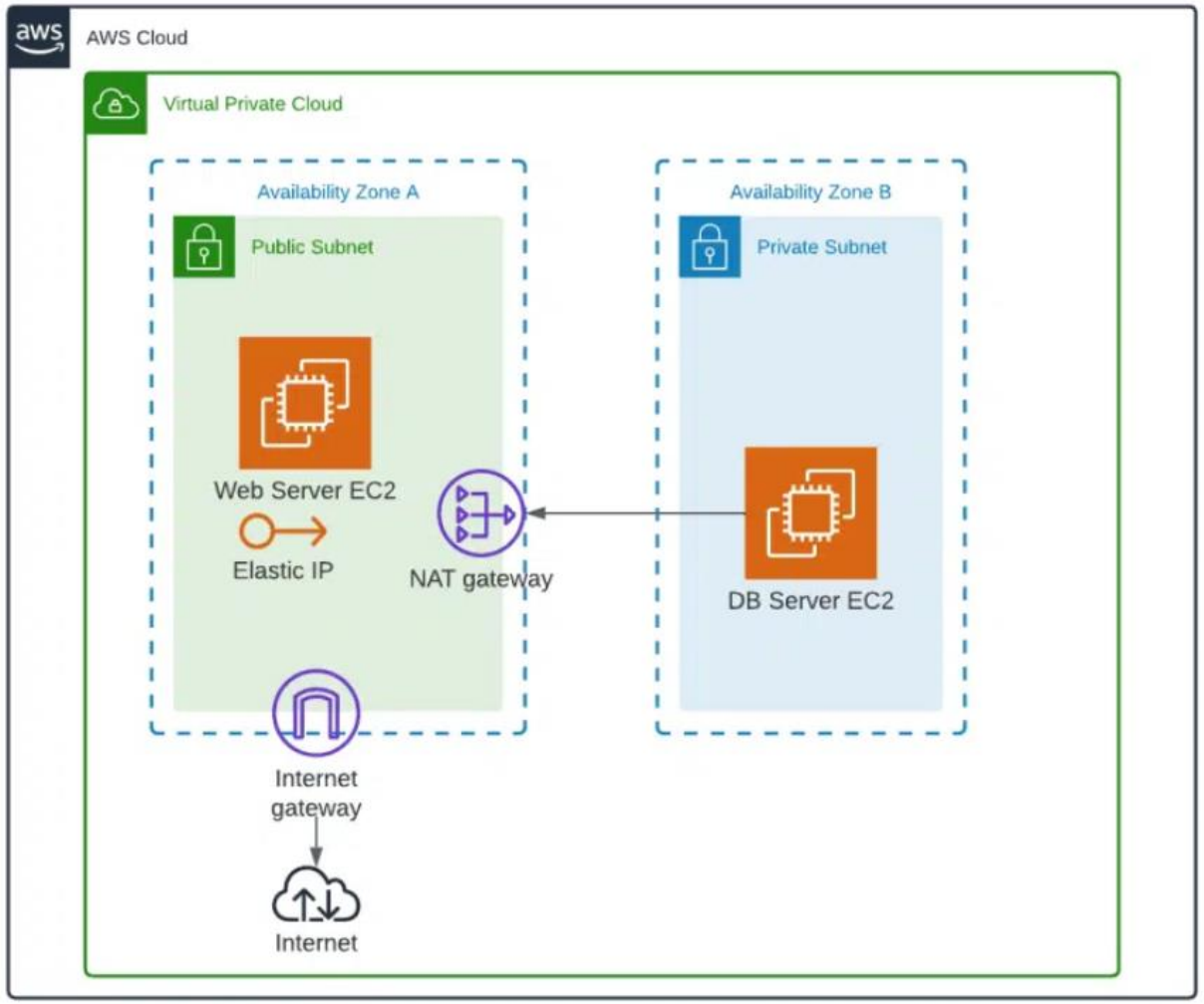
AWS Cloud
VPC (Virtual Private Cloud) +
EC2 (Elastic Cloud Computing)

VPC (Virtual Privacy Cloud)



By Default One VPC is Already Created

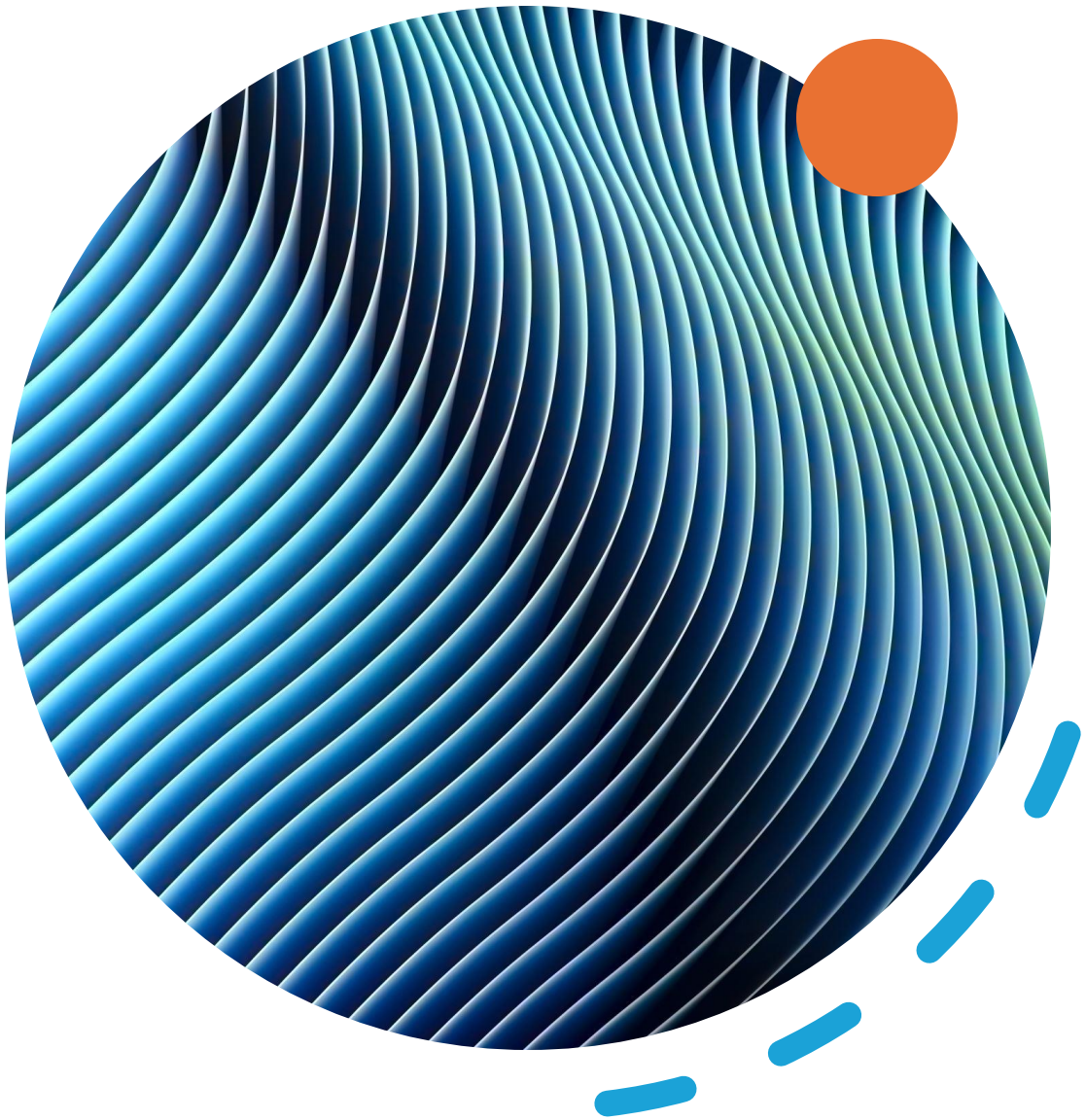
What Does the Default VPC Actually Do?

Launch EC2 Easily: It lets you **quickly launch EC2 instances** (virtual servers) that can **connect to the internet right away**, without needing to do extra setup.

Public IPs Assigned Automatically: When you launch an EC2 instance in the default VPC, it **automatically gets a public IP address**, so it can be accessed from the internet (if allowed by security settings).

Preconfigured Networking: It comes with all the **basic network setup already done** — like:

Internet Gateway for internet access	**Route tables** to manage traffic	**Security groups** and **network ACLs** for basic protection



Steps to Create VPC

- Aws Management Console
- VPC (Search)
- Create VPC
  - VPC only, Give Somename(test-VPC), IPv4 CIDR manual input (Range of IP Address)
  - Note:- CIDR.xyz website shows Range of IPs
  - No IPv6 CIDR block (select it because we don’t need to use ipv6)
  - Tenancy:- Default

  - VPC Created Successfully.

- What Does CIDR Do in a VPC?
- **CIDR** stands for **Classless Inter-Domain Routing,** In a **VPC (Virtual Private Cloud)**, the **CIDR block** defines **how big your private network is**—basically, **how many IP addresses** you can use inside that VPC.

- What Are IPv4 and IPv6?
- Ans:- Both are **types of IP addresses**, which are like **digital addresses** used to identify devices on a network (like your phone, computer, or server).
- IPV4 uses 32 bits to Create Address Around 4.3 Billion devices
- IPV6 Uses 128 Bits to Create Address They Can Create Trillions 7 Trillions Devices.

What is a CIDR Block?

- **CIDR** stands for **Classless Inter-Domain Routing** — sounds scary, but it's just a **way to define a range of IP addresses**.
- Example:- 10.0.0.0/16

- **What does it mean in simple words?**
- Let's break `10.0.0.0/16` into two parts:
- `10.0.0.0` → This is the **starting IP address**.
- /16 → This means **how many addresses** are included in the range.

CIDR BLOCK

- **Where is CIDR block used?**
- When creating a **VPC**, you define its **CIDR block** (the whole IP range inside your private network).

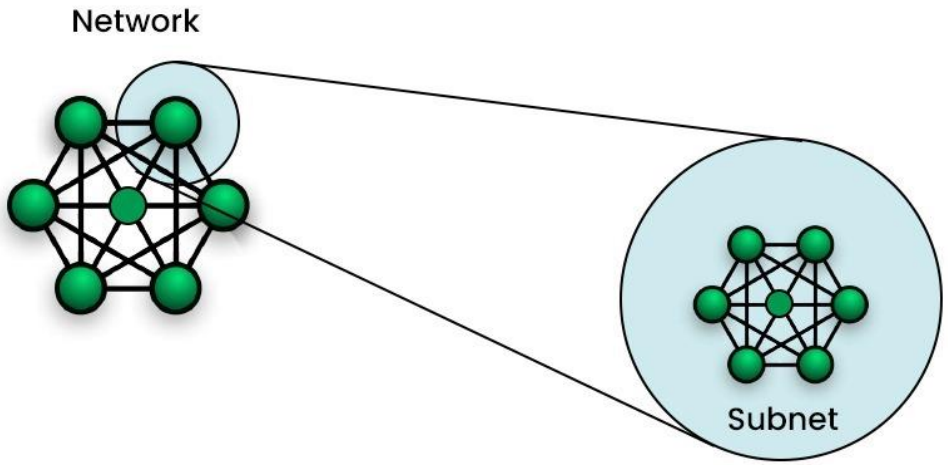
CIDR Block	Size (How many IPs)
/16	65,536 IP addresses
/24	256 IP addresses
/32	Just 1 IP address

Steps to Create Subnet

- Aws management Console
- VPC (Search)
- Subnets
- Create Subnets

- Give Subnet Name
- Select First Asia Pecific |ap-south-1

- Create Subnet



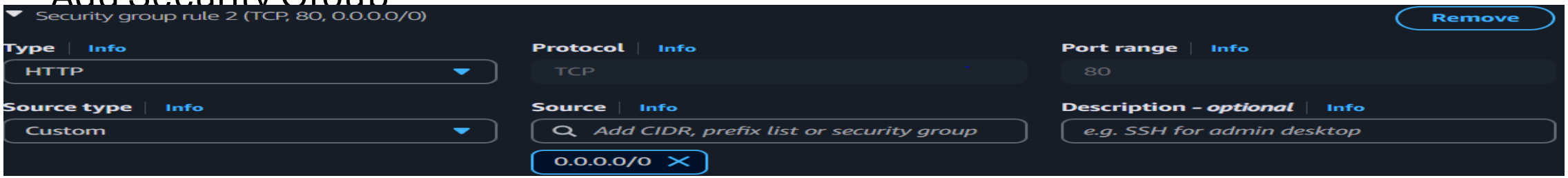
- What is Subnet ?
- Ans:- A **Subnet** (short for **sub-network**) is a **smaller part of a bigger network**.
- In AWS (or any network), when you create a VPC (which is your full private network), you can **divide it into smaller chunks** — these chunks are **Subnets**.

Why Use Subnets?

- To **organize your resources** (like EC2 instances) better
- To **control traffic** — for example, one subnet can connect to the internet, another one stays private
- To apply **security rules** to specific groups of resources

Create EC2 Instance

- Name (test-instance)
- Ubuntu (Free Tier)
- T2.micro
- Key pair name - required (Create + RSA+ .pem)
- Network Settings
- Edit --> Don’t Choose Default VPC ---> Choose our Test-VPC (which Created by Me)
- Auto-assign public IP ---> Disable to Enable
- Add Security Group



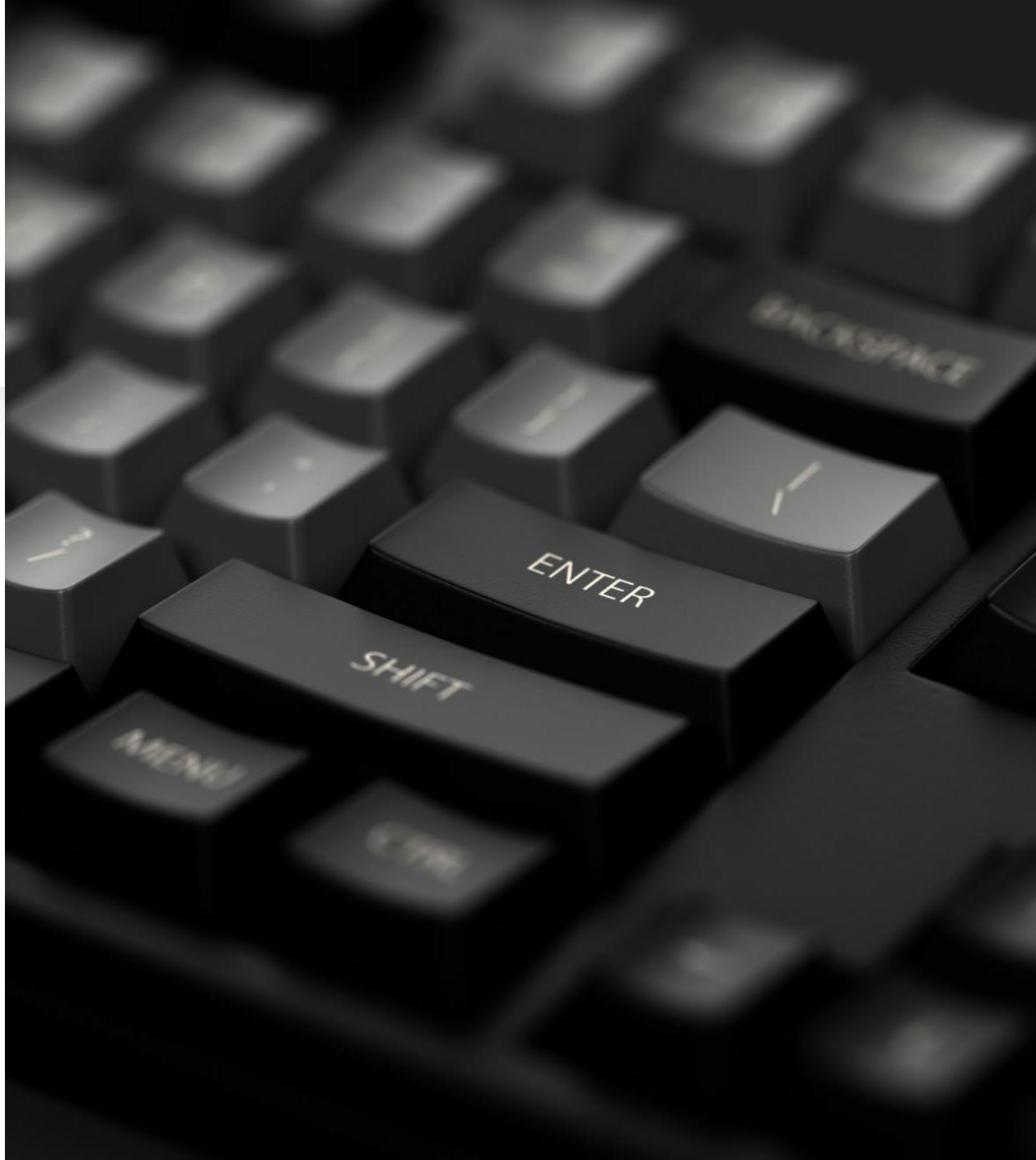
- Launch Instance
-

General steps to Create EC2 Instance

- Launch Instance
- Give Name of Instance
- AMI stands for Amazon Machine Image. (When you want to launch a new EC2 instance
- (a virtual server), you use an AMI to tell AWS, “Hey, I want a machine that looks like this.)
- -->The operating system (like Linux or Windows)
- -->Any software or applications you want
- -->The settings you need
- Select the Free Tier only
- instance type(CPU & Memory(RAM))
- Key Pair login (Helps to Access or Use Cloud Computer from Personal Computer)
- -->Give Name + Choose RSA+.pem ---> create key pair ---> One File Automatically gets Downloaded.
- Network Settings
- Firewall
- Allow SSH Traffic From ---> Anywhere (0.0.0.0/0)
- Allow HTTP Traffic from The Internet.
- Configure Storage
- Launch Instance

How to check Instance is Running or Not

- Login to AWS Manage Console
- Check Region ---> Hit Instance ---> Copy the Public IP Address (Its Dynamic)---> Paste
- On Chrome You can see its Working or not



Elastic IPs in EC2 Instance

- Note:- Dynamic IP Adress Means If You Check Same Thing from Different Device then
- The IP Address changes Everytime.

- So for that i need a proper static IP Address ? How Can i Buit that ?
- Ans:-
- Left Column has Network & Security Column
- choose Elastic IPs
- Allocate Elatic IP Address
- No need to change anything Just Hit Allocate.

- Fastly Connect Your IP Adress to Your Instance(Machin) Which You Created above
- Action (Inside Elastic Ips) ---> Associate Elastic IP Address ---> Instance
- ---> Choose On Which Instance You want to connect ---> Hit Associate.

- Now How to check You STatic IP Adress Is Connect with Your Instance or Machine ?
- Ans:- Go to Instance and check Public IP Adress The Dynamic Address gets Changed Into
- Static IP Adress Which You Connected with your machine. (THIS IP WOnt CHange now)

- Now This Instance is Running Inside VPC which Simply Means its Isolated from Outer Environment .If We try to Connect from The Instance (Running) Now then it won't Get Connected.

EC2 Instance

- EC2 Instance is stands for Elastic Cloud Compute.
- Which is a Computer or Server Located in Cloud.
- You can **turn it on**, **install software**, **run websites or apps**, and **access it from anywhere** — just like a real computer.

- Instead of buying a real laptop or server, you just **rent a virtual one** from AWS.
  You can choose:
- **How powerful it is (CPU, memory)**
- **What operating system it runs (Linux, Windows)**
- **How long you want to use it (pay per second or hour)**

Steps to Create Internet Gateway (it gives Internet Access to Subnet)

- AWS Management Console
- VPC
- Internet Gateway
- Create Internet Gateway
- Give name test-igw
- Create

Attaching VPC with Internet Gateway

- After Creating Internet Gateway You Automatically gets an option Attach VPC ---> Click on it.
- Or you can tick the Test-VPC & Click on Action & Select Attach
- Give test-VPC ---> Attach Internet Gateway.

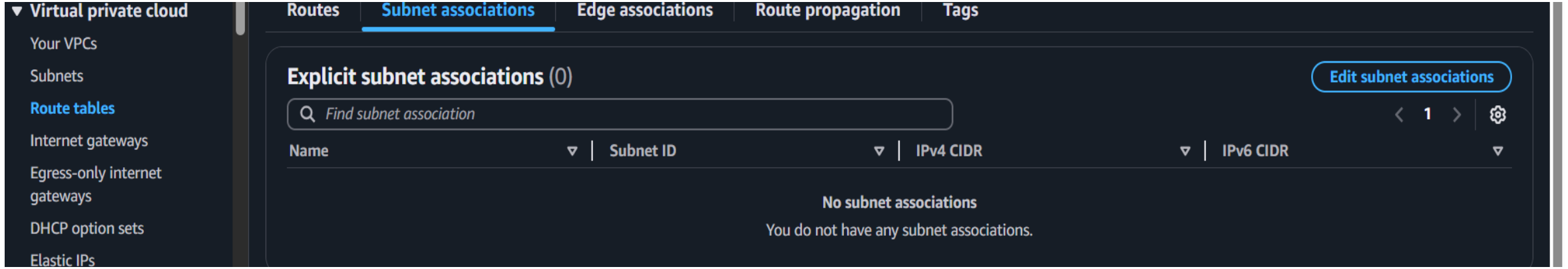
- Above Simply Means Internet Gateway Simply Comes Inside the VPC. (Both VPC still Not gets Connected with Internet Gateway for That We Need Routing Table Concepts this Connect)

Route Tables Inside VPC to Connect VPC to internet gateway.

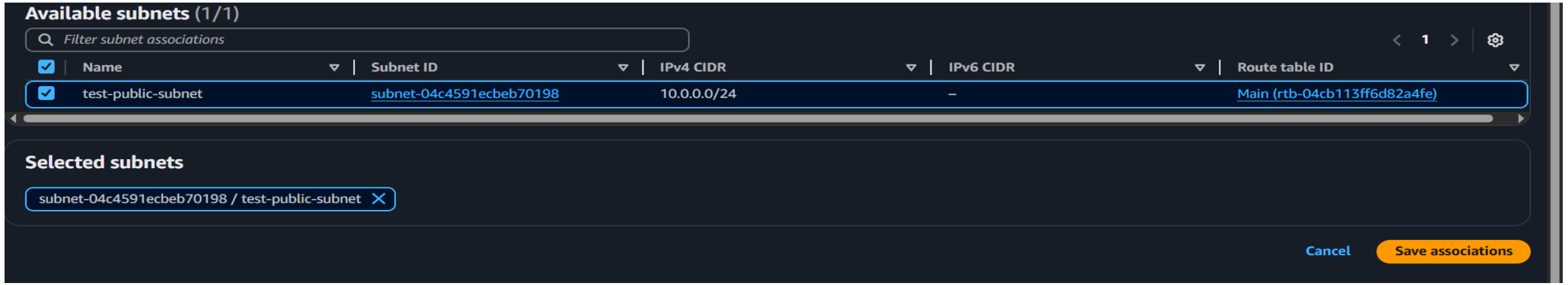
- Aws Management Console
- VPC
- Route Tables
- Create Route Tables
- Give name
- Test VPC (select)
- Create Route Table.
- However, Still They Don’t Know What to do

- Next Page

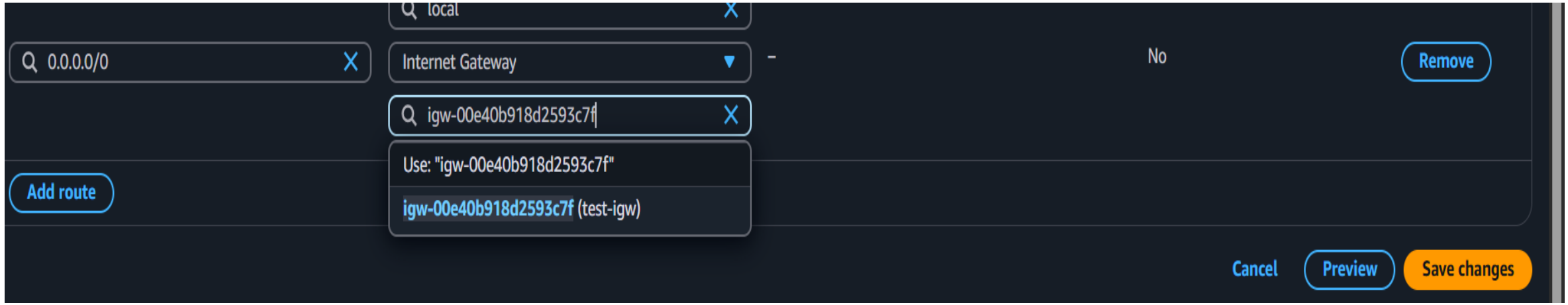
- Inside Route Table
- Subnet Association



- Edit subnet Association.
- Save Association



- This Means The Route Table Connects with Public Subnets After Save Association.
- Go to Route inside Route Table (giving one Route Which Actually Connects with Internet gateway)
- Edit Route.



After Save Changes

- It Means Internet Gateway + Route Tables + Public Subnet All Are Connect with Other **Successfully.**

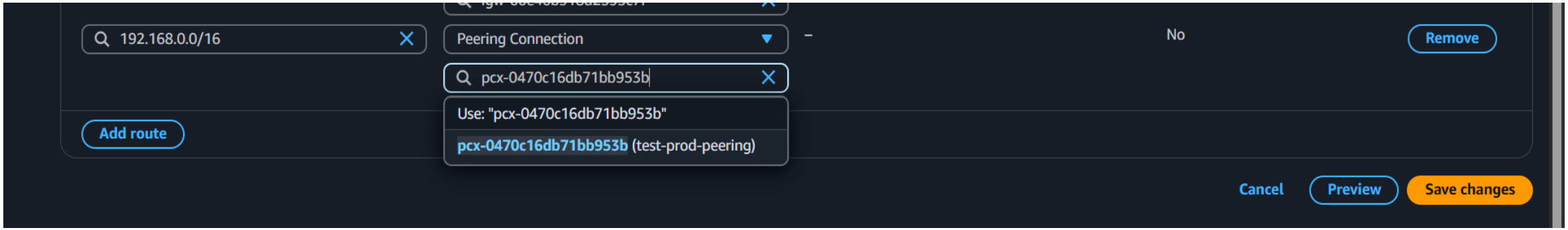
- Now If I am going to Run My EC2 Instance then it will Successfully Connect

Test VPC & Prod VPC Created now do VPC Peering Means

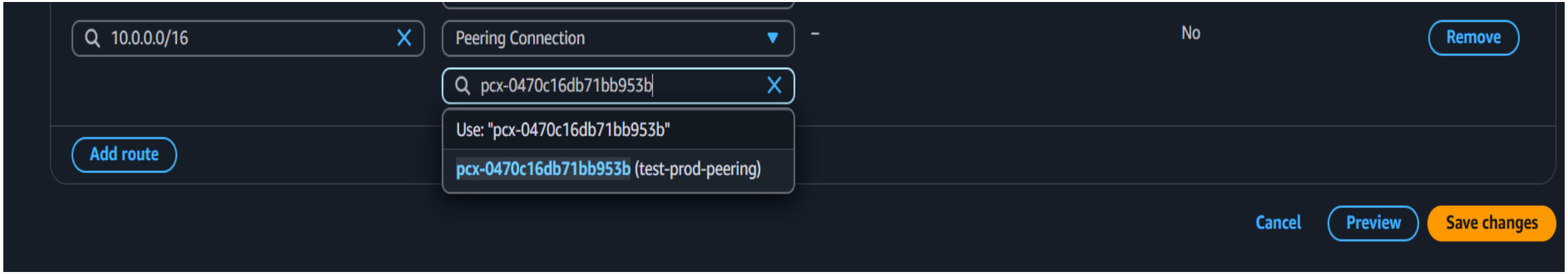
- One VPC can PING (ICMP ---> Internet Control Message Protocol)the Another VPC
- Aws
- VPC
- Peering Connection
- Create Peering Connections
- VPC ID (Requester) :-Select first VPC (test-VPC)
- VPC ID (Accepter) :- select the Second (Prod-VPC)
- Create Peering Connection.
- Until I doesn’t Accept the Peering Connection It Shows Pending
- Action ---> Accept Request
- Still TEST VPC can't PING (Sent Request) to PROD VPC. For Connecting we Need to Route Table to Connect two VPC.

- Go to route table &
- Select test ---> Go below Route ---> Edit Route ---> Add Route

Give Sider IP

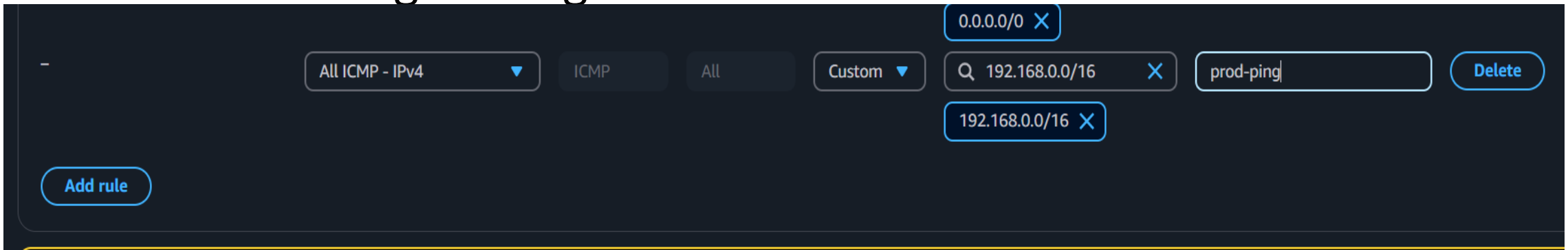


Do same thing for prod

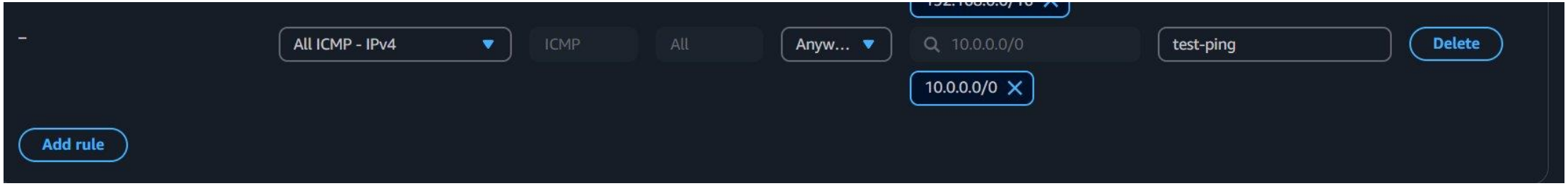


Still Test VPC Not Able to PING Prod VPC

- Aws Management Console
- Ec2
- Instance
- Tick the Test-instance ---> Security --> Security group ---> Inside Inbound Rule ---> Add Rule
- Type:- ALL ICMP-IPv4
- Inside test VPC give range or PROC VPC & Vice Versa



Save Changes



- Save Changes

- Connect Test VPC instance & type:- ping 192.168.0.150
- PING Comes That Means Its Successfully Connected.

- Connect Proc VPC Instance & Type:- 10.0.0.103
- PING Comes That Means Its Successfully Connected.

VPC Peering Works Successfully

```
ubuntu@ip-10-0-0-16:~$ ping 192.168.0.192
PING 192.168.0.192 (192.168.0.192) 56(84) bytes of data.
64 bytes from 192.168.0.192: icmp_seq=1 ttl=64 time=2.88 ms
64 bytes from 192.168.0.192: icmp_seq=2 ttl=64 time=0.767 ms
64 bytes from 192.168.0.192: icmp_seq=3 ttl=64 time=1.24 ms
```

i-0c9e7a85dbf929c04 (test-instance)

PublicIPs: 13.201.5.66 PrivateIPs: 10.0.0.16

```
ubuntu@ip-192-168-0-192:~$ ping 10.0.0.16
PING 10.0.0.16 (10.0.0.16) 56(84) bytes of data.
64 bytes from 10.0.0.16: icmp_seq=1 ttl=64 time=0.709 ms
64 bytes from 10.0.0.16: icmp_seq=2 ttl=64 time=0.871 ms
64 bytes from 10.0.0.16: icmp_seq=3 ttl=64 time=0.695 ms
64 bytes from 10.0.0.16: icmp_seq=4 ttl=64 time=1.04 ms
64 bytes from 10.0.0.16: icmp_seq=5 ttl=64 time=1.50 ms
64 bytes from 10.0.0.16: icmp_seq=6 ttl=64 time=0.756 ms
64 bytes from 10.0.0.16: icmp_seq=7 ttl=64 time=0.616 ms
64 bytes from 10.0.0.16: icmp_seq=8 ttl=64 time=0.622 ms
64 bytes from 10.0.0.16: icmp_seq=9 ttl=64 time=0.841 ms
64 bytes from 10.0.0.16: icmp_seq=10 ttl=64 time=0.830 ms
64 bytes from 10.0.0.16: icmp_seq=11 ttl=64 time=1.32 ms
64 bytes from 10.0.0.16: icmp_seq=12 ttl=64 time=1.10 ms
64 bytes from 10.0.0.16: icmp_seq=13 ttl=64 time=0.923 ms
```

i-0d4a3bf5974d11064 (prod-instance)

PublicIPs: 3.110.47.232 PrivateIPs: 192.168.0.192

VPC Peering

- Imagine you have **two separate houses** (VPCs).
  Each house has its own **rooms (subnets)**, people (servers), and **fence (security)**.
- By default, people in House A **cannot talk** to people in House B.
- Now, you want them to **communicate privately**, without using the public internet.
  You build a **private tunnel** (like a secret bridge) between the two houses(VPCs).
- 🎯 **That bridge is called VPC Peering.**

VPC Peering

- ✅ **What Does It Do?**
- VPC Peering allows **two different VPCs** to:
- Talk to each other **privately**.
- Share data and resources **without using the internet**.

- **Is it secure?**
- Yes! It's like a **private line** between your networks. No internet involved

- 🚫 **One thing to remember:**
- VPC Peering is **one-to-one**. If VPC A is peered with VPC B, and B is peered with C, **A cannot talk to C** unless there’s a separate peering.

Route Tables

- A **Route Table** in AWS is like a **map** that tells your network (VPC) **where to send traffic**.
- It helps your resources (like EC2 instances) know **how to reach other networks**, like:
- The **internet**
- **Other subnets** in your VPC
- **VPNs** or **on-premise networks**

- How It Works:
- Each **subnet** in your VPC is connected to **one route table**.

VPC Route Tables

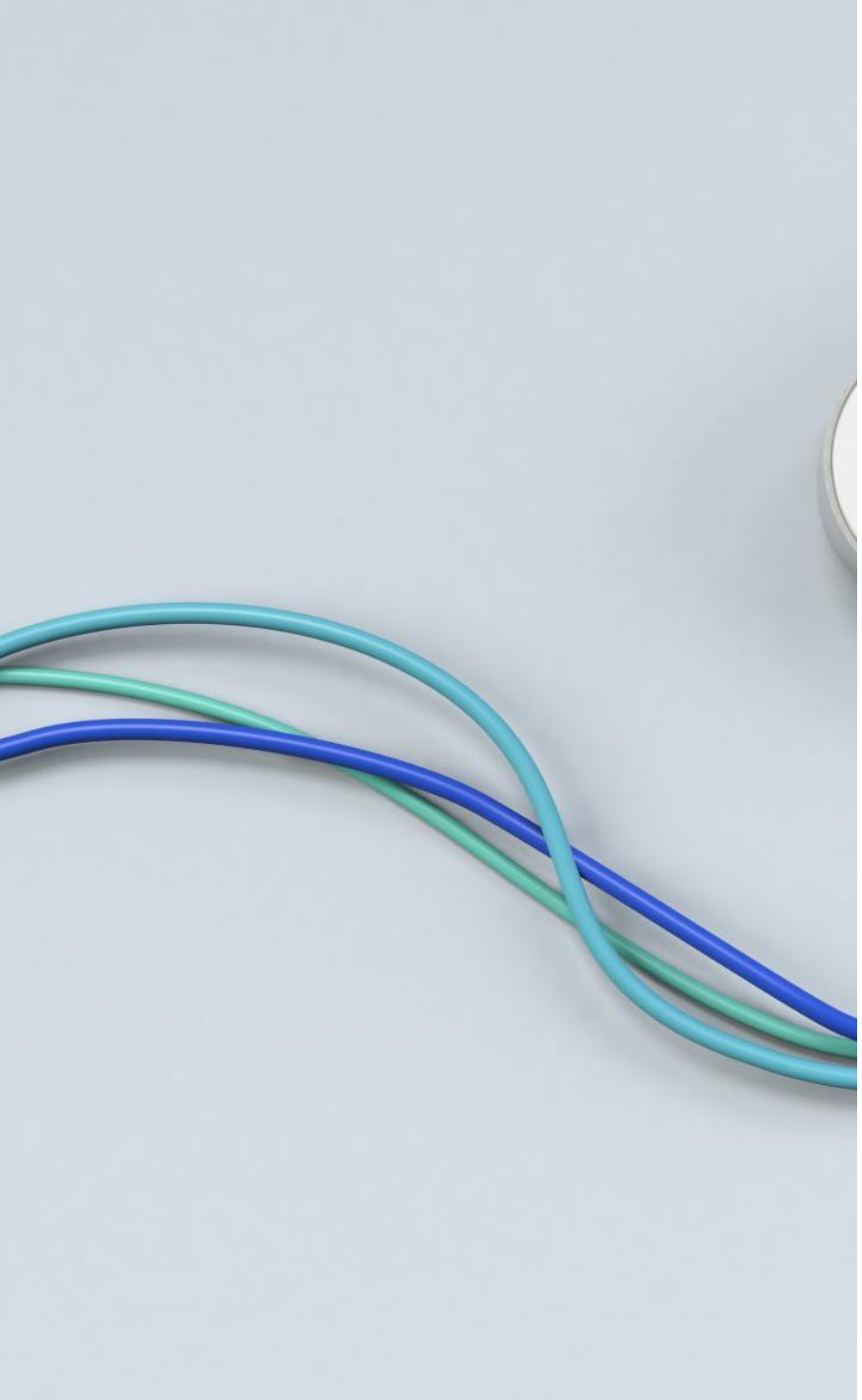
Without Route Table Updates?

Even if the VPC Peering is created, the servers **won’t talk**, because they don't know how to reach each other. Route tables are like saying:

"Here's the road to get there."

ĂĎĢJĬHĂĖĤĴKĪĂĖĴĤĪJĆĂĂĄƐĖĐĂĖĤĤĨĨIJỀ

- Go to AWS VPC → Create Peering Connection.
- Choose the two VPCs (can be in the same or different AWS accounts).
- Accept the peering request from the other VPC.
- Update **route tables** in both VPCs to allow traffic.



ICMP & VPC Peering

- **What is VPC Peering?**
- **VPC Peering** connects two VPCs (Virtual Private Clouds) so they can **talk to each other** — like connecting two private networks.

- **ICMP** stands for **Internet Control Message Protocol**.
  It's used by computers to **send test messages** to each other — like:
- **"Are you there?"** (ping)

- If One VPC Doest Accept the Request Then Other Can't Connect with the Other VPC.
- If One VPC Accept the Request then Both Can Communicate With Each Other.

General Notes

- **Internet Gateway** is connected to the **Public Subnet** to allow internet access.
- **NAT Gateway** is placed in the **Public Subnet** to give internet access to instances in the **Private Subnet**, without exposing them to the internet.
- Each **Subnet** in a VPC has a **Route Table** that defines **where to send traffic** (e.g., to Internet Gateway or NAT Gateway).
- A **Subnet** is a **smaller part of a VPC** (Virtual Private Cloud).
- Inside a **VPC**, we can create **multiple subnets** such as **Public Subnet** and **Private Subnet**.
- Inside each **Subnet**, we can launch **EC2 Instances** (virtual servers in the cloud).
- **Elastic IP** provides a **static public IP** to an EC2 instance, so the IP address stays the same.
- By default, EC2 instances get **dynamic public IPs** that can change if the instance is stopped and started again.
- **Elastic IP** is useful when you want **consistent access** to your EC2 instance over the internet.

Each Terms In VPC Explained In Simple words

- 🟩 **AWS Cloud (Rent Powerful Computer & Pay as much as you use their Service)**
- This is the big environment provided by **Amazon Web Services** where you can run your apps and services (like websites, databases, etc.).

- 🟢 **Virtual Private Cloud (VPC)**
- Think of this as your **own private network** inside the AWS cloud. Only your resources (like servers) live in this secure space.

- 🟦 **Availability Zone (AZ)**
- These are **data centers in different locations**. You use them for better performance and reliability:
- **Availability Zone A**
- **Availability Zone B**

- 🟩 **Public Subnet**
- A part of your network that is **open to the internet**. Resources in here can send/receive data from the internet.
- Example: **Web Server**
- 🟦 **Private Subnet**
- A part of your network that is **not directly connected to the internet**. It is more secure.
- Example: **Database Server**

- 🟧 **Web Server EC2**
- EC2 = Elastic Compute Cloud = **A virtual server**
- Web Server EC2 is a server that runs your **website or web app**.
- It lives in the **public subnet**.

- 🟧 **DB Server EC2**
- Another virtual server, but this one runs your **database**.
- It is placed in the **private subnet** for **security**, so it can’t be accessed directly from the internet.

- 🟠 **Elastic IP**
- A **permanent public IP address** assigned to your Web Server. This lets users from the internet reach your web server.

🟣 **Internet Gateway**

This is what **connects your VPC to the internet**. It allows the web server (with Elastic IP) to send and receive internet traffic.

🟣 **NAT Gateway:**

- Goes to the internet,
- Gets the data,
- Brings it back to the private room.

And guess what? **The internet never knows where the request really came from.**

It only sees **NAT Gateway**, not your private server. So your private server stays **safe and invisible**.

Final Setup in Simple Words:

- NAT Gateway **lives in public subnet** (with internet access).
- Private subnet **sends its internet requests to NAT Gateway**.
- NAT Gateway fetches from the internet and **returns the data back to private subnet**, while keeping it hidden.

HOW NAT GATEWAY IS CONNECTED:

- Step 1: **Create a NAT Gateway in the Public Subnet**
- Go to AWS Console → VPC → NAT Gateways.
- When creating it, AWS will **ask you to choose a public subnet** — this is where the NAT Gateway will **live**.
- Also, you must **attach an Elastic IP** to it — this gives it internet access.
- 📌 **Why in public subnet?** Because NAT Gateway needs **access to the internet**, and only public subnets can directly talk to the internet through an **Internet Gateway**.

- Step 2: **Update the Route Table for the Private Subnet**
- Go to VPC → Route Tables.
- Find the **route table attached to your private subnet**.
- Edit the routes and **add a new route**:
  - **Destination:** `0.0.0.0/0` (which means "all internet traffic")
  - **Target:** Choose the **NAT Gateway** you created earlier.
  - 📌 **Why do this?** This tells the private subnet:
  - "Hey, if you need to talk to the internet, don’t go directly — go through this NAT Gateway."

- ✅ Step 3: **Ensure the Public Subnet Has a Route to the Internet**
- The **public subnet** where NAT Gateway lives should have a route:
  - **Destination:** `0.0.0.0/0`
  - **Target:** Internet Gateway
  - This allows the NAT Gateway to **reach out to the internet**.

THANK YOU