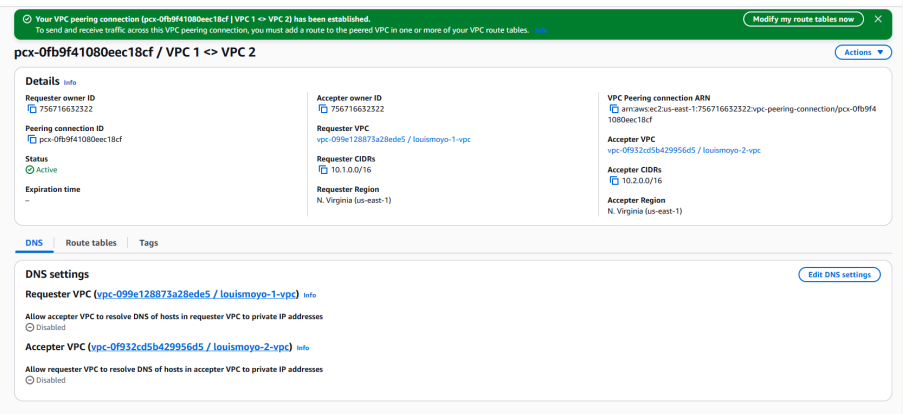
nextwork.org

VPC Peering

Louis Moyo



L
Louis Moyo
NextWork Student

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) lets you create a private, isolated network inside AWS where you control IP ranges, subnets, routing, and security. It's useful for securely hosting and managing cloud resources.

How I used Amazon VPC in this project

I created two separate VPCs, connected them with a VPC peering connection, updated route tables and security groups, and tested private communication between EC2 instances in each VPC.

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

I didn't expect to run into the EC2 Instance Connect error caused by not assigning a public IPv4 address, and then needing to fix it with an Elastic IP.

This project took me...

About 60 minutes from start to finish, including setup, troubleshooting, and testing.

In the first part of my project...

Step 1 - Set up my VPC

I'm going to use the VPC wizard and visual resource map to quickly create two brand-new VPCs from scratch, setting the foundation for connecting them later with VPC peering.

Step 2 - Create a Peering Connection

I'm going to create a VPC peering connection between my two VPCs, establishing a direct network link so resources in each VPC can communicate securely without using the public internet.

Step 3 - Update Route Tables

I'm going to update the route tables in both VPCs so traffic from VPC 1 knows to reach VPC 2 through the peering link, and traffic from VPC 2 knows to reach VPC 1 the same way.

Step 4 - Launch EC2 Instances

I'm going to launch one EC2 instance in each VPC so I can later test the VPC peering connection by sending traffic between them and confirming they can communicate privately.

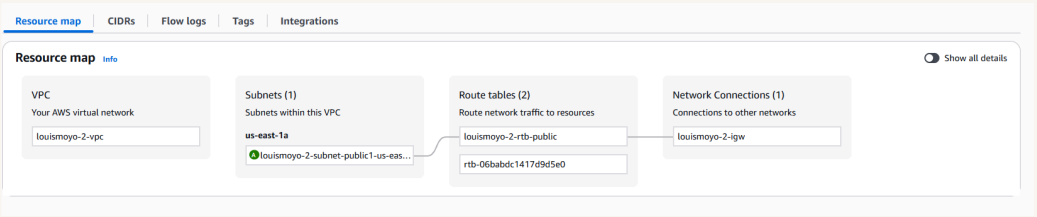
Multi-VPC Architecture

I started my project by launching the VPC wizard to quickly create two VPCs from scratch, each with a unique CIDR block, one public subnet, default tenancy, no NAT or endpoints, and ready for VPC peering setup.

The CIDR blocks for VPCs 1 and 2 are unique. They have to be unique because overlapping IP ranges would cause routing conflicts, making it impossible for the VPCs to communicate correctly when connected via VPC peering.

I also launched 2 EC2 instances

We're skipping the manual key pair setup this time because we've already practiced it in earlier projects, and we can use EC2 Instance Connect instead - AWS handles the key pair in the background, so we don't need to manage one ourselves.


Resource map
CIDRs
Flow logs
Tags
Integrations
Resource map
Info
Show all details
VPC
Your AWS virtual network
louismoyo-2-vpc
Subnets (1)
Subnets within this VPC
us-east-1a
louismoyo-2-subnet-public1-us-eas...
Route tables (2)
Route network traffic to resources
louismoyo-2-rtb-public
rtb-06babdc1417d9d5e0
Network Connections (1)
Connections to other networks
louismoyo-2-igw

Louis Moyo
NextWork Student

VPC Peering

A VPC peering connection is a networking link between two VPCs that enables them to route traffic to each other privately over the AWS network without using the public internet.

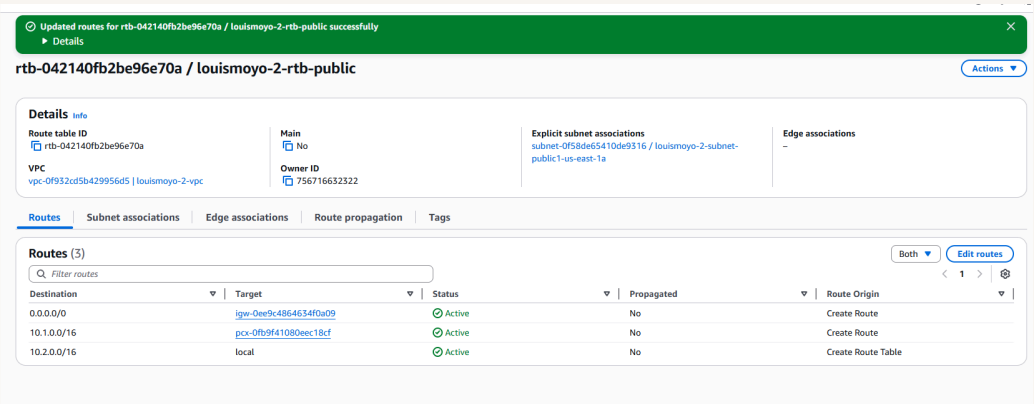
VPCs would use peering connections to share resources, allow cross-VPC communication, or connect isolated environments such as dev and prod without exposing them to the internet.

The difference between a Requester and an Accepter in a peering connection is the Requester sends the connection request, and the Accepter approves it before it becomes active.

Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because without routes pointing to the peering link, traffic won't know how to reach the other VPC's CIDR range.

My VPCs' new routes have a destination of the other VPC's CIDR block (10.1.0.0/16 or 10.2.0.0/16). The routes' target was the VPC peering connection ID.



In the second part of my project...

Step 5 - Use EC2 Instance Connect

I'm going to use EC2 Instance Connect to log in to my first EC2 instance, then troubleshoot and fix any connection errors so it can successfully communicate with the second instance over the VPC peering link.

Step 6 - Connect to EC2 Instance 1

I'm going to try connecting to Instance 1 again using EC2 Instance Connect now that it has an Elastic IP, and then troubleshoot and fix any new errors that might stop the connection from working.

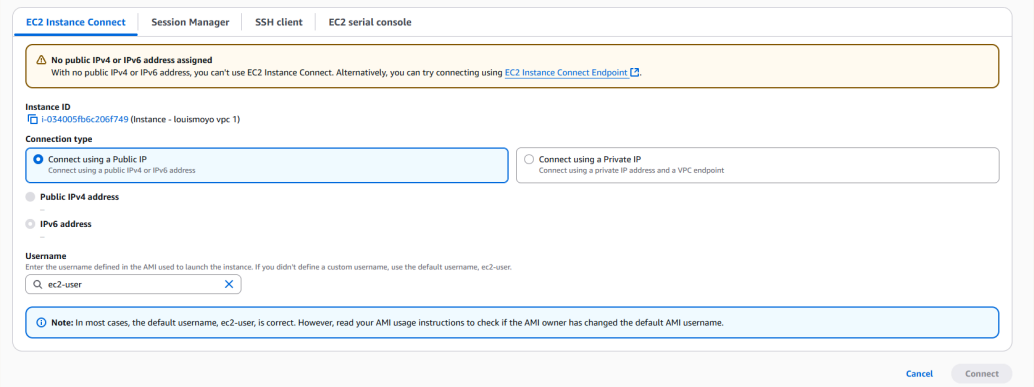
Step 7 - Test VPC Peering

I'm going to use Instance 1 to test the VPC peering connection by sending messages to Instance 2's private IP address, then troubleshoot and fix any issues until both instances can communicate successfully.

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to quickly and securely access my EC2 instance from the browser without needing to manage SSH keys, so I could run tests to verify the VPC peering connection.

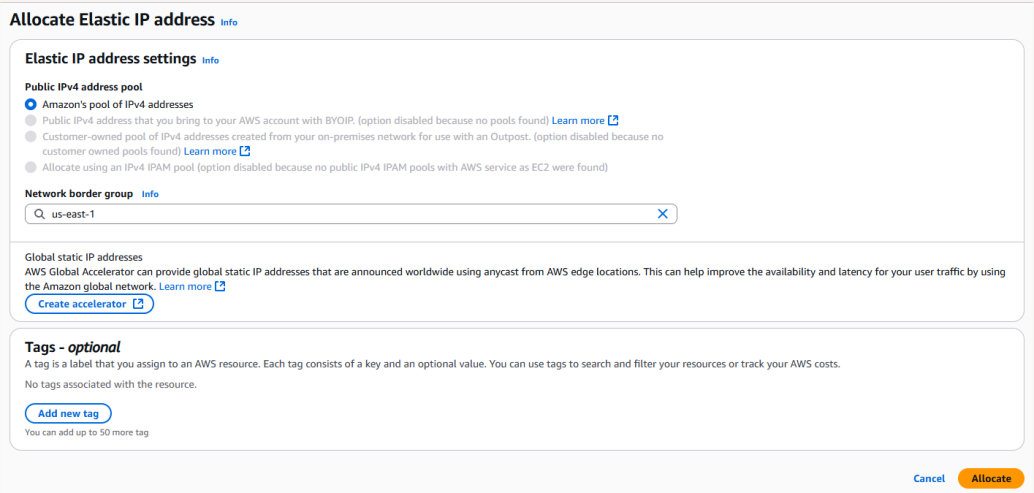
I was stopped from using EC2 Instance Connect as my EC2 instance didn't have a public IPv4 address assigned, so there was no way to reach it from the internet.



Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IP addresses are static public IPv4 addresses in AWS that you can assign to instances, ensuring they keep the same address even after stopping or restarting.

Associating an Elastic IP address resolved the error because it gave my EC2 instance a fixed public IPv4 address, allowing EC2 Instance Connect to reach it over the internet for login access.



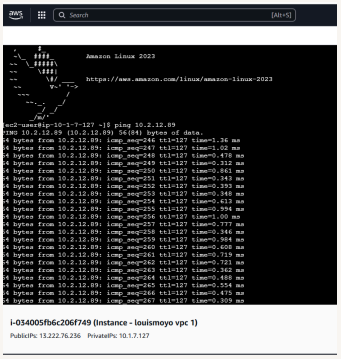
Louis Moyo
NextWork Student

Troubleshooting ping issues

To test VPC peering, I ran the command ping <private-IP-of-Instance-2> from Instance 1.

A successful ping test would validate my VPC peering connection because it proves traffic can travel over the private AWS network between the two VPCs.

I had to update my second EC2 instance's security group because its inbound rules didn't allow ICMP traffic from the first VPC. I added a new rule that permits ICMP (ping) from the CIDR block of VPC 1.



The place to learn & showcase your skills

Check out nextwork.org for more projects