



# VPC Peering

H

Harshada Kripal

A screenshot of the AWS VPC Peering Connection details page for connection `pcx-0a7461ab17d506918`. The connection status is `Active`. The requester information includes owner ID `051826730775`, VPC ID `vpc-063778841e689ea3a`, CIDR `10.1.0.0/16`, and region `Ohio (us-east-2)`. The accepter information includes owner ID `051826730775`, VPC ID `vpc-033a84a7dbc73b648`, CIDR `10.2.0.0/16`, and region `Ohio (us-east-2)`. The VPC Peering connection ARN is `arn:aws:ec2:us-east-2:051826730775:vpc-peering-connection/pcx-0a7461ab17d506918`. A green banner at the top indicates that the connection has been requested.



**Harshada Kripal**  
NextWork Student

[nextwork.org](http://nextwork.org)

# Introducing Today's Project!

## What is Amazon VPC?

Amazon VPC is Virtual Private Cloud which allows us to launch our resources in our VPC. It is a foundational networking service that let's us create our own network so that we can control traffic flow, security and organize resources into public and private subnet.

## How I used Amazon VPC in this project

In today's project, I used Amazon VPC to setup a multi VPC architecture (I setup 2 VPCs) create a peering connection between them and update security group rules to run a successful connectivity test to validate my VPC Peering setup.

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

One thing I didn't expect in this project was I didn't expect to need a public IPV4 address for EC2 instance connect to work. Also that elastic IP can assign static IPV4 to resources.

## This project took me...

This project took me almost 2 hours to complete.



## In the first part of my project...

### Step 1 - Set up my VPC

In this step, I'm using a VPC resource map/launch wizard to create Two VPC's and their components in just few minutes.

### Step 2 - Create a Peering Connection

In this step, I'm setting up an VPC peering connection which is a component to directly connect two VPC's together.

### Step 3 - Update Route Tables

In this step, I will be setting up route tables to set up a way for traffic coming from VPC1 to get to VPC2 so that traffic can be directed to the peering connection.

### Step 4 - Launch EC2 Instances

In this step, I will launch EC2 instances in each of the VPC's because to directly connect them and use them later to test the VPC peering connection.

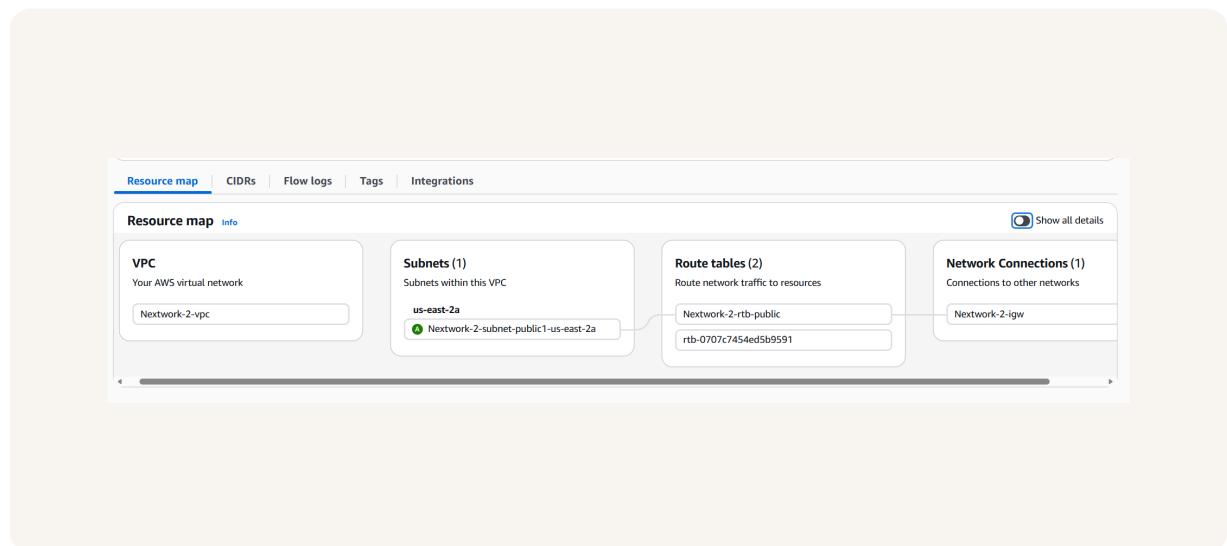
# Multi-VPC Architecture

I started my project by launching two VPC's. They have unique CIDR blocks and they each have one public subnet.

The CIDR blocks for VPCs 1 and 2 are 10.1.0.0/16 & 10.2.0.0/16 They have to be unique because once you setup a VPC peering coonection route tables need unique addressess for correct routing across VPC's.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances as I'm using EC2 instance connect to directly connect with my EC2 instance later in this project which handles key pair creation & management for us.



## VPC Peering

A VPC peering connection lets VPCs and their resources route traffic between them using their private IP addresses. This means data can now be transferred between VPCs without going through the public internet.

VPCs would use peering connections to let their resources route traffic between them using their private IP addresses. Transferring data between VPCs would use resources' public address - meaning VPCs have to communicate over the public internet.

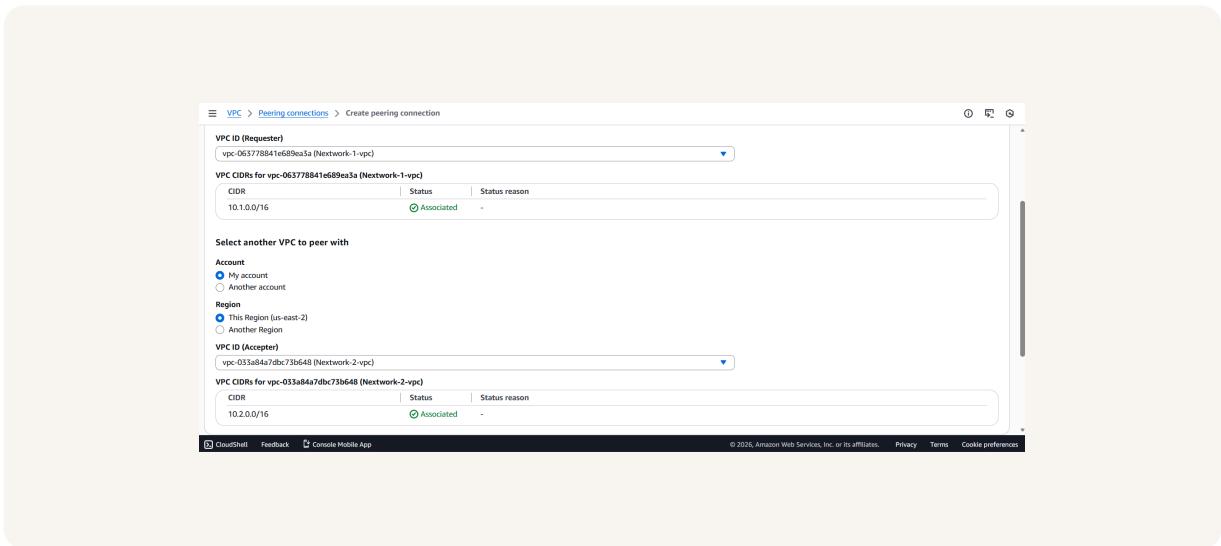
The difference between a Requester and an Acceptor in a peering connection is the Requester is the VPC that initiates a peering connection. As the requester, they will be sending the other VPC an invitation to connect the Acceptor is the VPC that receives a peering connection request. The Acceptor can either accept or decline the invitation. This means the peering connection isn't actually made until the other VPC also agrees to it.



# Harshada Kripal

NextWork Student

[nextwork.org](https://nextwork.org)



# Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because the default route table doesn't have a peering connection yet it needs to be setup so that the resources can be directed to the peering connection when trying to reach the other VPC.

My VPCs' new routes have a destination og the other VPC's CIDR block. The routes' target was the peering connection setup.

The screenshot shows the AWS Route Tables interface. At the top, there are tabs for 'Details' (selected), 'Info', 'Route table ID', 'Main', 'Owner ID', 'Explicit subnet associations', and 'Edge associations'. Below these are tabs for 'Routes' (selected), 'Subnet associations', 'Edge associations', 'Route propagation', and 'Tags'. The 'Routes' tab displays a table with three rows:

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	<a href="#">igw-04f542c0a885ed380</a>	Active	No	Create Route
10.1.0.0/16	<a href="#">pex-0a7461ab17d506918</a>	Active	No	Create Route
10.2.0.0/16	local	Active	No	Create Route Table

## In the second part of my project...

### Step 5 - Use EC2 Instance Connect

In this step, I will be using EC2 instance to connect directly with my first EC2 instance because i need to use my EC2 instance for connectivity test later in this project.

### Step 6 - Connect to EC2 Instance 1

In this step, I will be reattempting my connection to Instance Nextwork-vpc-1 and resolving another error preventing from using EC2 instance connect to directly connect to the instance.

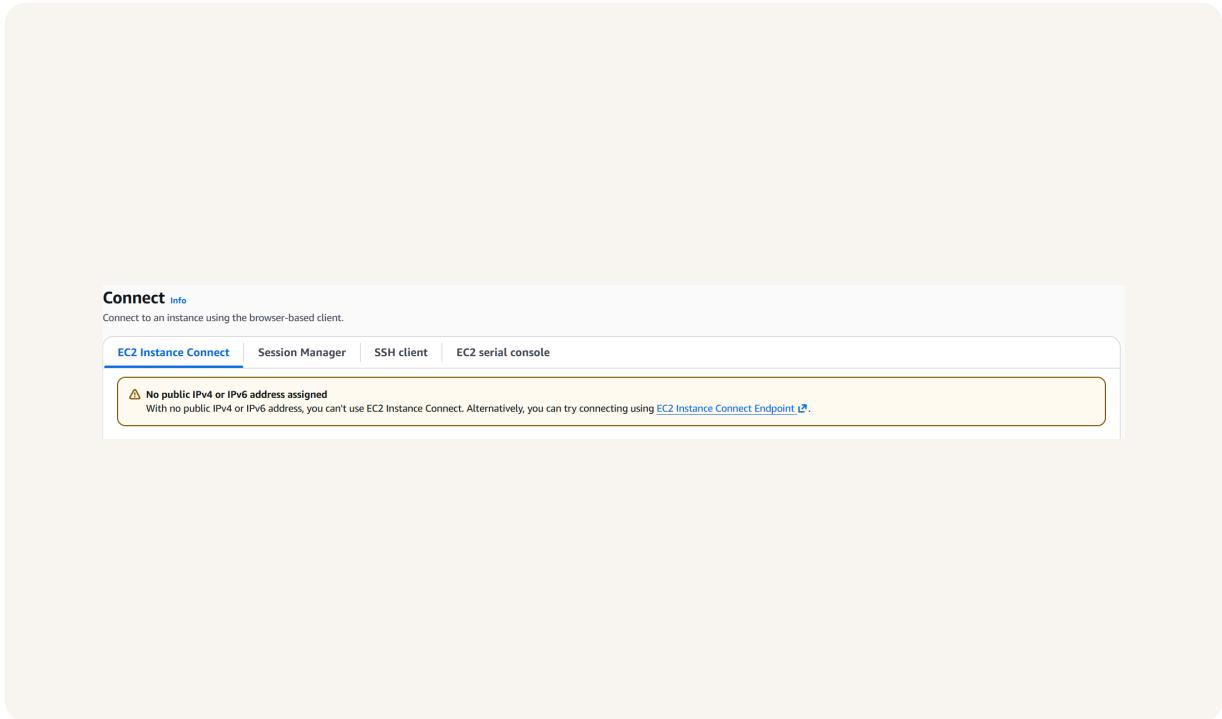
### Step 7 - Test VPC Peering

In this step, I will be using the instance Nextwork-vpc-1 to attempt a direct connection with Nextwork-vpc-2 instance because I can validate that my peering is setup properly.

# Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to directly connect with my instance Nextwork-vpc -1 just by using AWS management console.

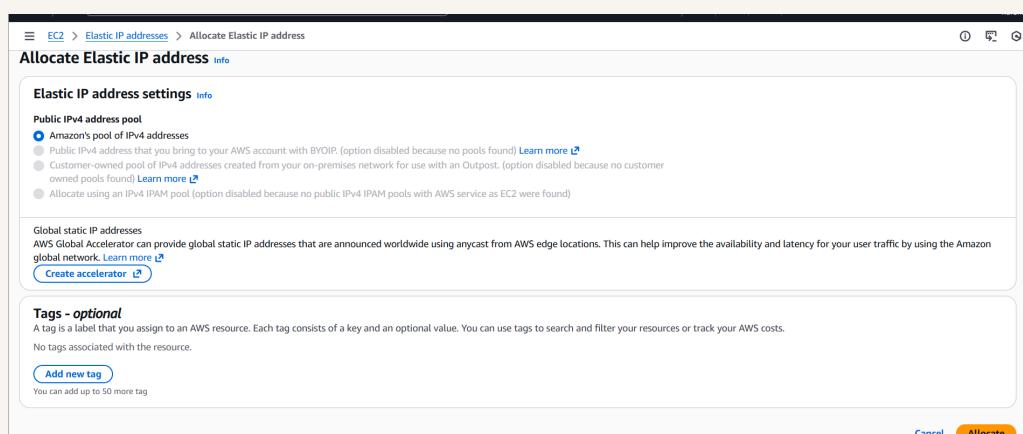
I was stopped from using EC2 Instance Connect as my instance did not have an public IPV4 address.



# Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IP addresses are public IPV4 addresses that we can request for our AWS account and then delegate to specific resources that will use this ip address.

Associating an Elastic IP address resolved the error because it gives my EC2 instance a public IP address.

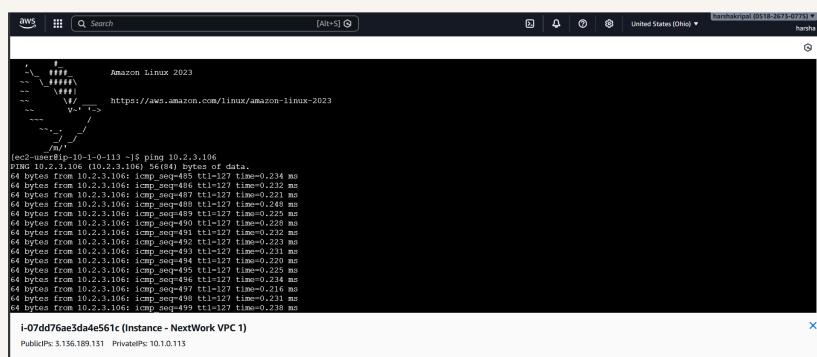


# Troubleshooting ping issues

To test VPC peering, I ran the command \$ping 10.2.X.XXX this is the private IPV4 address of other EC2 instance in VPC2.

A successful ping test would validate my VPC peering connection because this ping test would not get any replies from the other EC2 instance if the peering connection did not successfully connect to our two VPC's. Getting Ping replies equals hearing connection was set up properly.

I had to update my second EC2 instance's security group because it was not letting in ICMP traffic which is the traffic type for ping message. I added a new rule that allows ICMP traffic coming in from any resource in VPC2.



```
(ec2-user@ip-10-1-0-113 ~) $ ping 10.2.3.106
PING 10.2.3.106 (10.2.3.106) 56(84) bytes of data.
64 bytes from 10.2.3.106: icmp_seq=486 ttl=127 time=0.234 ms
64 bytes from 10.2.3.106: icmp_seq=487 ttl=127 time=0.222 ms
64 bytes from 10.2.3.106: icmp_seq=487 ttl=127 time=0.221 ms
64 bytes from 10.2.3.106: icmp_seq=488 ttl=127 time=0.225 ms
64 bytes from 10.2.3.106: icmp_seq=489 ttl=127 time=0.225 ms
64 bytes from 10.2.3.106: icmp_seq=490 ttl=127 time=0.228 ms
64 bytes from 10.2.3.106: icmp_seq=491 ttl=127 time=0.223 ms
64 bytes from 10.2.3.106: icmp_seq=492 ttl=127 time=0.223 ms
64 bytes from 10.2.3.106: icmp_seq=493 ttl=127 time=0.231 ms
64 bytes from 10.2.3.106: icmp_seq=494 ttl=127 time=0.225 ms
64 bytes from 10.2.3.106: icmp_seq=495 ttl=127 time=0.225 ms
64 bytes from 10.2.3.106: icmp_seq=496 ttl=127 time=0.234 ms
64 bytes from 10.2.3.106: icmp_seq=497 ttl=127 time=0.221 ms
64 bytes from 10.2.3.106: icmp_seq=498 ttl=127 time=0.238 ms
64 bytes from 10.2.3.106: icmp_seq=499 ttl=127 time=0.238 ms
```

i-07d476ae3da4e561c (Instance - NextWork VPC 1)  
PublicIPs: 5.136.189.131 PrivateIPs: 10.1.0.113



[nextwork.org](https://nextwork.org)

# The place to learn & showcase your skills

Check out [nextwork.org](https://nextwork.org) for more projects

