



nextwork.org

VPC Peering



Sean Calderon

A VPC peering connection `pcx-0576145674c353107` / VPC 1 <> VPC 2 has been requested.

`pcx-0576145674c353107` / VPC 1 <> VPC 2

Actions

Pending acceptance

You can accept or reject this peering connection request using the 'Actions' menu. You have until Thursday, August 21, 2025 at 20:17:41 GMT+8 to accept or reject the request, otherwise it expires.



Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is a wonderful service to create an easily scalable network infrastructure in the cloud. It is useful for engineers if the business requirement requires having different VPCs for peering or simply peering to a friend's VPC, isn't it wonderful?

How I used Amazon VPC in this project

In this project, I used Amazon VPC to setup scalable network infrastructure that allows me to peer two different VPCs using VPC peering connection.

This project took me...

I took this project approximately 50 minutes. I find it challenging for me to troubleshoot instance connection problems when the issue is only the default security group? That, only I didn't expect to be the problem and overlooked. It was rewarding to see when my two instances that reside on different VPC are able to talk to each other.



In the first part of my project...

Step 1 - Set up my VPC

In this step, I will create two VPCs from scratch with VPC wizard for super fast deploying of VPC within a few seconds.

Step 2 - Create a Peering Connection

In this step, I have two non-overlapping blocks of VPCs and I will deploy a VPC peering connection to link both VPCs.

Step 3 - Update Route Tables

In this step, I am going to set up a way for traffic coming from VPC 1 to get to VPC 2. On the other end, I am going to set up a way for traffic coming from VPC 2 to get to VPC 1. This way, the traffic will be allowed going back and forth.

Step 4 - Launch EC2 Instances

In this step, I am going to launch an EC2 instance in each VPC, so I can use them to test reachability between my VPCs with VPC peering connection.



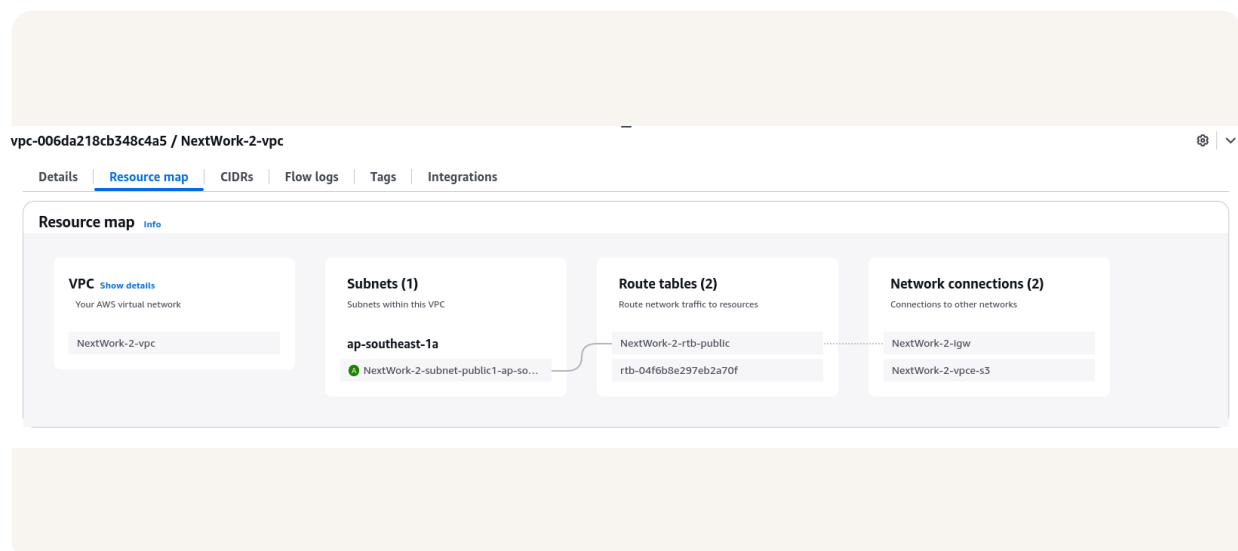
Multi-VPC Architecture

I started my project by launching two VPCs using VPC wizard and configured the IPv4 CIDR block to be non-overlapping of CIDR block of 10.1.0.0/16 and 10.2.0.0/16. Subsequently, I created only one public subnet per VPC.

The CIDR blocks for VPCs 1 and 2 are non-overlapping. They have to be unique because VPC peering is impossible since traffic is only to be routed within the VPC itself if other VPC has the same CIDR block causing conflicts and connectivity issues.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances as I am going to stick on dedicated EC2 instance connect.





VPC Peering

A VPC peering connection is a direct connection between two VPCs! A peering connection lets VPCs and their resources route traffic between them using their private IP addresses and without going out of the internet.

VPC peering connections eliminates traffic going out to the internet and use dedicated route instead to route traffic directly to the destination VPC, which reduces latency.

The difference between a Requester and an Acceptor in a peering connection is Requester is the one the initiates connection and invites to be peer while Acceptor is the one that accepts an invitation of peering connection.

The screenshot shows the AWS Management Console interface for creating a VPC peering connection. The page is titled 'Create peering connection' and is located under the 'VPC' service. It is divided into several sections:

- Select a local VPC to peer with:** This section contains a dropdown menu for 'VPC ID (Requester)' with the value 'vpc-027aa2cc4db28586b (NextWork-1-vpc)'. Below this is a table for 'VPC CIDRs for vpc-027aa2cc4db28586b (NextWork-1-vpc)' with one entry: CIDR '10.1.0.0/16', Status 'Associated', and Status reason '-'. The status is indicated by a green checkmark.
- Select another VPC to peer with:** This section has radio buttons for 'My account' (selected) and 'Another account'. Below are radio buttons for 'This Region (ap-southeast-1)' (selected) and 'Another Region'. It includes a dropdown for 'VPC ID (Acceptor)' with the value 'vpc-006da218db348c4a5 (NextWork-2-vpc)'. Below this is a table for 'VPC CIDRs for vpc-006da218db348c4a5 (NextWork-2-vpc)' with one entry: CIDR '10.2.0.0/16', Status 'Associated', and Status reason '-'. The status is indicated by a green checkmark.
- Tags:** A section explaining that a tag is a label assigned to an AWS resource. It includes a 'Key' field with a dropdown for 'Name' and a 'Value - optional' field with a dropdown for 'VPC 1 <-> VPC 2'. There are 'Add new tag' and 'Remove' buttons. A note at the bottom states 'You can add up to 50 tags.'

At the bottom right of the form, there are 'Cancel' and 'Create peering connection' buttons.



Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because there is still no instruction in the route table where to route traffic destined to each VPC's CIDR block.

My VPCs' new routes have a destination of 10.1.0.0/16 for my NextWork-2 route table and 10.2.0.0/16 for my NextWork-1 route table. The routes' target was of course the peering connection I made.

rtb-0ee9b56de3b4511e7 / NextWork-2-rtb-public

Actions

Details

Route table ID
rtb-0ee9b56de3b4511e7

VPC
vpc-006da218cb548c4a5 | NextWork-2-vpc

Main

No

Owner ID
146624863424

Explicit subnet associations

subnet-034a2772a9a7ea1e2 / NextWork-2-subnet-public1-ap-southeast-1a

Edge associations

-

Routes

Subnet associations

Edge associations

Route propagation

Tags

Routes (3)

Filter routes

Both

Edit routes

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0273e8717a886ed9e	Active	No	Create Route
10.1.0.0/16	pcx-0576145674c353107	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 use EC2 Instance Connect to connect to my first EC2 instance.

Step 6 - Connect to EC2 Instance 1

In this step, I will use EC2 Instance Connect to connect to Instance 1 once again.

Step 7 - Test VPC Peering

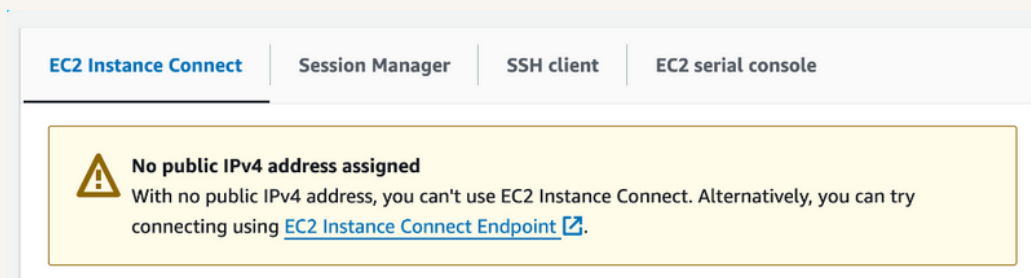
In this step, I will have my Instance 1 to send test messages to Instance 2.



Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to connect to my instance.

I was stopped from using EC2 Instance Connect as the instance does not have its public IP address assigned yet.





Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IP addresses are fixed allocation of IP address attached to the instance.

Associating an Elastic IP address resolved the error because it is impossible for an instance to be accessible over the internet if its only accessible inside the VPC subnet since it has only private IP address.

Allocate Elastic IP address info

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)

Network border group info

Q ap-southeast-1 X

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](#)

[Create accelerator](#)

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key

Q NextWork Elastic IP 1 X

Value - optional

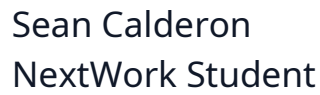
Q Enter value

[Remove](#)

[Add new tag](#)

You can add up to 49 more tag

[Cancel](#) [Allocate](#)



A successful ping test would validate my VPC peering connection because ping test actually verify connectivity going to the destination and to the destination coming back to the source.

I had to update my second EC2 instance's security group because the default only allows inbound for VPC only so I added a new rule that accepts traffic coming to the peered VPC.

```
/ #_
~\ ##### Amazon Linux 2023
~~ \#####\
~~      \####\
~~       \|#|_____ https://aws.amazon.com/linux/amazon-linux-2023
~~        V-' |'=>
~~~~~
~~~~~
~~~~~
~~~~~
~/m/'

[ec2-user@ip-10-1-0-235 ~]$ ping 10.2.8.139
PING 10.2.8.139 (10.2.8.139) 56(84) bytes of data:
64 bytes from 10.2.8.139: icmp_seq=1 ttl=127 time=0.747 ms
64 bytes from 10.2.8.139: icmp_seq=2 ttl=127 time=0.196 ms
64 bytes from 10.2.8.139: icmp_seq=3 ttl=127 time=0.196 ms
64 bytes from 10.2.8.139: icmp_seq=4 ttl=127 time=0.203 ms
64 bytes from 10.2.8.139: icmp_seq=5 ttl=127 time=0.208 ms
64 bytes from 10.2.8.139: icmp_seq=6 ttl=127 time=0.238 ms
64 bytes from 10.2.8.139: icmp_seq=7 ttl=127 time=0.219 ms
^C
--- 10.2.8.139 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6238ms
rtt min/avg/max/mdev = 0.196/0.286/0.747/0.188 ms
[ec2-user@ip-10-1-0-235 ~]$
```



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects

