VPC

Topics we'll explore in this guide :

🔹 Introduction to VPC

- What is a VPC?
- Analogy: VPC as a private city in a country (AWS Region)
- Why we need a VPC

🔹 Why Do We Need a VPC?

- Isolation from other AWS users
- Custom network setup
- Enhanced security
- Internet access control

🔹 VPC Key Components

1. **CIDR Block**
2. **Subnets** (Public and Private)
3. **Route Tables**
4. **Internet Gateway (IGW)**
5. **NAT Gateway**
6. **Security Groups & Network ACLs (NACLs)**

7. **VPC Peering & VPN/Direct Connect**

🔹 Getting Started with VPC

- Default VPC setup by AWS
- VPC vs Internet
- Types of networks in cyberspace
- Manual IPv4 CIDR block input

🔹 Creating a Subnet in VPC

- VPC ID selection
- Naming subnet
- Selecting availability zone
- CIDR block for subnet (e.g., /24)
- Enabling auto-assign public IPv4 address
- Importance of enabling it for public subnets

🔹 CIDR Explanation

- /16 and /24 subnet sizing
- IP calculation (total vs usable)
- AWS reserved IPs in each subnet

🔹 Internet Gateway (IGW)

- Creating and attaching an IGW
- Making the VPC internet-accessible

- Role of IGW in public subnet access

🔹 Task 2: VPC and Subnet Creation

- Creating a VPC with a /16 range
- Creating two subnets:
  - Public (10.0.0.0/24)
  - Private (10.0.1.0/24)
- Route table and NACL default attachments
- Creating and assigning a new Route Table for public subnet
- Separating public and private subnets via Route Tables
- Adding IGW to public subnet's route table
- Adding 0.0.0.0/0 route for internet access

🔹 Launching EC2 Instances

- Public EC2 in the public subnet
- Private EC2 in the private subnet

What is vpc and why do we need it?

If we imagine our AWS Region as a country, a Virtual Private Cloud (VPC) is like your own private city inside that country.

We can design neighborhoods, traffic rules, and security measures to control how resources, like EC2 instances and databases, connect and work together.

AWS Virtual Private Cloud (VPC) is a private, isolated network within AWS where we can launch resources like EC2 instances, databases, and load balancers.

VPC is like a own data center in the cloud with full control over:

✅ IP Addressing & Subnetting

✅ Internet & Private Access

✅ Security & Firewalls (Security Groups & NACLs)

✅ Connectivity (VPN, Direct Connect, Peering)

Why Do We Need a VPC?

- **Isolation** → Your AWS resources are protected from other AWS customers.
- **Custom Network Setup** → You define **IP ranges, subnets, and routes**.
- **Security** → Use **firewalls, encryption, and VPNs** for secure communication.
- **Internet Control** → Choose which resources can access the internet.

VPC Key Components

Here's what makes up a **VPC**:

1️⃣ CIDR Block (IP Range)

- A **VPC requires an IP range** in **CIDR notation** (e.g., `10.0.0.0/16`).
- **Example**: `10.0.0.0/16` provides **65,536 IP addresses**.

2️⃣ Subnets

- A **VPC is divided into subnets** (smaller network sections).
- **Public Subnet** → Directly connected to the internet (via an **Internet Gateway**).
- **Private Subnet** → No direct internet access (used for databases & backend servers).

3️⃣ Route Tables

- **Control how traffic flows** between subnets and the internet.
- Each **subnet must be associated with a route table**.

4️⃣ Internet Gateway (IGW)

- A **bridge between VPC and the internet**.
- Required for public subnets to have internet access.

5️⃣ NAT Gateway (For Private Subnets)

- **Allows private subnets to access the internet** without exposing them.
- **Example**: Private EC2 instances can download updates from the internet but remain hidden.

6️⃣ Security Groups & Network ACLs

- **Security Groups** → Work at the **instance level** (like a firewall).
- **Network ACLs (Access Control Lists)** → Work at the **subnet level** (rules for inbound/outbound traffic).

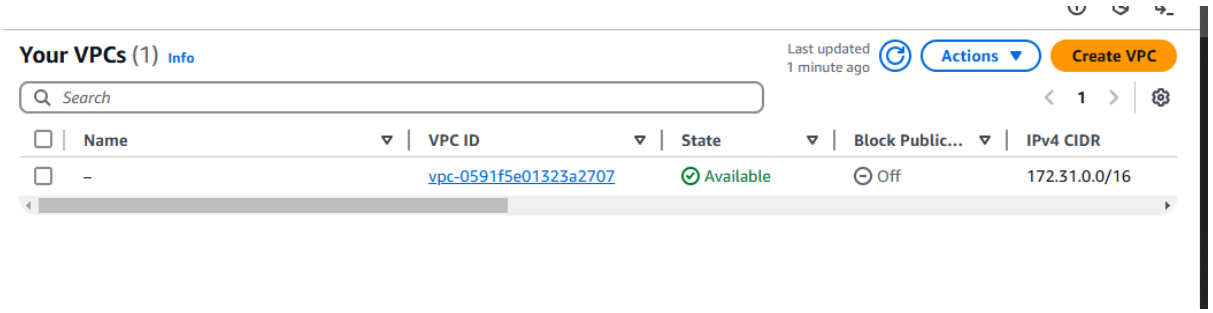
7️⃣ VPC Peering & VPN

- **VPC Peering** → Connects two VPCs **securely**.
- **VPN / Direct Connect** → Connects your on-premises network to AWS securely.

VPCs are the reason why resources can be made **private** to you. You also get control over resources in a VPC, so you can organize how they communicate and integrate with each other **without the public internet**.

Getting started with VPC:

Search for vpc in aws console -> go to Your VPCs.



A vpc will be there already

When you create your AWS account, AWS automatically sets up a **default** VPC for you! This default VPC is why you could launch resources (e.g. EC2 instances) and connect services together from Day 1 of using AWS.

(we can use *some* AWS services like Amazon S3 or AWS Lambda without setting up a VPC. These services are designed to work on the internet **without** needing a private network setup.

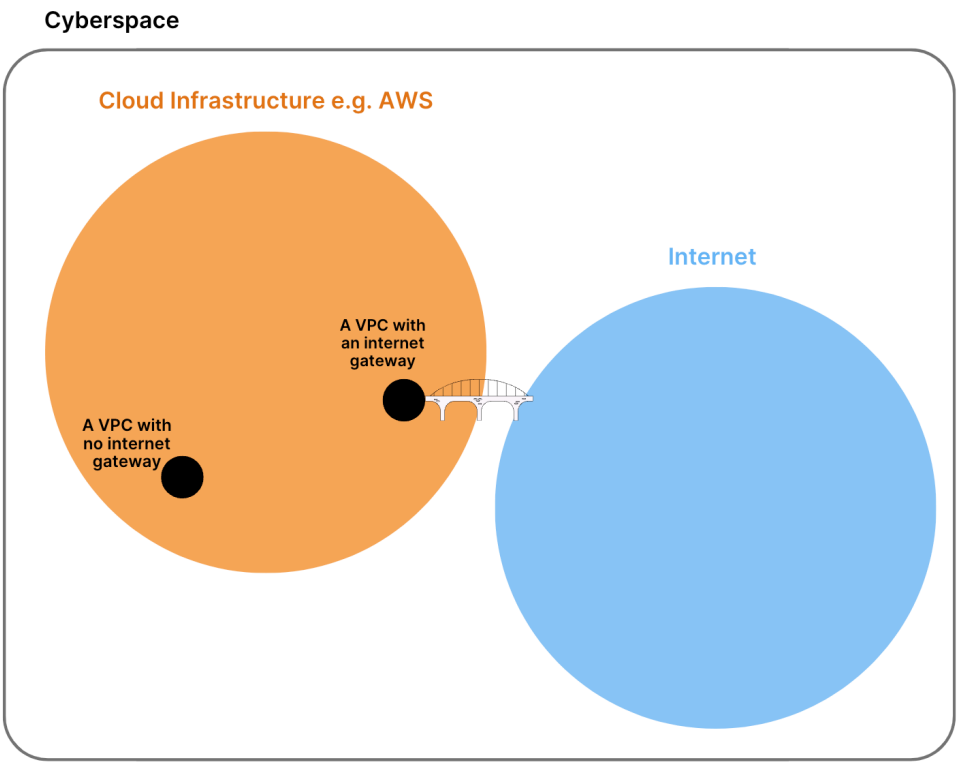
However, other services like Amazon EC2 or certain databases need a secure, isolated network to connect with each other and run securely. You would need a VPC in these cases.)

A **VPC is not the same as the internet**. While the internet is a **public, open network**, a **VPC is private and isolated** by default. Being "online" simply means connecting to a network for data exchange, but not all networks are public.

In the broader **cyberspace**, we have different types of networks:

- **Public internet** (accessible to everyone)
- **Private corporate networks**
- **Cloud infrastructures** (like AWS)

AWS provides **VPC as a private section** within its cloud. You can connect your **VPC to the internet** using an **Internet Gateway**, but a VPC itself remains **private** unless explicitly configured otherwise. 🚀



img src:https://www.nextwork.org/

Create VPC - > give name of the vpc - > ipv4 cidr manual input

Manual input : **IPv4 CIDR manual input in AWS** means that you manually specify the **IP address range** for your **VPC** instead of using a default one.

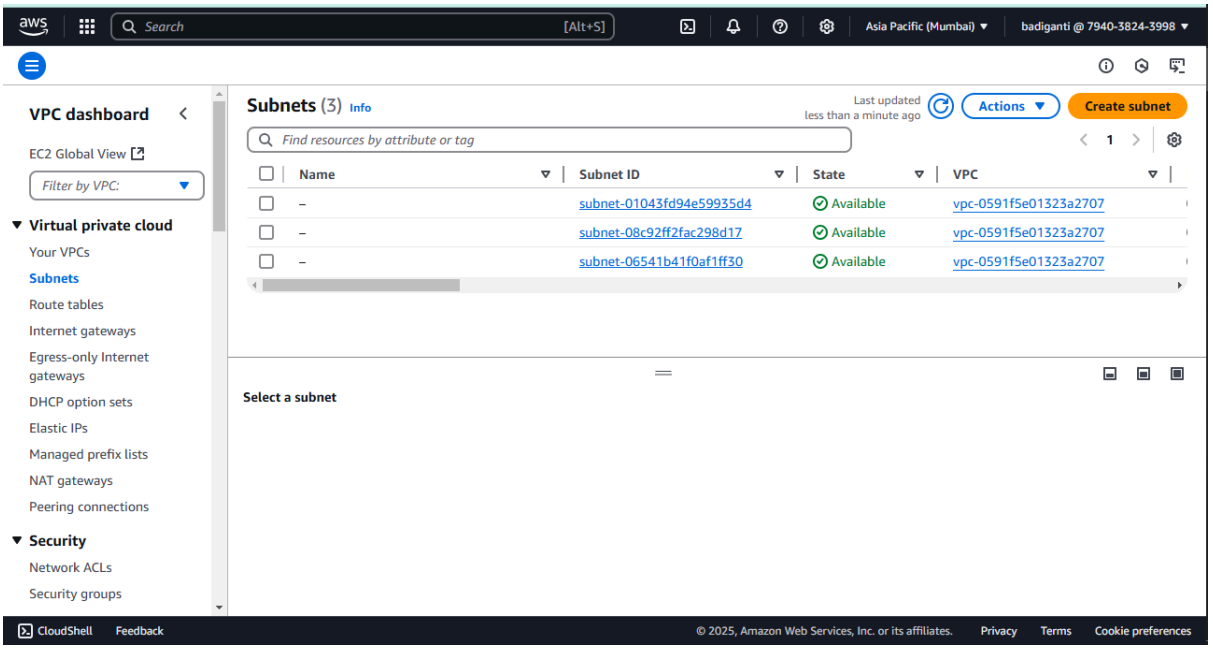
When creating a VPC in AWS, you must define an **IPv4 CIDR block**, which determines the range of private IP addresses that your VPC can use.

Example of Manual Input for IPv4 CIDR in AWS:

- `10.0.0.0/16` → Provides **65,536 IPs** (Large VPC)
- `192.168.1.0/24` → Provides **256 IPs** (Smaller VPC)

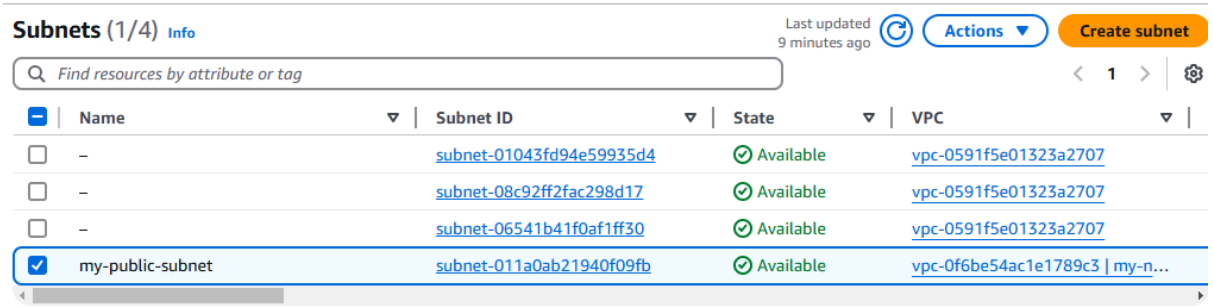
NOw we need to Launch a subnet inside out vpc :

In the **VPC Dashboard**, under **Virtual Private Cloud**, choose **Subnets**.



Create subnet -> Configure your subnet settings:

- **VPC ID:** my-ninja-network
- **Subnet name:** my-public-subnet
- **Availability Zone:** Select the first Availability Zone in the list.
- **IPv4 VPC CIDR block:** 10.0.0.0/16
- **IPv4 subnet CIDR block:** 10.0.0.0/**24**



- Check the box next to **Enable auto-assign public IPv4 address.**
- Choose **Save.**

Enabling **Auto-Assign Public IPv4 Address** in AWS subnet settings means that any **new EC2 instance** launched in that subnet will automatically receive a **public IPv4 address**.

Why Enable It?

✅ Allows instances to communicate **directly with the internet**.
✅ Eliminates the need to manually assign a **public IP**.
✅ Required for a **public subnet** (along with an Internet Gateway).

If disabled, instances will only have a **private IP**, requiring a **NAT Gateway** for outbound internet access.

CIDR -

Lets say we use CIDR block `10.0.0.0/16`

- A **/16 CIDR block** means that the **first 16 bits** of the IP address are fixed, and the remaining **16 bits** are available for host addresses.
- IPv4 addresses are **32-bit**, so a **/16 block** leaves **16 bits for hosts**:

```
2^16 = 65,536
```

If you use the **CIDR block `10.0.0.0/24`**, your subnet will have **256 total IP addresses**.

How is it calculated?

- A **/24 CIDR block** means that the **first 24 bits** of the IP address are fixed, leaving **8 bits** for hosts.
- IPv4 addresses are **32-bit**, so a **/24 block** leaves **8 bits for hosts**:

2^(32−24)=28=256 IPs 2^(32 - 24) = 2^8 = 256

Usable IPs After AWS Reservations

AWS **reserves 5 IPs per subnet**, so the usable IPs are:

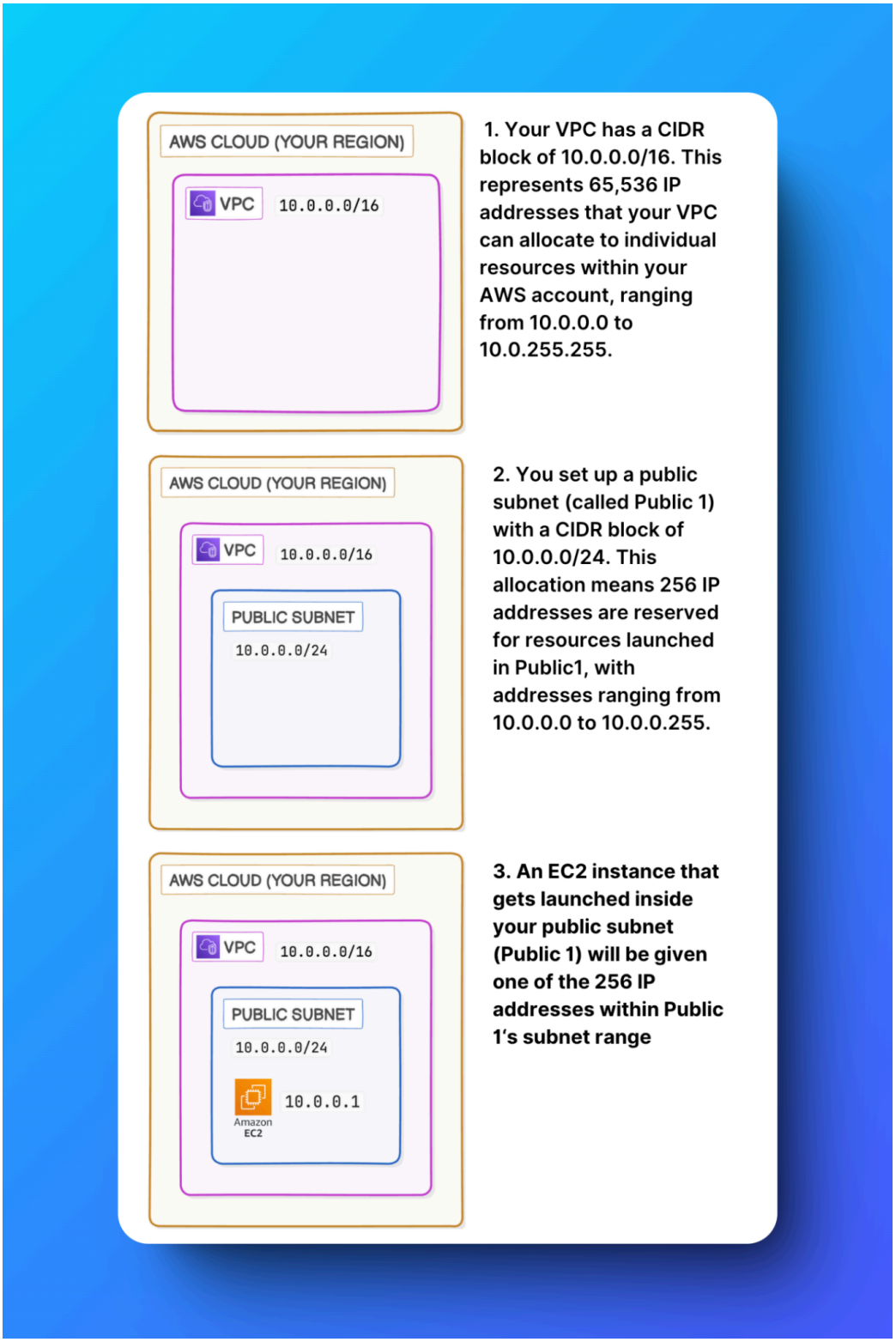
256−5=251 usable IPs256 - 5 = 251 \text{ usable IPs}

Reserved IPs in `10.0.0.0/24`

🔹 **10.0.0.0 → Network Address** (identifies the subnet)
🔹 **10.0.0.1 → VPC Router** (default gateway)
🔹 **10.0.0.2 → AWS DNS** (for name resolution)
🔹 **10.0.0.3 → Reserved for future use**
🔹 **10.0.0.255 → Broadcast Address** (not usable in AWS)

Final Answer:

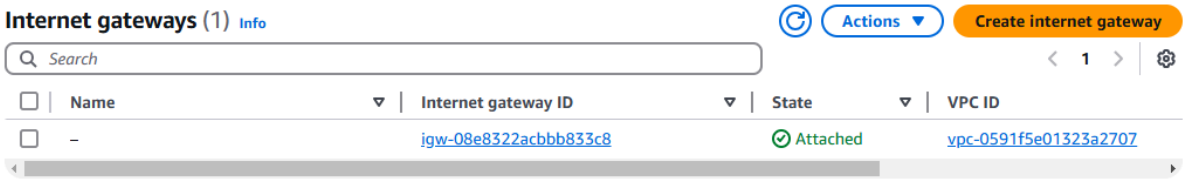
- **Total IPs: 256**
- **Usable IPs: 251** 🚀



VPC and subnet creation is now done.

Next we can create internet gateway ING

Go to internet gateways in the left menu and select it



We can see that there's already an ING by default.

It comes with the default VPC that AWS created for your account.

This default internet gateway is the reason why you could launch instances with a connection to the internet from the day you've created your AWS account.

Choose Create internet gateway.

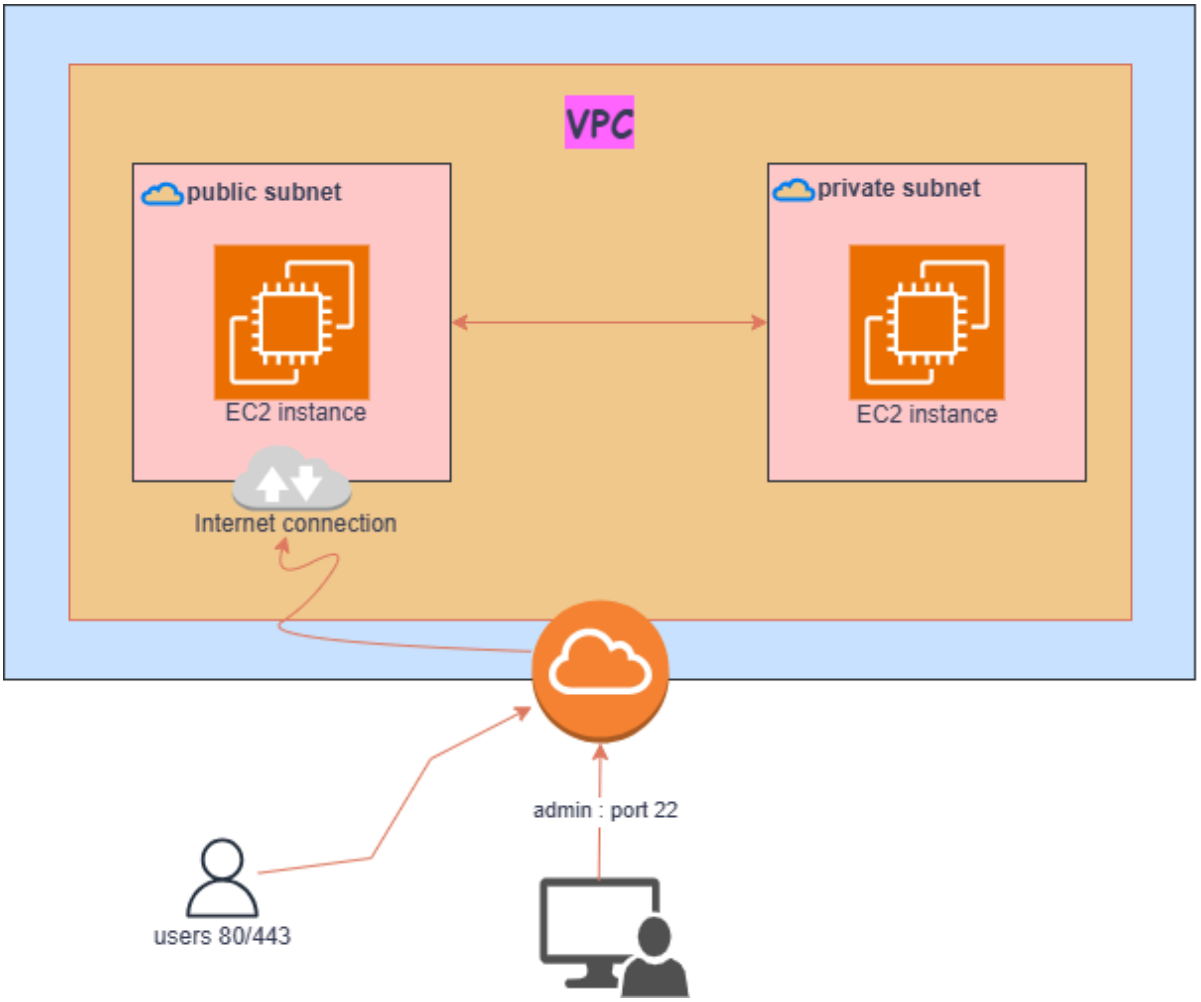
Name tag: my-gateway

Choose Create internet gateway.

Now attach this gateway to our vpc - *my-ninja-network*

Attaching an internet gateway means resources in your VPC can now access the internet. The EC2 instances with public IP addresses also become accessible to users, so your applications hosted on those servers become public too.

Task - 2



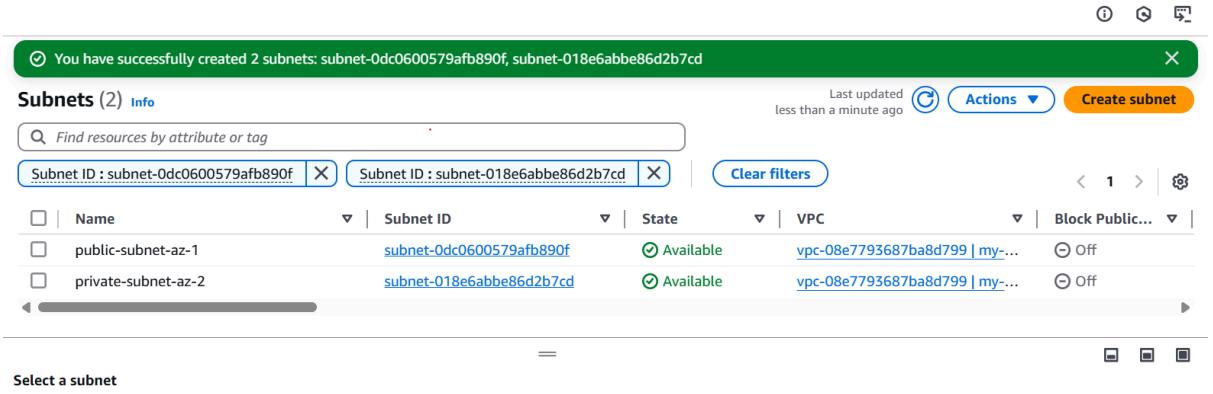
Creating a vpc and adding 2 subnets

1 - public subnet with 256 usable ips

2 - private subnet with 256 usable ips

VPC and Subnet configuration :

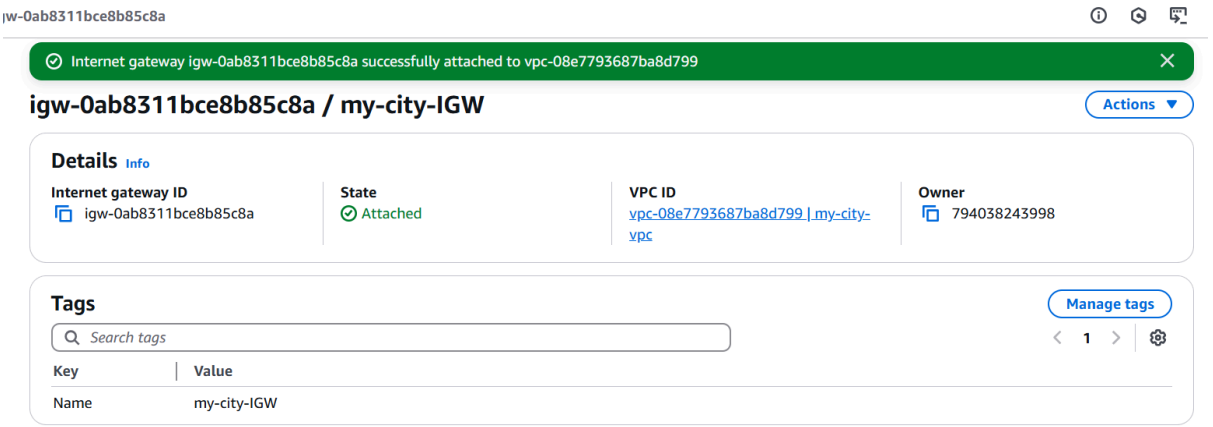
1. Created a vpc with 10.0.0.0/16 ip range.
2. Added 2 subnets with 256 each -> available 251 each
3. Subnet a -> 10.0.0.0/24
4. Subnet b -> 10.0.1.0/24



NOTE: For every subnet we create, by default there will be 1 NACL and 1 Route table attached to the subnet.

Next attached IGW:

Create IGW and attach to our VPC



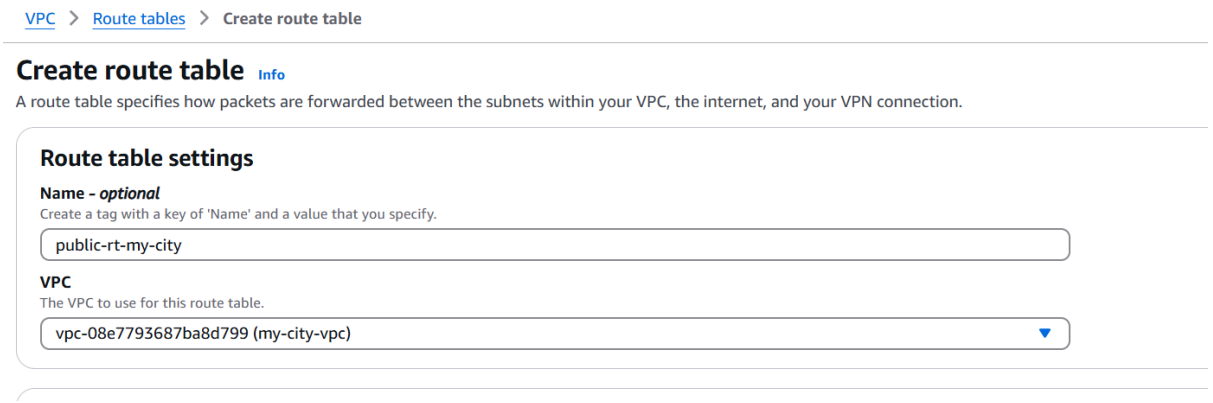
Now the internet is available for the overall vpc.. But we need only the public subnet to get internet access.

So we need to create the RT.

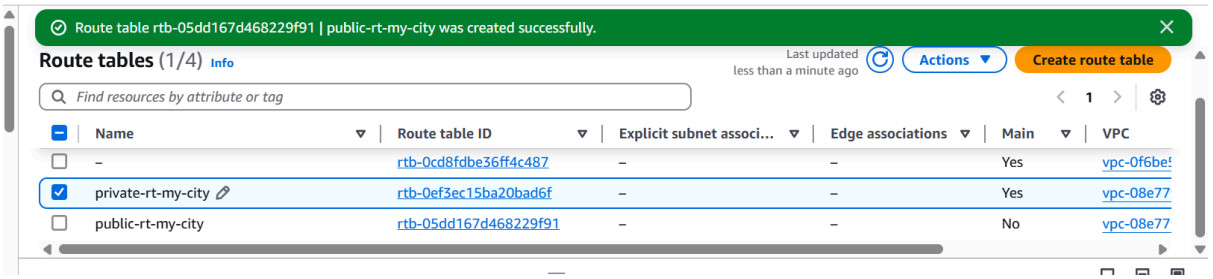
Initially both the subnets will have the same route tables : like rtb-0ef3ec15rf58bad6f

When we change a rule thenthe changes will be reflected to all the route tables

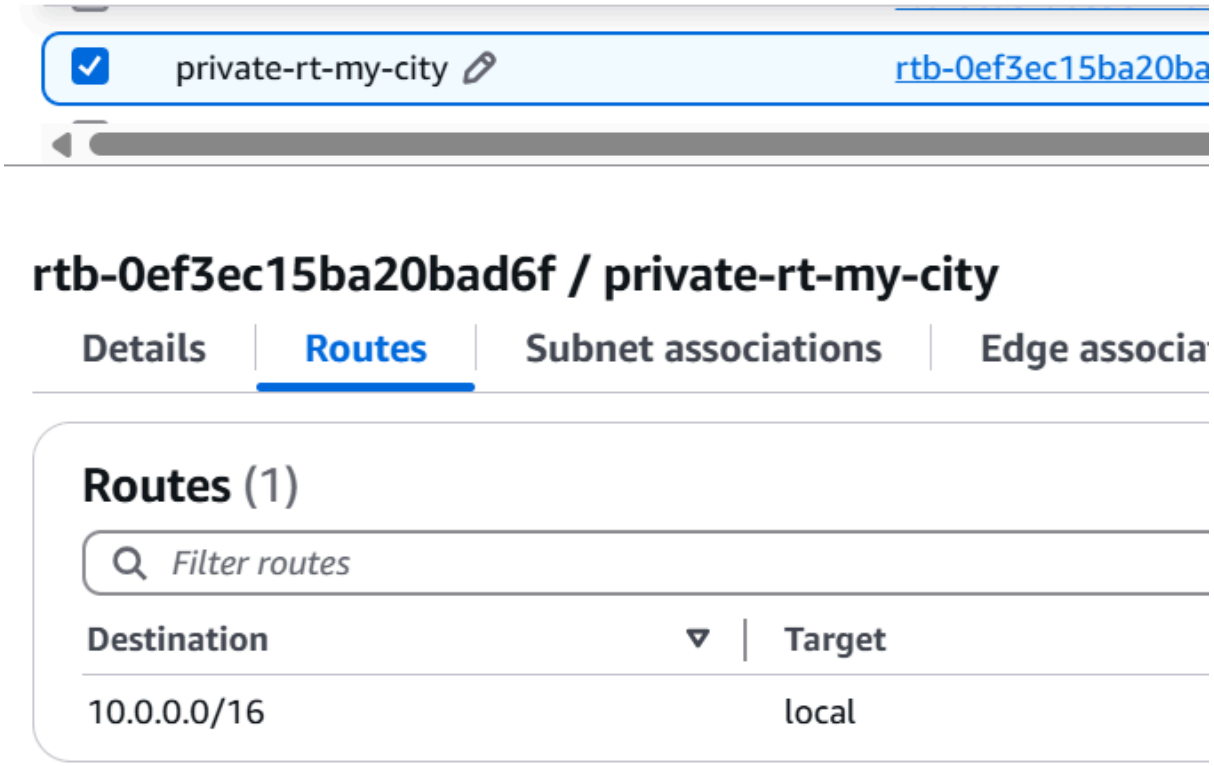
Therefore we create a new RT.. and connect it to the Public subnet



then after this we will rename the vpc which was created with the 2 subnet creations commonly (-bad6f) as Private



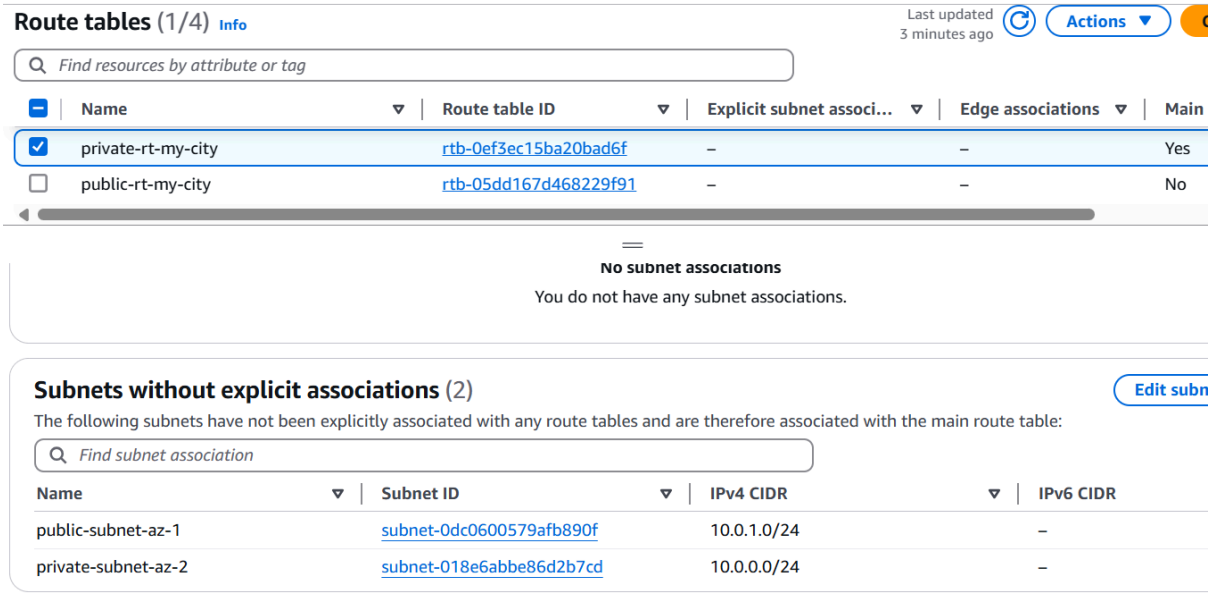
WE can see that our private rt has overall access to the vpc.. Hence the ip range for this is 10.0.0.0/16



We need to detach our public subnet from this rt..

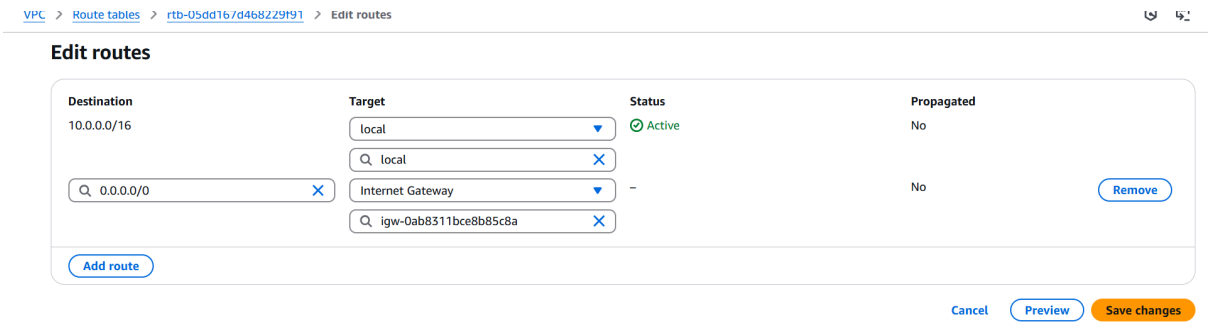
We can see there are 2 subnets that are associated with this private rt..

So lets detach the ***public-subnet-az-1***



Now that we have separated the route tables for public and private sn..

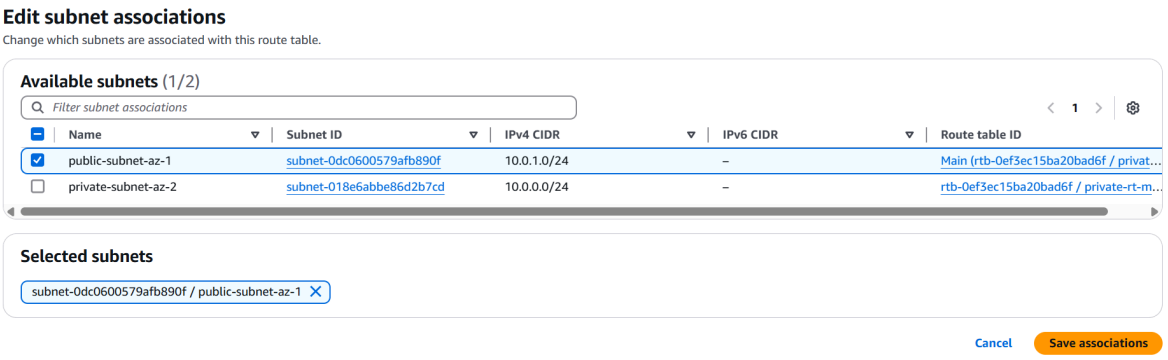
Lets add igw to the public subnet.



0.0.0.0/0 is required for the internet access.. for IPv4

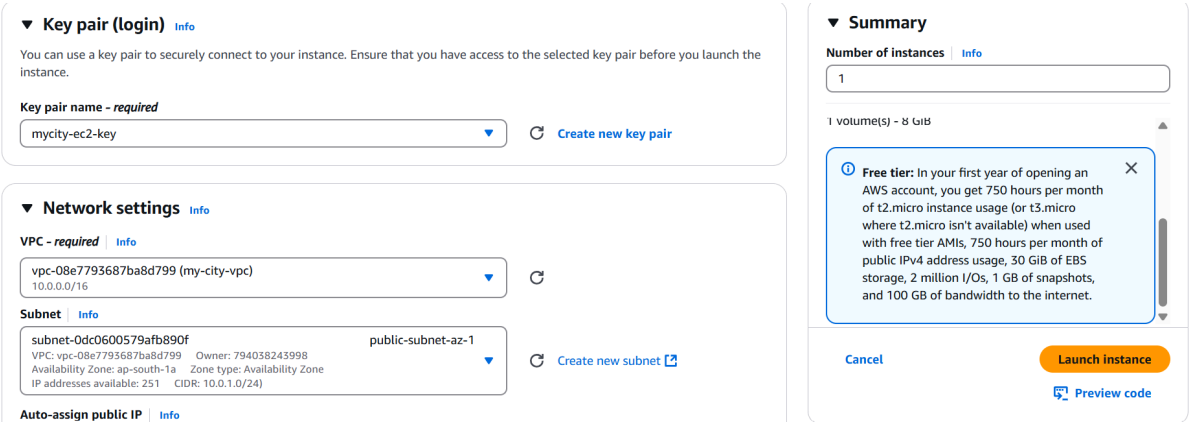
And add the igw connection of our vpc.

Now add the public to the subnet association:



Launching EC2 instances:

1 - public ec2 instance in the public subnet



2 - Similarly launched the private ec2 instance in the private subnet.

(We'll connect to these instances using our usual ssh protocol remotely)

See you on my next one. 👋