VPC Connectivity

By Haroon Zaman | November 2025

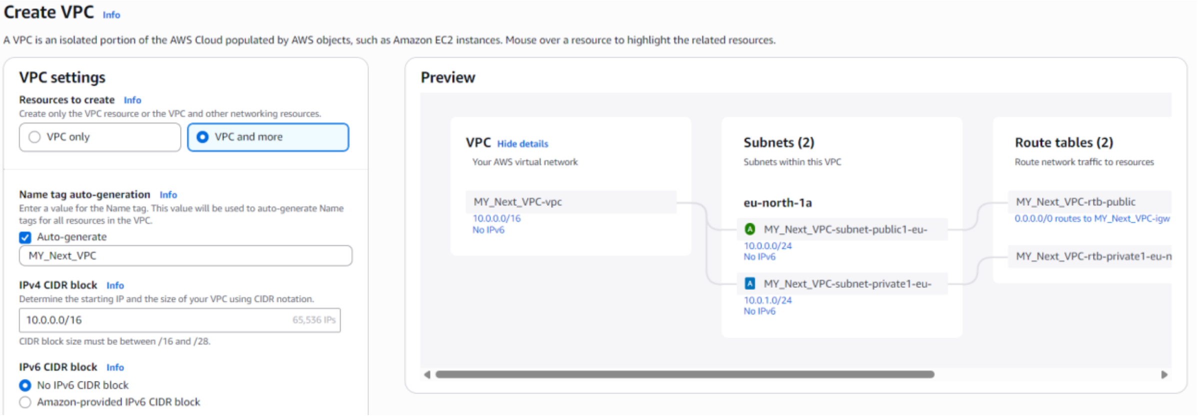
Introduction

In this project, I focused on understanding how different parts of my VPC communicate with each other and with the internet. After creating both public and private EC2 instances, this project helped me test real-world connectivity scenarios inside a cloud network.

The goal was to practice how traffic flows inside a VPC, how instances interact across subnets, and how public resources reach the internet. This is an important step in mastering AWS networking and troubleshooting cloud connectivity issues.

In this project, I completed the following tasks:

- **Connected to my Public EC2 Instance from the AWS Management Console** – to confirm that the instance has proper public access and my security group is configured correctly.
- **Tested Connectivity Between EC2 Instances** – to see how the public EC2 communicates with the private EC2 using internal IPs and security group rules.
- **Tested VPC Connectivity with the Internet** – to verify that internet-facing resources can reach the internet and respond properly.



Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances. Mouse over a resource to highlight the related resources.

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 auto-generation Info
Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.
☑ Auto-generate
MY_Next_VPC

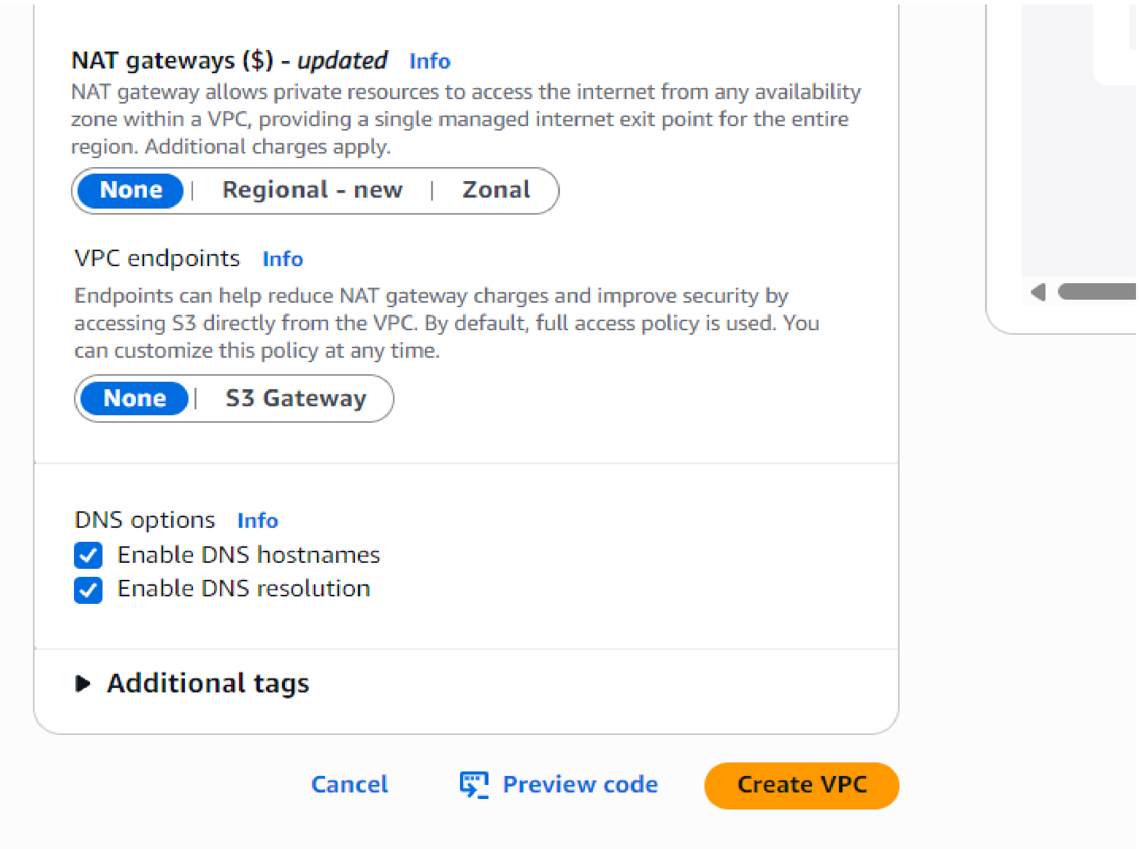
IPv4 CIDR block Info
Determine the starting IP and the size of your VPC using CIDR notation.
10.0.0.0/16 65,536 IPs
CIDR block size must be between /16 and /28.

IPv6 CIDR block Info
● No IPv6 CIDR block
○ Amazon-provided IPv6 CIDR block

Creating a VPC Using “VPC and More”

- I created a new VPC using the **VPC and More** option, which automatically generated **one public subnet** and **one private subnet**.
- The **public subnet** was routed to the internet through the Internet Gateway created by the wizard.
- I selected **No NAT Gateway** because my **private subnet should NOT access the internet**.

- I chose **No VPC Endpoints** since this project didn't require private access to AWS services like S3.



NAT gateways ($) - *updated* Info

NAT gateway allows private resources to access the internet from any availability zone within a VPC, providing a single managed internet exit point for the entire region. Additional charges apply.

[✓ None] | Regional - new | Zonal

VPC endpoints Info

Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.

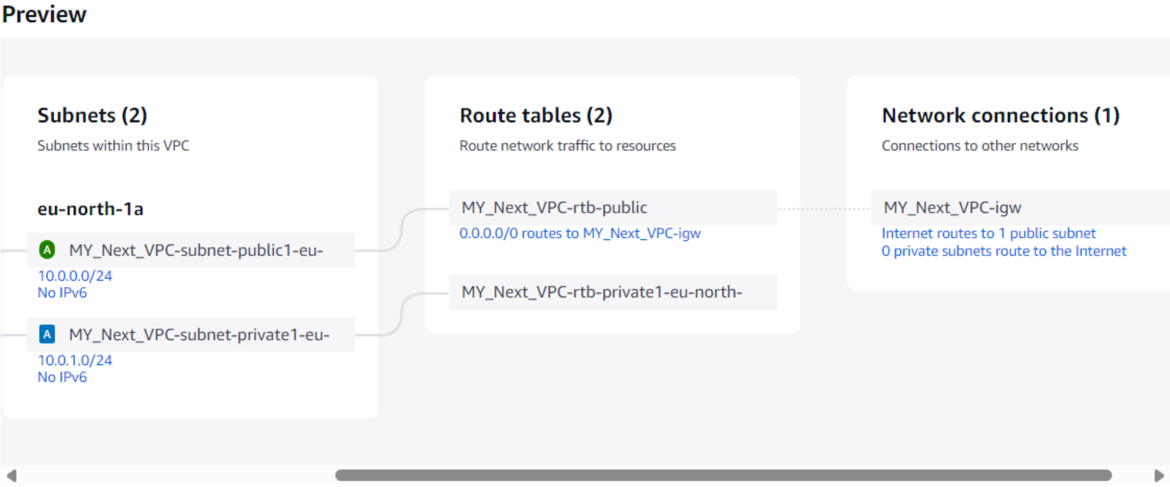
[✓ None] | S3 Gateway

DNS options Info

☑ Enable DNS hostnames

☑ Enable DNS resolution

▶ Additional tags



Preview

Subnets (2)

Subnets within this VPC

eu-north-1a

MY_Next_VPC-subnet-public1-eu-
10.0.0.0/24
No IPv6

MY_Next_VPC-subnet-private1-eu-
10.0.1.0/24
No IPv6

Route tables (2)

Route network traffic to resources

MY_Next_VPC-rtb-public
0.0.0.0/0 routes to MY_Next_VPC-igw

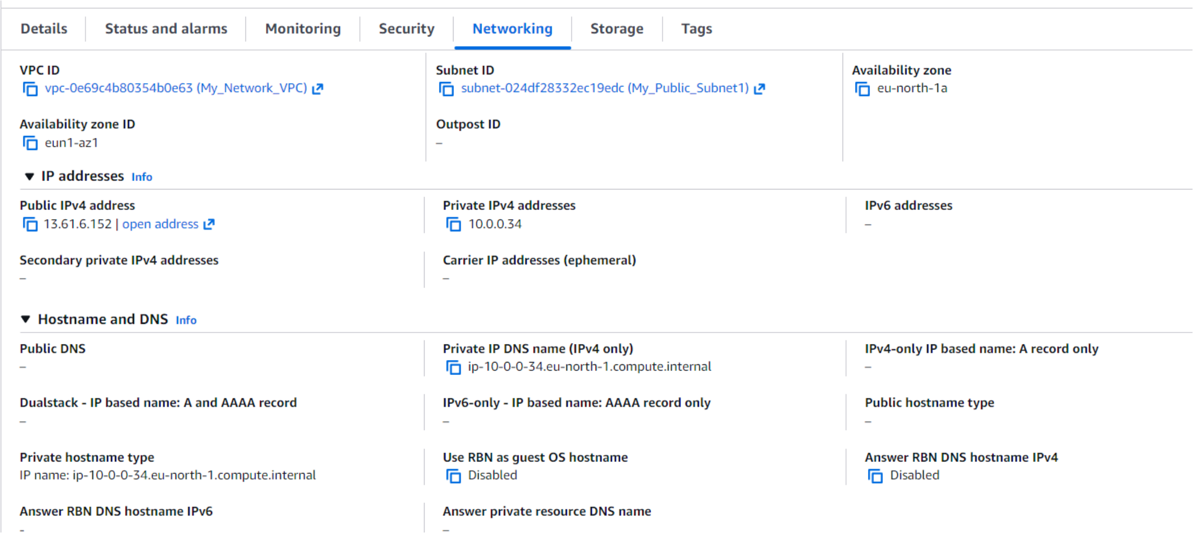
MY_Next_VPC-rtb-private1-eu-north-

Network connections (1)

Connections to other networks

MY_Next_VPC-igw
Internet routes to 1 public subnet
0 private subnets route to the Internet

- The preview window showed the full network layout the wizard would create — including the VPC, public subnet, private subnet, route tables, and internet gateway.
- After the VPC was created, I went through each component **one by one** and updated their names manually.
- Renaming everything (VPC, subnets, route tables, etc.) helped keep my environment organized and easy to recognize during the testing steps.



Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

VPC ID
vpc-0e69c4b80354b0e63 (My_Network_VPC)

Subnet ID
subnet-024df28332ec19edc (My_Public_Subnet1)

Availability zone
eu-north-1a

Availability zone ID
eun1-az1

Outpost ID
–

▼ IP addresses Info

Public IPv4 address
13.61.6.152 | open address

Private IPv4 addresses
10.0.0.34

IPv6 addresses
–

Secondary private IPv4 addresses
–

Carrier IP addresses (ephemeral)
–

▼ Hostname and DNS Info

Public DNS
–

Private IP DNS name (IPv4 only)
ip-10-0-0-34.eu-north-1.compute.internal

IPv4-only IP based name: A record only
–

Dualstack - IP based name: A and AAAA record
–

IPv6-only - IP based name: AAAA record only
–

Public hostname type
–

Private hostname type
IP name: ip-10-0-0-34.eu-north-1.compute.internal

Use RBN as guest OS hostname
Disabled

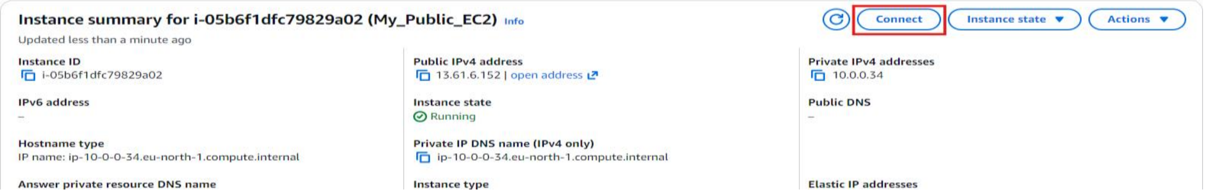
Answer RBN DNS hostname IPv4
Disabled

Answer RBN DNS hostname IPv6
–

Answer private resource DNS name
–

Public EC2 Networking Overview

- After creating the public EC2 instance, I reviewed the networking panel.
- It was correctly connected to **My_Public_Subnet1**, which is the public subnet created by the VPC wizard.
- The instance was assigned a **Public IP: 13.6.16.152**, allowing it to be accessed from the internet.
- It also received a **Private IP: 10.0.0.34**, used for internal communication inside the VPC.
- This confirmed that the public EC2 was set up correctly for both internet and VPC connectivity.



Instance summary for i-05b6f1dfc79829a02 (My_Public_EC2) Info

Updated less than a minute ago

Instance ID
i-05b6f1dfc79829a02

Public IPv4 address
13.61.6.152 | open address

Private IPv4 addresses
10.0.0.34

IPv6 address
–

Instance state
Running

Public DNS
–

Hostname type
IP name: ip-10-0-0-34.eu-north-1.compute.internal

Private IP DNS name (IPv4 only)
ip-10-0-0-34.eu-north-1.compute.internal

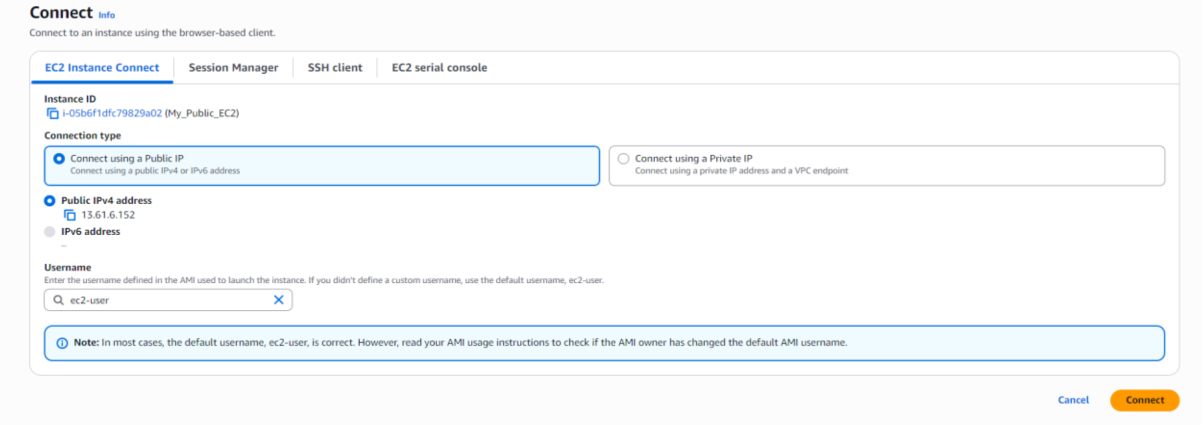
Answer private resource DNS name

Instance type

Elastic IP addresses

Connecting to the Public EC2 Instance

- From the instance summary page, I clicked the **Connect** button in the top-right corner.



Connect Info

Connect to an instance using the browser-based client.

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-05b6f1dfc79829a02 (My_Public_EC2)

Connection type

● 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

● Public IPv4 address
13.61.6.152

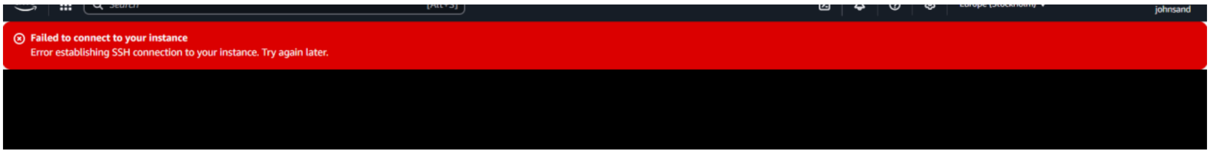
○ IPv6 address
–

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, ec2-user.

ec2-user

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

- A new window opened showing different connection options.
- I selected **Connect using Public IP**, since this instance is in a public subnet and has a public address.
- After selecting the option, I clicked **Connect**, and the browser-based terminal opened successfully.

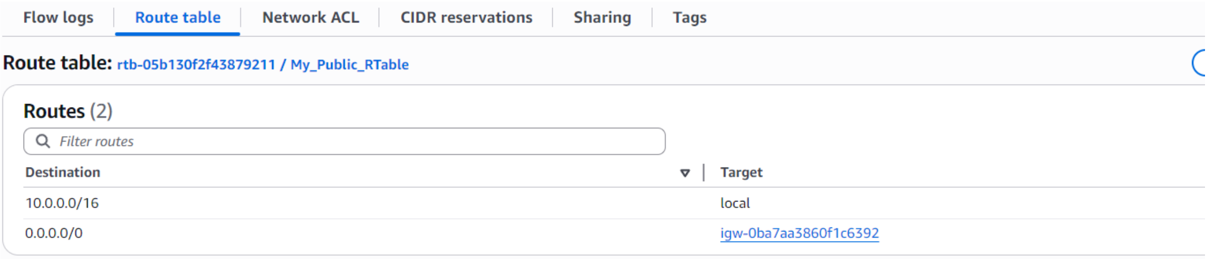


Failed to connect to your instance

Error establishing SSH connection to your instance. Try again later.

Initial Connection Issue

- At first, the instance did **not connect** through the browser terminal.
- This meant something in the networking or security configuration was blocking access.
- I had to **troubleshoot the issue** to find out what was preventing the connection.



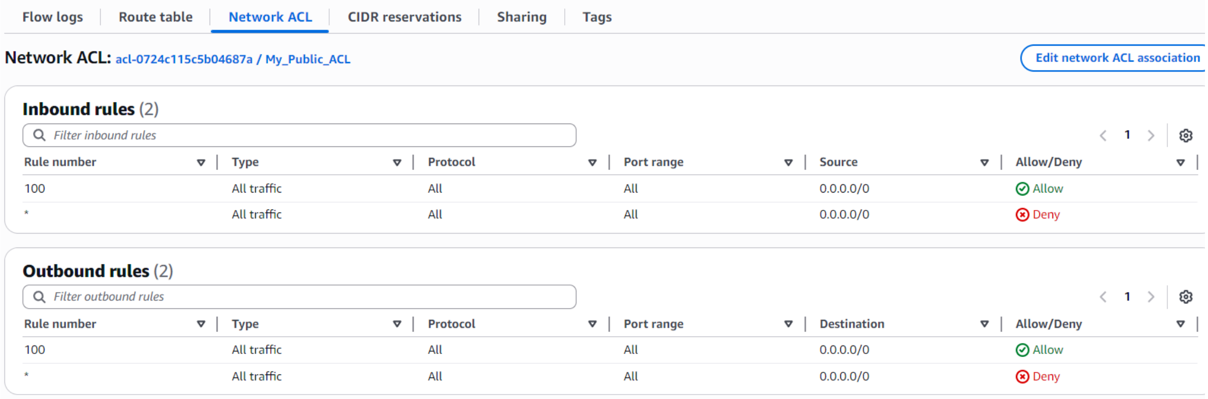
Route table: rtb-05b130f2f43879211 / My_Public_RTable

Routes (2)

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-0ba7aa3860f1c6392

Troubleshooting the Connection

- The first thing I checked was the **route table** associated with my public subnet.
- I confirmed that the route to the **Internet Gateway** was correctly configured.
- Everything in the route table looked good, so the issue was not related to routing.



Network ACL: acl-0724c115c5b04687a / My_Public_ACL

Inbound rules (2)

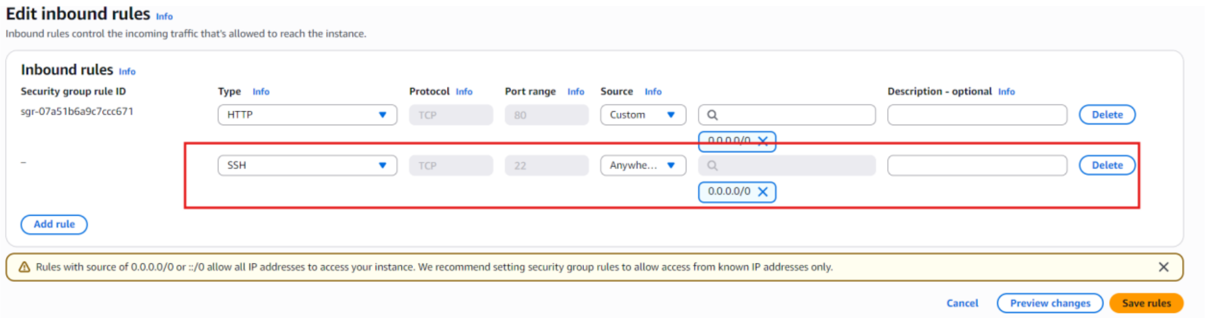
Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	✓ Allow
*	All traffic	All	All	0.0.0.0/0	⊗ Deny

Outbound rules (2)

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	✓ Allow
*	All traffic	All	All	0.0.0.0/0	⊗ Deny

Checking the Network ACL

- Next, I checked the **Network ACL** attached to my public subnet.
- Both **inbound and outbound rules** allowed all traffic, which confirmed that the ACL was not blocking the connection.
- Since the ACL was fully open, this wasn’t the cause of the connection issue.



Edit inbound rules Info

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

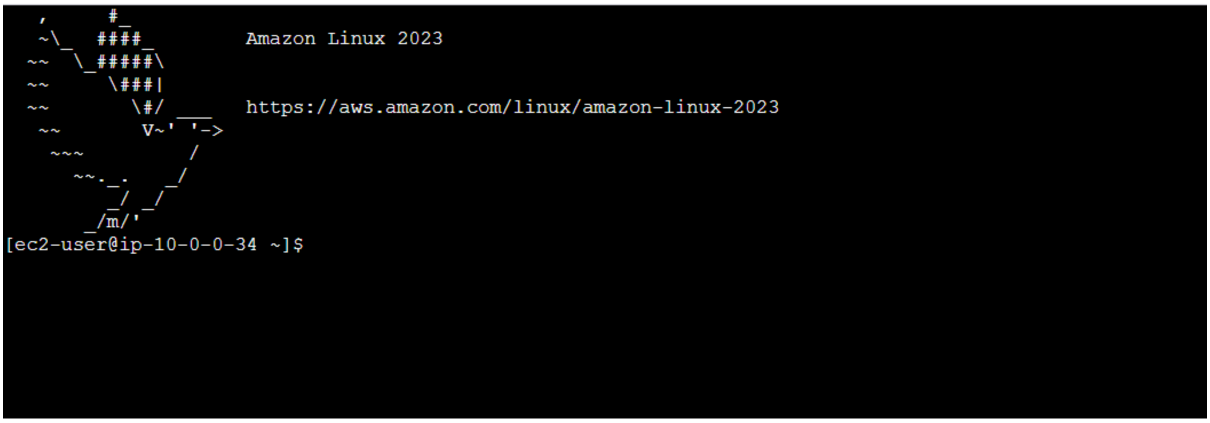
Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-07a51b6a9c7ccc671	HTTP	TCP	80	Custom	0.0.0.0/0	
–	SSH	TCP	22	Anywhe...	0.0.0.0/0	

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.

Fixing the Security Group

- Then I checked the **Security Group** attached to my public EC2 instance.
- I noticed that only **HTTP (port 80)** was allowed, which meant SSH connections were being blocked.
- To fix this, I added a new **inbound rule** allowing **SSH (port 22)**.
- I set the source to **0.0.0.0/0** since this is a public instance and needs to accept SSH connections from anywhere.
- After saving the rule, the instance became reachable.



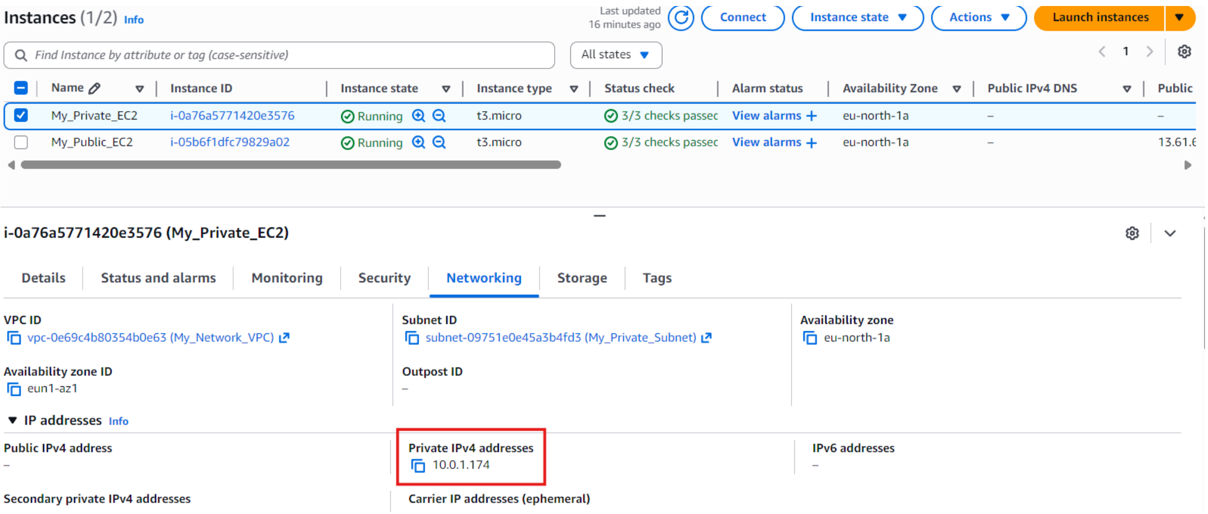
```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-0-34 ~]$
```

Connection Successful

- After updating the security group, I retried the connection.
- This time, the browser terminal connected successfully to my public EC2 instance.
- The issue was fully resolved, and the connection was established without any problems.
- This completed **Step 1** of the project.

Testing Connectivity between Public and Private EC2

- Next, I tested the connectivity between my **public EC2** and **private EC2**.
- Since both instances are in the same VPC, my **private EC2 should be able to talk to my public EC2** using its private IP.
- This test helps confirm that internal communication inside the VPC is working and that the security group rules allow the connection.



Instances (1/2) Info

Last updated 16 minutes ago

☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
☑	My_Private_EC2	i-0a76a5771420e3576	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	eu-north-1a	–	–
☐	My_Public_EC2	i-05b6f1dfc79829a02	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	eu-north-1a	–	13.61.6

i-0a76a5771420e3576 (My_Private_EC2)

VPC ID
vpc-0e69c4b80354b0e63 (My_Network_VPC)

Subnet ID
subnet-09751e0e45a3b4fd3 (My_Private_Subnet)

Availability zone
eu-north-1a

Availability zone ID
eun1-az1

Outpost ID
–

▼ IP addresses Info

Public IPv4 address
–

Private IPv4 addresses
10.0.1.174

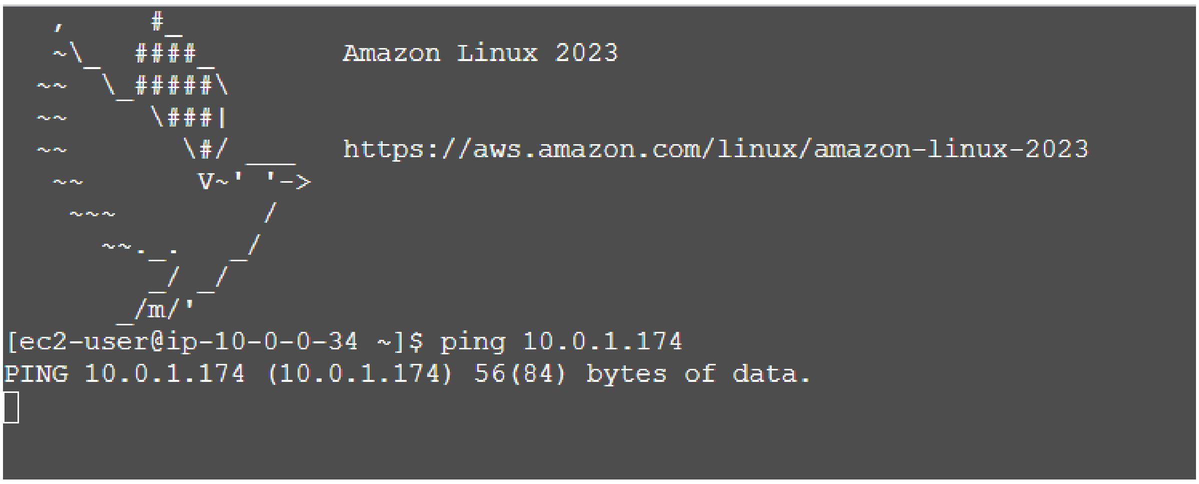
IPv6 addresses
–

Secondary private IPv4 addresses

Carrier IP addresses (ephemeral)

Preparing to Test Internal Connectivity

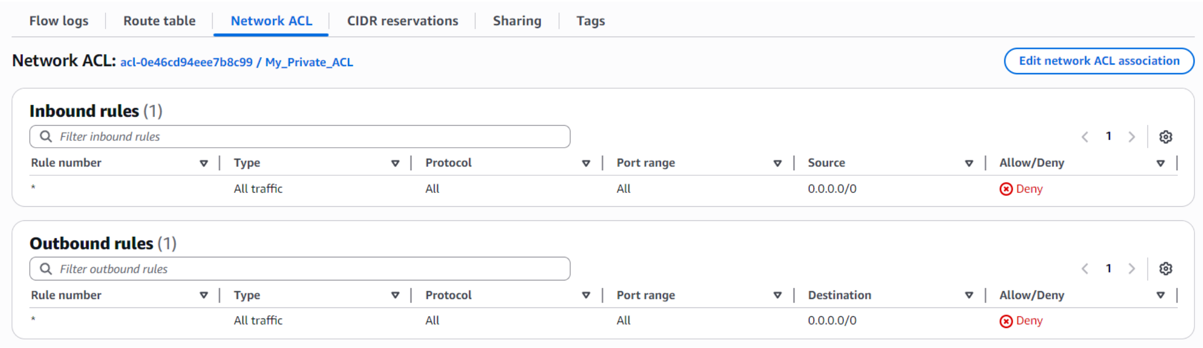
- First, I copied the **private IPv4 address** of my Private EC2 instance.
- This private IP is only reachable inside the VPC, so it's required for any internal communication tests.
- Then I went to my **Public EC2**, which was already running and connected through the browser terminal.
- From there, I planned to ping the private IP to check if the two instances could communicate internally.



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-0-34 ~]$ ping 10.0.1.174
PING 10.0.1.174 (10.0.1.174) 56(84) bytes of data.
```

Ping Attempt and Issue Found

- I tried to ping the private EC2 from my public EC2 using its private IP address.
- Only **one ping was sent**, and it stopped immediately afterward.
- This showed that something was blocking the communication between the two instances.
- I had to **troubleshoot the issue** to find out why the internal connectivity wasn't working.



Network ACL: acl-0e46cd94eee7b8c99 / My_Private_ACL

Inbound rules (1)

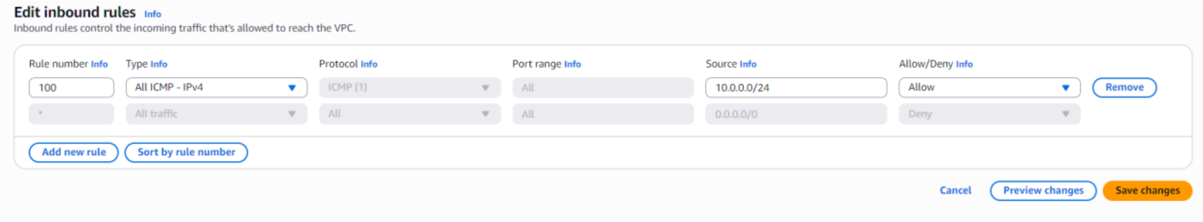
Rule number	Type	Protocol	Port range	Source	Allow/Deny
*	All traffic	All	All	0.0.0.0/0	⊗ Deny

Outbound rules (1)

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
*	All traffic	All	All	0.0.0.0/0	⊗ Deny

Checking the Private Subnet’s ACL

- I checked the **Network ACL** attached to my Private Subnet.
- I noticed that **no traffic at all was allowed** — both inbound and outbound rules were set to *deny*.
- Because of this, the private EC2 couldn’t receive or reply to any ping requests.
- This confirmed that the ACL was the reason internal communication wasn’t working.



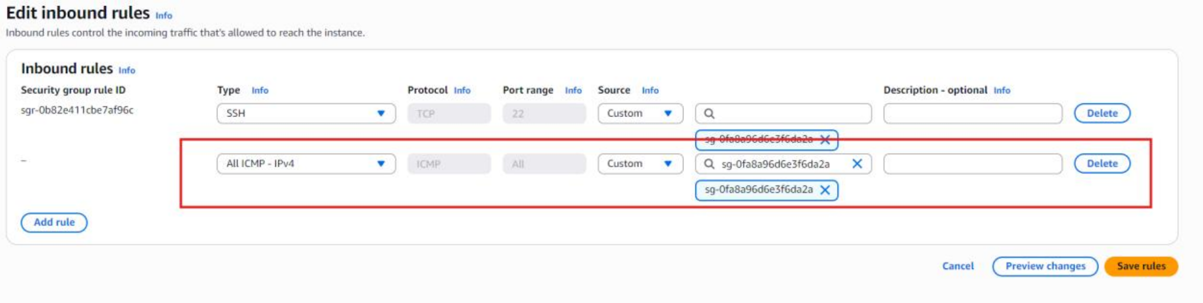
Edit inbound rules Info

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

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	All ICMP - IPv4	ICMP (1)	All	10.0.0.0/24	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Fixing the Private Subnet’s ACL

- To allow internal communication, I added new rules to the Private NACL.
- In the **Inbound Rules**, I allowed **ICMP** (used for ping) with **ICMP type: 100** and **source: 10.0.0.0/24**.
- In the **Outbound Rules**, I added the same ICMP rule with **destination: 10.0.0.0/24**.
- These rules allowed ping traffic between the public and private EC2 instances inside the VPC.



Edit inbound rules Info

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

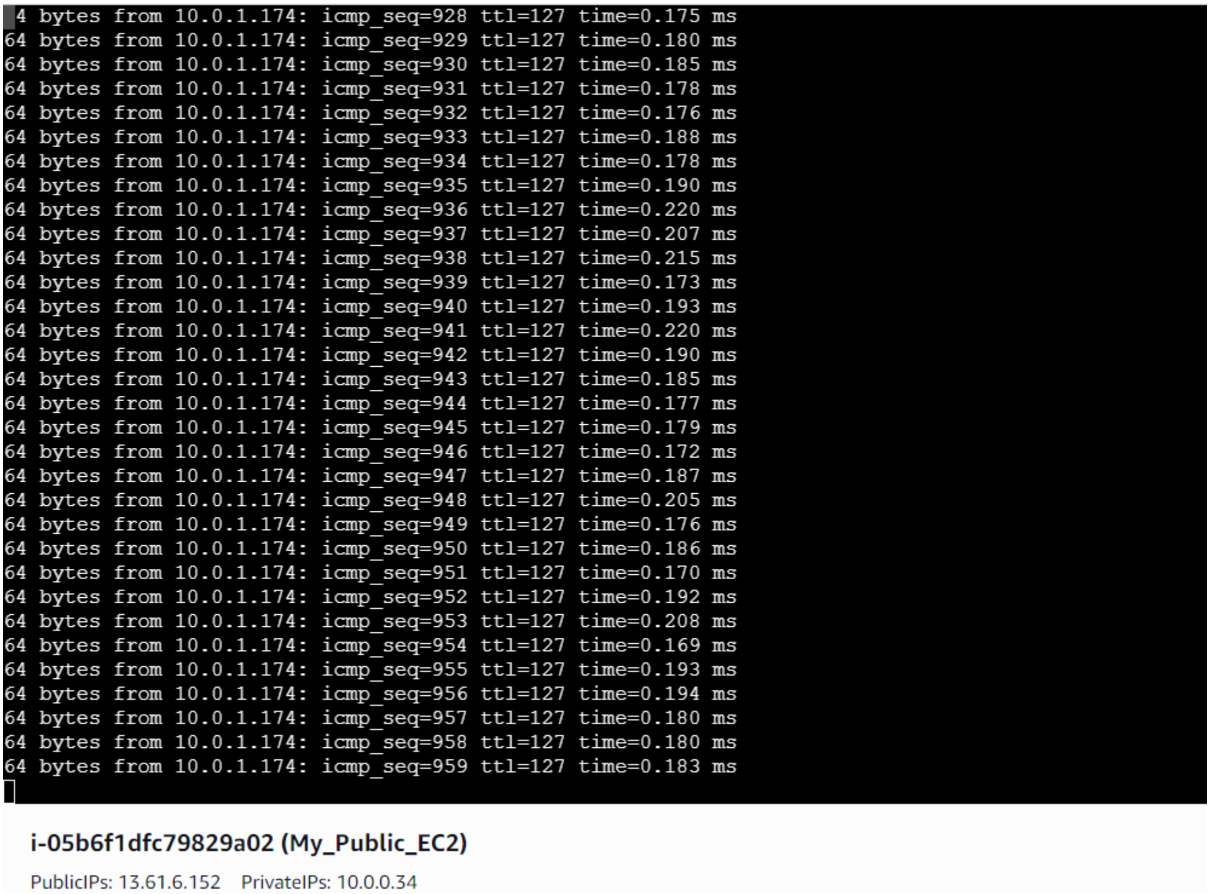
Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0b82e411cbe7af96c	SSH	TCP	22	Custom	sg-0fa8a96d6e3f6da2a	
–	All ICMP - IPv4	ICMP	All	Custom	sg-0fa8a96d6e3f6da2a sg-0fa8a96d6e3f6da2a	

Updating the Private Security Group

- After fixing the NACL, I checked the **Private Security Group** attached to my private EC2.
- I noticed that it only allowed **SSH**, and there were **no ICMP rules** for ping traffic.
- To allow the private EC2 to respond to pings, I added **ICMP** in both **Inbound** and **Outbound** rules.

- This ensured that ping requests coming from the public EC2 could reach the private EC2, and the private EC2 could send replies back.



```
64 bytes from 10.0.1.174: icmp_seq=928 ttl=127 time=0.175 ms
64 bytes from 10.0.1.174: icmp_seq=929 ttl=127 time=0.180 ms
64 bytes from 10.0.1.174: icmp_seq=930 ttl=127 time=0.185 ms
64 bytes from 10.0.1.174: icmp_seq=931 ttl=127 time=0.178 ms
64 bytes from 10.0.1.174: icmp_seq=932 ttl=127 time=0.176 ms
64 bytes from 10.0.1.174: icmp_seq=933 ttl=127 time=0.188 ms
64 bytes from 10.0.1.174: icmp_seq=934 ttl=127 time=0.178 ms
64 bytes from 10.0.1.174: icmp_seq=935 ttl=127 time=0.190 ms
64 bytes from 10.0.1.174: icmp_seq=936 ttl=127 time=0.220 ms
64 bytes from 10.0.1.174: icmp_seq=937 ttl=127 time=0.207 ms
64 bytes from 10.0.1.174: icmp_seq=938 ttl=127 time=0.215 ms
64 bytes from 10.0.1.174: icmp_seq=939 ttl=127 time=0.173 ms
64 bytes from 10.0.1.174: icmp_seq=940 ttl=127 time=0.193 ms
64 bytes from 10.0.1.174: icmp_seq=941 ttl=127 time=0.220 ms
64 bytes from 10.0.1.174: icmp_seq=942 ttl=127 time=0.190 ms
64 bytes from 10.0.1.174: icmp_seq=943 ttl=127 time=0.185 ms
64 bytes from 10.0.1.174: icmp_seq=944 ttl=127 time=0.177 ms
64 bytes from 10.0.1.174: icmp_seq=945 ttl=127 time=0.179 ms
64 bytes from 10.0.1.174: icmp_seq=946 ttl=127 time=0.172 ms
64 bytes from 10.0.1.174: icmp_seq=947 ttl=127 time=0.187 ms
64 bytes from 10.0.1.174: icmp_seq=948 ttl=127 time=0.205 ms
64 bytes from 10.0.1.174: icmp_seq=949 ttl=127 time=0.176 ms
64 bytes from 10.0.1.174: icmp_seq=950 ttl=127 time=0.186 ms
64 bytes from 10.0.1.174: icmp_seq=951 ttl=127 time=0.170 ms
64 bytes from 10.0.1.174: icmp_seq=952 ttl=127 time=0.192 ms
64 bytes from 10.0.1.174: icmp_seq=953 ttl=127 time=0.208 ms
64 bytes from 10.0.1.174: icmp_seq=954 ttl=127 time=0.169 ms
64 bytes from 10.0.1.174: icmp_seq=955 ttl=127 time=0.193 ms
64 bytes from 10.0.1.174: icmp_seq=956 ttl=127 time=0.194 ms
64 bytes from 10.0.1.174: icmp_seq=957 ttl=127 time=0.180 ms
64 bytes from 10.0.1.174: icmp_seq=958 ttl=127 time=0.180 ms
64 bytes from 10.0.1.174: icmp_seq=959 ttl=127 time=0.183 ms
```

i-05b6f1dfc79829a02 (My_Public_EC2)

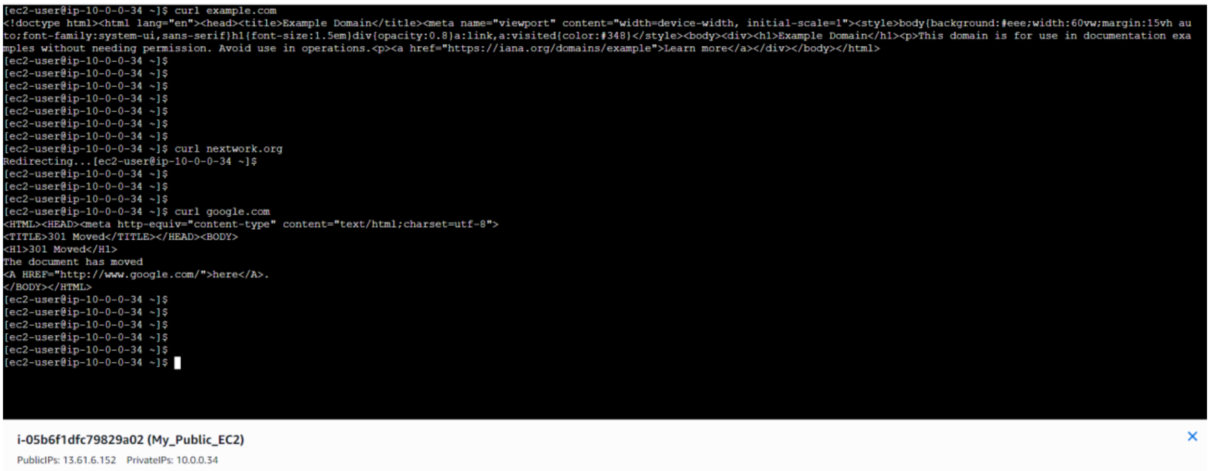
PublicIPs: 13.61.6.152   PrivateIPs: 10.0.0.34

Successful Internal Connectivity

- After updating the NACL and the security group, I went back to the public EC2 and tested the ping again.
- This time, the pings were **responding continuously**, without stopping.
- This confirmed that my **private EC2 instance was successfully connected** to my public EC2 through internal communication inside the VPC.

Testing VPC Connectivity With the Internet

- Next, I tested whether my VPC had proper internet connectivity using my **public EC2 instance**.
- Since the public EC2 is in a public subnet and has a public IP, it should be able to reach the internet through the Internet Gateway.
- Using the browser terminal on the public EC2, I ran basic commands to test external connectivity.



```
[ec2-user@ip-10-0-0-34 ~]$ curl example.com
<!doctype html><html lang="en"><head><title>Example Domain</title><meta name="viewport" content="width=device-width, initial-scale=1"><style>body{background:#eee;width:60vw;margin:15vh au
to;font-family:system-ui,sans-serif}h1{font-size:1.5em}div{opacity:0.8}a:link,a:visited{color:#348}</style><body><div><h1>Example Domain</h1><p>This domain is for use in documentation exa
mples without needing permission. Avoid use in operations.<p><a href="https://iana.org/domains/example">Learn more</a></div></body></html>
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$ curl nextwork.org
Redirecting...[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$ curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$
[ec2-user@ip-10-0-0-34 ~]$ █
```

i-05b6f1dfc79829a02 (My_Public_EC2)

PublicIPs: 13.61.6.152 PrivateIPs: 10.0.0.34

Testing Internet Access Using cURL

- To test internet connectivity, I used the **curl** command on my public EC2 instance.
- I used curl because it sends a request to a website and returns a response, which is a clear way to confirm if the instance can reach the internet.
- First, I ran:
  curl www.example.com
  This is a standard test website used for connectivity checks.
- Then I tested:
  curl nextwork.org
- Finally, I ran:
  curl google.com
- All three commands returned successful responses, which confirmed that my **public EC2 instance—and therefore my VPC—was connected to the external internet** through the Internet Gateway.

Conclusion

Through this project, I was able to fully test and understand how connectivity works inside a VPC. I successfully connected to my public EC2 instance, fixed security and routing issues, and validated internal communication between my public and private EC2 instances. I also confirmed that my VPC had proper internet access by testing external connectivity through the public EC2.

This project helped me understand how routing tables, NACLs, and security groups work together, and how each layer affects network communication. Overall, it strengthened my knowledge of AWS networking, troubleshooting, and secure cloud design.