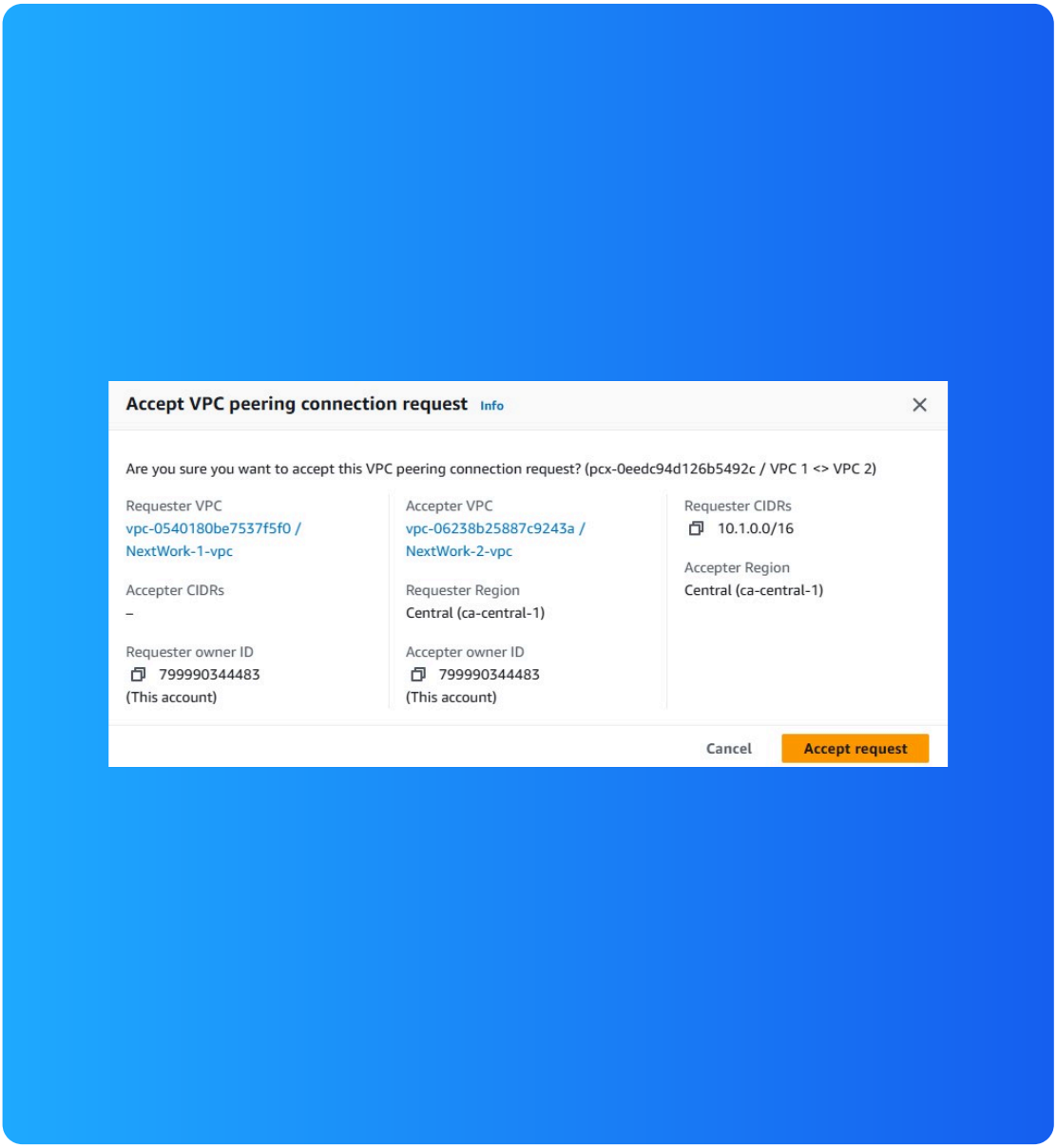
NextWork.org

VPC Peering

Nikhil Bhan



N

Nikhil Bhan
NextWork Student

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) allows users to create a private and isolated network in their AWS account. They can manage and organize resources as well as configure permissions and access to those resources.

How I used Amazon VPC in this project

In this project I used Amazon VPC to create two VPCs and a Peering Connection. I configured the route tables in each VPC in order to send traffic from one to the other. I also created EC2 instances in each VPC to test the peering connection.

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

I wasn't expecting how rewarding it was to troubleshoot and resolve a few issues while going through this project. I enjoyed fixing the connectivity issues between my VPCs and setting up proper configurations to use EC2 Instance Connect.

This project took me...

This project took me 2 hours to complete, which includes the 25 minutes I used to write my documentation.

N
Nikhil Bhan
NextWork Student

In the first part of my project...

Step 1 - Set up my VPC

In this step I'm going to create two VPCs by using the AWS Management Console. During this process I'll also be using the visual VPC resource map to see what my VPCs look like before I create them.

Step 2 - Create a Peering Connection

In this step I'm going to create a connection between my VPCs. This connection will allow my VPCs to see each other and with some testing, I'll be able to have them communicate with each other.

Step 3 - Update Route Tables

In this step I'm going to update the route tables so that traffic can be sent from VPC 1 to VPC 2 and from VPC 2 to VPC 1.

Step 4 - Launch EC2 Instances

In this step I'll create an EC2 instance in each of my VPCs and testing the peering connection between them.

N

Nikhil Bhan
NextWork Student

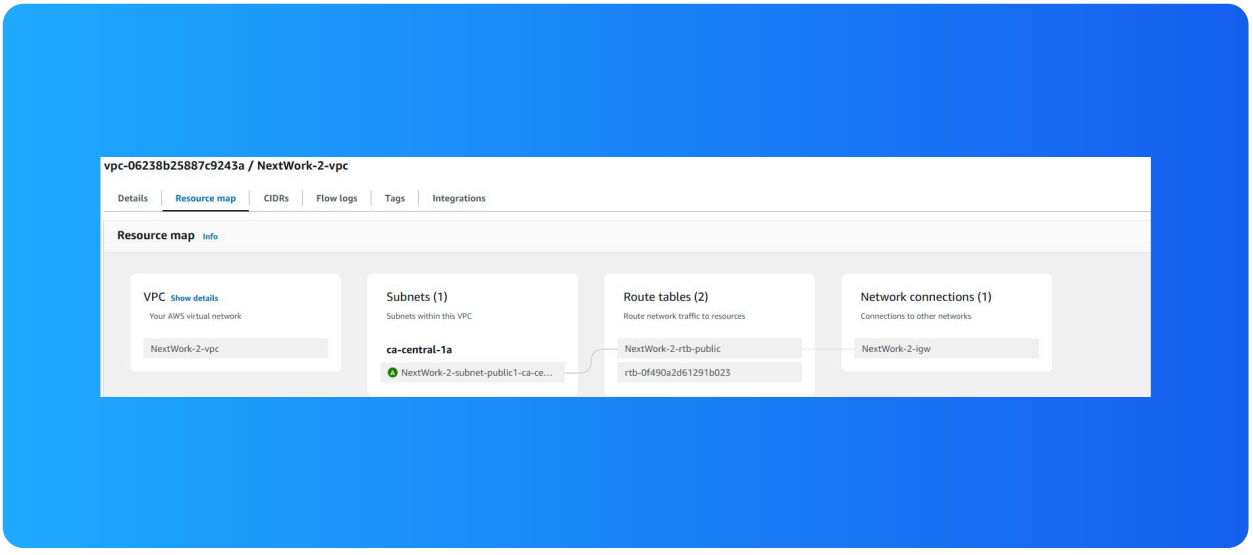
Multi-VPC Architecture

I started my project by launching two VPCs in the VPC Dashboard. In each one, I created one public subnet with one Availability Zone and its own IPv4 CIDR Block.

The CIDR blocks for VPCs 1 and 2 are: VPC 1: 10.1.0.0/16 VPC 2: 10.2.0.0/16 They have to be unique because I'm going to connect these two VPCs; in this case I can't have IP duplicates so the CIDR Blocks or range of IPs have to be unique.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances since I'll be using EC2 Instance Connect. By using this service I won't have to manage a key pair to log in to my instances; AWS will manage that key pair for me instead.



N

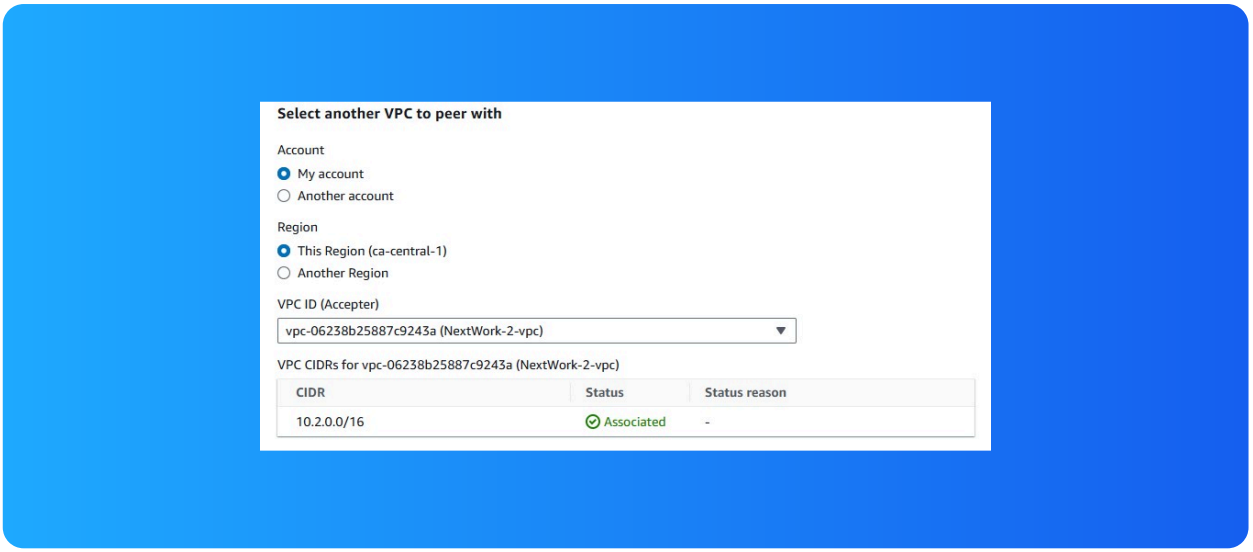
Nikhil Bhan
NextWork Student

VPC Peering

A VPC peering connection is a direct connection between two VPCs. This type of connection allows these VPCs and their resources to send and receive traffic between themselves by using their private IP addresses.

VPCs would use peering connections to avoid transferring traffic through the Internet; transferring data on the public Internet would be a potential security risk.

The difference between a Requester and an Accepter in a peering connection is that the requester is the VPC that is initiating that peering connection. The accepter is the VPC that will accept or decline the invitation to connect with the other VPC.



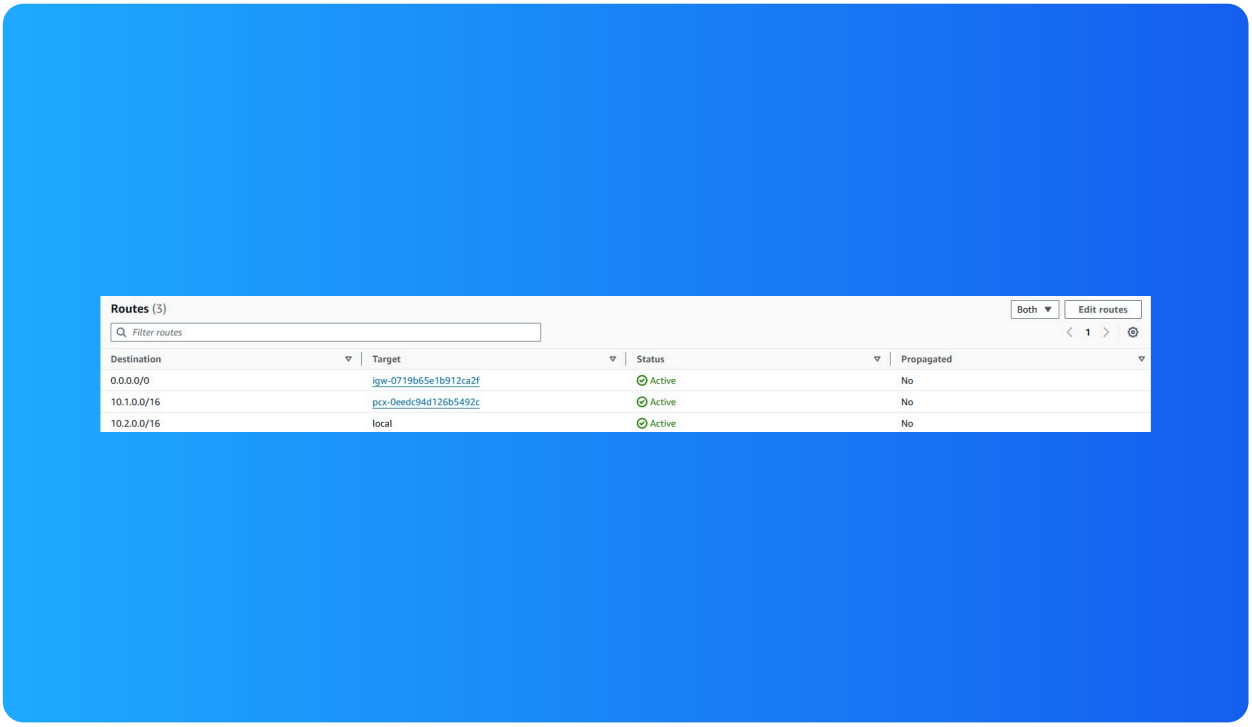
N

Nikhil Bhan
NextWork Student

Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because the resources in their respective VPC won't know how to reach their destination without a route. A route needs to be set up in each VPC to send out traffic.

My VPCs' new routes have a destination of 10.2.0.0/16 and 10.1.0.0/16, which correspond to VPC 1 and VPC 2 respectively. The routes' target was the peering connection that I created earlier, which will forward traffic from one VPC to the other one.



N
Nikhil Bhan
NextWork Student

In the second part of my project...

Step 5 - Use EC2 Instance Connect

In this step I'll be using EC2 Instance Connect to log in to my first EC2 instance.

Step 6 - Connect to EC2 Instance 1

In this step I'll use EC2 Instance Connect to establish a connection to my EC2 instance.

Step 7 - Test VPC Peering

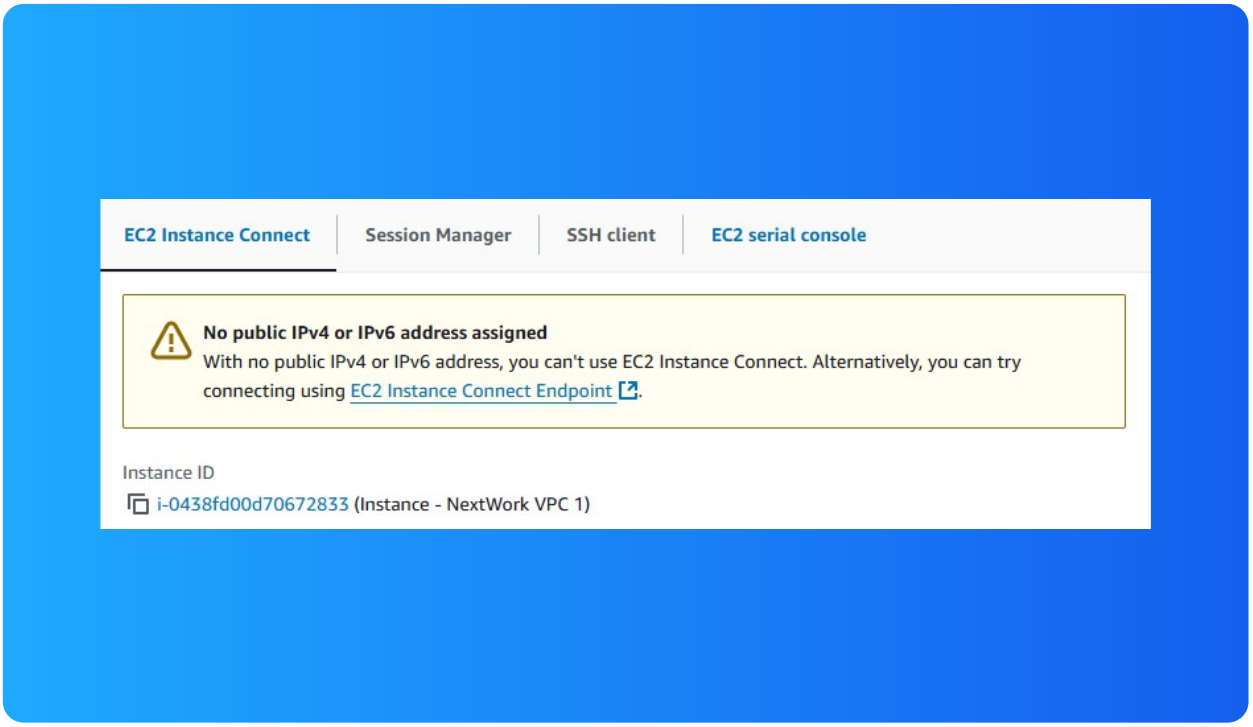
In this step I'll send test messages from my first EC2 instance in VPC 1 to my 2nd EC2 instance in VPC 2.

N
Nikhil Bhan
NextWork Student

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to log into my EC2 instance in VPC 1.

I was stopped from using EC2 Instance Connect since there was no public IPv4 or IPv6 address assigned to my EC2 instance in VPC 1. I'm connecting to it through the Internet, so therefore a public IP address and a public subnet are required.

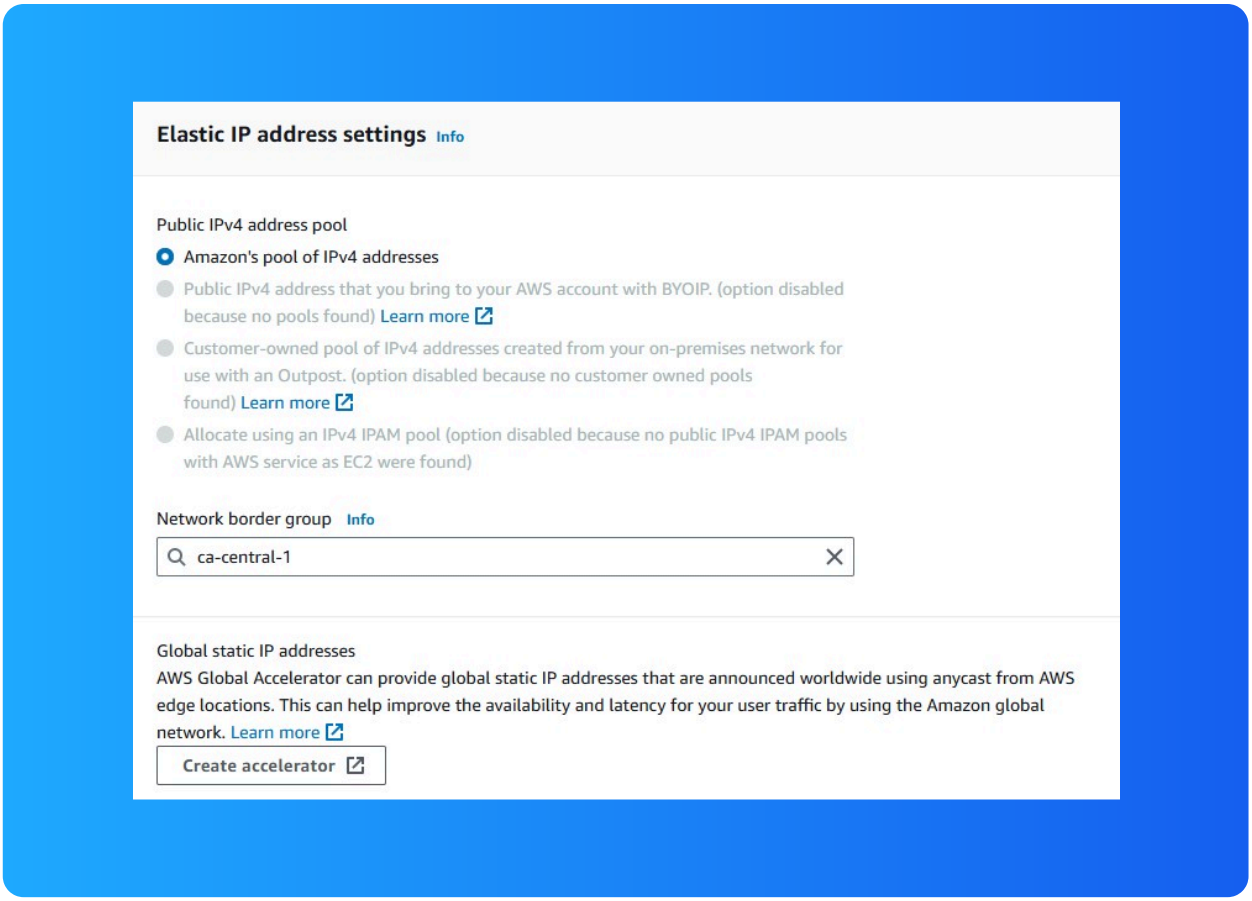


N
Nikhil Bhan
NextWork Student

Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IP addresses are static IPv4 addresses that are allocated to an AWS account and can be assigned to AWS resources.

Associating an Elastic IP address resolved the error because it provides the EC2 instance a public IP address; with this public IP address I would be able to connect through the Internet and log in to my instance.



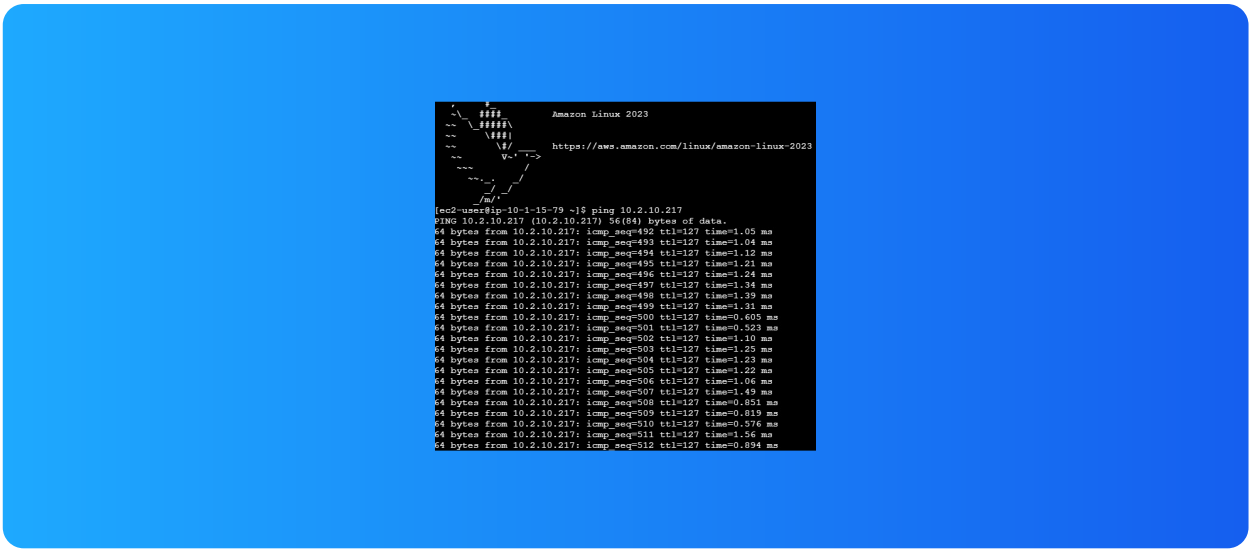
Nikhil Bhan
NextWork Student

Troubleshooting ping issues

To test VPC peering, I ran the command ping 10.2.10.217, which is the private IP address for the EC2 instance in VPC 2's public subnet.

A successful ping test would validate my VPC peering connection because it can verify that the instance in VPC 1 can send a message to the instance in VPC 2 and receive a response from it, which confirms that both instances can communicate.

I had to update my second EC2 instance's security group because the ICMP traffic from my first instance in VPC 1 was being blocked from entering the instance in VPC 2. I added a new rule that allows ICMP traffic from VPC 1's CIDR block 10.1.0.0/16.



Everyone should be in a job they love.

Check out nextwork.org for more projects