AWS Virtual Private Cloud (VPC) – Step-by-Step Hands-On Implementation

🔹 **What is VPC?**

- **VPC** stands for **Virtual Private Cloud**.
- It is a **region-specific** service that allows you to create an **isolated, customizable network** inside AWS.
- A VPC acts like your own private data center in the cloud where you control:
    - IP ranges
    - Subnets
    - Routing tables
    - Security layers

Inside a VPC, you can create **subnets** – these are smaller networks:

✅ **Public subnets** for web servers or internet-facing applications

✅ **Private subnets** for databases or internal apps

🔹 **Subnet Types**

✅ **Public Subnet**

- Connected to an **Internet Gateway**
- Supports **two-way** communication (internet ↔ instances)

✅ **Private Subnet**

- Connected through a **NAT Gateway or NAT Instance**
- Supports **one-way** communication (instances → internet, but not vice versa)

🔹 **Key Networking Components**

✅ **Internet Gateway (IGW)**

- Attached to the VPC for public connectivity
- Allows inbound and outbound internet access for resources in public subnets

✅ **NAT Gateway**

- Allows outbound internet traffic from private subnets
- **Blocks inbound requests** (keeps instances secure)

✅ **NAT Instance**

- Similar to NAT Gateway but uses an EC2 instance
- **Cheaper** but **less scalable**

🔹 **Route Tables**

- Define how network traffic flows
- Two main types:
  ✅ **Public Route Table** – routes via IGW
  ✅ **Private Route Table** – routes via NAT Gateway/Instance

🔹 **CIDR (Classless Inter-Domain Routing)**

CIDR determines:

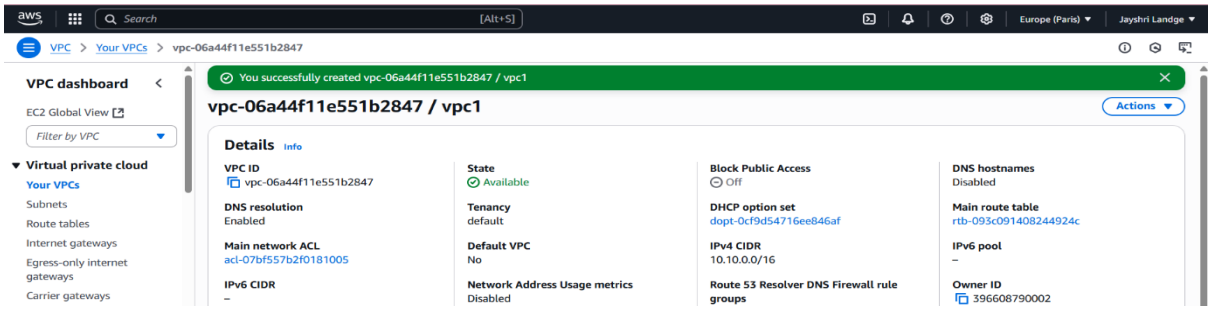
- How many **IP addresses** are available in the VPC
- How many **subnets** can be created

Example Calculation:
CIDR: 10.10.0.0/28
➞ 32 − 28 = 4 → **n = 4**
➞ 2^4 = 16 IPs (14 usable, as AWS reserves 5 IPs)

▶ **VPC Hands-On Setup**

Step 1: Create a VPC

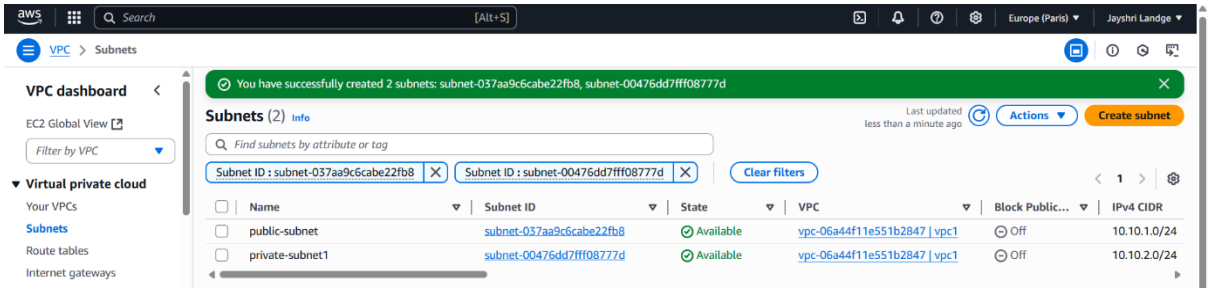
1. Go to **AWS Console → VPC → Your VPCs → Create VPC**
2. **Name:** myvpc
3. **CIDR Block:** 10.10.0.0/16
4. Click **Create**



Screenshot 1: VPC Creation

Step 2: Create Subnets

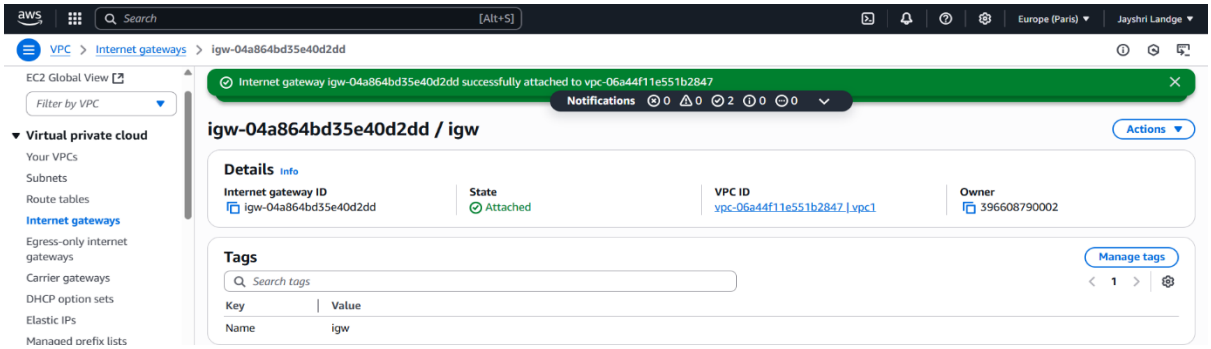
1. Navigate to **VPC → Subnets → Create Subnet**
2. **Public Subnet**
    - Name: public-subnet
    - Availability Zone: Zone A
    - CIDR: 10.10.1.0/24
3. **Private Subnet**
    - Name: private-subnet
    - Availability Zone: Zone B
    - CIDR: 10.10.2.0/24



Screenshot 2: Subnet Creation

Step 3: Attach an Internet Gateway

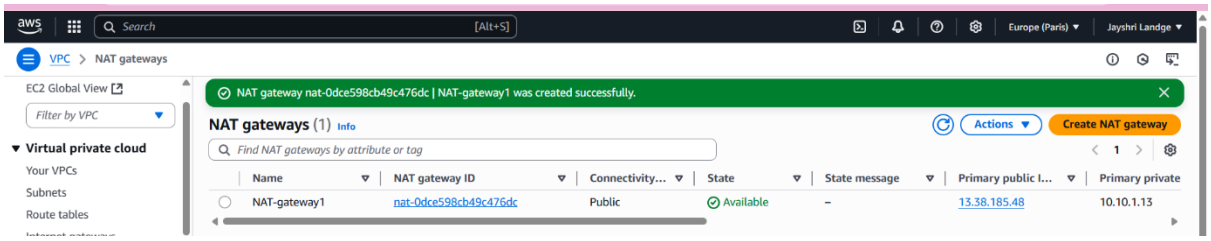
1. **VPC → Internet Gateways → Create Internet Gateway**
    - Name: pub-sub-igw
2. Attach IGW to myvpc



Screenshot 3: IGW Attached

Step 4: Create a NAT Gateway

1. Allocate an **Elastic IP**
2. **VPC → NAT Gateways → Create NAT Gateway**
    - Subnet: public-subnet
    - Elastic IP: (allocated above)



Screenshot 4: NAT Gateway Setup

Step 5: Configure Route Tables

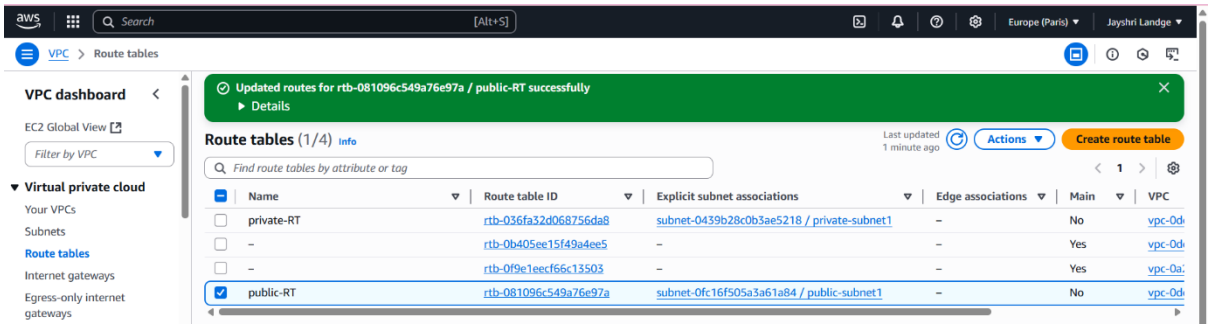
✅ Public Route Table

- Name: Public-RT
- Attach to myvpc
- Add Route:
    - Destination: 0.0.0.0/0
    - Target: Internet Gateway
- Associate **Public Subnet**

✅ Private Route Table

- Name: Private-RT

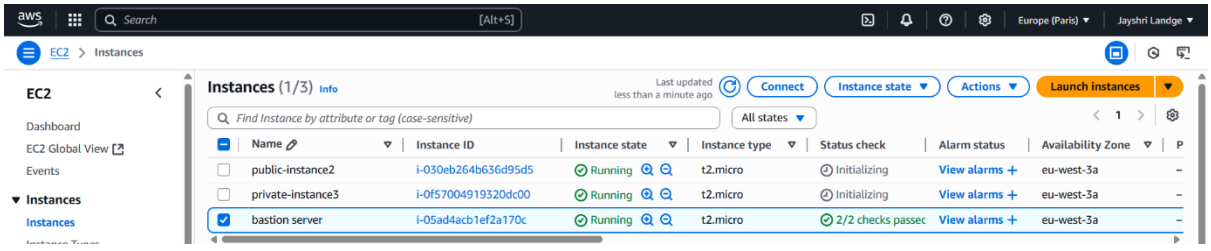
- Attach to myvpc
- Add Route:
    - Destination: 0.0.0.0/0
    - Target: NAT Gateway
- Associate **Private Subnet**



Screenshot 5: Route Table Associations

Step 6: Launch EC2 Instances

1. Go to **EC2 → Launch Instance**
    - Public Subnet → Enable **Public IP**
    - Create Security Group (allow SSH, HTTP, ICMP)
    - Launch instance
2. Repeat for **Private Subnet** (no public IP)



Screenshot 6: Instance Launch

▶ Accessing Instances

- **Public Instance:** Accessible directly via SSH from the internet
- **Private Instance:**
    - Connect via **public instance (bastion)**

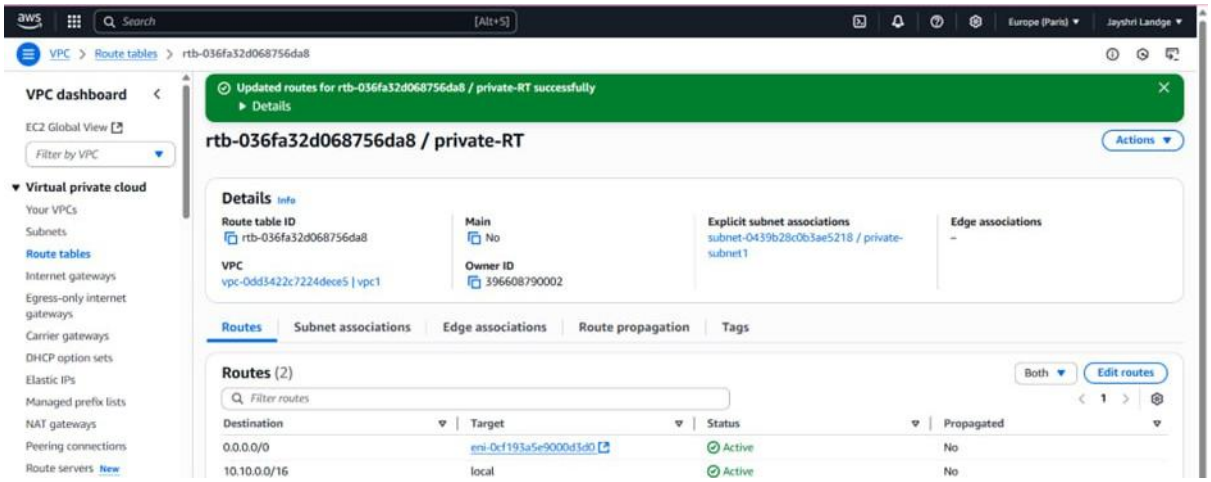
- Example:
- ssh -i keypair.pem ec2-user@private_instance_ip

```
[root@ip-10-10-1-143 ~]# vim keypair33.pem
[root@ip-10-10-1-143 ~]# chmod 400 keypair33.pem
[root@ip-10-10-1-143 ~]# ssh -i keypair33.pem ec2-user@10.10.2.102
The authenticity of host '10.10.2.102 (10.10.2.102)' can't be established.
ED25519 key fingerprint is SHA256:rY/DFD+0F6wBfAIC+LT/we+FKMUss8LjpN7k30TDDoE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.2.102' (ED25519) to the list of known hosts.
Register this system with Red Hat Insights: rhc connect

Example:
# rhc connect --activation-key <key> --organization <org>

The rhc client and Red Hat Insights will enable analytics and additional
management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

You can learn more about how to register your system
using rhc at https://red.ht/registration
[ec2-user@ip-10-10-2-102 ~]$ sudo su -
[root@ip-10-10-2-102 ~]#
```

Screenshot 7: SSH Access

▶ NAT Instance Setup (Alternative to NAT Gateway)

1. Go to **Community AMIs** → Search nat
2. Launch NAT Instance in **Public Subnet**
    - Allow **all traffic** in SG
3. Disable **Source/Destination Check**
    - Actions → Networking → Change Source/Destination Check → Stop
4. Add NAT Instance as the **target** in Private Route Table



Screenshot 8: Add NAT Instance in Private Route Table

▶ **CIDR Planning Example**

CIDR (Classless Inter-Domain Routing) defines how many IP addresses are available within a VPC or subnet.

Example:
10.10.0.0/16

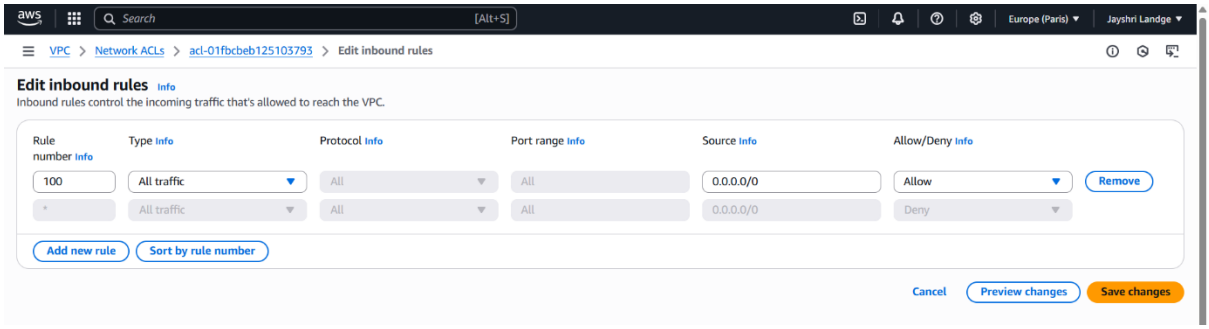
- Total addresses: **65,536**
- If divided into **/24 subnets**, each subnet will have **256 Ips**

CIDR Notation	Addresses	Addresses
/8	2^{24}	16,777,216
/9	2^{23}	8,388,608
/10	2^{22}	4,194,304
/11	2^{21}	2,097,152
/12	2^{20}	1,048,576
/13	2^{19}	524,288
/14	2^{18}	262,144
/15	2^{17}	131,072
/16	2^{16}	65,536
/17	2^{15}	32,768
/18	2^{14}	16,384
/19	2^{13}	8,192
/20	2^{12}	4,096
/21	2^{11}	2,048
/22	2^{10}	1,024
/23	2^{9}	512
/24	2^{8}	256
/25	2^{7}	128
/26	2^{6}	64
/27	2^{5}	32
/28	2^{4}	16
/29	2^{3}	8
/30	2^{2}	4

Screenshot 9: CIDR Range Calculation Table

▶ Understanding NACL

- **NACL (Network Access Control List)** controls **inbound and outbound** traffic at **subnet level**.
- **Differences from Security Group:**
    - **SG:** Instance-level, only allows rules
    - **NACL:** Subnet-level, can allow **or** deny rules
- **Rule Priority:** Lower number → higher priority



Screenshot 10: NACL Setup

Conclusion

In this walkthrough, we:

✓ Built a **custom AWS VPC**

✓ Created **public & private subnets**

✓ Configured **Internet Gateway, NAT Gateway & Route Tables**

✓ Deployed EC2 instances in both subnets

✓ Explored **NAT Gateway vs. NAT Instance**

✓ Learned **CIDR planning** & **NACL concepts**

By completing these steps, you gain a **clear understanding of AWS networking** and how to **design secure, scalable architectures** for real-world deployments.