

Vpc - 2 task

1)Create one VPC, with 1 public subnet and 1 private subnet.

—> created a vpc with ipv4 cidr 192.168.0.0/24

The screenshot shows the AWS VPC dashboard under the 'Your VPCs' section. A blue banner at the top introduces VPC encryption control. The main table lists two VPCs: 'new-vpc' (selected) and another unnamed VPC. 'new-vpc' has an IPv4 CIDR of 192.168.0.0/24 and is set to Off for Block Public Access. The table includes columns for Name, VPC ID, State, Encryption controls, Block Public Access, IPv4 CIDR, and IPv6 CIDR.

Name	VPC ID	State	Encryption controls	Block Public Access	IPv4 CIDR	IPv6 CIDR
-	vpc-01126d967b1bb73af	Available	-	Off	172.31.0.0/16	-
<input checked="" type="checkbox"/> new-vpc	vpc-09cc1e7ee377b05d1	Available	-	Off	192.168.0.0/24	-

—> created two subnets 1 priv and 1 public

The screenshot shows the AWS Subnets dashboard under the 'Subnets' section. A blue banner at the top indicates 3 minutes ago. The main table lists two subnets: 'pri-sub1' and 'pub-sub1'. Both are associated with the VPC 'new-vpc' and have an IPv4 CIDR of 192.168.0.0/28. The table includes columns for Name, Subnet ID, State, VPC, Block Public Access, IPv4 CIDR, and IPv6 CIDR.

Name	Subnet ID	State	VPC	Block Public Access	IPv4 CIDR	IPv6 CIDR
pri-sub1	subnet-02bdf601fb02480d6	Available	vpc-09cc1e7ee377b05d1 new...	Off	192.168.0.16/28	-
pub-sub1	subnet-026832339ca13bf47	Available	vpc-09cc1e7ee377b05d1 new...	Off	192.168.0.0/28	-

2)Enable VPC peering for cross-region.

—> open your current region Stockholm and from above created vpc launch one instance using public subnet

—> and from other region Ohio launch a instance with default vpc and subnet

—> and give all traffic inbound in SG

—> sg of Ohio instanced

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
Security group rule ID sgr-08d59bc4c516336b7	All traffic	All	All	Custom	<input type="text" value="sg-0e76175cb35964c64"/> Delete
-	All traffic	All	All	Anywhere... Info	<input type="text" value="0.0.0.0/0"/> Delete

[Add rule](#)

⚠ Rules with source of 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.

[Cancel](#) [Preview changes](#) [Save rules](#)

—> vpc of ohio

VPC > Your VPCs

Introducing VPC encryption control
Manage and enforce encryption settings across your Virtual Private Cloud (VPC) resources. This centralized control helps ensure compliance with security policies and data protection regulations.

Create encryption control

Your VPCs

Virtual private cloud

Your VPCs (1/1) [Info](#)

VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv4 CIDR	IPv6 CIDR
vpc-021bc537bb7f21263	Available	On	Off	Off	172.31.0.0/16	

Actions [Create VPC](#)

Security

Network ACLs
Security groups

—> now lets start peering this two regions Stockholm and Ohio vpc's

—> current region is Stockholm

The screenshot shows the 'Create peering connection' page in the AWS VPC console. It includes sections for 'Peering connection settings', 'Select a local VPC to peer with', 'Select another VPC to peer with', and 'Tags'. The 'Peering connection settings' section has a 'Name - optional' field containing 'my-stockholm-ohio-peering'. The 'Select a local VPC to peer with' section shows a requester VPC ID 'vpc-09cc1e7ee377b05d1 (new-vpc)' and its CIDR '192.168.0.0/24' with status 'Associated'. The 'Select another VPC to peer with' section shows an accepter account 'My account' and region 'United States (Ohio) (us-east-2)', with a VPC ID 'vpc-021bc537bb7f21263'. The 'Tags' section defines a tag 'Key: Value - optional'.

—> go to **vpc**

—> **peering connections**

—> **create peering connection**

Add details of **vpc's** and

—> **peering request send to Ohio region**

A screenshot of the AWS VPC Peering Connections page. A green banner at the top indicates a new peering connection request: "A VPC peering connection ppx-04c1c4040b4a00d40 / my-stockholm-ohio-peering has been requested. Remember to change your region to us-east-2 to accept the peering connection." The main section shows the peering connection details:

- Requester owner ID:** 207662791773
- Peering connection ID:** ppx-04c1c4040b4a00d40
- Status:** Initiating Request to 207662791773
- Expiration time:** Monday, December 1, 2025 at 15:12:21 GMT+5:30
- Acceptor owner ID:** 207662791773
- Requester VPC:** vpc-09cc1e7ee377b05d1 / new-vpc
- Requester CIDRs:** 192.168.0.0/24
- Requester Region:** Stockholm (eu-north-1)
- Acceptor VPC:** vpc-021bc537bb7f21263
- Acceptor CIDRs:** -
- Acceptor Region:** Ohio (us-east-2)

The navigation bar includes tabs for DNS, Route tables, and Tags. The DNS settings section shows that both the requester and acceptor VPCs have disabled the option for the other to resolve DNS of hosts in their private IP addresses.

→ open your Ohio region vpc → peering connections

A screenshot of the AWS VPC Peering Connections page in the Ohio region. It shows one peering connection request listed:

Name	Peering connection ID	Status	Requester VPC	Acceptor VPC
-	ppx-04c1c4040b4a00d40	Pending acceptance	vpc-09cc1e7ee377b05d1	vpc-021bc537bb7f21263

An orange context menu is open over the row, with "Accept request" highlighted. Other options include View details, Reject request, Edit DNS settings, Manage tags, and Delete peering connection.

→ we can see peering request from Stockholm is received

→ now go to both regions public routes and add Ohio ipv4 range for Stockholm and Stockholm ipv4 for ohio in public routes
 → below is Stockholm public route

A screenshot of the AWS Route Tables page. It shows an "Edit routes" dialog for route table rtb-05a4d6835592f0ed5. A new route is being added:

Destination	Target	Status	Propagated	Route Origin
192.168.0.0/24	local	Active	No	CreateRouteTable
Q. 0.0.0.0/0	Internet Gateway	Active	No	CreateRoute
Q. 172.31.0.0/16	Peer Connection	-	No	CreateRoute
Q. ppx-04c1c4040b4a00d40	Use: "ppx-04c1c4040b4a00d40"	-	-	-
	ppx-04c1c4040b4a00d40 (my-stockholm-ohio-peering)			

At the bottom right of the dialog are buttons for Cancel, Preview, and Save changes.

→ below is Ohio public route

Destination Target Status Propagated Route Origin

Destination	Target	Status	Propagated	Route Origin
172.31.0.0/16	local	Active	No	CreateRouteTable
0.0.0.0/0	Internet Gateway	Active	No	CreateRoute
192.168.0.0/24	Peering Connection	-	No	CreateRoute

Add route Cancel Preview Save changes

→ we can see peering target added in both route

Updated routes for rtb-0c5e753dc2d8ce8b9 successfully

Details

Route table ID: rtb-0c5e753dc2d8ce8b9
Owner ID: 207662791773

Routes (3)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-08b3d10e56fc24d1b	Active	No	Create Route
172.31.0.0/16	local	Active	No	Create Route Table
192.168.0.0/24	pxx-04c1c4040b4a00d40	Active	No	Create Route

→ now copy the Ohio instance private ip and

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv
my-ohio-inst...	i-056f6160867b5e3b0	Running	t3.micro	3/3 checks passed	View alarms +	us-east-2c	ec2-18-188-223-55.us...	18.188.223.55

i-056f6160867b5e3b0 (my-ohio-instance)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary

Instance ID: i-056f6160867b5e3b0
IPv6 address:

Public IPv4 address: 18.188.223.55 | open address

Private IPv4 address copied

Private IPv4 addresses: 172.31.37.105

Public DNS:

→ now open the server of Stockholm and ping the private ip of 172.31.37.105

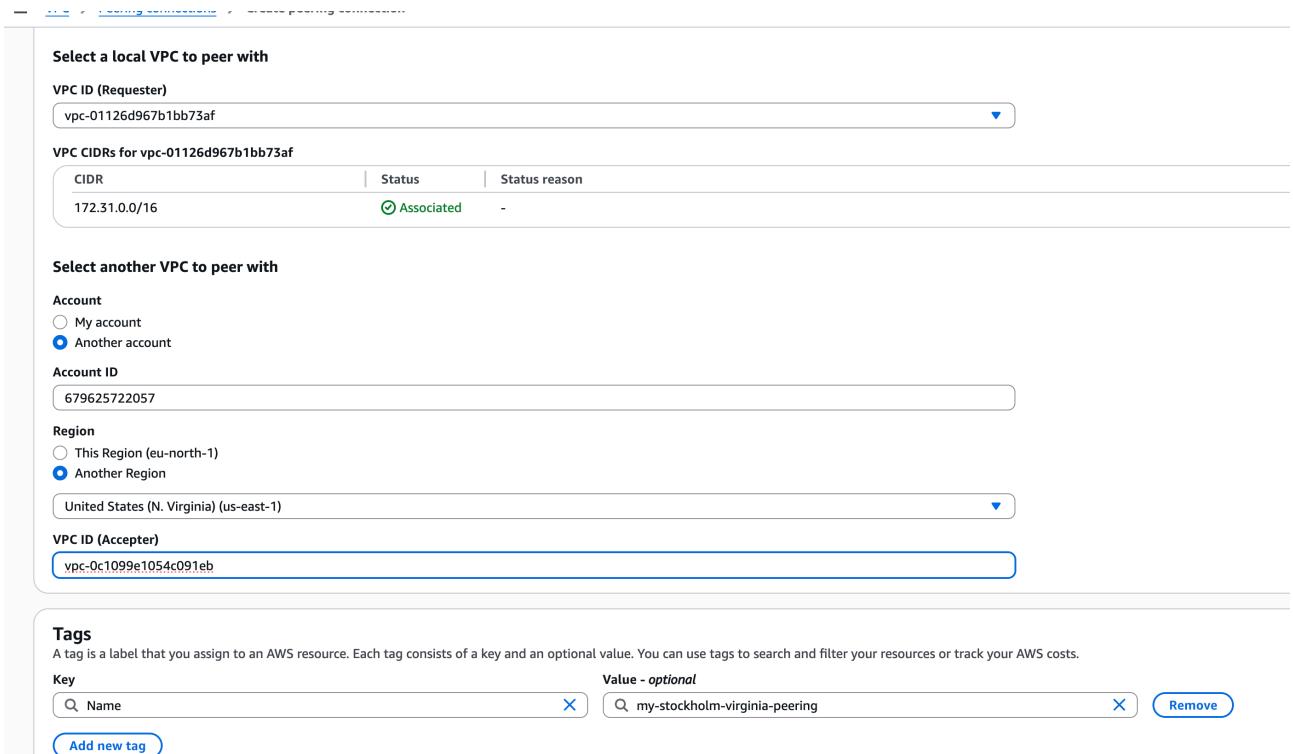
→ we can see the two instance are peering the private data of Ohio region instance

```
The authenticity of host 'ec2-56-228-11-214.eu-north-1.compute.amazonaws.com (56.228.11.214)' can't be established.  
ED25519 key fingerprint is SHA256:ihbgbz3fAT03RzqZd9J5UIlaeLm4HMWhq1ULDrvSm2G8.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-56-228-11-214.eu-north-1.compute.amazonaws.com' (ED25519) to the list of known hosts.  
, #  
~\ _ #####_ Amazon Linux 2023  
~~ \#####\  
~~ \###|  
~~ \#/ __ https://aws.amazon.com/linux/amazon-linux-2023  
~~ V~' '-'>  
~~ /  
~~ .-. /  
~~ / /  
~/m/  
[[ec2-user@ip-192-168-0-12 ~]$ sudo -i  
[[root@ip-192-168-0-12 ~]# ping 172.31.37.105  
PING 172.31.37.105 (172.31.37.105) 56(84) bytes of data.  
64 bytes from 172.31.37.105: icmp_seq=1 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=2 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=3 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=4 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=5 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=6 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=7 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=8 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=9 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=10 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=11 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=12 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=13 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=14 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=15 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=16 ttl=127 time=120 ms  
64 bytes from 172.31.37.105: icmp_seq=17 ttl=127 time=120 ms
```

3) Enable VPC peering for cross-account (you can collaborate with your friend to do this task).

- > created a vpc with 172.31.0.0/16 range
- > and created public subnet
- > created a route and added internet gate to it

- > now go to vpc
- > peering connections
- > give your vpc details
- > select another vpc to peer with
- > friends account id
- > region select according to friends region
- > add friends vpc id



The screenshot shows the 'Create Peering Connection' wizard. Step 1: Select a local VPC to peer with. The VPC ID (Requester) is set to 'vpc-01126d967b1bb73af'. Step 2: Select another VPC to peer with. The Account is set to 'Another account' (selected), and the Account ID is '679625722057'. The Region is set to 'Another Region' (selected), and the Region is 'United States (N. Virginia) (us-east-1)'. Step 3: Tags. A tag 'Name' is added with the value 'my-stockholm-virginia-peering'. The 'Add new tag' button is visible.

- > request send to the friends vpc peering

- > later after friend accepting the peer request

A screenshot of the AWS VPC Peering Connections console. A green banner at the top states: "A VPC peering connection ppx-05ef9b3706d1432cf / my-stockholm-virginia-peering has been requested. The owner of vpc-0c1099e1054c091eb must accept the peering connection." Below this, the peering connection details are shown:

Details		Actions	
Requester owner ID	207662791773	Acceptor owner ID	679625722057
Peering connection ID	ppx-05ef9b3706d1432cf	Requester VPC	vpc-01126d967b1bb73af
Status	Initiating Request to 679625722057	Requester CIDRs	172.31.0.0/16
Expiration time	Tuesday, December 2, 2025 at 19:46:35 GMT+5:30	Requester Region	Stockholm (eu-north-1)
DNS		Route tables	
Tags		Edit DNS settings	
DNS settings			
Requester VPC (vpc-01126d967b1bb73af) Info			

Peering connections (1) Info						Actions	Create peering connection
Find peering connections by attribute or tag						Actions	
Name	Peering connection ID	Status	Requester VPC	Acceptor VPC	Requester CIDRs		
my-stockholm-virginia-peering	ppx-05ef9b3706d1432cf	Active	vpc-01126d967b1bb73af	vpc-0c1099e1054c091eb	172.31.0.0/16		

- > go and add route of the peering request
- > now after my friend accept the peer request we can see there is active status means peering is now successfully running and active

next

—> my public route

A screenshot of the AWS Route Tables console. The "Edit routes" section shows a table for adding routes:

Destination	Target	Status	Propagated	Route Origin
172.31.0.0/16	local	Active	No	CreateRouteTable
Q. 0.0.0.0/0	Q. local			
Q. 10.0.0.0/16	Internet Gateway	Active	No	CreateRoute
	Q. igw-06827b286d7eba643			
	Peerings Connection	-	No	CreateRoute
	Q. ppx-05ef9b3706d1432cf			

At the bottom, there are buttons for "Add route", "Cancel", "Preview", and "Save changes".

→ added friends ipv4 range and added peering connectivity

And saved

rtb-0da8e9d61854221d2

Details [Info](#)

Route table ID [rtb-0da8e9d61854221d2](#)

Main Yes

VPC [vpc-01126d967b1bb73af](#)

Owner ID [207662791773](#)

Explicit subnet associations -

Edge associations -

Routes Subnet associations Edge associations Route propagation Tags

Routes (3)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-06827b286d7eba643	Active	No	Create Route
10.0.0.0/16	pxc-05ef9b3706d1432cf	Active	No	Create Route
172.31.0.0/16	local	Active	No	Create Route Table

→ on other side friends has also added the same my ipv4 172.31.0.0/16 as peering connectivity

→ edit the security group and add all traffic for it

EC2 > Security Groups > [sg-0853619dc44ef762e - default](#) > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [Info](#)

Type	Protocol	Port range	Source	Description - optional
All traffic	All	All	Custom	sg-0853619dc44ef762e
All traffic	All	All	Custom	0.0.0.0/0
All traffic	All	All	Custom	10.0.0.0/16

Add rule

⚠ 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.

Cancel Preview changes Save rules

```
The authenticity of host ec2-13-60-68-2.eu-north-1.compute.amazonaws.com (13.60.68.2) can't be established.
ED25519 key fingerprint is SHA256:UjJEth/1/4rYjn+jjDyWJxNGMR2+tcX2KednW/KP/dk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-60-68-2.eu-north-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
, #_
~\ ####_      Amazon Linux 2023
~~ \####\_
~~ \###|
~~  \#/ __ https://aws.amazon.com/linux/amazon-linux-2023
~~  V~' '-->
~~   /
~~_. /_
~/ /_
~/m'

[[ec2-user@ip-172-31-20-78 ~]$ ping 10.0.0.227
PING 10.0.0.227 (10.0.0.227) 56(84) bytes of data.
64 bytes from 10.0.0.227: icmp_seq=1 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=2 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=3 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=4 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=5 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=6 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=7 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=8 ttl=127 time=110 ms
64 bytes from 10.0.0.227: icmp_seq=9 ttl=127 time=110 ms
```

- > add the same range in your inbound security group also
- > ping the friends instance private ip 10.0.0.227
- > its working

4) Set up a VPC Transit Gateway.

- > create a no of vpc's
- Given below ranges for three vpc's

10.0.1.0/28
 172.31.0.0/16
 10.0.0.0/28

Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv4 CIDR	IPv6 CIDR
default	vpc-07968786753c2ea72	Available	-	-	Off	172.31.0.0/16	-
myvpc1	vpc-0bd93d506aa0af36	Available	-	-	Off	10.0.0.0/28	-
myvpc2	vpc-0250eb93b24951001	Available	-	-	Off	10.0.1.0/24	-

—> create three subnets with one subnet for one vpc

Go in vpc and check for transit gateway
 —> created a transit gateway

Name	Transit gateway ID	Owner ID	State
test-tg	tgw-024db055556f9d5c9	207662791773	Pending

—> now attach the all vpc's with the created transit gateway

The screenshot shows the AWS VPC console with the 'Transit gateway attachments' section. A green success message at the top says 'You successfully created VPC attachment tgw-attach-087ee0401372dc586 / default-tga.' Below it, a blue banner introduces the 'Metering Policy for Transit Gateway (TGW)' feature. A table lists one attachment: 'default-tga' with ID 'tgw-attach-087ee0401372dc586' and TGW ID 'tgw-024db055556f9d5c9', marked as 'Pending'. A 'Create transit gateway attachment' button is visible.

—> attached the three vpc's

The screenshot shows the AWS VPC console with the 'Transit gateway attachments' section. Three attachments are listed: 'vpc1-tga', 'default-tga', and 'vpc2-tga', all in 'Available' state. Each attachment is associated with a specific VPC and TGW ID. A 'Create transit gateway attachment' button is visible.

—> this is our transit gateway route table in which we have to add all the ranges in each route of vpc's

The screenshot shows the AWS VPC console with the 'Transit gateway route tables' section. One route table is listed: 'tgw-rtb-0b27ec658dddf4aeb' with TGW ID 'tgw-024db055556f9d5c9', marked as 'Available'. It has 'Yes' selected for both 'Default association route table' and 'Default propagation'. A 'Create transit gateway route table' button is visible.

—> route table of transit gate way which is attached to three vpc's

Transit gateway route tables (1/1) Info					
<input checked="" type="checkbox"/>	Name D	Transit gateway route table ID	Transit gateway ID	State	Default association route table
<input checked="" type="checkbox"/>	tgw-rtb-0b27ec658ddd4aeb	tgw-024db055556f9d5c9	Available	Yes	Yes

Transit gateway route tables: tgw-rtb-0b27ec658ddd4aeb					
Details	Associations	Propagations	Prefix list references	Routes	Tags
Associations (3) Info					
<input type="checkbox"/>	Attachment ID	Resource type	Resource ID	State	
<input type="checkbox"/>	tgw-attach-0e3cf8ff4fe7c7462	VPC	vpc-06efea456995169bce	Associated	Delete association Create association
<input type="checkbox"/>	tgw-attach-03d11a4501cb12f07	VPC	vpc-0bd93d506aa0a1f36	Associated	Delete association Create association
<input type="checkbox"/>	tgw-attach-087ee0401372dc586	VPC	vpc-07968786753c2ea72	Associated	Delete association Create association

—> launch the three instances from each each vpc's

EC2 > Instances										
Instances (3) Info										
Find Instance by attribute or tag (case-sensitive) Last updated less than a minute ago Connect Instance state Actions Launch instances										
<input type="checkbox"/>	Name D	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv	
<input type="checkbox"/>	default-vpc-ec2	i-04e774a1a95545bfc	Running Q Q	t3.micro	Initializing	View alarms +	us-east-1a	ec2-100-27-1-190.com...	100.27.1.1	
<input type="checkbox"/>	vpc2-ec2	i-0e453c9bf29c9a61b9	Running Q Q	t3.micro	—	View alarms +	us-east-1a	—	3.238.89.1	
<input type="checkbox"/>	vpc1-ec2	i-0e50691f8e6bb2754	Running Q Q	t3.micro	Initializing	View alarms +	us-east-1f	—	98.93.27.1	

—>and choose that subnet which we created after creating vpc's

—> now go inside the route table of each each vpc's and add other two vpc ipv4 range with transit gateway

The screenshot shows the 'Edit routes' page for a specific route table. The table lists the following routes:

Destination	Target	Status	Propagated	Route Origin
172.31.0.0/16	local	Active	No	CreateRouteTable
0.0.0.0/0	Internet Gateway igw-05d93d6a24c45086c	Active	No	CreateRoute
10.0.0.0/28	Transit Gateway tgw-024db055556f9d5c9	-	No	CreateRoute
10.0.1.0/28	Transit Gateway tgw-024db055556f9d5c9	-	No	CreateRoute

Buttons at the bottom include 'Add route', 'Cancel', 'Preview', and 'Save changes'.

—> like we are in default vpc with 172.31.0.0/16
 We are adding other vpc range of 10.0.0.0/16 (vpc1)
 And vpc range of 10.0.1.0/16 (vpc2)

—> similarly add other two route tables of other vpc's just like above

The screenshot shows the 'Details' page for a route table named 'rtb-0b5d29d80613137e7 / default-rt'. A green banner at the top indicates 'Updated routes for rtb-0b5d29d80613137e7 / default-rt successfully'. The page displays the following details:

- Details**: Route table ID: rtb-0b5d29d80613137e7, Main: Yes, Owner ID: vpc-07968786753c2ea72 | default, Last updated: 2023-09-14 10:30:00 UTC.
- Routes (4)**: A table showing routes with their destinations, targets, statuses, propagated status, and route origins:

Destination	Target	Status	Propagated	Route Origin
0.0.0.0	igw-05d93d6a24c45086c	Active	No	Create Route
10.0.0.0/28	tgw-024db055556f9d5c9	Active	No	Create Route
10.0.1.0/28	tgw-024db055556f9d5c9	Active	No	Create Route
172.31.0.0/16	local	Active	No	Create Route Table

→ these are your route tables of default vpc ,vpc1,vpc2

The screenshot shows the AWS VPC Route Tables page. On the left, there's a sidebar with 'Virtual private cloud' and 'Route tables' sections. The main area displays a table titled 'Route tables (3) Info' with columns for Name, Route table ID, Explicit subnet associations, Edge associations, Main, VPC, and Owner ID. The routes are:

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID
default-rt	rtb-0b5d29d80613137e7	-	-	Yes	vpc-07968786753c2ea72 defa...	207662791773
vpc2-rt	rtb-0d9fe83fd1fa635ae	-	-	Yes	vpc-06efa456995169bce vpc2	207662791773
vpc1-rt	rtb-0f764cc462ca4e7a6	-	-	Yes	vpc-0bd93d506aa0a1f36 vpc1	207662791773

vpc2 route

The screenshot shows the 'Edit routes' page for the vpc2-rt route table. It lists three routes:

- Destination: 10.0.1.0/28, Target: local, Status: Active, Propagated: No, Route Origin: CreateRouteTable.
- Destination: 10.0.0.0/28, Target: Transit Gateway, Status: -, Propagated: No, Route Origin: CreateRoute. The target is tgw-024db055556f9d5c9.
- Destination: 172.31.0.0/16, Target: Transit Gateway, Status: -, Propagated: No, Route Origin: CreateRoute. The target is tgw-024db055556f9d5c9.

Buttons at the bottom include 'Add route', 'Cancel', 'Preview', and 'Save changes'.

vpc1 route

The screenshot shows the 'Edit routes' page for the vpc1-rt route table. It lists three routes:

- Destination: 10.0.0.0/28, Target: local, Status: Active, Propagated: No, Route Origin: CreateRouteTable.
- Destination: 172.31.0.0/16, Target: Transit Gateway, Status: -, Propagated: No, Route Origin: CreateRoute. The target is tgw-024db055556f9d5c9.
- Destination: 10.0.1.0/28, Target: Transit Gateway, Status: -, Propagated: No, Route Origin: CreateRoute. The target is tgw-024db055556f9d5c9.

Buttons at the bottom include 'Add route', 'Cancel', 'Preview', and 'Save changes'.

→ and create internet gateway for each route which is attached to its vpc's

The screenshot shows the AWS VPC Internet Gateways page. On the left, there's a sidebar with 'Virtual private cloud' and 'Internet gateways' sections. The main area displays a table titled 'Internet gateways (3) Info' with columns for Name, Internet gateway ID, State, VPC ID, and Owner. The gateways are:

Name	Internet gateway ID	State	VPC ID	Owner
default-vpc	igw-05d93d6a24c45086c	Attached	vpc-07968786753c2ea72 default	207662791773
vpc1-igw	igw-011c66301b5057b6d	Attached	vpc-0bd93d506aa0a1f36 vpc1	207662791773
vpc2-igw	igw-00339370171070665	Attached	vpc-06efa456995169bce vpc2	207662791773

→ and add the internet gate way to other two route's also

The screenshot shows the AWS VPC Route Table Details page. The route table ID is rtb-0f764cc462ca4e7a6, associated with VPC vpc-0bd95d506aa0a1f36. There are no explicit subnet associations or edge associations. The routes section lists four routes:

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-011c66301b5057b6d	Active	No	Create Route
10.0.0.0/28	local	Active	No	Create Route Table
10.0.1.0/28	tgw-024db055556f9d5c9	Active	No	Create Route
172.31.0.0/16	tgw-024db055556f9d5c9	Active	No	Create Route

→ add internet gate way to all routes

The screenshot shows the 'Edit routes' page for the route table rtb-0d9fe83fd1fa635ae. Five new routes are being added:

Destination	Target	Status	Propagated	Route Origin
10.0.1.0/28	local	Active	No	CreateRouteTable
10.0.0.0/28	Transit Gateway	Active	No	CreateRoute
172.31.0.0/16	Transit Gateway	Active	No	CreateRoute
0.0.0.0/0	Internet Gateway	-	No	CreateRoute

The last route entry shows the target as 'Internet Gateway' with a status of '-'. The 'Add route' button is visible at the bottom left.

→ add all traffic to all security groups of each instances

sg-0cb0dfb2177545b2a - default

[Actions ▾](#)

Details		Security group ID		Description		VPC ID	
Security group name default		sg-0cb0dfb2177545b2a		default VPC security group		vpc-0bd93d506aa0a1f36	Edit
Owner 207662791773		Inbound rules count 2 Permission entries		Outbound rules count 1 Permission entry			

[Inbound rules](#) | [Outbound rules](#) | [Sharing - new](#) | [VPC associations - new](#) | [Tags](#)

Inbound rules (2)

<input type="checkbox"/> Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/> -	sgr-0f8b6da6038c54f6c	IPv4	All traffic	All	All	0.0.0.0/0
<input type="checkbox"/> -	sgr-0a26ee420bc8d7620	-	All traffic	All	All	sg-0cb0dfb2177545b2a...

—> this will allow to run the connectivity

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

[[ec2-user@ip-172-31-8-89 ~]$ ping 10.0.1.8
PING 10.0.1.8 (10.0.1.8) 56(84) bytes of data.
^C
--- 10.0.1.8 ping statistics ---
829 packets transmitted, 0 received, 100% packet loss, time 861097ms

[[ec2-user@ip-172-31-8-89 ~]$ ping 10.0.1.8
PING 10.0.1.8 (10.0.1.8) 56(84) bytes of data.
64 bytes from 10.0.1.8: icmp_seq=110 ttl=126 time=0.911 ms
64 bytes from 10.0.1.8: icmp_seq=111 ttl=126 time=0.416 ms
64 bytes from 10.0.1.8: icmp_seq=112 ttl=126 time=0.410 ms
64 bytes from 10.0.1.8: icmp_seq=113 ttl=126 time=0.555 ms
64 bytes from 10.0.1.8: icmp_seq=114 ttl=126 time=0.412 ms
64 bytes from 10.0.1.8: icmp_seq=115 ttl=126 time=0.427 ms
64 bytes from 10.0.1.8: icmp_seq=116 ttl=126 time=0.459 ms
64 bytes from 10.0.1.8: icmp_seq=117 ttl=126 time=0.423 ms
64 bytes from 10.0.1.8: icmp_seq=118 ttl=126 time=0.428 ms
64 bytes from 10.0.1.8: icmp_seq=119 ttl=126 time=0.454 ms
64 bytes from 10.0.1.8: icmp_seq=120 ttl=126 time=0.466 ms
64 bytes from 10.0.1.8: icmp_seq=121 ttl=126 time=0.412 ms
```

→ I have pinged the vpc 1 instance and also pinged the vpc 2 instance with default vpc

```
Warning: Permanently added '100.27.1.190' (ED25519) to the list of known hosts.  
 , #  
 ~\ _ #####_ Amazon Linux 2023  
 ~~ \#####\|  
 ~~ \#/ __ https://aws.amazon.com/linux/amazon-linux-2023  
 ~~ V-' '-'>  
 ~~~ /  
 ~~.-' _/  
 ~~ / _/  
 _/m'  
[ec2-user@ip-172-31-8-89 ~]$ ping 10.0.1.8  
PING 10.0.1.8 (10.0.1.8) 56(84) bytes of data.  
^C  
--- 10.0.1.8 ping statistics ---  
829 packets transmitted, 0 received, 100% packet loss, time 861097ms  
  
[ec2-user@ip-172-31-8-89 ~]$ ping 10.0.1.8  
PING 10.0.1.8 (10.0.1.8) 56(84) bytes of data.  
64 bytes from 10.0.1.8: icmp_seq=110 ttl=126 time=0.911 ms  
64 bytes from 10.0.1.8: icmp_seq=111 ttl=126 time=0.416 ms  
64 bytes from 10.0.1.8: icmp_seq=112 ttl=126 time=0.410 ms  
64 bytes from 10.0.1.8: icmp_seq=113 ttl=126 time=0.555 ms  
64 bytes from 10.0.1.8: icmp_seq=114 ttl=126 time=0.412 ms  
64 bytes from 10.0.1.8: icmp_seq=115 ttl=126 time=0.427 ms  
64 bytes from 10.0.1.8: icmp_seq=116 ttl=126 time=0.459 ms  
64 bytes from 10.0.1.8: icmp_seq=117 ttl=126 time=0.423 ms  
64 bytes from 10.0.1.8: icmp_seq=118 ttl=126 time=0.428 ms  
64 bytes from 10.0.1.8: icmp_seq=119 ttl=126 time=0.454 ms  
64 bytes from 10.0.1.8: icmp_seq=120 ttl=126 time=0.466 ms  
64 bytes from 10.0.1.8: icmp_seq=121 ttl=126 time=0.412 ms  
64 bytes from 10.0.1.8: icmp_seq=122 ttl=126 time=0.463 ms  
64 bytes from 10.0.1.8: icmp_seq=123 ttl=126 time=0.412 ms  
64 bytes from 10.0.1.8: icmp_seq=124 ttl=126 time=0.430 ms  
64 bytes from 10.0.1.8: icmp_seq=125 ttl=126 time=0.426 ms  
^C  
--- 10.0.1.8 ping statistics ---  
125 packets transmitted, 16 received, 87.2% packet loss, time 128995ms  
rtt min/avg/max/mdev = 0.410/0.469/0.911/0.119 ms  
[ec2-user@ip-172-31-8-89 ~]$ ping 10.0.0.14  
PING 10.0.0.14 (10.0.0.14) 56(84) bytes of data.  
64 bytes from 10.0.0.14: icmp_seq=1 ttl=126 time=2.44 ms  
64 bytes from 10.0.0.14: icmp_seq=2 ttl=126 time=1.02 ms  
64 bytes from 10.0.0.14: icmp_seq=3 ttl=126 time=0.669 ms  
64 bytes from 10.0.0.14: icmp_seq=4 ttl=126 time=0.761 ms  
64 bytes from 10.0.0.14: icmp_seq=5 ttl=126 time=0.730 ms  
64 bytes from 10.0.0.14: icmp_seq=6 ttl=126 time=0.729 ms  
64 bytes from 10.0.0.14: icmp_seq=7 ttl=126 time=0.665 ms  
64 bytes from 10.0.0.14: icmp_seq=8 ttl=126 time=1.05 ms  
64 bytes from 10.0.0.14: icmp_seq=9 ttl=126 time=0.794 ms  
64 bytes from 10.0.0.14: icmp_seq=10 ttl=126 time=0.650 ms  
64 bytes from 10.0.0.14: icmp_seq=11 ttl=126 time=0.649 ms  
64 bytes from 10.0.0.14: icmp_seq=12 ttl=126 time=0.679 ms  
64 bytes from 10.0.0.14: icmp_seq=13 ttl=126 time=0.659 ms  
64 bytes from 10.0.0.14: icmp_seq=14 ttl=126 time=0.682 ms  
64 bytes from 10.0.0.14: icmp_seq=15 ttl=126 time=0.702 ms
```

5) Set up a VPC Endpoint

- > go to vpc
- > search for endpoint

The screenshot shows the 'Create endpoint' wizard. In the 'Endpoint settings' step, the 'AWS services' option is selected, with the value 's3-endpoint'. In the 'Service Region' step, 'Enable Cross Region endpoint' is checked, and 'Europe (Stockholm) (eu-north-1)' is selected. The 'Network settings' and 'Route tables' steps are also visible.

Create endpoint Info

Create the type of VPC endpoint that supports the service, service network or resource to which you want to connect.

Endpoint settings

Specify a name and select the type of endpoint.

Name tag - optional

Creates a tag with a key of 'Name' and a value that you specify. Tags help you find and manage your endpoint.

s3-endpoint

Type Info

Select a category

AWS services
Connect to services provided by Amazon with an Interface endpoint, or a Gateway endpoint

EC2 Instance Connect Endpoint
An elastic network interface that allows you to connect to resources in a private subnet

Endpoint services that use NLBs and GWLBs
Find services shared with you by service name. Connect to a Network LoadBalancer (NLB) service with an Interface endpoint or to a Gateway LoadBalancer (GWLB) service with a Gateway Load Balancer endpoint

PrivateLink Ready partner services
Connect to SaaS services which have AWS Service Ready designation with an Interface endpoint. Uses AWS PrivateLink

Resources
Connect to resources like Amazon Relational Database Services (RDS) with a Resource endpoint. Uses AWS PrivateLink

AWS Marketplace
Connect to SaaS services which have AWS Service Ready designation with an Interface endpoint

Service network
Connect to VPC Lambda services via AWS PrivateLink

Service Region

Enable Cross Region endpoint Info
Connect to cross Region enabled services.

Europe (Stockholm) (eu-north-1)

- > Endpoint is a service which helps us to communicate with aws services without internet

The screenshot shows the 'Route tables' step of the endpoint creation wizard. It lists two route tables: 'rtb-0da8e9d61854221d2' (Main, Associated with 2 subnets) and 'rtb-0092524888a9180ad (private-routh)' (No). A note at the bottom explains that source IP addresses will be private IP addresses, not public IP addresses, and that existing connections from affected subnets to the AWS service that use public IP addresses may be dropped.

Route tables (1/2) Info

Search

Name	Route Table ID	Main	Associated Id
-	rtb-0da8e9d61854221d2	Yes	2 subnets
private-routh	rtb-0092524888a9180ad (private-routh)	No	subnet-05f27e3702839a75e (private-subnet)

When you use an endpoint, the source IP addresses from your instances in your affected subnets for accessing the AWS service in the same region will be private IP addresses, not public IP addresses. Existing connections from your affected subnets to the AWS service that use public IP addresses may be dropped. Ensure that you don't have critical tasks running when you create or modify an endpoint.

- > select s3 as services add vpc details ,select the private route
- > and create

—> endpoint is created

A screenshot of the AWS VPC console showing a success message: "Successfully created VPC endpoint vpce-01332b206155aed1d". Below it, a table lists the endpoint details: Name (s3-endpoint), VPC endpoint ID (vpce-01332b206155aed1d), Endpoint type (Gateway), Status (Available), and Service name (com.amazonaws.eu-north-1.s3). The table has columns for Name, VPC endpoint ID, Endpoint type, Status, and Service name.

—> check the private route we can see the endpoint is connected to private route

A screenshot of the AWS Network Border Group (N BG) configuration page. It shows the N BG settings: Network Address Usage metrics (Disabled), Route 53 Resolver DNS Firewall rule groups (empty), and Owner ID (207662791773). Below this, the "Route map" tab is selected, showing a network diagram. The diagram includes three subnets: eu-north-1a (public-subnet), eu-north-1b (private-subnet), and eu-north-1c (subnet-006e395d3b72e2ed7). These subnets are connected to two route tables: private-routh and rtb-0da8e9d61854221d2. The private-routh route table is connected to the igw-06827b286d7eba643 gateway and the s3-endpoint endpoint.

A screenshot of the AWS Route Table configuration page. It shows the route table details: Route table ID (rtb-0092524888a9180ad), Main (No), and Owner ID (207662791773). It also lists explicit subnet associations: subnet-05f27e3702839a75e / private-subnet. The "Routes" tab is selected, showing two routes: one to target vpce-01332b206155aed1d (Status: Active, Propagated: No, Route Origin: Create Route) and another to target local (Status: Active, Propagated: No, Route Origin: Create Route Table).

—> this will communicate with aws services without internet