

# 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](#)
  - [Technologies & Concepts](#)
  - [Step-by-Step Implementation](#)
    - [Step 1: Create Private Subnet](#)
    - [Step 2: Create Private Route Table](#)
    - [Step 3: Create Private Network ACL](#)
  - [Conclusion](#)
- 

## What I Built

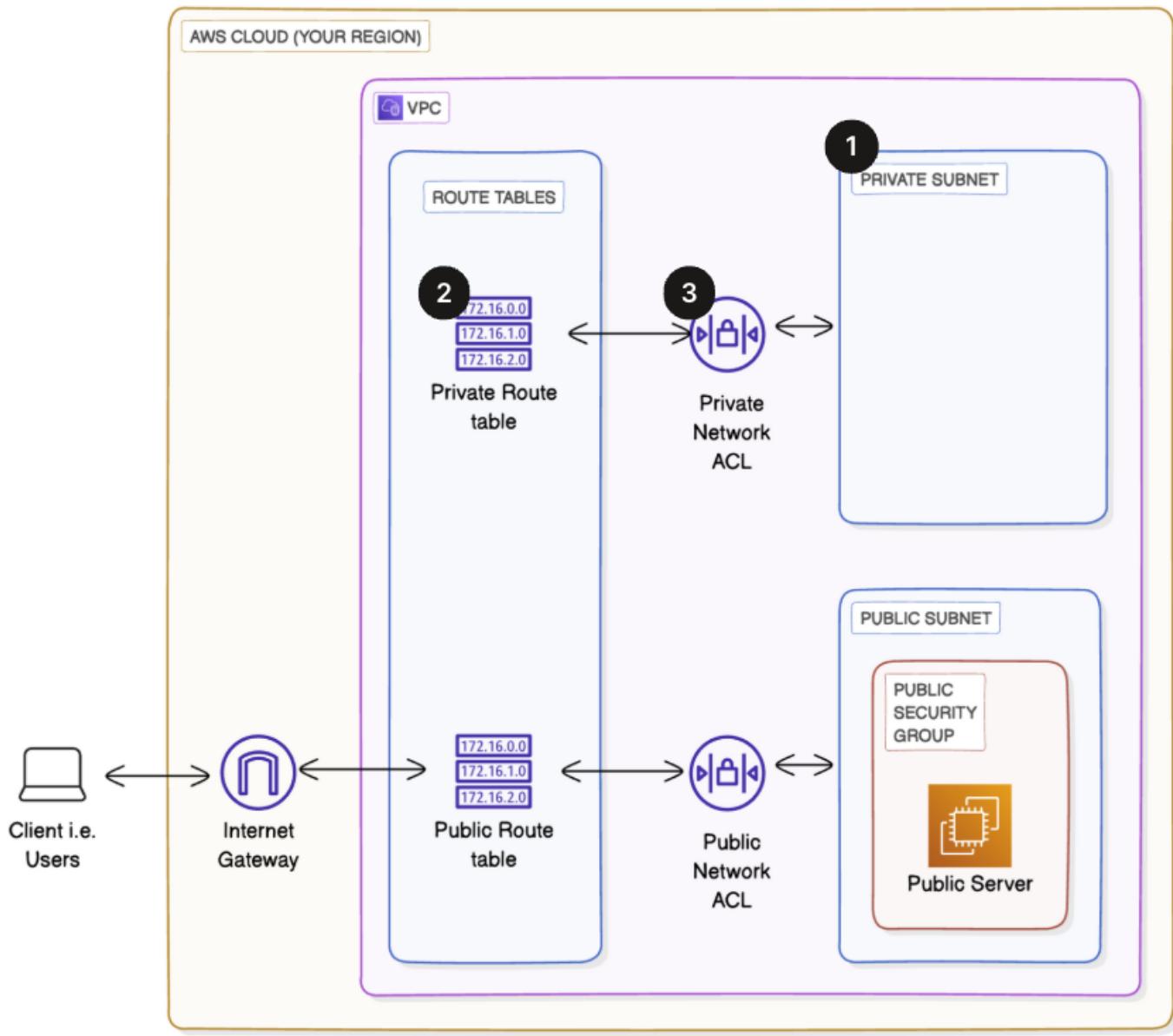
A complete private subnet infrastructure with isolated networking and security:

- **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:

[Create subnet](#) Info

**VPC**

**VPC ID**  
Create subnets in this VPC.  
vpc-0f45cc70ad0588bf9 (NextWork VPC)

**Associated VPC CIDRs**

**IPv4 CIDRs**  
10.0.0.0/16

---

**Subnet settings**  
Specify the CIDR blocks and Availability Zone for the subnet.

**Subnet 1 of 1**

**Subnet name**  
Create a tag with a key of 'Name' and a value that you specify.  
NextWork Private Subnet  
The name can be up to 256 characters long.

**Availability Zone** Info  
Choose the zone in which your subnet will reside, or let Amazon choose one for you.  
Europe (Paris) / euw3-az2 (eu-west-3b)

**IPv4 VPC CIDR block** Info  
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.  
10.0.0.0/16

**IPv4 subnet CIDR block**  
10.0.1.0/24 256 IPs  
[Add new subnet](#)

**Tags - optional**

Key	Value - optional
<input type="text"/> Name	<input type="text"/> NextWork Private Subnet

[Add new tag](#)  
You can add 49 more tags.  
[Remove](#)

[AWS Command Line Interface command](#)

[Cancel](#) [Create subnet](#)

## Key Decisions:

- 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.
- 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:

**Create route table** Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

**Route table settings**

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.

NextWork Private Route Table

**VPC**  
The VPC to use for this route table.

vpc-0f43cc70ad0588bf9 (NextWork VPC)

**Tags**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional
Q Name	Q NextWork Private Route Table X Remove

Add new tag

You can add 49 more tags.

**Create route table**

### Associating the Route Table with Private Subnet

#### Edit Subnet Associations:

**Edit subnet associations**

Change which subnets are associated with this route table.

**Available subnets (1/2)**

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
NextWork Public Subnet	subnet-0d5d815e62b0c9a45	10.0.0.0/24	-	rtb-035ab485f30bd9ffb / NextWork R...
<input checked="" type="checkbox"/> NextWork Private Subnet	subnet-0d471749d51ef6fd3	10.0.1.0/24	-	Main (rtb-035ab485f30bd9ffb / NextW...

**Selected subnets**

subnet-0d471749d51ef6fd3 / NextWork Private Subnet X
------------------------------------------------------

**Save 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:

### Create network ACL Info

A network ACL is an optional layer of security that acts as a firewall for controlling traffic in and out of a subnet.

**Network ACL settings**

Name - optional  
Creates a tag with a key of 'Name' and a value that you specify.

VPC  
VPC to use for this network ACL.

**Tags**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - <small>optional</small>
<input type="text" value="Name"/>	<input type="text" value="NextWork Private NACL"/> <span>X</span> <span>Remove tag</span>

**Add tag**  
You can add 49 more tags

Cancel Create network ACL

## Default Rules:

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

### Inbound Rules:

acl-032de9dd7095bac02 / NextWork Private NACL						
<a href="#">Details</a>		<a href="#">Inbound rules</a>	<a href="#">Outbound rules</a>	<a href="#">Subnet associations</a>	<a href="#">Tags</a>	
<b>Inbound rules (1)</b>						
Rule number	Type	Protocol	Port range	Source	Allow/Deny	
*	All traffic	All	All	0.0.0.0/0	<span>Deny</span>	

Activate Windows

### Outbound Rules:

acl-032de9dd7095bac02 / NextWork Private NACL						
<a href="#">Details</a>		<a href="#">Inbound rules</a>	<a href="#">Outbound rules</a>	<a href="#">Subnet associations</a>	<a href="#">Tags</a>	
<b>Outbound rules (1)</b>						
Rule number	Type	Protocol	Port range	Destination	Allow/Deny	
*	All traffic	All	All	0.0.0.0/0	<span>Deny</span>	

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:

**Edit subnet associations** Info  
Change which subnets are associated with this network ACL.

**Available subnets (1/2)**

Name	Subnet ID	Associated with	Availability Zone	IPv4 CIDR	IPv6 CIDR
<input checked="" type="checkbox"/> NextWork Public Subnet	<input type="text" value="subnet-0d5d815e62b0c9a45"/>	<input type="text" value="acl-0110d7ef8d85ad4cb / Next..."/>	euw3-az3 (eu-west-3c)	10.0.0.0/24	-
<input checked="" type="checkbox"/> NextWork Private Subnet	<input type="text" value="subnet-0d471749d51ef6fd3"/>	<input type="text" value="acl-032de9dd7095bac02 / Next..."/>	euw3-az2 (eu-west-3b)	10.0.1.0/24	-

**Selected subnets**

<input type="text" value="subnet-0d471749d51ef6fd3 / NextWork Private Subnet"/> <span>X</span>
------------------------------------------------------------------------------------------------

Cancel Save changes

## Final Network ACL List:

Network ACLs (4) <a href="#">Info</a>				<a href="#">Actions</a>	<a href="#">Create network ACL</a>	
<input type="checkbox"/>	Name	Network ACL ID	Associated with	Default	VPC ID	Inbound rule
<input type="checkbox"/>	NextWork Public NACL	acl-0110d7ef8d85ad4cb	subnet-0d5d815e62b0c9a45 / NextWork Public Subnet	No	vpc-0f43cc70ad0588bf9 / NextWork VPC	2 Inbound rul
<input type="checkbox"/>	NextWork Private NACL	acl-032de9dd7095bac02	subnet-0d471749d51ef6fd3 / NextWork Private Subnet	No	vpc-0f43cc70ad0588bf9 / NextWork VPC	1 Inbound rul

## 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:

- Distinguish between public and private subnets** - It's all about the route table!
- Plan CIDR blocks** to avoid overlaps and maximize VPC address space
- 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.*