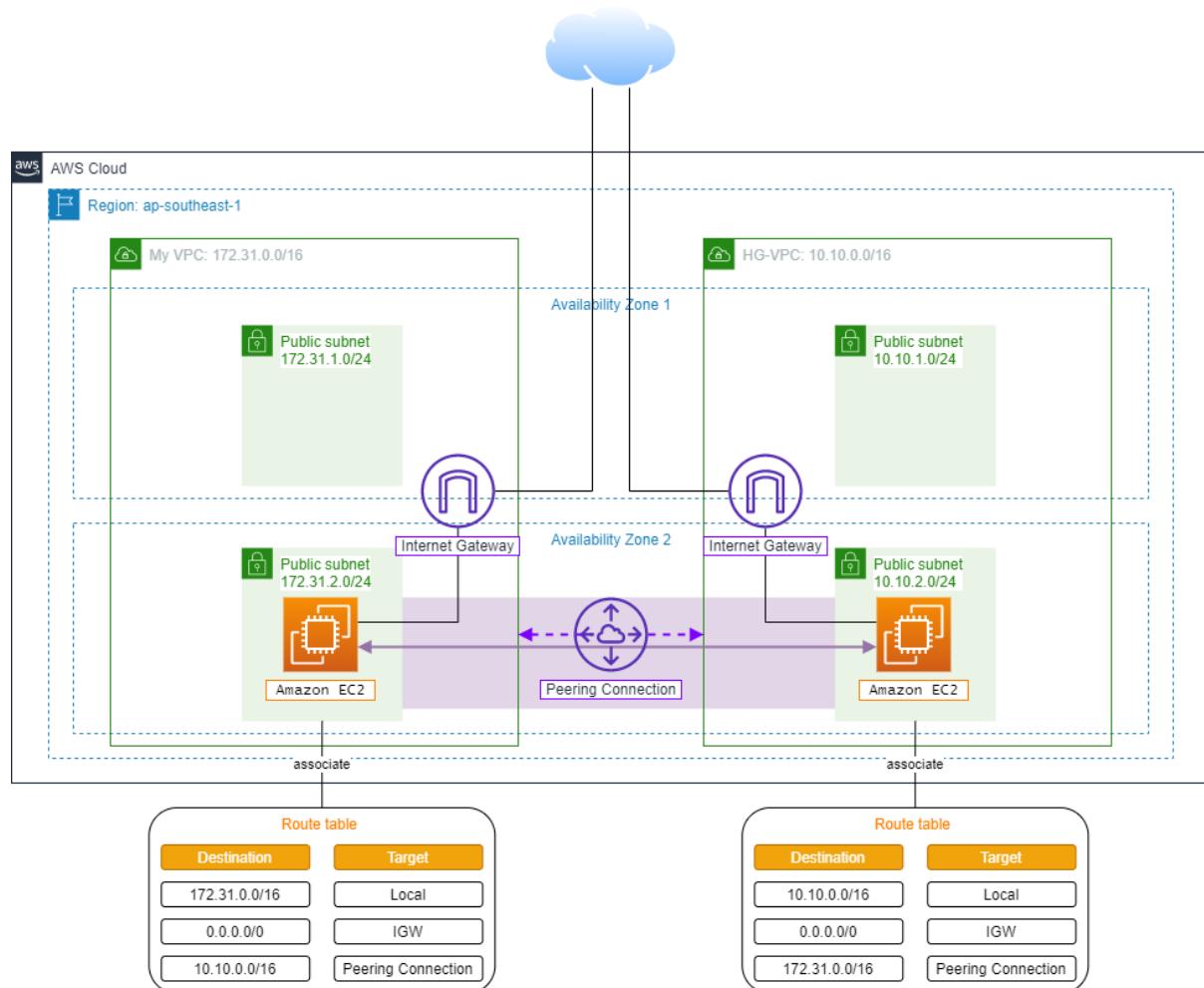


# VPC PEERING



VPC Peering is a networking feature in Amazon Web Services (AWS) that enables you to connect two Virtual Private Clouds (VPCs) together, allowing them to communicate as though they were in the same network.

- 🌐 It allows for private connectivity between the VPCs without the need for gateways, VPN, or separate physical infrastructure.
- 🌐 This can be useful in many scenarios, such as sharing resources across accounts, regions or deploying multi-tier applications across multiple VPCs.
- 🌐 The VPC Peering connection is established between two VPCs, where the peering connection is configured, by creating a route in both VPC route tables to allow traffic to flow between the two VPCs.
- 🌐 It should be noted that VPC Peering is only possible within the same region and between VPCs that have non-overlapping IP address ranges. To connect VPCs that have overlapping IP address ranges, you will need to use a different method, such as VPN or Direct Connect.



## Step 1: Create two VPCs

The first step of VPC peering is to create at least two VPCs. These VPCs should be in the same region but can be in different availability zones. You can create these VPCs using the AWS console or CLI.

CIDR (Classless Inter-Domain Routing) is a method used to allocate IP addresses and route traffic in a network. In the context of a VPC (Virtual Private Cloud) in AWS (Amazon Web Services), CIDR is used to define the IP address range that the VPC will use.

When creating a VPC in AWS, the user needs to specify a CIDR block that will be associated with the VPC. The CIDR block is a range of IP addresses expressed in CIDR notation, for example, 10.0.0.0/16. This means that the VPC will have a range of IP addresses from 10.0.0.0 to 10.0.255.255.

The CIDR block defines the maximum number of IP addresses that can be assigned to resources within the VPC. Once the CIDR block is defined, the user can create subnets within the VPC by dividing the CIDR block into smaller ranges. Each subnet can be associated with a different availability zone and can be used to launch instances or other resources.

CIDR notation also allows for efficient routing of traffic within the VPC and between the VPC and external networks. The VPC route table contains information on how to route traffic to and from different network addresses based on CIDR notation ranges. This allows for efficient use of network resources and optimal performance.

This screenshot shows the 'Create VPC' page in the AWS VPC console. The 'VPC settings' section is visible, with the 'Resources to create' dropdown set to 'VPC only'. A name tag 'vpc\_1' is entered in the 'Name tag - optional' field. Under 'IPv4 CIDR block', the value '10.0.0.0/16' is specified. An orange arrow points to the 'vpc\_1' input field.

This screenshot shows the 'Create VPC' page in the AWS VPC console, identical to the previous one but for VPC 2. The 'VPC settings' section shows 'Resources to create' set to 'VPC only', a name tag 'vpc\_2' entered in the 'Name tag - optional' field, and an IPv4 CIDR block of '11.0.0.0/16'. An orange arrow points to the 'vpc\_2' input field.

## Step 2: Route table creation

- 🌐 A route table in Amazon Web Services (AWS) is a key component of the networking functionality. A route table contains a set of rules called routes, which are used to determine where network traffic is directed.
- 🌐 When an instance in an AWS virtual private cloud (VPC) needs to send traffic to a destination, the VPC uses the route table associated with the subnet it's in to determine where to send the traffic.
- 🌐 Each route in a route table specifies a destination CIDR block and the target where traffic should be sent. Targets can be instances, gateways, or virtual private gateways.
- 🌐 Route tables can be associated with subnets, and a subnet can be associated with only one route table at a time. By default, each subnet in a VPC is associated with the main route table for the VPC, but you can create additional custom route tables and associate them with subnets as needed.

The screenshot shows the 'Create route table' wizard in the AWS VPC console. The 'Route table settings' section is active. A dropdown menu for 'VPC' lists three options: 'vpc-08e082744620260aa (vpc\_2)' (selected), 'vpc-08e082744620260aa (vpc\_2)' (disabled), and 'vpc-001166b7cdb1b8ca0 (default)'. Below the dropdown, there is a 'Key' field with a 'Value - optional' entry: 'Name' with value 'route-table-2-vpc-2'. A 'Create route table' button is at the bottom right.

### Step 3: Subnets

🌐 Subnet association in AWS refers to the process of associating a subnet with a particular resource or service. A subnet is a network partition that enables the segregation of resources within a virtual private cloud (VPC).

🌐 When creating resources within AWS, such as EC2 instances, a subnet can be associated with them to determine which IP addresses are reachable and to ensure that they operate within the defined network boundaries.

🌐 By associating a subnet with a resource, AWS ensures that it receives the correct IP configuration, and is assigned to the right availability zone, which enables easy access to other resources within the same subnet. Additionally, an AWS subnet can be linked to internet gateways (IGWs), virtual private gateways (VGWs), and other networking components to create and maintain a secure, scalable, and flexible network infrastructure.

The screenshot shows the 'Create subnet' wizard in the AWS VPC console. The 'VPC' section is active. A dropdown menu for 'VPC ID' lists three options: 'vpc-09fd955cc0e8d16b1 (vpc\_1)' (selected), 'vpc-08e082744620260aa (vpc\_2)' (disabled), and 'vpc-001166b7cdb1b8ca0 (default)'. Below the dropdown, there is a 'Subnet settings' section with a 'Subnet 1 of 1' entry. Under 'Subnet name', there is a field with the value 'my-subnet-01'. A 'Create subnet' button is at the bottom right.

CreateSubnet | VPC Console v vpc peering connection diagram x +

us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateSubnet:

aws Services Search [Alt+S]

**Subnet settings**  
Specify the CIDR blocks and Availability Zone for the subnet.

**Subnet 1 of 1**

**Subnet name**  
Create a tag with a key of 'Name' and a value that you specify.  
 The name can be up to 256 characters long.

**Availability Zone** [Info](#)  
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

**IPv4 VPC CIDR block** [Info](#)  
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

**IPv4 subnet CIDR block**  
 256 IPs  
< > ^ v

**Tags - optional**  
Key Value - optional  
  Remove  
Add new tag

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CreateSubnet | VPC Console v vpc peering connection diagram x +

us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateSubnet:

aws Services Search [Alt+S]

VPC > Subnets > Create subnet

**Create subnet** [Info](#)

**VPC**

**VPC ID**  
Create subnets in this VPC.  
   
 11.0.0.0/16  
 172.31.0.0/16 (default)  
 10.0.0.0/16

**Subnet settings**  
Specify the CIDR blocks and Availability Zone for the subnet.

**Subnet 1 of 1**

**Subnet name**  
Create a tag with a key of 'Name' and a value that you specify.  
 The name can be up to 256 characters long.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CreateSubnet | VPC Console v vpc peering connection diagram x +

us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateSubnet:

aws Services Search [Alt+S]

**Subnet 1 of 1**

**Subnet name**  
Create a tag with a key of 'Name' and a value that you specify.  
 The name can be up to 256 characters long.

**Availability Zone** [Info](#)  
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

**IPv4 VPC CIDR block** [Info](#)  
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

**IPv4 subnet CIDR block**  
 256 IPs  
< > ^ v

**Tags - optional**  
Key Value - optional  
  Remove  
Add new tag  
You can add 49 more tags.  
Remove

Add new subnet

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Step 4 : internet gateway

🌐 An internet gateway (IGW) in AWS is a horizontally scalable, redundant, and highly available virtual network component that serves as an essential entry and exit point between a Virtual Private Cloud (VPC) and the internet. It allows users to access the internet and provide a route for communication between the VPC and other resources or services outside the VPC, such as users, systems, applications, and third-party services.

🌐 An internet gateway plays a vital role in routing traffic between the VPC and the internet by providing a public IP address for the instances running in the VPC, allowing them to communicate with the internet. The IGW also performs network address translation (NAT), which enables instances in the VPC to access the internet by translating their private IP addresses to public IP addresses.

🌐 Additionally, an internet gateway is integrated with Amazon Route 53, which enables users to set up DNS servers for their domain name system (DNS) resolution.

🌐 Overall, an internet gateway is an essential component in creating a secure, scalable, and highly available environment in AWS, allowing users to access the internet and communicate with other resources or services outside their VPC.

The screenshot shows the AWS VPC Console interface. The top navigation bar includes tabs for 'InternetGateway | VPC Console' and 'vpc peering connection diagram'. The main content area displays a success message: 'The following internet gateway was created: igw-03c078697f8be86a0 - igw-1-vpc1. You can now attach to a VPC to enable the VPC to communicate with the internet.' Below this message is a green bar with the text 'Attach to a VPC'. The central panel shows the details of the newly created Internet Gateway, with the ID 'igw-03c078697f8be86a0' and the name 'igw-1-vpc1'. The 'Details' tab is selected, showing the Internet gateway ID, State (Detached), VPC ID (empty), and Owner (345119574351). The 'Tags' section lists a single tag: 'Name' with the value 'igw-1-vpc1'. On the left sidebar, the 'Internet gateways' section is expanded, showing options like 'Egress-only Internet gateways', 'Carrier gateways', 'DHCP option sets', 'Elastic IPs', 'Managed prefix lists', 'Endpoints', 'Endpoint services', 'NAT gateways', and 'Peering connections'. The bottom right corner features the AWS logo.

🌐 When an **Internet Gateway** is attached to a VPC, it provides a target for internet traffic destined for the instances in the VPC. The IGW acts as a gateway to the public internet, enabling instances to connect to services outside of the VPC and allowing inbound internet traffic to reach the instances inside the VPC. That's why it is essential to attach the Internet Gateway to a VPC in AWS.

The screenshot shows the AWS VPC console with the URL <https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#AttachInternetGateway:internetGatewayId=igw-03c078697f8be86a0>. The page title is "Attach internet gateway | VPC". A green banner at the top says "The following internet gateway was created: igw-03c078697f8be86a0 - igw-1-vpc1. You can now attach to a VPC to enable the VPC to communicate with the internet." Below this, there is a breadcrumb navigation: "VPC > Internet gateways > Attach to VPC (igw-03c078697f8be86a0)". The main content area is titled "Attach to VPC (igw-03c078697f8be86a0) Info". It contains a "VPC" section with the sub-instruction "Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.". Under "Available VPCs", there is a search bar with the placeholder "Attach the internet gateway to this VPC." and a dropdown menu showing three options: "vpc-09fd955cc0e8d16b1", "Use: 'vpc-09fd955cc0e8d16b1'", and "vpc-08e082744620260aa - vpc\_2". At the bottom right of this section is a yellow "Attach internet gateway" button. The bottom of the page includes standard AWS footer links: CloudShell, Feedback, © 2024 Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

🌐 Here, we will create another Internet Gateway (IGW) and attach it to the second VPC."

The screenshot shows the AWS VPC console with the URL <https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#AttachInternetGateway:internetGatewayId=igw-03cc01a5e969ba61c>. The page title is "Attach internet gateway | VPC". A green banner at the top says "The following internet gateway was created: igw-03cc01a5e969ba61c - igw-2-vpc-2. You can now attach to a VPC to enable the VPC to communicate with the internet." Below this, there is a breadcrumb navigation: "VPC > Internet gateways > Attach to VPC (igw-03cc01a5e969ba61c)". The main content area is titled "Attach to VPC (igw-03cc01a5e969ba61c) Info". It contains a "VPC" section with the sub-instruction "Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.". Under "Available VPCs", there is a search bar with the placeholder "Select a VPC" and a dropdown menu showing one option: "vpc-08e082744620260aa - vpc\_2". At the bottom right of this section is a yellow "Attach internet gateway" button. The bottom of the page includes standard AWS footer links: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

The screenshot shows the 'Create internet gateway' wizard in the AWS VPC console. The 'Internet gateway settings' section is active, displaying a 'Name tag' input field containing 'igw-2-vpc-2'. Below it, the 'Tags - optional' section shows a single tag 'Name: igw-2-vpc-2'. At the bottom right are 'Cancel' and 'Create internet gateway' buttons.

🌐 An instance in AWS refers to a virtual machine that can be used to run applications and services within the Amazon Web Services cloud computing platform. Instances can be launched on demand and can be configured with various configurations, including different operating systems, storage options, network settings, and security controls.

🌐 AWS provides a variety of instance types to suit different use cases, such as compute-intensive workloads, memory-intensive applications, and storage-intensive tasks. Customers can also customize instances with their own software and configurations to create a tailored environment that meets their specific needs.

The screenshot shows the 'Launch an instance' wizard in the AWS EC2 console. The 'Name and tags' section has 'instance-1-vpc-1' entered. The 'Application and OS Images (Amazon Machine Image)' section shows a search bar and a recent AMI selection. On the right, the 'Summary' section is expanded, showing 'Number of instances: 1', 'Software Image (AMI): Amazon Linux 2023 AMI 2023.4.2...', 'Virtual server type (instance type): t2.micro', 'Firewall (security group): New security group', and 'Storage (volumes): 1 volume(s) - 8 GiB'. At the bottom are 'Cancel', 'Launch instance', and 'Review commands' buttons.

🌐 In AWS, a **key pair** is a set of two security credentials: a public key and a private key. The public key is used to encrypt data, while the private key is used to decrypt it. Key pairs are commonly used for secure authentication.

🌐 For example, to login to an EC2 instance using SSH. In AWS, key pairs can be created and managed through the EC2 Management Console or through the AWS CLI. It is important to keep the private key secure and not share it with anyone.

The screenshot shows the AWS EC2 Management Console with a modal dialog titled "Create key pair". The "Key pair name" field contains "peering-key". The "Key pair type" section shows two options: "RSA" (selected) and "ED25519". Below this, "Private key file format" is set to ".ppk". A warning message in a yellow box states: "⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)". At the bottom right of the dialog are "Cancel" and "Create key pair" buttons.

The screenshot shows the AWS EC2 Management Console with a modal dialog titled "Launch instance". The "Number of instances" field is set to "1". The "Software Image (AMI)" section shows "Amazon Linux 2023 AMI 2023.4.2...read more ami-04e5276ebb8451442". The "Virtual server type (instance type)" is "t2.micro". The "Firewall (security group)" section shows "New security group". The "Storage (volumes)" section shows "1 volume(s) - 8 GiB". At the bottom right of the dialog are "Cancel", "Launch instance" (highlighted in orange), and "Review commands" buttons.

## 🌐 Here we will create a second instance for a second VPC

The screenshot shows the 'Launch an instance' wizard in the AWS EC2 console. The current step is 'Name and tags'. A single instance is selected. The 'Name' field contains 'instance-2-vpc-2'. The 'Software Image (AMI)' section shows 'Amazon Linux 2023 AMI 2023.4.2...'. The 'Virtual server type (instance type)' is set to 't2.micro'. The 'Summary' section indicates 1 instance. The 'Launch instance' button is highlighted in orange.

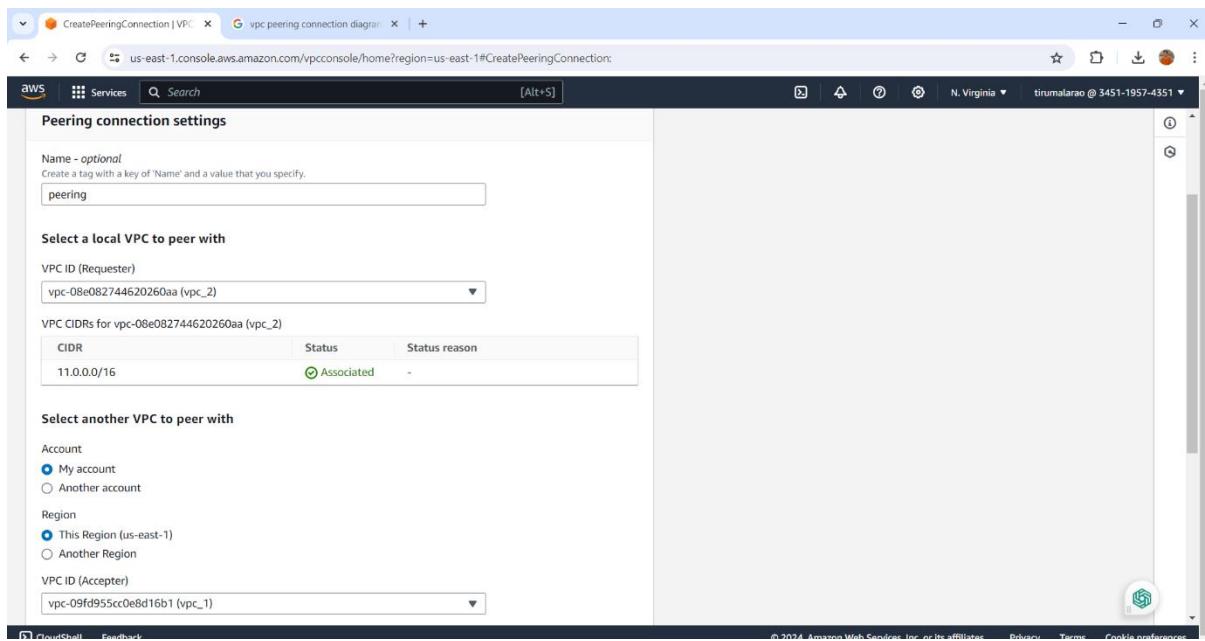
The screenshot shows the 'Launch an instance' wizard in the AWS EC2 console. The current step is 'Network settings'. A key pair named 'peerings-key' is selected. The 'VPC - required' dropdown shows 'vpc-08e082744620260aa (vpc\_2)'. The 'Subnet' dropdown shows 'subnet-2-vpc-2'. The 'Virtual server type (instance type)' is set to 't2.micro'. The 'Summary' section indicates 1 instance. The 'Launch instance' button is highlighted in orange.

## Step 5: Peering connection

In Amazon Web Services (AWS), a peering connection is a networking connection that allows two virtual private clouds (VPCs) to communicate with each other privately over the AWS network.

🌐 Peering connections can be established between two VPCs in the same AWS region or in different regions, as long as both VPCs are owned by the same AWS account.

🌐 Peering connections can improve the performance, security, and availability of applications that are deployed across different VPCs. They allow for faster data transfer and reduce the risk of data breaches since communication between VPCs is done over a private connection rather than the internet.



A **peering connection accepter** in AWS is an entity that approves or accepts a peering connection request. A peering connection is a direct network connection between two VPCs (Virtual Private Clouds) or between a VPC and an external network. When a peering connection request is initiated, it must be accepted by the owner of the requested VPC to establish the connection.

🌐 The accepter can be the owner of the requested VPC or a delegated AWS account with the appropriate permissions. Once the accepter approves the request, the peering connection is established, and resources can communicate with each other across the connection using private IP addresses.

🌐 In AWS, the peering connection accepter can approve the request using the AWS Management Console, the AWS CLI (Command Line Interface), or the AWS SDKs (Software Development Kits). The accepter must ensure that the VPC routing tables are properly configured to enable appropriate traffic flow across the peering connection.

**Editing routes** is essential for managing and maintaining AWS network configurations, ensuring that traffic flows efficiently and securely between different components.

🌐 Enable routing to different destinations including specific IP addresses or IP address ranges, as well as routing through virtual private gateways, VPN connections, and Direct Connect gateways.

🌐 Edit routes to redirect traffic in the event of an outage or to optimize network performance.

Establish custom routes for specific scenarios, such as sending traffic to specific internet gateways or peering connections.

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
route-table-1-vpc-1	rtb-056efab6f078b007c	-	-	No	vpc-09fd955cc0e8d16b
-	rtb-03be9aa1cf8df49a2	-	-	Yes	vpc-001166b7cdb1b8ca
-	rtb-052553bc90120f72b	-	-	Yes	vpc-09fd955cc0e8d16b
-	rtb-05b7d25c9701bae8c	-	-	Yes	vpc-08e082744620260
route-table-2-vpc-2	rtb-0899c7f284dc6c183	-	-	No	vpc-08e082744620260

rtb-056efab6f078b007c / route-table-1-vpc-1																					
Details	Routes	Subnet associations	Edge associations	Route propagation	Tags																
<table border="1"> <thead> <tr> <th colspan="2">Routes (1)</th> </tr> <tr> <th colspan="2">Filter routes</th> </tr> <tr> <th>Destination</th> <th>Target</th> <th>Status</th> <th>Propagated</th> <th></th> <th></th> </tr> </thead> <tbody> <tr> <td>10.0.0.0/16</td> <td>local</td> <td>Active</td> <td>No</td> <td></td> <td></td> </tr> </tbody> </table>						Routes (1)		Filter routes		Destination	Target	Status	Propagated			10.0.0.0/16	local	Active	No		
Routes (1)																					
Filter routes																					
Destination	Target	Status	Propagated																		
10.0.0.0/16	local	Active	No																		

EditRoutes | VPC Console vpcs | VPC Console us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#EditRoutes:RouteTableId=rtb-056efab6f078b007c

VPC > Route tables > rtb-056efab6f078b007c > Edit routes

### Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
Q 0.0.0.0/0	Internet Gateway	Active	No
Q 11.0.0.0/16	Peering Connection	-	No
	Q pcx-0904db7bc981ea828		

[Add route](#)

Cancel Preview Save changes

RouteTables | VPC Console vpcs | VPC Console us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#RouteTables:

VPC dashboard EC2 Global View Filter by VPC: Select a VPC Virtual private cloud Your VPCs Subnets Route tables Internet gateways Egress-only internet gateways Carrier gateways DHCP option sets Elastic IPs Managed prefix lists Endpoints Endpoint services NAT gateways Peering connections Security Network ACLs

### Route tables (1/5) info

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
route-table-1-vpc-1	rtb-056efab6f078b007c	-	-	No	vpc-09fd955cc0e8d16b
-	rtb-03be9aa1cf8df49a2	-	-	Yes	vpc-001166b7cdb1b8c
-	rtb-052553bc90120f72b	-	-	Yes	vpc-09fd955cc0e8d16b
-	rtb-05b7d25c9701bae8c	-	-	Yes	vpc-08e082744620260
route-table-2-vpc-2	rtb-0899c7f284dc6c183	-	-	No	vpc-08e082744620260

rtb-0899c7f284dc6c183 / route-table-2-vpc-2

Details Routes Subnet associations Edge associations Route propagation Tags

**Routes (1)**

Destination	Target	Status	Propagated
11.0.0.0/16	local	Active	No

Both Edit routes

EditRoutes | VPC Console vpcs | VPC Console us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#EditRoutes:RouteTableId=rtb-0899c7f284dc6c183

VPC > Route tables > rtb-0899c7f284dc6c183 > Edit routes

### Edit routes

Destination	Target	Status	Propagated
11.0.0.0/16	local	Active	No
Q 0.0.0.0/0	Internet Gateway	Active	No
Q 10.0.0.0/16	Peering Connection	-	No
	Q pcx-0904db7bc981ea828		

[Add route](#)

Cancel Preview Save changes

**Subnet association** In AWS, subnet association refers to the process of assigning a subnet to a specific Amazon VPC (Virtual Private Cloud) or network interface. This association allows instances in the VPC to communicate with resources outside the VPC or to connect with resources inside the VPC using a private IP address.

- 🌐 To associate a subnet with a VPC, you can use the Amazon VPC console or the AWS Command Line Interface (CLI). You can also associate subnets to network interfaces of instances using the console, CLI, or SDK.
- 🌐 When creating a VPC, you can create multiple subnets in different Availability Zones (AZs) to achieve high availability and fault tolerance. You can associate a subnet with a specific AZ to ensure that your instances are launched in the same AZ as the subnet.
- 🌐 In addition, you can use subnets to control traffic in and out of your VPC using Network Access Control Lists (NACLs) and Security Groups (SGs). NACLs provide a stateless, rule-based filter to control traffic at the subnet level, while SGs provide a stateful, instance-level firewall at the network interface level.
- 🌐 Overall, subnet association is a critical aspect of designing and deploying secure and scalable VPC architectures in AWS.

VPC > Route tables > rtb-056efab6f078b007c > Edit subnet associations

Edit subnet associations

Available subnets (1/1)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
subnet-1-vpc-1	subnet-00317783e88c92db8	10.0.1.0/24	-	Main (rtb-052553bc90120f72b)

Selected subnets

subnet-00317783e88c92db8 / subnet-1-vpc-1 X
---

Cancel Save associations

VPC > Route tables > rtb-0899c7f284dc6c183 > Edit subnet associations

Edit subnet associations

Available subnets (1/1)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
subnet-2-vpc-2	subnet-0e77265093495f948	11.0.1.0/24	-	Main (rtb-05b7d25c9701bae8c)

Selected subnets

subnet-0e77265093495f948 / subnet-2-vpc-2 X
---

Cancel Save associations

**Security group** in AWS is a virtual firewall that controls incoming and outgoing traffic for one or more instances. It acts as a piece of virtual networking infrastructure that can be used to filter and restrict traffic to and from an EC2 instance. It allows you to specify inbound and outbound traffic rules that allow network traffic from specific IP addresses, ports, or protocols. Security groups can be associated with one or more instances, and changes to a security group apply immediately to all instances associated with that group.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links like EC2 Dashboard, EC2 Global View, Events, and Instances. Under Instances, it shows sub-links for Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, and Reservations. Below that is a section for Images (AMIs, AMI Catalog) and Elastic Block Store (Volumes). The main content area has a heading 'Instances (1/2) Info' with a search bar and filters. It lists two instances:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
instance-2-vpc-2	i-05a3434d7f281cac5	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-	
instance-1-vpc-1	i-0d5579f5d8e901280	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-	

Below the table, there's a detailed view for 'instance-1-vpc-1' with tabs for Details, Status and alarms New, Monitoring, Security, Networking, Storage, and Tags. The Security tab is selected. It shows the following details:

- Instance summary: Public IPv4 address 54.237.197.176 [open address], Instance state Running, Private IP DNS name (IPv4 only) ip-10-0-1-152.ec2.internal, Instance type t2.micro.
- Private IPv4 addresses: 10.0.1.152.
- Public IPv4 DNS: -.
- Elastic IP addresses: -.

**Editing inbound rules** is necessary because it allows you to configure access to your instance according to your requirements. For example, you can add a new rule to allow traffic from a specific IP address or range, to restrict malicious traffic to your EC2 instance, or to allow users to access your application.

**Note : We need to edit the inbound rules on both of the two servers.**

The screenshot shows the 'ModifyInboundSecurityGroupR' wizard on the 'Edit inbound rules' step. At the top, it says 'EC2 > Security Groups > sg-05c548206e07c17db - launch-wizard-21 > Edit inbound rules'. The main area is titled 'Edit inbound rules' and contains a table for 'Inbound rules'.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-090ac212793f36425	SSH	TCP	22	Custom	0.0.0.0/0
-	All traffic	All	All	Anyw...	0.0.0.0/0

At the bottom, there's a note: 'Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' There are 'Add rule' and 'Delete' buttons. At the very bottom, there are 'Cancel', 'Preview changes', and 'Save rules' buttons.

## Step 6 : Peering connection testing

Command :-

Ping <Second server IP address> ->in first server

Ping <First server IP address> ->in second server

The image shows two terminal windows side-by-side, both titled "EC2 Instance Connect | us-east-1". Both windows are connected to the same instance, "tirumalarao @ 3451-1957-4351", running Amazon Linux 2023.

The left terminal window (Instance-1) has a red arrow pointing to the command "ping 3.235.88.118". The output shows a series of ICMP echo requests being sent to the private IP 3.235.88.118, which is the public IP of Instance-2.

```
[ec2-user@ip-10-0-1-152 ~]$ ping 3.235.88.118
PING 3.235.88.118 (3.235.88.118) 56(84) bytes of data.
64 bytes from 3.235.88.118: icmp_seq=1 ttl=126 time=0.998 ms
64 bytes from 3.235.88.118: icmp_seq=2 ttl=126 time=0.960 ms
64 bytes from 3.235.88.118: icmp_seq=3 ttl=126 time=0.957 ms
64 bytes from 3.235.88.118: icmp_seq=4 ttl=126 time=1.05 ms
64 bytes from 3.235.88.118: icmp_seq=5 ttl=126 time=1.03 ms
64 bytes from 3.235.88.118: icmp_seq=6 ttl=126 time=0.984 ms
64 bytes from 3.235.88.118: icmp_seq=7 ttl=126 time=1.04 ms
64 bytes from 3.235.88.118: icmp_seq=8 ttl=126 time=0.992 ms
64 bytes from 3.235.88.118: icmp_seq=9 ttl=126 time=1.03 ms
64 bytes from 3.235.88.118: icmp_seq=10 ttl=126 time=1.02 ms
64 bytes from 3.235.88.118: icmp_seq=11 ttl=126 time=1.02 ms
64 bytes from 3.235.88.118: icmp_seq=12 ttl=126 time=0.995 ms
64 bytes from 3.235.88.118: icmp_seq=13 ttl=126 time=0.946 ms
64 bytes from 3.235.88.118: icmp_seq=14 ttl=126 time=0.982 ms
64 bytes from 3.235.88.118: icmp_seq=15 ttl=126 time=1.02 ms
```

The right terminal window (Instance-2) has a red arrow pointing to the command "ping 54.237.197.176". The output shows a series of ICMP echo requests being sent to the public IP 54.237.197.176, which is the public IP of Instance-1.

```
[ec2-user@ip-11-0-1-203 ~]$ ping 54.237.197.176
PING 54.237.197.176 (54.237.197.176) 56(84) bytes of data.
64 bytes from 54.237.197.176: icmp_seq=1 ttl=126 time=0.933 ms
64 bytes from 54.237.197.176: icmp_seq=2 ttl=126 time=0.928 ms
64 bytes from 54.237.197.176: icmp_seq=3 ttl=126 time=0.943 ms
64 bytes from 54.237.197.176: icmp_seq=4 ttl=126 time=0.957 ms
64 bytes from 54.237.197.176: icmp_seq=5 ttl=126 time=0.997 ms
64 bytes from 54.237.197.176: icmp_seq=6 ttl=126 time=0.985 ms
64 bytes from 54.237.197.176: icmp_seq=7 ttl=126 time=0.989 ms
64 bytes from 54.237.197.176: icmp_seq=8 ttl=126 time=0.923 ms
64 bytes from 54.237.197.176: icmp_seq=9 ttl=126 time=0.964 ms
64 bytes from 54.237.197.176: icmp_seq=10 ttl=126 time=1.02 ms
64 bytes from 54.237.197.176: icmp_seq=11 ttl=126 time=0.921 ms
64 bytes from 54.237.197.176: icmp_seq=12 ttl=126 time=0.984 ms
64 bytes from 54.237.197.176: icmp_seq=13 ttl=126 time=0.985 ms
64 bytes from 54.237.197.176: icmp_seq=14 ttl=126 time=0.980 ms
64 bytes from 54.237.197.176: icmp_seq=15 ttl=126 time=0.981 ms
```

Both terminals also show their respective instance IDs and public/private IP addresses at the bottom:

Left Terminal (Instance-1): i-0d5579f5d8e901280 (instance-1-vpc-1)  
PublicIPs: 54.237.197.176 PrivateIPs: 10.0.1.152

Right Terminal (Instance-2): i-05a3434d7f281cac5 (instance-2-vpc-2)  
PublicIPs: 3.235.88.118 PrivateIPs: 11.0.1.203