

VPC Networking Lab

How do you document this lab?

1. Capture the full screenshot of the AWS console; this should contain your name on the screenshot.

Ex:

The screenshot shows the AWS VPC dashboard. At the top, there's a navigation bar with icons for Services, Search, and various AWS services like Lambda, CloudFormation, and RDS. The region is set to N. Virginia, and the account ID vxs220071 is visible. The main content area is titled "VPC dashboard" and includes sections for "Create VPC" and "Launch EC2 Instances". Below this, there's a "Resources by Region" section with cards for 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, and Peering Connections. To the right, there are "Service Health", "Settings" (with links to Zones and Console Experiments), "Additional Information" (with links to VPC Documentation, All VPC Resources, Forums, and Report an Issue), and sections for AWS Network Manager and Site-to-Site VPN Connections. A red circle highlights the account ID "vxs220071" in the top right corner of the dashboard header.

2. Make sure you do your labs in the North Virginia region; this is not mandatory but is advice. This will help troubleshoot a lot of things, including your billing.

Deliverables: Submit the screenshots where it mentions “**Note: This is a Deliverable.**” The screenshots should be submitted in the pdf format.

Step 5 – f

Step 6 – a

Step 7 – a

Step 9 – c and d

Table of Contents:

[Step 1: Create VPC.](#)

[Step 2: Create an Internet Gateway and Attach it to VPC.](#)

[Step 3: Create Public and Private Subnets.](#)

[Step 4: Create Private Route Tables.](#)

[Step 5: Associate route tables to the subnets appropriately.](#)

[Step 6: Test the Connectivity by Creating a Public EC2.](#)

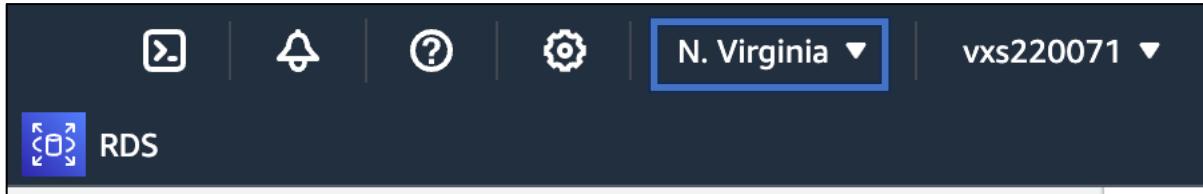
[Step 7: Test the Connectivity by Creating a Private EC2.](#)

[Step 8: How do you login to the Private server \(i.e., ec2 in the private subnet\)](#)

[Step 9: NAT Gateway creation and testing internet connection from the private server.](#)

Step 1: Create VPC

- Login to AWS Console and set the region to N.Virginia. This option is available at the top right corner beside your account name.



- Now, search for VPC service and click on **Create VPC**. Choose/fill out the options as given below and Create the VPC. Note: Understand each column in detail. Understand that VPC spans the entire region.

The screenshot shows the "Create VPC" wizard step 1: VPC settings. The URL in the browser is [VPC > Your VPCs > Create VPC](#).

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

VPC only VPC and more

Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.
my-first-vpc

IPv4 CIDR block [Info](#)
 IPv4 CIDR manual input
 IPAM-allocated IPv4 CIDR block

IPv4 CIDR
10.0.0.0/16
CIDR block size must be between /16 and /28.

IPv6 CIDR block [Info](#)
 No IPv6 CIDR block
 IPAM-allocated IPv6 CIDR block
 Amazon-provided IPv6 CIDR block
 IPv6 CIDR owned by me

Tenancy [Info](#)

Build AWS Network

The screenshot shows the 'Tags' section of the AWS VPC creation interface. It includes a description of what tags are, a key-value pair ('Name' set to 'my-first-vpc'), and buttons for 'Add tag' and 'Create VPC'.

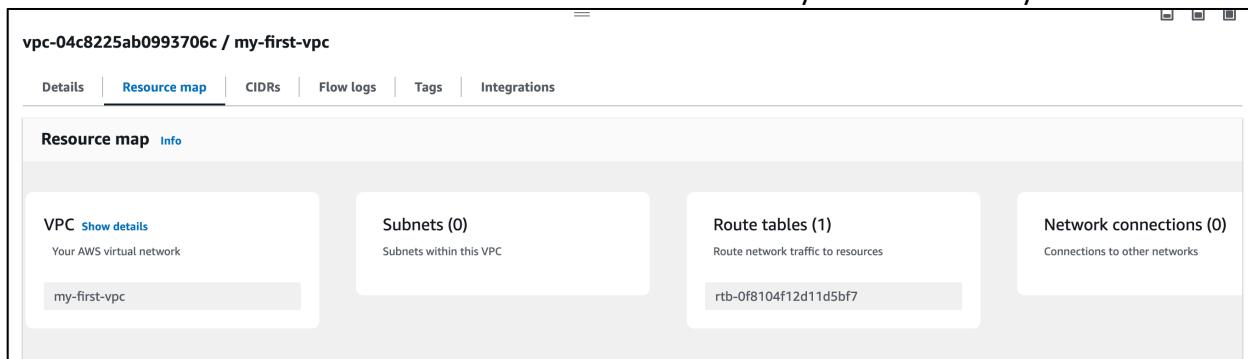
Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key Value - *optional*

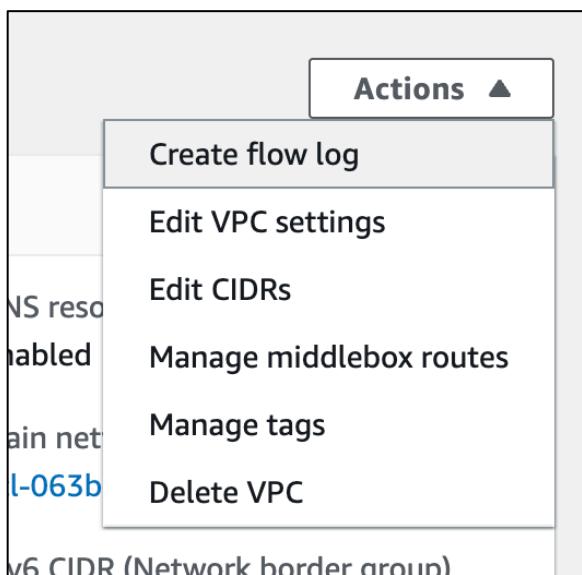
You can add 49 more tags

- c. This is how it looks when you first create your VPC. Observe the resource map as we move forward while creating each resource in VPC.

You have to understand that a route table is created by default for every VPC created.



- d. Enable DNS Hostnames: Click on **Edit VPC settings**, make the changes as shown below.



DNS settings

- Enable DNS resolution [Info](#)
- Enable DNS hostnames [Info](#)

Step 2: Create an Internet Gateway and Attach it to VPC.

- a. Now, Let's create an Internet Gateway.

What is an Internet Gateway? It's simply a virtual device to connect to the Internet. So, let's say we have a server created in a VPC. How do you connect to the internet from that server? That's where Internet Gateway comes into the picture. It's pretty similar to the Spectrum Wi-Fi, right? (Analogy)

- b. Go to the VPC page and look out for Internet Gateway. Create One as shown below.

VPC > [Internet gateways](#) > [Create internet gateway](#)

Create internet gateway [Info](#)

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

Name tag
Creates a tag with a key of 'Name' and a value that you specify.

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="my-first-igw"/> X
Remove	
Add new tag	

You can add 49 more tags.

Cancel Create internet gateway

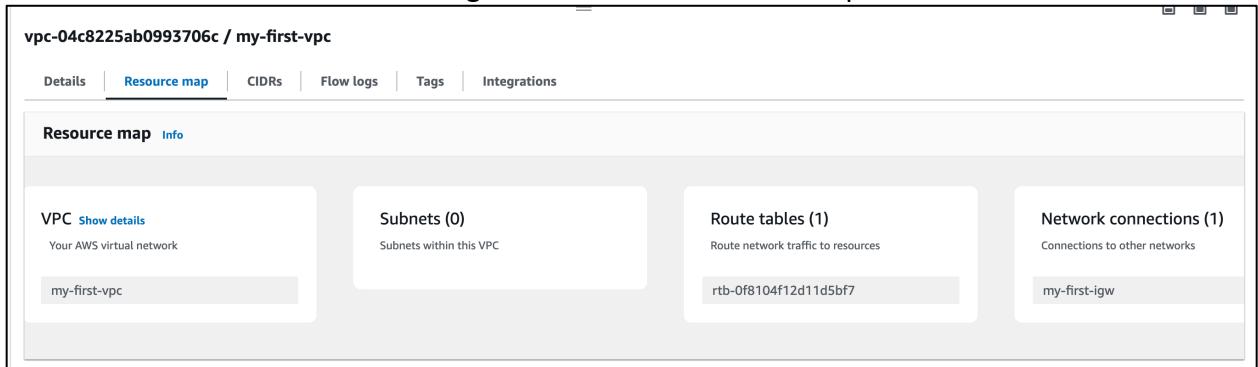
Build AWS Network

- c. Now that we have created an IGW= Internet Gateway, we must attach it to the VPC, just like we link our router to an Internet Port at home.

Click on Attach to VPC; on the next page, select the VPC created and attach it.

Internet gateways (1/2) Info			Actions ▾		Create internet gateway
Name	Internet gateway ID	State			Owner
-	igw-09ac8b7373929f424	Attached	View details	Detach from VPC	50696
my-first-igw	igw-0eda1b8b040963a7b	Detached	Attach to VPC	Manage tags	50696

- d. This is what the network diagram looks like after this setup.



Step 3: Create Public and Private Subnets.

- a. Let's divide our VPC network into public and private subnets.
Example: Think of an office building with different departments like HR, Finance, and IT, each occupying its section of the building. Similarly, subnets in AWS VPC segregate resources based on their function or purpose, ensuring that different parts of your cloud infrastructure remain isolated and secure.
- b. We are creating two public and two private subnets.
- c. Let's create Public Subnet 1

[VPC](#) > [Subnets](#) > Create subnet

Create subnet Info

VPC

VPC ID

Create subnets in this VPC.

vpc-04c8225ab0993706c (my-first-vpc) ▾

Associated VPC CIDRs

IPv4 CIDRs

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.

my-public-subnet-1

The name can be up to 256 characters long.

Beware of the availability zone chosen here.

Build AWS Network

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
<input type="text" value="Name"/> X	<input type="text" value="my-public-subnet-1"/> X	Remove

Add new tag
You can add 49 more tags.
Remove

Add new subnet

- d. Public Subnet 2: Your task is to create a public subnet with the name **my-public-subnet-2**, the CIDR block **10.0.2.0/24**, and the region is **us-east-1b**.
- e. Enable **auto-assign public IPv4 address** option.

my-public-subnet-1

Actions ▲

- Create flow log
- Edit subnet settings**
- Edit IPv6 CIDRs

State	IPv4 C
-------	--------

Build AWS Network

Auto-assign IP settings [Info](#)
Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

Enable auto-assign public IPv4 address [Info](#)

Enable auto-assign customer-owned IPv4 address [Info](#)
Option disabled because no customer owned pools found.

f. Do the above step for the second public subnet.

g. Private Subnet 1:

Subnet 3 of 4

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

▼ Tags - optional

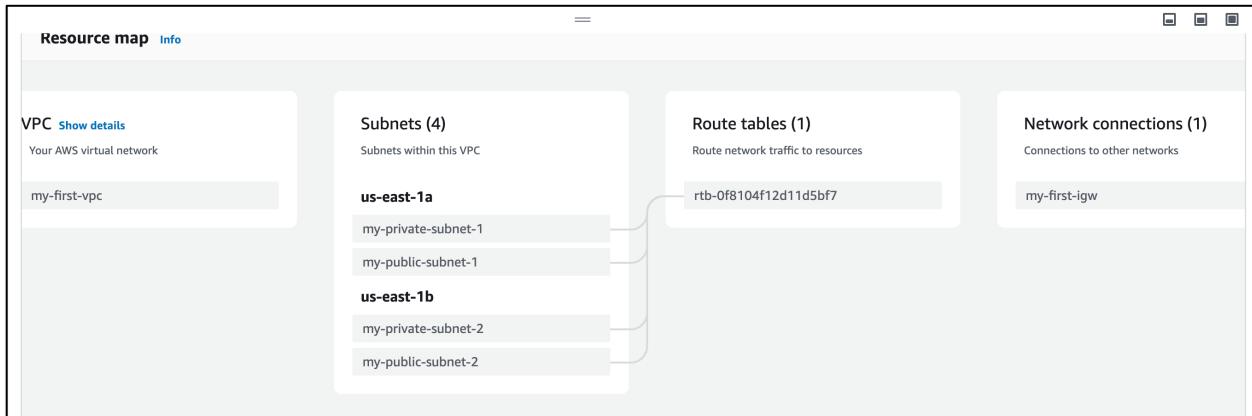
Key	Value - optional	Remove
<input type="text" value="Name"/> <input type="button" value="X"/>	<input type="text" value="my-private-subnet-1"/> <input type="button" value="X"/>	<input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>		

You can add 49 more tags.

h. Private Subnet 2: Your task is to create a private subnet named **my-private-subnet-2**, the CIDR block **10.0.4.0/24**, and the region is **us-east-1b**.

Build AWS Network

- i. After creating the subnets, my network map looks like this.



Step 4: Create Private Route Tables.

- a. After creating the subnets, we should define the networks by making the Route Tables. What is a Route Table, and what is it used for?

Example: In a city, there are different roads leading to various neighborhoods and landmarks. Likewise, route tables in AWS VPC act as digital roadmaps, steering data traffic to other parts of your cloud infrastructure and ensuring efficient communication between services and resources.

Remember that a default route table was created when we created the VPC. We will use it for public communication purposes.

- b. Let's create a Private Route Table.

This private route table allows communication for resources in a private subnet.

VPC > Route tables > Create route table

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.

VPC
The VPC to use for this route table.

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
<input type="text" value="Name"/> X	<input type="text" value="my-private-route-table-1"/> X Remove
Add new tag	

You can add 49 more tags.

Cancel Create route table

- c. Your task is to create another Route Table named **my-private-route-table-2**.
- d. Now, the updated resource map looks like it, as shown below. See the connection between subnets and route tables. The communication or transfer of data still occurs through the default route table. Now, we make the necessary changes to build the perfect network.
- e. Let's review the map after creating the route tables.

Build AWS Network

Your VPCs (1/3) [Info](#)

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
<input checked="" type="checkbox"/> my-first-vpc	vpc-04c8225ab0993706c	Available	10.0.0.0/16	-	dopt-00e21c30c5436b...
<input type="checkbox"/> -	vpc-0fb4c1852e7b23890	Available	172.31.0.0/16	-	dopt-00e21c30c5436b...
<input type="checkbox"/> project-vpc	vpc-0ab974441f15400d9	Available	10.0.0.0/16	-	dopt-00e21c30c5436b...

Resource map [Info](#)

VPC [Show details](#)
Your AWS virtual network
my-first-vpc

Subnets (4)
Subnets within this VPC

- us-east-1a**
 - my-private-subnet-1
 - my-public-subnet-1
- us-east-1b**
 - my-private-subnet-2
 - my-public-subnet-2

Route tables (3)
Route network traffic to resources

- my-private-route-table-1
- my-private-route-table-2
- rtb-0f8104f12d11d5bf7

Network connections (1)
Connections to other networks
my-first-igw

Step 5: Associate route tables to the subnets appropriately.

- e. Since we have the default route table, we want to use it for public communication, and we will make changes in the private route tables alone.
- f. Let's edit the subnet association and add private subnet – 1 to this route table.

VPC > Route tables > rtb-01f44331f86480a9f / my-private-route-table-1 [Actions](#)

Details [Info](#)

Route table ID <input type="checkbox"/> rtb-01f44331f86480a9f	Main <input type="checkbox"/> No	Explicit subnet associations -	Edge associations -
VPC vpc-04c8225ab0993706c my-first-vpc	Owner ID <input type="checkbox"/> 506960532205		

Routes **Subnet associations** **Edge associations** **Route propagation** **Tags**

Explicit subnet associations (0) [Edit subnet associations](#)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
No subnet associations You do not have any subnet associations.			

Build AWS Network

VPC > Route tables > rtb-01f44331f86480a9f > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1/4)					
	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/>	my-private-subnet-1	subnet-0f4deb81b18727418	10.0.3.0/24	-	Main (rtb-0f8104f12d11d5bf7)
<input type="checkbox"/>	my-public-subnet-1	subnet-0221cc7de703d33a8	10.0.10.0/24	-	Main (rtb-0f8104f12d11d5bf7)
<input type="checkbox"/>	my-private-subnet-2	subnet-0ab9d23345d2b6283	10.0.4.0/24	-	Main (rtb-0f8104f12d11d5bf7)
<input type="checkbox"/>	my-public-subnet-2	subnet-02c985fa8331ee838	10.0.2.0/24	-	Main (rtb-0f8104f12d11d5bf7)

Selected subnets

- subnet-0f4deb81b18727418 / my-private-subnet-1 X

[Cancel](#) [Save associations](#)

- g. Your task is to associate the private route table 2 with **my-private-subnet-2**.
- h. Let's review the Resource map now. This is what my map looks like right now. See, my public subnets are still associated with the default route table. So, we don't need to associate anything. But do you think about the communication between public resources and the internet? We have to add a route connecting to the internet gateway.

Your VPCs (1/2) [Info](#)

	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
<input checked="" type="checkbox"/>	my-first-vpc	vpc-04c8225ab0993706c	Available	10.0.0.0/16	-	dopt-00e21c30c5436b..
<input type="checkbox"/>	-	vpc-0fb4c1852e7b23890	Available	172.31.0.0/16	-	dopt-00e21c30c5436b..

Resource map [Info](#)

The Resource map displays the following components:

- VPC:** my-first-vpc (Your AWS virtual network)
- Subnets (4):**
 - us-east-1a: my-private-subnet-1, my-public-subnet-1 (selected)
 - us-east-1b: my-private-subnet-2, my-public-subnet-2 (selected)
- Route tables (3):**
 - my-private-route-table-1
 - my-private-route-table-2
 - rtb-0f8104f12d11d5bf7 (selected)
- Network connections (1):** my-first-igw

Associations shown in the map:

- my-public-subnet-1 is associated with rtb-0f8104f12d11d5bf7
- my-public-subnet-2 is associated with rtb-0f8104f12d11d5bf7

- i. Let's edit the route option by adding the internet gateway to the default route table. Only after adding this route do our public servers placed in the public subnets communicate to the internet.

Build AWS Network

Route tables (1/4) Info

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
my-private-route-table-1	rtb-01f44331f86480a9f	subnet-0f4deb81b18727...	-	No	vpc-04c8225ab0993706c my-first-vpc
my-private-route-table-2	rtb-0966be51bc1bdbf50	subnet-0ab9d23345d2b6...	-	No	vpc-04c8225ab0993706c my-first-vpc
-	rtb-0f8104f12d11d5bf7	-	-	Yes	vpc-04c8225ab0993706c my-first-vpc
-	rtb-02ae7471ccbf80341	-	-	Yes	vpc-0fb4c1852e7b23890

rtb-0f8104f12d11d5bf7

- [Details](#)
- [Routes](#)
- [Subnet associations](#)
- [Edge associations](#)
- [Route propagation](#)
- [Tags](#)

Routes (1)

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No

[VPC](#) > [Route tables](#) > [rtb-0f8104f12d11d5bf7](#) > Edit routes

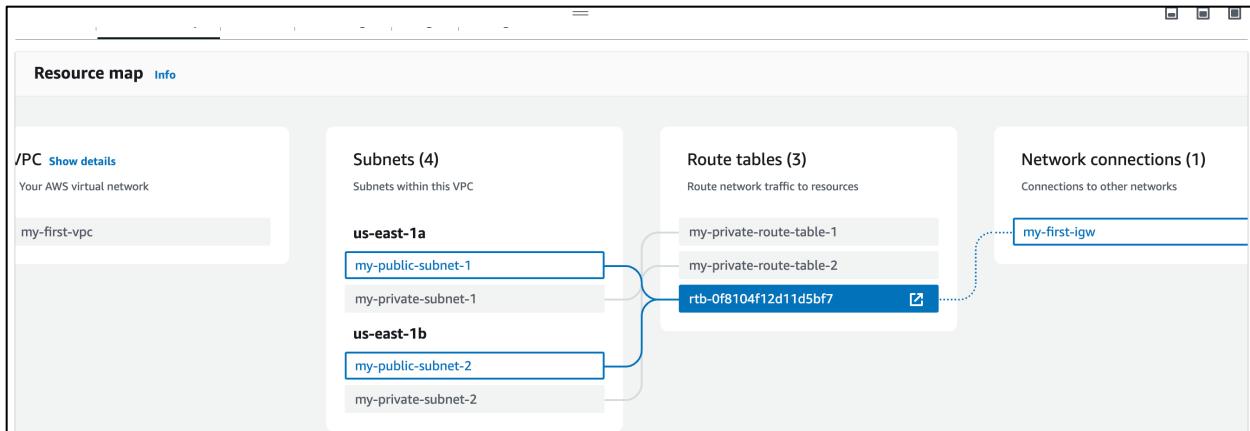
Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
<input type="text" value="0.0.0.0"/> X	<input type="button" value="Add route"/>	<input type="button" value="Remove"/>	<input type="button" value="Preview"/> Save changes
<input type="text" value="0.0.0.0/0"/> X <input type="button" value="Add route"/> <input type="button" value="Remove"/>			

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
<input type="text" value="0.0.0.0"/> X	<input type="button" value="Add route"/>	<input type="button" value="Remove"/>	<input type="button" value="Preview"/> Save changes
<input type="text" value="0.0.0.0/0"/> X <input type="button" value="Add route"/> <input type="button" value="Remove"/>			

j. Let's review our Resource Map again.

Build AWS Network



Note: This is a Deliverable.

Step 6: Test the Connectivity by Creating a Public EC2.

- Create an ec2 server in the **my-public-subnet-1**.

Name: PUB-LINUX

KeyPair: Use the existing one or create one. Refer to folder **SSH_into_EC2**.

VPC: my-first-vpc

Subnet: my-public-subnet-1

Auto Assign public IP: Enable

SG: Create a new security group

Name: my-public-security-group

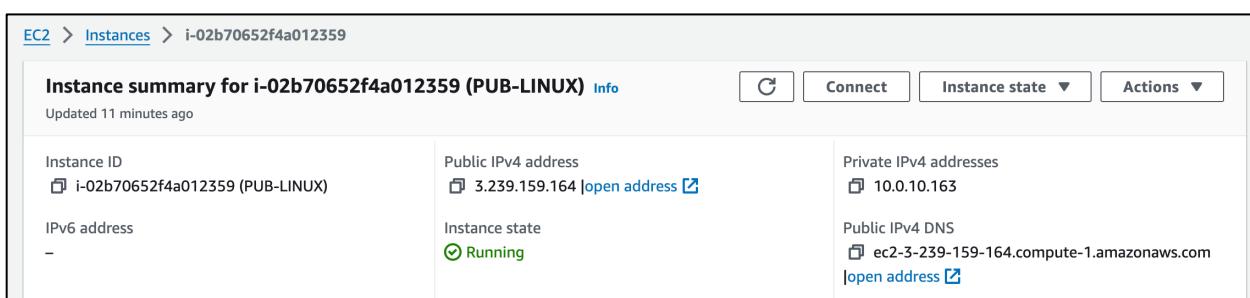
Description: Allow public communication

Type: SSH, Source: Anywhere

Type: All ICMP – IPV4, Source: Anywhere

The rest of the options are default.

Launch the instance.



The screenshot shows the AWS EC2 Instances page. The instance summary for 'i-02b70652f4a012359 (PUB-LINUX)' is displayed. The instance was updated 11 minutes ago. Key details include:

- Instance ID: i-02b70652f4a012359 (PUB-LINUX)
- Public IPv4 address: 3.239.159.164 (with a 'Connect' button)
- Private IPv4 addresses: 10.0.10.163
- IPv6 address: -
- Instance state: Running
- Public IPv4 DNS: ec2-3-239-159-164.compute-1.amazonaws.com (with a 'Connect' button)

Note: This is a Deliverable.

- Login to the ec2 and see if public communication is allowed.

Observe that the public Linux allows communication from the Internet.

My laptop can reach the public server.

```
(base) venkatgirisasanapuri@Venkatgiris-MacBook-Air Downloads % ping 3.239.159.164
PING 3.239.159.164 (3.239.159.164): 56 data bytes
64 bytes from 3.239.159.164: icmp_seq=0 ttl=112 time=68.058 ms
64 bytes from 3.239.159.164: icmp_seq=1 ttl=112 time=43.676 ms
64 bytes from 3.239.159.164: icmp_seq=2 ttl=112 time=52.904 ms
^C
--- 3.239.159.164 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 43.676/54.879/68.058/10.051 ms
(base) venkatgirisasanapuri@Venkatgiris-MacBook-Air Downloads %
```

We can access the internet from the public server.

```
~~~ /'~~~.~/'_/~/'_/~/m/'[ec2-user@ip-10-0-10-163 ~]$ ping google.com
PING google.com (172.253.115.102) 56(84) bytes of data.
64 bytes from bg-in-f102.1e100.net (172.253.115.102): icmp_seq=1 ttl=105 time=2.22 ms
64 bytes from bg-in-f102.1e100.net (172.253.115.102): icmp_seq=2 ttl=105 time=2.29 ms
64 bytes from bg-in-f102.1e100.net (172.253.115.102): icmp_seq=3 ttl=105 time=2.31 ms
^C
```

Step 7: Test the Connectivity by Creating a Private EC2.

- Create an ec2 server in the **my-public-subnet-1**.

Name: PRI-LINUX

KeyPair: Use the existing one or create one. Refer to folder **SSH_int_E2**.

VPC: my-first-vpc

Subnet: my-private-subnet-1

Auto Assign public IP: Disable

SG: Create a new security group

Name: my-private-security-group

Description: Does not allow public communication

Type: SSH, Source: Anywhere

Type: All ICMP – IPV4, Source: Search for **my-public-security-group** created in the last step and select the SG created in the earlier step.

So, that means we are allowing communication from the servers in the public subnet only.

Build AWS Network

The screenshot shows the AWS Security Groups page. A search bar at the top contains the text "my-pub". Below the search bar, a list of security groups is shown, with one item highlighted: "my-pub". A red warning message below the list states: "⚠️ CIDR block, a security group ID or a prefix list has to be specified." Other buttons visible include "Delete" and a dropdown menu.

The rest of the options are default.

Launch the instance.

The screenshot shows the AWS Instances page. A new instance is being launched with the ID "i-02c3489df5eb0b290". The instance summary table includes the following details:

Instance ID	Public IPv4 address	Private IPv4 addresses
i-02c3489df5eb0b290 (PRI-LINUX)	-	10.0.3.92
IPv6 address	Instance state	Public IPv4 DNS
-	Running	-

Note: This is a Deliverable.

- Let's test the ping from our public ec2 server to private ec2.
ping <private-ip of PRI-LINUX>

```
[ec2-user@ip-10-0-10-163 ~]$ ping 10.0.3.92
PING 10.0.3.92 (10.0.3.92) 56(84) bytes of data.
64 bytes from 10.0.3.92: icmp_seq=1 ttl=127 time=0.483 ms
64 bytes from 10.0.3.92: icmp_seq=2 ttl=127 time=0.534 ms
64 bytes from 10.0.3.92: icmp_seq=3 ttl=127 time=0.443 ms
^C
--- 10.0.3.92 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2097ms
rtt min/avg/max/mdev = 0.443/0.486/0.534/0.037 ms
```

Step 8: How do you login to the Private server (i.e., ec2 in the private subnet)

- a. Refer to the other files in the same assignment, which depends on your operating system.
- b. After you log in, check if you can reach the internet. See that we are unable to reach the internet. That is where NAT comes into the picture.

```
[ec2-user@ip-10-0-10-163 ~]$ ssh 10.0.3.92
'      #
~\_  ####_          Amazon Linux 2023
~~ \_#####\
~~   \###|
~~     \#/ ___  https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '-->
~~~           /
~~.._. _/
~/_/
/m/'[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.139) 56(84) bytes of data.
^C
--- google.com ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4162ms
[ec2-user@ip-10-0-3-92 ~]$
```

Step 9: NAT Gateway creation and testing internet connection from the private server.

- a. Do you think about how the servers in the private subnet communicate to the internet when needed? NAT Gateways is the answer. We will place this Nat Gateway in the public subnet.

Let's create one.

Build AWS Network

b. Click on Allocate Elastic IP

VPC > [NAT gateways](#) > Create NAT gateway

Create NAT gateway Info

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

Name - *optional*
Create a tag with a key of 'Name' and a value that you specify.

The name can be up to 256 characters long.

Subnet
Select a subnet in which to create the NAT gateway.

Connectivity type
Select a connectivity type for the NAT gateway.

Public
 Private

Elastic IP allocation ID Info
Assign an Elastic IP address to the NAT gateway.

► Additional settings Info

Connectivity type
Select a connectivity type for the NAT gateway.

Public
 Private

Elastic IP allocation ID Info
Assign an Elastic IP address to the NAT gateway.

► Additional settings Info

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - <i>optional</i>
<input type="text" value="Name"/> <input type="button" value="X"/>	<input type="text" value="my-nat-gateway"/> <input type="button" value="X"/> <input type="button" value="Remove"/>
<input type="button" value="Add new tag"/>	

You can add 49 more tags.

Build AWS Network

- c. Remember, we have added a route from the Main(public) Route Table to the internet gateway. Similarly, we will add a route from the private route table to the Nat gateway.

VPC > Route tables > rtb-01f44331f86480a9f

rtb-01f44331f86480a9f / my-private-route-table-1

Actions ▾

Details	Info
Route table ID rtb-01f44331f86480a9f	Main No
VPC vpc-04c8225ab0993706c my-first-vpc	Owner ID 506960532205
Explicit subnet associations subnet-0f4deb81b18727418 / my-private-subnet-1	
Edge associations -	

Routes Subnet associations Edge associations Route propagation Tags

Both ▾ Edit routes

Filter routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No

VPC > Route tables > rtb-01f44331f86480a9f > Edit routes

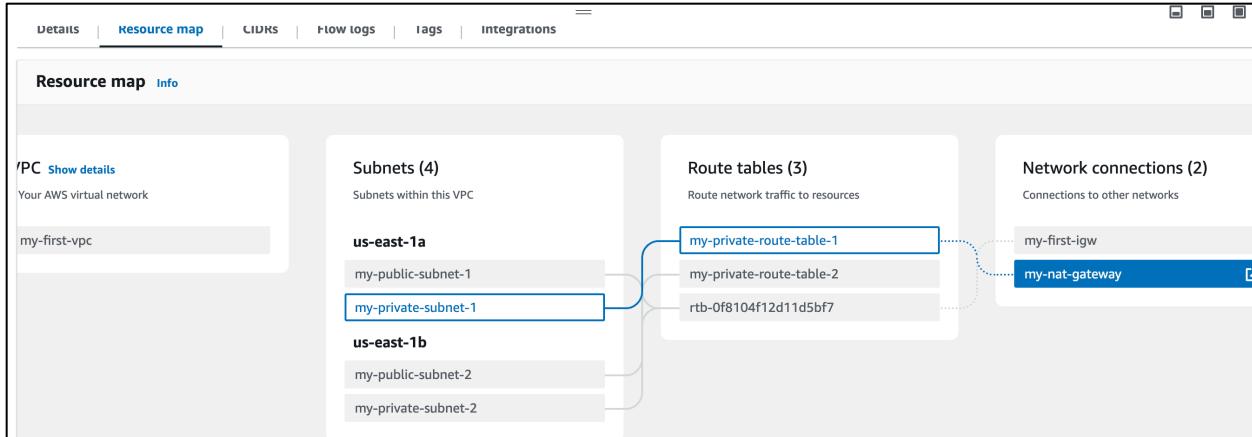
Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	NAT Gateway	-	No

Add route Remove

Cancel Preview Save changes

Build AWS Network



Note: This is a Deliverable.

- d. Once NAT is created, after two minutes, test the same command in the private server.

```
[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.139) 56(84) bytes of data.
^C
--- google.com ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4162ms

[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.101) 56(84) bytes of data.
^C
--- google.com ping statistics ---
19 packets transmitted, 0 received, 100% packet loss, time 18709ms

[ec2-user@ip-10-0-3-92 ~]$ ping google.com
PING google.com (172.253.115.100) 56(84) bytes of data.
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=1 ttl=107 time=3.09 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=2 ttl=107 time=2.54 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=3 ttl=107 time=2.48 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=4 ttl=107 time=2.49 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=5 ttl=107 time=2.46 ms
64 bytes from bg-in-f100.1e100.net (172.253.115.100): icmp_seq=6 ttl=107 time=2.63 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 2.464/2.616/3.085/0.217 ms
[ec2-user@ip-10-0-3-92 ~]$
```

Note: This is a Deliverable. (The private IP in the above screenshot should match the private ip of private ec2)

So, we use a NAT gateway for internet communications to a private server.

Build AWS Network

Deliverables: Submit the screenshots where it mentions “**Note: This is a Deliverable.**” The screenshots should be submitted in the pdf format.

Step 5 – f

Step 6 – a

Step 7 – a

Step 9 – c and d

Note: Delete NAT Gateway and the two EC2 instances created in this lab

Done!!!

