# AWS Multi-AZ Multi-Region Network using VPC Peering: A 3-Region, 9-Node EC2 Deployment Guide

Satyam Thakur 6 min read · Just now

In today's cloud-native era, building fault-tolerant, distributed infrastructures is more than a best practice, it's a necessity. This article dives into a hands-on walkthrough of deploying a secure and scalable **9-EC2 private network** spanning three **AWS regions** and **Availability Zones (AZs)** using **custom VPCs and VPC Peering**.

Let's get started.

## Objective

Create a private network architecture that simulates a real-world, distributed cloud environment using:

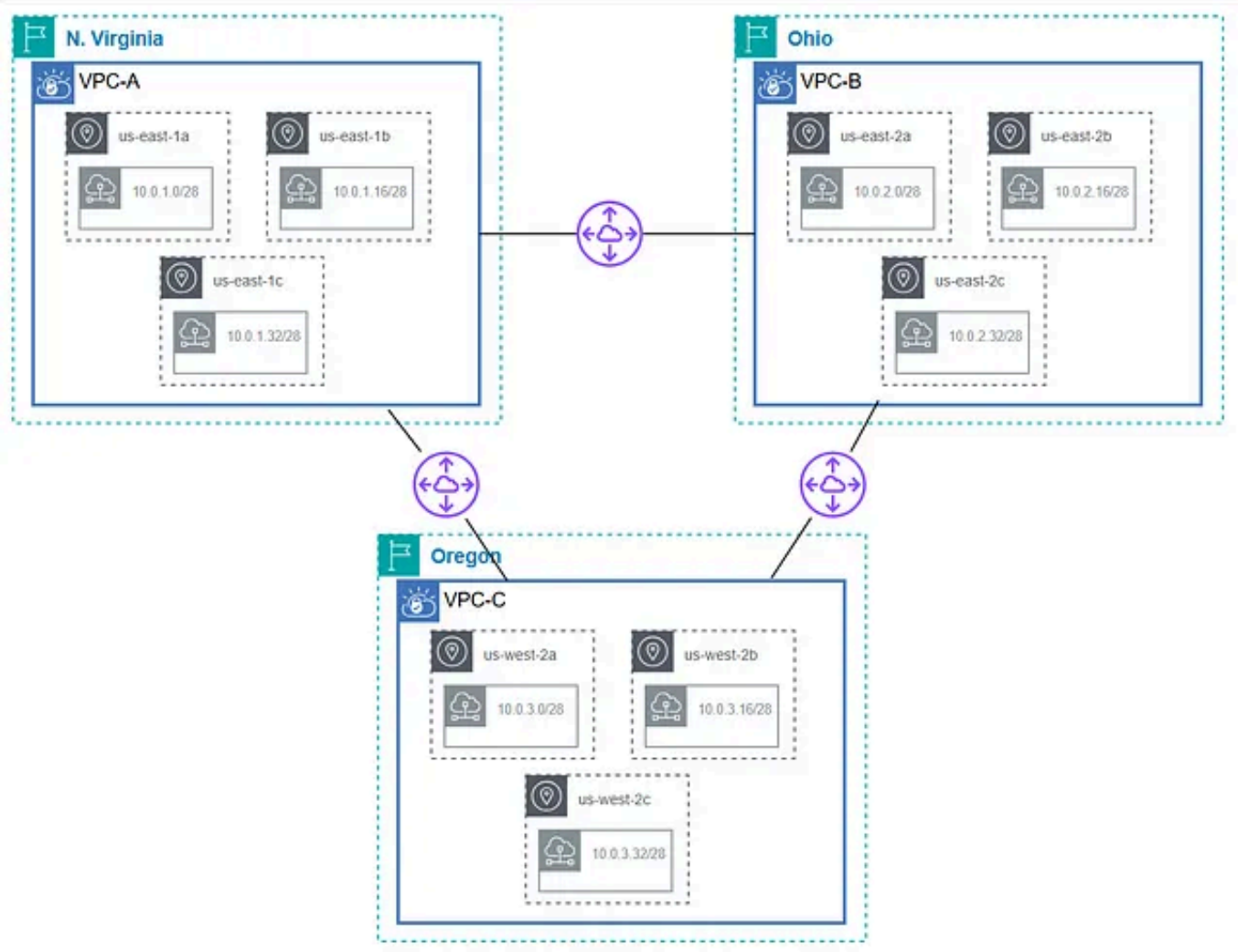
- **3 Regions:** `us-east-1`, `us-east-2`, `us-west-2`
- **9 EC2 Instances:** One in each AZ across three regions
- **3 Custom VPCs:** With defined CIDR ranges and peering across regions

## 🛠️ Step 1: VPC Creation Across Regions

A **VPC (Virtual Private Cloud)** in AWS is a private, isolated network where you can launch and manage resources like EC2 instances securely, controlling IP ranges, subnets, and routing. It's like having your own private data center in the cloud

For each region, create one VPC with three public subnets — one per AZ.

## Network Architecture




Network topology showing 3 VPCs across 3 regions

> *Figure: Network topology showing 3 VPCs across 3 regions*

## Subnet and Instance Mapping:

| VPC | Region | AZ | IPv4 CIDR | Subnet | Instance | Type |
|---|---|---|---|---|---|---|
| VPC-123 | us-east-1 | 1a | 10.0.1.0/24 | 10.0.1.0/28 | Org1 | c7a.xlarge |
| | | 1b | | 10.0.1.16/28 | Org2 | c7a.xlarge |
| | | 1c | | 10.0.1.32/28 | Org3 | c7a.xlarge |
| VPC-456 | us-east-2 | 2a | 10.0.2.0/24 | 10.0.2.0/28 | Org4 | c7a.xlarge |
| | | 2b | | 10.0.2.16/28 | Org5 | c7a.xlarge |
| | | 2c | | 10.0.2.32/28 | Org6 | c7a.xlarge |
| VPC-789 | us-west-2 | 2a | 10.0.3.0/24 | 10.0.3.0/28 | Org7 | c7a.xlarge |
| | | 2b | | 10.0.3.16/28 | Org8 | c7a.xlarge |
| | | 2c | | 10.0.3.32/28 | Org9 | c7a.xlarge |

IPv4 Planning 3 VPCs across 3 regions

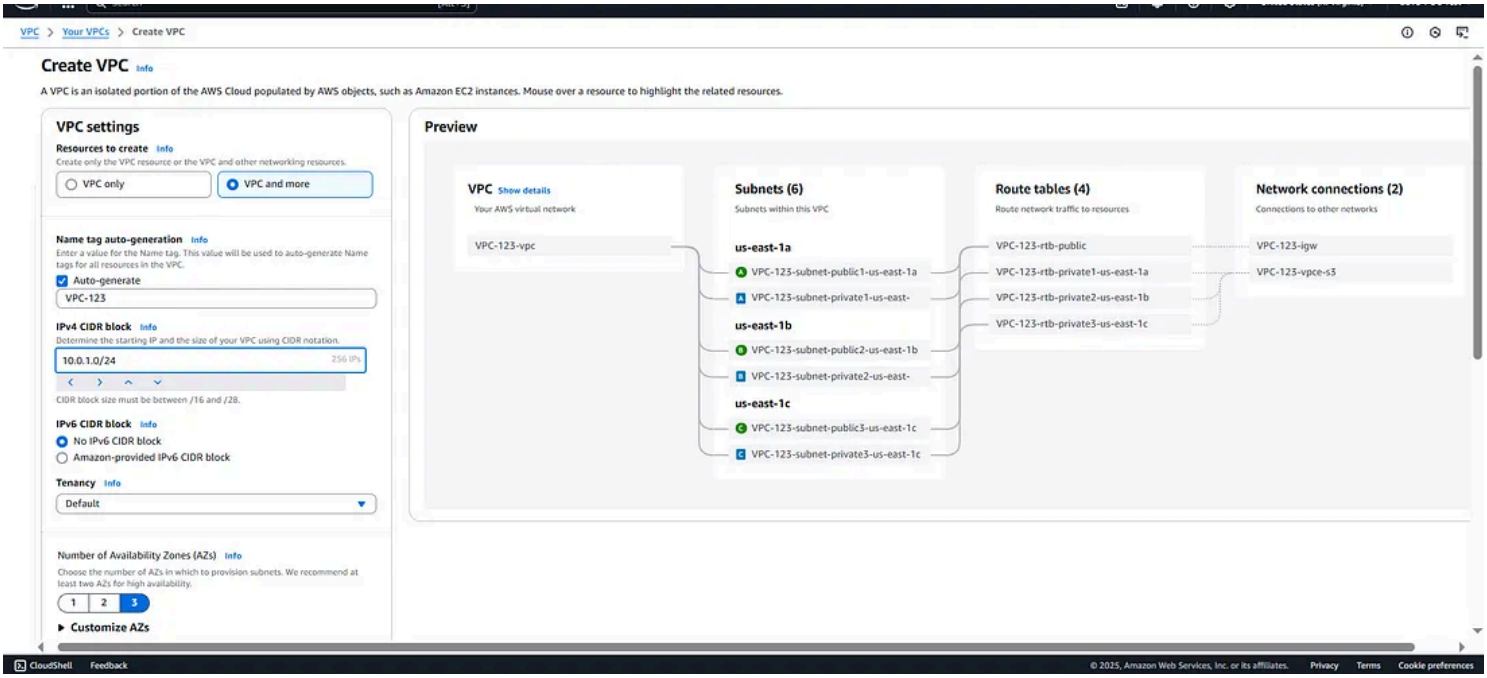
> *Figure: IPv4 Planning 3 VPCs across 3 regions*

## Create VPC-123 (Repeat for VPC-456 and VPC-789)

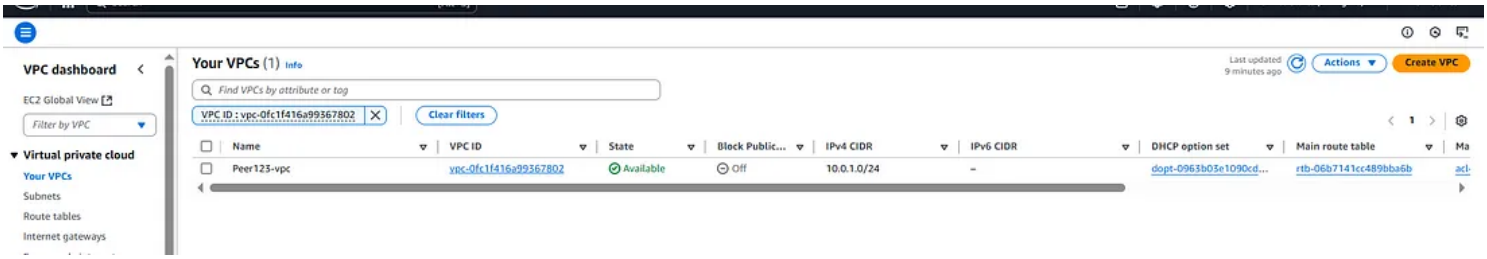
1. Navigate to: **VPC > Your VPCs > Create VPC**
2. Choose “**VPC and more**” for a streamlined setup
3. Configure the following:

- VPC Name: `VPC-123`
- IPv4 CIDR block: `10.0.1.0/24`
- Number of AZs: `3`
- Public Subnets: `3`
- NAT Gateways: `None` (optional if only Private subnets)
- VPC Endpoints: `None` (optional if required S3 Gateway)

> Repeat the same setup for VPC-456 ( `10.0.2.0/24` , `us-east-2` ) and VPC-789 ( `10.0.3.0/24` , `us-west-2` ).




Screenshot of AWS Create VPC page




Screenshot of Your VPC Table

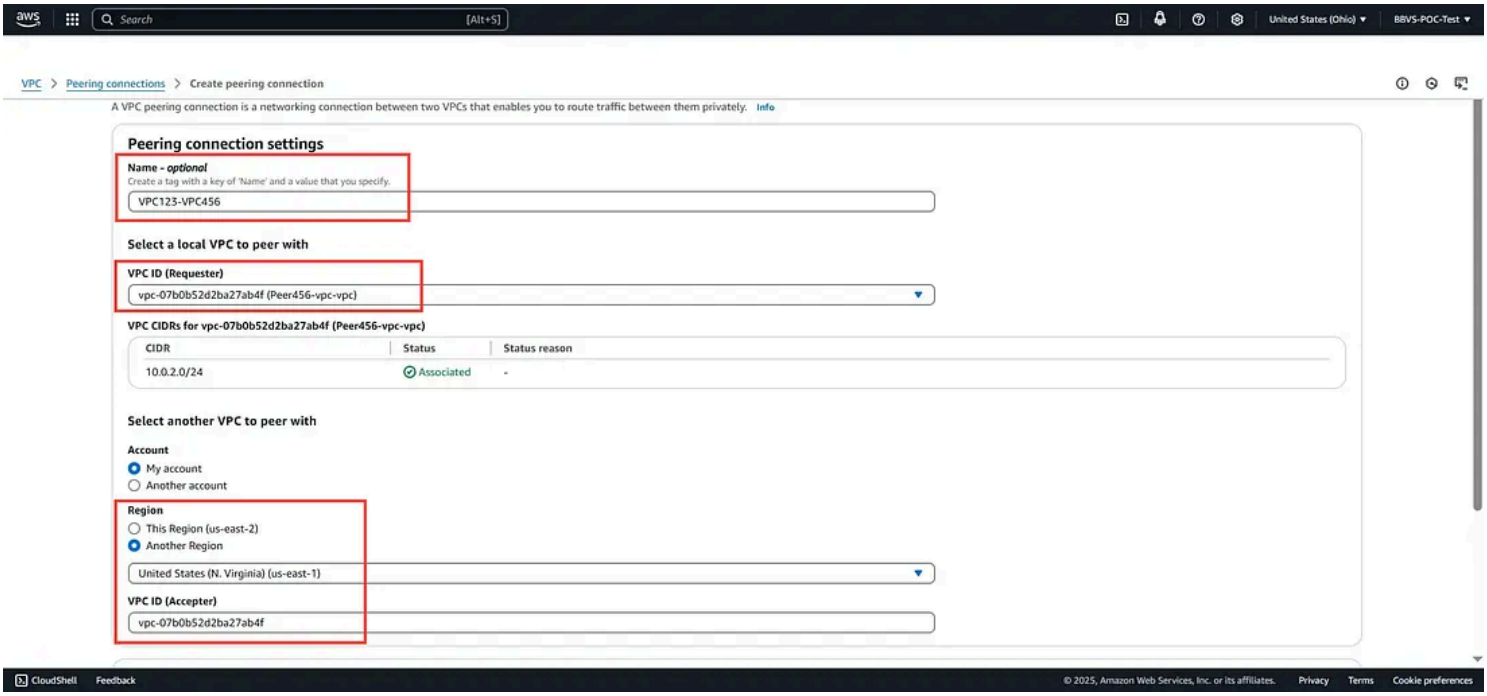
## 🔗 Step 2: Establish VPC Peering Connections

To enable inter-region communication between EC2 instances, create peering connections between all three VPCs.

### How to Create Peering:

Go to: **VPC > Peering connections > Create peering connection**

1. Name: `VPC-123-VPC-456`, `VPC-456-VPC-789`, etc.
2. Select VPC (Requester): e.g., `VPC-123`
3. Region: Another Region (e.g., `us-east-2`)
4. VPC ID (Accepter): e.g., `VPC-456`
5. Accept the connection manually from the target region




Screenshot of AWS VPC Peering page

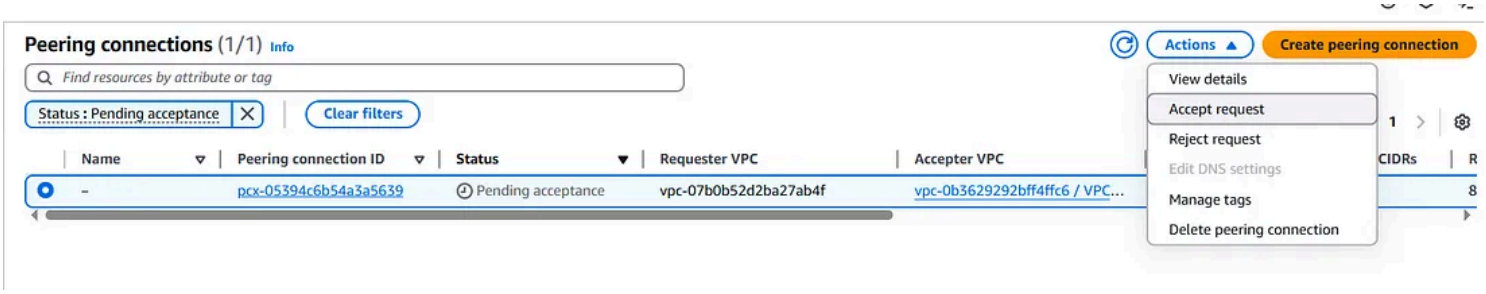
## Confirm the Connection

After creating the peering connection, it must be **acknowledged from the Accepter VPC**:

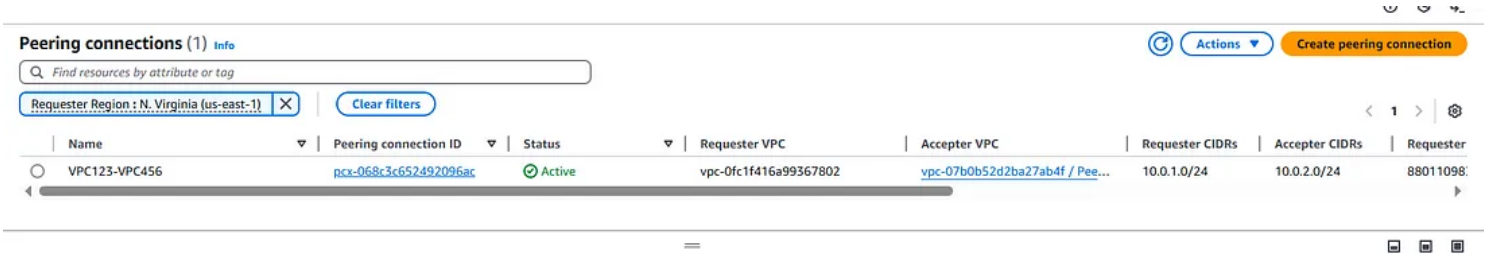
Navigate to: **VPC > Peering connections list**

1. Locate the new connection with status **“Pending Acceptance”**
2. Select it and click **“Accept Request”**

3. Once accepted, the status will change to **“Active”**, confirming that the VPC Peering connection is successfully established.




Screenshot of AWS VPC Peering Acceptance page




Screenshot of AWS VPC Peering Established page

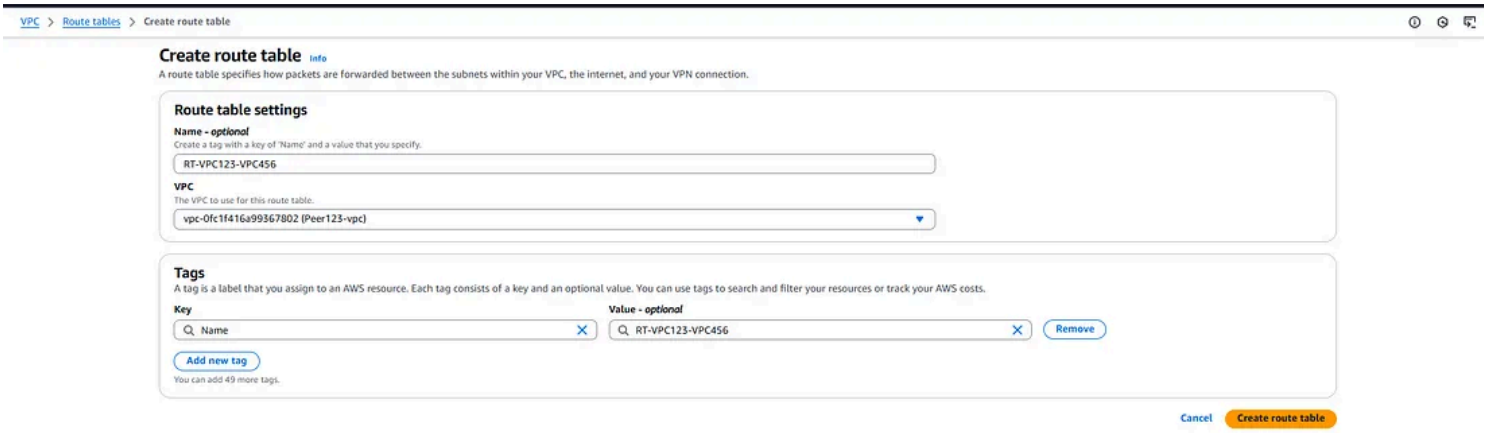
However, establishing a peering connection is not enough — **you must also configure subnet routing** so the VPCs can communicate. Let’s cover that next.

## 📬 Step 3: Configure Routing Tables

Once peering is active (as in last figure), configure routing tables to connect the networks.

### Route Table Setup:

Navigate to **VPC > Route tables > Create route table**

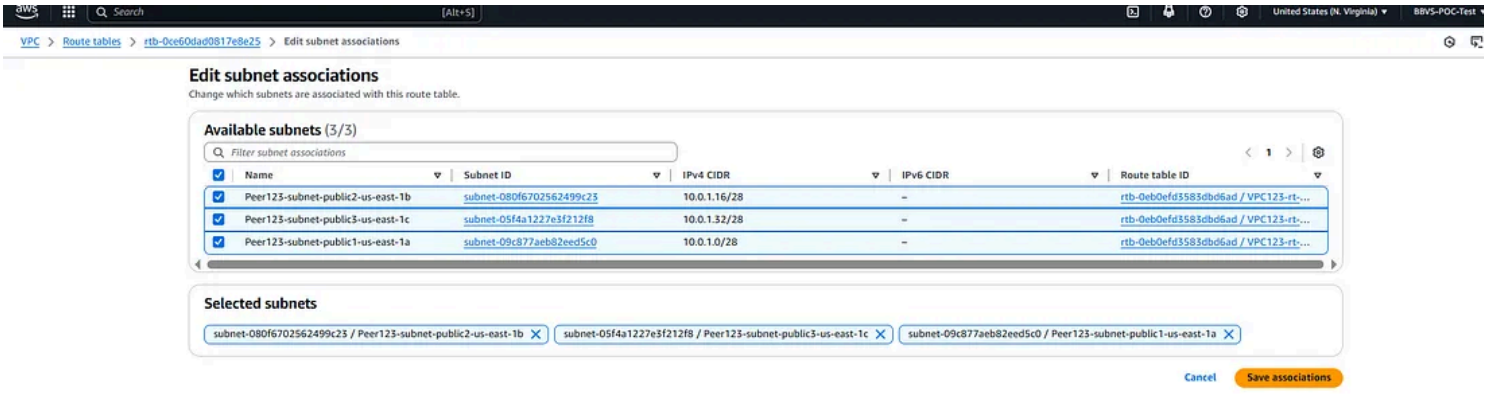



Screenshot of **Create route table**

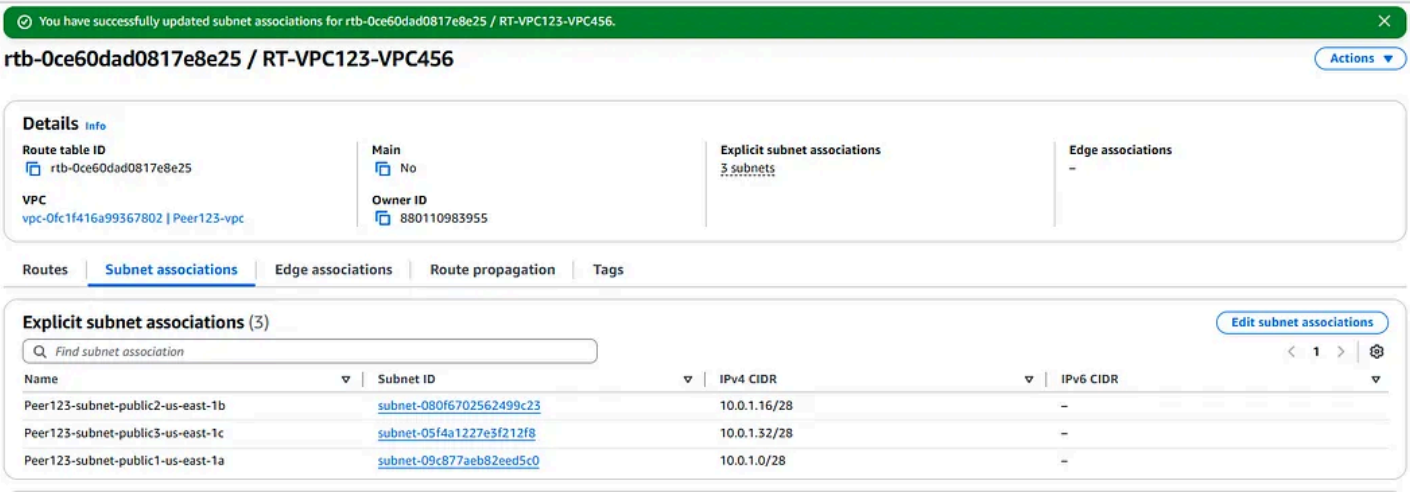
1. Associate the route table with corresponding subnets

Go to: **VPC > Route Table > Corresponding Created Route > Edit Subnet Association**

This will add the subnet you want this routing table to be populated (Note: Each subnet can be associated with only one Route Table at a time)




Screenshot of associating Subnet to Route Table




Screenshot of Subnet Successfully added to RT

2. Add custom routes:

Go to: **VPC > Route Table > Corresponding Created Route > Edit routes**

Example for `VPC-123` (For Establish the routing and communication between two VPC in different region)

- `10.0.1.0/24` → Local
- `10.0.2.0/24` → Peering Connection to `VPC-123`
- `10.0.3.0/24` → Peering Connection to `VPC-456`

| Destination | Target |
|---|---|
| 10.0.1.0/24 | local |
| | local |
| 0.0.0.0/0 {All traffic} | Internet Gateway |
| | igw-{local subnet} |
| 10.0.2.0/24 {Remote Subnet VPC456} | Peering Connection |
| | pcx-{Peering connection ID} |
| 10.0.3.0/24 {Remote Subnet VPC789} | Peering Connection |
| | pcx-{IPeering connection ID} |

Configuration of Routes




Screenshot of route table entries

3. Repeat for other VPCs accordingly. Follow the below table for reference:

| Route Table Configuration VPC456: | |
|---|---|
| **Destination** | **Target** |
| 10.0.2.0/24 | local |
| | local |
| 0.0.0.0/0 {All traffic} | Internet Gateway |
| | igw-{local subnet} |
| 10.0.1.0/24 {Remote Subnet VPC123} | Peering Connection |
| | pcx-{Peering connection ID} |
| 10.0.3.0/24 {Remote Subnet VPC789} | Peering Connection |
| | pcx-{lPeering connection ID} |

| Route Table Configuration VPC789: | |
|---|---|
| **Destination** | **Target** |
| 10.0.3.0/24 #local subnet | local |
| | local |
| 0.0.0.0/0 #All traffic | Internet Gateway |
| | igw-{local subnet} |
| 10.0.1.0/24 #Remote Subnet VPC123 | Peering Connection |
| | pcx-{Peering connection ID} |
| 10.0.2.0/24 #Remote Subnet VPC456 | Peering Connection |
| | pcx-{lPeering connection ID} |

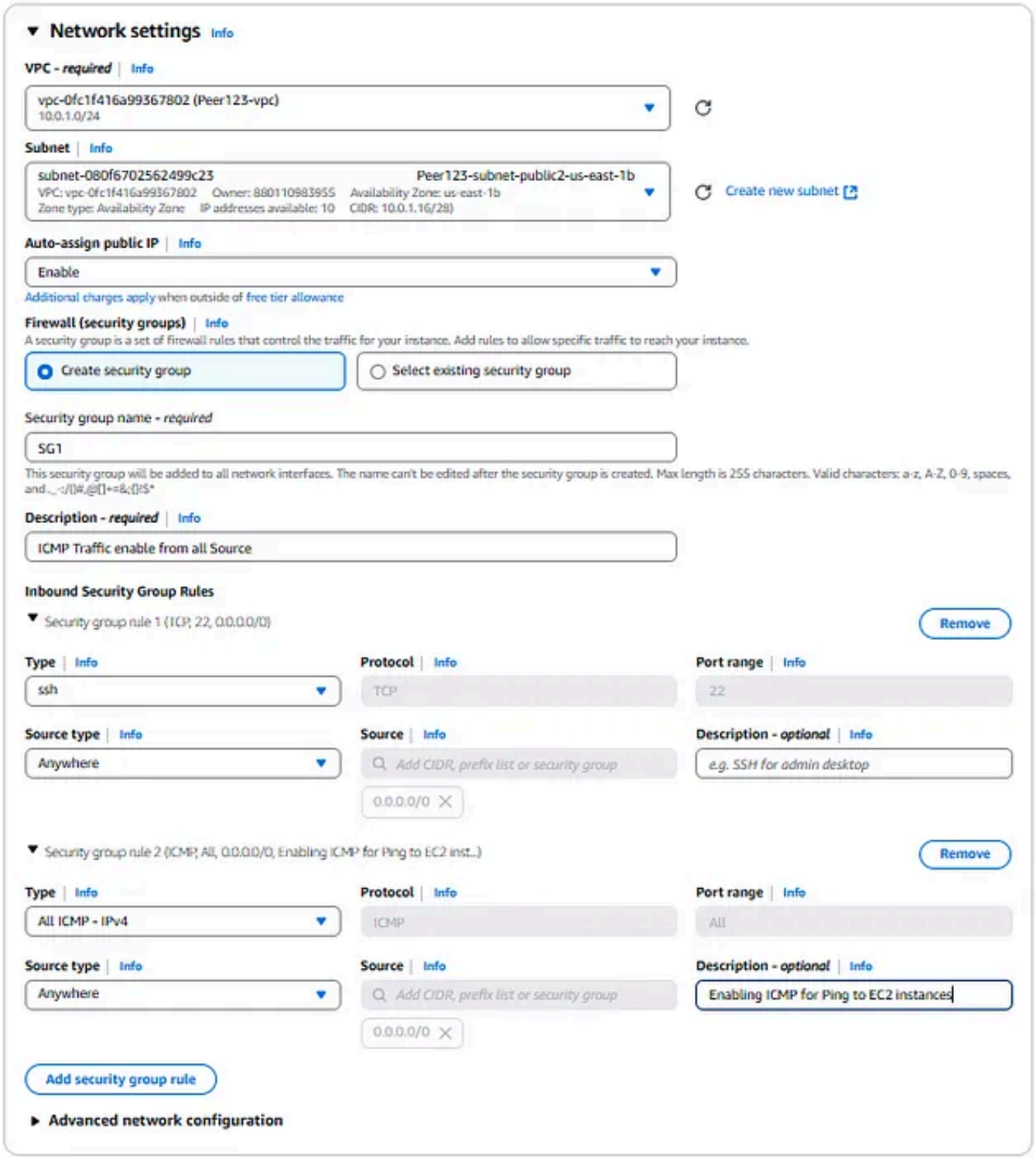
**Note:** The routes configured here are **static routes** and are **non-transitive.** This means that communication is **only allowed between VPCs that are directly peered.** If VPC-123 is peered with VPC-456, and VPC-456 is peered with VPC-789, **VPC-123 cannot automatically communicate with VPC-789** through VPC-456.
For any additional communication paths, you must **explicitly create separate VPC peering connections** and **manually update the route tables** to establish direct connectivity between each VPC pair.

## 📦 Step 4: Launch EC2 Instances

### Launch Process:

1. Go to **EC2 > Launch Instance**
2. Choose AMI: Amazon Linux 2 or Ubuntu 22.04
3. Choose instance type: `c7a.xlarge` (or t2.micro)
4. Network Settings:

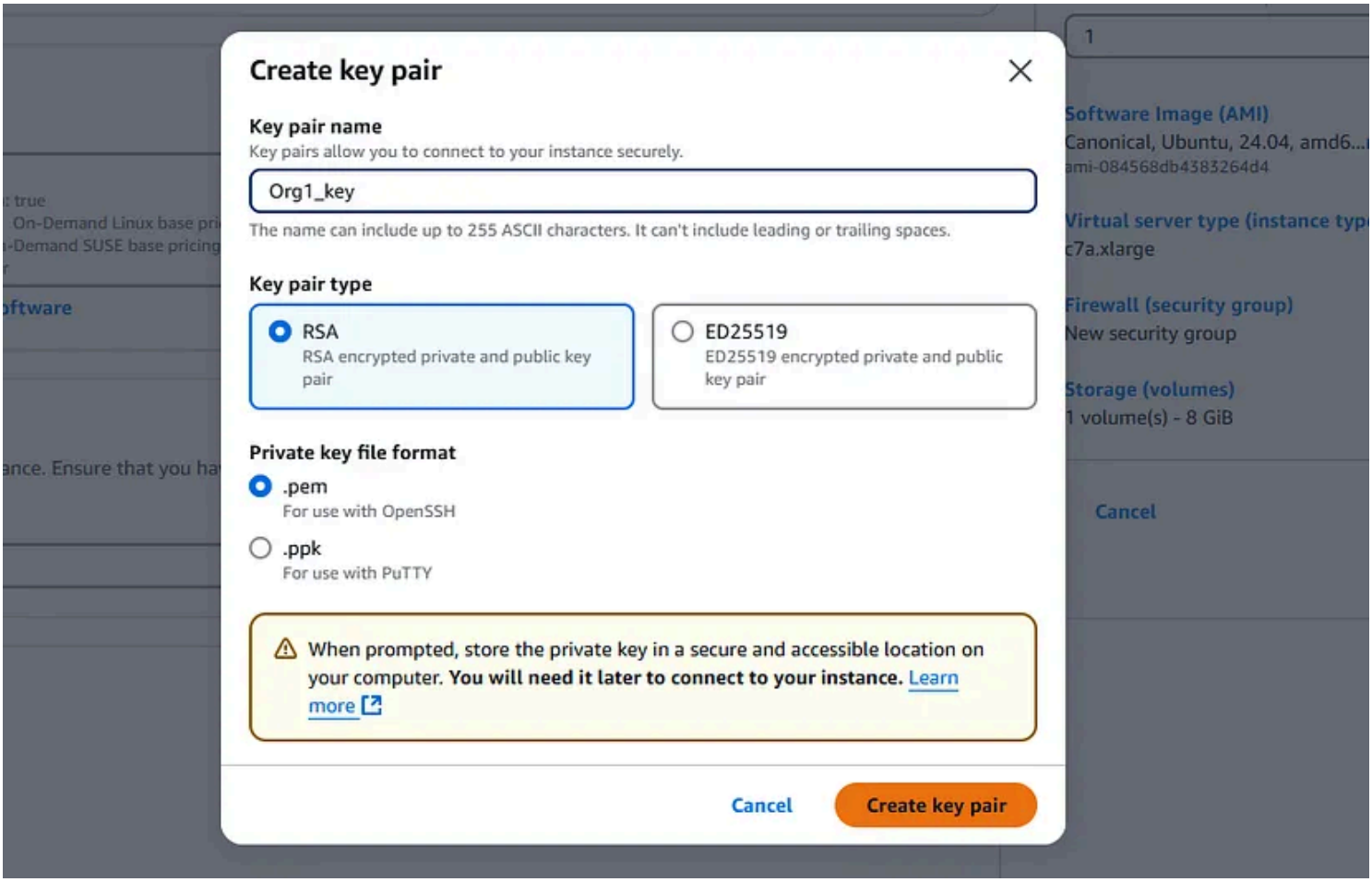
- Select the corresponding VPC and subnet as defined in your architecture plan (refer to the Subnet and Instance Mapping table).




Screenshot showing VPC selection, subnet mapping per AZ, and security group configurations during EC2 setup.

1. Key Pair:

- Either create a new SSH key pair or use an existing one.
- Download and securely store the `.pem` file.




Screenshot of Key Pair create

1. Launch 9 EC2 instances in total:

- 3 in each VPC, each placed in a different subnet corresponding to an Availability Zone.

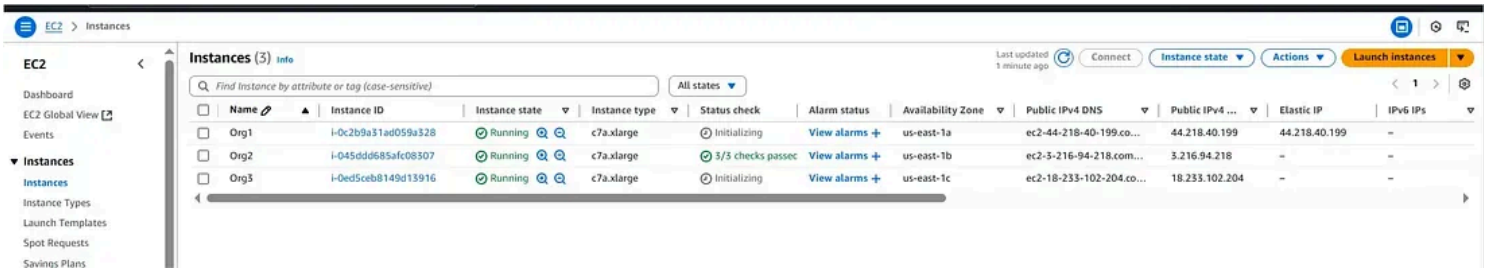



Table view from AWS Console summarizing launched instances with AMI, instance type, network/subnet ID, and private IP addresses.

## 🔍 Step 5: Test Network Reachability

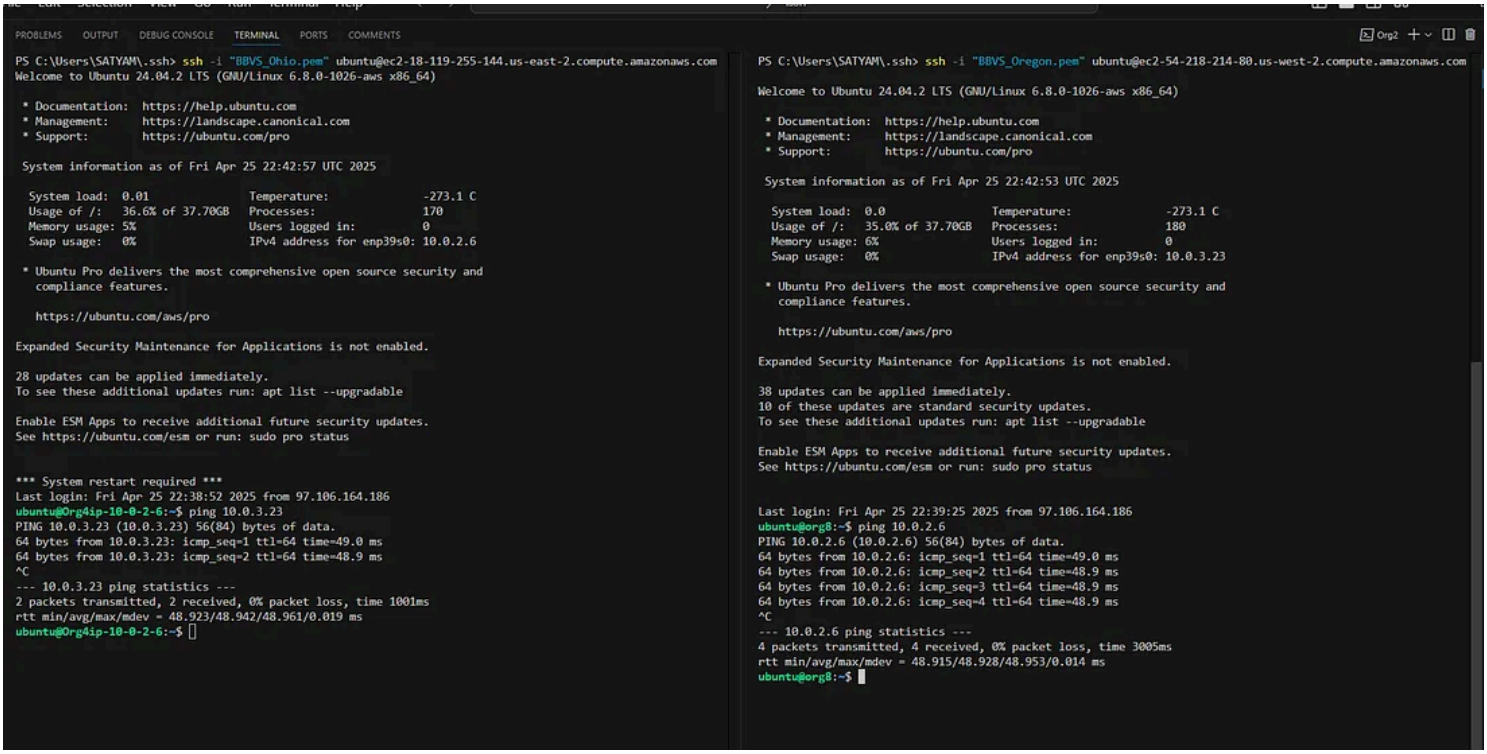
SSH into each EC2 instance using its Public IP and test connectivity:

```
ping 10.0.2.4   # from Org1 to Org4
ping 10.0.3.4   # from Org1 to Org7
```

Use security groups to allow ICMP and SSH between subnets.

## Key Pair Creation (CLI or Console):

- Key Name: `Org1-key`
- File Format: `.pem`
- Type: RSA or ED25519




> ***Screenshot** of Terminal showing ping/ssh tests*

You've now successfully:

✅ Created **3 VPCs** across **3 AWS regions**, each spanning **3 Availability Zones.**

✅ Deployed **9 EC2 instances** (one in each AZ).

✅ Established the **networks** and **tested connectivity** between instances.

🚀 This setup mirrors real-world architectures for **highly available** and **distributed systems**, ideal for applications such as **blockchain networks**, **enterprise systems**, and **global microservices.**

🌟 **Stay tuned for more updates on deploying and orchestrating distributed applications across this multi-region, multi-AZ environment!**

Aws Networking | Distributed Network | Aws Vpc Peering | Multi Az | Multi Region

## Written by Satyam Thakur

0 Followers · 5 Following

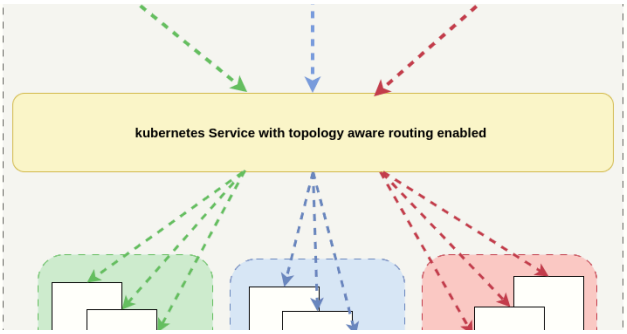
Edit profile

## No responses yet

Satyam Thakur he/him

What are your thoughts?

## Recommended from Medium




Sai Kiran Pikili

## Reduce Data Transfer Costs with Topology Aware Routing

Enabling Topology-Aware Routing in Kubernetes

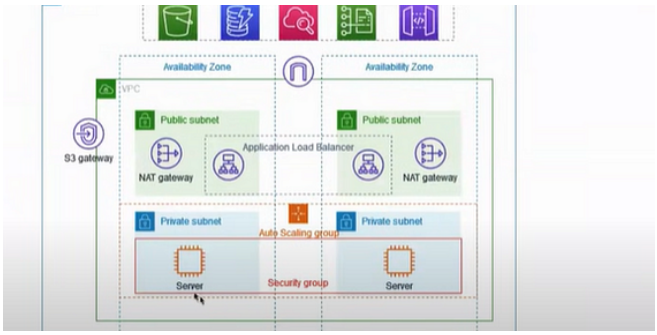
Apr 16 1


Devops Diaries

## Kubernetes Reinvented: What's Next with AI in 2025?

Artificial intelligence can learn from your habits and predict what you need next, just...

6d ago




Tarunvenkatsaikumar

## AWS VPC Project Implementation Day:7.

Hi! This is Tarun, Welcome to Day7 of AWS series in this session we will be demonstrati...

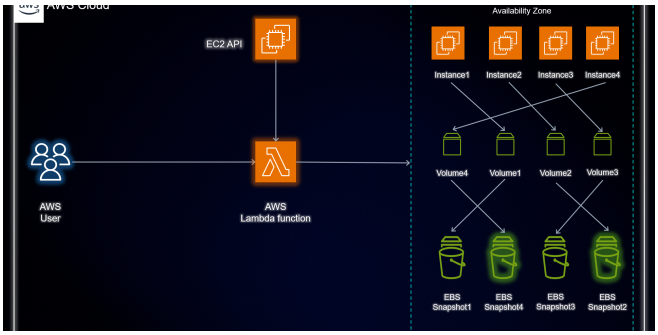
Mar 25

In DevelopersGlobal by Dhanush N

## Linux Networking Made Easy

Read the full article for free via the link in the comments—all resources are included at th...

6d ago 121 1




Abdihakim said

## How I Saved $12,000 in One Month by Automating EBS Snapshot...

Client Story: Tackling Rising AWS Costs

Apr 16


Everton Araújo

## Best Practices for Terraform Infrastructure

In this document, we will explore the best practices for managing infrastructure as cod...

Apr 14 1

See more recommendations