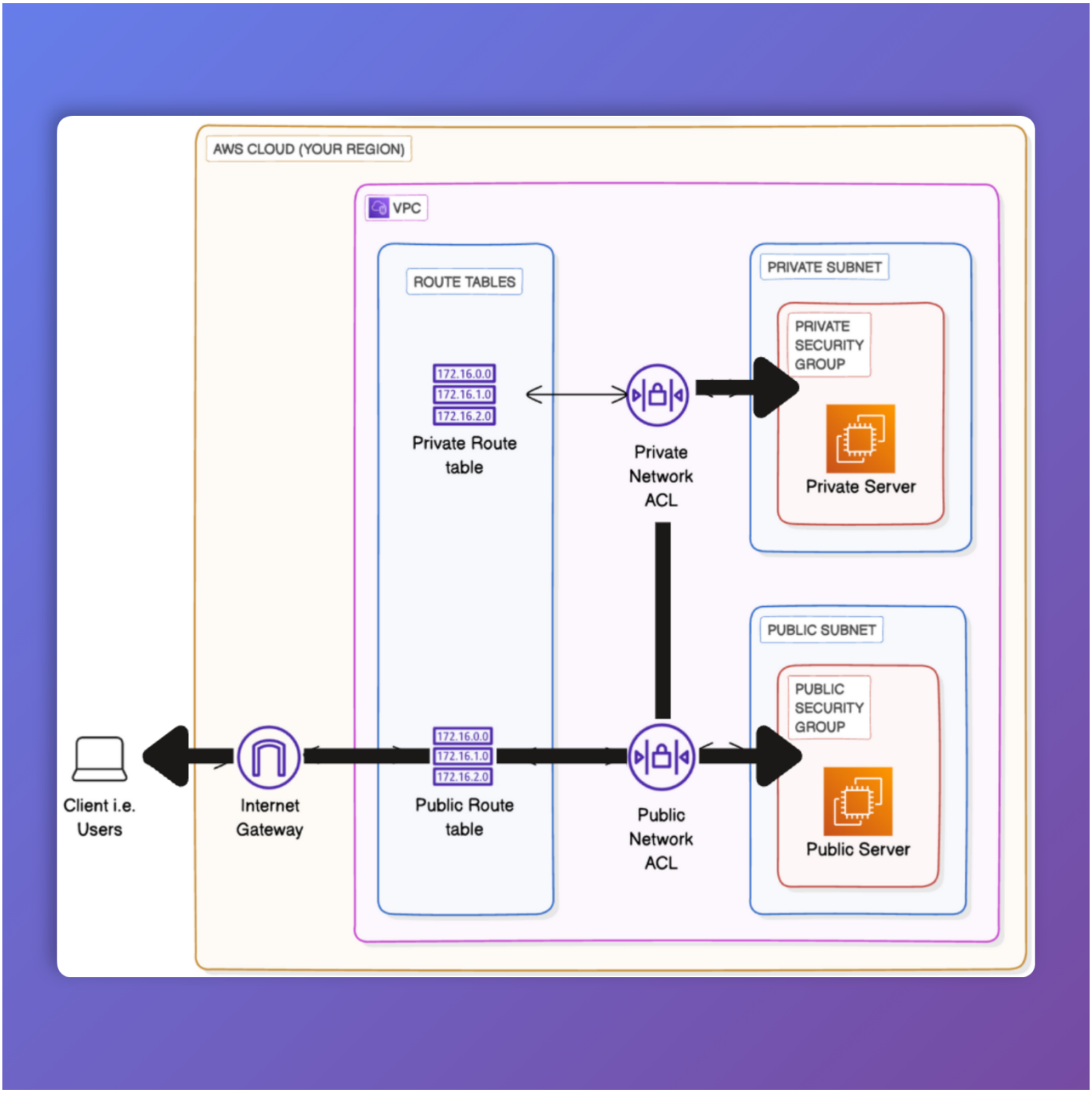
Testing VPC Connectivity

Project Overview

This project represents the part 5 in my AWS VPC series - **testing the connectivity** of the infrastructure I built! After launching EC2 instances in Part 4, I validated that my VPC architecture works as designed by testing three critical connectivity paths: connecting to the public server via SSH, establishing communication between public and private servers using ping, and verifying internet access from the public subnet using curl. Through hands-on troubleshooting, I learned how Security Groups and Network ACLs work together to control traffic flow.



Project Duration: Approximately 90 minutes
Difficulty Level: Spicy

AWS Region Used: eu-west-3 (Paris)
Project Series: Part 5 of NextWork VPC Challenge

Table of Contents

Technologies & Concepts

AWS Services & Tools Used

- **EC2 Instance Connect** - Browser-based SSH access without key management
- **Security Groups** - Instance-level stateful firewall
- **Network ACLs** - Subnet-level stateless firewall
- **ICMP Protocol** - For ping connectivity testing
- **SSH Protocol** - Secure Shell for remote instance access

Key Concepts Learned

1. EC2 Instance Connect

What is it?
EC2 Instance Connect is AWS's browser-based SSH solution that eliminates the need to manage SSH key pairs manually.

Traditional SSH Process:

1. Generate key pair (public + private keys)
2. Associate public key with EC2 instance
3. Securely store private key on local machine
4. Configure SSH client with private key
5. Establish encrypted connection

EC2 Instance Connect Process:

1. Click "Connect" in AWS Console
2. AWS generates one-time-use SSH key pair
3. AWS automatically installs public key on instance
4. Connection established (key expires after 60 seconds)
5. AWS removes the temporary key

Requirements:

- Instance must have public IP address
- Security Group must allow SSH (port 22) from EC2 Instance Connect IP ranges
- For simplicity, I used 0.0.0.0/0 (though more restrictive CIDR blocks are recommended for production)

Connection Details:

- **Username**: `ec2-user` (default for Amazon Linux 2023)
- **Public IP**: 51.44.10.214 (assigned to NextWork Public Server)
- **Private IP**: 10.0.0.225 (internal VPC address)

---

2. Connectivity Testing Fundamentals

What is connectivity?
Connectivity measures how effectively network components communicate with each other and external networks. It's the backbone of distributed systems.

Real-World Example:
Netflix uses over 100,000 EC2 instances that must communicate seamlessly for streaming. Without proper connectivity testing, video buffering and service interruptions would occur.

Three Types of Connectivity Tested:

Type	Description	Tools Used	Success Criteria
External → VPC	Internet users accessing public resources	EC2 Instance Connect (SSH)	Successful SSH session established
VPC → VPC	Resources within VPC communicating	`ping` (ICMP)	Replies received with latency metrics
VPC → Internet	VPC resources accessing external services	`curl` (HTTP/HTTPS)	HTML content retrieved successfully

---

3. The `ping` Command

Purpose:
Network diagnostic tool that checks if a host is reachable and measures round-trip time.

How it works:

1. Your computer sends an ICMP Echo Request to target IP
2. Target responds with ICMP Echo Reply
3. Your computer calculates round-trip time

Real-world analogy:
Like ringing someone's doorbell to check if they're home. If they answer (reply), you know they're there and measure how long it took.

My Ping Command:

```
ping 10.0.1.208
```

Initial Response (FAILED):

```
PING 10.0.1.208 (10.0.1.208) 56(84) bytes of data.
```

Only one line appeared - no replies received.

After Fixing NACLs and Security Groups (SUCCESS):

```
PING 10.0.1.208 (10.0.1.208) 56(84) bytes of data.
64 bytes from 10.0.1.208: icmp_seq=1 ttl=64 time=0.856 ms
64 bytes from 10.0.1.208: icmp_seq=2 ttl=64 time=0.842 ms
64 bytes from 10.0.1.208: icmp_seq=3 ttl=64 time=0.831 ms
```

Understanding the Output:

- **64 bytes**: Size of the ICMP packet
- **icmp_seq=1**: Sequence number (1st, 2nd, 3rd packet)
- **ttl=64**: Time To Live (hops remaining before packet expires)
- **time=0.856 ms**: Round-trip latency (incredibly fast!)

---

4. The `curl` Command

Purpose:
Command-line tool for transferring data using various protocols (HTTP, HTTPS, FTP, etc.). It's like a browser, but text-based.

Difference Between `ping` and `curl`:

Feature	`ping`	`curl`
Protocol	ICMP	HTTP/HTTPS
Purpose	Check if host is alive	Retrieve web content
Response	Echo reply (yes/no)	Full HTML/data
Use Case	Network diagnostics	Web API testing, downloading files

My Commands:

Test 1 - Simple Website:

```
curl example.com
```

Response: Full HTML of example.com homepage (confirms internet access)

Test 2 - NextWork Website:

```
curl nextwork.org
```

Response:

```
<a href="https://learn.nextwork.org/projects/aws-host-a-website-on-s3">Found</a>
```

This indicates a redirect (HTTP 302) to the full URL.

Test 3 - Full NextWork Page:

```
curl https://learn.nextwork.org/projects/aws-host-a-website-on-s3
```

Response: Complete HTML document (hundreds of lines) containing:

- Metadata tags
- CSS styling
- Page structure
- Content for the S3 hosting project

Why this matters:
Successful `curl` commands prove that:

- Public Subnet route table has correct route (0.0.0.0/0 → Internet Gateway)
- Internet Gateway is properly attached to VPC
- Security Groups allow outbound HTTP/HTTPS traffic
- DNS resolution is working

---

5. Network ACLs vs Security Groups - The Complete Picture

I learned the hard way why BOTH layers are essential!

Feature	Network ACL	Security Group
Scope	Subnet-level	Instance-level
State	Stateless (must configure inbound + outbound)	Stateful (return traffic auto-allowed)
Rules	Numbered (evaluated in order)	Not numbered (all rules evaluated)

Feature	Network ACL	Security Group
Default Action	Deny all (unless explicitly allowed)	Deny all inbound, allow all outbound
Granularity	Broad (affects all instances in subnet)	Specific (per instance)

Real-World Scenario I Encountered:

Problem: Ping from Public Server (10.0.0.225) to Private Server (10.0.1.208) failed.

Troubleshooting Journey:

Step 1 - Check Route Tables:
☑ Public Subnet route table has 10.0.0.0/16 → local (correct!)
☑ Private Subnet route table has 10.0.0.0/16 → local (correct!)

Step 2 - Check Network ACLs:
❌ Private NACL DENIES all traffic (inbound rule: _ → Deny)
❌ Private NACL DENIES all traffic (outbound rule: _ → Deny)

Step 3 - Fix Network ACLs:

- Added Rule 100: All ICMP-IPv4, Source 10.0.0.0/24, Allow (inbound)
- Added Rule 100: All ICMP-IPv4, Destination 10.0.0.0/24, Allow (outbound)

Step 4 - Ping Again:
Still no response! 😳

Step 5 - Check Security Groups:
❌ Private Security Group only allows SSH from Public Security Group
❌ No ICMP rule exists!

Step 6 - Fix Security Group:

- Added rule: All ICMP-IPv4, Source sg-0dc112d906793eb42 (Public Security Group)

Step 7 - Ping Again:
☑ SUCCESS! Replies received with <1ms latency!

Key Insight:
Think of it like getting into a gated community:

1. **Network ACL** = Security checkpoint at the neighborhood entrance (everyone must pass)
2. **Security Group** = Security guard at individual house door (specific to that house)

Even if you pass the neighborhood checkpoint (NACL), you still need permission at the house door (Security Group)!

---

6. Defense in Depth - Layered Security

My VPC implements multiple security layers:

Layer 1 - Network Design:

- Private subnet has NO route to Internet Gateway (architectural isolation)

Layer 2 - Network ACLs:

- Private NACL only allows traffic from Public Subnet (10.0.0.0/24)
- All other traffic denied by default

Layer 3 - Security Groups:

- Private Security Group only allows SSH from Public Security Group
- Private Security Group only allows ICMP from Public Security Group
- Cannot be accessed directly from internet

Layer 4 - Instance-Level:

- No public IP assigned to private instance
- SSH key pair required for authentication

Security Architecture:

```
Internet
    ↓
[Internet Gateway] ← Layer 1
    ↓
[Public Network ACL] ← Layer 2
    ↓
[Public Security Group] ← Layer 3
    ↓
[Public Server]
    ↓ (only internal traffic)
[Private Network ACL] ← Layer 2
    ↓
[Private Security Group] ← Layer 3
    ↓
[Private Server] ← Layer 4 (no public IP)
```

Real-World Benefit:
If a malicious actor compromises the Public Server, they STILL cannot:

- Access Private Server directly from their own computer (no public IP)
- SSH into Private Server from anywhere except the Public Server
- Even from Public Server, they're restricted by Security Group rules

---

7. ICMP Protocol

What is ICMP?
Internet Control Message Protocol - a supporting protocol used for diagnostics and error reporting.

Common ICMP Message Types:

- **Echo Request (Type 8)**: "Are you there?" (sent by ping)
- **Echo Reply (Type 0)**: "Yes, I'm here!" (response to ping)
- **Destination Unreachable (Type 3)**: "Can't reach that host"
- **Time Exceeded (Type 11)**: "Packet expired (TTL reached 0)"

Why was ICMP blocked by default?
Security best practice! ICMP can be exploited for:

- **Ping floods**: Overwhelming a server with ping requests (DoS attack)
- **Network reconnaissance**: Attackers mapping your network topology
- **Smurf attacks**: Amplification attacks using ping

My Configuration:

- Allowed ICMP only from Public Subnet (10.0.0.0/24)
- More secure than allowing 0.0.0.0/0 (entire internet)

Step-by-Step Testing

Step 1: Connect to Public Server with EC2 Instance Connect

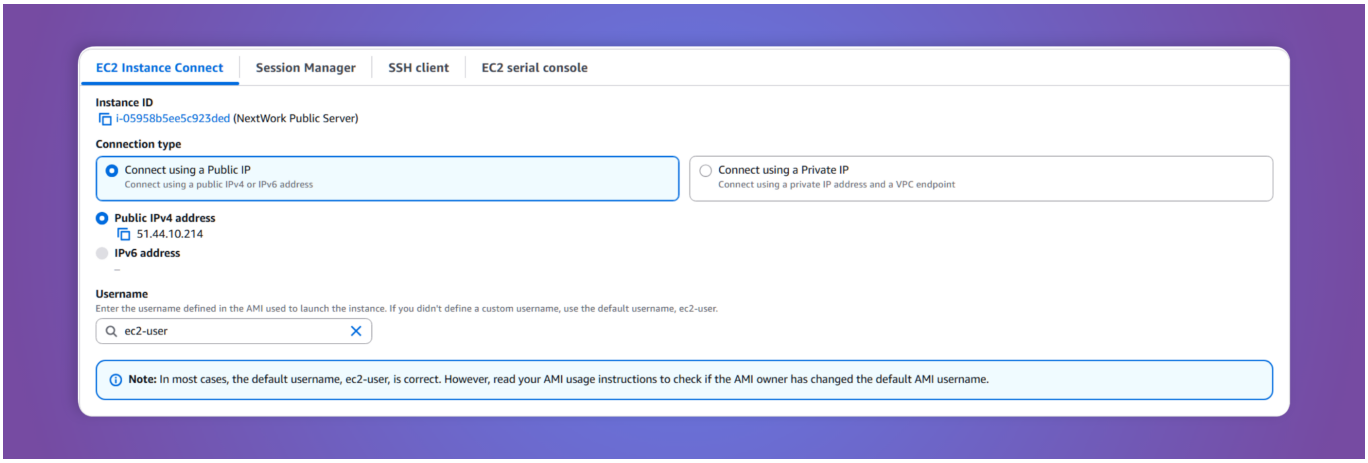
Initial Connection Attempt

What I did:
Attempted to connect to NextWork Public Server (i-05958b5ee5c923ded) using EC2 Instance Connect.

Navigation:

1. EC2 Console → Instances
2. Selected NextWork Public Server
3. Clicked "Connect"

Connection Settings:



Result:

Troubleshooting the Connection Failure

My Investigative Process:

Step 1 - Checked VPC Configuration:

- Verified Public Subnet exists ☑
- Verified Public Subnet has route to Internet Gateway ☑
- Checked Network ACL - allows all inbound/outbound traffic ☑

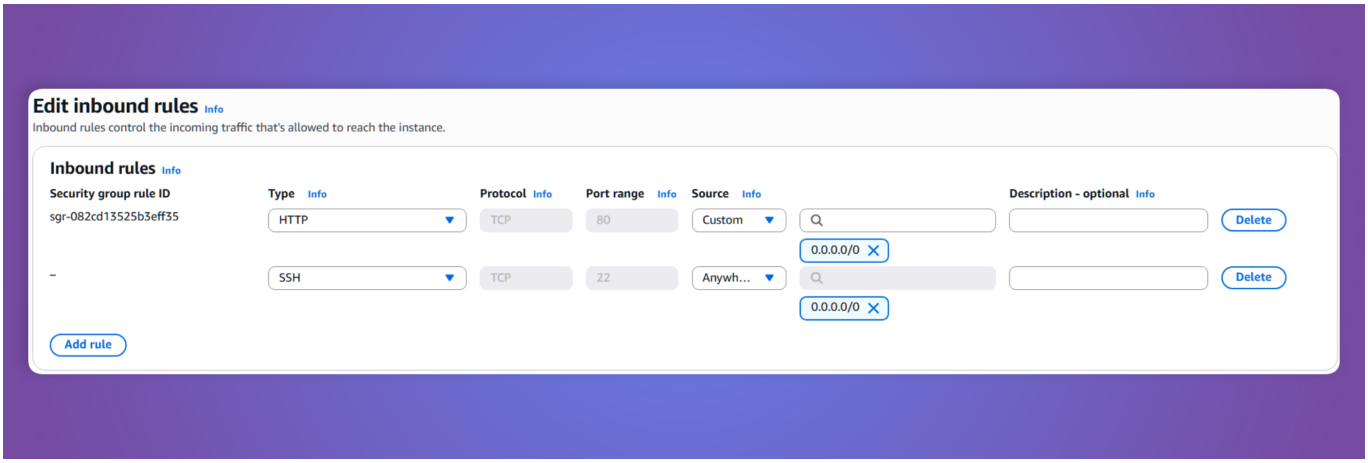
Step 2 - Checked Security Group:

- Navigated to: VPC Console → Security Groups
- Selected: NextWork Public Security Group (sg-082cd13525b3eff35)
- Examined Inbound Rules

Root Cause Identified:
EC2 Instance Connect uses SSH (port 22) to establish connections. My Security Group only allowed HTTP traffic, blocking the SSH connection attempt.

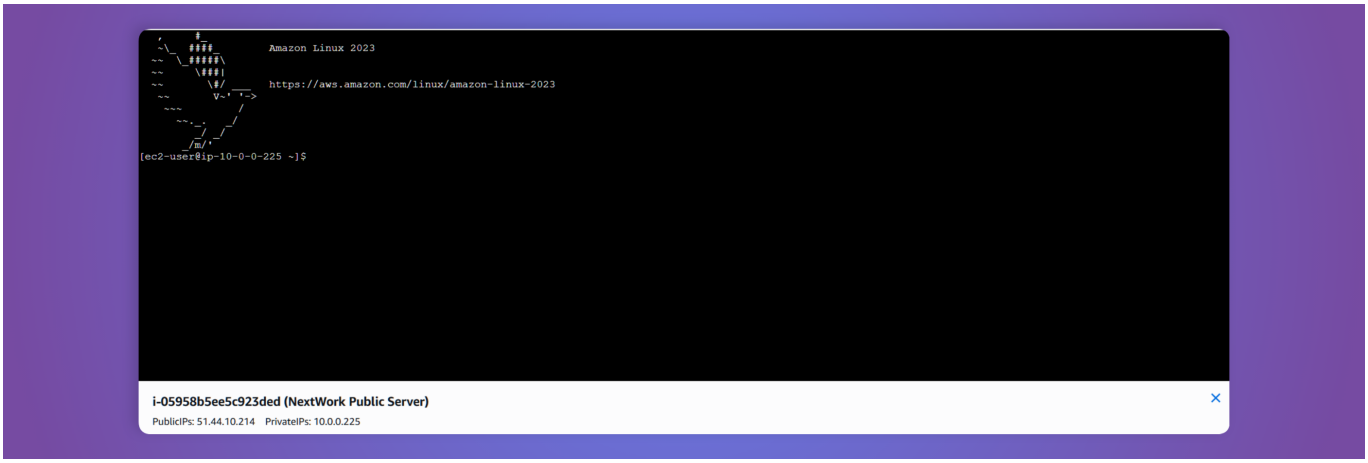
Fixing the Security Group

Edit Inbound Rules



Security Consideration:
In production, I would restrict SSH to specific EC2 Instance Connect IP ranges for the eu-west-3 region. However, for learning purposes and because EC2 Instance Connect uses various IP ranges, I used 0.0.0.0/0.

Successful Connection



Breaking down the terminal prompt:

- **ec2-user**: Current logged-in user (default Amazon Linux user)
- **@**: "at" (indicates separation between user and host)
- **ip-10-0-0-225**: Private IP of the instance (10.0.0.225)
- **~**: Current directory (home directory /home/ec2-user)
- **$**: Regular user prompt (# would indicate root user)

Achievement Unlocked! 🎉
I now have command-line access to my EC2 instance running in the public subnet!

Step 2: Test Connectivity Between EC2 Instances

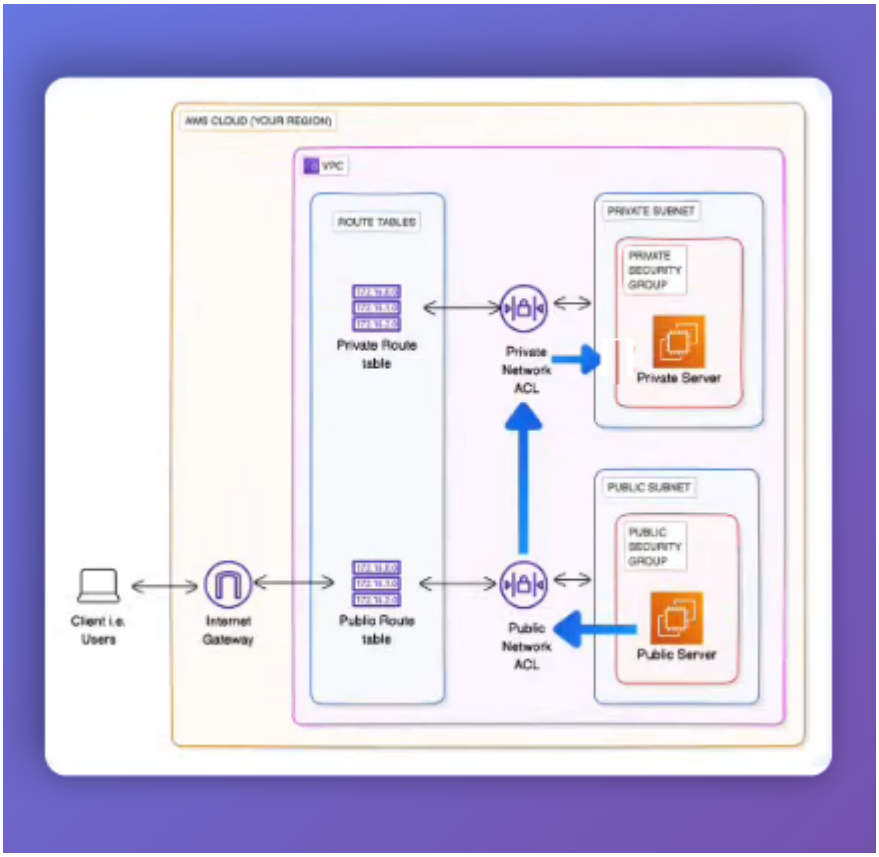
Understanding Inter-Instance Communication

The Goal:
Test whether NextWork Public Server (10.0.0.225) can communicate with NextWork Private Server (10.0.1.208) despite being in different subnets and availability zones.

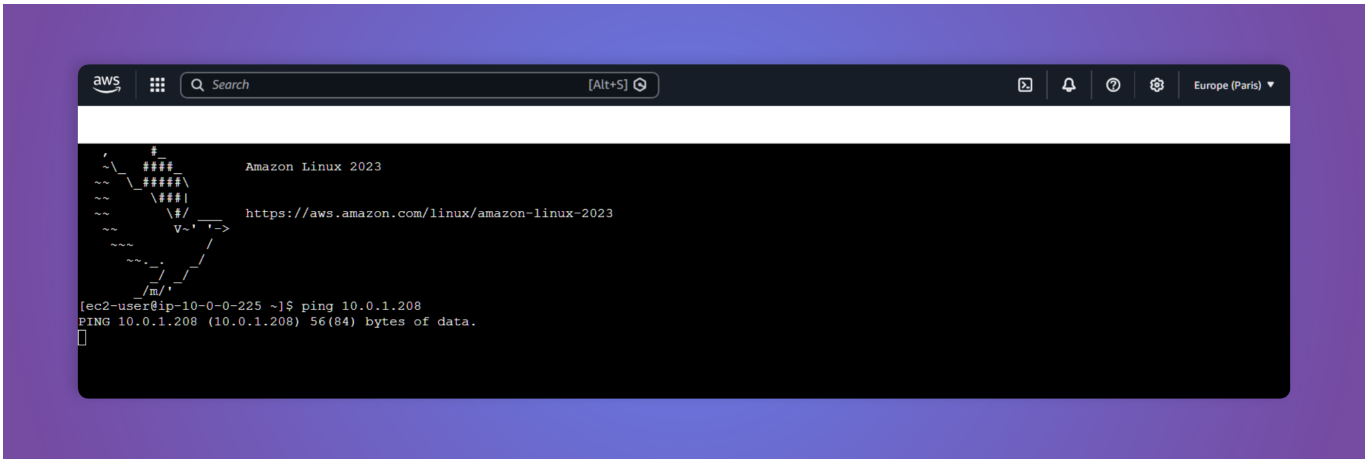
Why This Matters:
In real-world architectures:

- Public servers (web servers, load balancers) need to fetch data from private servers (databases, application servers)
- Example: A web server in the public subnet queries a database in the private subnet when a user visits a website

The Network Path:



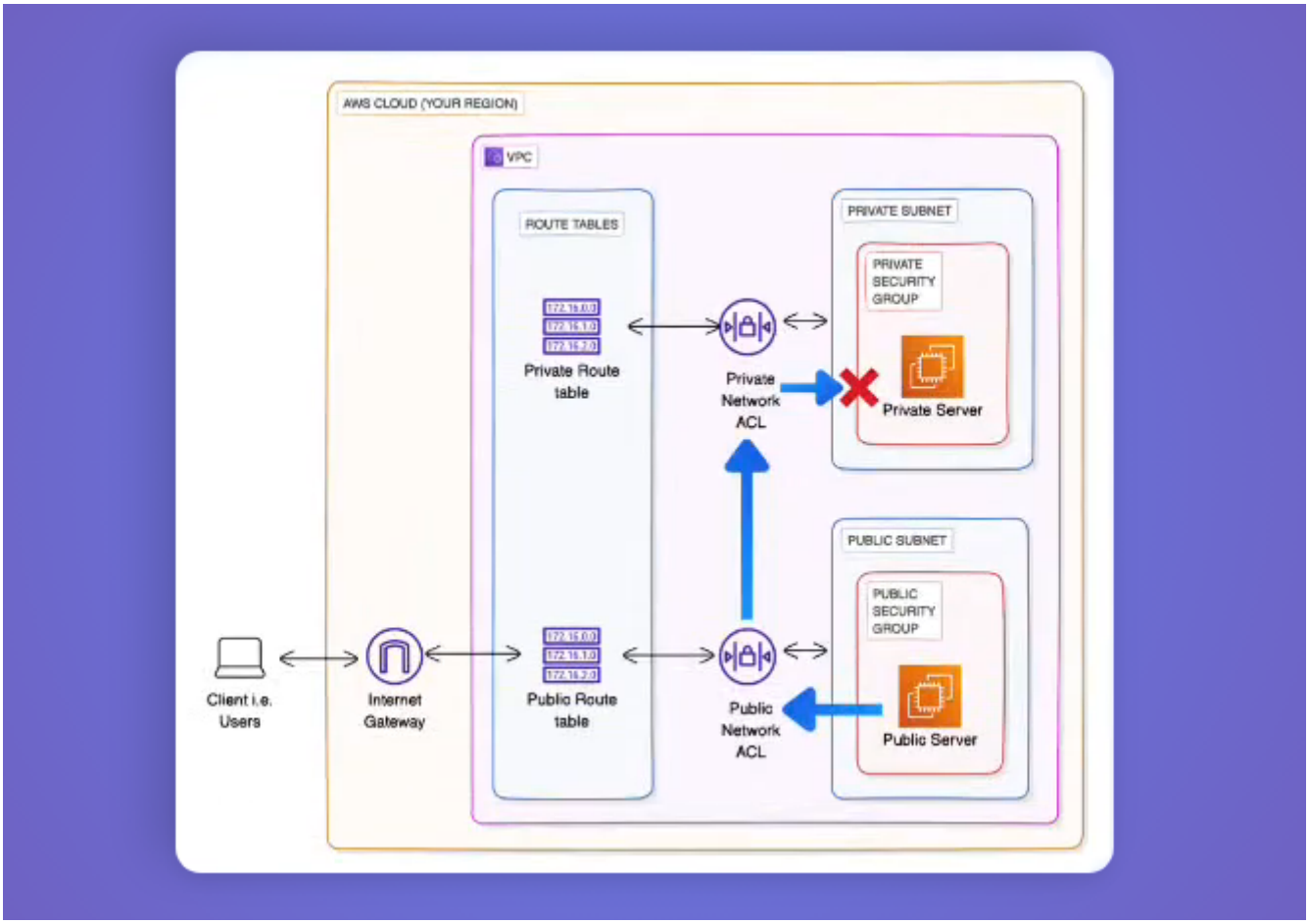
Initial Ping Test



Only one line! No replies received. The ping requests are being sent, but no responses are coming back.

What this means:

- Ping packets are leaving the Public Server ☑
- Ping packets are NOT reaching the Private Server OR
- Ping replies are being blocked from returning



Troubleshooting - Investigating Network ACLs

My Investigation Plan:

Check the following in order:

1. Route Tables (does traffic know where to go?)
2. Network ACLs (is traffic allowed at subnet level?)
3. Security Groups (is traffic allowed at instance level?)

Step 1 - Check Private Subnet Route Table:

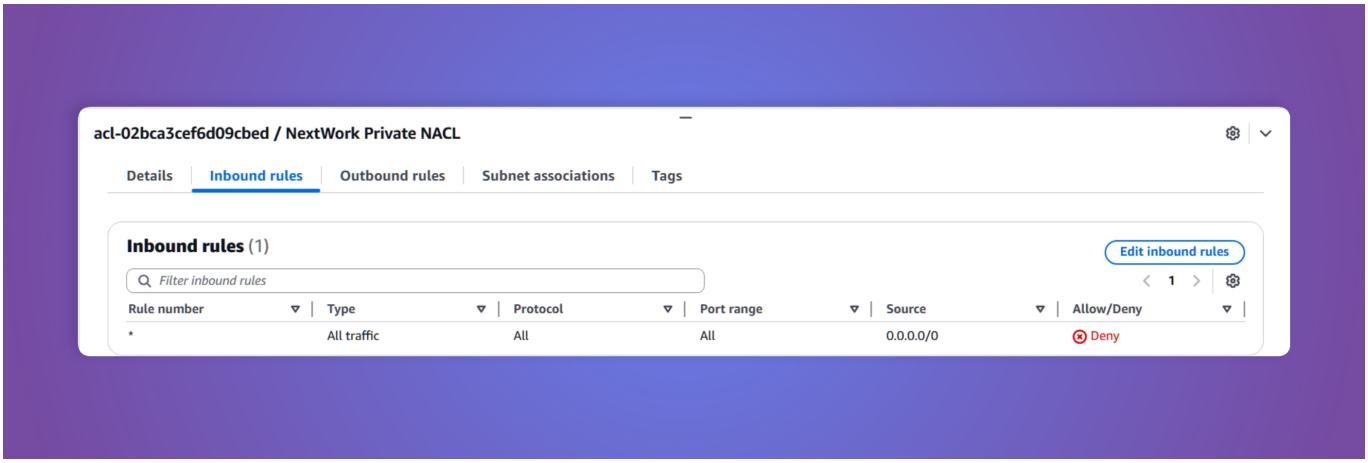
Private Route Table:

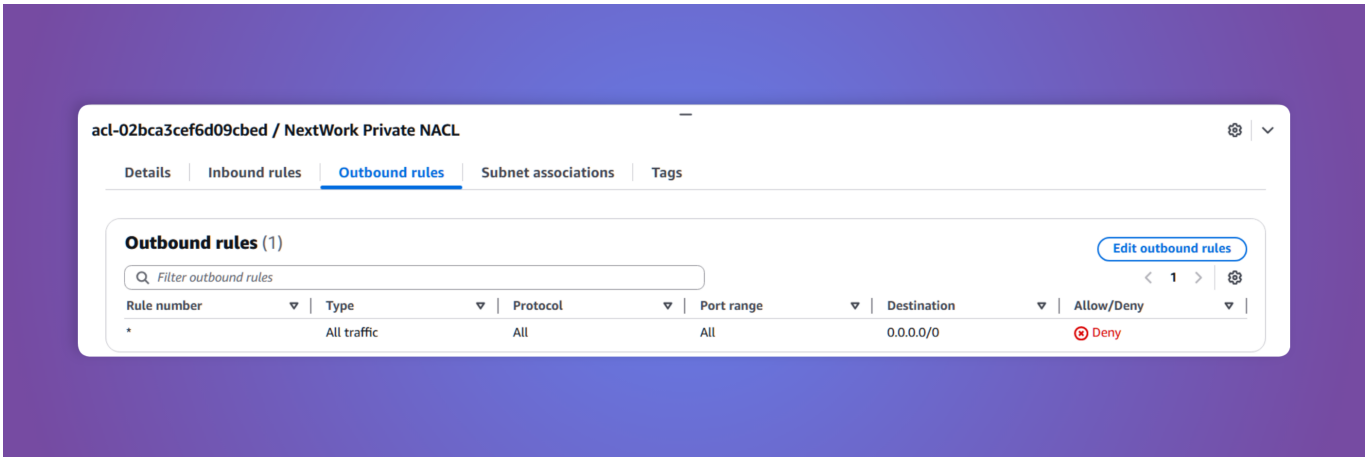
- 10.0.0.0/16 → local ☑ (correct!)

This means traffic from 10.0.0.0/24 knows how to reach 10.0.1.0/24 within the VPC.

Step 2 - Check Private Subnet Network ACL:

Discovery:

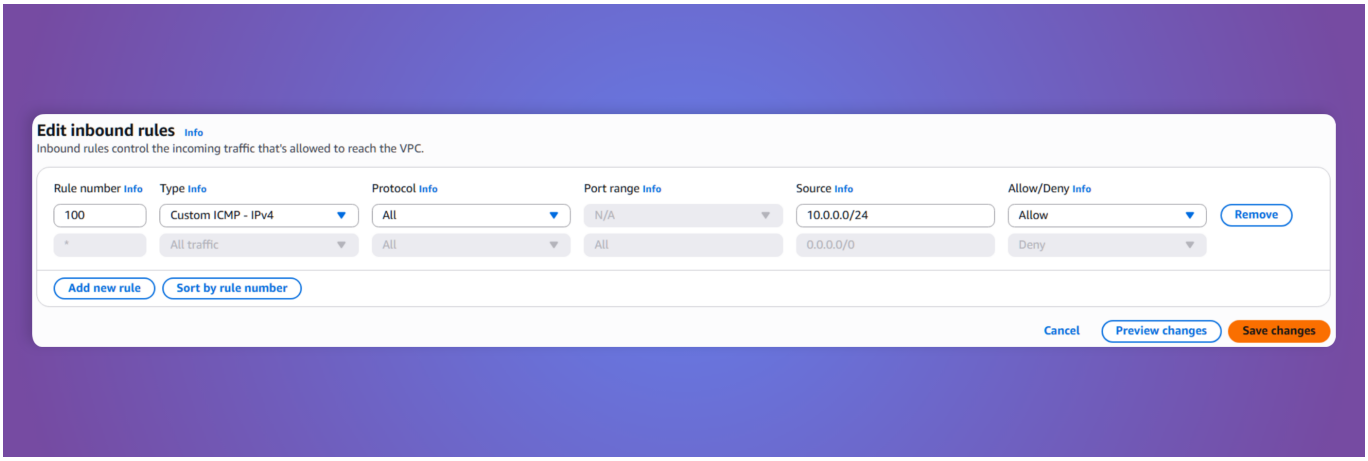




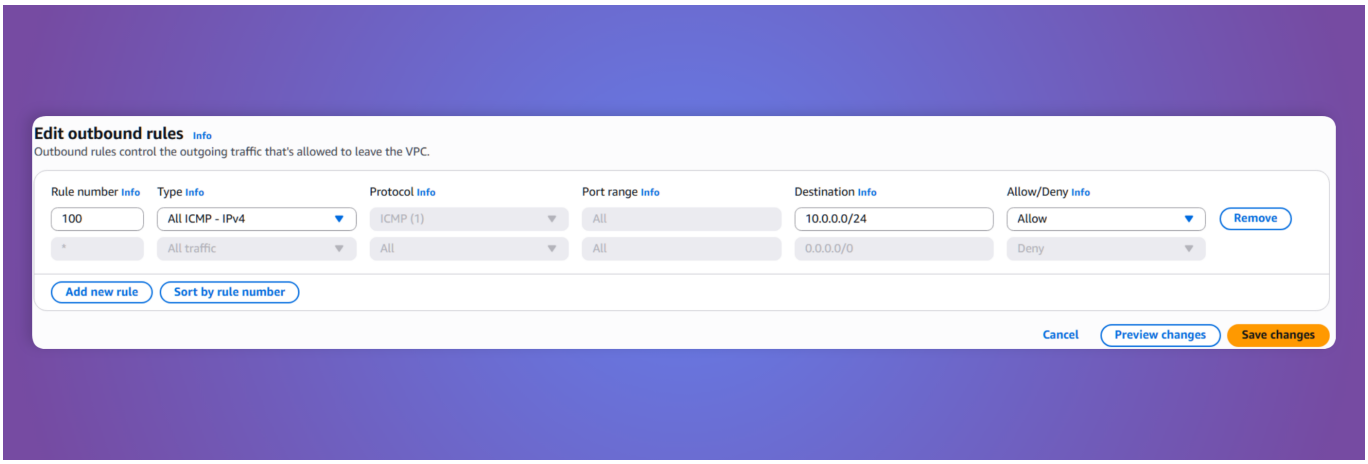
Root Cause Found!

The Private Network ACL is blocking ALL traffic in both directions. Even though the route table knows where to send traffic, the NACL acts as a firewall preventing it.

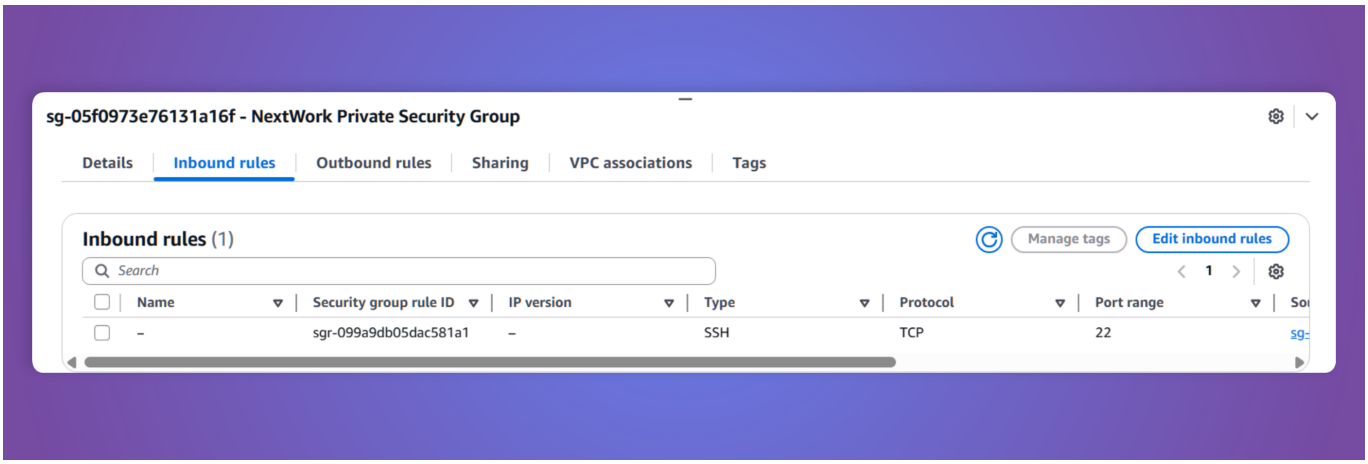
Fixing Network ACL - Inbound Rules



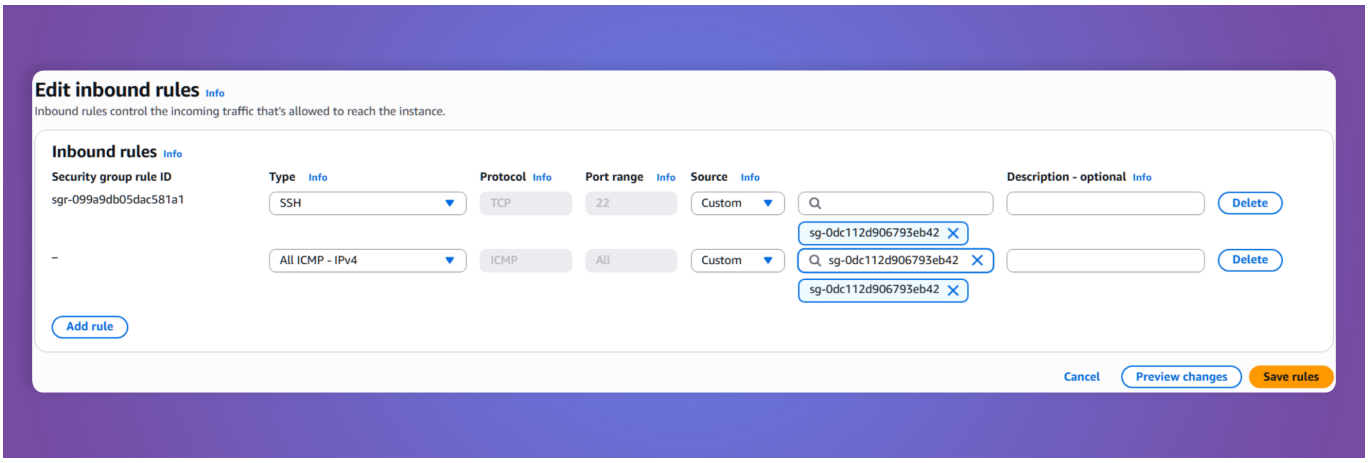
Fixing Network ACL - Outbound Rules



Still No Success - Security Group Investigation



Fixing Private Security Group



Why source = Security Group ID (not CIDR)?

More granular security! This rule allows ICMP ONLY from instances that have the Public Security Group attached.

Benefits:

- If I launch another instance in the Public Subnet with a different Security Group, it WON'T be able to ping the Private Server

- If an attacker compromises my laptop and tries to ping the Private Server directly, it's blocked (no public IP + different source)

NACL vs Security Group Source Difference:

Component	Source	Reason
Private NACL	10.0.0.0/24 (CIDR)	Subnet-level - must allow all traffic from public subnet
Private Security Group	sg-0dc112d906793eb42 (SG ID)	Instance-level - can be more granular and specific

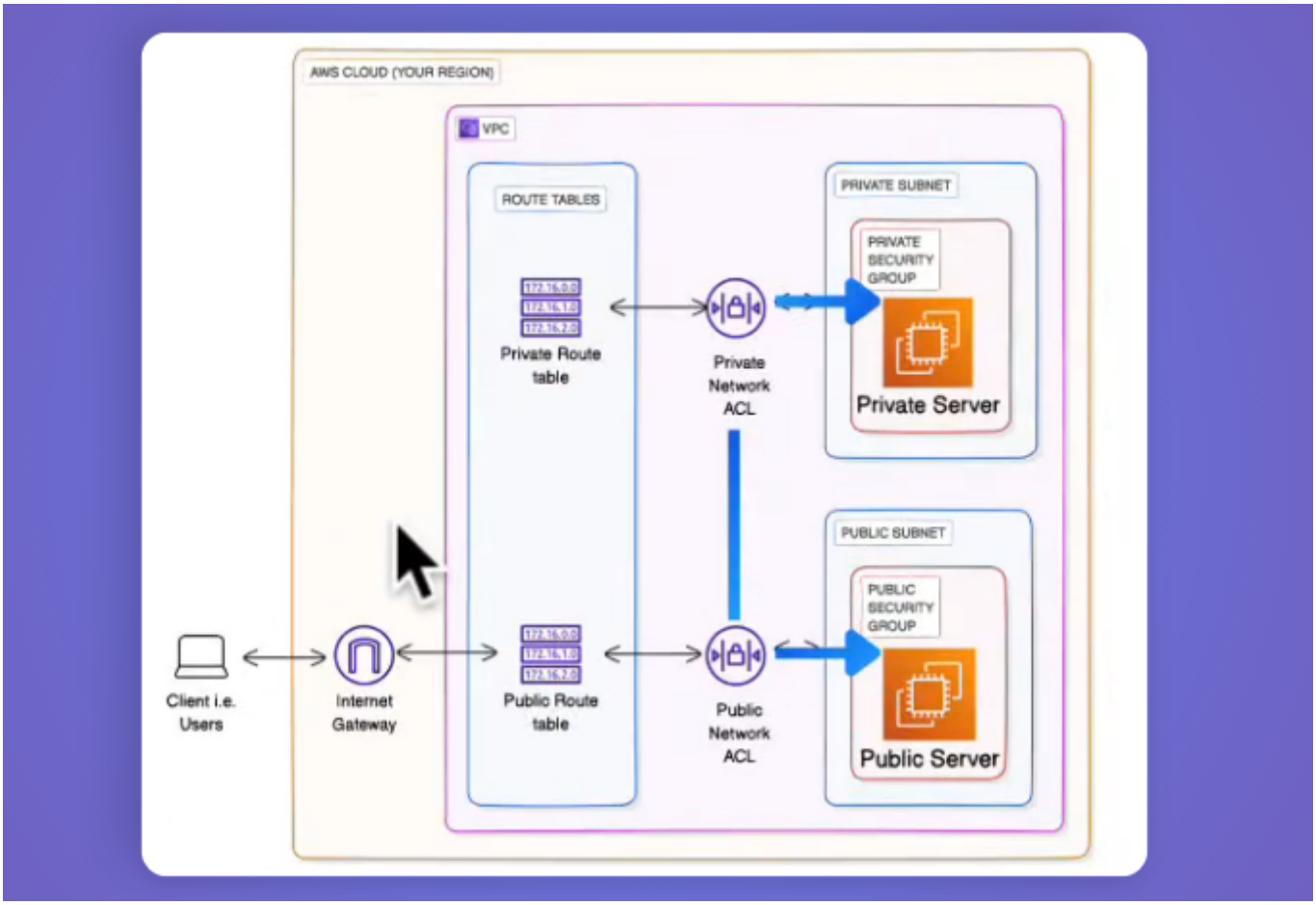
Successful Ping Test

Understanding the Output:


Successful Ping Responses

- **icmp_seq=15, 16, 17**: Sequce numbers (ping had been running, so numbers are higher)
- **ttl=64**: Time To Live = 64 hops remaining
- **time=0.856 ms**: Round-trip latency less than 1 millisecond (extremely fast!)

Updated Architecture Diagram:

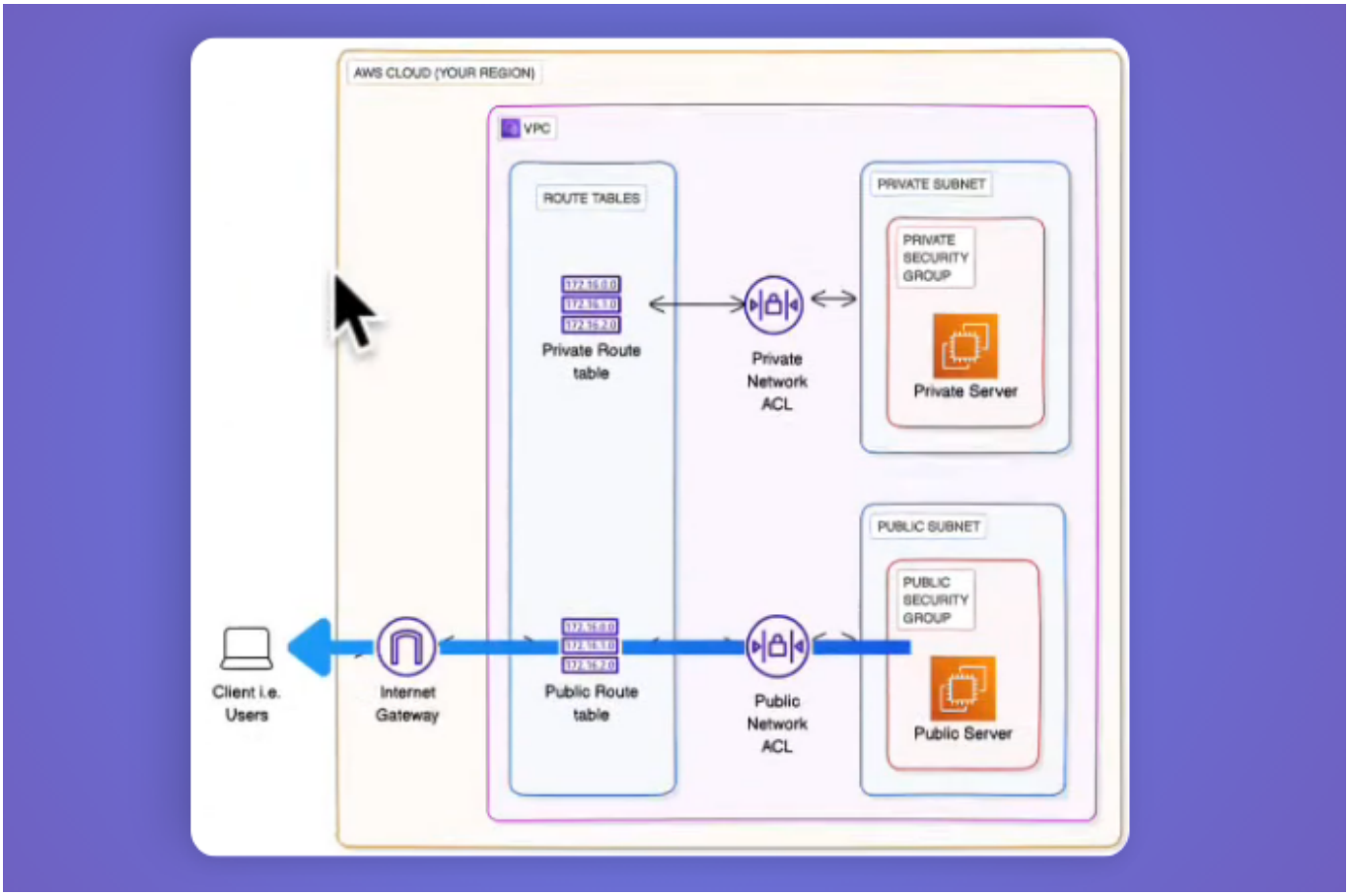


Step 3: Test VPC Internet Connectivity

Understanding Internet Connectivity

The Goal:
Verify that resources in the Public Subnet can access the internet through the Internet Gateway.

The Network Path:



Test 1 - Simple Website

What `curl` does:

1. Sends HTTP GET request to example.com
2. Receives HTML response
3. Displays HTML in terminal

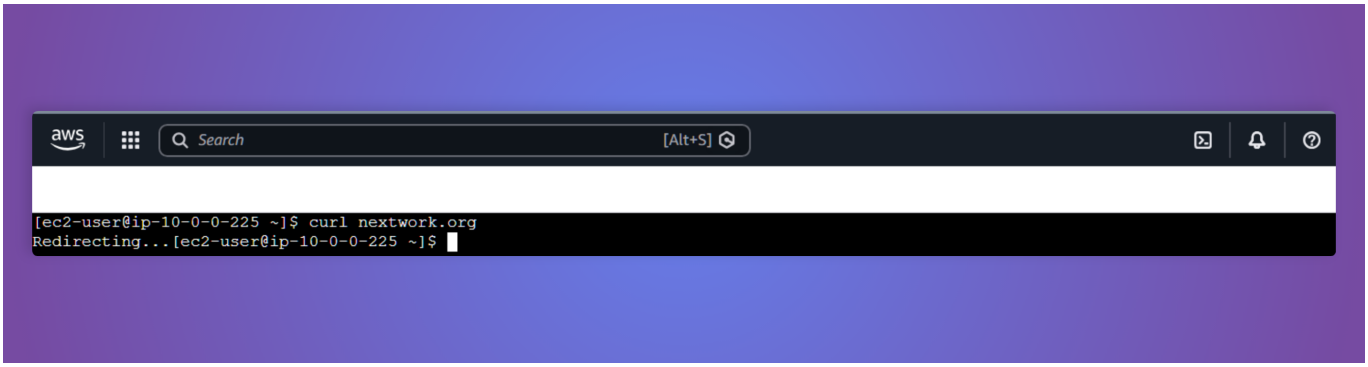
Result:

Success! ☑

The Public Server successfully retrieved HTML content from example.com, confirming:

- DNS resolution is working (domain → IP address)
- Outbound HTTP traffic is allowed by Security Group
- Route table correctly routes 0.0.0.0/0 to Internet Gateway
- Internet Gateway allows traffic to flow to internet

Test 2 - NextWork Website (With Redirect)



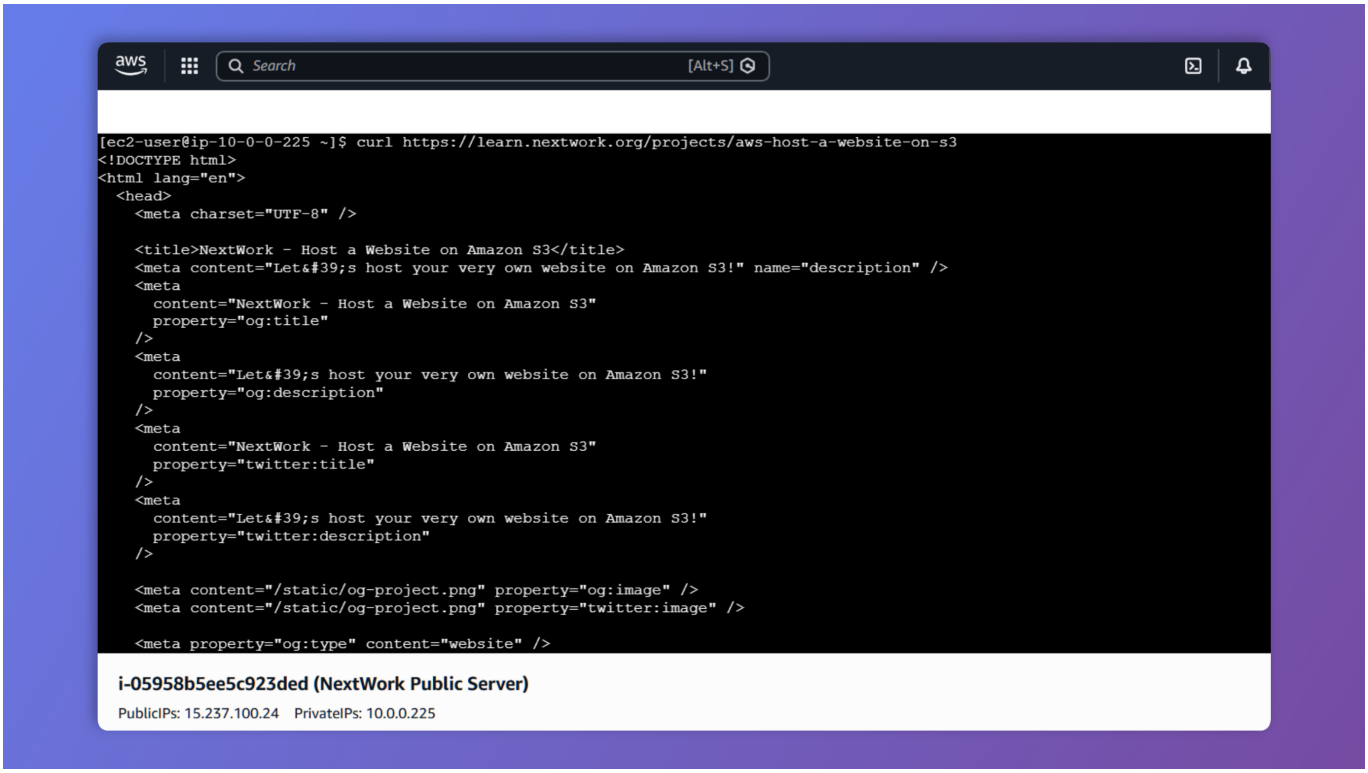
What happened?

The server returned an HTTP 302 redirect response. This means:

- nextwork.org redirects visitors to learn.nextwork.org
- The `curl` command received the redirect message but didn't automatically follow it

---

Test 3 - Following the Redirect



Success! 🎉
The Public Server retrieved the complete HTML document for the NextWork S3 hosting project page!

What this proves:

- HTTPS (port 443) traffic is allowed outbound
- TLS/SSL encryption is working
- DNS can resolve learn.nextwork.org
- Large data transfers work (entire webpage downloaded)
- Internet Gateway handles bidirectional traffic correctly

Cleanup

Resource Deletion Order

Important: Resources must be deleted in the correct order to avoid dependency errors!

1. Terminate EC2 Instances (dependent on VPC)
2. Delete Network Interfaces (if not auto-deleted)
3. Delete VPC (cascade deletes subnets, route tables, NACLs, security groups, IGW)

Conclusion

This project was a pivotal learning experience - I moved from building infrastructure to actually **testing and troubleshooting real network connectivity**!

What's Next?

This project is **Part 5** of the NextWork VPC series. In the next projects, I'll explore **Part 6: VPC Peering**

Acknowledgments

- **NextWork** for the comprehensive VPC challenge series
- **AWS Documentation** for detailed technical reference
- **AWS Free Tier** for making hands-on learning accessible