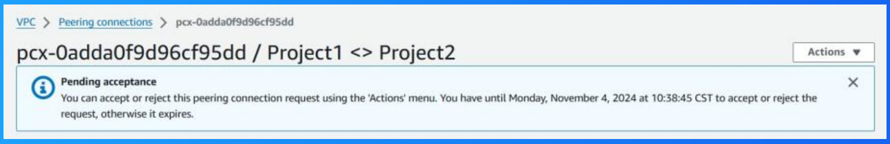
VPC Peering

Maulik Patel



pcx-0adda0f9d96cf95dd / Project1 <> Project2

Pending acceptance

You can accept or reject this peering connection request using the 'Actions' menu. You have until Monday, November 4, 2024 at 10:38:45 CST to accept or reject the request, otherwise it expires.

Maulik Patel

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is AWS's foundational networking service that lets us create our own network, control traffic flow and security and oraganize our resources into public/private subents!

How I used Amazon VPC in this project

I used Amazon VPC to set up a multi-VPC architecture, Create a peering connection between the and update security group rules to run a successful connectivity test to validate VPC peering set up.

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 didn't expect that Elastic IPs cna assign static public IPv4 addresses to resources.

This project took me...

This project took me 1.5 Hours!

In the first part of my project...

Step 1 - Set up my VPC

In this step, we leverage the VPC launch wizard to rapidly configure two VPCs along with their necessary components.

Step 2 - Create a Peering Connection

I am setting up a VPC Peering Connection, a component that directly links two VPCs, allowing seamless communication between them.

Step 3 - Update Route Tables

We are updating the route tables for VPC 1 and VPC 2 to route traffic through the established peering connection, ensuring seamless communication between the two networks.

Step 4 - Launch EC2 Instances

We are launching an EC2 instance in each VPC (VPC 1 and VPC 2) to facilitate direct connections for testing the VPC peering connection later on.

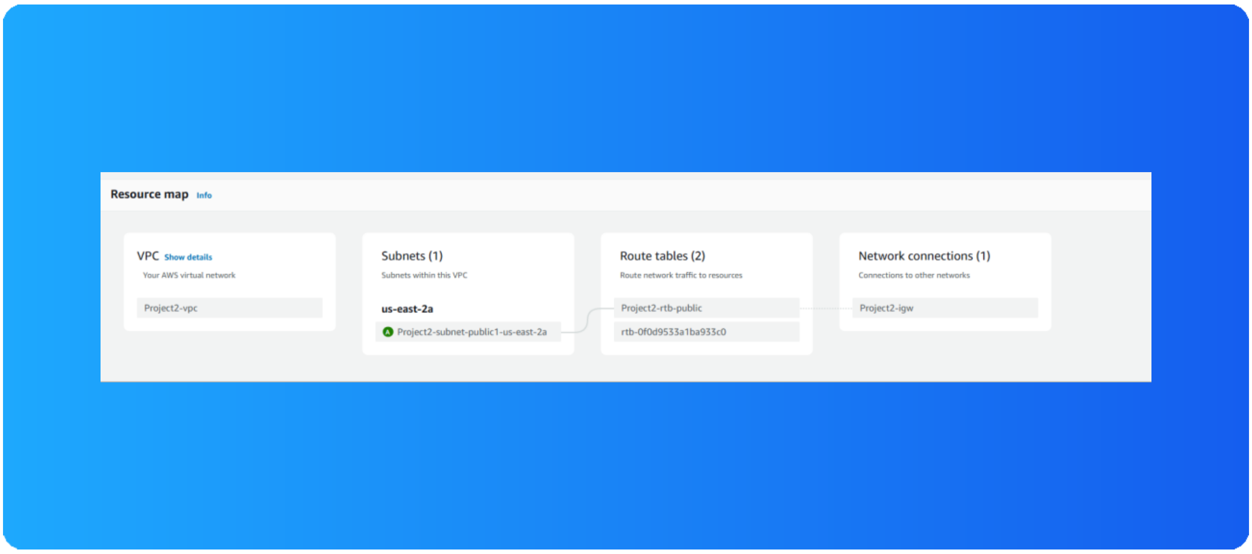
Multi-VPC Architecture

To start my project, I launched two VPCs, each with a unique CIDR block and a public subnet, setting up a foundational network structure.

The CIDR blocks for VPC 1 and VPC 2 are 10.0.0.0/16 and 10.1.0.0/16. These blocks must be unique to enable VPC peering, as route tables require distinct address ranges for routing traffic between VPCs effectively.

I also launched 2 EC2 instances

I chose not to set up key pairs to simplify the instance access process, allowing for easier management and connection without the need for SSH keys.



Resource map Info

VPC Show details
Your AWS virtual network
Project2-vpc

Subnets (1)
Subnets within this VPC
us-east-2a
Project2-subnet-public1-us-east-2a

Route tables (2)
Route network traffic to resources
Project2-rtb-public
rtb-0f0d9533a1ba933c0

Network connections (1)
Connections to other networks
Project2-igw

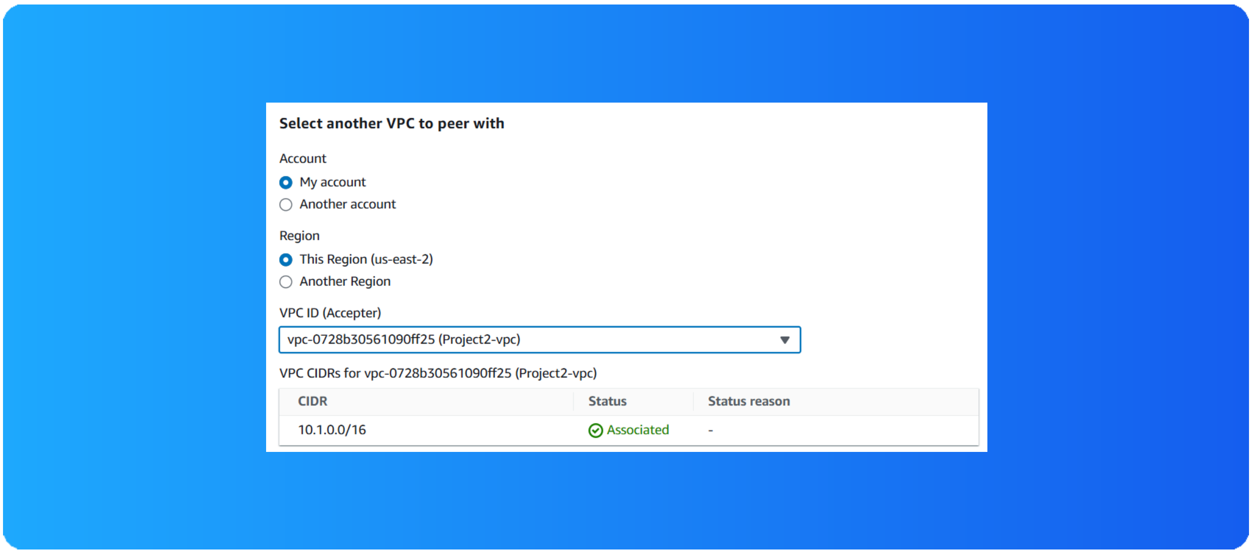
Maulik Patel

VPC Peering

A VPC peering connection allows two Virtual Private Clouds (VPCs) to connect directly for secure communication and resource sharing, requiring unique CIDR blocks for effective routing.

Peering connections enable secure, direct communication between Virtual Private Clouds (VPCs), allowing for efficient resource sharing and collaboration without using the public internet.

In a VPC peering connection, the Requester is the VPC that initiates the peering request, seeking to establish a direct connection with another VPC. The Accepter is the VPC that receives and approves this request, thereby forming the connection.



Select another VPC to peer with

Account

● My account

○ Another account

Region

● This Region (us-east-2)

○ Another Region

VPC ID (Accepter)

vpc-0728b30561090ff25 (Project2-vpc) ▼

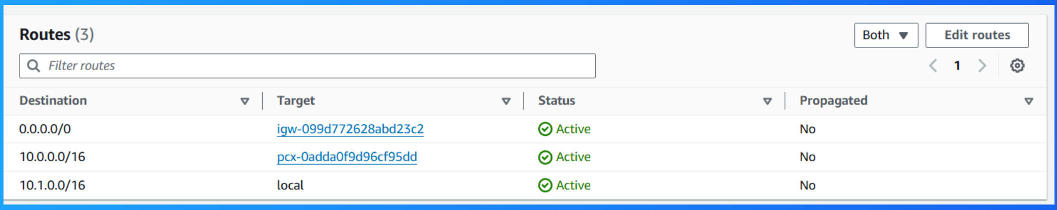
VPC CIDRs for vpc-0728b30561090ff25 (Project2-vpc)

CIDR	Status	Status reason
10.1.0.0/16	✓ Associated	-

Updating route tables

I added a new route to the VPCs' route tables to enable traffic to flow between the peered VPCs. This routing update ensures that instances in one VPC can communicate with instances in the other VPC by directing the traffic.

The new route added to the VPCs' route tables directs traffic destined for the CIDR block of the peered VPC through the VPC peering connection.



Routes (3)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-099d772628abd23c2	✓ Active	No
10.0.0.0/16	pcx-0adda0f9d96cf95dd	✓ Active	No
10.1.0.0/16	local	✓ Active	No

In the second part of my project...

Step 5 - Use EC2 Instance Connect

I am using EC2 Instance Connect to directly access our first EC2 instance for connectivity tests to validate our VPC peering setup later in the project, ensuring seamless communication between the VPCs.

Step 6 - Connect to EC2 Instance 1

I am reattempting our connection to Instance - Project1 VPC, and resolving error preventing me from using EC2 Instance Connect to directly connect with the instance.

Step 7 - Test VPC Peering

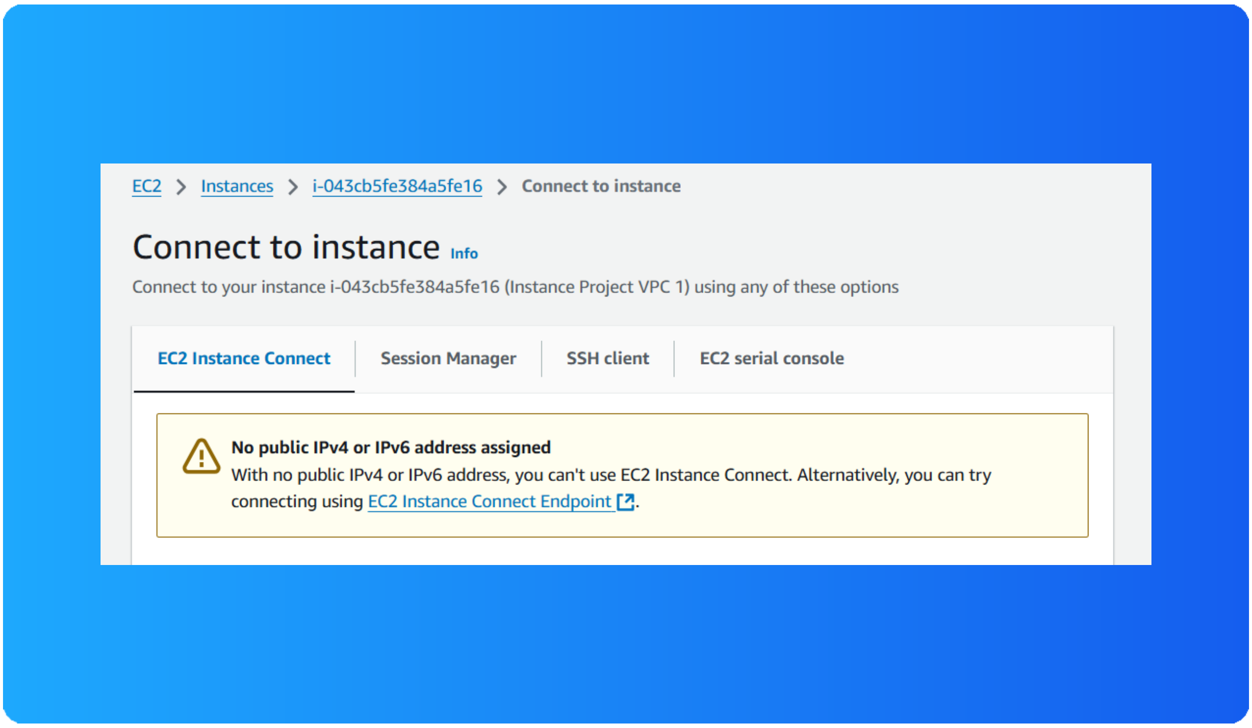
I am going to use Instance Project VPC 1 to attempt a direct connection with Instance Project VPC 2 so that we can validate that our peering connection is set up properly.

Maulik Patel

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to directly connect with Instance - Project1 - vpc just by using the AWS Management Console.

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



Connect to instance

Connect to your instance i-043cb5fe384a5fe16 (Instance Project VPC 1) using any of these options

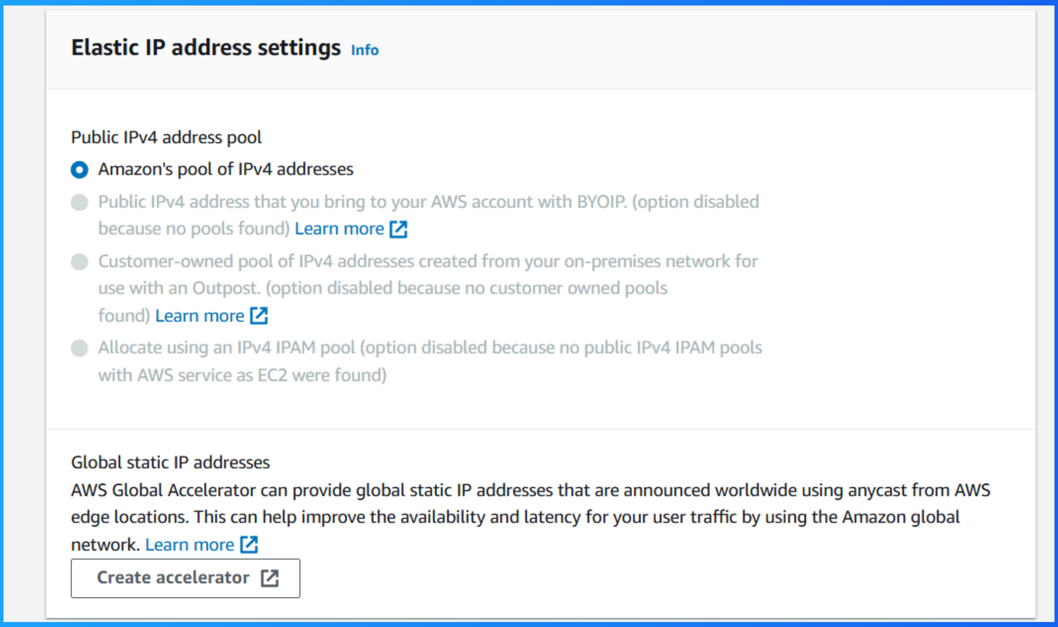
No public IPv4 or IPv6 address assigned

With no public IPv4 or IPv6 address, you can't use EC2 Instance Connect. Alternatively, you can try connecting using EC2 Instance Connect Endpoint.

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 AWS account, and then delegate to specific resources.

Associating an Elastic IP address resolved the error because it gives our EC2 instance a public IP address, fulfilling the requirments for Instance Connect to work!



Elastic IP address settings Info

Public IPv4 address pool

● Amazon's pool of IPv4 addresses

○ Public IPv4 address that you bring to your AWS account with BYOIP. (option disabled because no pools found) Learn more

○ Customer-owned pool of IPv4 addresses created from your on-premises network for use with an Outpost. (option disabled because no customer owned pools found) Learn more

○ Allocate using an IPv4 IPAM pool (option disabled because no public IPv4 IPAM pools with AWS service as EC2 were found)

Global static IP addresses

AWS Global Accelerator can provide global static IP addresses that are announced worldwide using anycast from AWS edge locations. This can help improve the availability and latency for your user traffic by using the Amazon global network. Learn more

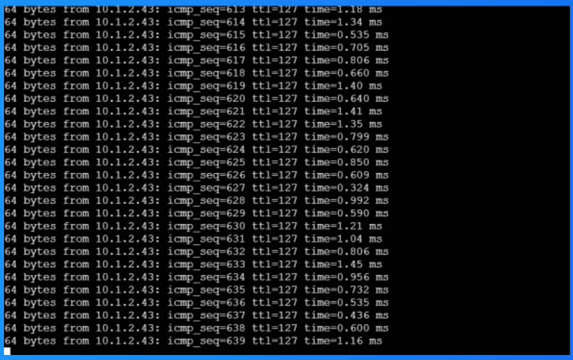
Maulik Patel

Troubleshooting ping issues

To test VPC peering, I ran the command Ping 10.0.x.x i.e. the private IPv4 address of the other EC2 insatance in VPC 2.

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 our two VPCs.

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.



```
64 bytes from 10.1.2.43: icmp_seq=613 ttl=127 time=1.18 ms
64 bytes from 10.1.2.43: icmp_seq=614 ttl=127 time=1.34 ms
64 bytes from 10.1.2.43: icmp_seq=615 ttl=127 time=0.535 ms
64 bytes from 10.1.2.43: icmp_seq=616 ttl=127 time=0.705 ms
64 bytes from 10.1.2.43: icmp_seq=617 ttl=127 time=0.806 ms
64 bytes from 10.1.2.43: icmp_seq=618 ttl=127 time=0.660 ms
64 bytes from 10.1.2.43: icmp_seq=619 ttl=127 time=1.40 ms
64 bytes from 10.1.2.43: icmp_seq=620 ttl=127 time=0.640 ms
64 bytes from 10.1.2.43: icmp_seq=621 ttl=127 time=1.41 ms
64 bytes from 10.1.2.43: icmp_seq=622 ttl=127 time=1.35 ms
64 bytes from 10.1.2.43: icmp_seq=623 ttl=127 time=0.799 ms
64 bytes from 10.1.2.43: icmp_seq=624 ttl=127 time=0.620 ms
64 bytes from 10.1.2.43: icmp_seq=625 ttl=127 time=0.850 ms
64 bytes from 10.1.2.43: icmp_seq=626 ttl=127 time=0.609 ms
64 bytes from 10.1.2.43: icmp_seq=627 ttl=127 time=0.324 ms
64 bytes from 10.1.2.43: icmp_seq=628 ttl=127 time=0.992 ms
64 bytes from 10.1.2.43: icmp_seq=629 ttl=127 time=0.590 ms
64 bytes from 10.1.2.43: icmp_seq=630 ttl=127 time=1.21 ms
64 bytes from 10.1.2.43: icmp_seq=631 ttl=127 time=1.04 ms
64 bytes from 10.1.2.43: icmp_seq=632 ttl=127 time=0.806 ms
64 bytes from 10.1.2.43: icmp_seq=633 ttl=127 time=1.45 ms
64 bytes from 10.1.2.43: icmp_seq=634 ttl=127 time=0.956 ms
64 bytes from 10.1.2.43: icmp_seq=635 ttl=127 time=0.732 ms
64 bytes from 10.1.2.43: icmp_seq=636 ttl=127 time=0.535 ms
64 bytes from 10.1.2.43: icmp_seq=637 ttl=127 time=0.436 ms
64 bytes from 10.1.2.43: icmp_seq=638 ttl=127 time=0.600 ms
64 bytes from 10.1.2.43: icmp_seq=639 ttl=127 time=1.16 ms
```