1. Create one VPC, with 1 public subnet and 1 private subnet.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Enable VPC peering for cross-region.

**A. Launch pub-server in Stockholm (eu-north-1)**

1. In the AWS console, region is set to **Europe (Stockholm)**.
2. Go to **EC2 → Launch instance**.
3. Under **Name and tags**, set:
   * **Name** = pub-server

**B. Launch pub-two-hyd in Hyderabad (ap-south-2)**

1. Change region in the top-right to **Asia Pacific (Hyderabad)**.
2. Go to **EC2 → Launch instance** again.
3. Under **Name and tags**, set:
   * **Name** = pub-two-hyd

**C. Create cross-region VPC peering (Stockholm → Hyderabad)**

1. Switch back to **Europe (Stockholm)** region.
2. Go to **VPC → Peering connections**.
3. Click **Create peering connection**.
4. Fill **Peering connection settings**:
   * **Name** = peering-01
5. In **Select a local VPC to peer with**:
   * **VPC ID (Requester)** = vpc-068d558547ab00f25 (my-vpc-01)
   * This VPC has CIDR 172.168.0.0/24.
6. In **Select another VPC to peer with**:
   * **Account** = *My account*
   * **Region** = **Another Region** → choose **Asia Pacific (Hyderabad) (ap-south-2)**
   * **VPC ID (Accepter)** = vpc-09197ad95e1cc0cce (the default VPC in Hyderabad).
7. Click **Create peering connection**.
   * A new peering pcx-0f3aff59a02e7a4a2 is created with status **Pending acceptance**.

**D. Accept the peering request in Hyderabad**

1. Switch region (top-right) to **Asia Pacific (Hyderabad)**.
2. Go to **VPC → Your VPCs** and confirm you have VPC:
   * vpc-09197ad95e1cc0cce with CIDR 172.31.0.0/16.
3. In the left menu, click **Peering connections**.
4. You see the peering connection from Stockholm with status **Pending acceptance**.
5. Select that peering connection row.
6. Click **Actions → Accept request**.
7. A confirmation popup **“Accept VPC peering connection request”** shows:
   * **Requester VPC** = vpc-068d558547ab00f25, Region = Stockholm, CIDR 172.168.0.0/24
   * **Accepter VPC** = vpc-09197ad95e1cc0cce, Region = Hyderabad, CIDR 172.31.0.0/16
8. Click **Accept request**.
9. The peering connection status changes to **Active**.

* **Update route tables** in both VPCs:
  + In **Stockholm VPC (172.168.0.0/24)**: add a route to 172.31.0.0/16 with target = the **peering connection**.
  + In **Hyderabad VPC (172.31.0.0/16)**: add a route to 172.168.0.0/24 with target = the **same peering connection**.

**Check CIDR Block of Hyderabad VPC**

Region: **Asia Pacific (Hyderabad)**

1. Go to **VPC → Your VPCs**
2. Verify CIDR: 172.31.0.0/16

**Edit Route Table in Hyderabad Region (Accepter VPC)**

Region: **Asia Pacific (Hyderabad)**  
 VPC: vpc-09197ad95e1cc0cce → CIDR: 172.31.0.0/16

1. Go to **VPC → Route tables**
2. Select **Main route table**
3. Click **Routes → Edit routes**
4. Add new route:
   * **Destination**: 172.168.0.0/24
   * **Target**: Select **Peering Connection** (pcx-0f3aff59a02e7a4a2)
5. Click **Save changes**

This allows return traffic from Hyderabad to Stockholm.

**Check Private IP of Hyderabad EC2 instance**

Region: **Hyderabad**

1. EC2 Dashboard → Instances
2. Select instance **pub-server-hyd**
3. Note the **Private IP**:
   * 172.31.35.127

**Test Connectivity from Stockholm EC2 → Hyderabad EC2**

Connected via SSH to **Stockholm EC2** (172.168.0.6)  
 Run ping command:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with a white background

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with a red arrow pointing to a white box

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Enable VPC peering for cross-account (you can collaborate with your friend to do this task).

**Step-1: Create VPC Peering (Requester Side)**

Region: Europe (Stockholm)

Navigate → **VPC Console → Peering Connections → Create Peering Connection**

Fill in:

| **Field** | **Value** |
| --- | --- |
| Name | peering-with-friend |
| VPC (Requester) | vpc-068d558547ab00f25 |
| Account | **Another Account** |
| Friend AWS Account ID | **155499699692** |
| Region | **US East (N. Virginia)** |
| VPC ID (Accepter) | vpc-06e41558dd417fc5c2 |

Click **Create Peering Connection**

Status → **Pending Acceptance**

**Step-2: Accept Peering Request (Accepter Side)**

Region: US-East-1 (Friend)

Log in to friend AWS account  
 Switch region → **N. Virginia**  
 Go to **VPC → Peering Connections**  
Select the pending connection  
 **Actions → Accept Request → Confirm**

**Step – 3 Configure Route Tables on Both Sides**

Routing must be bidirectional to allow communication.

**Requester Route (Stockholm)**

Go to:  
**VPC → Route Tables → public-route-igw → Edit routes**

Add:

| **Destination (Friend VPC CIDR)** | **Target** |
| --- | --- |
| 172.31.0.0/16 | Peering Connection (pcx-0ca8d9854d3aeba50) |

Save routes

**Accepter Route (N. Virginia)**

Go to:  
**VPC → Route Tables → Main Route Table → Edit routes**

Add:

| **Destination (Your CIDR)** | **Target** |
| --- | --- |
| 172.168.0.0/24 | Peering Connection (pcx-0ca8d9854d3aeba50) |

Save routes

**Step-4: Configure Security Groups on Both EC2 Instances**

Allow private traffic from each other’s VPC networks.

**Stockholm EC2 SG**

| **Type** | **Protocol** | **Port Range** | **Source** |
| --- | --- | --- | --- |
| All Traffic | All | All | 172.31.0.0/16 |

**N. Virginia EC2 SG**

| **Type** | **Protocol** | **Port Range** | **Source** |
| --- | --- | --- | --- |
| All Traffic | All | All | 172.168.0.0/24 |

**Step-5: Private Network Connectivity Test**

From Stockholm EC2 → Friend EC2:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

1. Set up a VPC Transit Gateway.

**VPC Setup Overview**

| **VPC Name** | **CIDR** | **Region** |
| --- | --- | --- |
| **my-vpc-01** | 172.168.0.0/24 | Europe (Stockholm) |
| **vpc-2** | 10.0.0.0/28 | Europe (Stockholm) |
| **vpc-3** | 10.0.1.0/28 | Europe (Stockholm) |
| **vpc-4** | 10.0.2.0/28 | Europe (Stockholm) |

**Step-1: Create Subnets in Each VPC**

Example:

| **VPC** | **Subnet CIDR** |
| --- | --- |
| my-vpc-01 | 172.168.0.0/26 |
| vpc-2 | 10.0.0.0/28 |
| vpc-3 | 10.0.1.0/28 |
| vpc-4 | 10.0.2.0/28 |

Each subnet must be **Available** and **auto-assign IPv4 turned ON** if public.

**Step-2: Launch EC2 Instances (1 per VPC)**

| **Instance Name** | **VPC** | **Private IP Example** |
| --- | --- | --- |
| vpc-1-ec2 | my-vpc-01 | 172.168.0.6 |
| vpc-2-ec2 | vpc-2 | 10.0.0.12 |
| vpc-3-ec2 | vpc-3 | 10.0.1.46 |
| vpc-4-ec2 | vpc-4 | 10.0.2.13 |

All in **running** state  
 Allow SSH with your key pair

**Step-3: Create Transit Gateway (TGW)**

**VPC Console → Transit Gateways → Create TGW**

**Step-4: Create TGW Attachments for Each VPC**

**Transit gateway attachments → Create attachment**

| **Attachment Name** | **VPC** |
| --- | --- |
| vpc-1-tga | my-vpc-01 |
| vpc-2-tga | vpc-2 |
| vpc-3-tga | vpc-3 |
| vpc-4-tga | vpc-4 |

State shows **Available**

**Step-5: Update Route Tables in Each VPC**

**Example Routing (for vpc-2 route table)**

| **Destination** | **Target** |
| --- | --- |
| 10.0.0.0/28 | local |
| 10.0.1.0/28 | Transit Gateway |
| 10.0.2.0/28 | Transit Gateway |
| 172.168.0.0/24 | Transit Gateway |

Repeat same logic:

**Routing summary**

| **VPC** | **Needs route to** | **Target** |
| --- | --- | --- |
| **my-vpc-01** | All 10.0.x.x CIDRs | TGW |
| **vpc-2** | 172.168.0.0/24 & other 10.x | TGW |
| **vpc-3** | 172.168.0.0/24 & other 10.x | TGW |
| **vpc-4** | 172.168.0.0/24 & other 10.x | TGW |

Save routes on all VPC route tables

**Step-6: Modify Security Groups**

Edit inbound rules to allow internal network:

| **Type** | **Port** | **Source** |
| --- | --- | --- |
| All Traffic | All | 172.168.0.0/16 |
| All Traffic | All | 10.0.0.0/24 |

This allows ping/SSH inside the TGW network

**Step-7: Connectivity Test Between VPCs**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

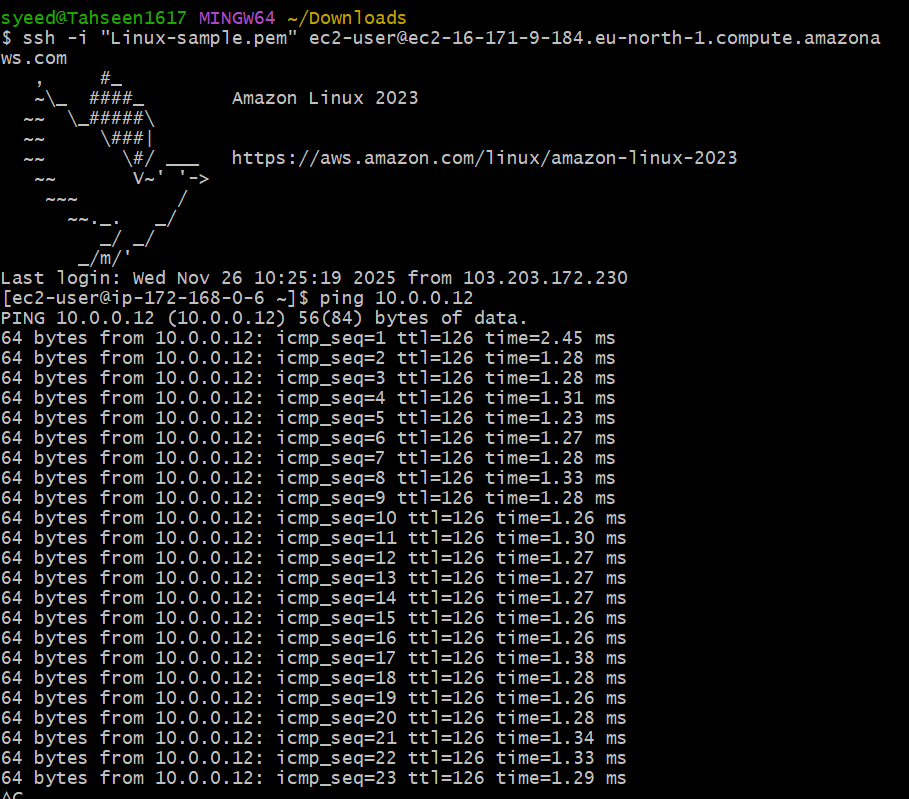
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

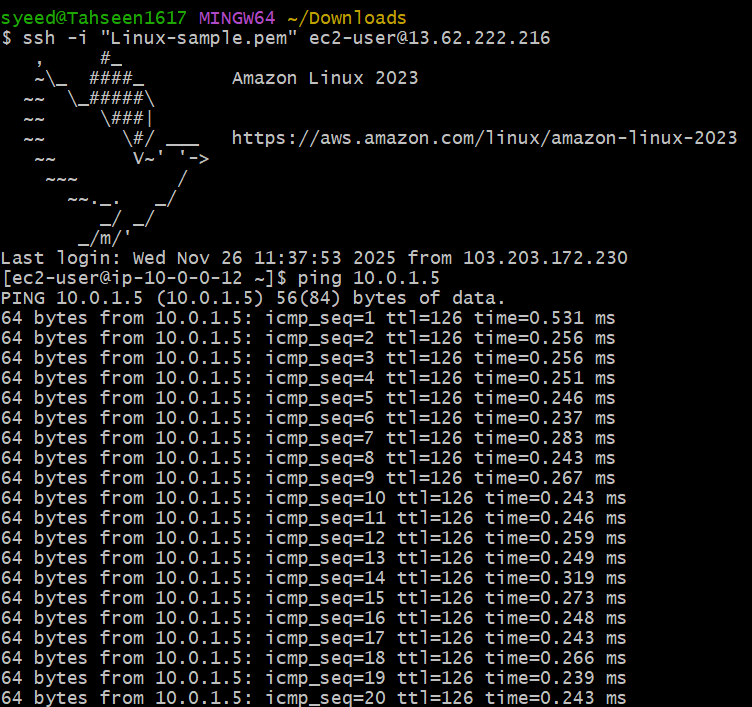
A screenshot of a computer

AI-generated content may be incorrect.



A computer screen shot of a computer program

AI-generated content may be incorrect.



A computer screen shot of a black screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Set up a VPC Endpoint

**Step-1: Create S3 Bucket**

AWS Management Console → **S3 → Create bucket**

Fill in:

| **Field** | **Value** |
| --- | --- |
| Region | Europe (Stockholm) eu-north-1 |
| Bucket type | General purpose |
| Bucket name | my-s3-bucket-mujju |
| Block Public Access | Enabled (default recommended) |

Click **Create bucket**

S3 bucket successfully created

**Step-2: Create AWS Access Key**

IAM → Security credentials → Create access key (root/ IAM user)

You generated:

* **Access Key ID**
* **Secret Access Key**

**download** the .csv file

**Step-3: Configure AWS CLI on EC2 Instance**

SSH into EC2:

ssh -i "linux-sample.pem" ec2-user@<EC2-Public-IP>

Run AWS configure:

aws configure

Enter the credentials:

AWS Access Key ID: <your-access-key>

AWS Secret Access Key: <your-secret-key>

Default region: eu-north-1

Default output format: json

Verify S3 access:

aws s3 ls

**Bucket visible**

**Step-4: Create VPC S3 Endpoint**

VPC Console → **Endpoints → Create endpoint**

Endpoint name: my-endpoint  
 Service category: **AWS services**  
 Search for S3 service:

com.amazonaws.eu-north-1.s3

**Select** **Type: Gateway**

Choose VPC:  
vpc-068d558547ab00f25 (my-vpc-01)

**Select Route Tables:**  
 private-rt  
 public-rt

Policy: **Full access** (default)

Click **Create endpoint**

Endpoint created successfully

**Step-5: Test Private Access to S3 (No Internet Required)**

From private subnet instance or same EC2:

aws s3 ls

Success again

This confirms:

* S3 is reachable **privately**
* EC2 uses **VPC Endpoint**, not the Internet

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with a red arrow pointing to the black screen

AI-generated content may be incorrect.