



VPC Peering



shravanirg03@gmail.com

>Your VPC peering connection (pcx-0c914897428fb4d74 | VPC 1 <=> VPC 2) has been established.
To send and receive traffic across this VPC peering connection, you must add a route to the peered VPC in one or more of your VPC route tables. [Info](#)

Modify my route tables now X

VPC > Peering connections > pcx-0c914897428fb4d74

pcx-0c914897428fb4d74 / VPC 1 <=> VPC 2

[Actions ▾](#)

Details	Info
Requester owner ID 975050239788	Acceptor owner ID 975050239788
Peering connection ID pcx-0c914897428fb4d74	Requester VPC vpc-090b5d5a10a7eddf / NextWork-1-vpc
Status Active	Requester CIDRs 10.1.0.0/16
Expiration time ~	Requester Region Mumbai (ap-south-1)
	VPC Peering connection ARN arn:aws:ec2:ap-south-1:975050239788:vpc-peering-connection:pcx-0c914897428fb4d74
	Acceptor VPC vpc-01c5d48a952ad651f / NextWork-2-vpc
	Acceptor CIDRs 10.2.0.0/16
	Acceptor Region Mumbai (ap-south-1)

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is a core networking service in AWS that allows us to build secure and scalable network architectures to host and manage resources effectively within the cloud.

How I used Amazon VPC in this project

In today's project, I used Amazon VPC to implement VPC Peering by setting up a multi-VPC architecture and establishing a peering connection between them. I verified the connection using EC2 Instance Connect and updating security group rules.

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

I did not expect that Elastic IPs can assign static IPv4 addresses to resources. I also did not expect that an EC2 instance can have multiple private IPs.

This project took me...

This project took me 2 hours to complete including the time to write documentation.

In the first part of my project...

Step 1 - Set up my VPC

To streamline the VPC creation process, I'm utilizing the VPC resource map/launch wizard to quickly deploy two VPCs with their respective components.

Step 2 - Create a Peering Connection

Setting up a VPC Peering Connection, which is a VPC component designed to directly connect two VPCs together.

Step 3 - Update Route Tables

In this step, I am going to update the route tables so that the traffic coming from both VPCs can navigate easily.

Step 4 - Launch EC2 Instances

In this step, we are launching EC2 instances in the subnets of each VPC that we set up. These EC2 instances will be used to test the connectivity of the VPC Peering Connection.

Multi-VPC Architecture

I started my project by launching two VPCs - 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 and 10.2.0.0/16, respectively. These CIDR blocks must be unique to ensure correct routing across VPCs after establishing a VPC peering connection.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances as it was not necessary in this project. The previous projects covered the working of SSH and EC2 Instance Connect using which AWS manages the key-pairs for the user.



VPC Peering

A VPC peering connection creates a link between two VPCs that enables securely routing traffic between them using private addresses.

VPCs would use peering connections to communicate with each other without traversing the public internet. This provides them with a secure and efficient way to communicate.

A Requester is a VPC that initiates a peering connection by sending an invitation to connect. An Acceptor is the VPC that receives a peering connection request. It can either accept or decline the invitation.

Select another VPC to peer with

Account

My account
 Another account

Region

This Region (ap-south-1)
 Another Region

VPC ID (Acceptor)

vpc-01c5d48a952ad651f (NextWork-2-vpc)

VPC CIDRs for vpc-01c5d48a952ad651f (NextWork-2-vpc)

CIDR	Status	Status reason
10.2.0.0/16	Associated	-

Updating route tables

The traffic from the VPCs need directions from route table while navigating. There doesn't exist a route that directs the traffic to the peering connection. Thus, route tables need to be updated for traffic from one VPC to reach the other VPC.

My VPCs' new routes have a destination of the other VPC's CIDR block. The routes' target was the Peering Connection that I set up.

rtb-0d792ad1436814a07 / NextWork-2-rtb-public				
Details	Routes	Subnet associations	Edge associations	Route propagation
Routes (3)				
<input type="text"/> Filter routes				
Destination	Target	Status	Propagated	
0.0.0.0/0	igw-0c731c9d5ddc12e9	Active	No	
10.1.0.0/16	px-0c914897428fb4d74	Active	No	
10.2.0.0/16	local	Active	No	

In the second part of my project...

Step 5 - Use EC2 Instance Connect

In this step, we are using EC2 Instance Connect to connect to the first EC2 instance while fixing any connection errors that might occur in the process.

Step 6 - Connect to EC2 Instance 1

Reattempting to connect to Instance - NextWork VPC 1, and resolving another error preventing us from using EC2 Instance Connect to directly connect with the instance.

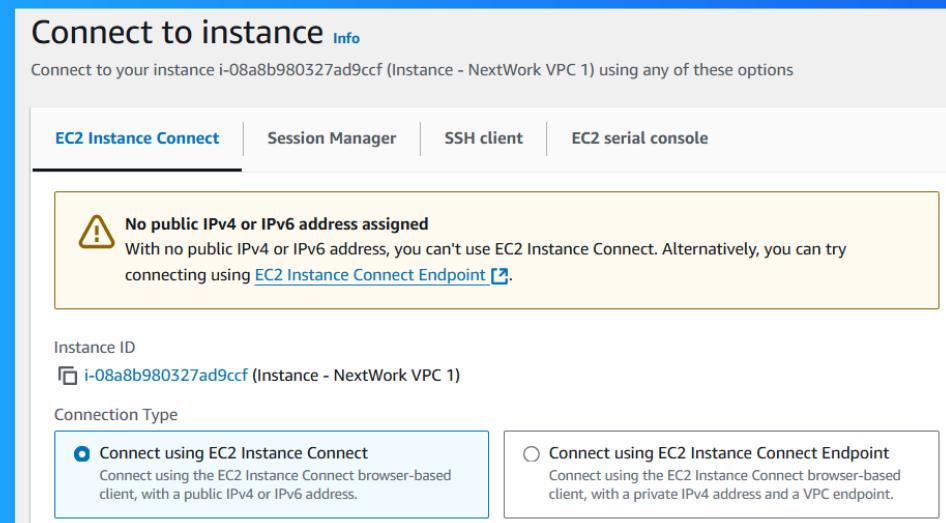
Step 7 - Test VPC Peering

We are going to send a message to Instance 2 from Instance 1 and solve all the connection errors that arise in the process until Instance 2 responds to the message.

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to directly connect with Instance - NextWork VPC 1 just by using the AWS Management Console.

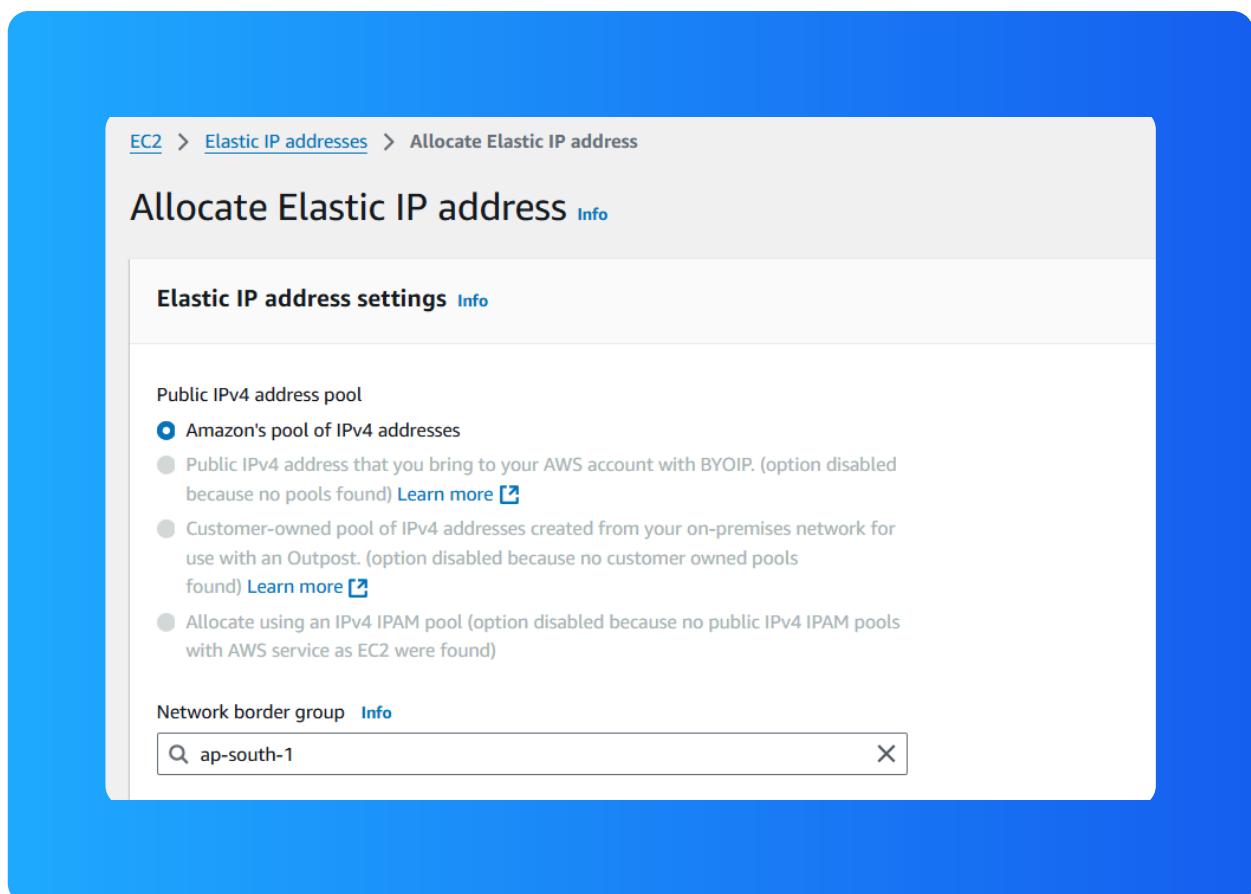
I was stopped from using EC2 Instance Connect as the instance did not have a public IPv4 address. In order for EC2 Instance Connect to work, the EC2 instance must have a public IPv4 address and be in a public subnet.



Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IP addresses are static public IPv4 addresses that we can request for our AWS account and then delegate to specific resources.

Associating an Elastic IP address resolved the error because it assigned the EC2 instance with a public address, fulfilling the requirements for Instance Connect to work.

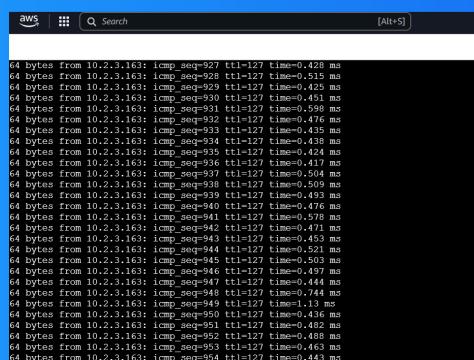


Troubleshooting ping issues

To test VPC peering, I ran the command 'ping 10.2.3.163'. Here, '10.2.3.163' is the private IPv4 address of the other instance which is in VPC 2.

A successful ping test would validate my VPC peering connection because the destination instance in the ping command would respond to the message sent by an instance if the peering connection between the two VPCs was successful.

I had to update my second EC2 instance's security group because it was not letting in ICMP traffic - which is the traffic type of a ping message. I added a new rule that allows ICMP traffic coming in from any resource in VPC 2.



```
aws | [ ] Search [Alt+S]
64 bytes from 10.2.3.163: icmp_seq=927 ttl=127 time=0.428 ms
64 bytes from 10.2.3.163: icmp_seq=928 ttl=127 time=0.519 ms
64 bytes from 10.2.3.163: icmp_seq=929 ttl=127 time=0.451 ms
64 bytes from 10.2.3.163: icmp_seq=930 ttl=127 time=0.459 ms
64 bytes from 10.2.3.163: icmp_seq=931 ttl=127 time=0.598 ms
64 bytes from 10.2.3.163: icmp_seq=932 ttl=127 time=0.476 ms
64 bytes from 10.2.3.163: icmp_seq=933 ttl=127 time=0.486 ms
64 bytes from 10.2.3.163: icmp_seq=934 ttl=127 time=0.438 ms
64 bytes from 10.2.3.163: icmp_seq=935 ttl=127 time=0.424 ms
64 bytes from 10.2.3.163: icmp_seq=936 ttl=127 time=0.417 ms
64 bytes from 10.2.3.163: icmp_seq=937 ttl=127 time=0.504 ms
64 bytes from 10.2.3.163: icmp_seq=938 ttl=127 time=0.493 ms
64 bytes from 10.2.3.163: icmp_seq=939 ttl=127 time=0.493 ms
64 bytes from 10.2.3.163: icmp_seq=940 ttl=127 time=0.493 ms
64 bytes from 10.2.3.163: icmp_seq=941 ttl=127 time=0.476 ms
64 bytes from 10.2.3.163: icmp_seq=942 ttl=127 time=0.578 ms
64 bytes from 10.2.3.163: icmp_seq=943 ttl=127 time=0.452 ms
64 bytes from 10.2.3.163: icmp_seq=944 ttl=127 time=0.521 ms
64 bytes from 10.2.3.163: icmp_seq=945 ttl=127 time=0.503 ms
64 bytes from 10.2.3.163: icmp_seq=946 ttl=127 time=0.497 ms
64 bytes from 10.2.3.163: icmp_seq=947 ttl=127 time=0.495 ms
64 bytes from 10.2.3.163: icmp_seq=948 ttl=127 time=0.744 ms
64 bytes from 10.2.3.163: icmp_seq=949 ttl=127 time=0.13 ms
64 bytes from 10.2.3.163: icmp_seq=950 ttl=127 time=0.436 ms
64 bytes from 10.2.3.163: icmp_seq=951 ttl=127 time=0.486 ms
64 bytes from 10.2.3.163: icmp_seq=952 ttl=127 time=0.486 ms
64 bytes from 10.2.3.163: icmp_seq=953 ttl=127 time=0.463 ms
64 bytes from 10.2.3.163: icmp_seq=954 ttl=127 time=0.443 ms
```



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for
more projects

