

# AWS VPC Networking with Public and Private Subnets

31-August

—  
By:

Lalit Kumar  
<https://bento.me/lalit192977>

## Overview

In this project, I designed and implemented a **custom Virtual Private Cloud (VPC)** in AWS to demonstrate a secure and scalable networking architecture. The setup includes both **public and private subnets**, each hosting an EC2 instance, and proper routing to enable internet access where required.

### ◆ Key Highlights:

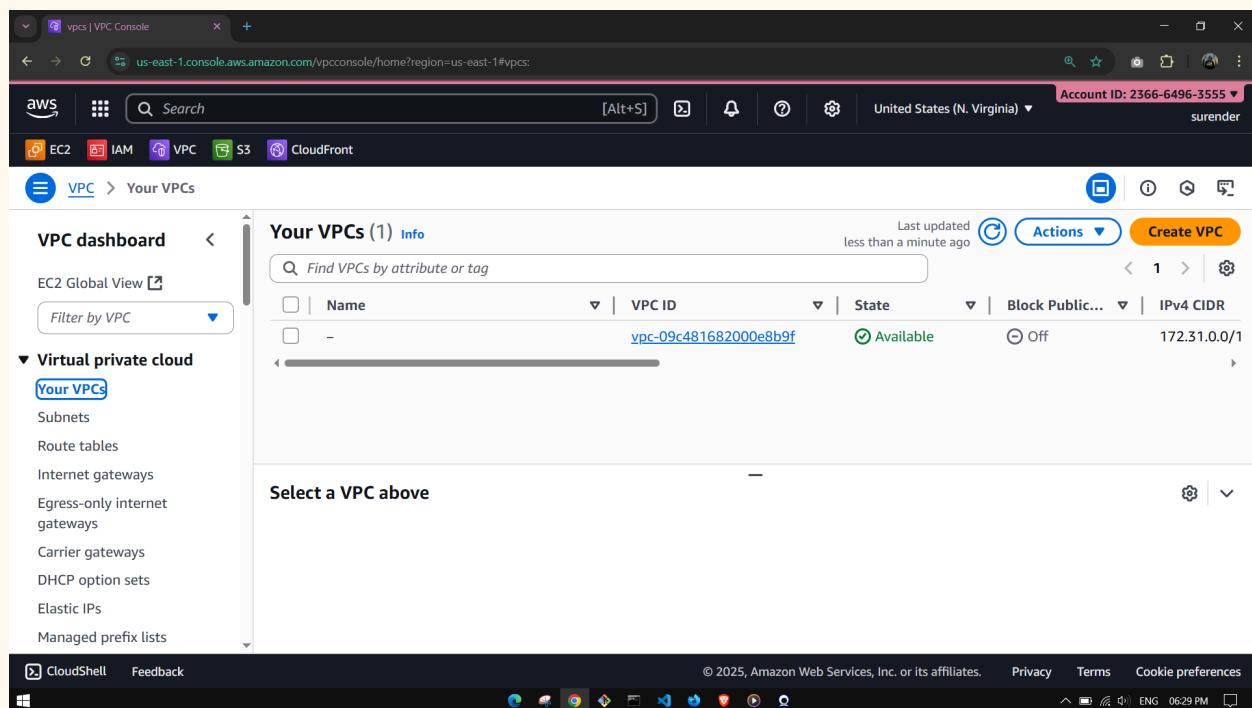
1. **Custom VPC Creation** – A dedicated VPC was created to logically isolate resources within AWS.
2. **Subnet Design** – Two subnets were deployed:
  - **Public Subnet:** Connected to the Internet Gateway for direct internet access.
  - **Private Subnet:** Isolated from direct internet exposure, but enabled outbound internet via a NAT Gateway.
3. **Routing Configuration** –
  - Public subnet route table mapped to the Internet Gateway.
  - Private subnet route table mapped to the NAT Gateway.
4. **EC2 Instances** –
  - A **public EC2 instance** was launched in the public subnet to act as a **bastion/jump server**.
  - A **private EC2 instance** was launched in the private subnet without direct internet exposure.
5. **Secure Connectivity** – Using the public instance as a bastion host, SSH access was established to the private instance. The private instance successfully accessed the internet through the NAT Gateway.

### ◆ Purpose of the Project:

This project demonstrates how to build a **secure AWS network architecture** where sensitive workloads are placed in private subnets, and only controlled access is allowed through a bastion host. Such architectures are commonly used in **production environments** to ensure security, scalability, and controlled resource access.

## Let's Start

Go to vpc service and create a vpc from right top corner



4

Now vpc name and CIDR range/block

The screenshot shows the 'Create VPC' page in the AWS VPC console. The 'VPC only' option is selected under 'Resources to create'. A 'Name tag - optional' field contains 'my-vpc'. Under 'IPv4 CIDR block', the 'IPv4 CIDR manual input' option is selected, and the CIDR block '192.168.0.0/16' is entered.

Now create a subnet from the subnet service of vpc

The screenshot shows the 'Subnets' page in the AWS VPC console. It lists six subnets, all of which are available and associated with the VPC 'vpc-09c481682000e8b9f'. The table includes columns for Name, Subnet ID, State, and VPC.

Name	Subnet ID	State	VPC
-	subnet-09f9533c333b4ab5a	Available	vpc-09c481682000e8b9f
-	subnet-0ebbd2f340e4e84f1	Available	vpc-09c481682000e8b9f
-	subnet-06b0ccae5c0b9adeb	Available	vpc-09c481682000e8b9f
-	subnet-03f5cc9b66a30f772	Available	vpc-09c481682000e8b9f

5

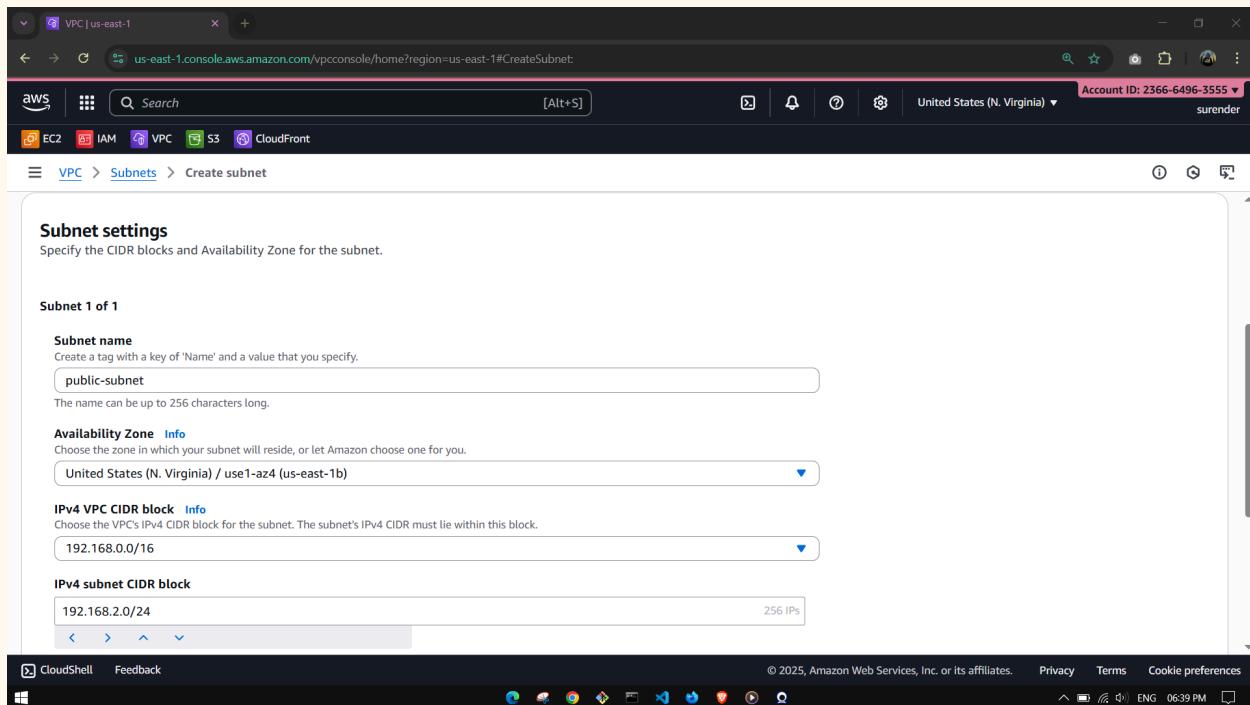
Select your vpc which you created before and create a private subnet

The screenshot shows the AWS VPC console interface for creating a new subnet. At the top, the URL is `us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateSubnet`. The navigation bar includes tabs for EC2, IAM, VPC, S3, and CloudFront. Below the navigation bar, the breadcrumb trail shows `VPC > Subnets > Create subnet`. The main content area is titled "Create subnet" with an "Info" link. It has two main sections: "VPC" and "Subnet settings". In the "VPC" section, the "VPC ID" dropdown is set to "vpc-0c861956942502e15 (my-vpc)". Under "Associated VPC CIDRs", the IPv4 CIDR is listed as "192.168.0.0/16". The "Subnet settings" section is expanded, showing the instruction "Specify the CIDR blocks and Availability Zone for the subnet." Below this, the "Subnet 1 of 1" configuration is shown. It includes fields for "Subnet name" (set to "private-subnet"), "Availability Zone" (set to "United States (N. Virginia) / us-east-1a (us-east-1a)"), and "IPv4 subnet CIDR block" (set to "192.168.1.0/24"). The status bar at the bottom indicates "CloudShell Feedback" and the system status.

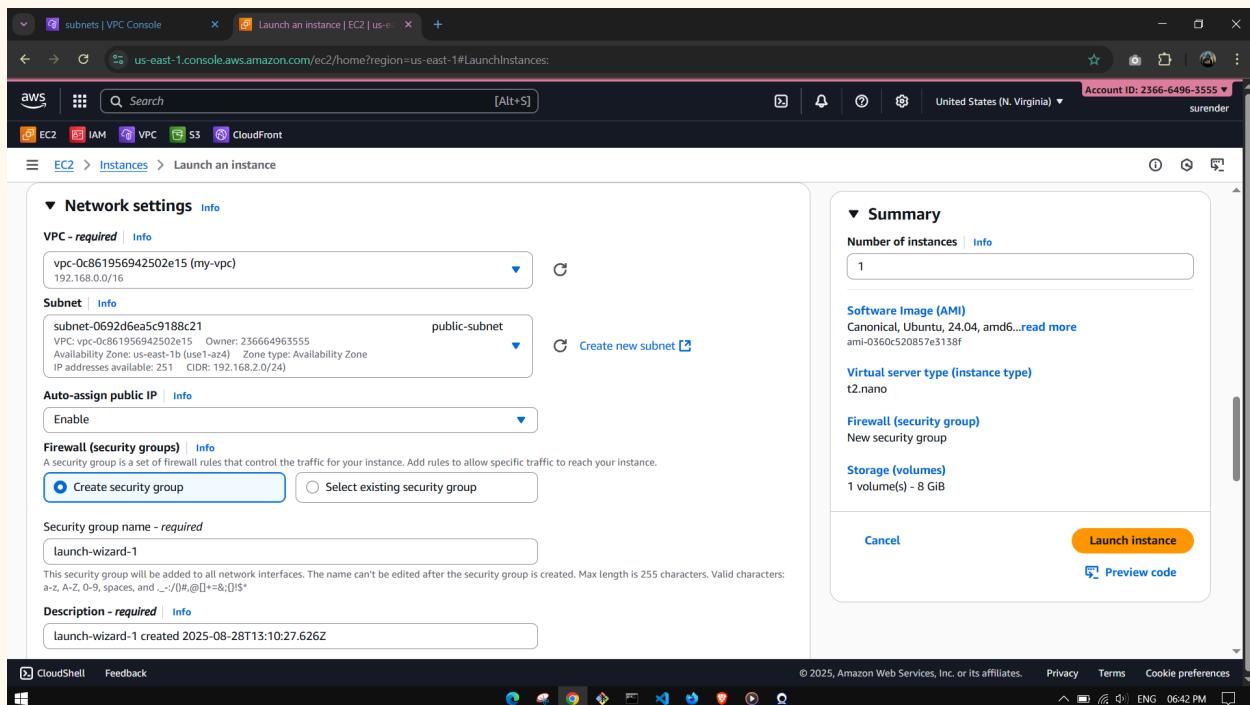
Now give a name and CIDR range

This screenshot continues from the previous one, focusing on the "Subnet 1 of 1" configuration. The "Subnet name" is set to "private-subnet". The "Availability Zone" dropdown shows "United States (N. Virginia) / us-east-1a (us-east-1a)". The "IPv4 subnet CIDR block" dropdown is set to "192.168.1.0/24". A note below the CIDR block states "256 IPs". The status bar at the bottom indicates "CloudShell Feedback" and the system status.

Now create another public subnet with a name and CIDR block



Now After creating public and private subnet create an instance in your public subnet



Select vpc which we created first, select public subnet and enable public ip to your instance

Allow icmp from anywhere in inbound rule of your security group and launch instance

**Inbound Security Group Rules**

- Security group rule 1 (TCP, 22, 0.0.0.0/0)**
  - Type: ssh
  - Protocol: TCP
  - Port range: 22
  - Source type: Anywhere
  - Description - optional: e.g. SSH for admin desktop
- Security group rule 2 (ICMP, All, 0.0.0.0/0)**
  - Type: All ICMP - IPv4
  - Protocol: ICMP
  - Port range: All
  - Source type: Anywhere
  - Description - optional: e.g. SSH for admin desktop

**Summary**

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd64... [read more](#)  
ami-0360c520857e5138f

Virtual server type (instance type): t2.nano

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

[Launch instance](#) [Preview code](#)

Try to connect your instance and you see you are not able to connect to your instance this happens because you are not provided because any traffic is not allowed to your instance and for that we have to attach **internet gateway** and assign a **route table**.

**Connect** [Info](#)

Connect to an instance using the browser-based client.

**EC2 Instance Connect** [Session Manager](#) [SSH client](#) [EC2 serial console](#)

**Instance is not in public subnet**  
Associated subnet [subnet-0692d6ea5c9188c21 \(public-subnet\)](#) is not a public subnet.  
To use EC2 Instance Connect, your instance must be in a public subnet. [To make the subnet a public subnet, add a route in the subnet route table to an internet gateway.](#)

**Connection type**

**Public IPv4 address**  
Connect using a public IPv4 or IPv6 address  
 **Private IP**  
Connect using a private IP address and a VPC endpoint

**Public IPv4 address**  
54.208.14.78

**IPv6 address**  
-

**Username**  
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.  
ubuntu

So first create **Internet Gateway** and attach to your vpc you created

VPC dashboard < Details Info

Internet gateway ID: **igw-0993faaa097036e56** State: **Detached** VPC ID: **-** Owner: **236664963555**

**Tags**

Key	Value
Name	my-vpc-gateway

Actions: **Attach to VPC**, **Detach from VPC**, **Manage tags**, **Delete**

VPC

Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs

Attach the internet gateway to this VPC.

vpc-0c861956942502e15  
Use: "vpc-0c861956942502e15"  
**vpc-0c861956942502e15 - my-vpc**

Cancel **Attach internet gateway**

Now create a public **Route Table** for your vpc

**Route table settings**

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

vpc-public-route-table

**VPC**  
The VPC to use for this route table.

vpc-0c861956942502e15 (my-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** Name **Value - optional** vpc-public-route-table **Add new tag**

You can add 49 more tags.

Create route table

Associate your **public subnet** to your **public route table**

**Edit subnet associations**  
Change which subnets are associated with this route table.

**Available subnets (1/2)**

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table
private-subnet	subnet-052690ca458e4f826	192.168.1.0/24	-	Main (rtb-02)
<input checked="" type="checkbox"/> public-subnet	subnet-0692d6ea5c9188c21	192.168.2.0/24	-	Main (rtb-02)

**Selected subnets**

subnet-0692d6ea5c9188c21 / public-subnet

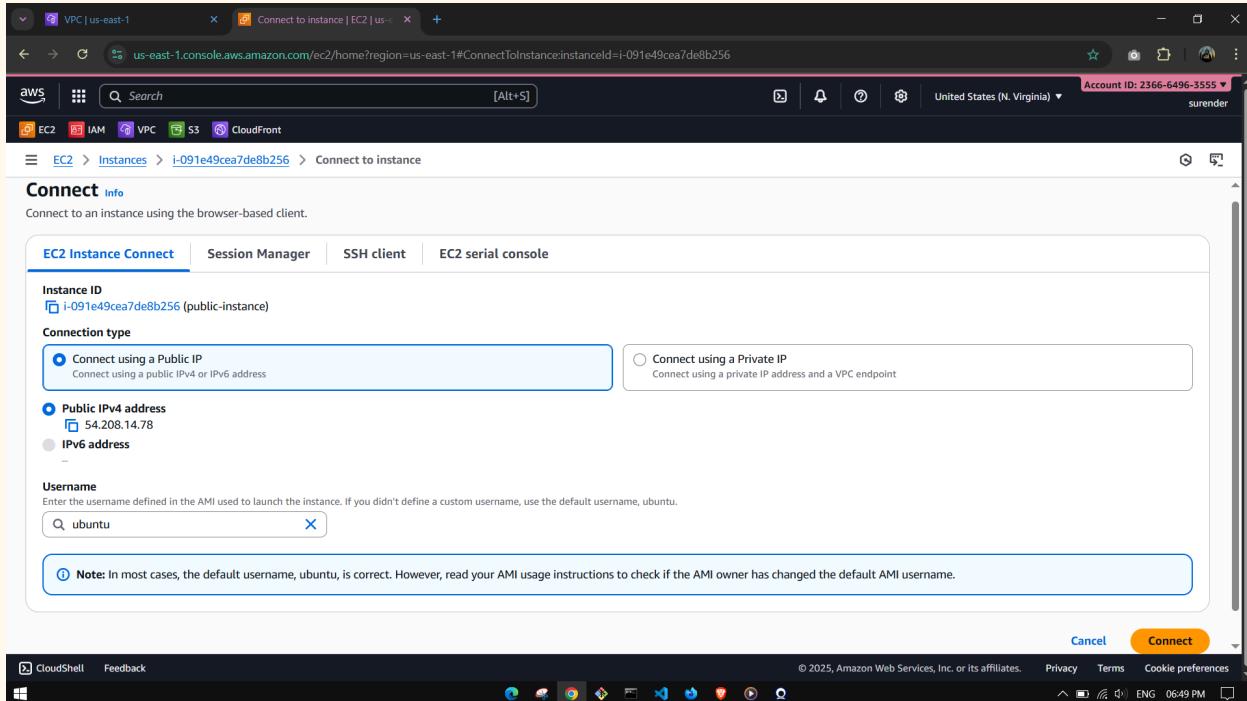
Save associations Cancel

Create a route in your public route table for your internet gateway

The screenshot shows the AWS VPC console interface. On the left, there's a navigation sidebar with options like VPC dashboard, EC2 Global View, Virtual private cloud (Your VPCs, Subnets, Route tables), Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, and Route servers. The 'Route tables' section is currently selected. The main content area displays the details of a route table named 'rtb-027121fef10f35586'. It shows the route table ID, VPC (my-vpc), Main status (No), Owner ID (236664963555), and explicit subnet associations (subnet-0692d6ea5c9188c21 / public-subnet). Below this, the 'Routes' tab is selected, showing one route entry: Destination 192.168.0.0/16, Target local, Status Active, Propagated No, and Route Origin Create Route Table. There are buttons for Actions, Edit routes, and a search bar.

This screenshot shows the 'Edit routes' dialog for the same route table. It lists the existing route (Destination 192.168.0.0/16, Target local, Status Active, Propagated No, Route Origin CreateRouteTable) and allows adding new routes. A new route is being added with Destination 0.0.0.0/0, Target Internet Gateway, and Status Active. The target dropdown shows 'local' and 'Internet Gateway'. A search bar shows 'igw-' and a suggestion 'igw-0993faaa097036e56 (my-vpc-gateway)'. At the bottom, there are buttons for Add route, Remove, Cancel, Preview, and Save changes.

And now you are able to connect your public instance which you created in the public subnet of your VPC and you can run the internet in your instance

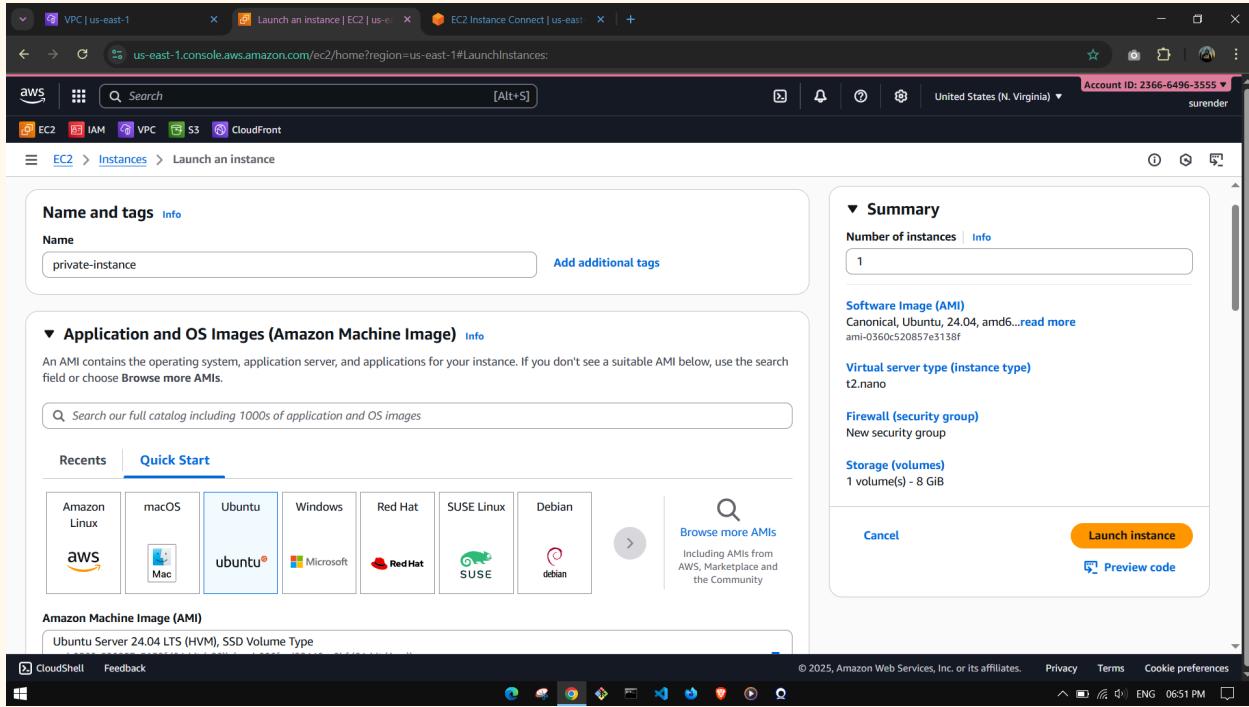


```
root@ip-192-168-2-206:/home/ubuntu# apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu/noble/main amd64 Packages [1106 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble/multiverse amd64 c-n-f Metadata [8328 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/main amd64 Packages [1374 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/main Translation-en [272 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/main amd64 Components [175 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/main amd64 c-n-f Metadata [14.9 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/universe amd64 Packages [1122 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/universe Translation-en [288 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/universe amd64 Components [377 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/universe amd64 c-n-f Metadata [27.7 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/restricted amd64 Packages [1773 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/restricted Translation-en [395 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/restricted amd64 Components [212 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/noble-updates/multiverse amd64 Packages [33.2 kB]
```

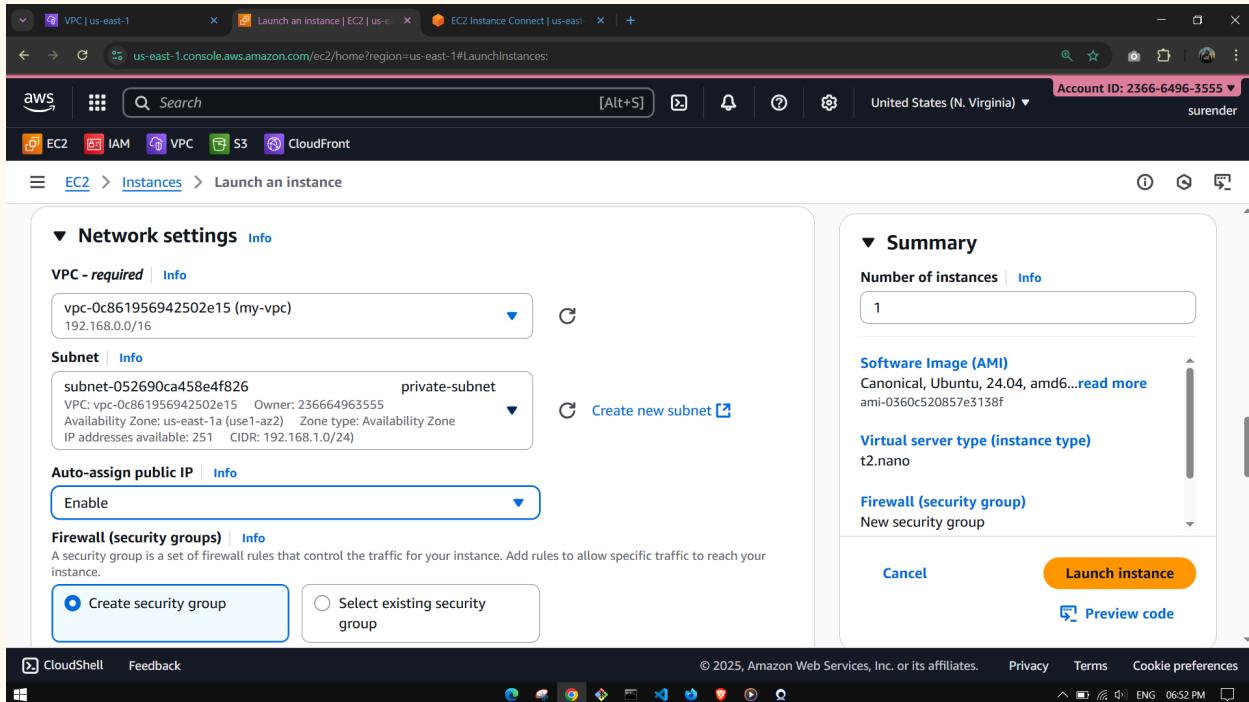
i-091e49cea7de8b256 (public-instance)

PublicIPs: 54.208.14.78 PrivateIPs: 192.168.2.206

Now create another instance in your private subnet (i.e. private instance)



Select your vpc and private subnet and assign public ip to your instance



Allow icmp rule from anywhere and launch instance

The screenshot shows the AWS EC2 Instances Launch an instance page. In the Inbound Security Group Rules section, there are two rules:

- Security group rule 1 (TCP, 22, 0.0.0.0/0)**: Type: ssh, Protocol: TCP, Port range: 22, Source type: Anywhere.
- Security group rule 2 (ICMP, All, 0.0.0.0/0)**: Type: All ICMP - IPv4, Protocol: ICMP, Port range: All, Source type: Anywhere.

On the right side, the Summary panel shows:

- Number of instances**: 1
- Software Image (AMI)**: Canonical, Ubuntu, 24.04, amd64...read more
- Virtual server type (instance type)**: t2.nano
- Firewall (security group)**: New security group
- Storage (volumes)**: 1 volume(s) - 8 GiB

At the bottom right, there are **Launch instance** and **Preview code** buttons.

Try to connect your instance

The screenshot shows the AWS EC2 Instances Connect to instance page for the instance ID i-077e255c3f05a3c64. The Connect tab is selected.

A warning message in a box states: "Instance is not in public subnet. Associated subnet subnet-052690ca458e4f826 (private-subnet) is not a public subnet. To use EC2 Instance Connect, your instance must be in a public subnet. To make the subnet a public subnet, add a route in the subnet route table to an internet gateway." Below this, the Instance ID is listed as i-077e255c3f05a3c64 (private-instance).

The Connection type section has two options:
 

- Connect using a Public IP: Connect using a public IPv4 or IPv6 address.
- Connect using a Private IP: Connect using a private IP address and a VPC endpoint.

Below these options, there are radio buttons for "Public IPv4 address" (selected) and "IPv6 address".

At the bottom, there is a "Username" field and standard browser navigation buttons.

Your are not able to connect your instance directly because private subnet is not connected to the internet gateway and does not have any route table for this private subnet

So create a route table for this private subnet

**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"/>	<input type="text" value="vpc-private-route-table"/>

**Create route table**

**rtb-08717e650031c43d5 / vpc-private-route-table**

**Details**

<b>Route table ID</b> <input type="text" value="rtb-08717e650031c43d5"/>	<b>Main</b> <input type="checkbox"/> No	<b>Explicit subnet associations</b> -	<b>Edge associations</b> -
<b>VPC</b> <input type="text" value="vpc-0c861956942502e15   my-vpc"/>	<b>Owner ID</b> <input type="text" value="236664963555"/>		

**Routes (1)**

Destination	Target	Status	Propagated	Route Origin
192.168.0.0/16	local	<span style="color: green;">Active</span>	No	<a href="#">Create Route Table</a>

From the subnet associations tab associate your private subnet to your private route table from the button **edit subnet associations**

**VPC dashboard**

**Route table ID:** rtb-08717e650031c43d5

**VPC:** my-vpc

**Subnet associations:** 0

**Explicit subnet associations:** 0

**Edge associations:** 0

**Routes:** 0

**Subnet associations:** (Selected)

**Edge associations:**

**Route propagation:**

**Tags:**

**Explicit subnet associations (0):**

No subnet associations

You do not have any subnet associations.

**Edit subnet associations**

Change which subnets are associated with this route table.

**Available subnets (1/2):**

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
private-subnet	subnet-052690ca458e4f826	192.168.1.0/24	-	Main (rtb-024fc3d795b0a150d)
public-subnet	subnet-0692d6ea5c9188c21	192.168.2.0/24	-	rtb-027121fef10f35586 / vpc-pub

**Selected subnets:**

subnet-052690ca458e4f826 / private-subnet

**Buttons:**

- Cancel
- Save associations

Select your private instance and try to ping it with your public instance

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, and Elastic Block Store. The main area displays two instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
public-instance	i-091e49cea7de8b256	Running	t2.nano	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1b
private-instance	i-077e255c3f05a3c64	Running	t2.nano	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1a

Below the table, the details for the 'private-instance' are shown:

- Instance summary:** Instance ID: i-077e255c3f05a3c64, Public IPv4 address: 52.204.76.158, Instance state: Running, Hostname type: IP name: ip-192-168-1-124.ec2.internal.
- Public IPv4 address:** 52.204.76.158
- Private IPv4 addresses:** 192.168.1.124
- Public DNS:** -

Here you can see instances can communicate internally with each other even if they are in different subnets and even if they have not internet connection.

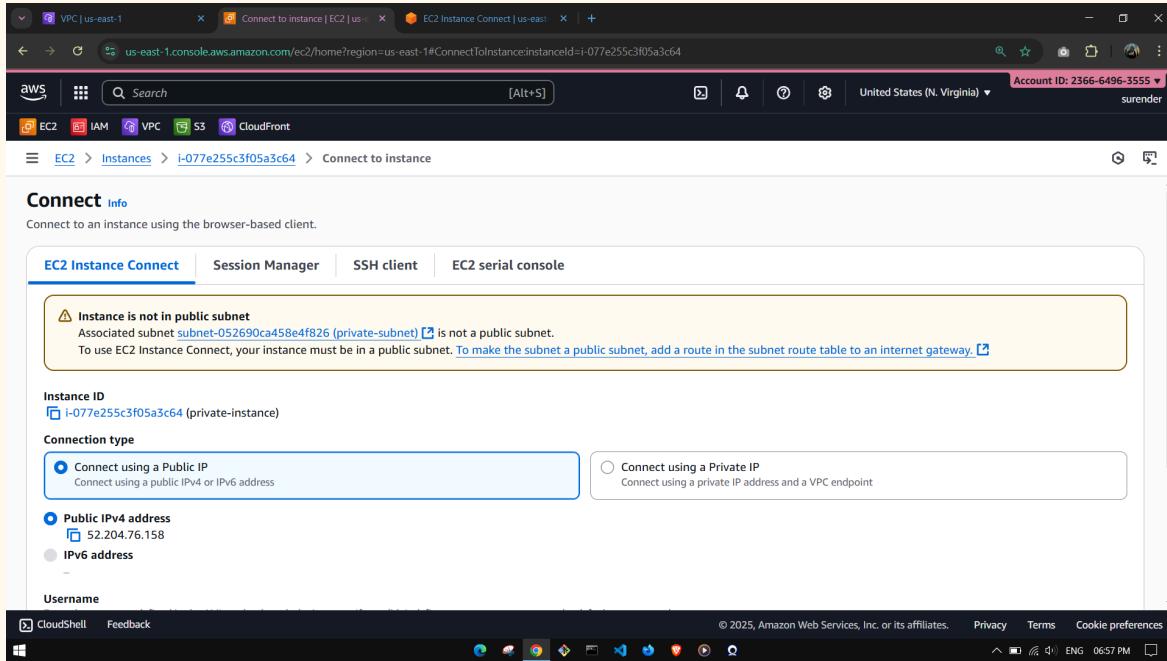
The screenshot shows the AWS CloudShell terminal window. The terminal output is as follows:

```
root@ip-192-168-2-206:/home/ubuntu# ping 192.168.1.124
PING 192.168.1.124 (192.168.1.124) 56(84) bytes of data.
64 bytes from 192.168.1.124: icmp_seq=1 ttl=64 time=0.600 ms
64 bytes from 192.168.1.124: icmp_seq=2 ttl=64 time=0.666 ms
```

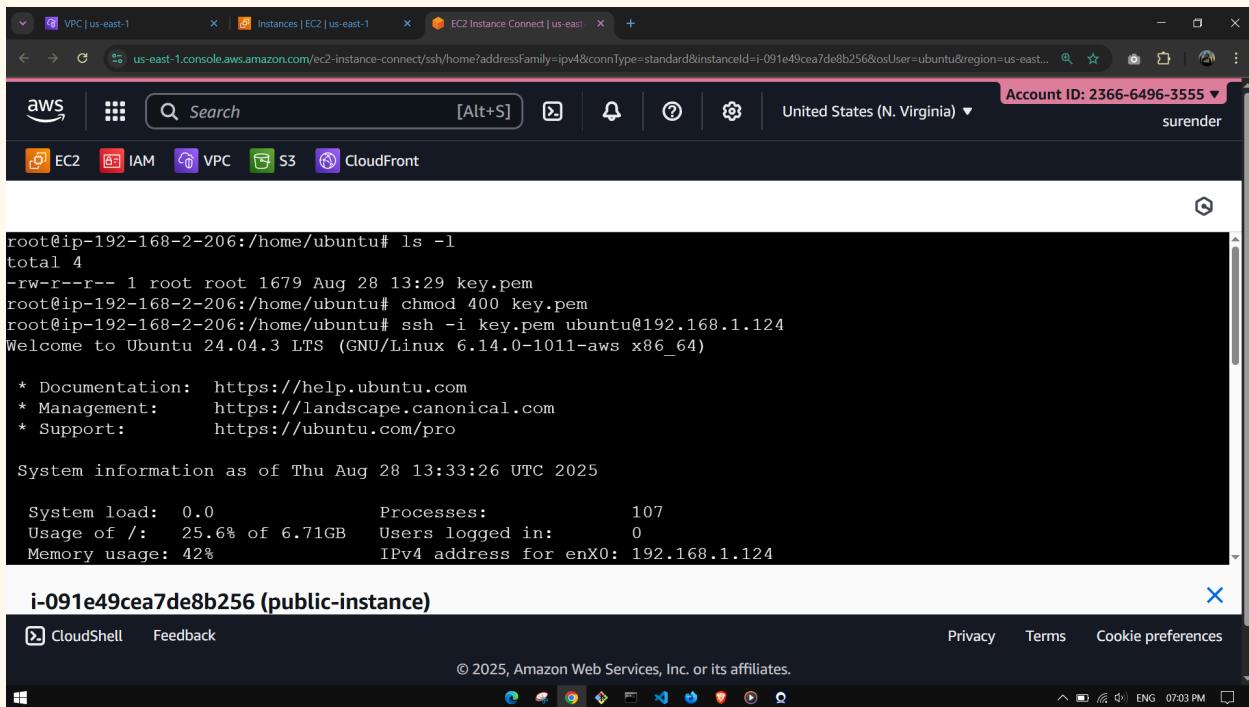
At the bottom of the terminal, it says:

i-091e49cea7de8b256 (public-instance)  
PublicIPs: 54.208.14.78 PrivateIPs: 192.168.2.206

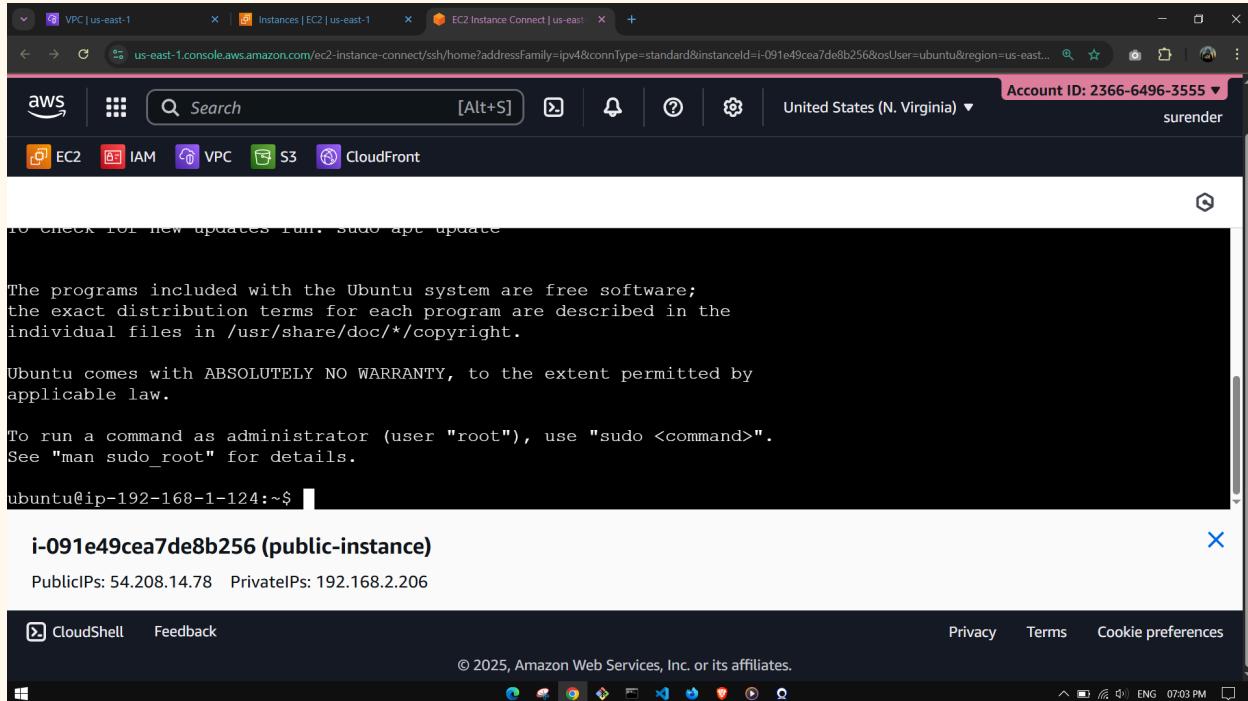
But you can't ssh your private instance because it has only internal network to communicate you can see clearly that you can not ssh directly because it don't have network connection with you



But there is a way to ssh your private instance, you can ssh your private instance with your public instance, firstly change the permission of your key as shown in the image and try to ssh it



You can clearly see, you have done ssh connection with the private instance with your public instance, now this public instance is called jump server or bastien host



```

to check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-192-168-1-124:~$ 

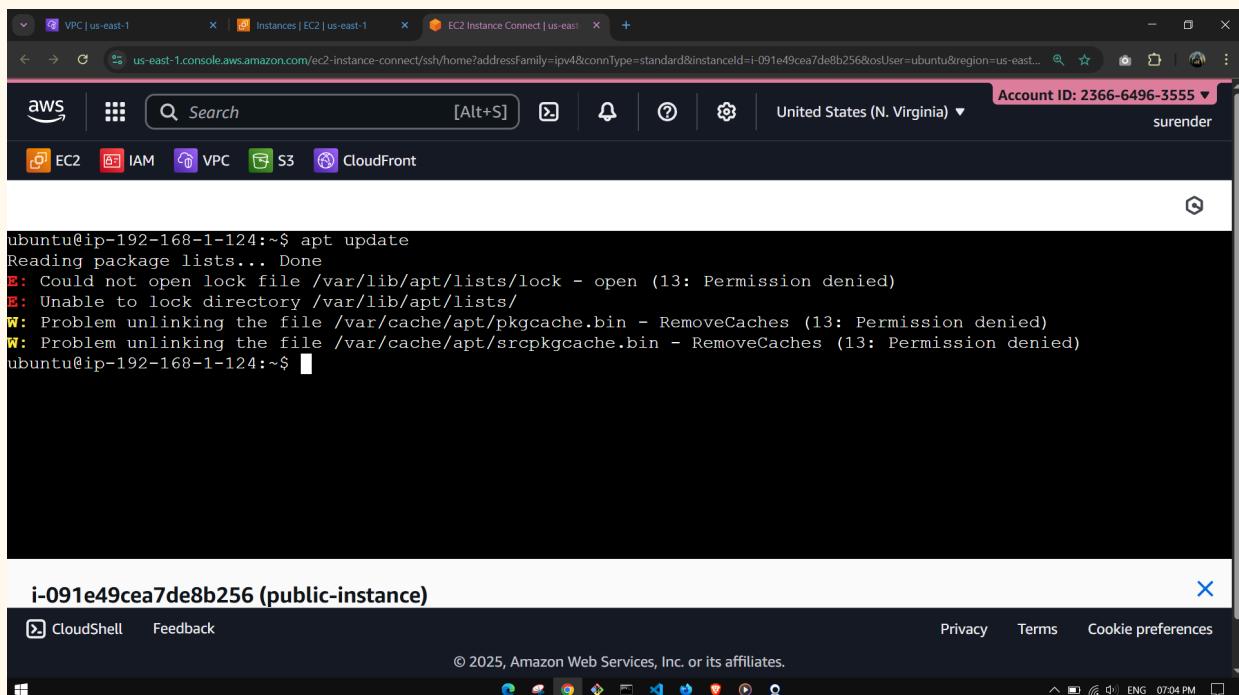
```

**i-091e49cea7de8b256 (public-instance)**

Public IPs: 54.208.14.78 Private IPs: 192.168.2.206

CloudShell Feedback Privacy Terms Cookie preferences © 2025, Amazon Web Services, Inc. or its affiliates. ENG 07:03 PM

Your are able to ssh your private instance from your public instance due to internal connectivity but it does not have internet connection as you can see in the below image



```

ubuntu@ip-192-168-1-124:~$ apt update
Reading package lists... Done
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)
E: Unable to lock directory /var/lib/apt/lists/
W: Problem unlinking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permission denied)
W: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveCaches (13: Permission denied)
ubuntu@ip-192-168-1-124:~$ 

```

**i-091e49cea7de8b256 (public-instance)**

CloudShell Feedback Privacy Terms Cookie preferences © 2025, Amazon Web Services, Inc. or its affiliates. ENG 07:04 PM

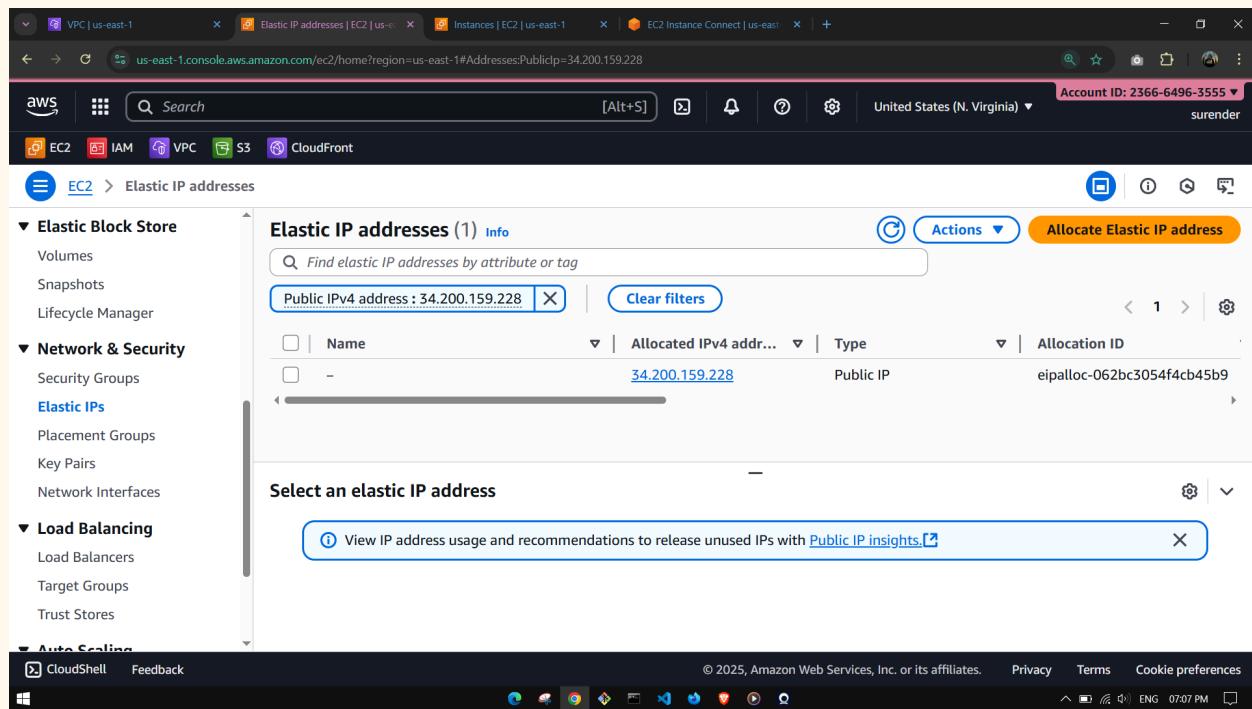
For the internet connection in your private instance you have two common option

1. Through the NAT instance
2. Through the NAT gateway

We will use NAT gateway because it is simple and easy. AWS manages the NAT gateway otherwise we have to manage the NAT instance which may be very serious task and AWS suggest to use NAT gateway because it is secure and managed by AWS itself. No need to take haptic.

Go to NAT gateway and create one for your private instance for secure internet connection in your private instance.

But firstly we have to create an **Elastic IP** for your NAT Gateway. From the EC2 service create an Elastic ip



The screenshot shows the AWS EC2 console interface. The left sidebar navigation includes 'Elastic Block Store', 'Network & Security' (with 'Elastic IPs' selected), 'Load Balancing', and 'Auto Scaling'. The main content area is titled 'Elastic IP addresses (1) info'. It displays a table with one row, showing a Public IPv4 address of 34.200.159.228, Allocated IPv4 address of 34.200.159.228, Type as Public IP, and Allocation ID as eipalloc-062bc3054f4cb45b9. Below the table is a section titled 'Select an elastic IP address' with a note about using Public IP insights.

NAT gateways Info

Name	NAT gateway ID	Connectivity...	State
No NAT gateways found			

Select a NAT gateway

Attach your elastic ip which you created before creating you NAT Gateway

Elastic IP allocation ID: eipalloc-062bc3054f4cb45b9

Allocate Elastic IP

**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
Name	NAT-Private-Subnet

Add new tag

You can add 49 more tags.

Create NAT gateway

Now go your private route table and add one more route for your NAT Gateway

Target should be your NAT Gateway and select your NAT gateway and click to save changes.

This is your output after adding route for NAT gateway

The screenshot shows the AWS VPC Route Tables console. On the left, there's a sidebar for 'Virtual private cloud' with 'Route tables' selected. The main area displays the details for route table 'rtb-08717e650031c43d5'. It shows the 'Main' status is 'No', it's associated with VPC 'vpc-0c861956942502e15 | my-vpc', and has an explicit subnet association to 'subnet-052690ca458e4f826 / private-subnet'. The 'Routes' tab is selected, showing two routes:

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	<a href="#">nat-03d2db4553e10...</a>	Active	No	Create Route
192.168.0.0/16	local	Active	No	Create Route Table

Now you are able to run the internet in your private instance. For confirmation try to ping google as shown below.

The screenshot shows the AWS EC2 Instance Connect terminal. It displays the output of a 'ping' command from a root shell on an Ubuntu instance:

```
root@ip-192-168-1-124:/home/ubuntu# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=1.95 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=0.932 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=1.04 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=1.02 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=0.986 ms
```

Below the terminal window, the instance ID 'i-091e49cea7de8b256 (public-instance)' and its public and private IP addresses ('PublicIPs: 54.208.14.78 PrivateIPs: 192.168.2.206') are displayed.

## *Let's Connect :-*

**Lalit Kumar**

Follow: [LinkedIn](#)

Profile: [Portfolio](#)