

VPC PEERING

➤ SET UP YOUR VPCs

Amazon VPC allows you to launch and manage resources such as EC2 Instances and other services.

- [Log in to your AWS Account.](#)
- Head to your **VPC** console - search for VPC at the search bar at top of your page.
- From the left hand navigation bar, select **Your VPCs**.
- Select **Create VPC**.
- Select **VPC and more**.

Create VPC 1

- Under **Name tag auto-generation**, enter NextWork-1
- The VPC's **IPv4 CIDR block** is already pre-filled to 10.0.0.0/16 - change that to 10.1.0.0/16
- For **IPv6 CIDR block**, we'll leave in the default option of **No IPv6 CIDR block**.
- For **Tenancy**, we'll keep the selection of **Default**.
- For **Number of Availability Zones (AZs)**, we'll use just **1** Availability Zone.
- Make sure the **Number of public subnets** chosen is **1**.
- For **Number of private subnets**, we'll keep thing simple and go with **0** private subnets.

The screenshot shows the AWS VPC console configuration interface. It has three main sections, each with a title, an 'Info' link, a descriptive paragraph, and a selection of radio buttons.

- Number of Availability Zones (AZs)** Info: "Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability." The radio buttons for 1, 2, and 3 are shown, with '1' selected.
- Number of public subnets** Info: "The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet." The radio buttons for 0 and 1 are shown, with '1' selected.
- Number of private subnets** Info: "The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access." The radio buttons for 0, 1, and 2 are shown, with '0' selected.

Below each section is a link to "Customize" the respective settings.

- Next, for the **NAT gateways (\$)** option, make sure you've selected **None**. As the dollar sign suggests, NAT gateways cost money!

NAT (Network Address Translation) gateways let instances in **private** subnets access the internet for updates and patches, while blocking inbound traffic.

- Next, for the **VPC endpoints** option, select **None**.
VPC endpoints let you connect your VPC privately to AWS services without using the public internet.
- You can leave the **DNS options** checked.
- Select **Create VPC**.

Set up VPC 2

We're going to set up another VPC with *slightly* different settings

- Select **Create VPC**.
- Select **VPC and more**.
- Under **Name tag auto-generation**, enter NextWork-2
- The VPC's **IPv4 CIDR block** should be unique! Make sure the CIDR block is NOT 10.1.0.0/16 - it should be 10.2.0.0/16
- For **IPv6 CIDR block**, we'll leave in the default option of **No IPv6 CIDR block**.
- For **Tenancy**, we'll keep the selection of **Default**.
- For **Number of Availability Zones (AZs)**, we'll use just **1** Availability Zone.
- Make sure the **Number of public subnets** chosen is **1**.
- For **Number of private subnets**, we'll go with **0** for today's project. Let's keep it simple with just a single subnet!
- For the **NAT gateways (\$)** option, select **None**.
- For the **VPC endpoints** option, select **None**.
- You can leave the **DNS options** checked.
- Select **Create VPC**.

➤ CREATE A PEERING CONNECTION

. Now that you have two VPCs ready to go, let's bridge them together with a peering connection by setting up a connection link between the two VPCs.

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. This means data can now be transferred between VPCs without going through the public internet. Without a peering connection, data transfers between VPCs would use resources' public address - meaning VPCs have to communicate over the public internet

- Still in the VPC console, click on **Peering connections** on the left hand navigation panel.
- Click on **Create peering connection** in the right hand corner.
- Name your **Peering connection name** as VPC 1 < > VPC 2
- Select **NextWork-1-VPC** for your **VPC ID (Requester)**.
In VPC peering, the **Requester** is the VPC that initiates a peering connection. As the requester, they will be sending the other VPC an **invitation** to connect!
- Under **Select another VPC to peer with**, make sure **My Account** is selected. VPC peering can occur between VPCs in different AWS accounts.
- For **Region**, select **This Region**.

- For **VPC ID (Accepter)**, select **NextWork-2-VPC**
In VPC peering, the **Accepter** is the VPC that receives a peering connection request! The Accepter can either accept or decline the invitation. This means the peering connection isn't actually made until the other VPC also agrees to it.

Select another VPC to peer with

Account

☒ My account

☐ Another account

Region

☒ This Region (us-east-1)

☐ Another Region

VPC ID (Accepter)

vpc-0d8a6f967536186cf (Nextwork-2-vpc) ▼

VPC CIDRs for vpc-0d8a6f967536186cf (Nextwork-2-vpc)

CIDR	Status	Status reason
10.2.0.0/16	✓ Associated	-

- Click on **Create peering connection**.
Your newly created peering connection isn't finished yet. The green success bar says the peering connection **has been requested**.
- On the next screen, select **Actions** and then select **Accept request**.

Accept VPC peering connection request [Info](#) ✕

Are you sure you want to accept this VPC peering connection request? (pcx-01b556f5275a2775f / VPC 1 <> VPC 2)

<p>Requester VPC</p> <p>vpc-0c5b756e17fc4a022 / Nextwork-1-vpc</p>	<p>Accepter VPC</p> <p>vpc-0d8a6f967536186cf / Nextwork-2-vpc</p>	<p>Requester CIDRs</p> <p>10.1.0.0/16</p>
<p>Accepter CIDRs</p> <p>-</p>	<p>Requester Region</p> <p>N. Virginia (us-east-1)</p>	<p>Accepter Region</p> <p>N. Virginia (us-east-1)</p>
<p>Requester owner ID</p> <p>831926586806 (This account)</p>	<p>Accepter owner ID</p> <p>831926586806 (This account)</p>	

Cancel **Accept request**

- Click on **Accept request** again on the pop up panel.
- Click on **Modify my route tables now** on the top right corner.

➤ UPDATE ROUTE TABLES

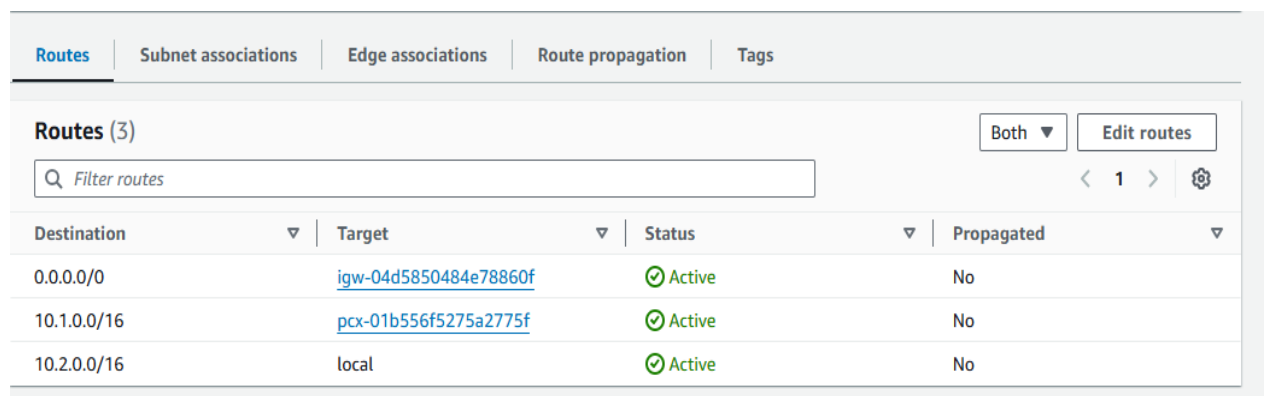
Even if your peering connection has been accepted, traffic in VPC 1 won't know how to get to resources in VPC 2 without a route in your route table! You need to set up a route that directs traffic bound for VPC 2 to the peering connection you've set up.

Update VPC 1's route table

- Select the checkbox next to VPC 1's route table i.e. called **NextWork-1-rtb-public**.
- Scroll down and click on the **Routes** tab.
- Click **Edit routes**.
- Let's add a new route
- Add a new route to **VPC 2** by entering the CIDR block 10.2.0.0/16 as our **Destination**.
- Under Target, select **Peering Connection**.
- Select **VPC 1 < > VPC 2**.
- Click **Save changes**.
- Confirm that the new route appears in VPC 1's **Routes** tab

Update VPC 2's route table

- The route table you're updating is **NextWork-2-rtb-public**.
- The **Destination** is the CIDR block 10.1.0.0/16



Routes	Subnet associations	Edge associations	Route propagation	Tags
Routes (3)				
<input type="text" value="Filter routes"/>				
Both Edit routes				
< 1 > ⚙️				
Destination	Target	Status	Propagated	
0.0.0.0/0	igw-04d5850484e78860f	✓ Active	No	
10.1.0.0/16	pcx-01b556f5275a2775f	✓ Active	No	
10.2.0.0/16	local	✓ Active	No	

➤ LAUNCH EC2 INSTANCE

In this step, you will be launching an EC2 Instance in each VPC so you can use them to test your VPC peering connection.

Launch an instance in VPC 1

- Head to the **EC2 console** - search for EC2 in the search bar at the top of screen.
- Select **Instances** at the left hand navigation bar.
- Select **Launch instances**.
- Since your first EC2 instance will be launched in your first VPC, let's name it Instance - NextWork VPC 1
- For the **Amazon Machine Image**, select **Amazon Linux 2023 AMI**.
- For the **Instance type**, select **t2.micro**.
- For the **Key pair (login)** panel, select **Proceed without a key pair (not recommended)**.
- At the **Network settings** panel, select **Edit** at the right hand corner.

By default, all resources are launched into the default VPC that AWS has set up for your account. We need to tell AWS that we actually want to launch this instance in **NextWork-vpc-1**

- Under **VPC**, select **NextWork-vpc-1**.
- Under **Subnet**, select your VPC's public subnet.
- Keep the **Auto-assign public IP** setting to **Disable**.
- For the **Firewall (security groups)** setting, Amazon VPC already created a security group for your VPC - let's use that
- Choose **Select existing security group**.
- Select the **default** security group for your VPC.
- Select **Launch instance**.

Launch an instance in VPC 2

- The **Name** is Instance - NextWork VPC 2
- The **VPC** is **NextWork-vpc-2**

➤ CONNECT TO INSTANCE 1

To test our VPC peering connection, we'll need to get one of our EC2 instances to try talk to the other.

- Still in your **EC2** console, select the checkbox next to **Instance - NextWork VPC 1**.
- Select **Connect**.



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](#).

Check out the error message: "**No public IPv4 address assigned. With no public IPv4 address, you can't use EC2 Instance Connect.** Keeping **Disable** for the **Auto-assign IP address** option in our EC2 instance's network settings caused this error. If you want to connect to your instance over EC2 Instance Connect, then your instance must have a public IP address and be in a public subnet. This is because using EC2 Instance Connect connects to your server **over the internet** by default.

- Verify this by heading back to the **Instances** page in your EC2 console, and checking the Public IPv4 address field, you will notice it is empty.
- On your EC2 console's left hand navigation panel, select **Elastic IPs**.

Elastic IPs are **static** IPv4 addresses that get allocated to your AWS account, and is yours to delegate to an EC2 instance. Having an Elastic IP is like having a permanent address in a city, instead of having to move from location to location every time your instance restarts. Elastic IPs are also a great way to assign an instance a public IPv4 address after launching it. That's how we're using it in this step.

- Select **Allocate Elastic IP** addresses.
- Leave all default options.

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](#)

us-east-1

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

The default setting is **Amazon's pool of IPv4 addresses**. Think of this as a public pool of all IPv4 addresses that Amazon has saved to allocate to AWS accounts, so you won't need to find one yourself.

- Select **Allocate**.
- Refresh your page, then select the new IP address you've set up.
- Select the **Actions** dropdown, then select **Associate Elastic IP address**.
- Under **Instance**, select **Instance - NextWork VPC 1**.
- Click **Associate**.
- Awesome! Your EC2 Instance should have a public IP address now.
- To check this, select **Instances** from the left hand navigation panel.
- Select the checkbox next to **Instance - NextWork VPC 1**.
- Do you see a **Public IPv4 address** for your EC2 instance now?
- Select **Connect**.
- In the EC2 Instance Connect set up page, select **Connect** again.
- Oh no! We've failed to connect to our instance?!
- Head back to your **VPC console**.
- Select **Subnets** from the left hand navigation panel.
- Select the checkbox next to **NextWork-1-subnet-public1...**
- Investigate the **Route table** and **Network ACL** tabs - what do you see?

Everything looks good! Our Network ACL allows all traffic in and out, and our route table is correctly setting up a route to the Internet Gateway.

- Copy the **VPC ID** of **NextWork-1-vpc**.
- Head into the **Security groups** page from the left hand navigation panel.
- To find the VPC 1's default security group, let's use the handy search bar.
- Click into the search bar, and paste the ID you copied into the search bar. Make sure there aren't any empty spaces before your text.
- Select the filter!
- Select the checkbox next to VPC 1's **default** security group.
- Select the **Inbound rules** tab.
- In the **Inbound rules** tab, select **Edit inbound rules**.
- Select **Add rule**.
- For your new rule, configure the **Type** as **SSH**.
- Then, under **Source type**, select **Anywhere-IPv4**.
- Select **Save rules**.
- With that modified, refresh your EC2 console's **Instances** page.
- Select your **Instance-NextWork VPC 1** and select **Connect** again.
- Select **Connect** in the EC2 Instance Connect setup page.
- Phew! Success.

➤ **TEST VPC PEERING**

- Leave open the **EC2 Instance Connect** tab, but head back to your **EC2** console in a new tab.
- Select **Instance - NextWork VPC 2**.
- Copy Instance - NextWork VPC 2's **Private IPv4 address**.
- Switch back to the **EC2 Instance Connect** tab.
- Run ping [the Private IPv4 address you just copied] in the terminal.
 - Your final result should look similar to something like **ping 10.0.1.227**
 - Don't know where to enter this prompt? Look for the \$ sign at the bottom line of the black window, and type in your command after the \$ sign.

Ping is a common computer network tool used to check whether your computer can communicate with another computer or device on a network. Think of it like sending a tiny message that says "hello, are you there?" to another computer. When you "ping" a specific IP address, your server (in this case, Instance - NextWork VPC 1) sends a small packet of data to the target server (Instance - NextWork VPC 2), asking for a response. Ping will tell you whether you get a response back and how long it took to get a response. If you receive a response quickly, it means the connection between your computer and the other computer is good. If it takes a long time or you get no response, there might be a problem with the connection.

- You should see a response similar to this
- To resolve this connectivity error, let's investigate whether **Instance - NextWork VPC 2** is allowing inbound ICMP traffic.
- Leave open the **EC2 Instance Connect** tab, but head back to your **VPC** console in a new tab.
- In the VPC console, select the **Subnets** page.
- Select VPC 2's subnet i.e. **NextWork-2-subnet-public1-...**
- Let's investigate the **Route tables** and **Network ACL** tabs for your public subnet.

- The network ACL allows all types of inbound traffic from anywhere! So this looks perfectly fine.
- Before we finish, let's check the security groups!
- Copy the **VPC ID** of VPC 2.
- Select **Security groups** from the left hand navigation panel.
- Paste the **VPC ID** in the search bar, and select the suggested filter.
- Check your security group's **Inbound rules** tab - does this security group allow ICMP traffic from sources outside of VPC 2? (No, it does not)

Let's fix this by letting inbound ICMP traffic from VPC 1.

- Select **Edit inbound rules**.
- Select **Add new rule**.
- Change the **Type** to **All ICMP - IPv4**.
When you set a rule for **All ICMP - IPv4**, you're allowing **all** types of ICMP messages for IPv4 addresses. This covers a wide range of operational messages that are essential for diagnosing network connectivity issues, ping requests and responses are just one type of ICMP messages.
- Set the **Source** to traffic coming from VPC 1 - 10.1.0.0/16
- Select **Save rules**.

All updated!

- Revisit the **EC2 Instance Connect** tab that's connected to Instance - NextWork VPC 1.
- Woah! Lots of new lines coming through in the terminal.
The multiple new lines appearing in your terminal are a sign of successful communication between the two EC2 instances. Each line represents a reply from the ping command you sent. This means that the ICMP (Internet Control Message Protocol) traffic is now successfully reaching Instance - NextWork VPC 2, thanks to the adjustments made in the network ACLs and Security Groups.

Congratulations!

You've set up a peering architecture that connects VPC 1 to VPC 2 AND validated it with ping.


```

64 bytes from 10.2.1.98: icmp_seq=375 ttl=127 time=0.832 ms
64 bytes from 10.2.1.98: icmp_seq=376 ttl=127 time=0.792 ms
64 bytes from 10.2.1.98: icmp_seq=377 ttl=127 time=0.844 ms
64 bytes from 10.2.1.98: icmp_seq=378 ttl=127 time=0.868 ms
64 bytes from 10.2.1.98: icmp_seq=379 ttl=127 time=0.582 ms
64 bytes from 10.2.1.98: icmp_seq=380 ttl=127 time=0.912 ms
64 bytes from 10.2.1.98: icmp_seq=381 ttl=127 time=0.760 ms
64 bytes from 10.2.1.98: icmp_seq=382 ttl=127 time=1.25 ms
64 bytes from 10.2.1.98: icmp_seq=383 ttl=127 time=0.718 ms
64 bytes from 10.2.1.98: icmp_seq=384 ttl=127 time=0.542 ms
64 bytes from 10.2.1.98: icmp_seq=385 ttl=127 time=0.756 ms
64 bytes from 10.2.1.98: icmp_seq=386 ttl=127 time=1.04 ms
64 bytes from 10.2.1.98: icmp_seq=387 ttl=127 time=0.890 ms
64 bytes from 10.2.1.98: icmp_seq=388 ttl=127 time=1.08 ms
64 bytes from 10.2.1.98: icmp_seq=389 ttl=127 time=0.992 ms
64 bytes from 10.2.1.98: icmp_seq=390 ttl=127 time=0.866 ms
64 bytes from 10.2.1.98: icmp_seq=391 ttl=127 time=0.850 ms
64 bytes from 10.2.1.98: icmp_seq=392 ttl=127 time=1.51 ms
64 bytes from 10.2.1.98: icmp_seq=393 ttl=127 time=0.769 ms
64 bytes from 10.2.1.98: icmp_seq=394 ttl=127 time=0.917 ms
64 bytes from 10.2.1.98: icmp_seq=395 ttl=127 time=0.997 ms
64 bytes from 10.2.1.98: icmp_seq=396 ttl=127 time=0.807 ms
64 bytes from 10.2.1.98: icmp_seq=397 ttl=127 time=0.658 ms

```

➤ DELETE YOUR RESOURCES

Delete your EC2 Instances

- Select **Elastic IPs** from the left hand navigation panel.
- Select the IP address you've created.
- Select Actions, then **Release Elastic IP addresses**
- Head back to the **Instances** page of your EC2 console.
- Select the checkboxes next to **Instance - NextWork VPC 1** and **Instance - NextWork VPC 2**.
- Select **Instance state**, then select **Terminate Instance**.
- Select **Terminate**.

Delete your Elastic IP address

- Select **Elastic IPs** from the left hand navigation panel.
- Select the IP address you've created.
- Select Actions, then **Release Elastic IP addresses**
When you release an Elastic IP address, that address returns to Amazon's pool of IPv4 address and will eventually get reallocated to another AWS user!
- Select **Release**.

Delete VPC Peering Connections

- Head back to your **VPC** console.
- Select **Peering connections** from your left hand navigation panel.
- Select the VPC 1 < > VPC 2 peering connection.
- Select **Actions**, then **Delete peering connection**.

- Select the checkbox to **Delete related route table entries**.
- Type delete in the text box and click **Delete**.

Delete your VPCs

- Select **Your VPCs** from your left hand navigation panel.
- Select **NextWork-1-vpc**, then **Actions**, and **Delete VPC**.
- Type delete in the text box and click **Delete**.
- If you get stopped from deleting your VPC because **network interfaces** are still attached to your VPC - delete all the attached network interfaces first

Network interfaces get created automatically when you launch an EC2 instance. Think of them as a component that attaches to an EC2 instance on one end and your VPC on another - so that your EC2 instance is connected to your network and can send and receive data! Network interfaces are usually deleted automatically with your EC2 instance, but on some occasions it'd be faster to delete them manually.

- Select **NextWork-2-vpc**, then **Actions**, and **Delete VPC**.
- Type delete in the text box and click **Delete**.

Other network components should be automatically deleted with your VPC, but it's always a good idea to check anyway:

1. Subnets
2. Route tables
3. Internet gateways
4. Network ACLs
5. Security groups

Don't forget to **refresh** each page before checking if the resources are still in your account.