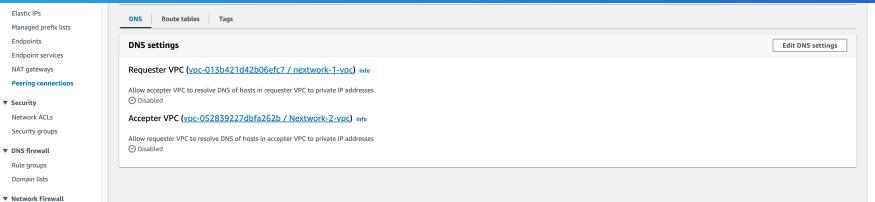




VPC Peering

 alexten3517@gmail.com



The screenshot shows the AWS VPC Peering Connections console. On the left, a sidebar lists various VPC management options: Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections, Security (Network ACLs, Security groups), DNS Firewall (Rule groups, Domain lists), and Network Firewall. The 'Peering connections' option is selected. The main pane displays the 'DNS' tab of a peering connection configuration. It shows two sections: 'Requester VPC' (vpc-013bd421d42b06efc / nextwork-1-vpc) and 'Acceptor VPC' (vpc-05289227dbfa262b / Nextwork-2-vpc). Under each section, there is a checkbox labeled 'Allow accepter VPC to resolve DNS of hosts in requester VPC to private IP addresses'. Both checkboxes are currently set to 'Disabled'. There is also an 'Edit DNS settings' button at the top right of the DNS settings box.

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is AWS's foundational networking service that lets us create our own networks, control traffic flow and security and organize our resources into public / private subnets.

How I used Amazon VPC in this project

We used Amazon VPC to set up a multi-VPC architecture (we set up two VPC's), create a peering connection between them and update security group rules to run a successful connectivity test to validate our VPC setup.

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

I did not expect to troubleshoot our peering connection by evaluating our subnet, network ACL and our security groups to see exactly where the bottleneck was which was holding back our connection.

This project took me...

I spent a total of 2 hours learning, evaluating and creating this project.

In the first part of my project...

Step 1 - Set up my VPC

In this step , we are using the VPC resource map/launch wizard to create two VPCs and their components in just minutes.

Step 2 - Create a Peering Connection

In this step we are setting up a VPC Peering Connection, which is a VPC component designed to directly connect two VPC's together.

Step 3 - Update Route Tables

In this step we have are creating the component that allows a connection(peering connection) between VPC1 and VPC2 to communicate with each other.

Step 4 - Launch EC2 Instances

We are launching an EC2 insstance in each of the VPC's so that we can directly connect with an EC2 our instances later and test our vpc peering connection.

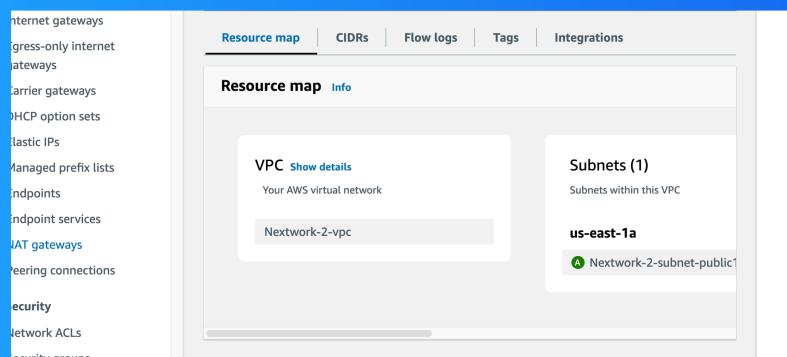
Multi-VPC Architecture

I started my project by launching two VPC's. They each have a unique CIDR block and they each have 1 public subnet.

The CIDR blocks for VPC's 1 and 2 are 10.1.0.0/16 and the second one is 10.2.0.0/16 to avoid overlapping of ip addresses. Once you setup a VPC peering connection, route tables need unique addresses for correct routing accross VPC's.

I also launched 2 EC2 instances

We did not choose to create a set up key pairs because we are using EC2 instance connect to directly connect with our EC2 instance later in this project, which handles key pair creation and management for us.



VPC Peering

A peering connection would be when two VPC's are able to communicate with each other privately without the use of the internet. This connection is one that allows for data transfer between both VPC's in a more secure and robust way.

VPCs would use peering connections to allow the communication to held privately. No longer would we need to send data through the public internet to communicate with another VPC - with a peering connection we can connect VPC 1 to VPC 2 privately.

The difference between a Requester and an Acceptor in a peering connection is VPC 1 requesting permission to connect to VPC 2.

Select another VPC to peer with

Account
 My account
 Another account

Region
 This Region (us-east-1)
 Another Region

VPC ID (Acceptor)
vpc-052839227dbfa262b (Nextwork-2-vpc)

VPC CIDRs for vpc-052839227dbfa262b (Nextwork-2-vpc)

CIDR	Status	Status reason
10.2.0.0/16	Associated	-

Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because only public traffic was accepted and we needed a way to have our routing table set up to create effective communication with our VPC2.

The new route that was added was a route that will allow for intercommunication between VPC's in a more direct way without the any other traffic interference.

Routes (3)				
Destination	Target	Status	Propagated	
0.0.0.0/0	igw-068cc0bb...	Active	No	
10.10.0.0/16	local	Active	No	
10.2.0.0/16	pclx-056a3f802...	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 directly with our first EC2 instance. We need to use our EC2 instance for connectivity test later in this project.

Step 6 - Connect to EC2 Instance 1

We are reattempting our connection to Instance VPC 1, and resolving another error preventing us from using EC2 Instance connect to directly connect with the Instance.

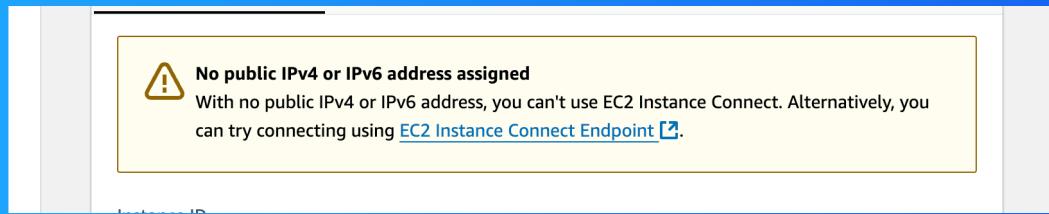
Step 7 - Test VPC Peering

In this step we are going to create a connectivity test to see if we are able to pass our test which will give us a direct connection from VPC1 to VPC2.

Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to directly connect with Instance - VPC1 just by using the AWS Management console.

I was stopped from using EC2 Instance Connect as our instance did not have 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.

Associating an Elastic IP address resolved the error because it gives our EC2 instance a public IP address. Fulfilling all the requirements for instance connect to work.



Troubleshooting ping issues

To test VPC peering, I ran a ping test to see if communication was successful. Using ping and the private ip address to get a connection is what was needed in getting a peering connection.

A successful ping test would validate my VPC peering connection because we were able to get a response from VPC 2 which means communication between VPC's 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 it from any resource in VPC.

```
last login: Sat Nov 23 01:27:17 2024 from 18.206.107.29
ssh-user@ip-10-1-6-173: ~ % ping 1.6.173
PING 1.6.173(1.6.173) 56(84) bytes of data.
4 bytes from 10.1.6.173: icmp_seq=1 ttl=127 time=0.017 ms
4 bytes from 10.1.6.173: icmp_seq=2 ttl=127 time=0.030 ms
4 bytes from 10.1.6.173: icmp_seq=3 ttl=127 time=0.025 ms
4 bytes from 10.1.6.173: icmp_seq=4 ttl=127 time=0.028 ms
4 bytes from 10.1.6.173: icmp_seq=5 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=6 ttl=127 time=0.029 ms
4 bytes from 10.1.6.173: icmp_seq=7 ttl=127 time=0.026 ms
4 bytes from 10.1.6.173: icmp_seq=8 ttl=127 time=0.026 ms
4 bytes from 10.1.6.173: icmp_seq=9 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=10 ttl=127 time=0.026 ms
4 bytes from 10.1.6.173: icmp_seq=11 ttl=127 time=0.028 ms
4 bytes from 10.1.6.173: icmp_seq=12 ttl=127 time=0.025 ms
4 bytes from 10.1.6.173: icmp_seq=13 ttl=127 time=0.025 ms
4 bytes from 10.1.6.173: icmp_seq=14 ttl=127 time=0.030 ms
4 bytes from 10.1.6.173: icmp_seq=15 ttl=127 time=0.029 ms
4 bytes from 10.1.6.173: icmp_seq=16 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=17 ttl=127 time=0.025 ms
4 bytes from 10.1.6.173: icmp_seq=18 ttl=127 time=0.028 ms
4 bytes from 10.1.6.173: icmp_seq=19 ttl=127 time=0.026 ms
4 bytes from 10.1.6.173: icmp_seq=20 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=21 ttl=127 time=0.025 ms
4 bytes from 10.1.6.173: icmp_seq=22 ttl=127 time=0.032 ms
4 bytes from 10.1.6.173: icmp_seq=23 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=24 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=25 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=26 ttl=127 time=0.028 ms
4 bytes from 10.1.6.173: icmp_seq=27 ttl=127 time=0.026 ms
4 bytes from 10.1.6.173: icmp_seq=28 ttl=127 time=0.027 ms
4 bytes from 10.1.6.173: icmp_seq=29 ttl=127 time=0.027 ms
```



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for
more projects

