VPC Peering

🌐 What is VPC Peering?

Imagine you have **two houses** (VPCs) in different neighborhoods.
By default, the people inside these houses **cannot talk to each other**.

Now, if you want them to communicate **securely and privately**, you can **build a private road** between the two houses.
That **private road** is called **VPC Peering**.

🧠 Why Do We Need VPC Peering?

Let's say:

- You have one VPC for your **frontend app**,
- And another VPC for your **database backend**.

But they are in **different VPCs**, maybe even in **different AWS accounts**.

To make your app talk to the database **privately (not over internet)** — you use **VPC Peering**.

🚪 What Happens After Peering?

Once VPCs are peered:

- EC2 in VPC A can access EC2 in VPC B using **private IP**.

- No need to use **public IP**, **internet**, or **VPN**.
- It's **faster**, **secure**, and **low cost**.

📝 **Important Notes for New Learners**

- VPC Peering is **one-to-one** (A <--> B).
  If you need many-to-many, use **Transit Gateway**.
- It's used within **same region** or **between regions** (called inter-region VPC peering).
- You must **update route tables and security groups** to allow traffic after peering.

🧩 **Real-life Example:**

You work for a company with:

- A **DevOps team** in one VPC managing tools (like Jenkins, Prometheus).
- A **Development team** in another VPC running apps.

You use **VPC Peering** so DevOps tools can monitor or deploy to the Development team's servers — **without using the internet**.

Project Title: VPC Peering

🧠 Introduction:

Amazon VPC (Virtual Private Cloud) allows you to launch AWS resources in a logically isolated network. Sometimes, you might want two VPCs to communicate with each other (like when microservices are deployed across VPCs). VPC Peering is a way to connect two VPCs so that resources in them can communicate using private IP addresses.

This project will walk you through:

- Creating two VPCs
- Creating subnets and EC2 instances in both VPCs
- Establishing VPC peering
- Testing connectivity using private IPs

🛠️ Project Setup

✅ Requirements:

- AWS Free Tier account
- Basic knowledge of AWS Console
- IAM permissions to create VPC, EC2, and Networking resources

---

🗂️ Step-by-Step Guide

Step 1: Create Two VPCs

- Go to the VPC Dashboard
- Create VPC-A:
    - CIDR block: 10.0.0.0/16
- Create VPC-B:
    - CIDR block: 10.1.0.0/16

Step 2: Create Subnets

Create one public subnet in each VPC:

- VPC-A: 10.0.1.0/24
- VPC-B: 10.1.1.0/24

Step 3: Launch EC2 Instances

- Launch one EC2 instance in each VPC using the subnets you created
- Allow SSH (port 22) and ICMP (ping) in security groups
- Note down the private IPs of both EC2 instances

Step 4: Create VPC Peering Connection

- Go to VPC > Peering Connections
- Click **Create Peering Connection**
- Choose:
    - Requester VPC: VPC-A
    - Accepter VPC: VPC-B
- Accept the peering request from the VPC-B side

Step 5: Update Route Tables

- Go to the Route Tables of both VPCs
- In VPC-A's route table:
    - Destination: 10.1.0.0/16
    - Target: The Peering Connection
- In VPC-B's route table:
    - Destination: 10.0.0.0/16
    - Target: The Peering Connection

Step 6: Test Connectivity

- SSH into the EC2 instance in VPC-A
- Ping the private IP of the EC2 instance in VPC-B

- If ping works, your VPC Peering is successful

📘 **What You Learn:**

- How AWS VPCs work
- Basics of IP ranges and subnetting
- Route table configuration
- Establishing VPC Peering
- Testing private network connectivity

🔄 **Cross-Region VPC Peering**

🧠 **Introduction:**

AWS VPCs in different regions can also be peered, enabling global application communication securely.

🛠️ **Steps:**

1. **Create VPCs in Different Regions**
    - VPC-A in us-east-1 (CIDR: 10.0.0.0/16)
    - VPC-B in us-west-1 (CIDR: 10.1.0.0/16)
2. **Launch EC2 Instances**

- One EC2 in each VPC's public subnet

3. **Create VPC Peering (Cross-Region)**
    - Go to VPC > Peering Connections > Create Peering
    - Requester: VPC-A (us-east-1)
    - Accepter: VPC-B (us-west-1)
4. **Accept Peering Request**
    - Accept from the us-west-1 VPC dashboard
5. **Update Route Tables**
    - Add routes in both VPCs for each other's CIDR via the peering connection
6. **Test with Private IP**
    - SSH and ping across EC2s using private IPs

🌐 Add Private Hosted Zone (Route 53) for DNS Between VPCs

🧠 Introduction:

Private Hosted Zones enable you to access resources by custom internal DNS names (e.g., web.internal.local) between peered VPCs.

🛠️ Steps:

1. **Create a Private Hosted Zone**

- Route 53 > Hosted Zones > Create Hosted Zone
- Domain: internal.local
- Type: Private
- Associate with VPC-A

2. **Add DNS Record**
   - Name: web.internal.local
   - Type: A
   - Value: Private IP of EC2 in VPC-B
3. **Associate the Hosted Zone with VPC-B**
   - Edit Hosted Zone > Associate another VPC (select VPC-B)
4. **Update EC2 DNS (optional)**
   - Ensure DNS resolution is enabled (Amazon Linux enables it by default)

5. **Test DNS Resolution**
   From EC2 in VPC-A:

   nslookup web.internal.local

   ping web.internal.local

🔐 Set Up Security Groups and NACLs for Tighter Control

🧠 Introduction:

VPC Peering allows traffic, but access still depends on security groups and NACLs to control what traffic is allowed.

🛠️ Steps:

1. **Security Groups**
    - On both EC2s:
        - Allow inbound ICMP (ping) and SSH (port 22)
        - Source: CIDR of peer VPC
2. Example for EC2 in VPC-B:
    - Type: Custom TCP Rule
    - Port: 22
    - Source: 10.0.0.0/16 (VPC-A)
3. **Network ACLs**
    - Modify NACLs of subnets to allow:
        - Inbound/Outbound:
            - Protocol: TCP / ICMP

- Port: 22 / All
- Source/Destination: Peer VPC CIDR

4. Example Rule:

Rule #	Type	Protocol	Port	Source/Dest	Action
100	SSH	TCP	22	10.0.0.0/16	ALLOW
110	All	All	All	0.0.0.0/0	DENY

Note: Security Groups are stateful, NACLs are stateless.

🧩 **Final Architecture Summary**

- VPC-A (us-east-1) and VPC-B (us-west-1)
- Connected using Cross-Region VPC Peering
- EC2s secured with proper Security Groups and NACLs
- Route 53 Private Hosted Zone for internal DNS (web.internal.local)

Why Use VPC Peering?

- Connects two separate AWS networks so they can share data securely.
- Keeps your communication private, not over the public internet.
- Makes your apps run faster by using AWS's internal network.
- Saves money since no need for extra hardware or VPNs.
- Easy to set up, even for beginners.
- Supports connections across different regions for global reach.

VPC Peering vs Transit Gateway

VPC Peering vs Transit Gateway: When to Use Which

When to Use VPC Peering

- You have **just two VPCs** (or very few) that need to communicate directly.
- You want a **simple, low-cost** solution.
- You **don't need transitive routing** (no communication through one VPC to another).
- You want **low-latency, private communication** inside the AWS network.
- You can manage **manual route table updates** for each peering connection.
- Your VPC CIDR ranges **do not overlap** (required for peering).

When to Use Transit Gateway

- You have **multiple VPCs** (dozens or more) or expect to scale your network.
- You want **centralized routing** — connect many VPCs, on-premises data centers, or VPNs via one hub.
- You need **transitive routing** (VPC A can talk to VPC C via Transit Gateway if both attached).
- You want **simpler management** without managing many peering connections.
- You are OK with the **additional cost** for Transit Gateway usage.
- You need to connect **across regions easily** (supports cross-region peering).
- Your VPC CIDR ranges **may overlap** but you can configure routing accordingly with Transit Gateway.

Quick Summary Table

Scenario	Use VPC Peering	Use Transit Gateway
Number of VPCs	2 or few	Many (10s or more)
Network complexity	Simple	Complex / large scale
Routing	Direct, no transitive	Centralized, supports transitive
Cost	No extra fee except data	Additional Transit Gateway fees

Management	Manual routes per peering	Centralized route management
Cross-region support	Supported but manual	Supported and easier to manage
On-premises / VPN integration	Not supported	Supported via Transit Gateway Attachments