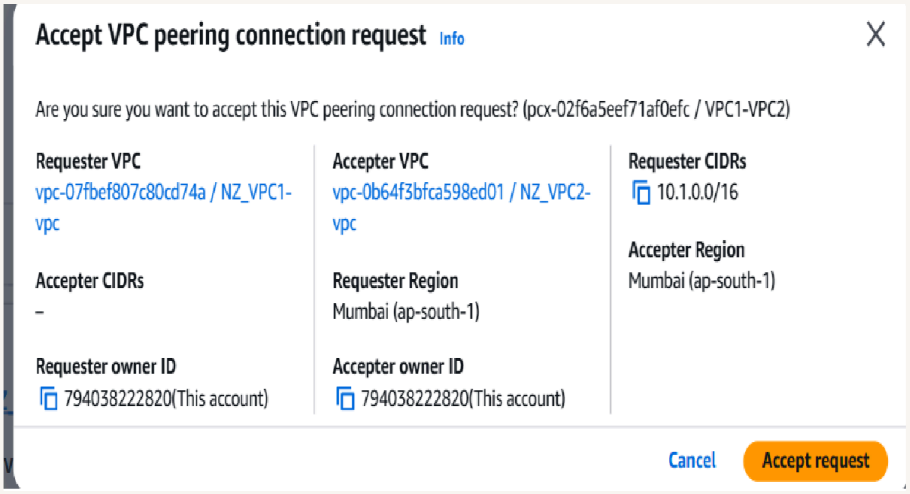
VPC Peering

Sanjana Tripathy



Sanjana Tripathy
NextWork Student

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) lets you create a secure, isolated network within AWS. It gives full control over IP ranges, subnets, route tables, and gateways—enabling secure hosting of resources and communication within a virtual network.

How I used Amazon VPC in this project

In today’s project, I used Amazon VPC to set up VPC peering between two separate VPCs and tested the connection between their resources. This allowed private communication over internal IPs without using the public internet.

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

One thing I didn’t expect in this project was how straightforward VPC peering turned out to be. I initially thought it would be a complex networking setup, but AWS made it simple to implement and test communication between VPCs.

This project took me...

This project took me about 2 hours

In the first part of my project...

Step 1 - Set up my VPC

In the first step of this project, we'll create two VPCs from scratch using the Management Console. We'll leverage the simplified VPC creation wizard that sets up the VPCs along with essential components like subnets and route tables in just minutes.

Step 2 - Create a Peering Connection

In this step, I am establishing a connection between the two VPCs I created. I'll use a VPC peering connection to bridge them, enabling secure and private communication between resources in both VPCs without traversing the public internet.

Step 3 - Update Route Tables

In this step, we're configuring route tables in both VPCs to enable bidirectional communication. This allows traffic from VPC 1 to reach VPC 2, and vice versa, using their private IP addresses through the established peering connection.

Step 4 - Launch EC2 Instances

In this step I will be launching instances in both my VPCs. This is essential step as later I will be using them to check the VPCs peering connection.

Multi-VPC Architecture

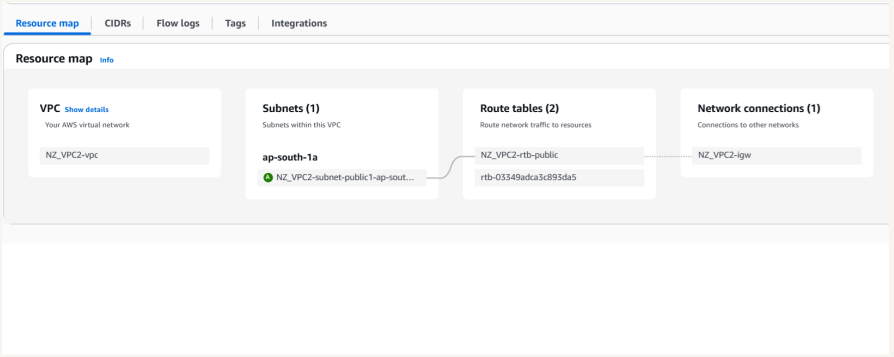
I began my project by launching two separate VPCs, each with one public subnet placed in a single Availability Zone. No private subnets were created, keeping the setup simple for establishing and testing VPC peering connectivity.

The CIDR blocks for VPCs 1 and 2 are 10.1.0.0/16 and 10.2.0.0/16 respectively. They must be unique to avoid IP address overlap, which would prevent routing between VPCs. Unique CIDRs ensure successful communication during VPC peering.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances as AWS EC2 Instance Connect manages the key exchange for us. In earlier projects, we configured key pairs manually to understand how secure access to instances works.

Sanjana Tripathy
NextWork Student



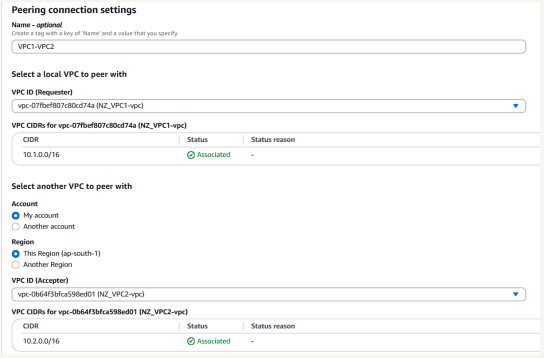
Sanjana Tripathy
NextWork Student

VPC Peering

A VPC peering connection enables two VPCs and their resources to communicate with each other using private IP addresses. It provides a secure, low-latency way for resources in different VPCs to interact without needing public internet access.

VPCs use peering connections to securely share resources, enable cross-VPC communication, or centralize services like databases or monitoring tools—without exposing traffic to the public internet.

The difference between a Requester and an Accepter in a peering connection is, a requestor is the VPC that initiates the peering connection by sending a request to another VPC.An acceptor is the VPC that receives the peering connection request.

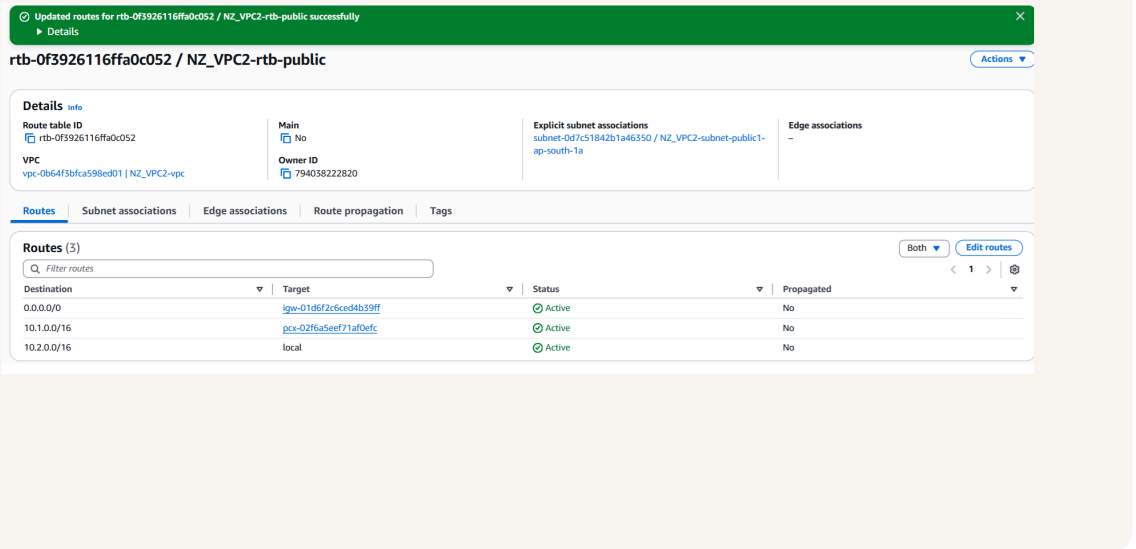


Sanjana Tripathy
NextWork Student

Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because, although the connection is established, the resources don’t yet know where to send traffic for the other VPC. Routing entries make that communication possible.

My VPCs' new routes had destinations set to the CIDR block of the opposite VPC—VPC 1 pointed to VPC 2's CIDR and vice versa. The target for each route was the established VPC peering connection, enabling seamless cross-VPC communication.



In the second part of my project...

Step 5 - Use EC2 Instance Connect

To test VPC peering connection, we need to get one of the insrtance to communicate with another instance.In this step, we will connect to one instance using "EC2 instance connect".

Step 6 - Connect to EC2 Instance 1

In the previous step, I attempted to connect to my instance using EC2 Instance Connect but encountered an issue. After applying troubleshooting steps, I'm now using EC2 Instance Connect again to verify whether the connection issue has been resolved.

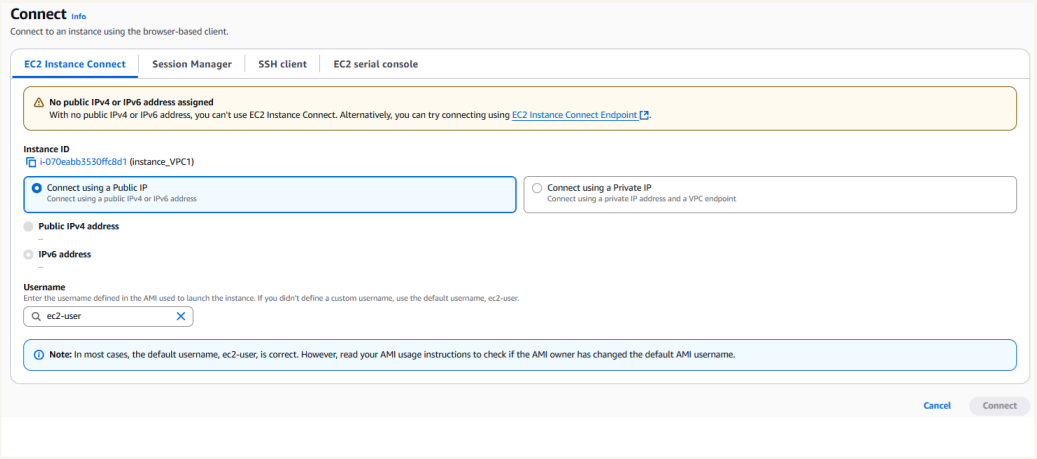
Step 7 - Test VPC Peering

In this step, I'm testing connectivity between the two VPCs. I'll initiate communication from Instance 1 in VPC 1 to Instance 2 in VPC 2, troubleshoot any connection issues, and ensure both instances can successfully exchange messages.

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to securely access my EC2 instance through the AWS Management Console. This provided browser-based SSH access without needing a key pair setup, simplifying connectivity for testing and configuration.

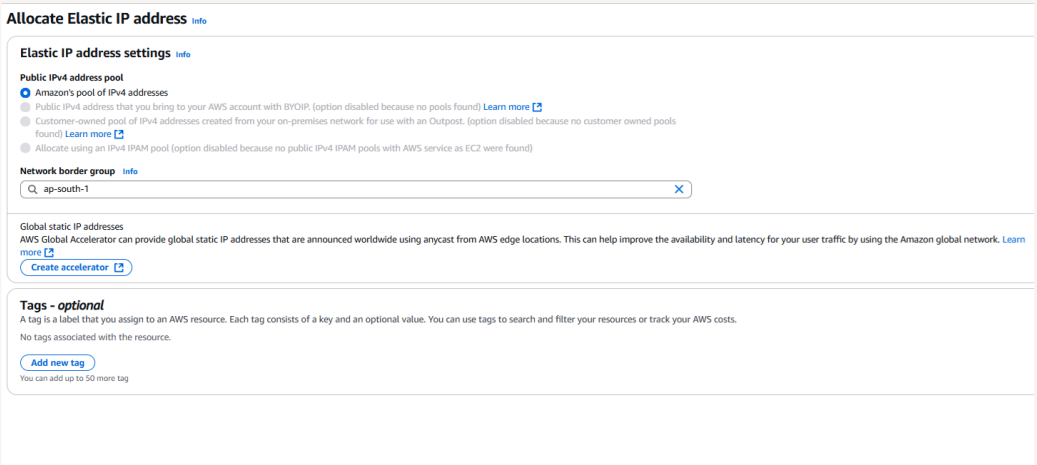
I was unable to use EC2 Instance Connect because my instance didn't have a public IP address. Since EC2 Instance Connect relies on internet-based access, a public IP is required for successful connection.



Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IPs are static, public IPv4 addresses provided by AWS that allow instances to be reachable over the internet, even if they're stopped or restarted—ensuring consistent connectivity.

Associating an Elastic IP address resolved the error because it gave my instance a stable, public IP—making it accessible over the internet and enabling EC2 Instance Connect to work successfully for remote access.

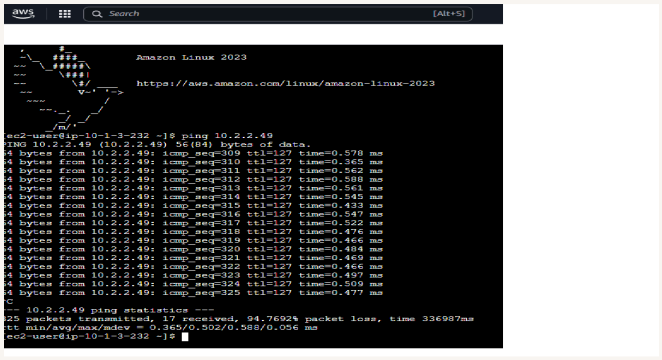


Troubleshooting ping issues

To test the VPC peering connection, I used the ping command from Instance 1 in VPC 1 to the private IP address of Instance 2 in VPC 2. This helped verify if the peering setup and routing between the two VPCs was working correctly.

A ping test validates VPC peering by checking if an instance in one VPC can reach another instance in a peered VPC using its private IP. A successful ping confirms proper routing, peering setup, and open security/NACL rules between the VPCs.

I had to update my second EC2 instance's security group because it didn't allow SSH inbound traffic. I added a new rule that permits SSH access, enabling connections from my VPC 1 instance for testing VPC peering.



The place to learn & showcase your skills

Check out nextwork.org for more projects