instance amid bastion host in public subnet and one in private subnet
2. Modified the security group inbound rules of private instance to allow traffic on port 22(SSH Port) to point FROM THE SECURITY GROUP OF BASTION HOST
3. Now I ssh into the Bastion host
4. From here I ssh into the private instance – make a .pem file ,import it in the terminal itself and echo a “Hello” in Private instance
5. As soon as I curl cmd(For calling) it in the public instance it returns the desired output

The screenshot shows the AWS EC2 Instances page. A single instance is selected, with its ID being I-03750df17e07efeb. The instance is running an Amazon Linux 2023 AMI. It has a public IP of 13.232.248.169 and a private IP of 10.0.1.60. The instance is associated with a security group sg-01e1592c1232ef243 (Private-50). In the Security tab, there is an inbound rule for port 22 (TCP) from the Private SG (sg-02e7f0850ef46715) to the instance. The instance was launched on Thu Jul 24 2025 14:51:12 GMT+0530 (India Standard Time).

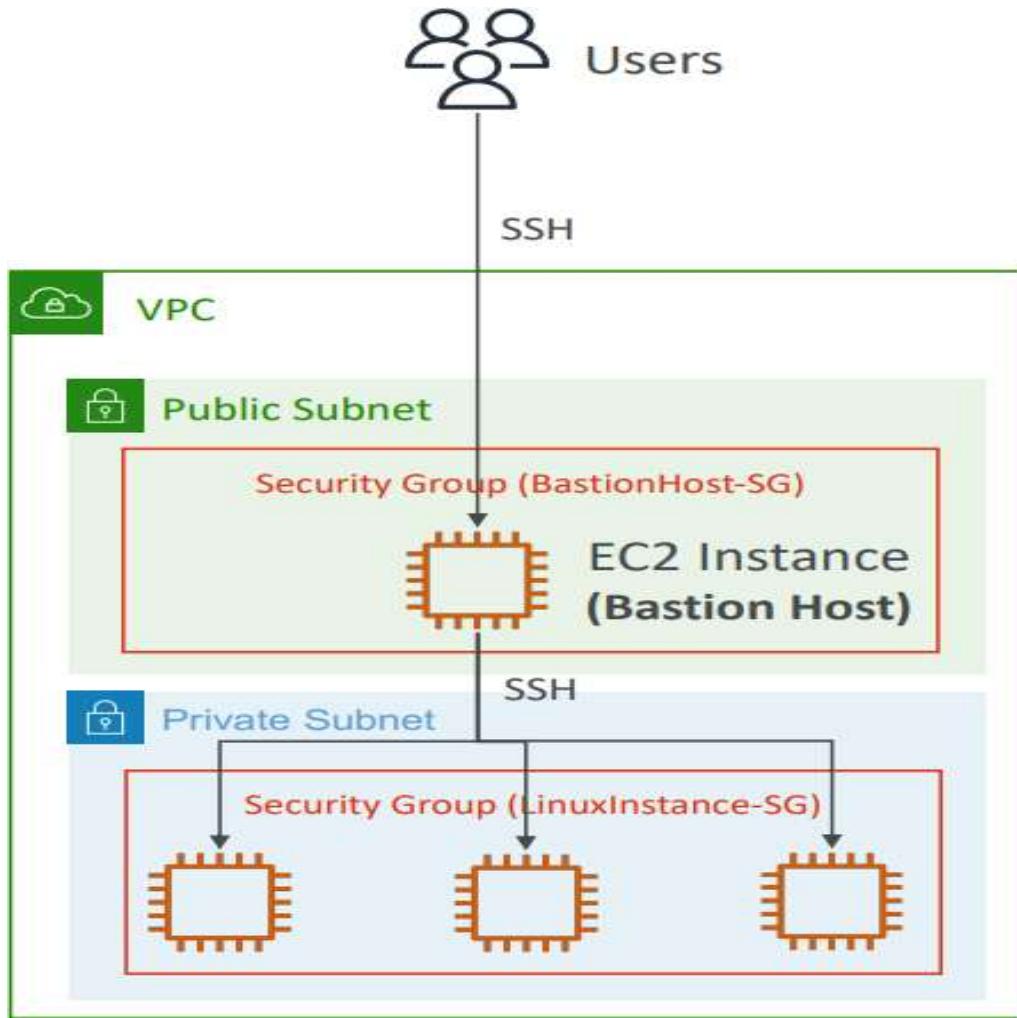
```
ssh -i /root/.ssh/AmazonLinux2023.pem ec2-user@10.0.0.54
The authenticity of host '10.0.0.54 (10.0.0.54)' can't be established.
ECDSA key fingerprint is 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00.
This key is not known by any other name.
Are you sure you want to continue connecting (yes/no)? yes
Warning: permanently added '10.0.0.54' (00:00:00:00:00:00) to the list of known hosts.
```

The terminal session shows the user attempting to SSH into the private instance using the provided PEM key. The host key fingerprint is displayed, and the user confirms the connection. The instance's name is listed as 'Amazon Linux 2023'.

Expected Output:

```
[ec2-user@ip-172-31-3-7 ~]$ curl 10.0.1.50
hello
[ec2-user@ip-172-31-3-7 ~]$ █
```

NOW THE Structure LOOK LIKE THIS



# Step 3

Increasing level

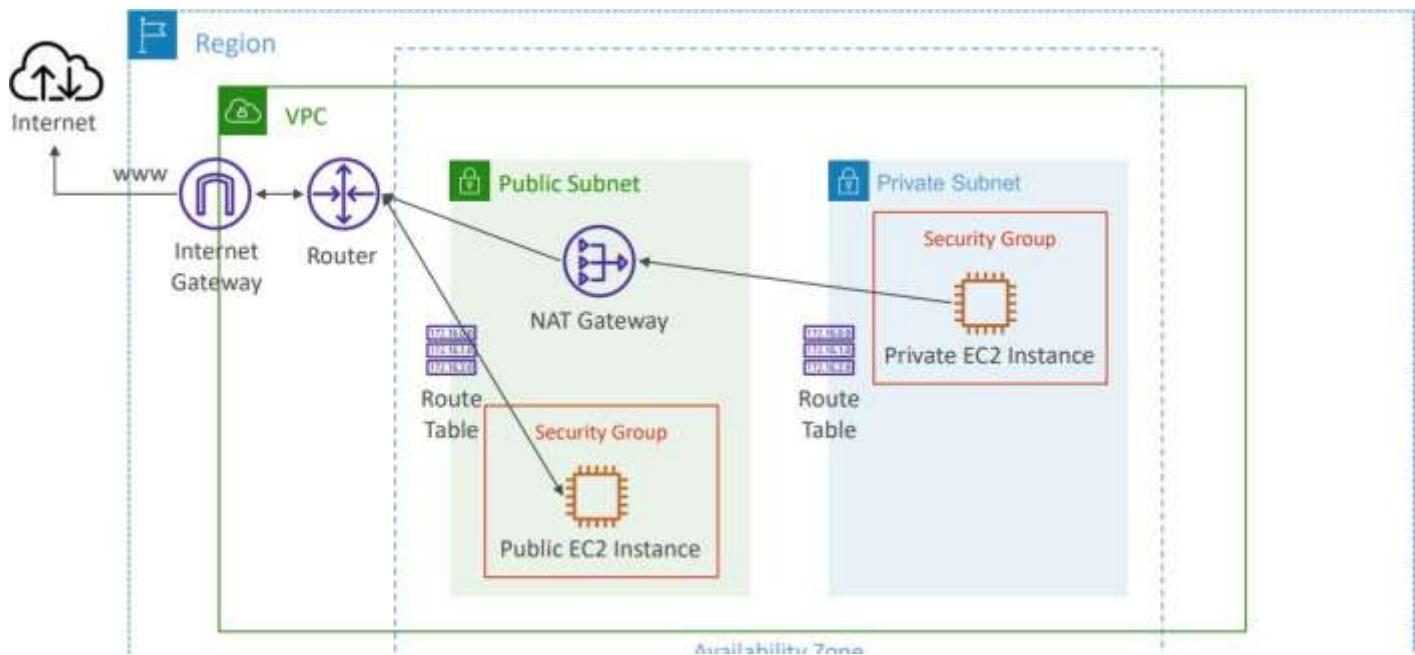
This method is great but not reliable for bigger workloads so

## NAT GATEWAY

This thing ensures a stable internet connection both in and out be given to private subnets with a window for security checks of!

AWS-managed NAT, higher bandwidth, high availability, no administration

- NATGW is created in a specific Availability Zone, uses an Elastic IP
- Can't be used by EC2 instance in the same subnet (only from other subnets)
- Requires an IGW (Private Subnet => NATGW => IGW)
- 5 Gbps of bandwidth with automatic scaling up to 100 Gbps



Now Comes the Security Part to secure the infrastructure for

1. Malicious Traffic
2. Misguided Traffic

## Step-4

### NACL(s) /Security Groups

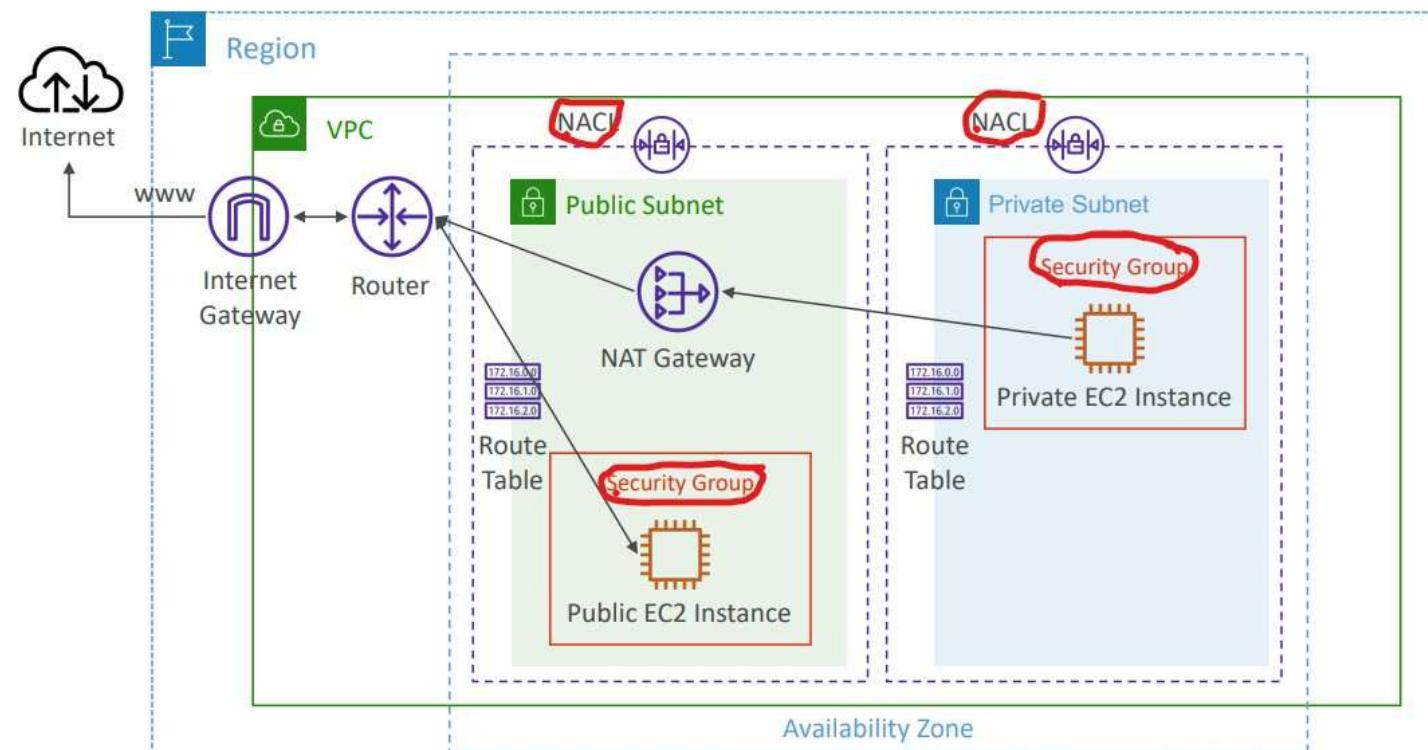
NACL(s) are Subnet Level completely stateless (Check both in and out traffic) and are based on Rules priority no's.

Security groups are instance level to further prevent the misguided traffic to reach its correct path and may not activate a wrong instance these are stateful (logical people if in then out and vice versa types) and have no priority no. System just declarations

There are many places where I used NACLs to block certain CIDRs

And off those Which enter inside to guide through security groups

So here is our progress



## **NOTE:**

**EPHIMERAL PORTS**- these are on spot usable ports used instead of defined ports for communication for added security.

Now If the app is Enterprise grade and the infrastructure is also Enterprise grade It cannot rely on only 1 VPC

We have to have multiple VPC but that is not the point that is simple the point is **WE NEED SYNC** and updated Information between all the VPCs for the companies Also an Advantage of this will be to share info across VPC ,,,next Feature, so that's why we are using VPC peering.

## **Step 4**

### **VPC Peering**

It simply means establishing a peering connection or simply a connection between VPCs **CROSS ACCOUNT & CROSS REGION**

Here I set up

- 1. TWO VPCs in different region**
- 2. Accept the request of peering**
- 3. MOST IMP. ALWAYS update it in the route tables which I will see in common once both VPCs are connected as shown**

## Accept VPC peering connection request

Info

Are you sure you want to accept this VPC peering connection request? (pcx-00ca29223482e24c7 / my-pc-1)

Requester VPC

vpc-03d1f5a9b8e9c526e / demo-vpc

Acceptor CIDRs

-

Requester owner ID

254159011250 (This account)

Acceptor VPC

vpc-02c0cb56c281c19c9 / defaultVPC

Requester Region

Mumbai (ap-south-1)

Acceptor owner ID

254159011250 (This account)

Requester CIDRs

10.0.0.0/16

Acceptor Region

Mumbai (ap-south-1)

Cancel

Accept request

VPC > Peering connections > Create peering connection

Select a local VPC to peer with

VPC ID (Requester)

vpc-02c0cb56c281c19c9 (defaultVPC)

VPC CIDRs for vpc-02c0cb56c281c19c9 (defaultVPC)

CIDR	Status	Status reason
172.31.0.0/16	<input checked="" type="radio"/> Associated	-

Select another VPC to peer with

Account

My account  
 Another account

Region

This Region (ap-south-1)  
 Another Region

VPC ID (Acceptor)

vpc-02c0cb56c281c19c9 (defaultVPC)

VPC CIDRs for vpc-02c0cb56c281c19c9 (defaultVPC)

CIDR	Status	Status reason
172.31.0.0/16	<input checked="" type="radio"/> Associated	-

Tags

## Changing the route Table

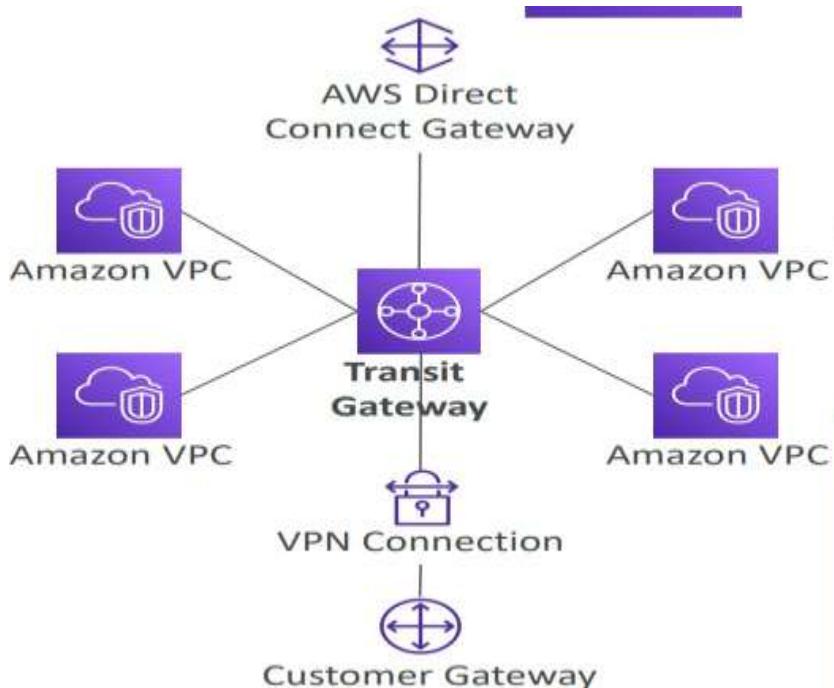
The screenshot shows the 'Edit routes' section of the AWS VPC Route Tables interface. It displays two routes:

- Route 1:** Destination 10.0.0.0/16, Target local, Status Active.
- Route 2:** Destination 172.31.0.0/16, Target Peering Connection, Status In Progress. The target dropdown shows "pcx-00ca29223482e24c7" selected, with a tooltip "Use: 'pcx-00ca29223482e24c7'" and "pcx-00ca29223482e24c7 (my-pc-1)".

## **ADDITIONAL STEP FOR ENTERPRISE GRADE**

Setup a Transit Gateway for incase of connecting multiple VPCs together using peering

**Reason:** Network Topology becomes **complicated** and difficult to maintain



# Some Cost Optimizing approaches....

Okay So Now the thing is I want my Private Instance to access AWS services or my private DATA Center

Why should i incur such costs of setting up a NAT gateway then accessing those services through public

That's expensive and not even secure

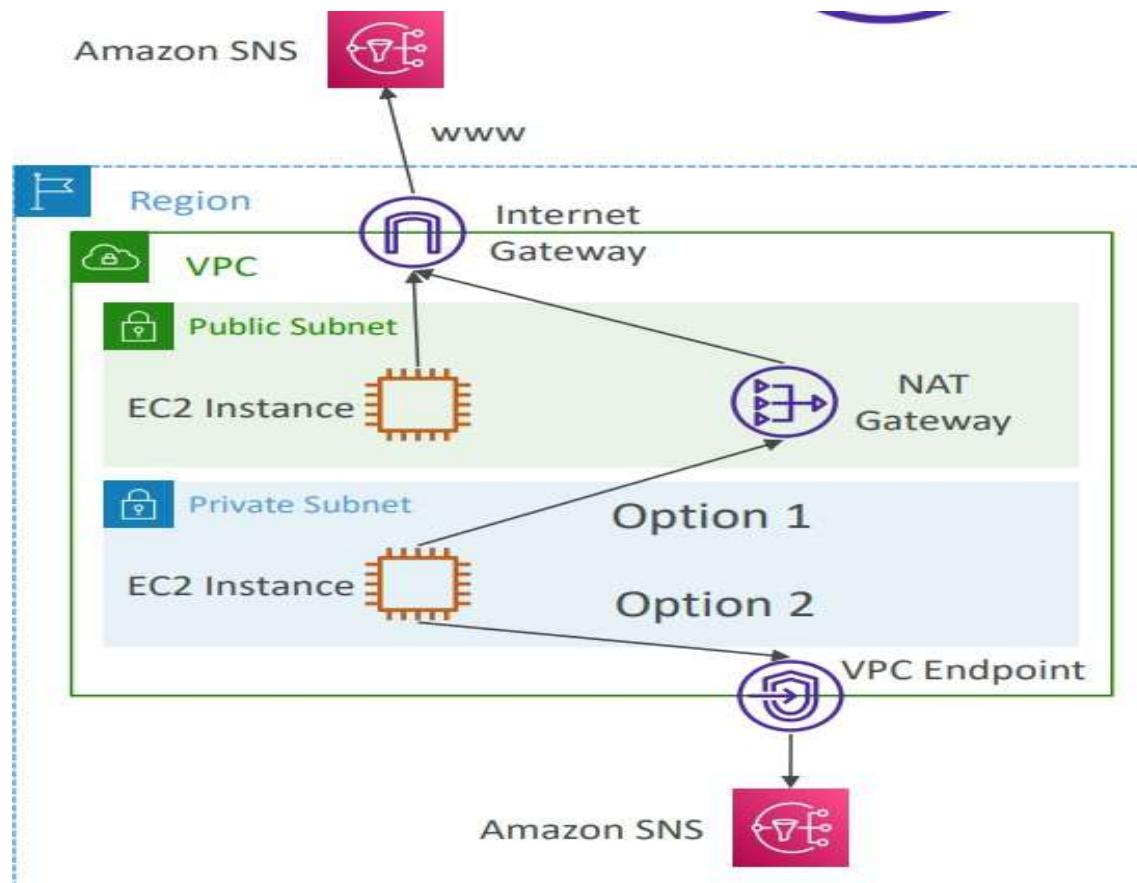
Hence

## Step 5

### **VPC Endpoints**

These are like metro connecting dots 

VPC Endpoints (powered by AWS Private Link) allows you to connect to AWS services using a private network instead of using the public Internet



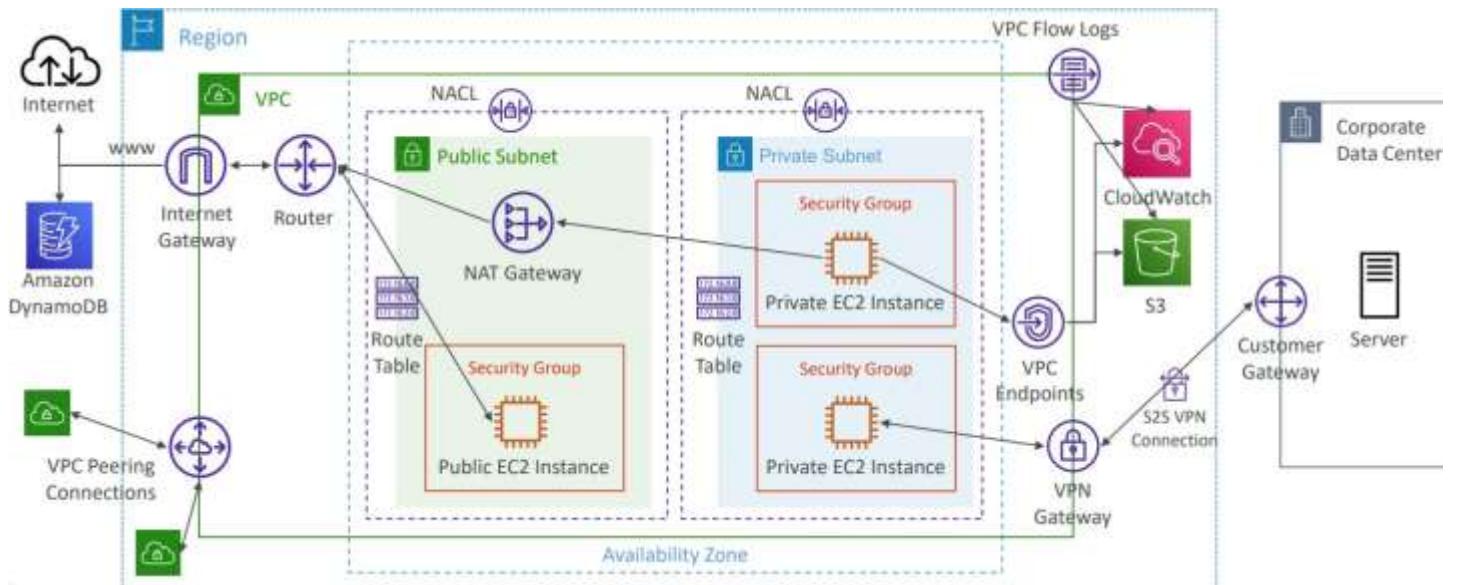
## **Now this is what I meant**

These are also of two types one is a gateway endpoint supporting less services (FOR Small companies)

2. Is Interface endpoints (Basically Endpoint premium subscription for Enterprises)

**Finally.... my VPC is all set !!!!**

**A final step remains.....**



# THANK YOU