VPC Connectivity

By Haroon Zaman | December 2025

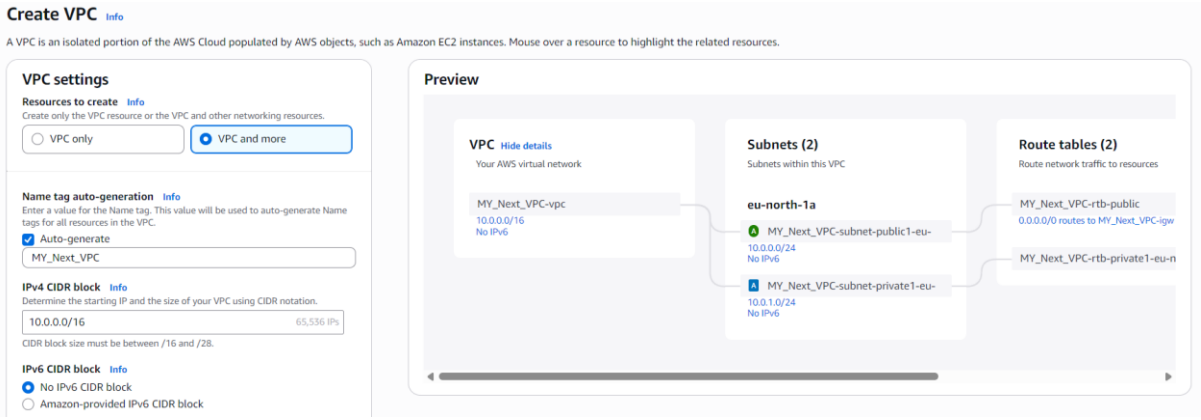
Introduction

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:

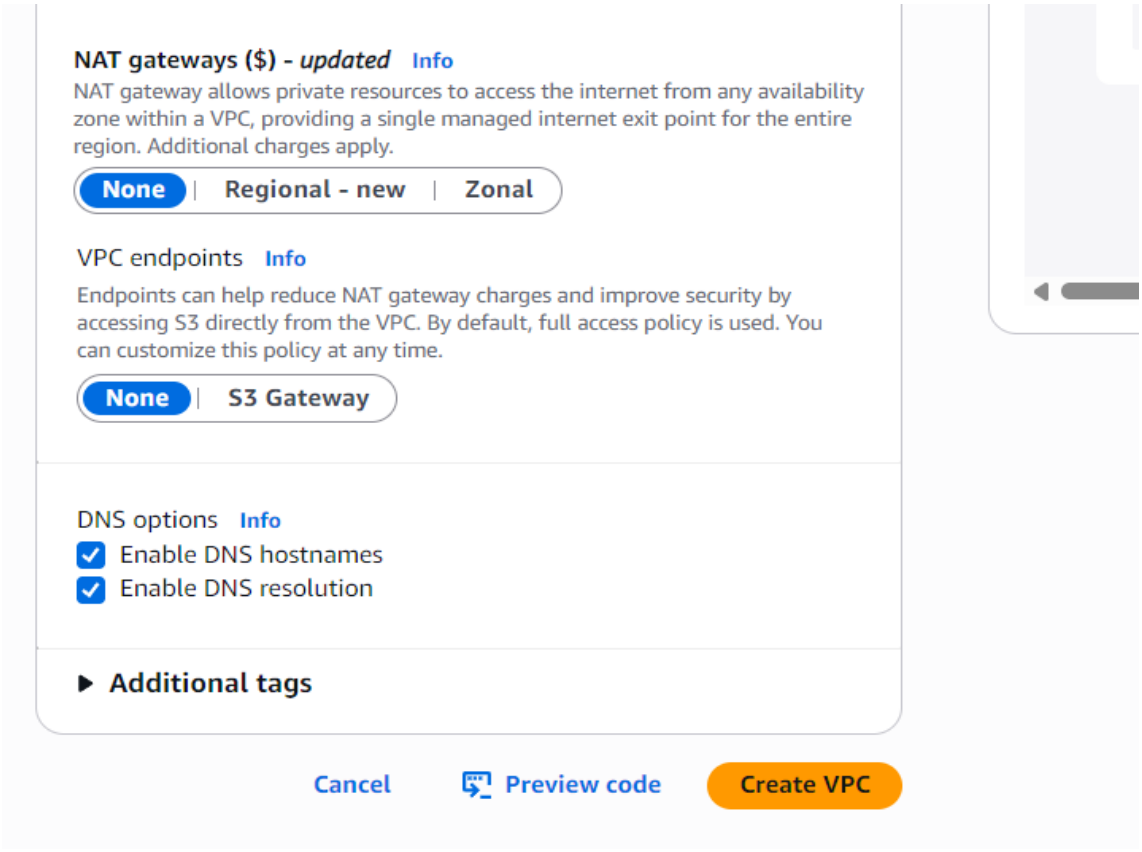
- **Connected to my Public EC2 Instance from the AWS Management Console** – to confirm that the instance has proper public access and my security group is configured correctly.
- **Tested Connectivity Between EC2 Instances** – to see how the public EC2 communicates with the private EC2 using internal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

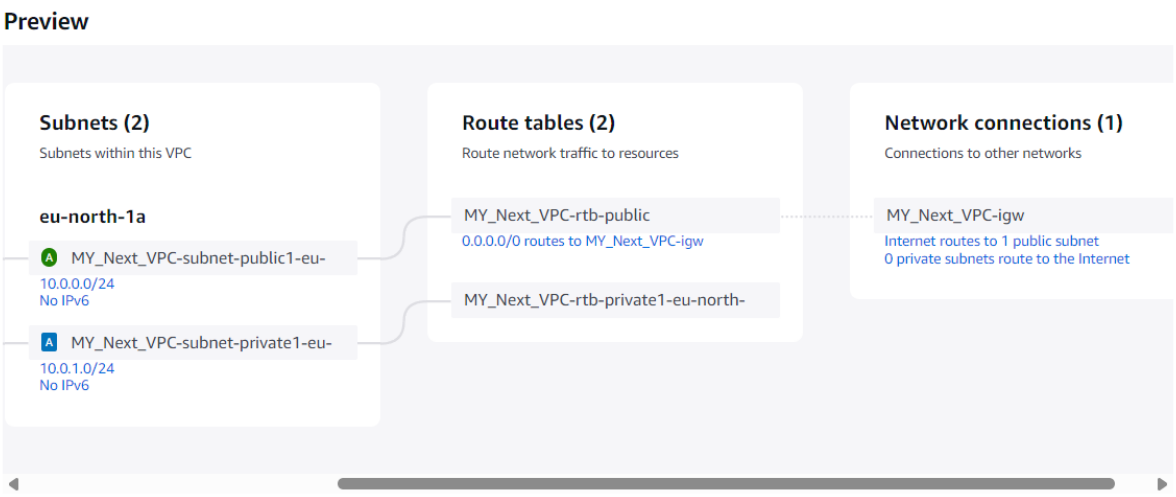


Creating a VPC Using "VPC and More"

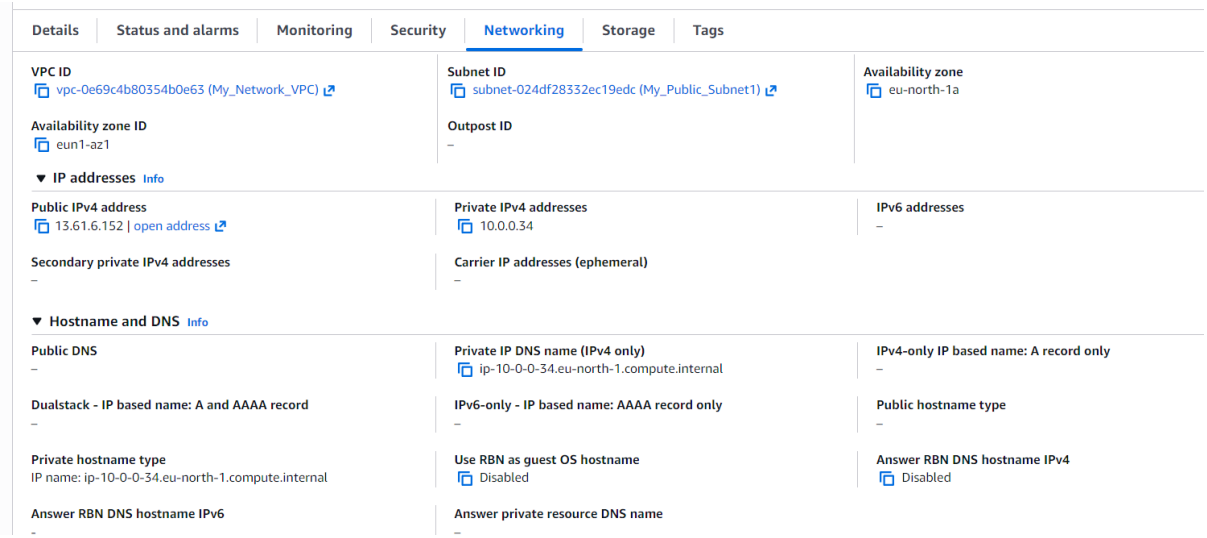
- I created a new VPC using the **VPC and More** option, which automatically generated **one public subnet** and **one private subnet**.
- The **public subnet** was routed to the internet through the Internet Gateway created by the wizard.
- I selected **No NAT Gateway** because my **private subnet should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

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





- 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.

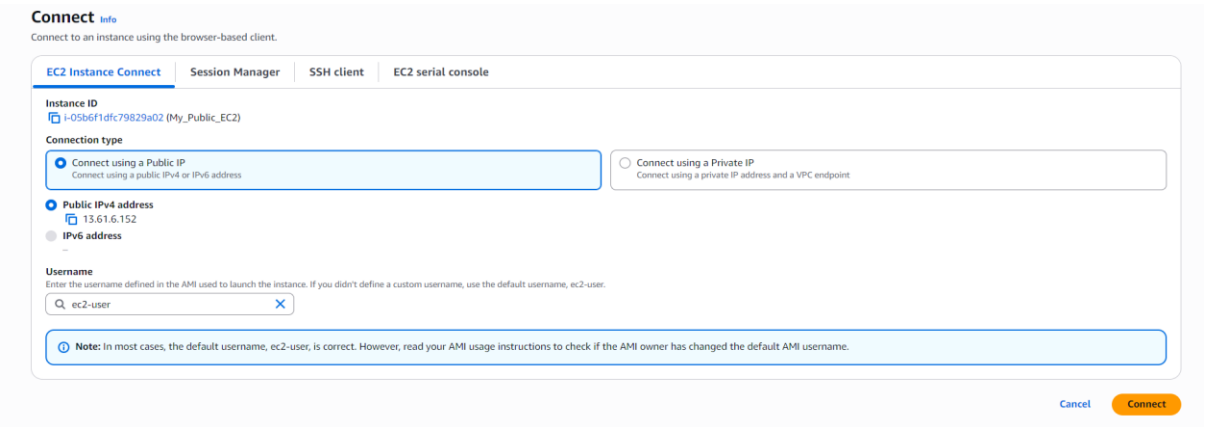


Public EC2 Networking Overview

- 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.

Connecting to the Public EC2 Instance

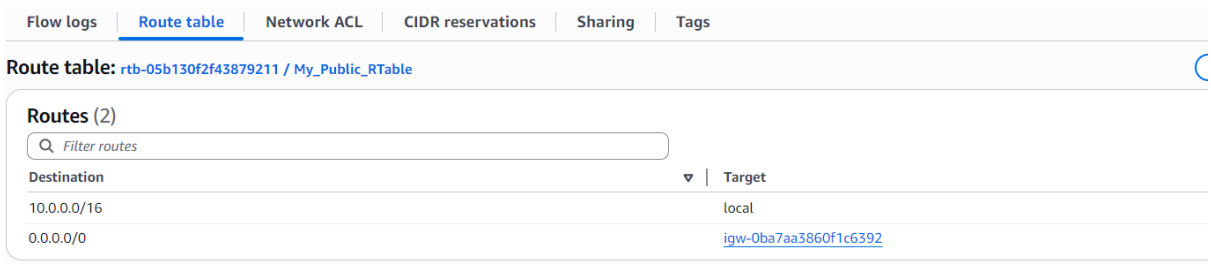
- From the instance summary page, I clicked the **Connect** button in the top-right corner.



- 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.

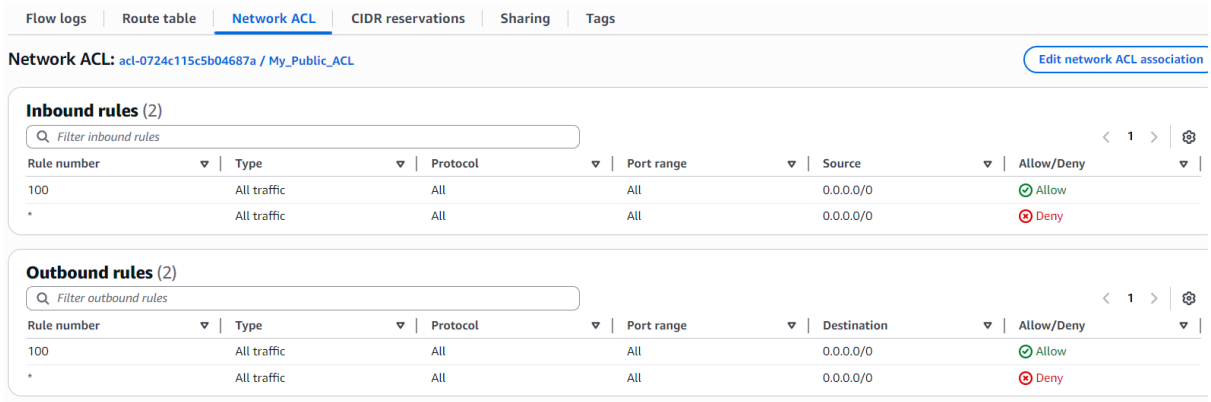
Initial Connection Issue

- 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.



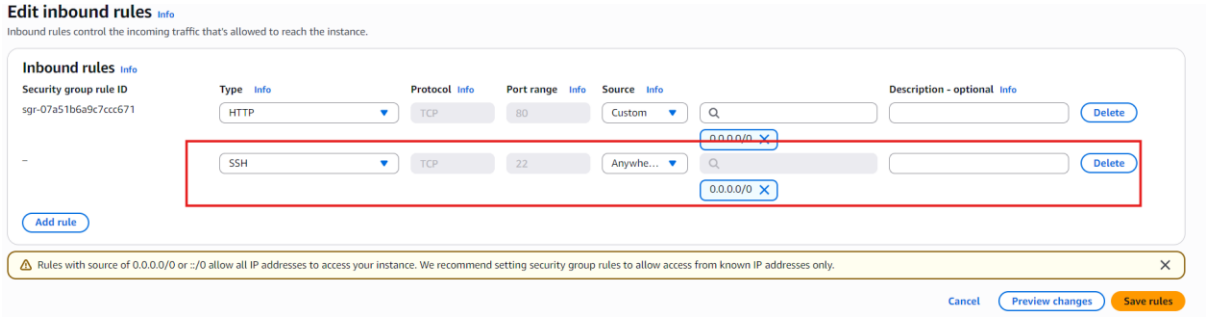
Troubleshooting the Connection

- 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.



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.



Fixing the Security Group

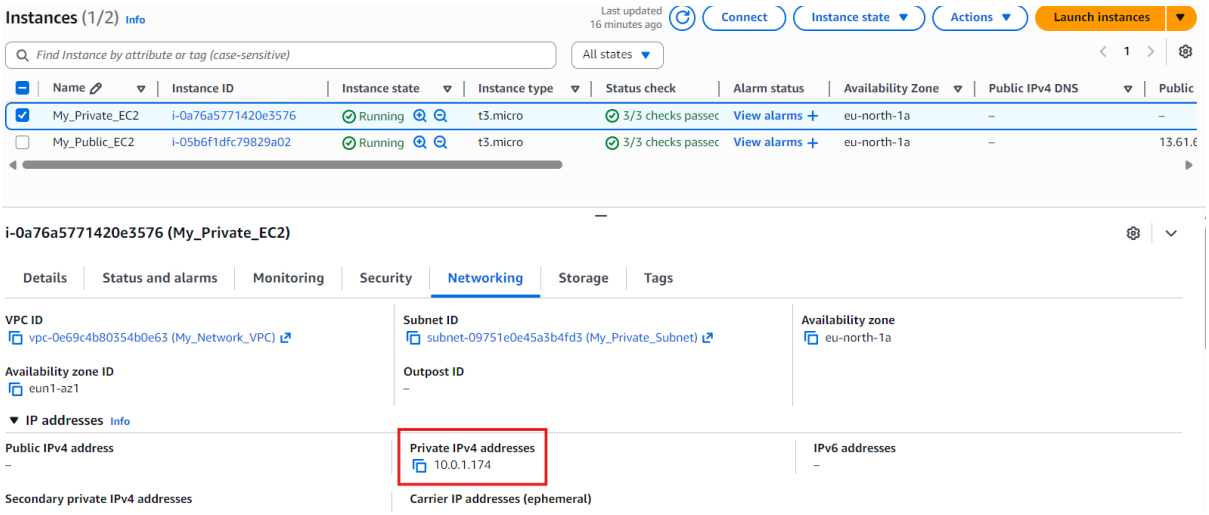
- Then I checked the **Security Group** attached to my public EC2 instance.
- I noticed that only **HTTP (port 80)** was allowed, which meant SSH connections were being blocked.
- To fix this, I added a new **inbound rule** allowing **SSH (port 22)**.
- I set the source to **0.0.0.0/0** since this is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

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.

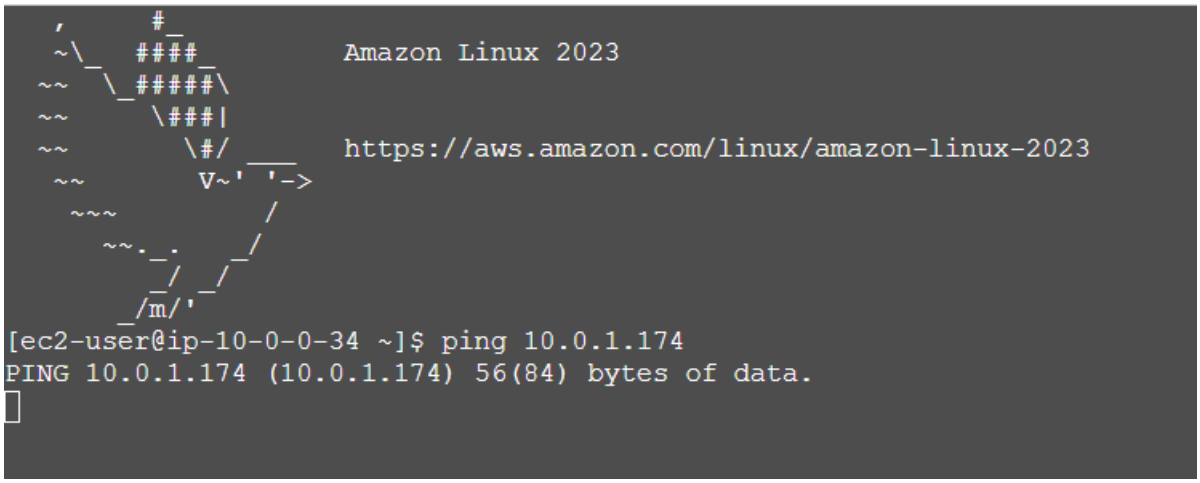
Testing Connectivity between Public and Private EC2

- 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.



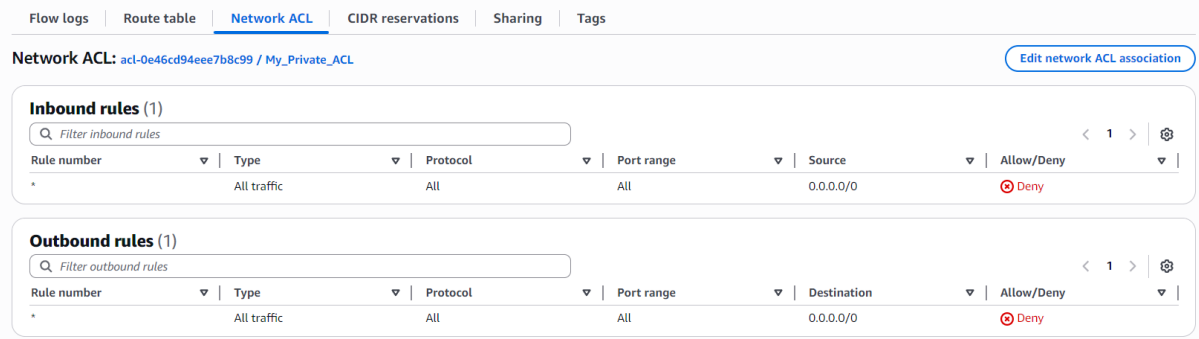
Preparing to Test Internal Connectivity

- 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.



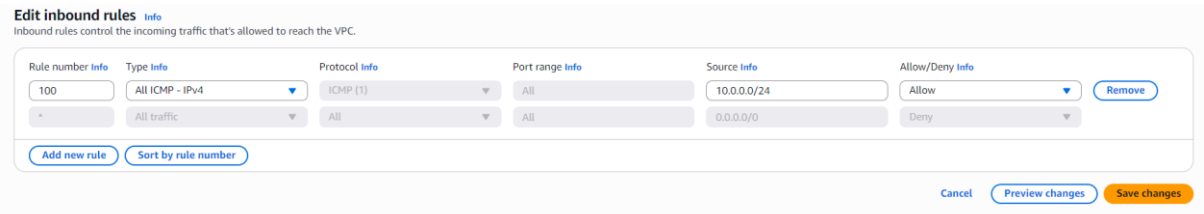
Ping Attempt and Issue Found

- 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.



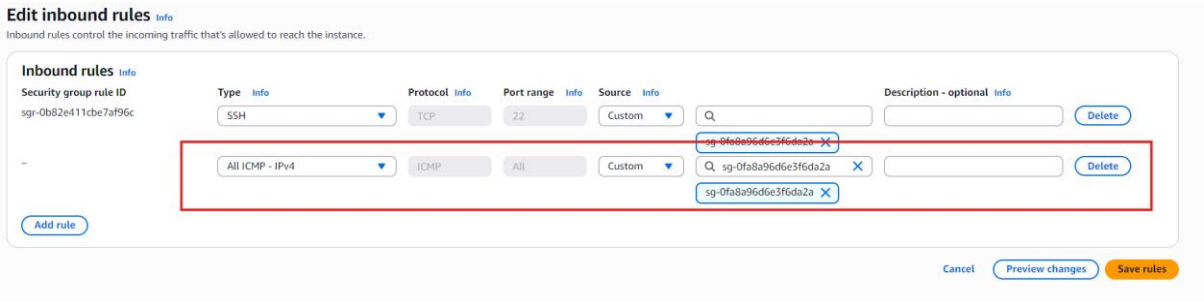
Checking the Private Subnet's ACL

- 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.



Fixing the Private Subnet's ACL

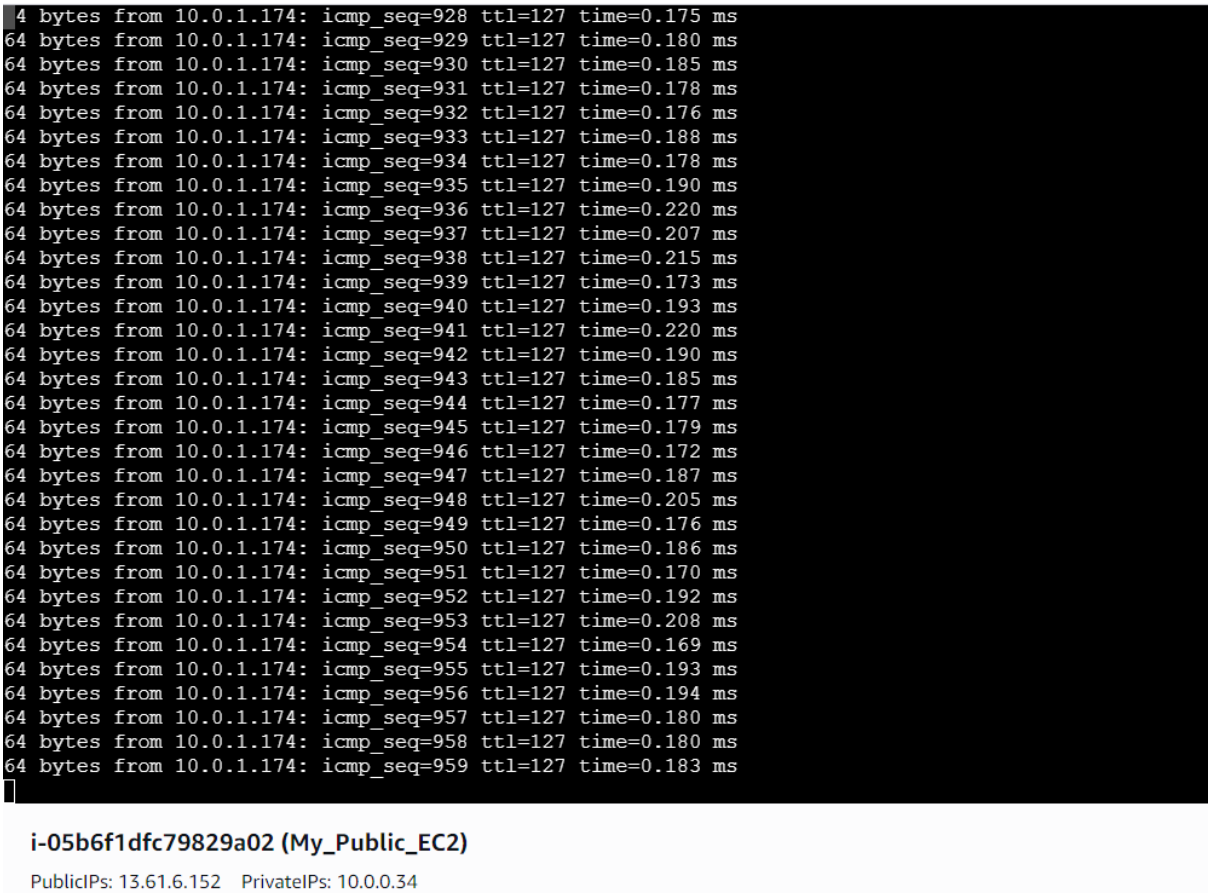
- To allow internal communication, I added new rules to the Private NACL.
- In the **Inbound Rules**, I allowed **ICMP** (used for ping) with **ICMP type: 100** and **source: 10.0.0.0/24**.
- In the **Outbound Rules**, I added the same ICMP rule with **destination: 10.0.0.0/24**.
- These rules allowed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



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.



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.

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.