VPC Traffic Flow and Security

Project Overview

This project demonstrates advanced AWS networking by configuring traffic flow and implementing multi-layered security controls in a Virtual Private Cloud. I successfully set up route tables to direct internet-bound traffic, created security groups for resource-level protection, and deployed Network ACLs for subnet-level security - building a production-ready network infrastructure with defense in depth.

Project Duration: Approximately 90 minutes
Difficulty Level: Easy
AWS Region Used: eu-west-3 (Paris)

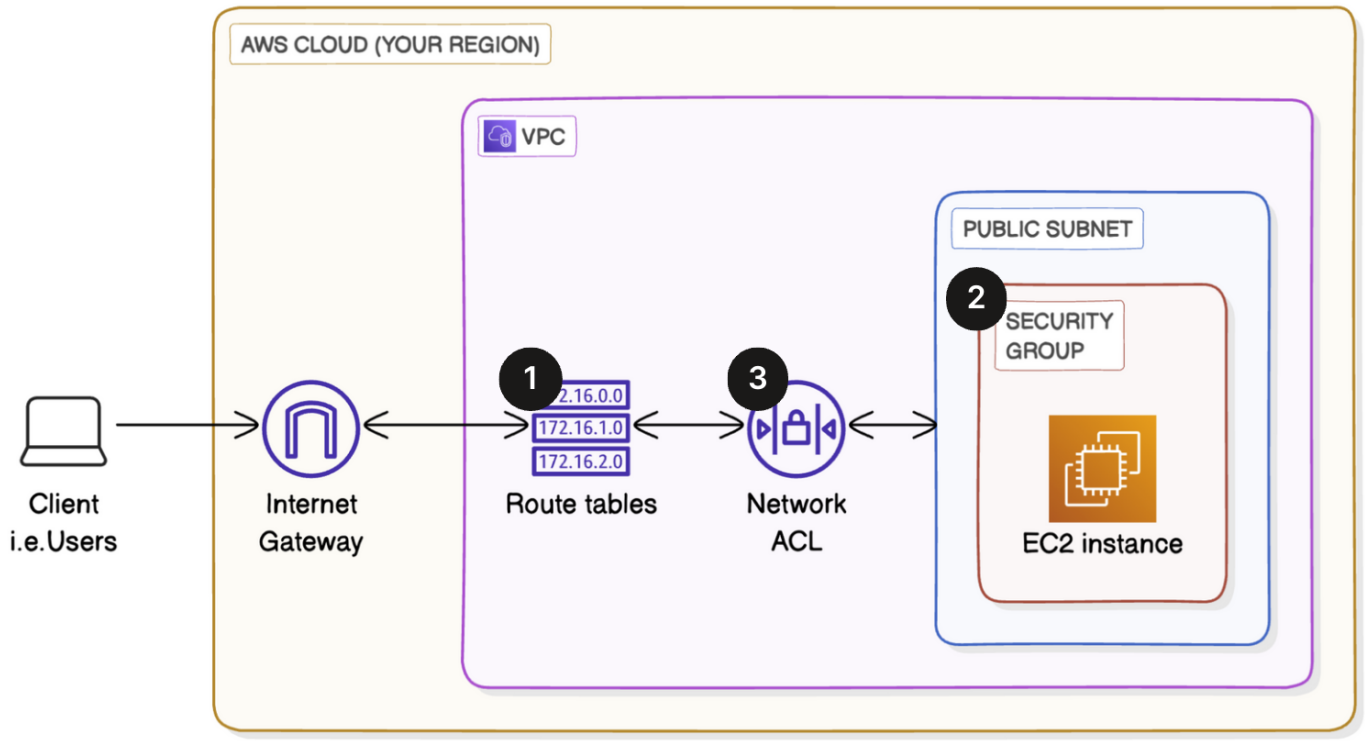
Table of Contents

- What I Built
- Technologies & Concepts
- Step-by-Step Implementation
  - Step 1: Create VPC and Subnet
  - Step 2: Create Route Tables
  - Step 3: Create Security Group
  - Step 4: Create Network ACL
- Understanding Traffic Flow
- Cleanup
- Conclusion

What I Built

A complete VPC networking infrastructure with layered security:

- **VPC**: NextWork VPC (10.0.0.0/16) - Private network space
- **Public Subnet**: Public 1 (10.0.1.0/24) - For internet-accessible resources
- **Internet Gateway**: NextWork IG - Connection to the internet
- **Custom Route Table**: Routes internet traffic (0.0.0.0/0) to Internet Gateway
- **Security Group**: NextWork Security Group - Resource-level firewall with HTTP access
- **Custom Network ACL**: NextWork Network ACL - Subnet-level firewall allowing all traffic



This architecture demonstrates the three critical layers of AWS networking:

1. **Connectivity Layer**: Route tables directing traffic flow
2. **Resource Security Layer**: Security groups protecting individual resources
3. **Subnet Security Layer**: Network ACLs guarding entire subnets

---

Technologies & Concepts

AWS Services Used

- **Amazon VPC** - Virtual Private Cloud for isolated networking
- **Subnets** - Network subdivisions within the VPC
- **Internet Gateway** - Enables internet connectivity
- **Route Tables** - Define traffic routing rules
- **Security Groups** - Stateful firewalls at the resource level
- **Network ACLs** - Stateless firewalls at the subnet level

Key Networking Concepts Learned

1. **Route Tables**
    - GPS systems for network traffic
    - Define where packets should go based on destination
    - Each subnet must be associated with exactly one route table
    - Default route (0.0.0.0/0) sends all internet traffic to Internet Gateway
2. **Security Groups**
    - Virtual firewalls for individual resources (EC2 instances)
    - **Stateful**: Return traffic is automatically allowed

- Applied at the resource level
- Rules specify: Protocol, Port, and Source/Destination
- By default, allow all outbound traffic

3. **Network ACLs**
   - Traffic cops at subnet entry/exit points
   - **Stateless**: Must explicitly allow both inbound and outbound traffic
   - Applied at the subnet level
   - Rules are evaluated in numbered order (100, 200, etc.)
   - Asterisk (*) rule acts as catch-all deny
4. **Defense in Depth**
   - Multiple security layers protect resources
   - Traffic must pass through: Network ACL → Subnet → Security Group → Resource
   - If any layer blocks traffic, the request fails
   - Provides redundancy and granular control
5. **Public vs Private Subnets**
   - **Public subnet requirements**:
     - Route to Internet Gateway (0.0.0.0/0 → IGW)
     - Resources have public IP addresses
     - Network ACL allows traffic
   - **Private subnets**: No route to Internet Gateway

Step-by-Step Implementation

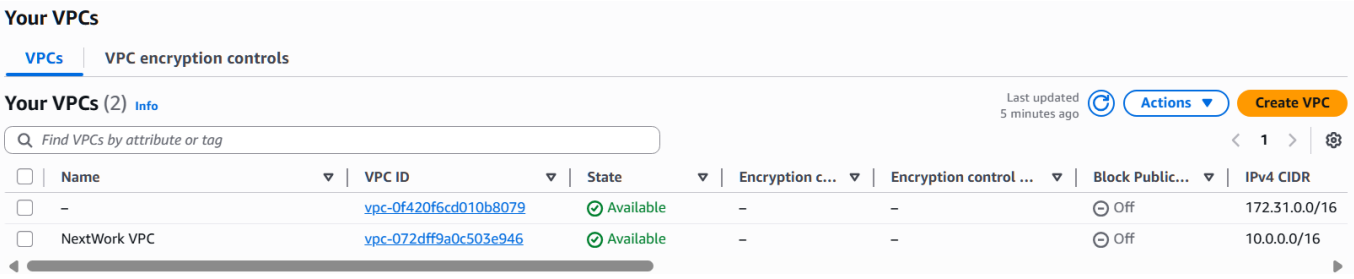
Step 1: Create VPC and Subnet

What I did:

I started by creating the foundational networking components - a VPC and a public subnet.

Create the VPC

VPC Configuration:



- **Name**: NextWork VPC
- **VPC ID**: vpc-072dff9a0c503e946
- **IPv4 CIDR**: 10.0.0.0/16

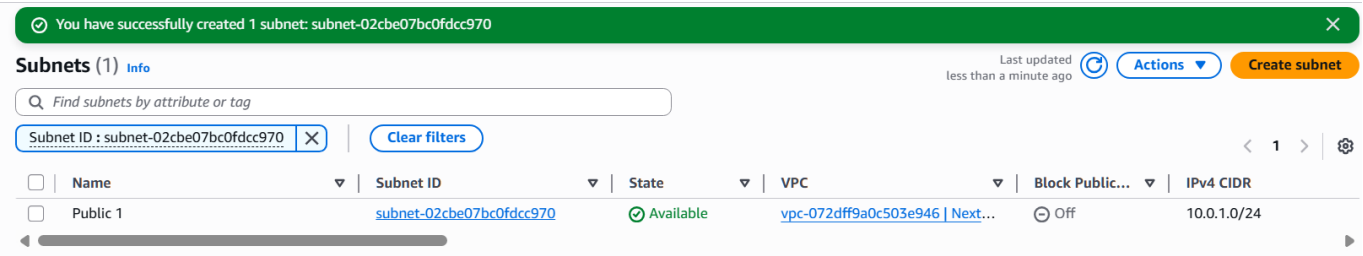
- **State**: Available

Why this matters:

The VPC is the container for all networking resources. The 10.0.0.0/16 CIDR block provides 65,536 IP addresses, giving plenty of room for growth.

Create the Public Subnet

Subnet Configuration:



- **Name**: Public 1
- **Subnet ID**: subnet-02cbe07bc0fdcc970
- **VPC**: vpc-072dff9a0c503e946 (NextWork VPC)
- **Availability Zone**: eu-west-3a
- **IPv4 CIDR**: 10.0.1.0/24
- **State**: Available
- **Available IP addresses**: 251 (out of 256 total)

Key Point:

Even though it's named "Public 1", this subnet isn't actually public yet! It needs a route to an Internet Gateway to become truly public.

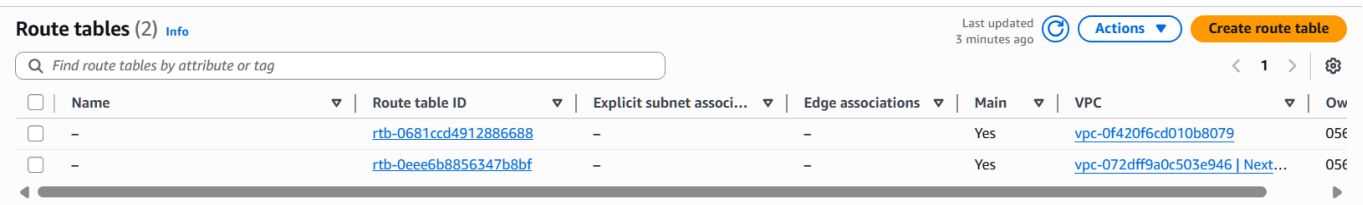
Step 2: Create Route Tables

What I did:

I configured route tables to direct traffic from my subnet to the internet via the Internet Gateway.

Understanding Route Tables

Route tables are like GPS systems - they tell network traffic where to go based on the destination IP address.

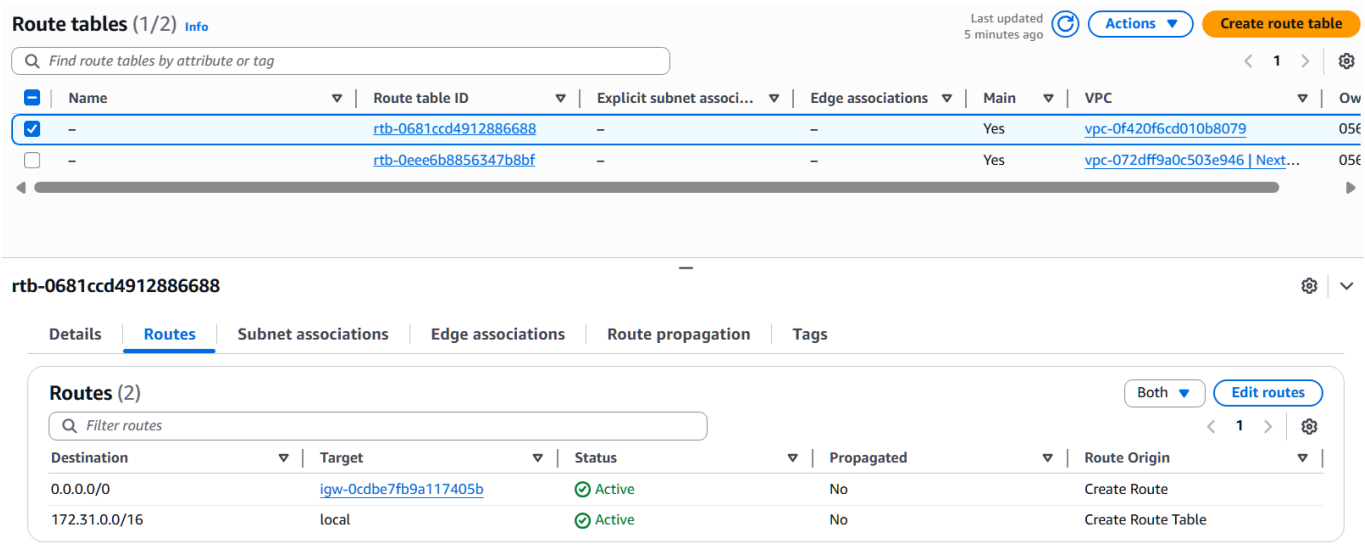


I can see two route tables in my account:

- **rtb-0681ccd4912886688**: Default VPC's main route table
- **rtb-0eee6b8856347b8bf**: NextWork VPC's main route table (this is the one I'm working with)

Examining the Default Route Table

Default VPC Route Table:



The default VPC already has internet connectivity configured:

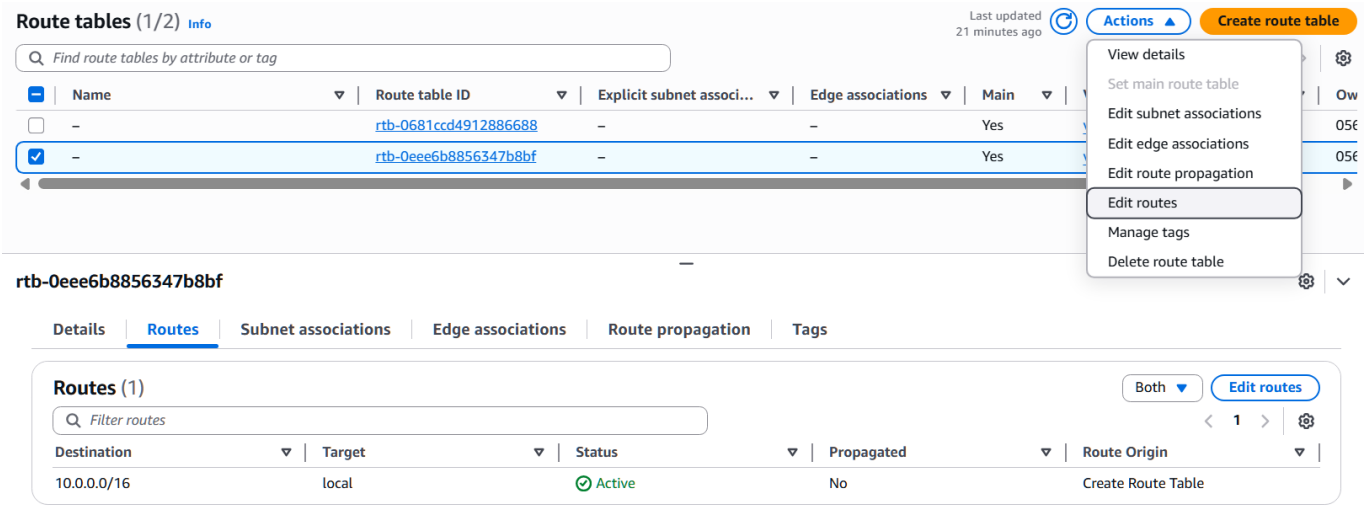
- **Route 1**: 0.0.0.0/0 → igw-0cdbe7fb9a117405b (Internet Gateway)
- **Route 2**: 172.31.0.0/16 → local (internal VPC traffic)

What this means:

- Any traffic destined for the internet (0.0.0.0/0) goes to the Internet Gateway
- Traffic within the VPC (172.31.0.0/16) stays local

Examining NextWork VPC's Route Table

NextWork VPC Route Table (Before):



Initially, my route table only has one route:

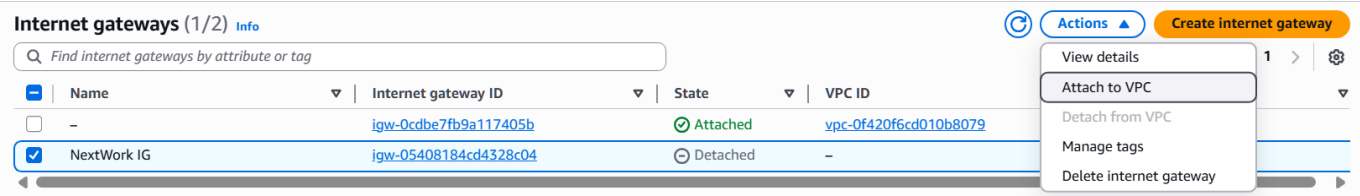
- **Route**: 10.0.0.0/16 → local

Problem: There's no route to the Internet Gateway, so traffic can't reach the internet!

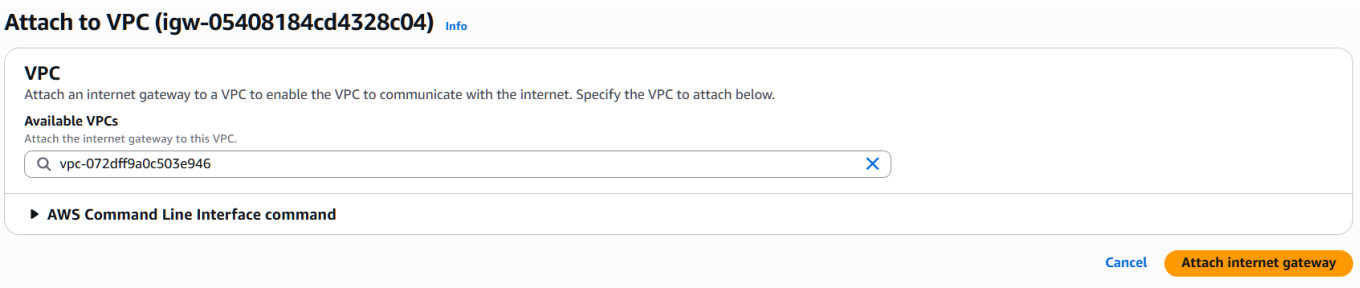
Adding the Internet Gateway Route

Creating the Internet Gateway:

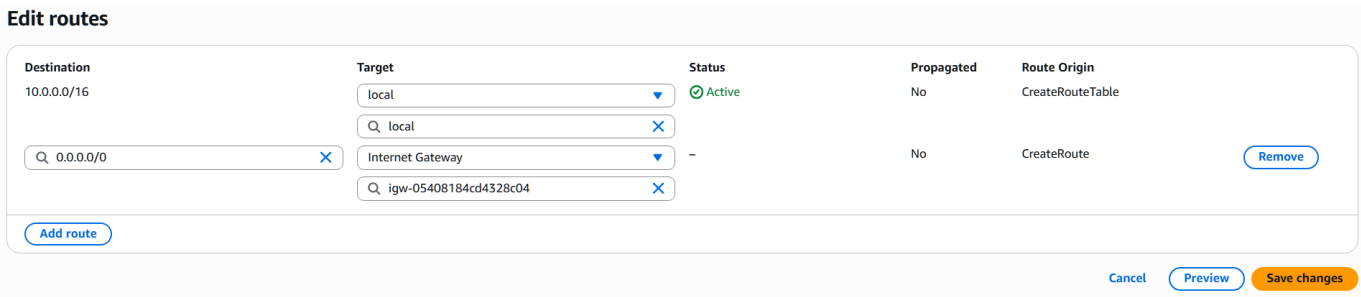
First, I created and attached an Internet Gateway to my VPC:



- **Name**: NextWork IG
- **Internet Gateway ID**: igw-05408184cd4328c04
- **State**: Attached
- **VPC**: vpc-072dff9a0c503e946



Editing Routes:



I added a new route to direct internet-bound traffic to the Internet Gateway:

- **Destination**: 0.0.0.0/0 (all IP addresses)
- **Target**: Internet Gateway (igw-05408184cd4328c04)

Why 0.0.0.0/0?

This is called a "default route" - it matches ANY IP address that doesn't match a more specific route. When traffic doesn't match the local route (10.0.0.0/16), it takes this route to the internet.

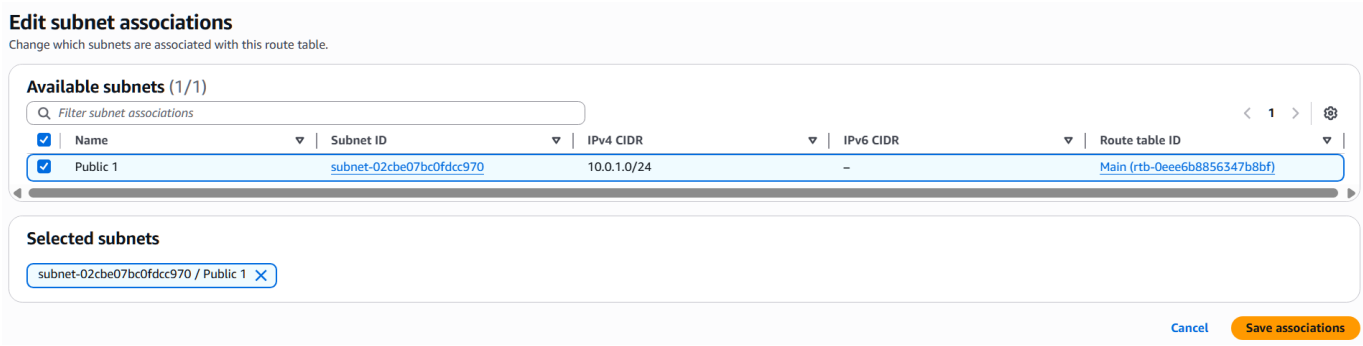
Route Evaluation:

Routes are evaluated from most specific to least specific:

1. First, check if destination is in 10.0.0.0/16 → send to local
2. If not, check 0.0.0.0/0 → send to Internet Gateway

Associate Route Table with Subnet

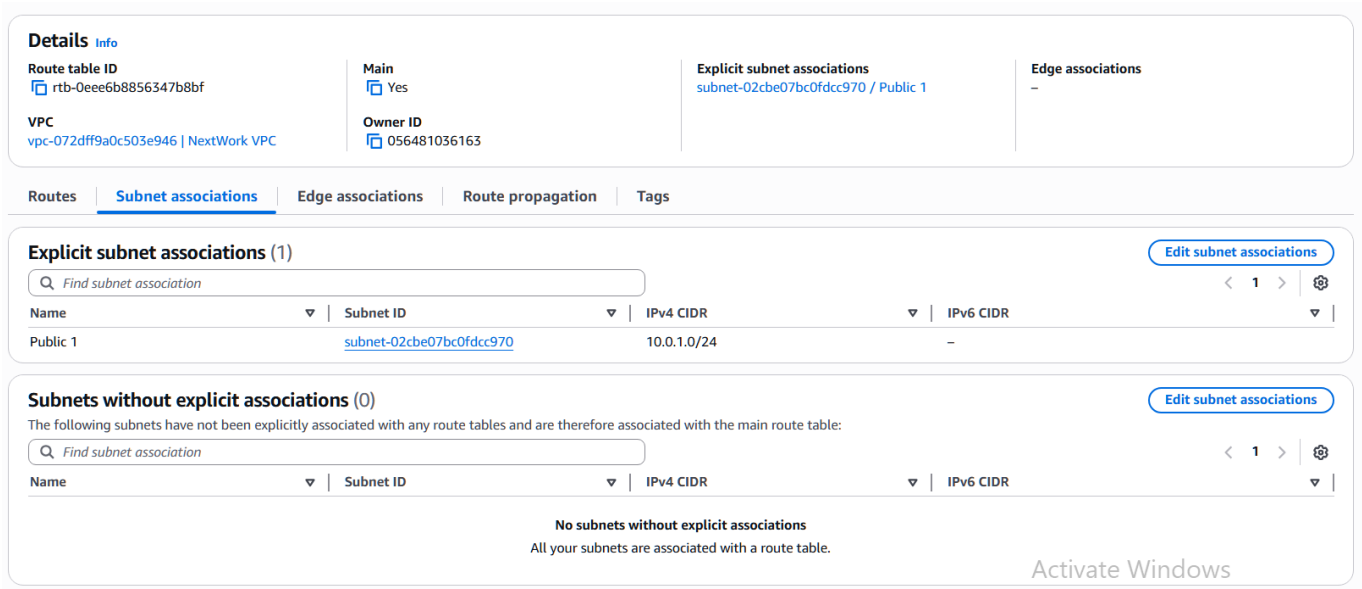
Edit Subnet Associations:



I associated my Public 1 subnet with the route table:

- **Selected subnet**: subnet-02cbe07bc0fdcc970 / Public 1
- **Route table**: Main (rtb-0eee6b8856347b8bf)

Result:



The route table now shows:

- **Explicit subnet associations**: subnet-02cbe07bc0fdcc970 / Public 1
- **VPC**: vpc-072dff9a0c503e946 | NextWork VPC
- **Main**: Yes

Success! My subnet now has a path to the internet through the route table! 🎉

Step 3: Create Security Group

What I did:

I created a security group to control inbound and outbound traffic at the resource level (for EC2 instances).

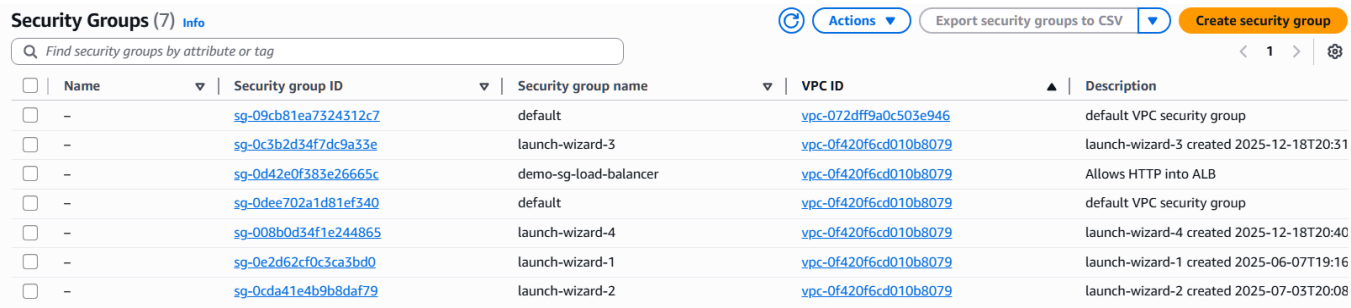
What Are Security Groups?

If VPCs are cities and subnets are neighborhoods, **security groups are security checkpoints at each building** (resource). They check who's coming in and going out.

Key Characteristics:

- Work at the **resource level** (EC2 instances, databases, etc.)
- **Stateful**: If you allow inbound traffic, the response is automatically allowed
- By default, allow all outbound traffic
- Rules specify: Type, Protocol, Port, and Source/Destination

Existing Security Groups



I can see several existing security groups:

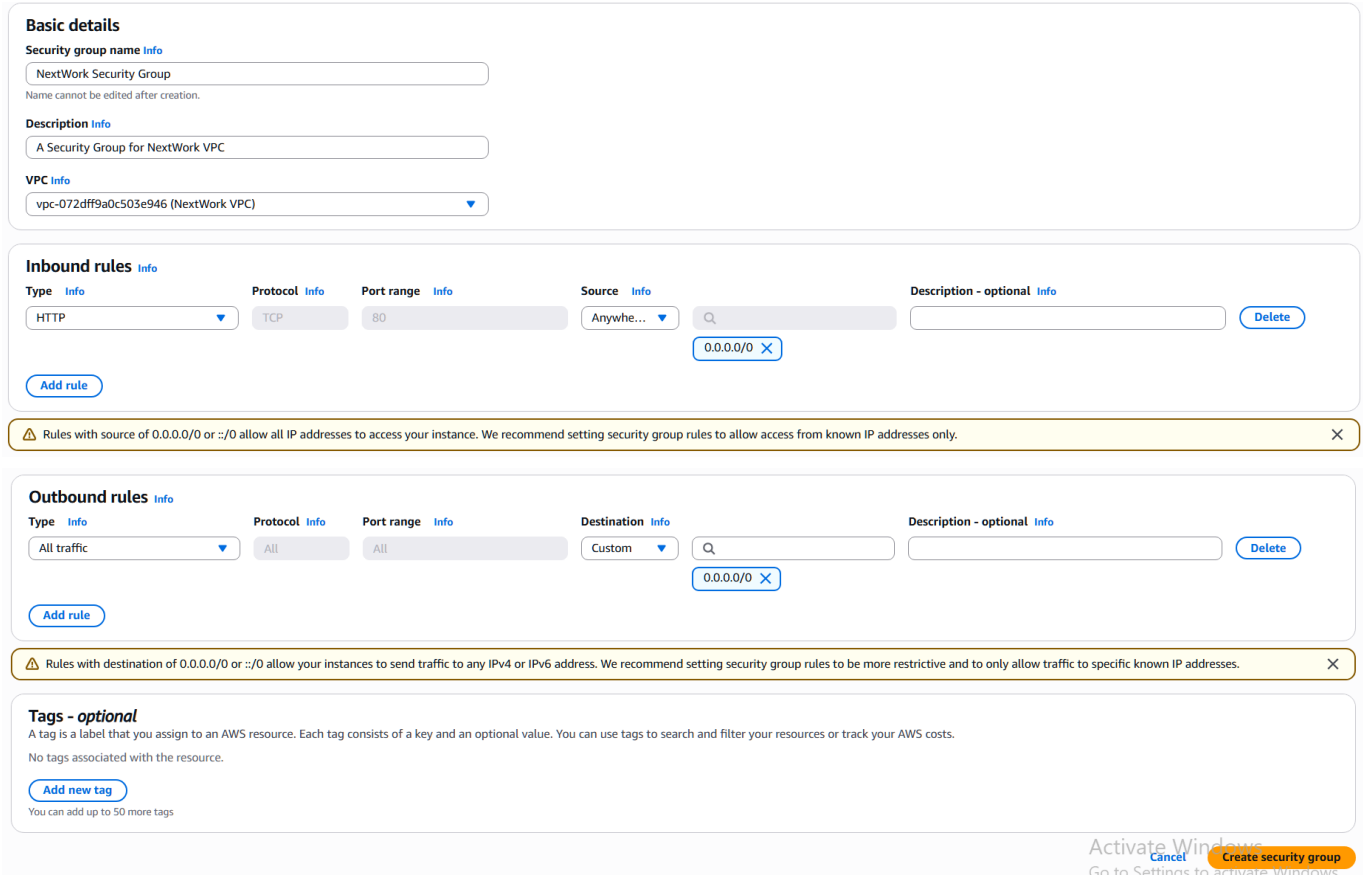
- **sg-09cb81ea7324312c7**: default (for NextWork VPC)
- **sg-0c3b2d34f7dc9a33e**: launch-wizard-3
- **sg-0d42e0f383e26665c**: demo-sg-load-balancer
- **sg-0dee702a1d81ef340**: default (for default VPC)
- And more...

Why are there existing security groups?

AWS automatically creates a default security group for every VPC. This allows internal communication between resources in the same VPC.

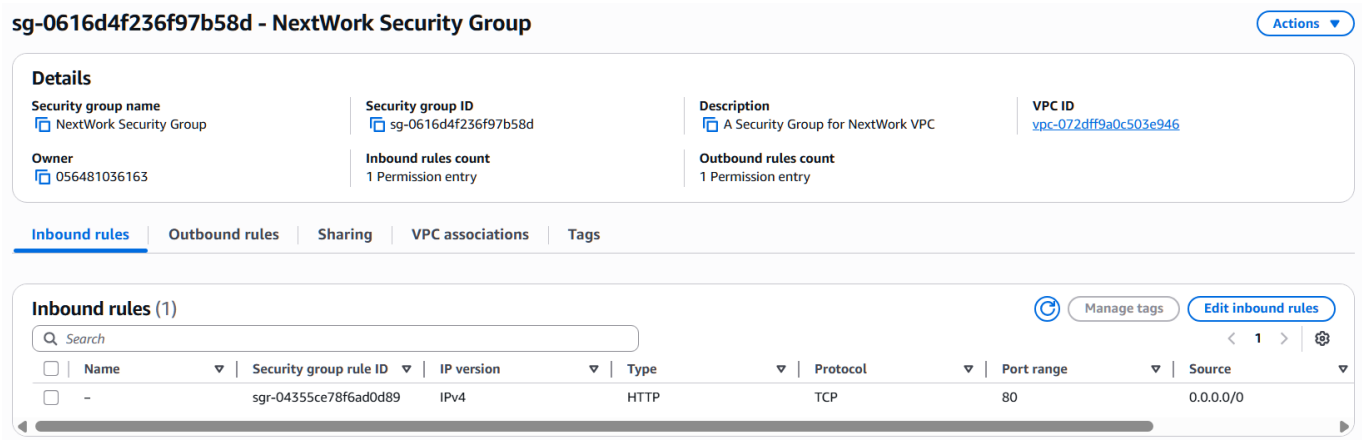
Creating NextWork Security Group

Basic Configuration:



- **Security group name**: NextWork Security Group
- **Security group ID**: sg-0616d4f236f97b58d
- **Description**: A Security Group for NextWork VPC
- **VPC ID**: vpc-072dff9a0c503e946
- **Inbound rules count**: 1 Permission entry
- **Outbound rules count**: 1 Permission entry

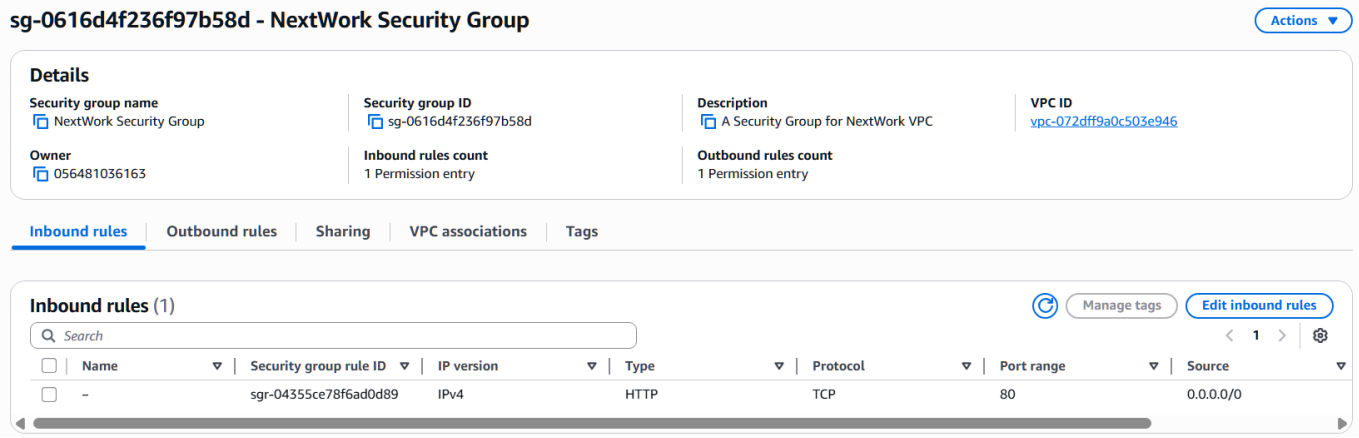
Inbound Rules:



I configured one inbound rule:

- **Type**: HTTP
- **Protocol**: TCP (automatically filled)
- **Port**: 80 (automatically filled)
- **Source**: Anywhere-IPv4 (0.0.0.0/0)

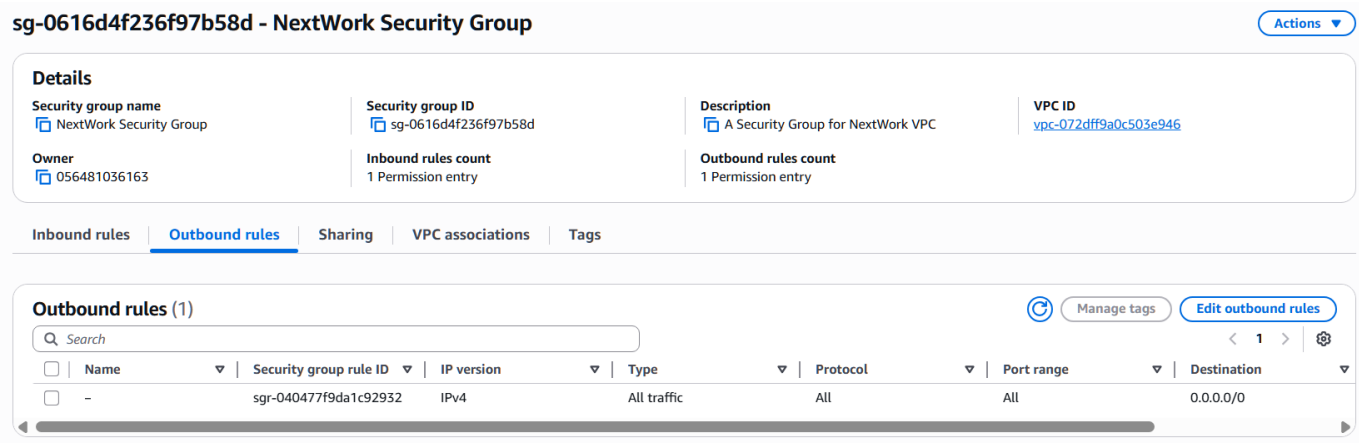
⚠ Yellow Warning Banner:



AWS warns me that allowing 0.0.0.0/0 means ANY IP address can access my resource on port 80. This is necessary for public websites but risky for sensitive resources.

Best practice: For production, restrict access to known IP addresses when possible.

Outbound Rules:



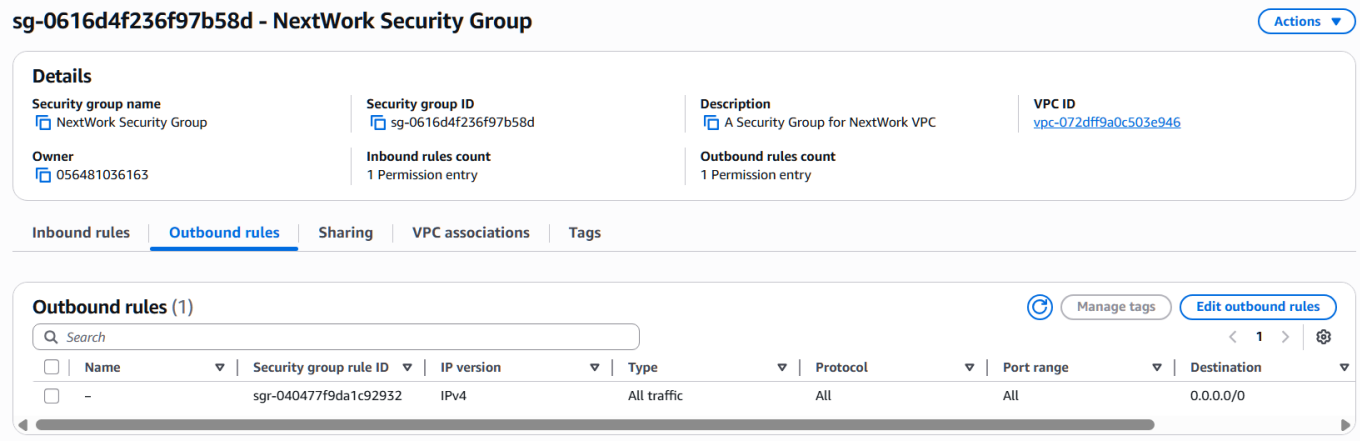
The default outbound rule allows all traffic:

- **Type**: All traffic
- **Protocol**: All
- **Port**: All
- **Destination**: Custom (0.0.0.0/0)

Why allow all outbound by default?

Resources need to communicate with various services (databases, APIs, software updates). AWS trusts that if you launched a resource, you want it to communicate outward.

⚠ Another Warning:



AWS recommends being more restrictive with outbound rules for better security.

Step 4: Create Network ACL

What I did:

I created a Network ACL (NACL) to add an additional layer of security at the subnet level.

What Are Network ACLs?

Think of **Network ACLs as traffic cops stationed at every entry and exit point of a subnet**, checking each data packet against a table of rules.

Key Characteristics:

- Work at the **subnet level** (affect all resources in the subnet)
- **Stateless**: Must explicitly allow both inbound AND outbound traffic
- Rules are numbered and evaluated in order (100, 200, 300, etc.)
- Asterisk (*) rule is the catch-all that denies everything not explicitly allowed

Understanding Data Packets

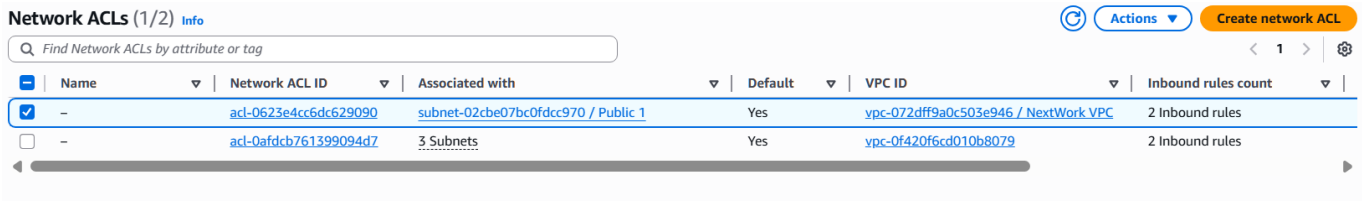
What are data packets?

When you browse a website, your request is broken into tiny pieces called **data packets**. Each packet contains:

- Part of the data being sent
- Source IP address
- Destination IP address
- Protocol information

Network ACLs inspect these packets at the subnet boundary.

Default Network ACL



I can see two existing Network ACLs:

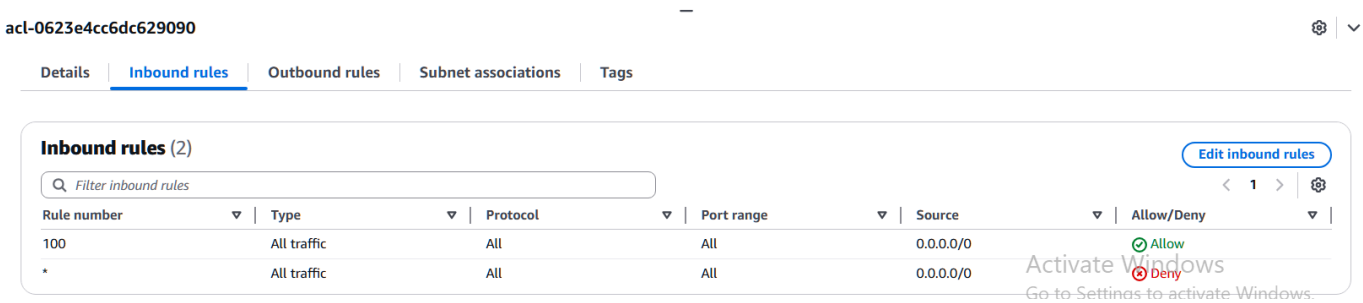
- **acl-0623e4cc6dc629090**: Default NACL for NextWork VPC (associated with Public 1)
- **acl-0afdcb761399094d7**: Default NACL for default VPC (3 subnets)

Why do default NACLs exist?

AWS creates a default NACL for every VPC that allows all inbound and outbound traffic. This way, your subnet works immediately without configuration.

Examining Default NACL Rules

Inbound Rules:



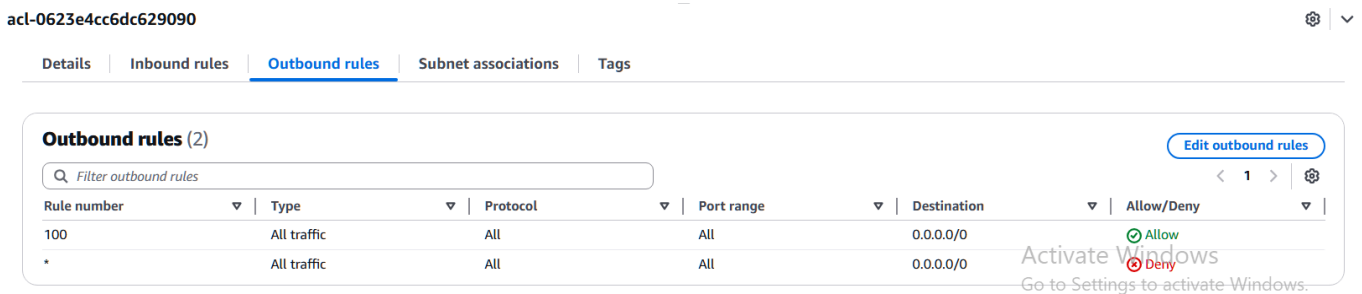
Default NACL has two rules:

- **Rule 100**: Type: All traffic, Protocol: All, Port: All, Source: 0.0.0.0/0, **Allow**
- **Asterisk (*)**: Type: All traffic, Protocol: All, Port: All, Source: 0.0.0.0/0, **Deny**

How rule evaluation works:

1. Traffic arrives at subnet
2. Check Rule 100 → Matches! → ALLOW (stop checking)
3. If Rule 100 didn't match, check * rule → DENY

Outbound Rules:



Same pattern:

- **Rule 100**: Destination: 0.0.0.0/0, **Allow**

- **Asterisk (*)**: Destination: 0.0.0.0/0, **Deny**

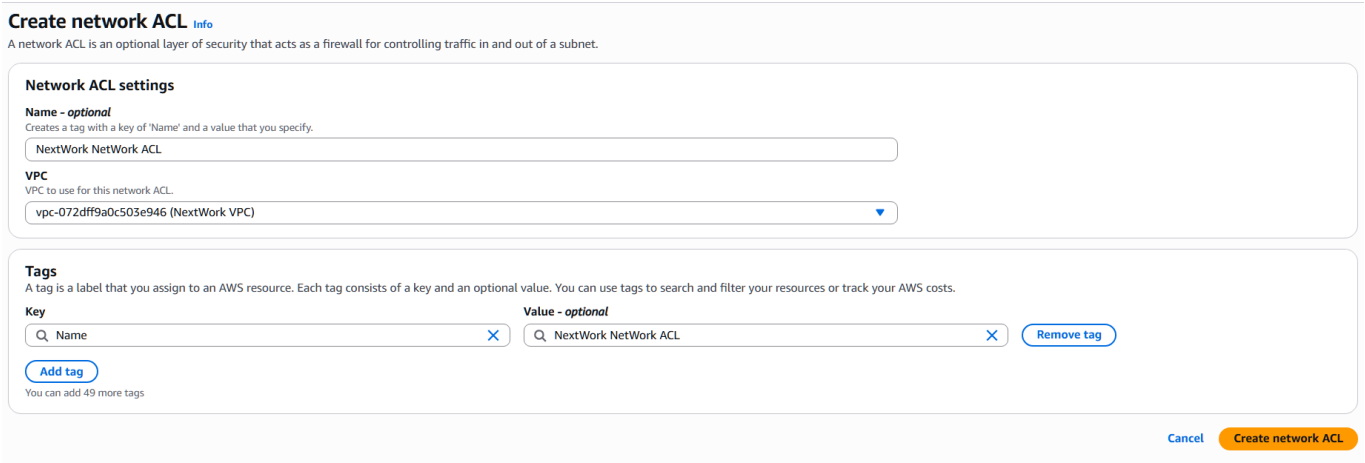
Result: The default NACL allows ALL traffic in and out.

Creating Custom Network ACL

Why create a custom NACL?

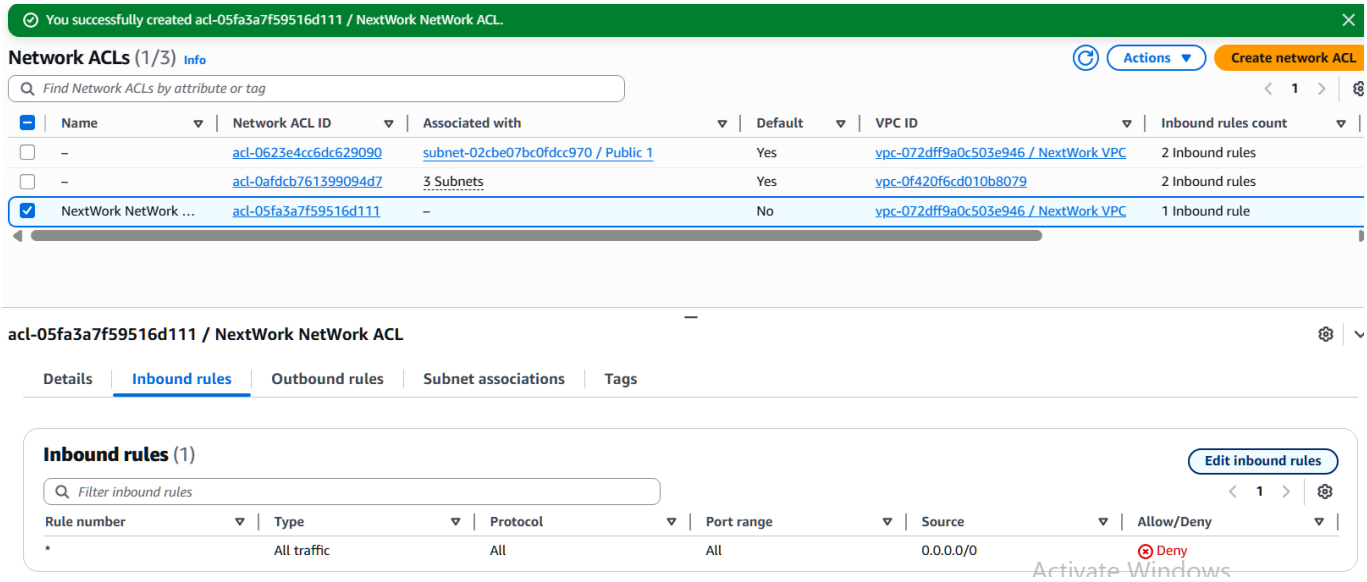
The project description suggests recreating the default NACL configuration manually to understand how it works. In production, you'd customize these rules based on security requirements.

Create Network ACL:



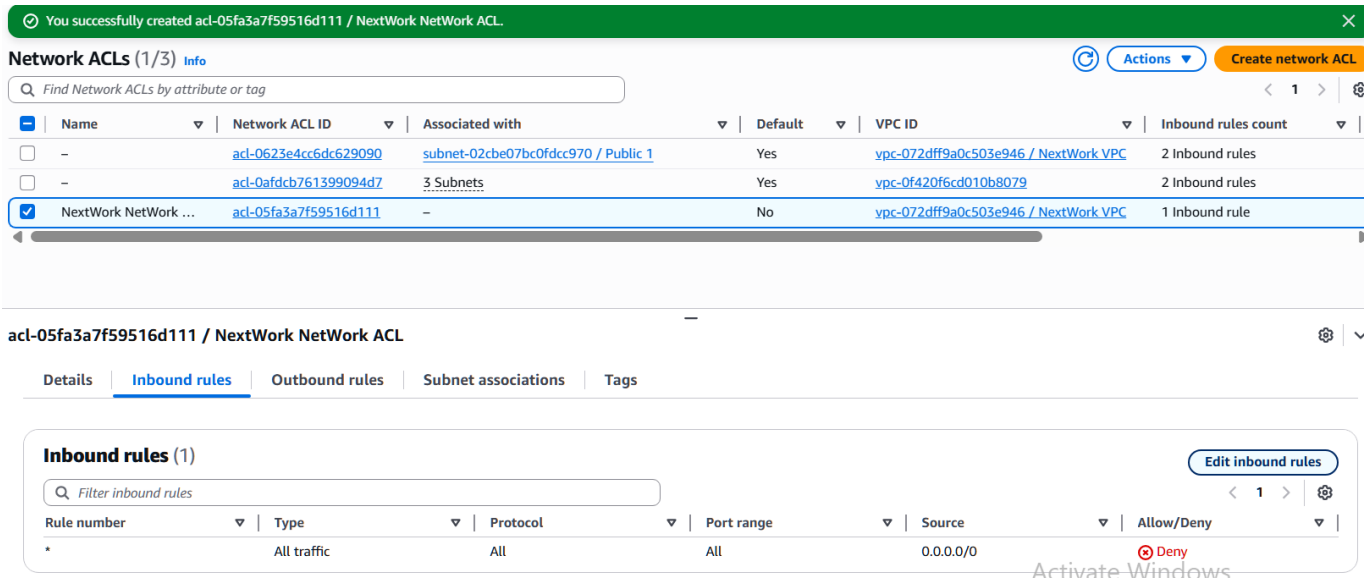
- **Name**: NextWork Network ACL
- **VPC**: vpc-072dff9a0c503e946 (NextWork VPC)
- **Tag**: Key=Name, Value=NextWork Network ACL

Success Message:



"You successfully created acl-05fa3a7f59516d111 / NextWork NetWork ACL."

Custom NACL Overview

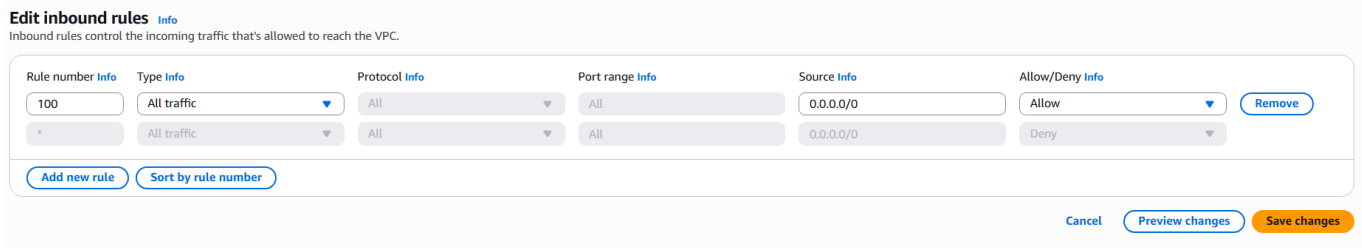


Now I have three Network ACLs:

- **acl-0623e4cc6dc629090**: Default (associated with Public 1)
- **acl-0afdcb761399094d7**: Default (3 subnets)
- **acl-05fa3a7f59516d111**: NextWork NetWork ACL (not associated yet)

Adding Inbound Rules

Edit Inbound Rules:



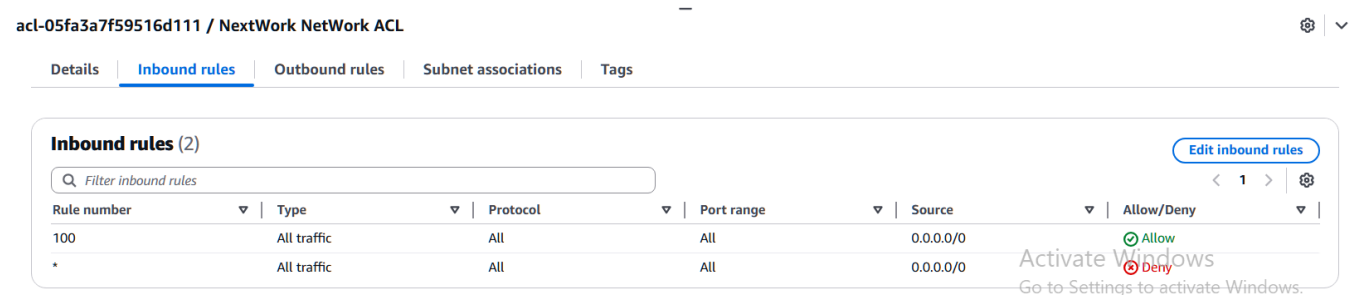
I added Rule 100 to allow all traffic:

- **Rule number**: 100
- **Type**: All traffic
- **Protocol**: All (automatically filled)
- **Port range**: All (automatically filled)
- **Source**: 0.0.0.0/0
- **Allow/Deny**: Allow

Why Rule Number 100?

Starting at 100 gives you room to add rules before it (e.g., Rule 50, Rule 75) if you need to block specific traffic later. Lower numbers are evaluated first.

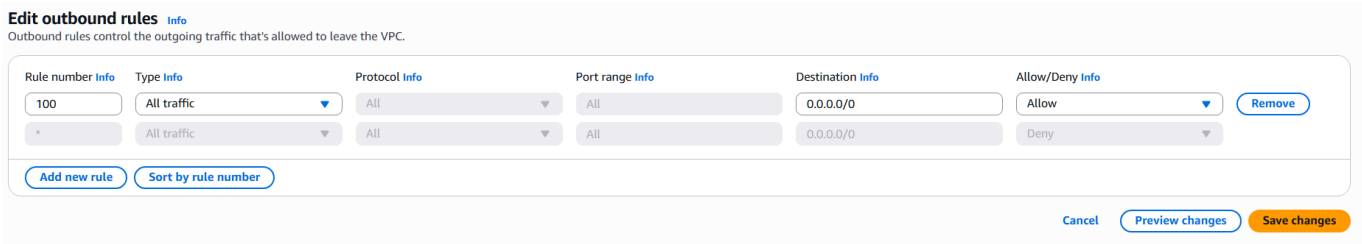
Final Inbound Rules:



- **Rule 100**: All traffic from 0.0.0.0/0 → **Allow**
- **Asterisk (*)**: All traffic from 0.0.0.0/0 → **Deny**

Adding Outbound Rules

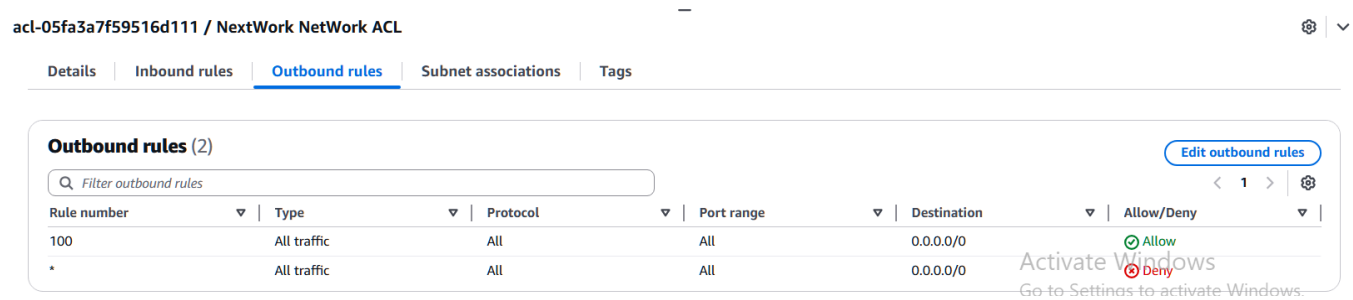
Edit Outbound Rules:



I added Rule 100 to allow all outbound traffic:

- **Rule number**: 100
- **Type**: All traffic
- **Destination**: 0.0.0.0/0
- **Allow/Deny**: Allow

Final Outbound Rules:

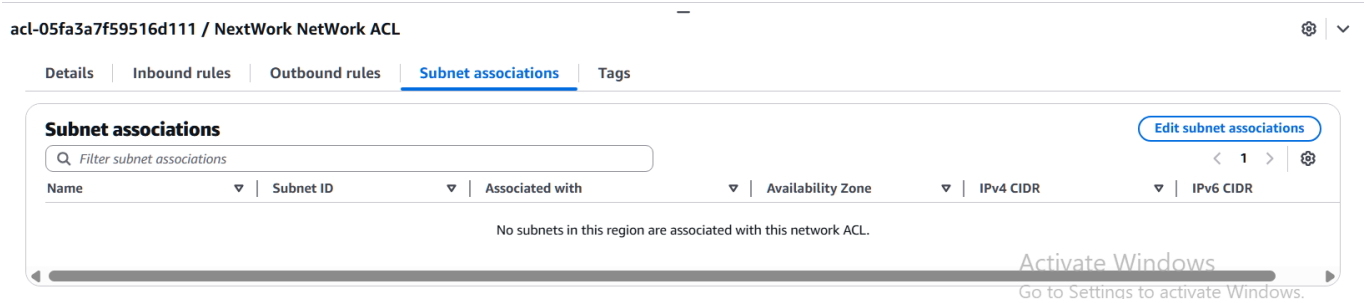


- **Rule 100**: All traffic to 0.0.0.0/0 → **Allow**
- **Asterisk (*)**: All traffic to 0.0.0.0/0 → **Deny**

Associating NACL with Subnet

Important: A Network ACL won't do anything until it's associated with a subnet!

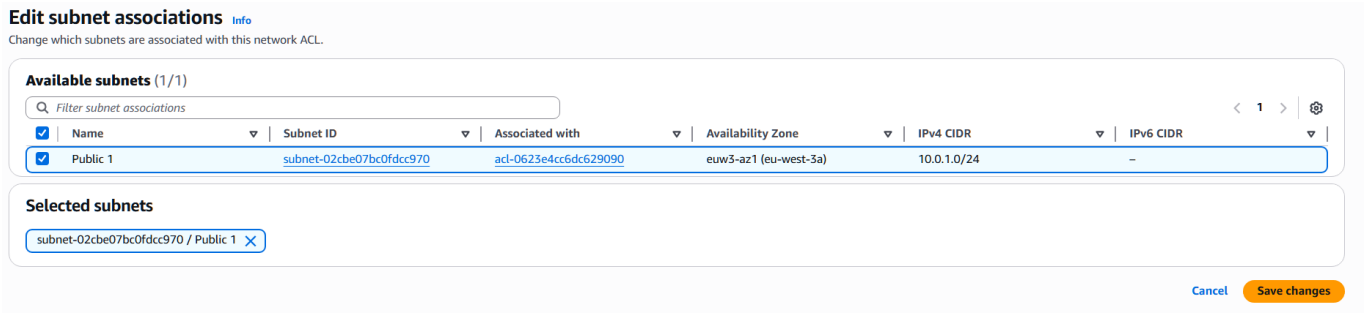
Subnet Associations (Before):



"No subnets in this region are associated with this network ACL."

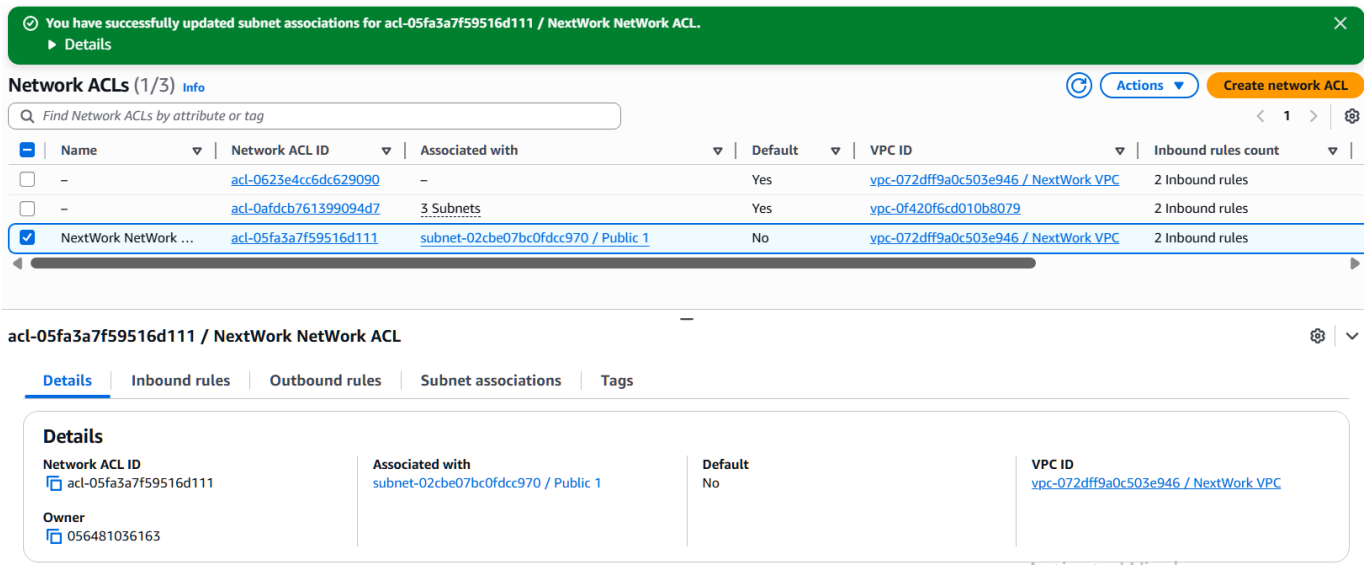
Edit Subnet Associations:

I associated my custom NACL with Public 1 subnet.



"You have successfully updated subnet associations for acl-05fa3a7f59516d111 / NextWork NetWork ACL."

Final Network ACL List:



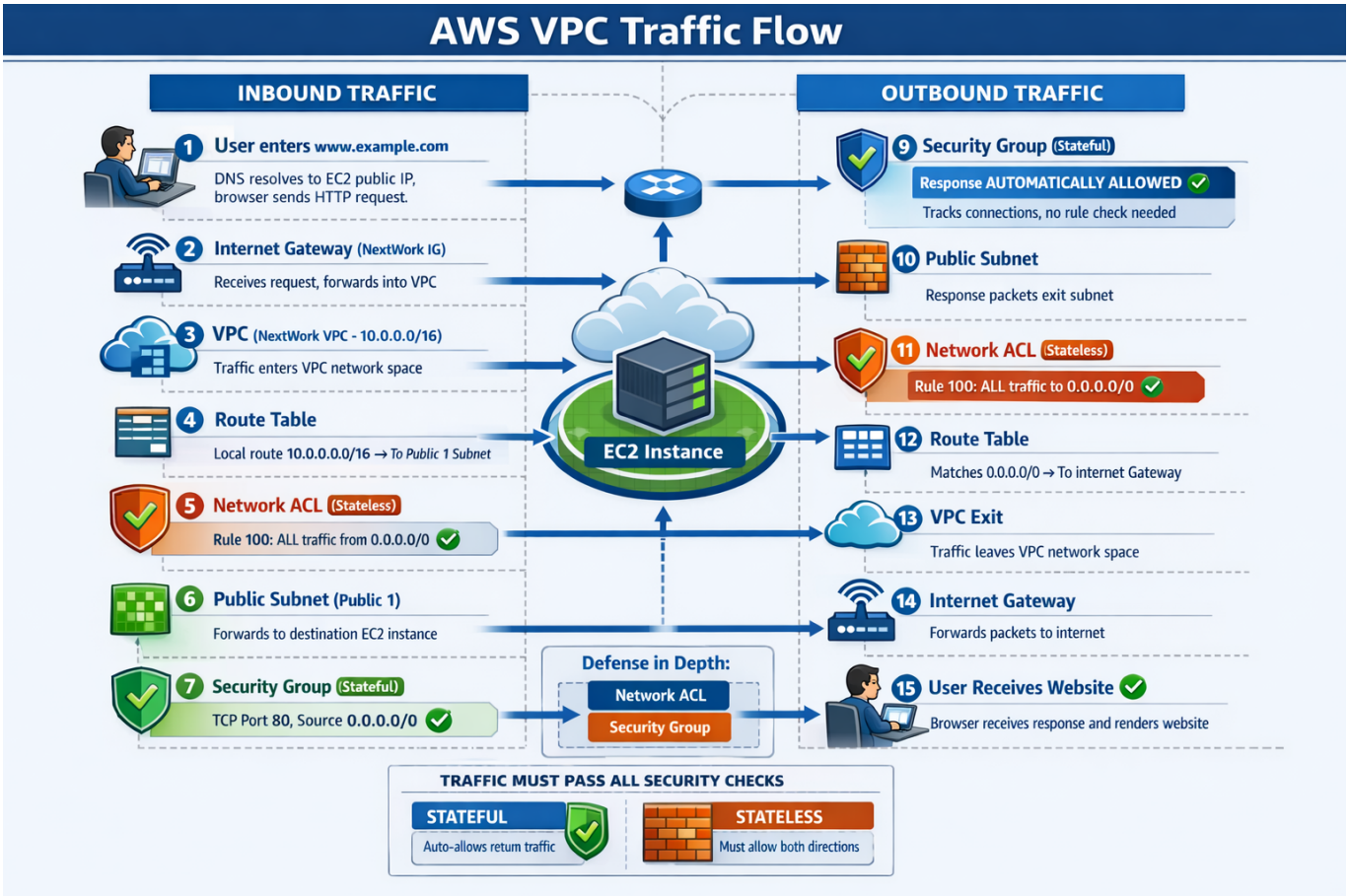
Now showing:

- **acl-0623e4cc6dc629090**: Associated with nothing (replaced by custom NACL)
- **acl-0afdcb761399094d7**: Associated with 3 Subnets
- **acl-05fa3a7f59516d111**: **Associated with subnet-02cbe07bc0fdcc970 / Public 1**

Key Point: Only ONE Network ACL can be associated with a subnet at a time. When I associated my custom NACL, it replaced the default NACL for Public 1.

Subnet now protected by custom Network ACL! 🌓

Understanding Traffic Flow



Cleanup

Why cleanup matters:

AWS charges for some resources (NAT Gateways, EC2 instances). While VPCs, subnets, and route tables are free, cleanup helps you avoid accidental charges, stay organized, and maintain good cloud hygiene.

Deletion Order

Delete in reverse order of dependencies:

1. **Delete Custom Network ACL** - Disassociate from subnet, then delete
2. **Delete Security Group** - Ensure no instances use it, then delete
3. **Delete VPC** - Auto-deletes subnets, route tables, Internet Gateway attachments, and NACLs

Steps:

1. Go to VPC Console
2. Select NextWork VPC → Actions → Delete VPC
3. Confirm with VPC ID
4. Verify deletion of subnets, route tables, Internet gateways, NACLs, and security groups

Conclusion

This project provided hands-on experience with AWS networking fundamentals by building a complete VPC infrastructure with multi-layered security.

What's Next?

This project is **Part 2** of a 9-part VPC series. The journey continues

---

Project Completed: December 2025
Author: YOUHAD AYOUB
Region: eu-west-3 (Paris)
NextWork Challenge: AWS Beginners Challenge - Project 4

This project was completed as part of the NextWork AWS Beginners Challenge. Special thanks to the NextWork community for their comprehensive project structure and excellent educational materials. The hands-on approach of configuring route tables, security groups, and Network ACLs provided invaluable experience in understanding AWS networking at a deep level.