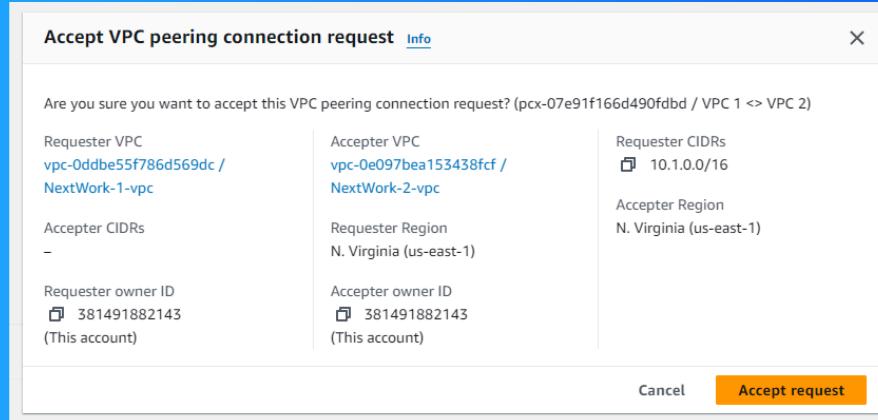




VPC Peering



Sai Prasanth Reddy





Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is a service that allows you to create isolated cloud networks within AWS. It's useful because it provides control over network configurations, enhances security, and allows you to manage resources within a secure, scalable environment.

How I used Amazon VPC in this project

I used Amazon VPC in today's project to create two VPCs, set up public subnets, established a VPC peering connection, configured route tables for cross-VPC communication, and tested connectivity between the VPCs using ping and other tools.

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

One thing I didn't expect in this project was the need to assign Elastic IP addresses to my instances for remote access, as disabling auto-assign IP prevented me from connecting using EC2 Instance Connect initially.

This project took me...

This project took me little over an hour, including setting up the VPCs, configuring subnets, establishing the peering connection, updating security groups, and testing the connectivity between the instances.



In the first part of my project...

Step 1 - Set up my VPC

In this step, we're creating two VPCs from scratch! We'll use the visual VPC resource map to quickly set up the VPCs, defining their IP ranges and subnets, so we can later connect them using VPC peering for communication between both networks.

Step 2 - Create a Peering Connection

In this step, we're setting up a connection link between the two VPCs using VPC peering. This will allow the VPCs to communicate with each other, enabling resource sharing and network traffic flow between them.

Step 3 - Update Route Tables

In this step, we're setting up routing so traffic from VPC 1 can reach VPC 2, and vice versa. We'll update the route tables in both VPCs to ensure traffic flows smoothly between them via the peering connection.

Step 4 - Launch EC2 Instances

Launching an EC2 instance in each VPC, so we can use them to test our VPC peering connection later.



Multi-VPC Architecture

I started my project by launching two VPCs, both with public subnets using the "VPC and more" option. This setup ensures easy and quick configuration of the VPCs, allowing us to proceed with VPC peering and connectivity testing.

The CIDR blocks for VPCs 1 and 2 are unique. They have to be unique because overlapping IP ranges would cause routing conflicts, making it impossible for the two VPCs to communicate with each other through VPC peering.

I also launched 2 EC2 instances

I didn't set up key pairs for these EC2 instances as with EC2 Instance Connect, AWS actually manages a key pair for us! We don't need to manage key pairs ourselves, making the connection process simpler and more secure.

The screenshot shows the AWS VPC Details page for the 'NextWork-2-vpc' VPC. The top navigation bar includes 'VPCs', 'Your VPCs', 'vpc-0e097bea153438fcf / NextWork-2-vpc', and an 'Actions' dropdown. The main content area is divided into several sections:

- Details**:
 - VPC ID: vpc-0e097bea153438fcf
 - Tenancy: Default
 - Default VPC: No
 - Network Address Usage metrics: Disabled
 - State: Available
 - DHCP option set: depk-03a24b4e0557d0d
 - IPv4 CIDR: 10.20.0/16
 - Route 53 Resolver DNS Firewall rule groups: -
 - DNS hostnames: Enabled
 - Main route table: rtb-022f168a90ba1764
 - IPv6 pool: -
 - Owner ID: 381491882145
 - DNS resolution: Enabled
 - Main network ACL: ad-0615d2a207e3042c
 - IPv6 CDR (Network border group): -
- Resource map**:
 - VPC: Your AWS virtual network (NextWork-2-vpc)
 - Subnets (1): Subnets within this VPC (us-east-1a: NextWork-2-subnet-public1-us-east-1a)
 - Route tables (2): Route network traffic to resources (NextWork-2-rtb-public: rtb-022f168a90ba1764; NextWork-2-igw: rtb-022f168a90ba1764)
 - Network connections (1): Connections to other networks (NextWork-2-igw)



VPC Peering

A VPC peering connection is a networking link between two VPCs that allows them to route traffic to each other privately, without using the internet. It enables secure communication between resources in different VPCs.

VPCs would use peering connections to enable secure, private communication between resources in different VPCs without routing traffic through the internet, allowing efficient collaboration and data sharing across isolated networks.

The difference between a Requester and an Acceptor in a peering connection is that the Requester initiates the peering request between two VPCs, while the Acceptor is the VPC that receives and approves the connection request, establishing the link.

The screenshot shows the AWS VPC Peering Connection configuration interface. At the top, it displays the CIDR range for the requester's VPC: 10.1.0.0/16, which is associated. Below this, there is a section titled "Select another VPC to peer with". Under "Account", the "My account" option is selected. Under "Region", the "This Region (us-east-1)" option is selected. In the "VPC ID (Acceptor)" field, the value "vpc-0e097bea153438fcf (NextWork-2-vpc)" is entered. At the bottom, it shows the CIDR range for the acceptor's VPC: 10.2.0.0/16, which is also associated.



Updating route tables

After accepting a peering connection, my VPCs' route tables need to be updated because traffic between the two VPCs requires explicit routes, allowing each VPC to direct traffic through the peering connection to reach the other VPC.

My VPCs' new routes have a destination of the CIDR block of the opposite VPC. The routes' target was the VPC peering connection, enabling traffic between the two VPCs.

The screenshot shows the AWS Route Tables interface for a specific route table. At the top, a green banner indicates "Updated routes for rtb-0014a9fa448dbea36 / NextWork-2-rtb-public successfully". The main page displays the route table ID (rtb-0014a9fa448dbea36) and its association with the VPC (vpc-0e097bea153438cf). The "Details" tab is selected, showing the route table's configuration: Main (No), Owner ID (381491882143), and Explicit subnet associations (subnet-0a57d5ff169b685d3 / NextWork-2-subnet-public1-us-east-1a). Edge associations are listed as "-". Below this, the "Routes" section shows three explicit routes:

Destination	Target	Status	Propagated
0.0.0.0/0	igw-08b8bcd638ca1e2e6	Active	No
10.1.0.0/16	pcx-07e91f166d490fdbd	Active	No
10.2.0.0/16	local	Active	No



In the second part of my project...

Step 5 - Use EC2 Instance Connect

Using EC2 Instance Connect to connect to our first EC2 instance and fix a connection error.

Step 6 - Connect to EC2 Instance 1

Using EC2 Instance Connect to connect to Instance 1 and fix another error.

Step 7 - Test VPC Peering

Getting Instance 1 to send test messages to Instance 2 and to solve connection errors until Instance 2 is able to send messages back.



Troubleshooting Instance Connect

Next, I used EC2 Instance Connect to securely access my EC2 instances without needing to manage SSH key pairs, allowing me to quickly test and configure my instances through a browser-based connection.

I was stopped from using EC2 Instance Connect because auto-assign IP was disabled, and therefore there was no public IP for the instance, preventing me from connecting to it remotely.

The screenshot shows the EC2 Instance Connect service interface. At the top, there are tabs: EC2 Instance Connect, Session Manager, SSH client, and EC2 serial console. Below the tabs, there are two error messages in yellow boxes:

- EC2 Instance Connect service IP addresses are not authorized**
Port 22 (SSH) is authorized in [your security group](#). However, to use EC2 Instance Connect, it is recommended to also authorize port 22 for the EC2 Instance Connect service IP addresses in your Region: 18.206.107.24/29. [Learn more](#).
- No public IPv4 address assigned**
With no public IPv4 address, you can't use EC2 Instance Connect. Alternatively, you can try connecting using [EC2 Instance Connect Endpoint](#).

Below the error messages, there are fields for Instance ID (i-0d376cd8ee5f81f80) and Connection Type. The 'Connect using EC2 Instance Connect' option is selected, with a note: "Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address." There is also an unselected option for "Connect using EC2 Instance Connect Endpoint".

Further down, there is a field for Public IP address, which is currently empty. Below that is a Username field containing "ec2-user". A note states: "Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user." Finally, there is a note at the bottom: "Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username."



Elastic IP addresses

To resolve this error, I set up Elastic IP addresses. Elastic IPs are static, public IPv4 addresses in AWS that give instances a fixed public IP for remote access, ensuring continuity even if the instance is stopped or restarted.

Associating an Elastic IP address resolved the error because it provided the instance with a public IP, enabling remote access through EC2 Instance Connect, which requires a publicly reachable IP address for connectivity.

The screenshot shows the 'Allocate Elastic IP address' configuration page. It includes sections for 'Elastic IP address settings' (choosing 'Amazon's pool of IPv4 addresses'), 'Network border group' (set to 'us-east-1'), and 'Global static IP addresses' (with a note about AWS Global Accelerator and a 'Create accelerator' button).



Troubleshooting ping issues

To test VPC peering, I ran the command `ping 10.2.9.137`, which checked the connectivity between instances in VPC 1 and VPC 2, confirming that the peering connection was working properly.

A successful ping test would validate my VPC peering connection because it shows that traffic can flow between the two VPCs, confirming that the route tables and peering setup are correctly allowing communication between the instances.

I had to update my second EC2 instance's security group because it wasn't allowing traffic from the first VPC. I added a new rule that permits inbound ICMP (ping) traffic from the private IP range of the first VPC to enable successful communication.

```
aws [Services] Q Search [Alt+S]
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

ec2-user@ip-10-1-6-39: ~ % ping 10.2.9.137
PING 10.2.9.137 (10.2.9.137) 56(84) bytes of data.
bytes from 10.2.9.137: icmp_seq=76 ttl=127 time=1.09 ms
bytes from 10.2.9.137: icmp_seq=77 ttl=127 time=1.09 ms
bytes from 10.2.9.137: icmp_seq=78 ttl=127 time=1.09 ms
bytes from 10.2.9.137: icmp_seq=79 ttl=127 time=1.10 ms
bytes from 10.2.9.137: icmp_seq=80 ttl=127 time=1.10 ms
bytes from 10.2.9.137: icmp_seq=81 ttl=127 time=0.986 ms
bytes from 10.2.9.137: icmp_seq=82 ttl=127 time=1.00 ms
bytes from 10.2.9.137: icmp_seq=83 ttl=127 time=1.26 ms
bytes from 10.2.9.137: icmp_seq=84 ttl=127 time=1.52 ms
bytes from 10.2.9.137: icmp_seq=85 ttl=127 time=1.44 ms
bytes from 10.2.9.137: icmp_seq=86 ttl=127 time=1.44 ms
bytes from 10.2.9.137: icmp_seq=87 ttl=127 time=1.31 ms
bytes from 10.2.9.137: icmp_seq=88 ttl=127 time=1.67 ms
```



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for
more projects

