

## 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  $\leftrightarrow$  instances)

#### Private Subnet

- Connected through a **NAT Gateway or NAT Instance**
- Supports **one-way** communication (instances  $\rightarrow$  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

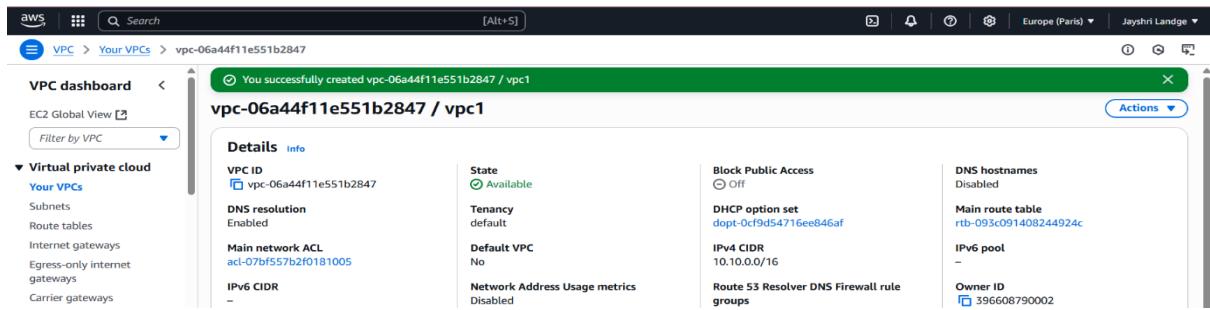
→  $32 - 28 = 4 \rightarrow n = 4$

→  $2^4 = 16$  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

Subnets (2) <small>Info</small>						
<input type="text" value="Find subnets by attribute or tag"/> <input type="button" value="Clear filters"/>						
	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
<input type="checkbox"/>	public-subnet	subnet-037aa9c6cabe22fb8	Available	vpc-06a44f11e551b2847   vpc1	Off	10.10.1.0/24
<input type="checkbox"/>	private-subnet	subnet-00476dd7ff08777d	Available	vpc-06a44f11e551b2847   vpc1	Off	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

The screenshot shows the AWS VPC console under the 'Internet gateways' section. A success message at the top states: "Internet gateway igw-04a864bd35e40d2dd successfully attached to vpc-06a44f11e551b2847". Below it, the internet gateway details are displayed: ID igw-04a864bd35e40d2dd, State Attached, VPC ID vpc-06a44f11e551b2847, and Owner 396608790002. A single tag named 'igw' is listed under Tags.

**Screenshot 3: IGW Attached**

## Step 4: Create a NAT Gateway

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

The screenshot shows the AWS VPC console under the 'NAT gateways' section. A success message at the top says: "NAT gateway nat-0dce598cb49c476dc | NAT-gateway1 was created successfully.". The table below lists the created NAT gateway: Name NAT-gateway1, NAT gateway ID nat-0dce598cb49c476dc, Connectivity Public, State Available, Primary public IP 13.38.185.48, and Primary private IP 10.10.1.13.

**Screenshot 4: NAT Gateway Setup**

## Step 5: Configure Route Tables

### Public Route Table

- Name: Public-RT
- Attach to myvpc
- Add Route:
  - o Destination: 0.0.0.0/0
  - o 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**

The screenshot shows the AWS VPC dashboard with the 'Route tables' section selected. A green success message at the top states: 'Updated routes for rtb-081096c549a76e97a / public-RT successfully'. Below it, a table lists four route tables: private-RT, rtb-0b405ee15f49a4ee5, rtb-0f9e1eecf66c13503, and public-RT. The public-RT row is highlighted with a blue border. The table includes columns for Name, Route table ID, Explicit subnet associations, Edge associations, Main, and VPC.

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
private-RT	rtb-036fa52d068756da8	subnet-0439b28c0b3ae5218 / private-subnet1	-	No	vpc-0d1
-	rtb-0b405ee15f49a4ee5	-	-	Yes	vpc-0d1
-	rtb-0f9e1eecf66c13503	-	-	Yes	vpc-0d1
public-RT	rtb-081096c549a76e97a	subnet-0fc16f505a3a61a84 / public-subnet1	-	No	vpc-0d1

**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)

The screenshot shows the AWS EC2 Instances page. It displays three instances: 'public-instance2', 'private-instance3', and 'bastion server'. All instances are currently running. The 'bastion server' instance is selected, indicated by a blue border around its row. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
public-instance2	i-030eb264b636d95d5	Running	t2.micro	Initializing	<a href="#">View alarms +</a>	eu-west-3a
private-instance3	i-0f57004919320d00	Running	t2.micro	Initializing	<a href="#">View alarms +</a>	eu-west-3a
bastion server	i-05ad4acb1ef2a170c	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	eu-west-3a

**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>

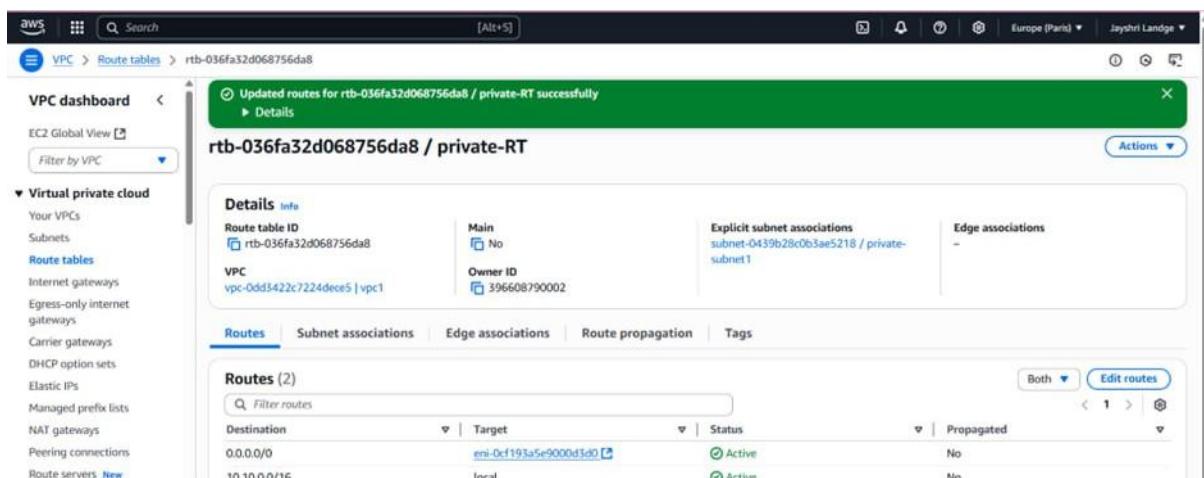
The rhc client and Red Hat Insights will enable analytics and additional
management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

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

The screenshot shows the AWS VPC Network ACLs 'Edit inbound rules' interface. It displays two rules:

Rule number	Type	Protocol	Port range	Source	Action
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Buttons at the bottom include 'Add new rule', 'Sort by rule number', 'Preview changes', and 'Save changes'.

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.