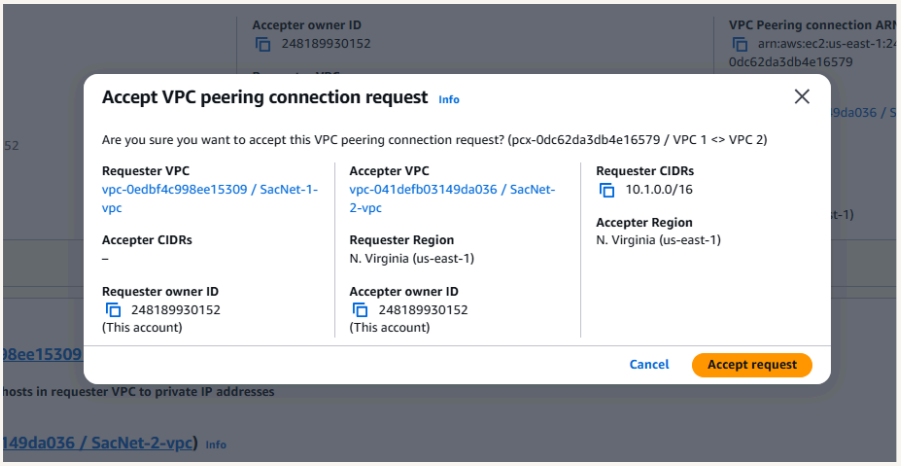
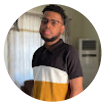
nextwork.org

VPC Peering

Jonathan Nutsugah





Jonathan Nutsugah
NextWork Student

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) lets you create a secure, isolated network in AWS where you control IP ranges, subnets, and traffic. It's useful for customizing network setup, enhancing security, and supporting hybrid cloud deployments.

How I used Amazon VPC in this project

Through the management console, I used it to virtually isolate a space to keep my resources away from external access.

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

How easy it is to navigate AWS.

This project took me...

1 Hour

In the first part of my project...

Step 1 - Set up my VPC

In this step, we create 2 VPCs from scratch.

Step 2 - Create a Peering Connection

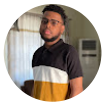
In this step, I'm going to create a connection between my VPCs so they can communicate with each other. This means setting up a way for resources in one VPC (like EC2 instances) to talk to resources in another, even though they're in separate VPCs.

Step 3 - Update Route Tables

In this step, I'm setting up route tables in both VPCs to allow traffic to flow between them through the peering connection. This ensures that VPC 1 can reach VPC 2, and VPC 2 can reach VPC 1 using their private IP addresses.

Step 4 - Launch EC2 Instances

In this step, I'm going to launch one EC2 instance in each VPC. These instances will act as test machines so I can later check if the VPC peering connection is working correctly by trying to connect from one instance to the other.



Jonathan Nutsugah
NextWork Student

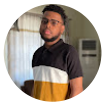
Multi-VPC Architecture

I started my project by launching VPCs, 2 separate VPCs.

The CIDR blocks for VPCs 1 and 2 are 10.1.0.0/16 and 10.2.0.0/16, respectively. They have to be unique because 2 VPCs cannot have the same address block.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances as ~~I won't be using SSH to connect, and~~ AWS actually manages a key pair for us! We don't need to manage key pairs ourselves.



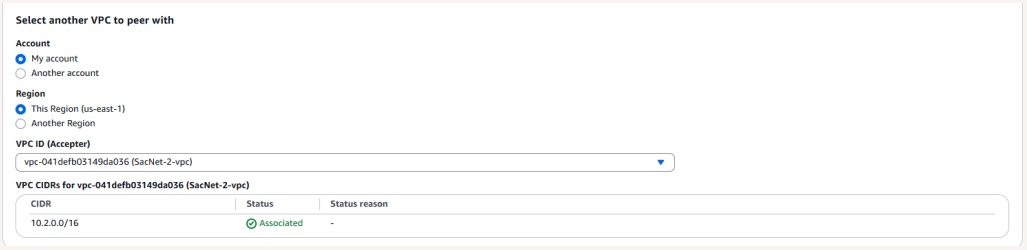
Jonathan Nutsugah
NextWork Student

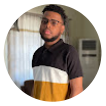
VPC Peering

A VPC peering connection is a network link between two VPCs that lets them communicate privately as if they were on the same network. With peering, I can send traffic between VPCs using private IP addresses—no need for the internet, VPN, or NAT.

Peering connections exist to let VPCs communicate securely and privately without going over the internet. They're useful when I want to share data or services between VPCs, like connecting a frontend in one VPC to a database in another.

In a VPC peering connection, the Requester is the VPC that sends the connection request, and the Accepter is the VPC that receives and approves it. Both must agree for the peering connection to be established.



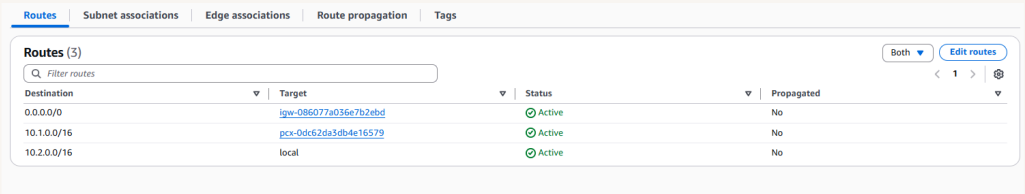


Jonathan Nutsugah
NextWork Student

Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because they don't automatically know how to reach each other. By adding a new route, I'm telling each VPC how to send traffic to the other through the peering link.

My VPCs' new routes have a destination of 10.2.0.0/16 and 10.1.0.0/16, respectively. The routes' target was peering connection.



In the second part of my project...

Step 5 - Use EC2 Instance Connect

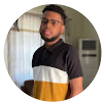
I am about to connect to my first instance, but it does not have a public IP address, and I need to fix that.

Step 6 - Connect to EC2 Instance 1

In this step, I'm going to use EC2 Instance Connect to access Instance 1 again, so I can run tests to check if it can reach Instance 2 through the VPC peering connection. This helps confirm that the connection between the two VPCs is working.

Step 7 - Test VPC Peering

In this step, I'm going to test the communication between Instance 1 and Instance 2 by sending messages from one to the other. If there are any connection issues, I'll troubleshoot and fix them until both instances can successfully talk to each other

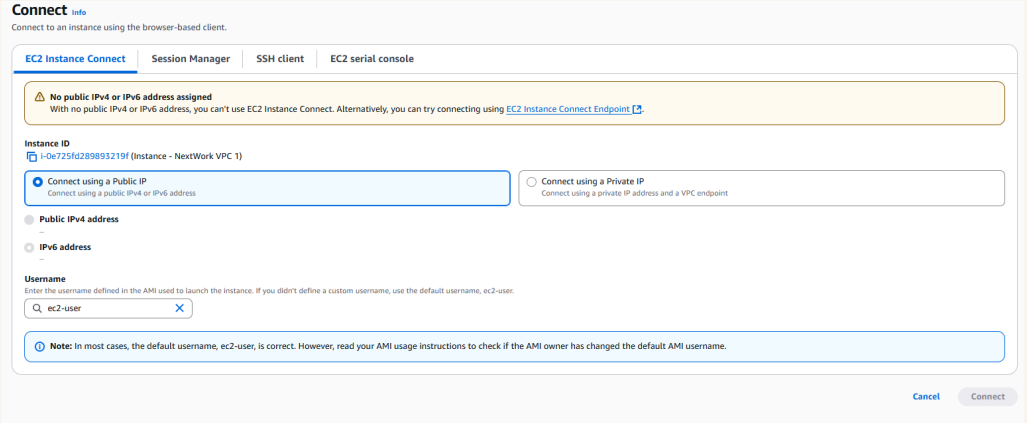


Jonathan Nutsugah
NextWork Student

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to securely access my EC2 instance through the AWS Console without needing a private key. It's a quick and easy way to run commands and test the connection between my VPCs.

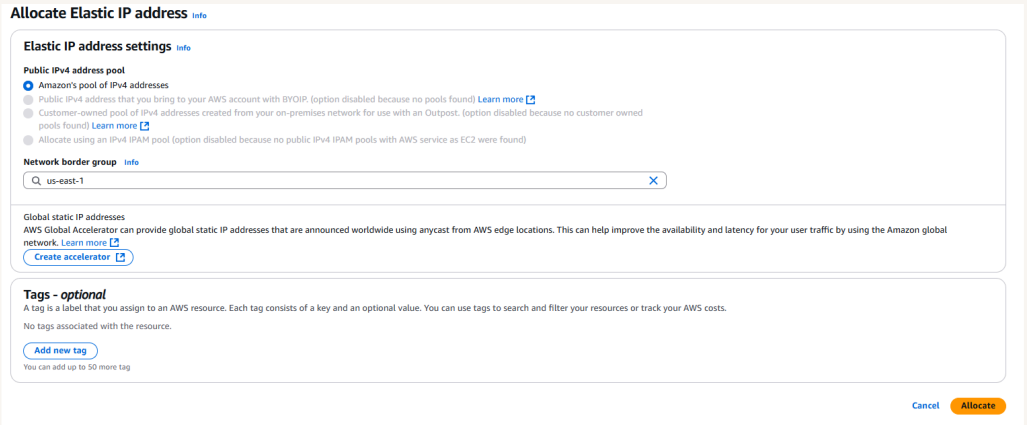
I was stopped from using EC2 Instance Connect as the instance had no public IP address.

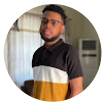


Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IP addresses are static public IPs provided by AWS that I can assign to my EC2 instances. Unlike regular public IPs, they don't change when an instance is stopped or restarted.

Associating an Elastic IP address resolved the error because it gave my instance a stable public IP, allowing me to connect from the internet reliably. Without it, the instance's public IP could change or be unavailable, causing connection issues.





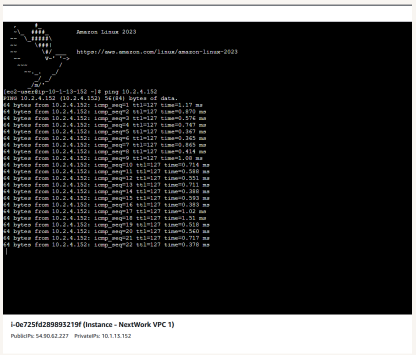
Jonathan Nutsugah
NextWork Student

Troubleshooting ping issues

'To test VPC peering, I ran the command ping 10.2.4.152

A successful ping test would validate my VPC peering connection because it shows that Instance 1 can reach Instance 2 using private IPs, meaning the peering connection, route tables, and security settings are all correctly configured.

I had to update my second EC2 instance's security group because by default, ICMP packets will not go through. I added a new rule that allows ICMP packets from the first VPC.



The place to learn & showcase your skills

Check out nextwork.org for more projects