JIDNYASA CORPORATE
TRAINING CENTRE

AWS/DevOps Notes

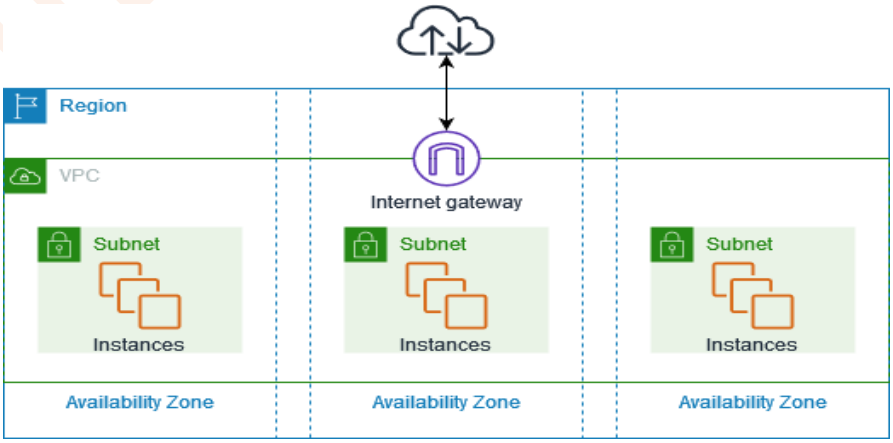
# AWS VPC
# Virtual Private Cloud

# Amazon Virtual Private Cloud (VPC):

- Logically isolated network section over AWS cloud where you can launch AWS resources in a virtual network that you define.
- You have complete control over your virtual networking environment, including **selection of your own IP address ranges**, **creation of subnets**, and **configuration of route tables and network gateways**.
- You can easily customize the network configuration for your Amazon VPC. For example, you can create a public subnet for your web servers that have access to the Internet, and place your backend systems such as databases or application servers in a private subnet with no Internet access. You can leverage multiple layers of security, including **Security Groups (SG)** and **Network Access Control Lists (NACL)**, to help control access to Amazon EC2 instances in each subnet.
- There are no additional charges for creating and using the VPC itself. . There are, however, charges for some VPC components, such as NAT gateways, IP Address Manager, traffic mirroring, Reachability Analyzer, and Network Access Analyzer.
- *IMP Points related to VPC

✓ VPC is a virtual network of data centre inside AWS for one client.
✓ Every VPC is logically isolated from other virtual network in the AWS cloud.
✓ Maximum 5 VPC can be created in one region and 200 subnets in 1 VPC.
✓ You can increase the limit by contacting aws support or using service quota service.
✓ We can allocate maximum 5 Elastic IP
✓ When we create VPC, DHCP, NACL, one Private route table & one security group gets create automatically.
✓ A VPC is confined to an AWS region and does not extend between regions.
✓ Once the VPC is created you cannot change its CIDR block range.
✓ If you need a different CIDR size create a new VPC.
✓ The different subnets within a VPC cannot overlap their IP range.
✓ You can however expand your VPC CIDR by adding new/extra IP address ranges (except GovCloud and AWS China)

The following diagram shows an example VPC. The VPC has one subnet in each of the Availability Zones in the Region, EC2 instances in each subnet, and an internet gateway to allow communication between the resources in your VPC and the internet.



# Components of VPC:

Amazon VPC comprises a variety of objects that will be familiar to customers with existing networks:

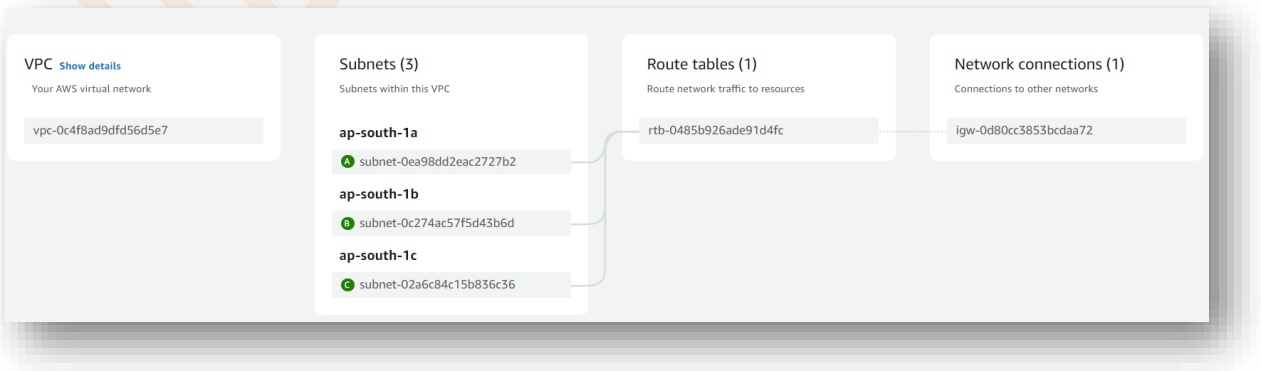
- **A Virtual Private Cloud:** A logically isolated virtual network in the AWS cloud. You define a VPC's IP address space from ranges you select.
- **Subnets:** A segment of a VPC's IP address range where you can place groups of isolated resources.
- **Internet Gateway:** The Amazon VPC side of a connection to the public Internet.
- **NAT Gateway:** A highly available, managed Network Address Translation (NAT) service for your resources in a private subnet to access the Internet.
- **Virtual private gateway:** The Amazon VPC side of a VPN connection.
- **Peering Connection:** A peering connection enables you to route traffic via private IP addresses between two peered VPCs.
- **VPC Endpoints:** Enables private connectivity to services hosted in AWS, from within your VPC without using an Internet Gateway, VPN, Network Address Translation (NAT) devices, or firewall proxies.
- **Egress-only Internet Gateway:** A stateful gateway to provide egress only (inbound traffic only) access for IPv6 traffic from the VPC to the Internet.

Let's understand VPC & its components in detail:

# VPC- Virtual Private Cloud

Amazon VPC enables you to create a virtual network in the AWS cloud - no VPNs, hardware, or physical datacentres required.
You can define your own network IP range, and control how your network and the Amazon EC2 resources inside your network are exposed to the Internet (Public/Private).

The following is a visual representation of a simple VPC and its resources from the Preview pane shown when you create a VPC using the AWS Management Console. For an existing VPC, you can access this visualization on the Resource map tab. This example shows the resources that are initially selected on the Create VPC page when you choose to create the VPC plus other networking resources. This VPC is configured with an IPv4 CIDR, 3 subnets in three Availability Zones, one route table, and one internet gateway.




**Types of VPC's-**

There are two types of VPC's-

1. **Default VPC-** Comes with AWS account, in each region we have one default VPC.
2. **Custom VPC-** Created by us/customers
3. **Private VPC & Public VPC**

# VPC Quotas:

Below is the VPC quotas or you can say limit per AWS account:
If you request a quota increase that applies per resource, AWS increase the quota for all resources in the Region.

**VPC & Subnets**

| Name | Default | Adjustable | Comments |
|---|---|---|---|
| **VPCs per Region** | 5 | Yes | Increasing this quota increases the quota on internet gateways per region by the same amount.<br>You can increase this limit so that you can have hundreds of VPCs per Region. |
| **Subnets per VPC** | 200 | Yes | |

**Public IP Addresses:**

| Name | Default | Adjustable | Comments |
|---|---|---|---|
| **Elastic IP addresses per region** | 5 | Yes | This quota applies to individual AWS account VPCs and shared VPCs. |

# Subnets in AWS VPC:

- Subnet is a subnetwork or you can say part of Network (VPC).
- It's a range of IP addresses within the VPC
- We launch our ec2 instances or keep our resources in subnets
- A subnet is tied to a single Availability Zone within an AWS region.
- This provides redundancy and fault tolerance across different Availability Zones.

**Types of Subnets:**

- **Public Subnet:**
  - Subnet which is accessible over the internet
  - A subnet that has a route to an Internet Gateway
  - allowing instances within it to have direct access to the internet.
- **Private Subnet:**
  - Private subnets do not have internet gateway connectivity so cant be accessed over internet
  - Private subnets do not have a route to the Internet Gateway,
  - Instances within it cannot directly access the internet.
  - They can, however, use a NAT Gateway or NAT instance to send traffic out to the internet.

**CIDR Block:**

Each subnet is associated with a specific CIDR (Classless Inter-Domain Routing) block, which defines its IP address range. The CIDR block of a subnet must fall within the range of the VPC's CIDR block.

**Example:**

If you have a VPC with a **CIDR block of 10.0.0.0/16,** you might create two subnets:

- Public Subnet- 10.0.1.0/24 in Availability Zone us-east-1a
- Private Subnet- 10.0.2.0/24 in Availability Zone us-east-1b

# Routing in AWS VPC:

- **Route Tables**:
  - We create Route tables that defines the rules for directing traffic.
  - The main route table is associated with every subnet by default unless you specify otherwise.
- **Peering Connections**:
  - Enables communication between two VPCs using private IP addresses as if they were in the same network.

# Security in VPC:

- Security Groups:
  - Act as a virtual firewall for your instance to control inbound and outbound traffic.
- NACLs:
  - Operate at the subnet level, with rules that are applied to all instances within the subnet. They are stateless, meaning rules must be explicitly defined for both inbound and outbound traffic.

| Security Group | Network Access Control List |
|---|---|
| Acts as a firewall for associated Amazon EC2 instances. | Acts as a firewall for associated subnets. |
| You can secure your VPC instances using only security groups. | Network ACLs are an additional layer of defence. |
| Supports allow rules only. | Supports allow rules and deny rules. |
| Stateful , no need to specify outbound rule for inbound rule | Stateless, outbound rule must be specified of any inbound rule |
| Evaluates all rules before deciding whether to allow traffic. | Evaluates rules in number order when deciding whether to allow traffic, starting with the lowest numbered rule. |
| Applies only to the instance that is associated to it. | Applies to all instances in the subnet it is associated with. |
| A newly created security group denies all inbound traffic by default. | A default NACL allows inbound and outbound traffic by default. |
| A newly created security group has an outbound rule that allows all outbound traffic by default | A custom NACLs deny inbound and outbound traffic by default unless you explicitly add rules to allow traffic. |
| Instances associated with a security group can't talk to each other unless you add rules allowing it. | Each subnet in your VPC must be associated with a network ACL. If none is associated, the default NACL is selected. |
| Security groups are associated with network interfaces. | You can associate a network ACL with multiple subnets; however, a subnet can be associated with only one network ACL at a time. |

For example:

**NACL PORTS SETTINGS FOR THE WEB SERVER:**

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 100 | SSH (22) | TCP (6) | 22 | 0.0.0.0/0 | Allow |
| 200 | HTTP (80) | TCP (6) | 80 | 0.0.0.0/0 | Allow |
| 300 | HTTPS (443) | TCP (6) | 443 | 0.0.0.0/0 | Allow |
| 400 | Custom TCP | TCP (6) | 1024 - 65535 | 0.0.0.0/0 | Allow |

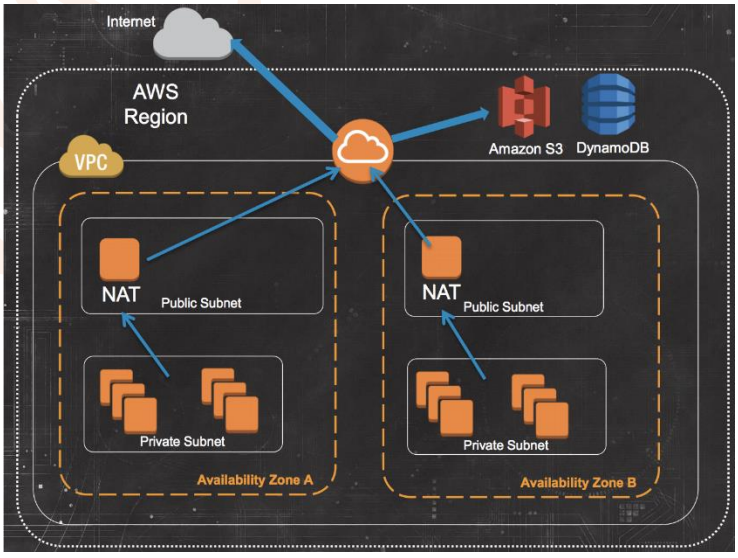
# Internet gateways Vs NAT Gateway:

## Internet Gateway (IGW):

- Internet Gateway allows resources in your VPC to connect to the internet.
- It facilitates both inbound and outbound traffic between your VPC and the internet.
- Resources in a public subnet use an Internet Gateway to communicate with the internet.
- The IGW allows instances with public IP addresses to send and receive traffic from the internet.
- To use an IGW, the route table associated with your subnet must include a route that directs internet-bound traffic (0.0.0.0/0) to the IGW.
- Resources in the subnet with a route to the IGW are directly accessible from the internet, assuming the security group and network ACL rules allow it.
- The IGW is stateful, meaning it tracks the state of connections, allowing responses to be sent back through the same connection.

## NAT Gateway:

- NAT Gateway enables instances in a private subnet to initiate outbound traffic to the internet or other AWS services, but it prevents the internet from initiating connections to those instances.
- NAT Gateways are used by instances in private subnets to access the internet (e.g., for software updates or API calls) without exposing the instances directly to the internet.




- The route table of the private subnet must include a route that directs internet-bound traffic (0.0.0.0/0) to the NAT Gateway.
- Resources in a private subnet using a NAT Gateway cannot be accessed directly from the internet. The NAT Gateway only allows outbound traffic.
- NAT Gateways are also stateful, meaning they allow responses to traffic that was initiated by the instance in the private subnet.
- The NAT Gateway performs Network Address Translation (NAT) to allow instances in private subnets to use a public IP address for internet communication, without exposing their private IP addresses to the internet.

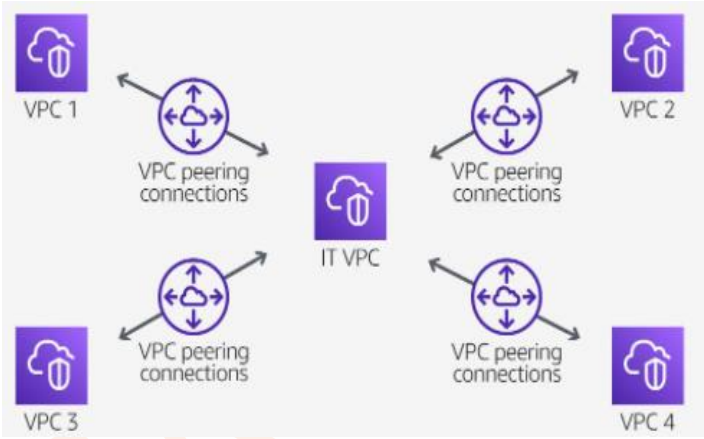
# VPC Peering:

- A VPC peering connection is a network connection between two VPC's that enables you to route traffic between them using private IPV4 addresses or IPV6 addresses.
- Traffic between two VPC's will flow through AWS's private network
- Instances in either VPC can communicate with each other as if they are within the same network.
- You can create a VPC peering connection between your own VPC or with a VPC in another AWS account. The VPC can be in different region as well.

## Limitations of VPC peering:

**1. No Transitive Peering**: If VPC A is peered with VPC B, and VPC B is peered with VPC C, VPC A cannot communicate directly with VPC C. Each pair of VPCs needs a separate peering connection.

**2. No Overlapping CIDR Blocks**: VPCs that have overlapping IP address ranges cannot be peered. This requires careful planning of CIDR blocks to ensure no conflicts.




# VPC Endpoint:

A VPC endpoint enables you to privately connect your VPC to supported AWS services, instances in your VPC do not require public IP address to communicate with resources in the service. Endpoints are virtual devices.

**There are two types of VPC Endpoints:**

1. **Interface Endpoints**: Use an Elastic Network Interface (ENI) with a private IP address to connect your VPC to an AWS service. These are used for services like Amazon S3, DynamoDB, and many others.
2. **Gateway Endpoints**: Specifically, for S3 and DynamoDB, these endpoints create a gateway that you can route traffic to through your VPC route tables.

# -: PRACTICAL STEPS :-

## VPC with Public Subnet:

### **STEP 1:** Create a VPC

- → Go to the **VPC Dashboard** in the AWS Management Console.
- → Click on **Create VPC**
- → Provide a name for your VPC.
- → Specify the IPv4 CIDR block (e.g., 10.0.0.0/16).
- → Leave the rest of the settings as default.
- → Click **Create VPC**.

### **STEP 2:** Create an Internet Gateway

- → In the VPC Dashboard, go to **Internet Gateways**.
- → Click on **Create Internet Gateway**.
- → Name your internet gateway and click **Create**.
- → Once created, select the internet gateway, then click **Actions** > **Attach to VPC**.
- → Choose the VPC you just created and click **Attach Internet Gateway**.

### **STEP 3:** Create a Public Subnet

- → In the VPC Dashboard, go to **Subnets**.
- → Click on **Create Subnet**.
- → Select your VPC, and give name to subnet.
- → Specify the Availability Zone (optional) and the CIDR block (e.g., 10.0.1.0/24).
- → Click **Create Subnet**.

### **STEP 5:** Create a Route Table

- → In the VPC Dashboard, go to **Route Tables**.
- → Click on **Create Route Table**.
- → Name your route table and associate it with your VPC.
- → Click **Create**.
- → Select the newly created route table, and under the **Routes** tab, click **Edit routes**.
- → Add a new route:
  - **Destination**: 0.0.0.0/0 (for all traffic)
  - **Target**: Choose the **Internet Gateway** you created.
- → Click **Save routes**.
- → Under the **Subnet Associations** tab, click **Edit subnet associations** and select your public subnet.

### **TEST:** Launch an EC2 Instance

- → Go to the **EC2 Dashboard** in the AWS Management Console.
- → Click on **Launch Instance**.
- → Choose an AMI (e.g., Amazon Linux 2) and select the instance type (e.g., t2.micro).
- → Choose or create a key pair for SSH access and launch the instance.
- → In the **Network Settings:**
  - Select your newly created VPC and public subnet as a network for your EC2 instance.
  - Ensure that **Auto-assign Public IP** is enabled.

- → In the **Configure Security Group** step, create a new security group or choose an existing one.
    - Allow SSH (port 22) for remote access, and other ports as needed.
- → Review and launch the instance.
- → Once the instance is running, you can access it via SSH using the public IP address assigned to the instance.

---

# VPC with Public and Private Subnet:

### **STEP 1:** Create a VPC

- → Go to the **VPC Dashboard** in the AWS Management Console.
- → Click on **Create VPC**
- → Provide a name for your VPC.
- → Specify the IPv4 CIDR block (e.g., 10.0.0.0/16).
- → Leave the rest of the settings as default.
- → Click **Create VPC**.

### **STEP 2:** Create an Internet Gateway

- → In the VPC Dashboard, go to **Internet Gateways**.
- → Click on **Create Internet Gateway**.
- → Name your internet gateway and click **Create**.
- → Once created, select the internet gateway, then click **Actions** > **Attach to VPC**.
- → Choose the VPC you just created and click **Attach Internet Gateway**.

### **STEP 3:** Create Public and Private Subnets

- → Create **a Public Subnet**:
    - Go to **Subnets** in the VPC Dashboard.
    - Click on **Create subnet**.
    - Choose the VPC you created.
    - Enter a **Subnet name** (e.g., Public-Subnet).
    - Choose an **Availability Zone**.
    - Enter a **CIDR block** for the public subnet (e.g., 10.0.1.0/24).
    - Click **Create subnet**.
- → Create **a Private Subnet**:
    - Click on **Create subnet** again.
    - Choose the VPC you created.
    - Enter a **Subnet name** (e.g., Private-Subnet).
    - Choose an **Availability Zone**.
    - Enter a **CIDR block** for the private subnet (e.g., 10.0.2.0/24).
    - Click **Create subnet**.

### **STEP 4:** Configure Route Tables

- → **Create a** Route **Table for Public Subnet**:
    - Go to **Route Tables** in the VPC Dashboard.
    - Click on **Create route table**.
    - Enter a **Name tag** (e.g., Public-Route-Table).
    - Choose your VPC.
    - Click **Create route table**.

→ **Edit** Routes **for Public Route Table**:
- Select the newly created route table.
- Click on the **Routes** tab, then **Edit routes**.
- Click **Add route**.
- For **Destination**, enter 0.0.0.0/0.
- For **Target**, select the Internet Gateway you created.
- Click **Save routes**.

→ **Associate Public Subnet with Public Route Table**:
- Go to the **Subnet Associations** tab of your public route table.
- Click on **Edit subnet associations**.
- Select your public subnet.
- Click **Save associations**.

2. **Create a Route Table for Private Subnet**:
- Optionally, you can use the default route table for the private subnet or create a new one if needed.
- Follow similar steps as above, but **do not add routes to the Internet Gateway** for the private subnet.

## **TEST:** Launch an EC2 Instance

**Launch Public Server:**

→ Go to the **EC2 Dashboard** in the AWS Management Console.
→ Click on **Launch Instance**.
→ Choose an AMI (e.g., Amazon Linux 2) and select the instance type (e.g., t2.micro).
→ Choose or create a key pair for SSH access and launch the instance.
→ In the **Network Settings:**
- Select your newly created VPC and public subnet as a network for your EC2 instance.
- Ensure that **Auto-assign Public IP** is enabled.

→ In the **Configure Security Group** step, create a new security group or choose an existing one.
- Allow SSH (port 22) for remote access, and other ports as needed.

→ Review and launch the instance.
→ Once the instance is running, you can access it via SSH using the public IP address assigned to the instance.

**Launch Private Server:**

→ Go to the **EC2 Dashboard** in the AWS Management Console.
→ Click on **Launch Instance**.
→ Choose an AMI (e.g., Amazon Linux 2) and select the instance type (e.g., t2.micro).
→ Choose or create a key pair for SSH access and launch the instance.
→ In the **Network Settings:**
- Select your newly created VPC and private subnet as a network for your EC2 instance.
- Ensure that **Auto-assign Public IP** is enabled.

→ In the **Configure Security Group** step, create a new security group or choose an existing one.
- Allow SSH (port 22) for remote access, and other ports as needed.

→ Review and launch the instance.

→ Once the instance is running, check if you can SSH using the public IP address assigned to the instance. (It should not work!)

---

## NAT Gateway: Provide internet to private subnet:

### **STEP 1:** Create a VPC

→ Go to the **VPC Dashboard** in the AWS Management Console.
→ Click on **Create VPC**
→ Provide a name for your VPC.
→ Specify the IPv4 CIDR block (e.g., 10.0.0.0/16).
→ Leave the rest of the settings as default.
→ Click **Create VPC**.

### **STEP 2:** Create an Internet Gateway

→ In the VPC Dashboard, go to **Internet Gateways**.
→ Click on **Create Internet Gateway**.
→ Name your internet gateway and click **Create**.
→ Once created, select the internet gateway, then click **Actions** > **Attach to VPC**.
→ Choose the VPC you just created and click **Attach Internet Gateway**.

### **STEP 3:** Create Public and Private Subnets

→ Create **a Public Subnet**:
  - Go to **Subnets** in the VPC Dashboard.
  - Click on **Create subnet**.
  - Choose the VPC you created.
  - Enter a **Subnet name** (e.g., Public-Subnet).
  - Choose an **Availability Zone**.
  - Enter a **CIDR block** for the public subnet (e.g., 10.0.1.0/24).
  - Click **Create subnet**.

→ Create **a Private Subnet**:
  - Click on **Create subnet** again.
  - Choose the VPC you created.
  - Enter a **Subnet name** (e.g., Private-Subnet).
  - Choose an **Availability Zone**.
  - Enter a **CIDR block** for the private subnet (e.g., 10.0.2.0/24).
  - Click **Create subnet**.

### **STEP 4:** Create a NAT Gateway for Private Subnet (If you need outbound internet access for resources in the private subnet, you can create a NAT Gateway)

→ **Create an Elastic IP Address**: (Or you can create one during the time of NAT Gateway creation)
  - Go to **Elastic IPs** in the VPC Dashboard.
  - Click **Allocate new address**.
  - Click **Allocate**.

→ **Create a NAT Gateway**:
  - Go to **NAT Gateways** in the VPC Dashboard.
  - Click **Create NAT gateway**.
  - Choose your public subnet.

- Select the Elastic IP you allocated.
- Click **Create NAT gateway**.

## **STEP 4:** Configure Route Tables

1. Create a **Route Table for Public Subnet:**
   - Go to **Route Tables** in the VPC Dashboard.
   - Click on **Create route table**.
   - Enter a **Name tag** (e.g., Public-Route-Table).
   - Choose your VPC.
   - Click **Create route table**.

→ **Edit** Routes **for Public Route Table**:

- Select the newly created route table.
- Click on the **Routes** tab, then **Edit routes**.
- Click **Add route**.
- For **Destination**, enter 0.0.0.0/0.
- For **Target**, select the Internet Gateway you created.
- Click **Save routes**.

→ **Associate Public Subnet with Public Route Table**:

- Go to the **Subnet Associations** tab of your public route table.
- Click on **Edit subnet associations**.
- Select your public subnet.
- Click **Save associations**.

2. Create a **Route Table for Private Subnet**:
   - Optionally, you can use the default route table for the private subnet or create a new one if needed.
   - Follow similar steps as above, but **do not add routes to the Internet Gateway** for the private subnet.

→ **Update Private Route Table**:

- Go to **Route Tables**.
- Select your private route table.
- Click on the **Routes** tab, then **Edit routes**.
- Click **Add route**.
- For **Destination**, enter 0.0.0.0/0.
- For **Target**, select the NAT Gateway you created.
- Click **Save routes**.

## **TEST:** Launch an EC2 Instance

**Launch Public Server:**

→ Go to the **EC2 Dashboard** in the AWS Management Console.
→ Click on **Launch Instance**.
→ Choose an AMI (e.g., Amazon Linux 2) and select the instance type (e.g., t2.micro).
→ Choose or create a key pair for SSH access and launch the instance.
→ In the **Network Settings:**

- Select your newly created VPC and public subnet as a network for your EC2 instance.
- Ensure that **Auto-assign Public IP** is enabled.

→ In the **Configure Security Group** step, create a new security group or choose an existing one.

- Allow SSH (port 22) for remote access, and other ports as needed.

- → Review and launch the instance.
- → Once the instance is running, you can access it via SSH using the public IP address assigned to the instance.

**Launch Private Server:**

- → Go to the **EC2 Dashboard** in the AWS Management Console.
- → Click on **Launch Instance**.
- → Choose an AMI (e.g., Amazon Linux 2) and select the instance type (e.g., t2.micro).
- → Choose or create a key pair for SSH access and launch the instance.
- → In the **Network Settings:**
  - Select your newly created VPC and private subnet as a network for your EC2 instance.
  - Ensure that **Auto-assign Public IP** is enabled.
- → In the **Configure Security Group** step, create a new security group or choose an existing one.
  - Allow SSH (port 22) for remote access, and other ports as needed.
- → Review and launch the instance.
- → Once the instance is running, check if you can SSH using the public IP address assigned to the instance. (It should not work!)

---

# Connect to Private Server- Bastion host/Jump Server

To connect to a private server in your VPC using a Public Server, follow these steps:

## **STEP 1:** Set Up the Public Server

- → **Launch the public ec2 Instance**:
  - Ensure that your public EC2 instance (Public Server) is running in the **public subnet** of your VPC.
  - The Public Server should have a public IP address.
- → **Configure Security Groups for the Public Server**:
  - Go to the **EC2 Dashboard**.
  - Select the **Instances** section and find your Public Server.
  - Under **Security groups**, ensure the security group allows inbound SSH traffic (port 22) from your IP address or the IP range you need.

## **STEP 2:** Configure the Private Server

- → **Launch the Private Server Instance**:
  - Ensure that your private server is running in the **private subnet** of your VPC.
  - The private server should not have a public IP address.
- → **Configure Security Groups for the Private Server**:
  - Go to the **EC2 Dashboard**.
  - Find your private server instance and check its security group.
  - Ensure the security group allows inbound SSH traffic (port 22) from the security group of the Public Server or its private IP address.

## **STEP 3:** Connect to the Private Server via the Public Server

1. **Connect to the P**
   - Use SSH to connect to your Public Server from your local machine:

- Create key file using vi editor with the key name used for private server (.pem) file, you can copy the content of the .pem key file from local system to file created on public server.
- Now run below command to connect to your private server:

```
ssh -i /path/to/your-key.pem ec2-user@<private -ip>
```

Note- Replace /path/to/your-key.pem with the path to your key file and <private-ip> with private servers private IP address

By following these steps, you'll be able to securely connect to a private server in your VPC.

TEST:

Run ping command to see if you are able to get internet access on private server.

Summary:

- **Public EC2 (Public Server)**: Provides access to the private subnet from the public internet.
- **Private EC2**: Resides in the private subnet and is accessible only through the Public Server.

---

# VPC Peering:

Prerequisites

1. **AWS Accounts**: Ensure you have access to both AWS accounts.
2. **VPCs**: Both VPCs should be created in their respective accounts and should have non-overlapping CIDR blocks.

### **STEP 1:** In the AWS Account with VPC A (Requester Account):

→ **Sign in to the AWS Management Console**.

→ **Navigate to the VPC Dashboard**.

→ **Select the Requesting VPC**:

- Go to **Peering Connections** in the VPC Dashboard on the left-hand side menu.
- Click on **Create Peering Connection**.

→ **Configure the Peering Connection**:

- **Name tag**: Enter a descriptive name for the peering connection.
- **VPC (Requester)**: Select the VPC from this account that you want to peer.
- **Account**: Choose **Another account** and enter the AWS account ID of the other account (the accepter account).
- **VPC (Accepter)**: Enter the VPC ID of the VPC in the other account (or leave it if you want to request the owner to accept).

→ **Click Create Peering Connection**.

### **STEP2:** Accept the VPC Peering Connection Request

**In the AWS Account with VPC B (Accepter Account):**

→ **Sign in to the AWS Management Console**.

→ **Navigate to the VPC Dashboard**.
→ **Select the Accepter VPC**:
- Go to **Peering Connections** in the VPC Dashboard.
- You should see the peering request from the requester account listed here.

→ **Accept the Peering Connection Request**:
- Select the peering request.
- Click **Actions** and then **Accept Request**.
- Confirm by clicking **Yes, Accept**.

## **STEP 3:** Update Route Tables

**In Both Accounts (Requester and Accepter):**

→ **Update Route Tables in VPC A**:
- Navigate to **Route Tables** in the VPC Dashboard.
- Select the route table(s) associated with the subnets that need access to the peered VPC.
- Click on the **Route** tab, then **Edit routes**.
- Click **Add route**.
- For **Destination**, enter the CIDR block of VPC B.
- For **Target**, select the VPC Peering Connection ID.
- Click **Save routes**.

→ **Update Route Tables in VPC B**:
- Navigate to **Route Tables** in the VPC Dashboard.
- Select the route table(s) associated with the subnets that need access to the peered VPC.
- Click on the **Routes** tab, then **Edit routes**.
- Click **Add route**.
- For **Destination**, enter the CIDR block of VPC A.
- For **Target**, select the VPC Peering Connection ID.
- Click **Save routes**.

## **STEP 4:** Update Security Groups (***Optional,** no need to set if you are using all access)

**In Both Accounts:**

→ **Allow Traffic in Security Groups**:
- Go to **Security Groups** in the VPC Dashboard.
- Edit the inbound and outbound rules to allow traffic from the CIDR block of the peered VPC.
- For example, if you want to allow all traffic from the peered VPC, you might add a rule like:
  - **Type**: All Traffic
  - **Protocol**: All
  - **Port Range**: All
  - **Source/Destination**: CIDR block of the peered VPC

## **TEST:** Verify the Connection

→ **Test Connectivity**:
- Launch EC2 instances in both VPCs.

- Test connectivity between instances (e.g., using ping or SSH) to ensure that the peering connection is working correctly.

Summary

1. **Requester Account**: Create the peering request.
2. **Accepter Account**: Accept the peering request.
3. **Both Accounts**: Update route tables to route traffic between VPCs.
4. **Both Accounts**: Modify security groups to allow traffic between VPCs.
5. **Verify**: Test the connectivity between instances in the peered VPCs.

---

# NACL- Network Access Control List:

### **STEP 1:** Create a Network ACL

→ **Navigate to the VPC Dashboard**:
- Go to **Services** and select **VPC**.

→ **Create a New Network ACL**:
- In the VPC Dashboard, go to **Network ACLs** option on the left hand side menu.
- Click **Create network ACL**.
- Enter a **Name tag** (e.g., MyNACL).
- Choose the VPC you want to associate the NACL with.
- Click **Create network ACL**.

### **STEP 2:** Configure Inbound and Outbound Rules

→ **Edit Inbound Rules**:
- Select the newly created NACL.
- Click on the **Inbound Rules** tab, then **Edit inbound rules**.
- Click **Add rule**.
- Configure the rule as follows:
  - **Rule number**: A number to order the rules (e.g. 100).
  - **Type**: Select the type of traffic (SSH, HTTP, HTTPS, Custom TCP).
  - **Protocol**: The protocol to allow (TCP).
  - **Port range**: Specify the port range (e.g., 22, 80, 443 & 1024 -65635).
  - **Source**: Define the source IP range (e.g., 0.0.0.0/0).
  - **Allow/Deny**: Choose whether to allow or deny traffic (Allow)
- Click **Save changes**.

→ **Edit Outbound Rules**: (Must open traffic on same port create in inbound rules)
- Click on the **Outbound Rules** tab, then **edit outbound rules**.
- Click **Add rule**.
- Configure the rule as follows:
  - **Rule number**: A number to order the rules (e.g., 100).
  - **Type**: Select the type of traffic (SSH, HTTP, HTTPS, Custom TCP).
  - **Protocol**: The protocol to allow (TCP).
  - **Port range**: Specify the port range (22, 80, 443 & 1024-65635).
  - **Destination**: Define the destination IP range (0.0.0.0/0).
  - **Allow/Deny**: Choose whether to allow or deny traffic. (Allow)
- Click **Save changes**.

Final rule should look like this:

| Rule number | Type | Protocol | Port range | Source | Allow/Deny |
|---|---|---|---|---|---|
| 100 | SSH (22) | TCP (6) | 22 | 0.0.0.0/0 | Allow |
| 200 | HTTP (80) | TCP (6) | 80 | 0.0.0.0/0 | Allow |
| 300 | HTTPS (443) | TCP (6) | 443 | 0.0.0.0/0 | Allow |
| 400 | Custom TCP | TCP (6) | 1024 - 65535 | 0.0.0.0/0 | Allow |

### **STEP 3:** Associate the NACL with Subnets

→ **Associate with Subnets**:

- In the VPC Dashboard, go to **Network ACLs**.
- Select the NACL you created.
- Click on the **Subnet Associations** tab.
- Click **Edit subnet associations**.
- Select the subnets you want to associate with this NACL.
- Click **Save associations**.

### **TEST:** Review and Test

→ **Review NACL Rules**:

- Ensure that the rules are properly configured for both inbound and outbound traffic.
- Verify that the rule numbers are correct and that rules are ordered properly.

→ **Test Connectivity**:

- Launch EC2 instances in the subnets associated with the NACL.
- Test the connectivity based on the rules you've configured (using ping, curl and SSH).

### Key Points to Remember

- **Rule Number**: NACLs process rules in ascending order of rule numbers. The first rule to match traffic will determine whether it is allowed or denied.
- **Default NACL**: Each VPC comes with a default NACL that allows all inbound and outbound traffic. You can modify it or create custom NACLs.
- **Stateless**: NACLs are stateless, meaning rules for inbound and outbound traffic are separate. If you allow inbound traffic, you must also allow the corresponding outbound traffic for responses.

### Summary

1. **Create a Network ACL**: Set up a new NACL in the VPC Dashboard.
2. **Configure Rules**: Define inbound and outbound rules to control traffic.
3. **Associate with Subnets**: Link the NACL to your subnets.
4. **Review and Test**: Verify rules and test connectivity to ensure proper functionality.

By following these steps, you can effectively manage and secure traffic to and from your subnets using NACLs in your VPC.