Creating a Private Subnet

Project Overview

This project builds upon my previous VPC networking work by creating a **private subnet** with its own isolated route table and Network ACL. I learned the critical differences between public and private subnets, mastered CIDR block planning to avoid overlaps, and implemented security controls for resources that should never have direct internet access - like databases, internal APIs, and backend services.

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

Table of Contents

What I Built

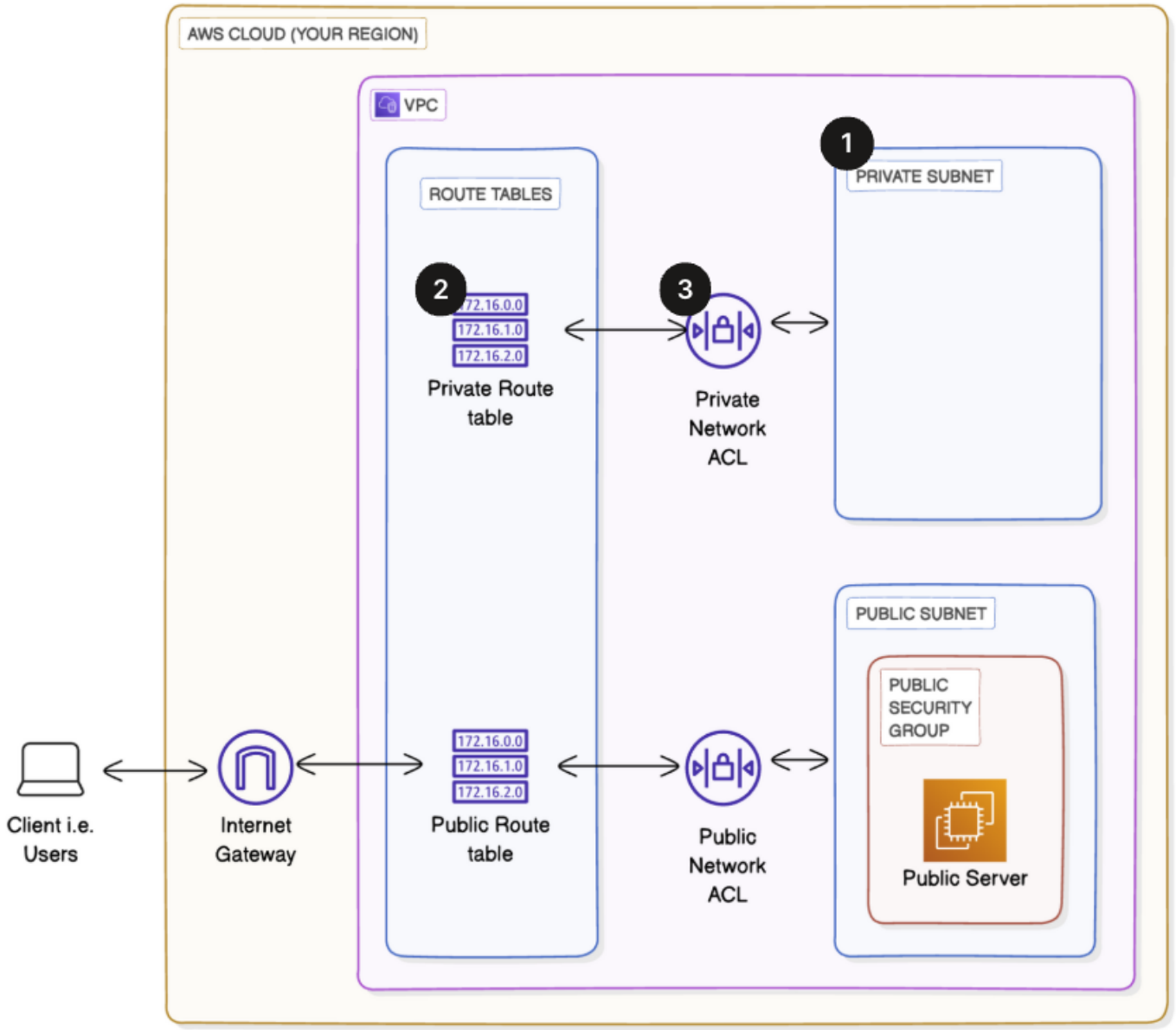
A complete private subnet infrastructure with isolated networking and security:

- **Private Subnet**: NextWork Private Subnet (10.0.1.0/24) - For backend resources without direct internet access
- **Private Route Table**: NextWork Private Route Table - Routes only local VPC traffic
- **Private Network ACL**: NextWork Private NACL - Subnet-level firewall for private resources

Existing Infrastructure (from Part 2):

- **VPC**: NextWork VPC (10.0.0.0/16)
- **Public Subnet**: NextWork Public Subnet (10.0.0.0/24)
- **Internet Gateway**: NextWork IG
- **Public Route Table**: Routes internet traffic to IGW
- **Public Network ACL**: NextWork Public NACL

Key Achievement: I now have a VPC with BOTH public and private subnets, mirroring real-world production architectures where web servers live in public subnets and databases live in private subnets.



Technologies & Concepts

AWS Services Used

- **Amazon VPC** - Virtual Private Cloud for isolated networking
- **Subnets** - Public and private network subdivisions
- **Route Tables** - Define traffic routing rules
- **Network ACLs** - Stateless firewalls at the subnet level
- **CIDR Notation** - IP address range planning

Key Concepts Learned

1. **Public vs Private Subnets**

   Public Subnet Characteristics:

   - Has a route to an Internet Gateway (0.0.0.0/0 → IGW)
   - Resources can receive public IP addresses
   - Accessible from the internet (with proper security group rules)

- Use case: Web servers, load balancers, bastion hosts

Private Subnet Characteristics:

- NO route to an Internet Gateway
- Resources only have private IP addresses
- NOT directly accessible from the internet
- Use case: Databases, application servers, internal APIs

2. **CIDR Block Planning**

Think of CIDR blocks like street addresses in a city. Each subnet needs its own unique range of addresses that doesn't overlap with others.

The Problem:

- My VPC has 10.0.0.0/16 (65,536 IP addresses)
- Public Subnet uses 10.0.0.0/24 (256 addresses: 10.0.0.0 to 10.0.0.255)
- Private Subnet CANNOT use 10.0.0.0/24 - that would be like having two streets with the same name!

The Solution:

- Private Subnet uses 10.0.1.0/24 (256 addresses: 10.0.1.0 to 10.0.1.255)
- This creates a separate, non-overlapping address range

3. **Route Table Isolation**

Each subnet needs its own route table to control traffic flow:

- **Public Route Table**: Sends internet traffic (0.0.0.0/0) to Internet Gateway
- **Private Route Table**: Only routes local VPC traffic (10.0.0.0/16)

Without a route to the Internet Gateway, private subnet resources cannot directly access or be accessed from the internet.

4. **Why Private Subnets Matter**

Real-world scenario:

- Your web application runs on EC2 instances in a public subnet
- Your database runs on RDS in a private subnet
- Users can access your website, but they can NEVER directly access your database
- Even if your web server is compromised, the attacker can't get direct internet access to your database

---

Step-by-Step Implementation

Step 1: Create Private Subnet

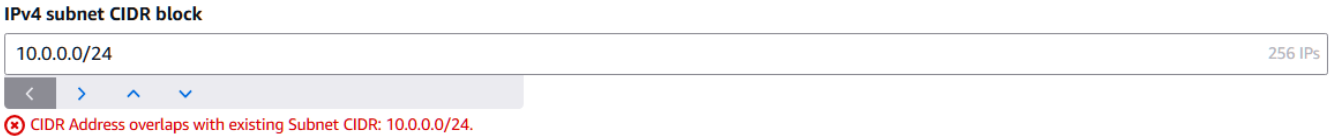
What I did:

I created a new subnet with a non-overlapping CIDR block to host private resources.

Understanding CIDR Block Requirements

The Challenge:

My first attempt failed! When I tried to create a private subnet with CIDR block 10.0.0.0/24, AWS rejected it with this error:



The Solution: CIDR Block Planning

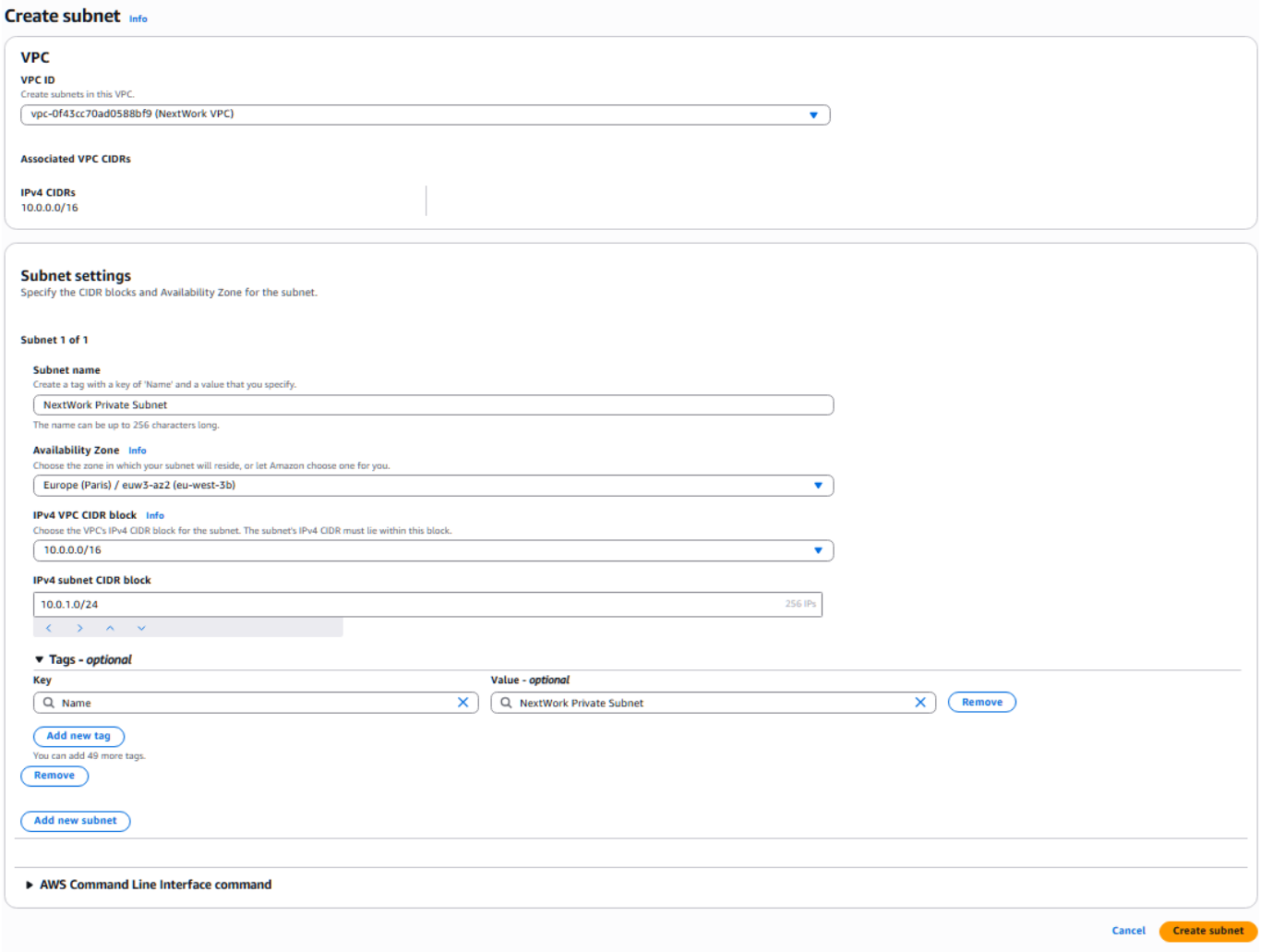
I needed to choose a different range within my VPC's 10.0.0.0/16 space:

- **VPC Range**: 10.0.0.0/16 → Provides 10.0.0.0 to 10.0.255.255
- **Public Subnet**: 10.0.0.0/24 → Uses 10.0.0.0 to 10.0.0.255 (256 addresses)
- **Private Subnet**: 10.0.1.0/24 → Uses 10.0.1.0 to 10.0.1.255 (256 addresses)

Creating the Subnet with Correct CIDR

Subnet Configuration:

I configured the private subnet with proper settings:



Key Decisions:

1. **Different Availability Zone**: I chose eu-west-3b while my public subnet is in eu-west-3a. This provides high availability - if one AZ has issues, the other stays operational.
2. **Non-overlapping CIDR**: 10.0.1.0/24 doesn't conflict with 10.0.0.0/24.

Step 2: Create Private Route Table

What I did:

I created a dedicated route table for my private subnet that routes ONLY local VPC traffic, with no route to the Internet Gateway.

Why a Separate Route Table?

The Problem:

By default, new subnets are associated with the VPC's main route table. In my case, the main route table (NextWork route table) has a route to the Internet Gateway:

- Route 1: 10.0.0.0/16 → local
- Route 2: 0.0.0.0/0 → igw-05408184cd4328c04

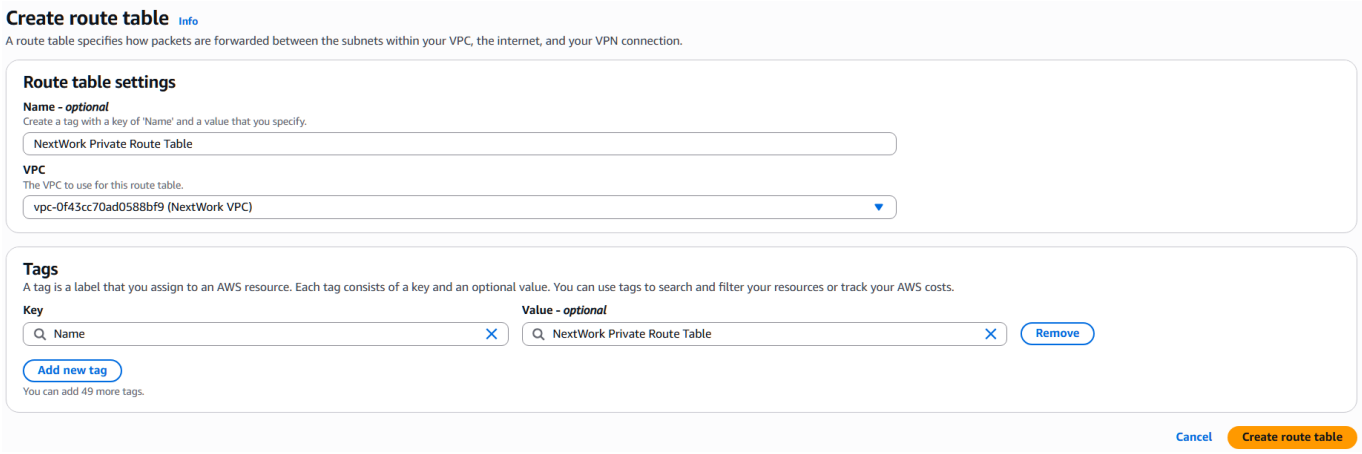
If my private subnet uses this route table, it would become a PUBLIC subnet!

The Solution:

Create a new route table with ONLY the local route, ensuring no internet access.

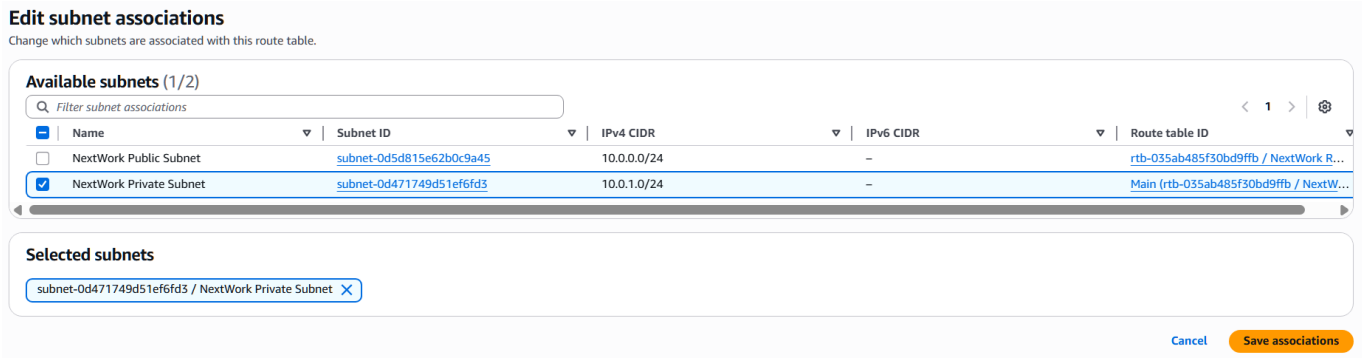
Creating NextWork Private Route Table

Route Table Creation:



Associating the Route Table with Private Subnet

Edit Subnet Associations:



Step 3: Create Private Network ACL

What I did:

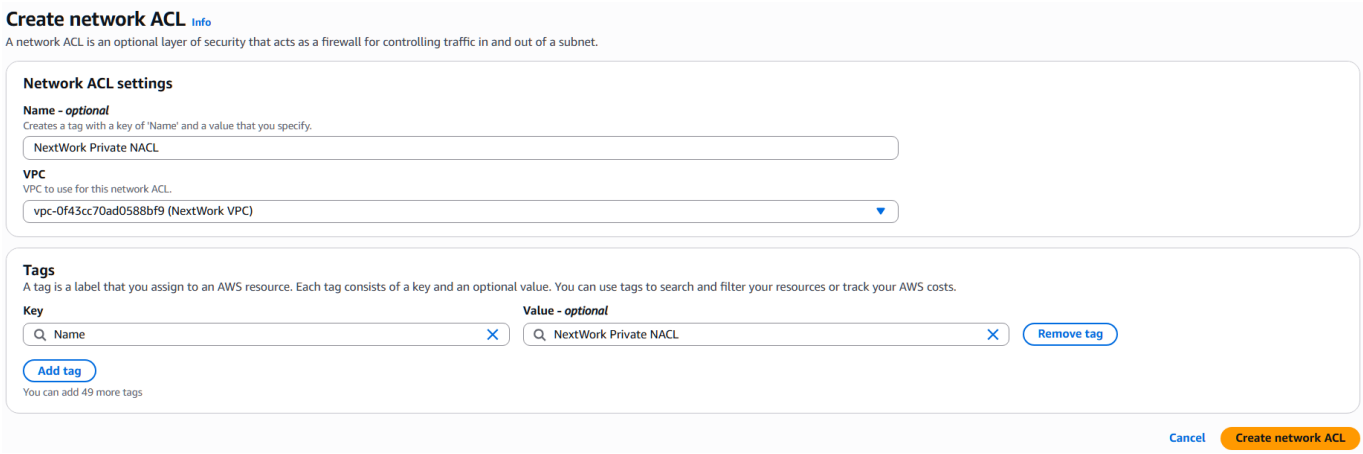
I created a custom Network ACL for my private subnet to add an additional layer of security at the subnet boundary.

Default NACL Behavior:

By default, subnets use the VPC's default NACL, which allows ALL traffic. For a private subnet, I want explicit control over traffic rules.

Creating NextWork Private NACL

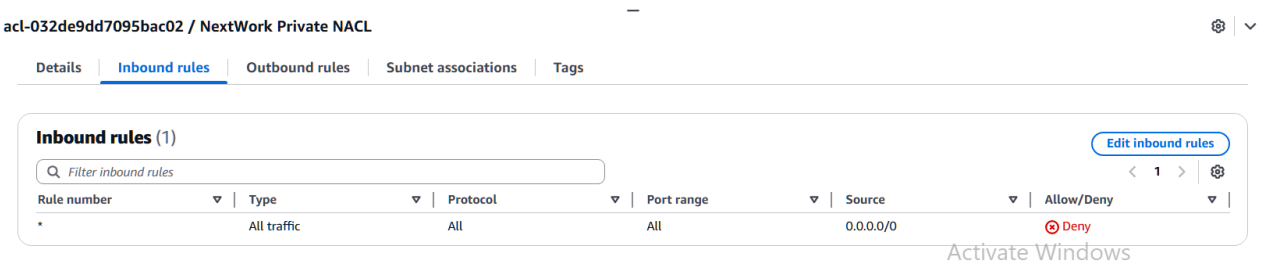
Network ACL Creation:



Default Rules:

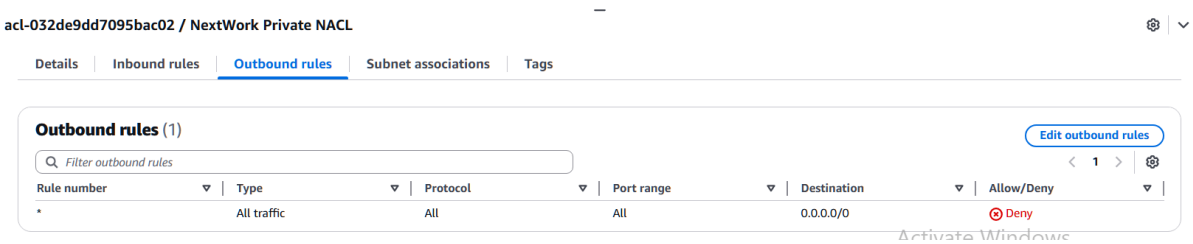
When created, a custom NACL starts with very restrictive rules:

Inbound Rules:



- Activate Windows

Outbound Rules:



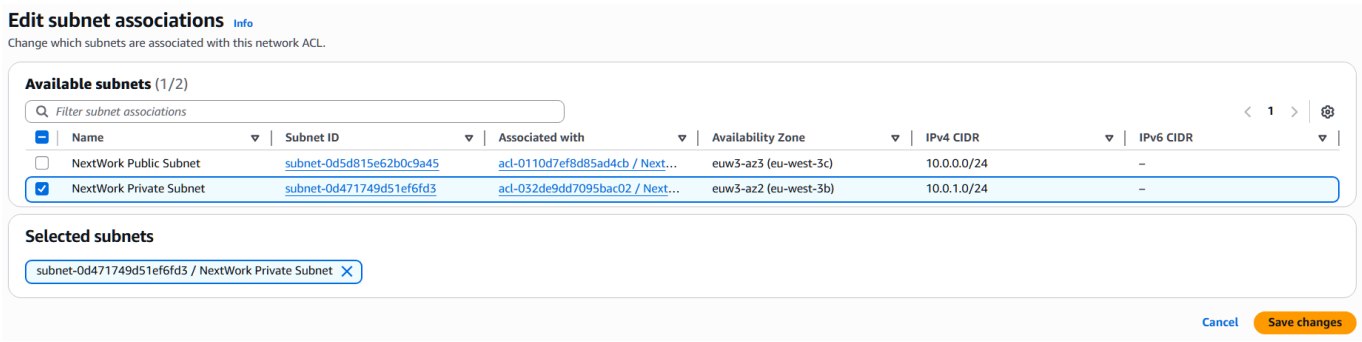
- Activate Windows

⚠ **Critical Point:** Custom NACLs DENY everything by default! This is opposite of the default NACL which allows everything.

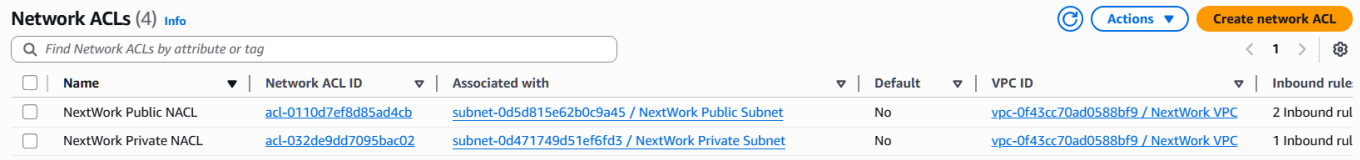
Associating NACL with Private Subnet

Edit Subnet Associations:

I associated the custom NACL with my private subnet:



Final Network ACL List:



Conclusion

This project taught me the fundamental principles of network segmentation in AWS. By creating a private subnet with isolated routing and security controls, I learned how to:

Key Takeaways:

1. ☑ **Distinguish between public and private subnets** - It's all about the route table!
2. ☑ **Plan CIDR blocks** to avoid overlaps and maximize VPC address space
3. ☑ **Create isolated route tables** - Private subnets need their own routing configuration

---

What's Next?

This project is **Part 3** of the NextWork VPC series. In the next projects it about launching resources into our VPC

Project Completed: December 2025
Author: YOUHAD AYOUB
Region: eu-west-3 (Paris)
NextWork Challenge: AWS Beginners Challenge - Project 6

This project was completed as part of the NextWork AWS Beginners Challenge. By building upon the networking foundation from Part 2, I gained hands-on experience with subnet isolation, CIDR planning, and the architectural principles that separate public from private network segments. Special thanks to the NextWork community for their excellent project structure and clear explanations of AWS networking concepts.