

◆ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



# 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:

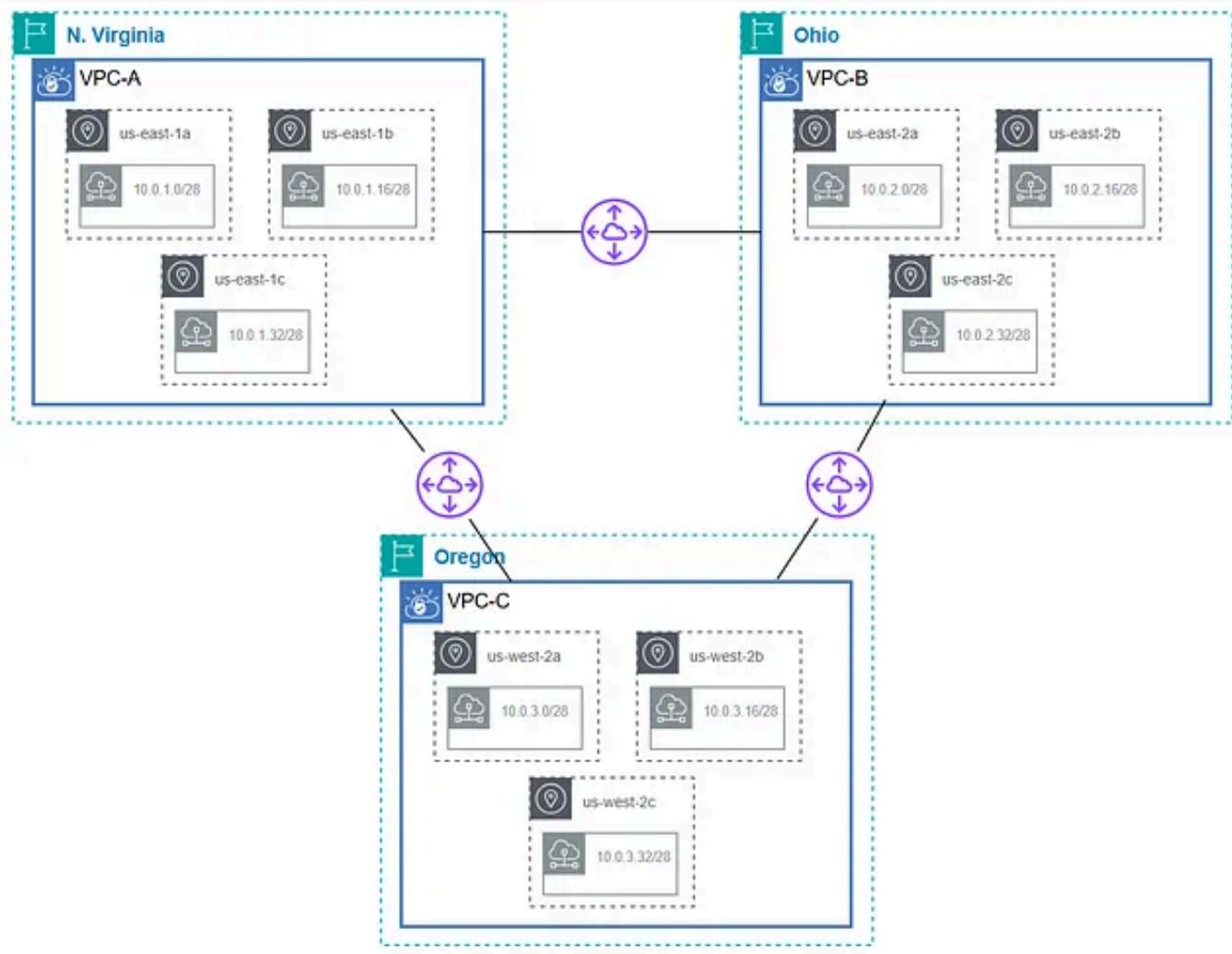
- **3 Regions:** us-east-1, us-east-2, us-west-2
- **9 EC2 Instances:** One in each AZ across three regions
- **3 Custom VPCs:** With defined CIDR ranges and peering across regions

## 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 ).

The screenshot shows the 'Create VPC' page in the AWS Management Console. On the left, the 'VPC settings' section includes fields for 'Name tag auto-generation' (set to 'Auto-generate' with 'VPC-123'), 'IPv4 CIDR block' (set to '10.0.1.0/24'), and 'Tenancy' (set to 'Default'). Below these are options for 'Number of Availability Zones (AZs)' (set to 3). On the right, the 'Preview' section shows the VPC structure: a single VPC named 'VPC-123-vpc' containing 6 subnets across three AZs ('us-east-1a', 'us-east-1b', 'us-east-1c'), 4 route tables, and 2 network connections. The 'Route tables' section lists four route tables: 'VPC-123-rtb-public', 'VPC-123-rtb-private1-us-east-1a', 'VPC-123-rtb-private2-us-east-1b', and 'VPC-123-rtb-private3-us-east-1c'. The 'Network connections' section shows a connection to 'VPC-123-igw' via 'VPC-123-vpc-e-s3'.

Screenshot of AWS Create VPC page

The screenshot shows the 'Your VPCs' table in the AWS Management Console. The table lists one VPC entry: 'Peer123-vpc' with VPC ID 'vpc-0fc1f416a99367802'. The table includes columns for Name, VPC ID, State, Block Public..., IPv4 CIDR, IPv6 CIDR, DHCP option set, Main route table, and Actions. The 'Actions' column shows a 'Create peering connection' button for this specific VPC.

Screenshot of Your VPC Table

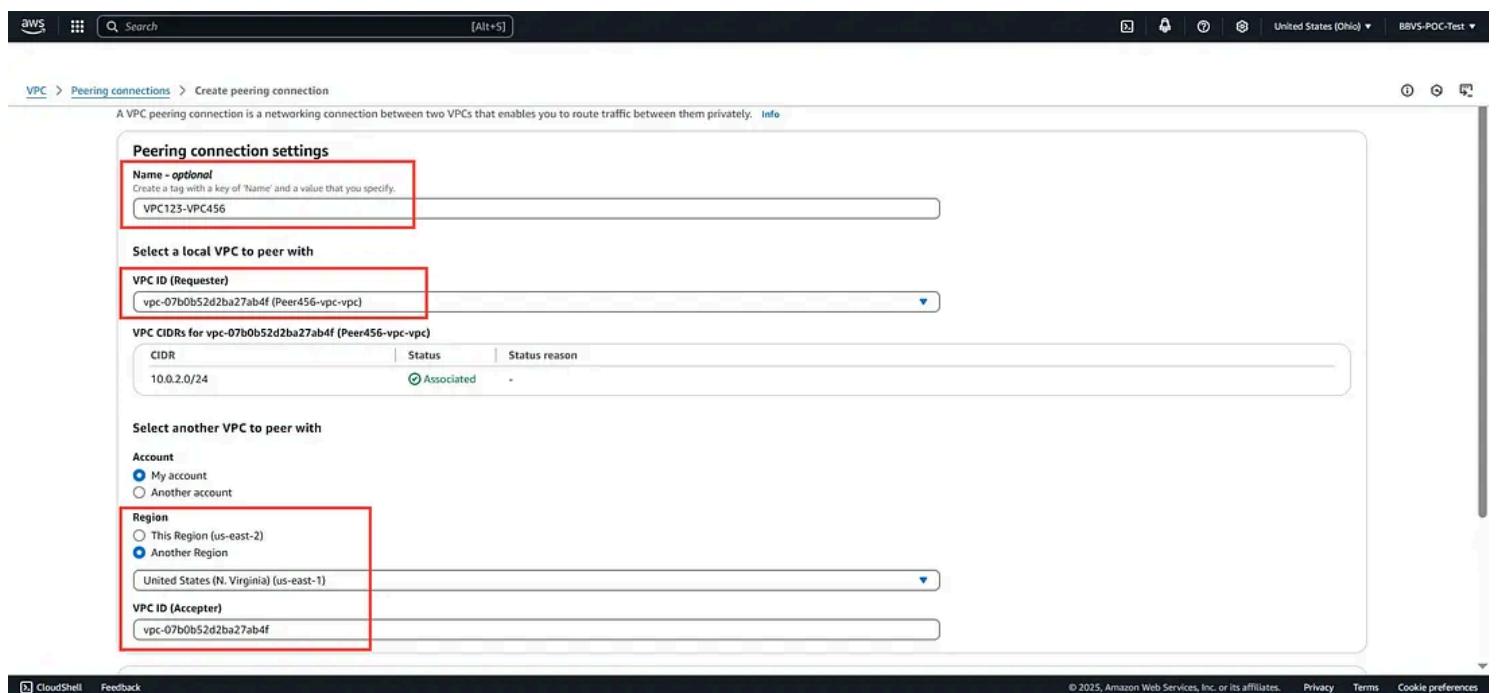
## ⌚ Step 2: Establish VPC Peering Connections

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 (Acceptor): 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 Acceptor VPC:**

Navigate to: VPC > Peering connections list

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

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

The screenshot shows the AWS VPC Peering connections page with one entry:

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC
-	pcx-05394c6b54a3a5639	Pending acceptance	vpc-07b0b52d2ba27ab4f	vpc-0b3629292bff4ffc6 / VPC...

Actions dropdown menu options include: View details, Accept request (highlighted), Reject request, Edit DNS settings, Manage tags, and Delete peering connection.

Screenshot of AWS VPC Peering Acceptance page

The screenshot shows the AWS VPC Peering connections page with one entry:

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDRs	Acceptor CIDRs	Requester
VPC123-VPC456	pcx-068c3c652492096ac	Active	vpc-0fc1f416a99367802	vpc-07b0b52d2ba27ab4f / Pee...	10.0.1.0/24	10.0.2.0/24	88011098...

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

The screenshot shows the 'Create route table' wizard. In the 'Route table settings' section, a name 'RT-VPC123-VPC456' is entered. In the 'VPC' section, a VPC ID 'vpc-0fc1f416a99367802 (Peer123-vpc)' is selected. Under 'Tags', a single tag 'Name: RT-VPC123-VPC456' is added. At the bottom right are 'Cancel' and 'Create route table' buttons.

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)

The screenshot shows the 'Edit subnet associations' page. It lists three subnets under 'Available subnets': 'Peer123-subnet-public2-us-east-1b', 'Peer123-subnet-public3-us-east-1c', and 'Peer123-subnet-public1-us-east-1a'. These subnets are checked and associated with the route table 'rtb-0eb0efdf3583dbd6ad / VPC123-rt...'. Under 'Selected subnets', the same three subnet IDs are listed. At the bottom right are 'Cancel' and 'Save associations' buttons.

Screenshot of associating Subnet to Route Table

The screenshot shows the AWS Route Table configuration for 'rtb-0ce60dad0817e8e25 / RT-VPC123-VPC456'. In the 'Details' tab, it lists the Route table ID (rtb-0ce60dad0817e8e25), Main status (No), Owner ID (880110983955), and VPC (vpc-0fc1f416a99367800). Under the 'Subnet associations' tab, there are three explicit subnet associations: Peer123-subnet-public2-us-east-1b, Peer123-subnet-public3-us-east-1c, and Peer123-subnet-public1-us-east-1a, each with its Subnet ID, IPv4 CIDR, and IPv6 CIDR.

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-{Peering connection ID}

## Configuration of Routes

[Open in app ↗](#)**Medium**

Search



Write



The screenshot shows the AWS Lambda function configuration interface for route table entries. It displays three routes:

- Route 1: Destination 0.0.0.0/0, Target Internet Gateway (igw-03eebd7425eb277e2).
- Route 2: Destination 10.0.2.0/24, Target Peering Connection (pcx-068c3e652492096ac).
- Route 3: Destination 10.0.3.0/24, Target Peering Connection (pcx-04bb64cdbe9429a76).

Buttons at the bottom include 'Add route', 'Cancel', 'Preview', and 'Save changes'.

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-{Peering 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-{IPeering 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).

**▼ Network settings [Info](#)**

**VPC - required [Info](#)**

vpc-0fc1f416a99367802 (Peer123-vpc)  
10.0.1.0/24

**Subnet [Info](#)**

subnet-080f6702562499c23 Peer123-subnet-public2-us-east-1b  
VPC: vpc-0fc1f416a99367802 Owner: 880110983955 Availability Zone: us-east-1b  
Zone type: Availability Zone IP addresses available: 10 CIDR: 10.0.1.16/28

**Create new subnet [\[+\]](#)**

**Auto-assign public IP [Info](#)**

Enable

Additional charges apply when outside of free tier allowance

**Firewall (security groups) [Info](#)**

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group    Select existing security group

**Security group name - required**

SG1

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and \_.-:/[@]=&`{|}!\$^

**Description - required [Info](#)**

ICMP Traffic enable from all Source

**Inbound Security Group Rules**

**▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)**

**Type [Info](#)**: ssh   **Protocol [Info](#)**: TCP   **Port range [Info](#)**: 22

**Source type [Info](#)**: Anywhere   **Source [Info](#)**: 0.0.0.0/0   **Description - optional [Info](#)**: e.g. SSH for admin desktop

**Remove**

**▼ Security group rule 2 (ICMP, All, 0.0.0.0/0, Enabling ICMP for Ping to EC2 instances.)**

**Type [Info](#)**: All ICMP - IPv4   **Protocol [Info](#)**: ICMP   **Port range [Info](#)**: All

**Source type [Info](#)**: Anywhere   **Source [Info](#)**: 0.0.0.0/0   **Description - optional [Info](#)**: Enabling ICMP for Ping to EC2 instances

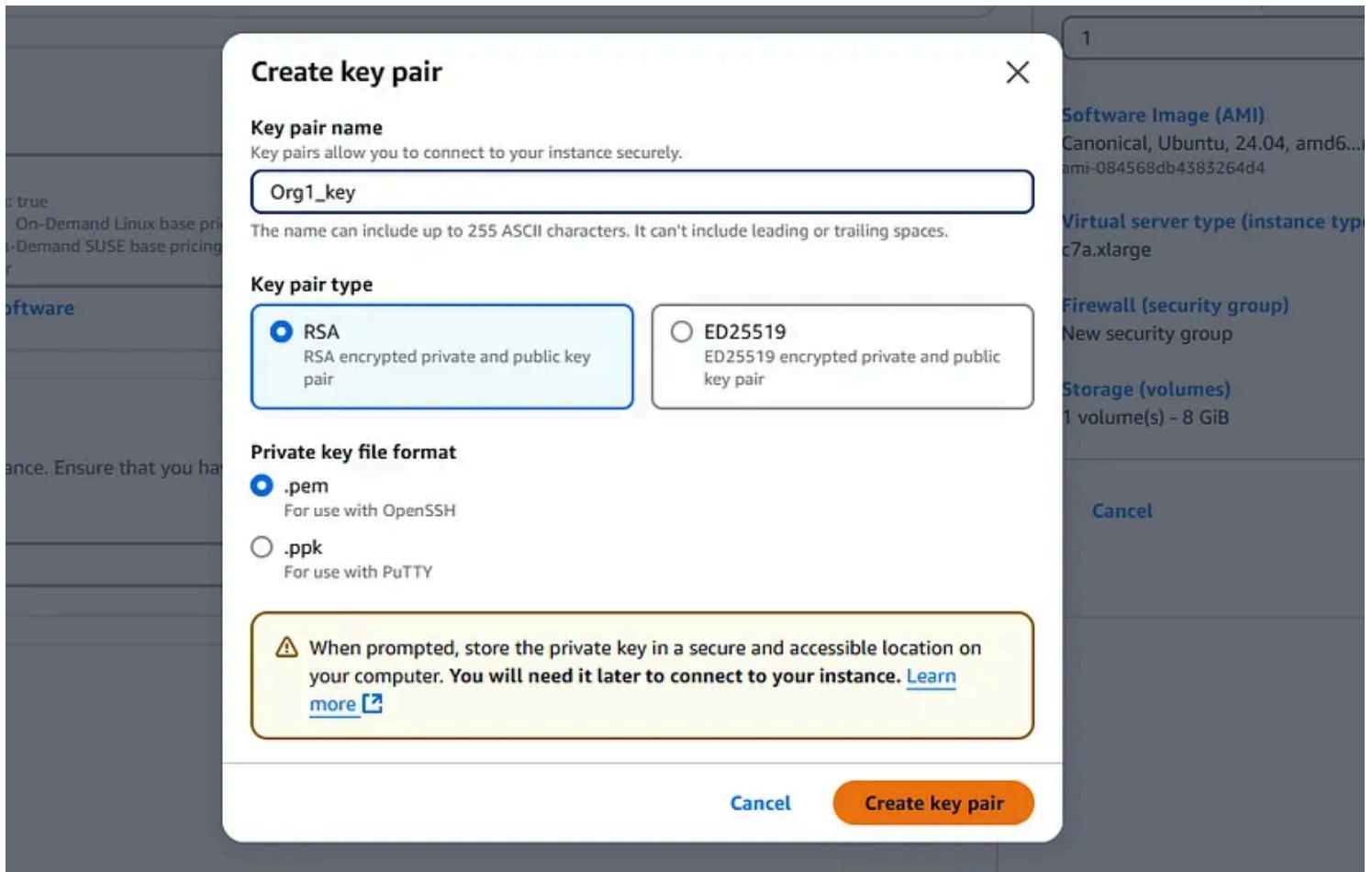
**Add security group rule**

**► Advanced network configuration**

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.

Instances (3) <small>Info</small>											
Find Instance by attribute or tag (case-sensitive) <small>All states</small>											
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	
Org1	i-0c2b9a31ad059a328	Running	c7a.xlarge	Initializing	<a href="#">View alarms +</a>	us-east-1a	ec2-44-218-40-199.co...	44.218.40.199	44.218.40.199	-	
Org2	i-045ddd685afc08507	Running	c7a.xlarge	3/3 checks passed	<a href="#">View alarms +</a>	us-east-1b	ec2-3-216-94-218.com...	3.216.94.218	-	-	
Org3	i-0ed5ceb8149d13916	Running	c7a.xlarge	Initializing	<a href="#">View alarms +</a>	us-east-1c	ec2-18-233-102-204.co...	18.233.102.204	-	-	

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

The screenshot shows two terminal windows side-by-side. Both windows are titled 'Terminal' and have tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and COMMENTS. The left window is connected to a host at 'ubuntu@ec2-18-119-255-144.us-east-2.compute.amazonaws.com' via SSH with key 'BBV5\_Ohio.pem'. The right window is connected to a host at 'ubuntu@ec2-54-218-214-80.us-west-2.compute.amazonaws.com' via SSH with key 'BBV5\_Oregon.pem'. Both terminals show system information, security status, and a history of network tests including pings and ssh connections between the three regions.

```

PS C:\Users\SATYAM\.ssh> ssh -i "BBV5_Ohio.pem" ubuntu@ec2-18-119-255-144.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1026-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Fri Apr 25 22:42:57 UTC 2025

System load: 0.01 Temperature: -273.1 C
Usage of /: 36.6% of 37.70GB Processes: 170
Memory usage: 5% Users logged in: 0
Swap usage: 0% IPv4 address for enp39s0: 10.0.2.6

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

28 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Fri Apr 25 22:38:52 2025 from 97.106.164.186
ubuntu@org4ip-10-0-2-6:~$ ping 10.0.3.23
PING 10.0.3.23 (10.0.3.23) 56(84) bytes of data.
64 bytes from 10.0.3.23: icmp_seq=1 ttl=64 time=49.0 ms
64 bytes from 10.0.3.23: icmp_seq=2 ttl=64 time=48.9 ms
^C
--- 10.0.3.23 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 48.923/48.942/48.961/0.019 ms
ubuntu@Org4ip-10-0-2-6:~$ []

PS C:\Users\SATYAM\.ssh> ssh -i "BBV5_Oregon.pem" ubuntu@ec2-54-218-214-80.us-west-2.compute.amazonaws.com
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1026-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Fri Apr 25 22:42:53 UTC 2025

System load: 0.0 Temperature: -273.1 C
Usage of /: 35.0% of 37.70GB Processes: 180
Memory usage: 6% Users logged in: 0
Swap usage: 0% IPv4 address for enp39s0: 10.0.3.23

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

38 updates can be applied immediately.
10 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Fri Apr 25 22:39:25 2025 from 97.106.164.186
ubuntu@org8:~$ ping 10.0.2.6
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
64 bytes from 10.0.2.6: icmp_seq=1 ttl=64 time=49.0 ms
64 bytes from 10.0.2.6: icmp_seq=2 ttl=64 time=48.9 ms
64 bytes from 10.0.2.6: icmp_seq=3 ttl=64 time=48.9 ms
64 bytes from 10.0.2.6: icmp_seq=4 ttl=64 time=48.9 ms
^C
--- 10.0.2.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 48.915/48.928/48.953/0.014 ms
ubuntu@org8:~$ []

```

## 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](#)

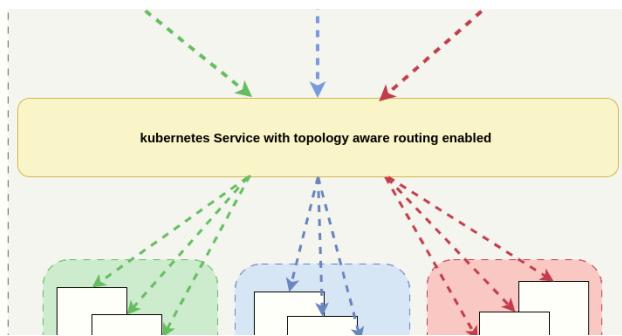
## No responses yet

[...](#)

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

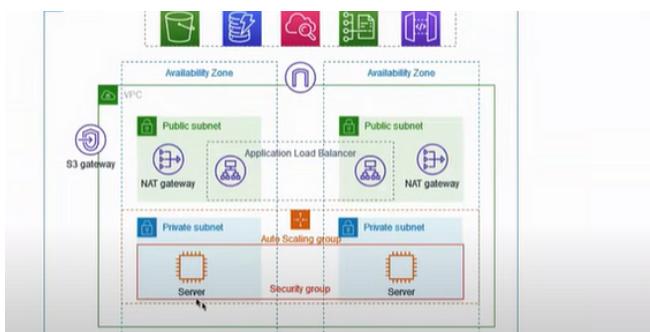


Devops Diaries

## 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

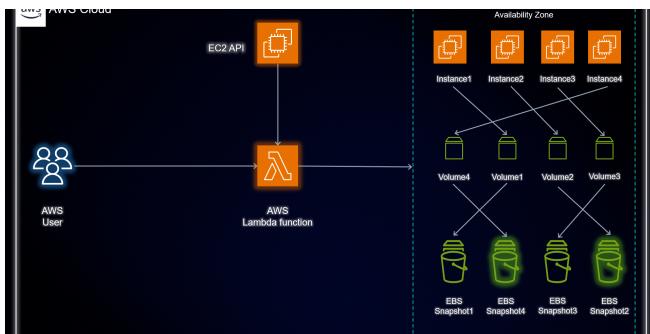
Read the full article for free via the link in the comments—all resources are included at th...

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](#)