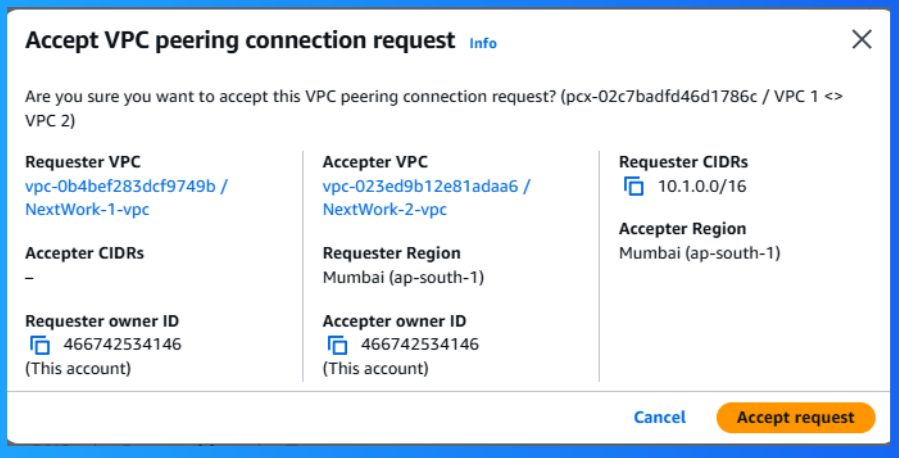
NextWork.org

VPC Peering

DI dineshrajdhanapathy@gmail.com



DI
dineshrajdhanapathy@g...
NextWork Student
NextWork.org

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is a service that allows you to create isolated networks within AWS. It’s useful because it provides full control over network configurations, security, and resource placement, ensuring secure and scalable architectures.

How I used Amazon VPC in this project

In today's project, I used Amazon VPC to create isolated network environments for two EC2 instances. I configured VPC peering, security groups, and route tables to enable secure communication between the instances.

One thing I didn't expect in this project was...

One thing I didn't expect in this project was the troubleshooting required for security group settings and connectivity issues. It took extra steps to ensure the instances could communicate across the VPC peering connection.

This project took me...

This project took me approximately 2-3 hours to complete. It included setting up VPCs, establishing peering connections, launching EC2 instances, configuring security groups, and troubleshooting connectivity issues.

In the first part of my project...

Step 1 - Set up my VPC

In this step, we're setting up VPC Peering to connect two separate VPCs, allowing resources in both VPCs to communicate with each other securely, expanding our network architecture.

Step 2 - Create a Peering Connection

In this step, we're establishing a peering connection between our VPCs to enable secure and seamless communication. This allows resources in both VPCs to interact as if on the same network.

Step 3 - Update Route Tables

In this step, we're configuring route tables for each VPC to direct traffic through the peering connection. This ensures traffic from VPC 1 reaches VPC 2 and vice versa, enabling smooth communication.

Step 4 - Launch EC2 Instances

In this step, we are launching an EC2 instance in each VPC to serve as test instances. These will help verify that the VPC peering connection is functioning correctly by allowing communication between them.

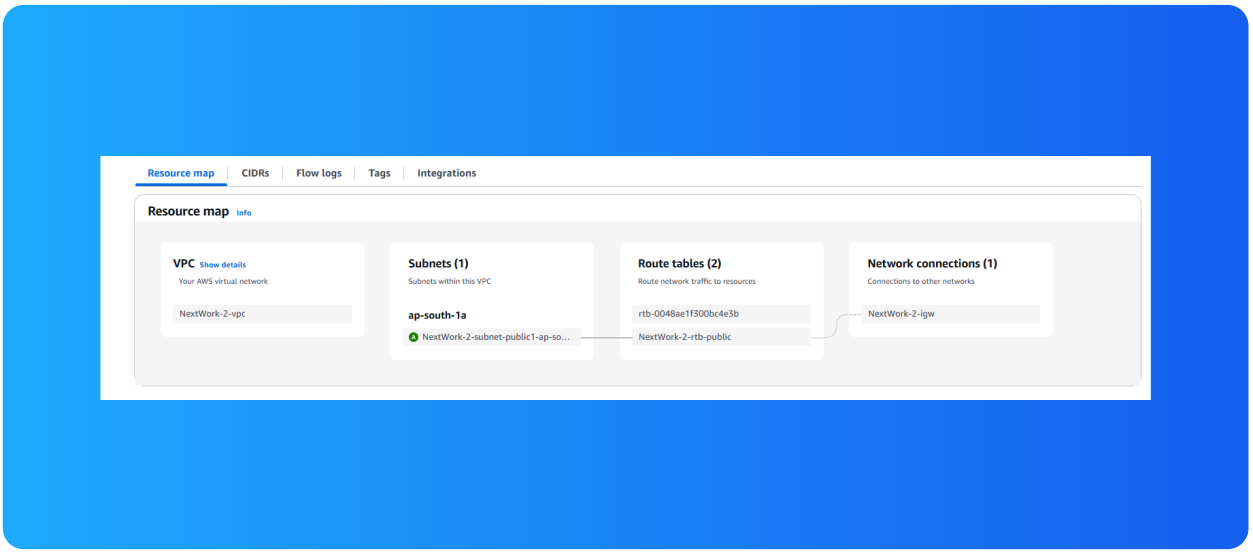
Multi-VPC Architecture

I started my project by launching a two VPC in AWS to set up a secure network environment. Within this VPC, I created subnets: one public subnet for external access and 0 private subnets for internal resources.

The CIDR blocks for VPCs 1 and 2 are unique, like 10.1.0.0/16 and 10.2.0.0/16. They have to be unique because overlapping IP ranges would cause routing conflicts, preventing proper communication between resources in each VPC.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances as they are only used for testing purposes. I plan to manage access through other means, such as security groups or other temporary credentials.

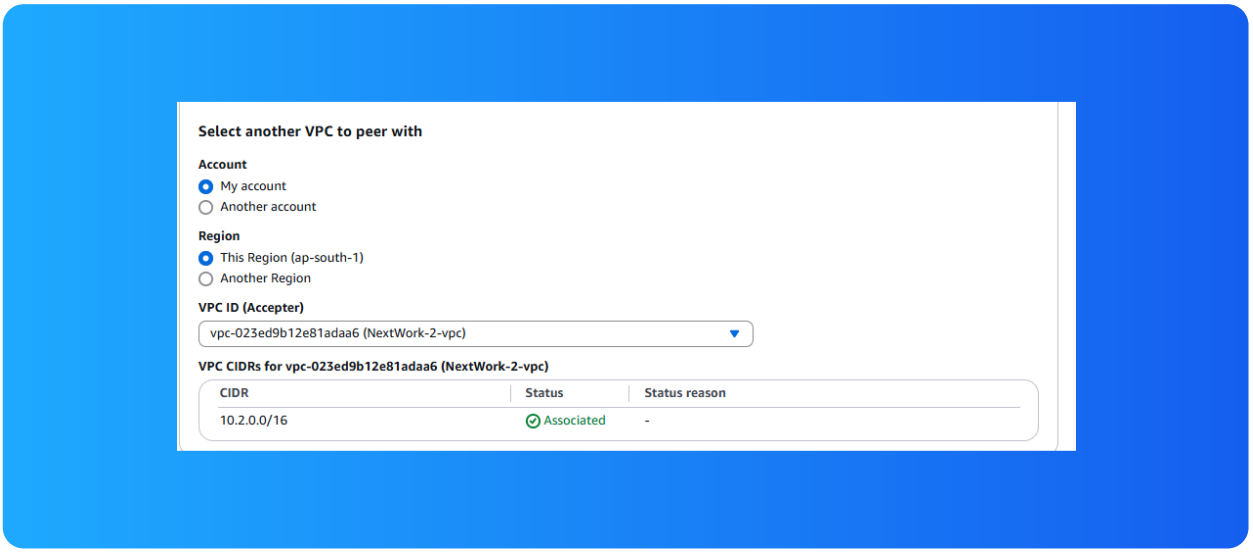


VPC Peering

A VPC peering connection is a secure network link between two VPCs, enabling direct communication without needing internet gateways, VPNs, or other external networking setups, ensuring low latency.

VPCs would use peering connections to enable seamless, secure communication between resources across different VPCs. This avoids the need for external networks, reduces latency, and enhances scalability.

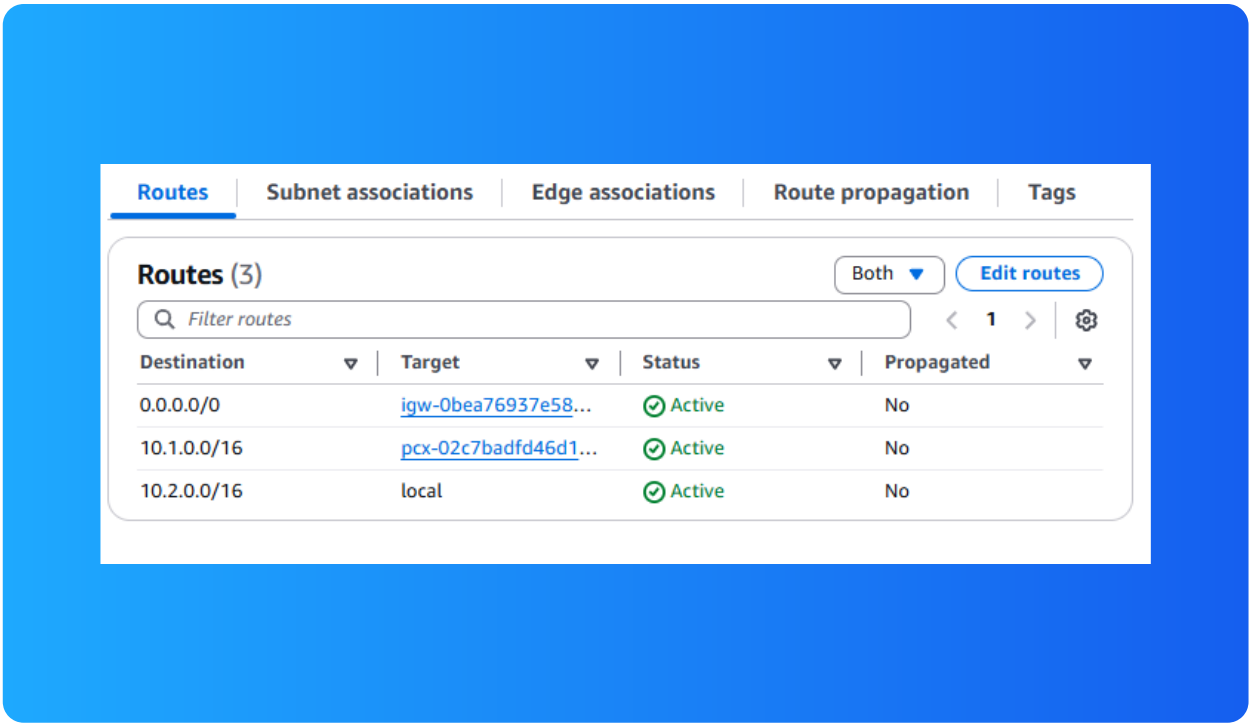
The difference between a Requester and an Accepter in a peering connection is that the Requester initiates the connection, while the Accepter approves it to establish secure communication between VPCs.



Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because routes direct traffic to the peering connection, enabling resources in each VPC to communicate seamlessly.

My VPC's new routes have a destination of the CIDR block of the other VPC of 10.1.0.0/16 and 10.2.0.0/16. The route's target was the peering connection ID to enable direct communication between the VPCs.



In the second part of my project...

Step 5 - Use EC2 Instance Connect

In this step, we are using EC2 Instance Connect to establish a connection to the first EC2 instance. If any connection errors occur, we'll troubleshoot and fix them to ensure successful access.

Step 6 - Connect to EC2 Instance 1

In this step, we are attempting to use EC2 Instance Connect to connect to Instance 1 again. If another error occurs, we'll troubleshoot and resolve it to ensure we can access the instance successfully.

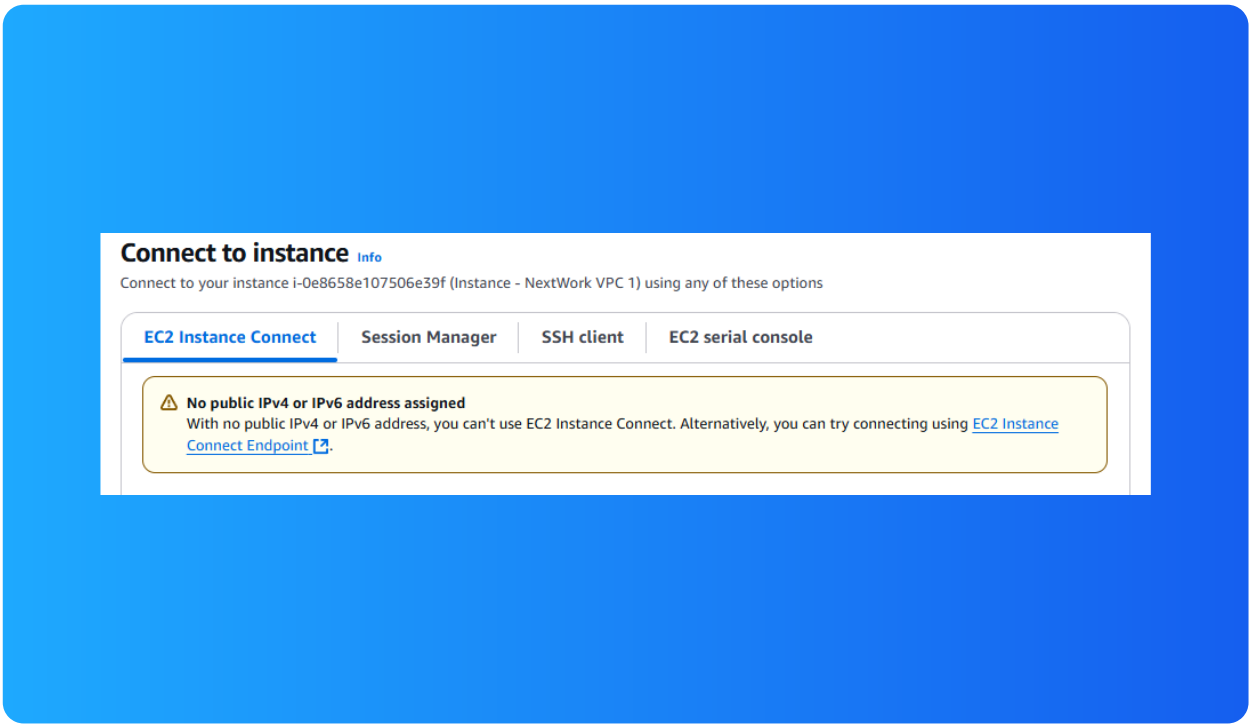
Step 7 - Test VPC Peering

In this step, we are configuring both EC2 instances to send test messages between each other. We'll troubleshoot any connection errors to ensure that Instance 2 can send messages back to Instance 1 successfully.

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to securely access my EC2 instance without needing to manage SSH keys. It provides a convenient, browser-based method for connecting to instances in AWS.

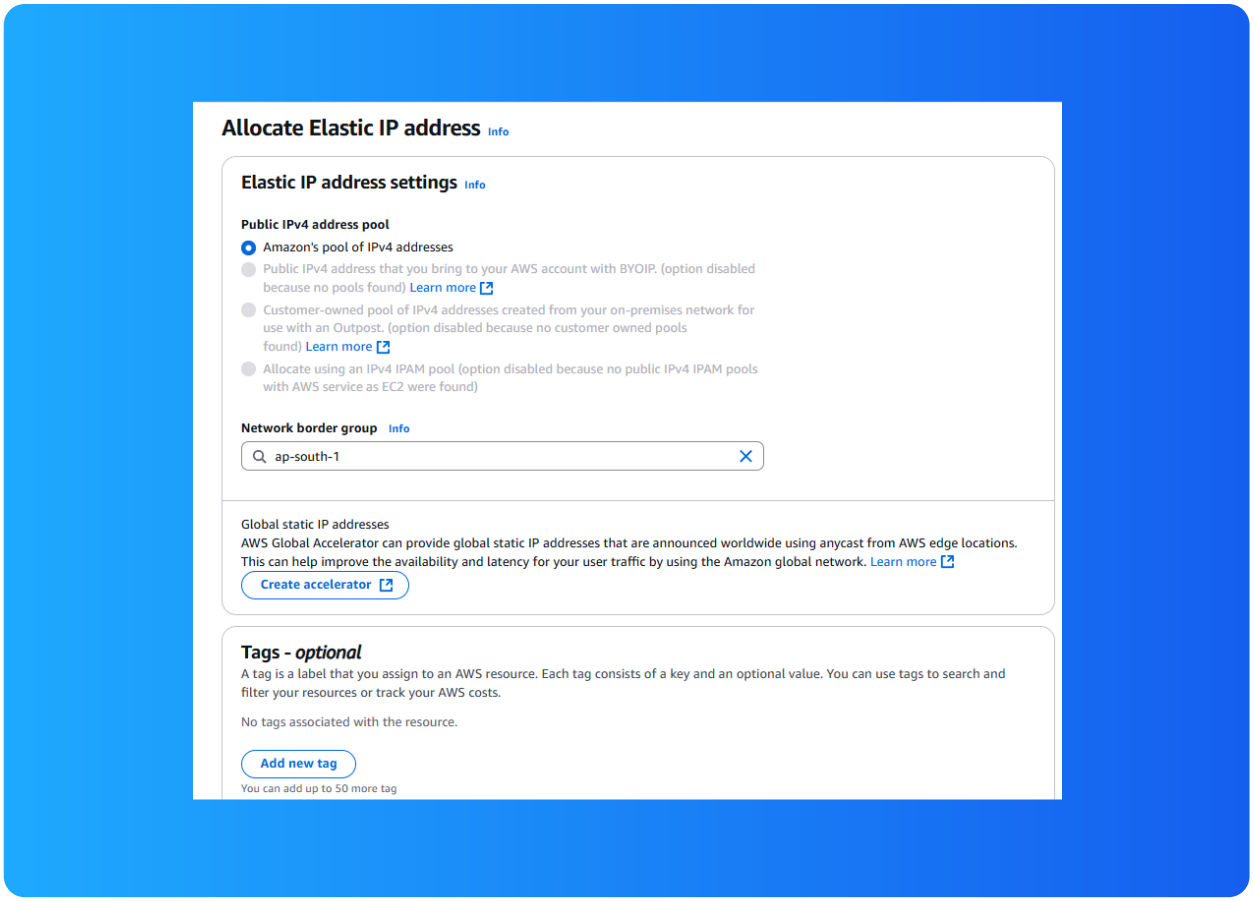
I was stopped from using EC2 Instance Connect as the instance had no public IPv4 or IPv6 assigned, preventing a successful connection. I needed to adjust the Elastic ip allocated to the EC2 instances.



Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IP addresses are static, public IPv4 addresses that can be associated with EC2 instances, ensuring they maintain consistent connectivity even after restarts.

Associating an Elastic IP address resolved the error because it provided a consistent, reachable public IP for the EC2 instance. This allowed for a stable connection, bypassing any issues with dynamic IPs.

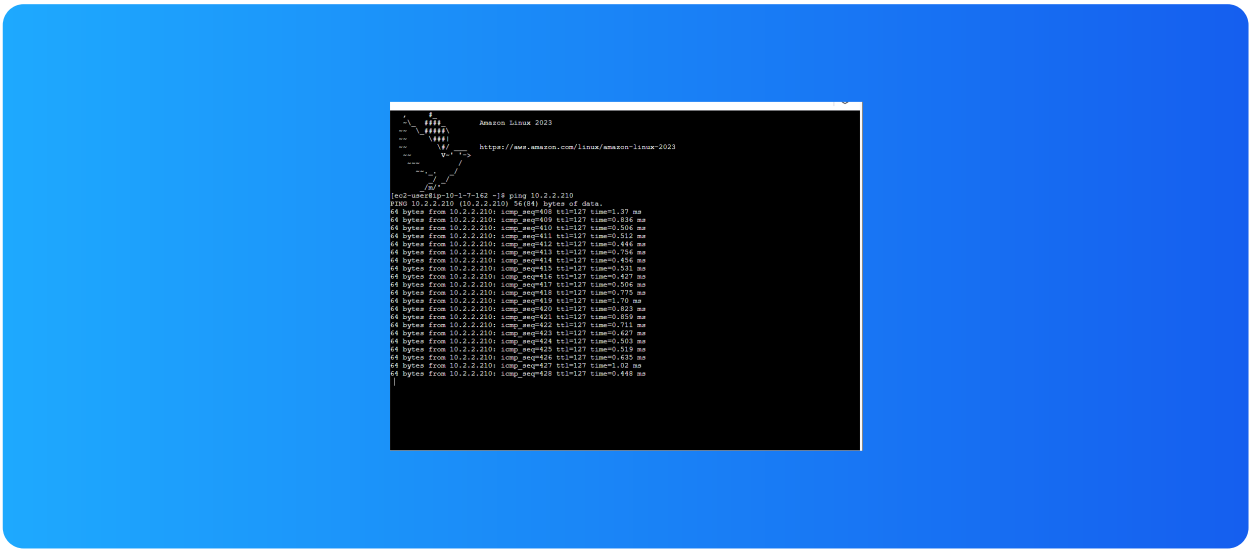


Troubleshooting ping issues

To test VPC peering, I ran the command ping <Instance 2 private IP> (i.e., 10.2.x.xxx) from Instance 1. This sent a test message to Instance 2, verifying that the VPC peering connection allowed communication between them.

A successful ping test would validate my VPC peering connection because it confirms that the instances in different VPCs can communicate with each other, indicating that the routing and network access are correctly configured.

I had to update my second EC2 instance's security group because it wasn't allowing traffic from the first instance. I added a new rule that permitted inbound ICMP traffic on port 0, allowing the ping test to pass.



Everyone should be in a job they love.

Check out nextwork.org for more projects