Launching VPC Resources

Project Overview

This project marks a major milestone in my AWS networking journey - bringing my VPC to life by launching actual EC2 instances! After building the networking foundation in previous projects, I successfully deployed a **public EC2 instance** accessible from the internet and a **private EC2 instance** isolated for backend operations. I also discovered the VPC Wizard, a powerful tool that can create complete VPC architectures in minutes with automatic resource naming and configuration.

Project Duration: Approximately 60 minutes
Difficulty Level: Mildly Spicy
AWS Region Used: eu-west-3 (Paris)
Project Series: Part 4 of NextWork VPC Challenge

Table of Contents

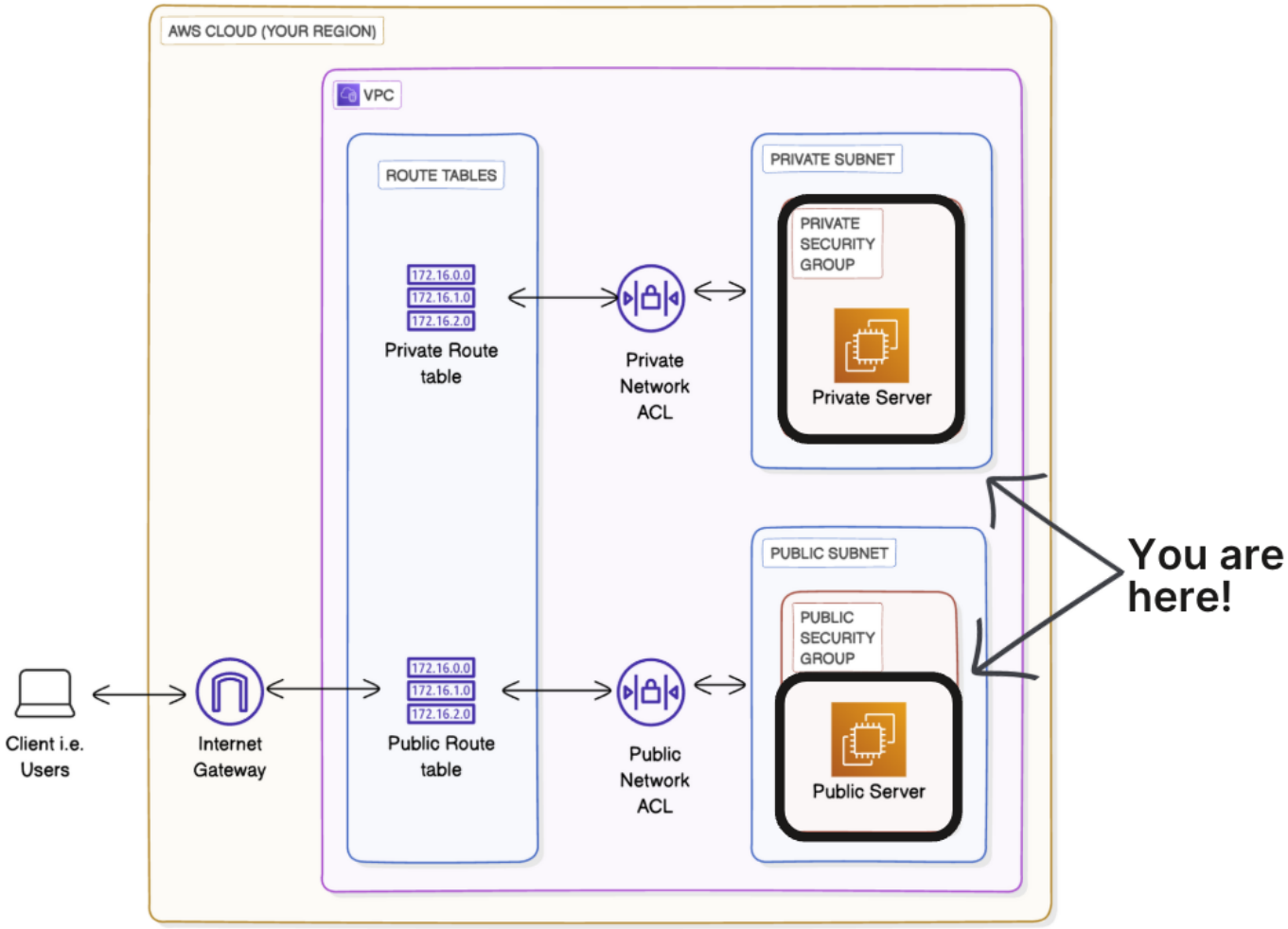
What I Built

A complete VPC with running EC2 instances and explored the VPC Wizard:

Manual VPC Infrastructure (NextWork VPC):

- **Public EC2 Instance**: NextWork Public Server (t2.micro) in eu-west-3a
  - Public IPv4: 51.44.82.224
  - Private IPv4: 10.0.0.172
  - Amazon Linux 2023 AMI
  - Accessible from the internet via HTTP
- **Private EC2 Instance**: NextWork Private Server (t2.micro) in eu-west-3b
  - No Public IP (secure!)
  - Private IPv4: 10.0.1.237
  - Amazon Linux 2023 AMI
  - Only accessible from Public Server via SSH

- **Key Pair**: NextWork Key Pair (RSA, .pem format)
    - Used for secure SSH access to both instances
- **Security Groups**:
    - NextWork Security Group (Public) - Allows HTTP from anywhere
    - NextWork Private Security Group - Allows SSH only from Public Security Group



VPC Wizard-Created Infrastructure (nextwork-vpc):

- Complete VPC with auto-generated resources
- 2 Subnets (1 public, 1 private) across 1 Availability Zone
- 3 Route Tables (1 public, 1 private, 1 main)
- Internet Gateway
- Automatic resource naming with "nextwork" prefix

Key Achievement: I now understand the difference between manual VPC creation (full control, more time) vs VPC Wizard (quick setup, best practices baked in).

Technologies & Concepts

AWS Services Used

- **Amazon EC2** - Elastic Compute Cloud for virtual servers

- **Amazon VPC** - Virtual Private Cloud for isolated networking
- **Key Pairs** - RSA cryptographic keys for secure instance access
- **Security Groups** - Resource-level firewalls
- **AMI** - Amazon Machine Image (operating system template)

Key Concepts Learned

1. **EC2 Instances**

   Virtual servers running in the cloud. Think of them as computers you can access remotely over the internet.

   Key Components:

   - **AMI (Amazon Machine Image)**: The operating system and pre-installed software (like buying a computer with Windows or macOS)
   - **Instance Type**: The hardware specifications (CPU, RAM, storage) - like choosing between a laptop vs desktop
   - **Key Pair**: Your secure key to access the server via SSH

2. **Key Pairs and SSH**

   Key Pairs consist of two cryptographic keys:

   - **Public Key**: Installed on the EC2 instance
   - **Private Key**: Stored on your computer (.pem file)

   SSH (Secure Shell) is the protocol used to connect:

   - Encrypts all communication between you and the server
   - Prevents unauthorized access
   - Standard method for managing remote servers

   Real-world analogy: Think of it like a hotel room - the public key is the lock on the door, and your private key is the key card only you have.

3. **Public vs Private Instances**

   Public EC2 Instance:

   - Has a public IPv4 address (accessible from internet)
   - Lives in a public subnet with route to Internet Gateway
   - Use case: Web servers, load balancers, bastion hosts
   - Example: 51.44.82.224 can be reached from anywhere

   Private EC2 Instance:

   - Has NO public IP address (only private IP)
   - Lives in a private subnet with no internet route
   - Use case: Databases, application servers, sensitive workloads
   - Example: 10.0.1.237 can only be reached from within the VPC

4. **Security Group Source Types**

   For Public Instance:

   - Source: 0.0.0.0/0 (Anywhere) - Anyone can access
   - Acceptable for HTTP traffic (port 80) - public websites need this!

   For Private Instance:

   - Source: Security Group ID (e.g., sg-01d9df52af4817bbb)
   - Only resources with that security group can connect
   - Provides defense in depth - even if someone breaches the public server, they can't directly access the private server from the internet

5. **VPC Wizard Benefits**

   Manual Creation (Parts 1-3):

   - Full control over every component
   - Learn deeply by doing each step
   - Time-consuming (multiple projects)
   - Must manually name each resource

   VPC Wizard:

   - Creates complete VPC in minutes
   - Follows AWS best practices automatically
   - Auto-generates consistent resource names
   - Perfect for quick testing environments
   - Configures route tables and associations automatically

6. **Instance State and Public IPs**

   Important behavior:

   - When you STOP an instance, it loses its public IP
   - When you START it again, it gets a NEW public IP
   - Private IPs remain the same
   - Solution: Use Elastic IPs for persistent public addresses (costs apply when not attached to running instance)

Step-by-Step Implementation

Step 1: Create Key Pair

What I did:

Before launching any EC2 instances, I needed to create a key pair for secure SSH access.

Understanding Key Pairs

Why do we need key pairs?

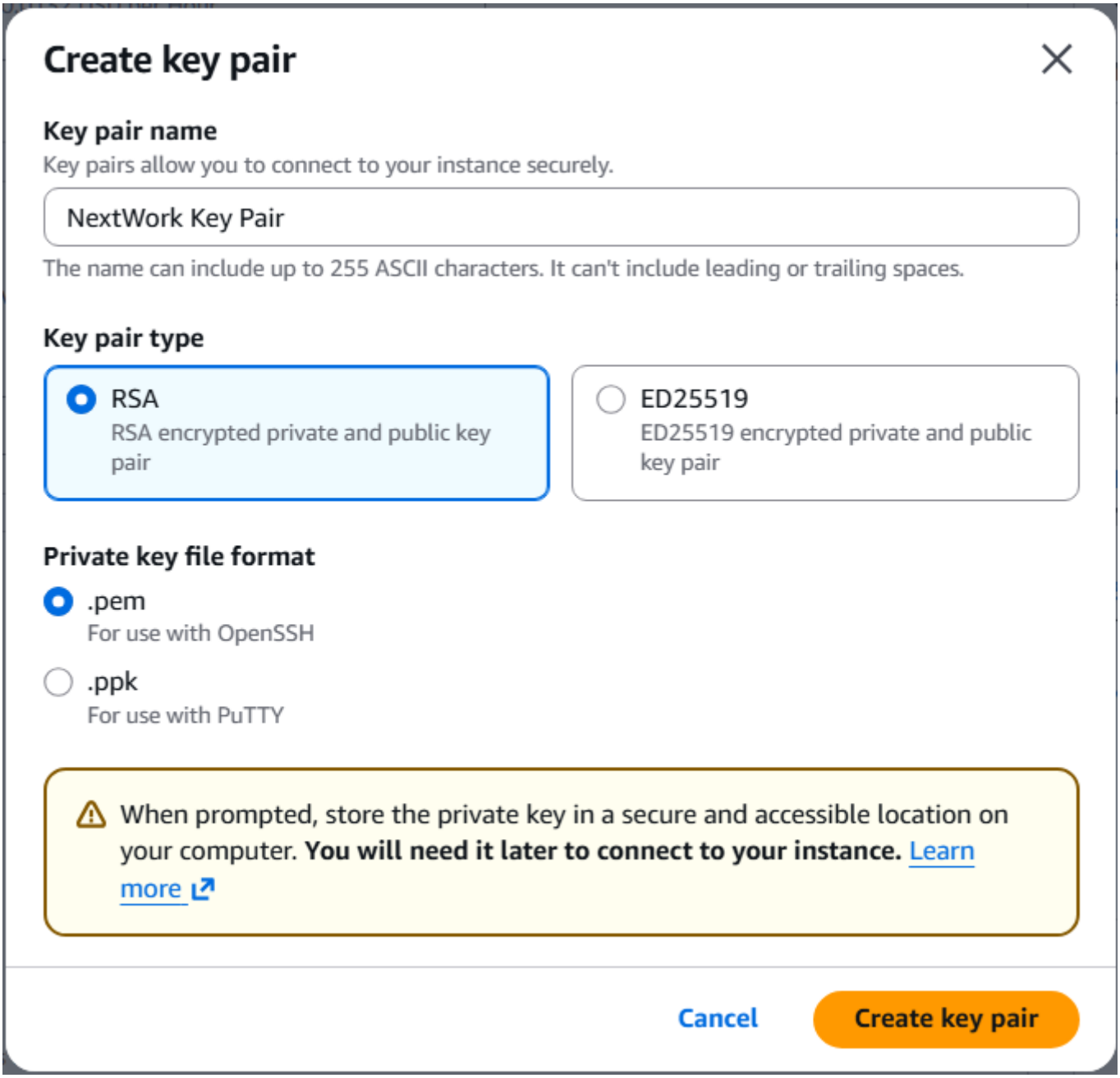
Imagine if anyone could access your server just by knowing its IP address - that would be a security nightmare! Key pairs ensure that only authorized people (with the private key) can access your instances.

How they work:

1. AWS installs the **public key** on your EC2 instance
2. You download and keep the **private key** (.pem file) secure on your computer
3. When connecting via SSH, your private key proves you're authorized
4. The connection is encrypted - no one can intercept your session

Creating the Key Pair

Key Pair Configuration:



⚠ **Critical Warning:**

"When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.**"

Why .pem format?

- .pem (Privacy Enhanced Mail) works with OpenSSH on Mac/Linux
- .ppk format is for PuTTY on Windows
- I chose .pem for maximum compatibility

Security Best Practice:

After downloading, I stored the key in a secure location and set proper permissions:

```
chmod 400 NextWork-Key-Pair.pem
```

This ensures only I can read the file, preventing unauthorized access.

Step 2: Launch Public EC2 Instance

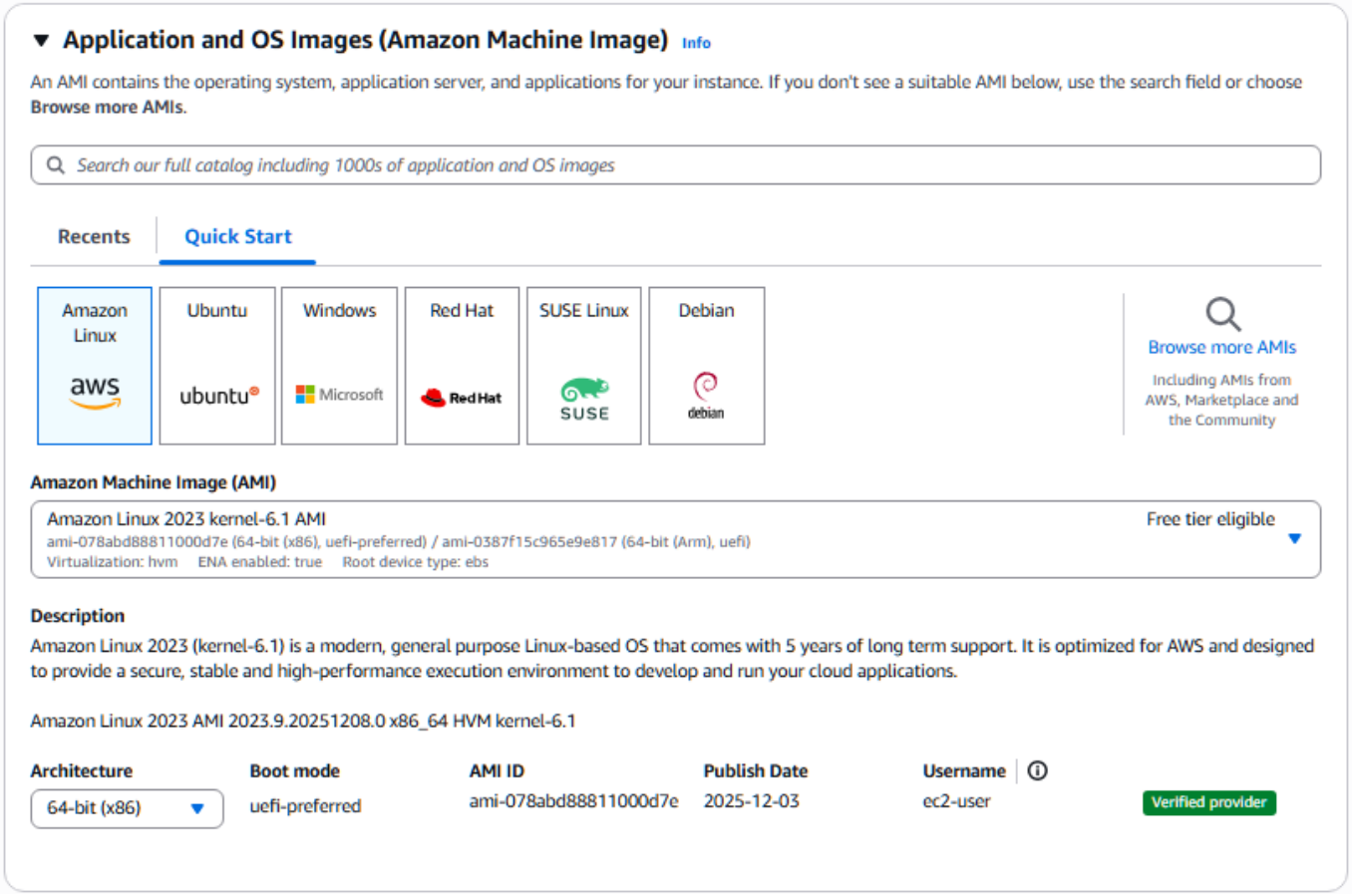
What I did:

I launched my first EC2 instance in the public subnet, making it accessible from the internet.

Instance Configuration

Name and AMI Selection:

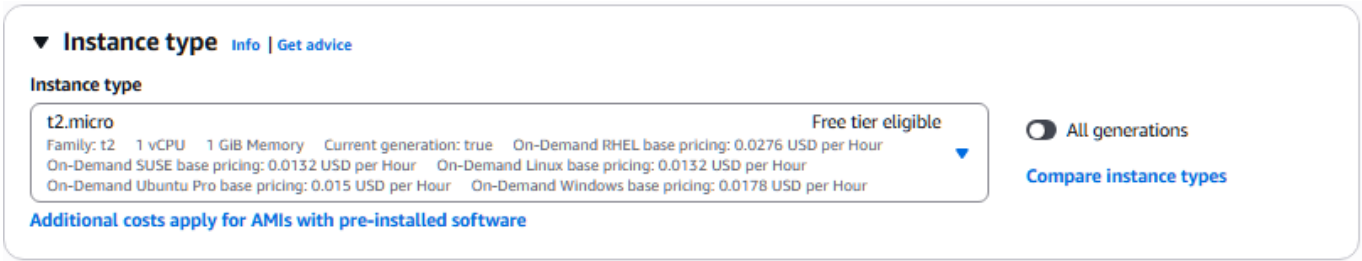
AMI Selection:



Why Amazon Linux 2023?

- Designed specifically for AWS (best performance and integration)
- Free tier eligible (no charges!)
- 5 years of long-term support
- Comes with AWS CLI and tools pre-installed
- Regular security updates

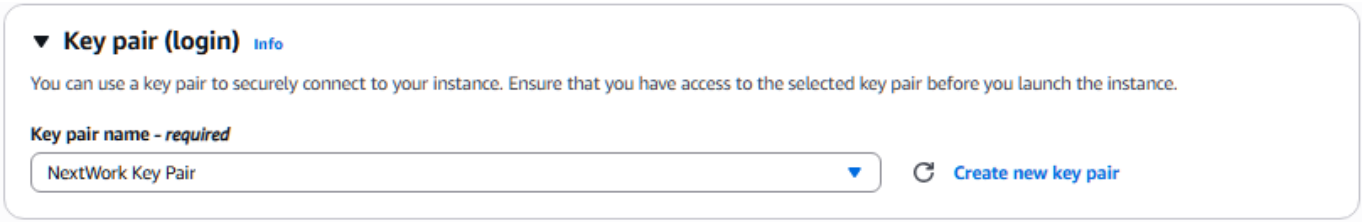
Instance Type



What does "burstable" mean?

t2.micro instances can "burst" above baseline CPU performance when needed, using CPU credits. Perfect for workloads with occasional spikes (like web servers with variable traffic).

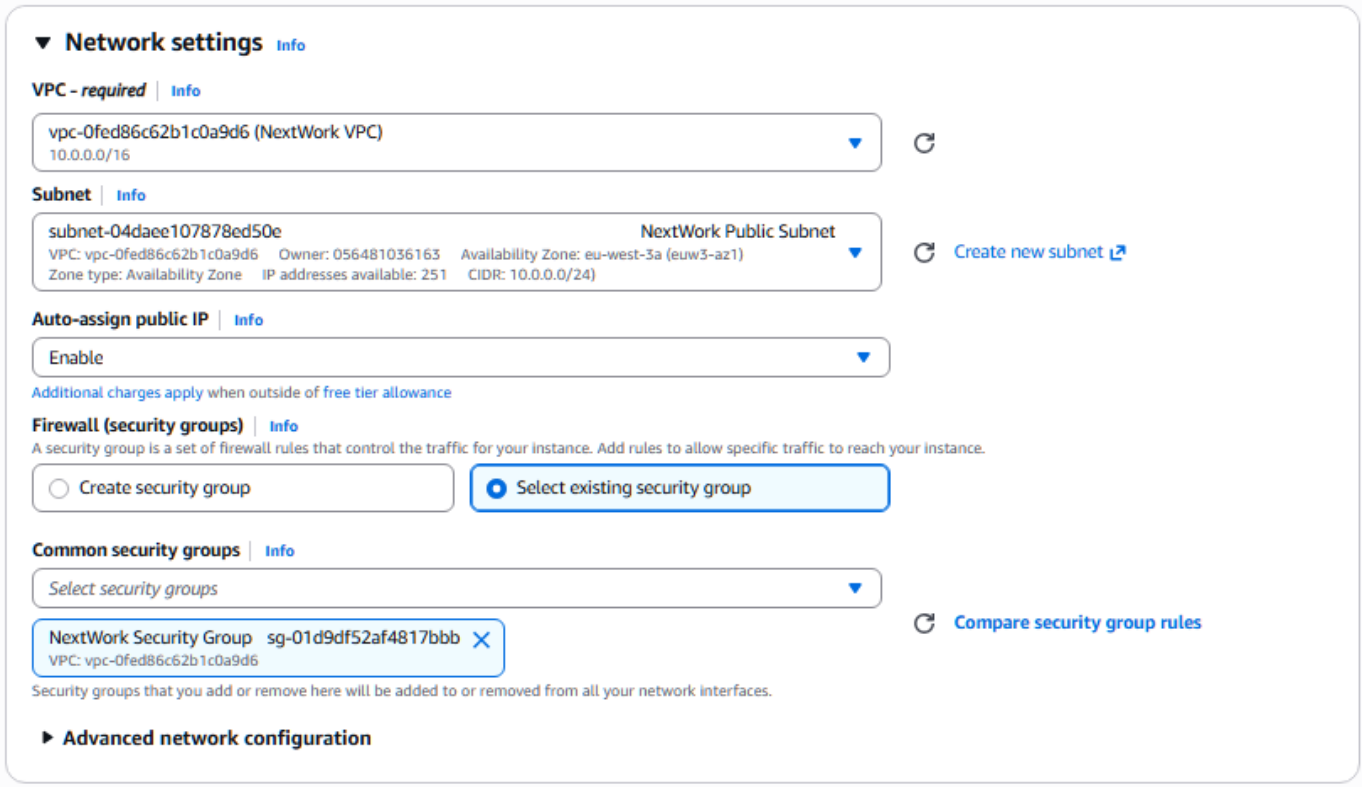
Key Pair Selection



- **Key pair name**: NextWork Key Pair (the one I just created!)

Important: Without selecting a key pair, I wouldn't be able to SSH into my instance later.

Network Settings



Why enable auto-assign public IP?

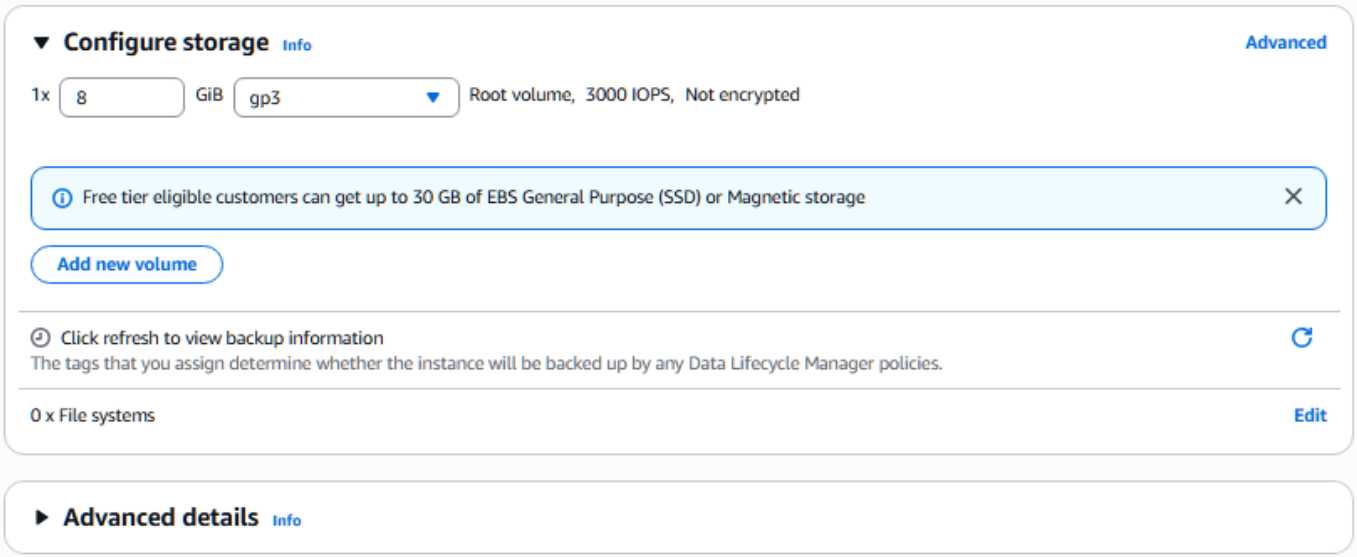
Without a public IP, my instance would be unreachable from the internet, even though it's in a public subnet!

Firewall (Security Groups):

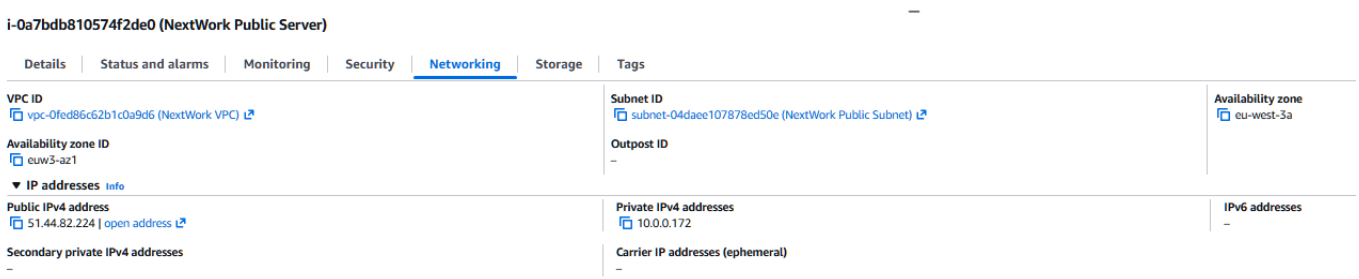
- **Option selected**: Select existing security group
- **Security group**: NextWork Security Group (sg-01d9df52af4817bbb)

This security group already has HTTP (port 80) configured from my previous project, allowing web traffic from anywhere (0.0.0.0/0).

Storage Configuration



Networking Details



Success! My public server is now running and accessible from the internet! 🎉

Step 3: Launch Private EC2 Instance

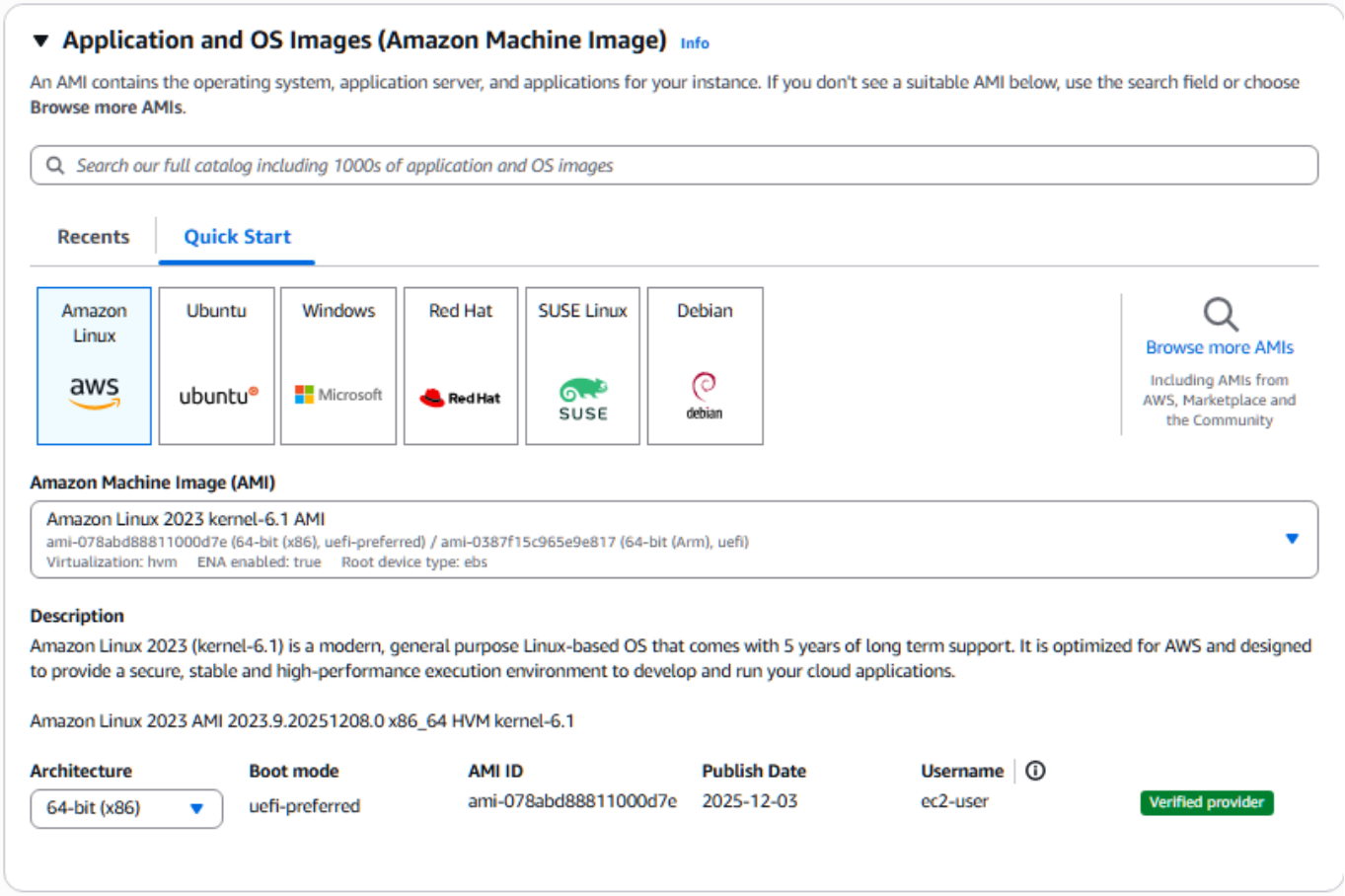
What I did:

I launched a second EC2 instance in the private subnet with restricted SSH access - only allowing connections from the public security group.

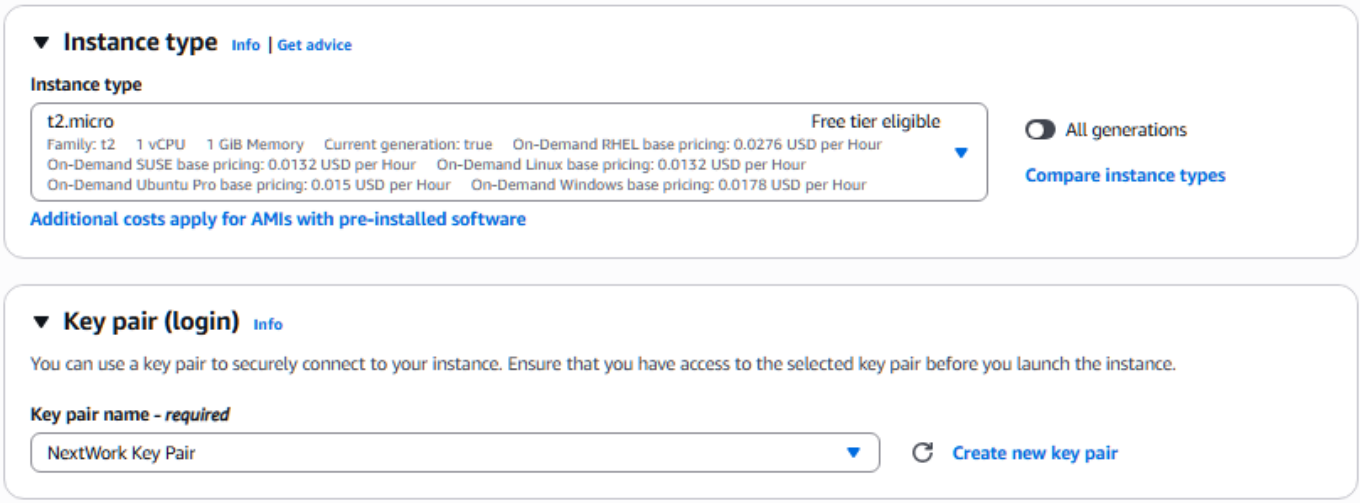
Instance Configuration

Name and Basic Settings:

- **Name**: NextWork Private Server



- **AMI**: Amazon Linux 2023 AMI (same as public instance)



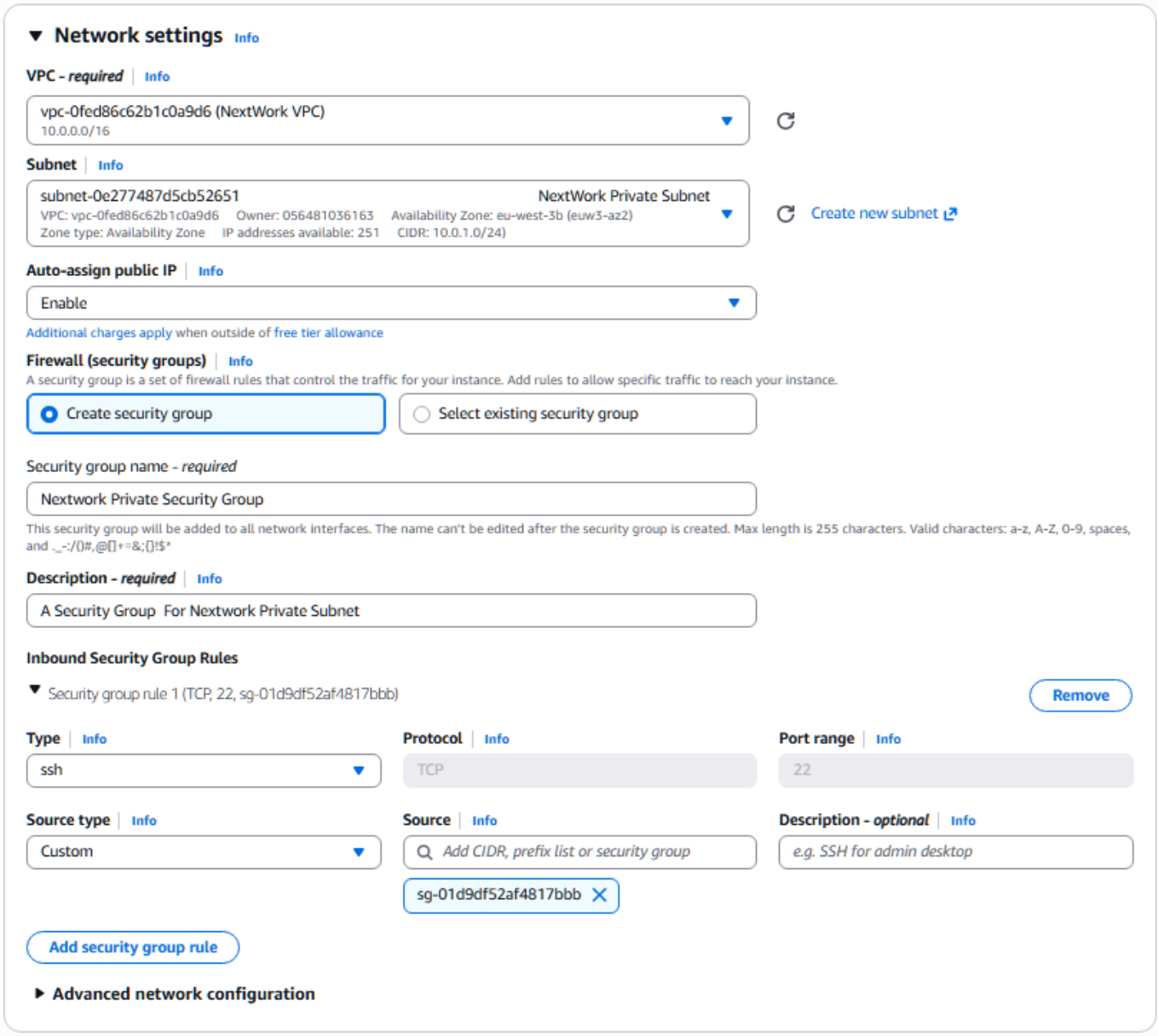
Instance Type and Key Pair:

- **Instance type**: t2.micro
- **Key pair**: NextWork Key Pair (reusing the same key pair!)

Can I use the same key pair for multiple instances?

Yes! This makes management easier - one key to access both instances. However, security consideration: anyone with this key can access ALL instances using it.

Network Settings & Creating the Private Security Group - The Critical Part!



What does this mean?

Instead of allowing SSH from "Anywhere" (0.0.0.0/0), I'm restricting SSH access to ONLY resources that have the **NextWork Public Security Group** attached.

⚠ Yellow Warning Explained:

AWS initially warns: "Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance."

By changing the source to a security group ID, I eliminated this warning and implemented proper security!

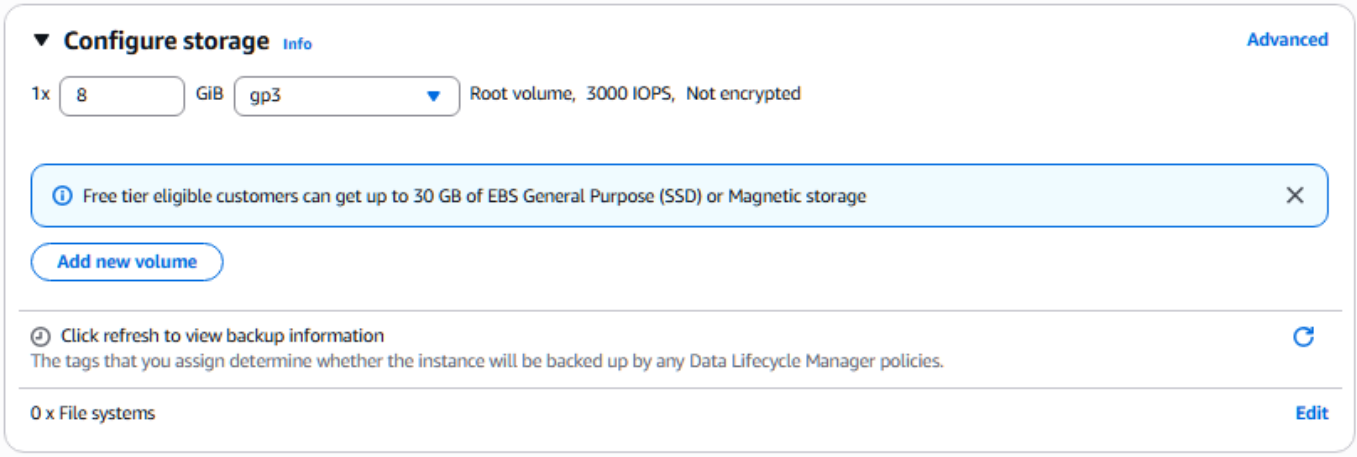
Security Architecture:


The Internet
[Anyone can access]
Public Server (sg-01d9df52af4817bbb)
[SSH allowed because it has the right security group]
Private Server
(only accepts SSH from sg-01d9df52af4817bbb)
Database / Backend Services

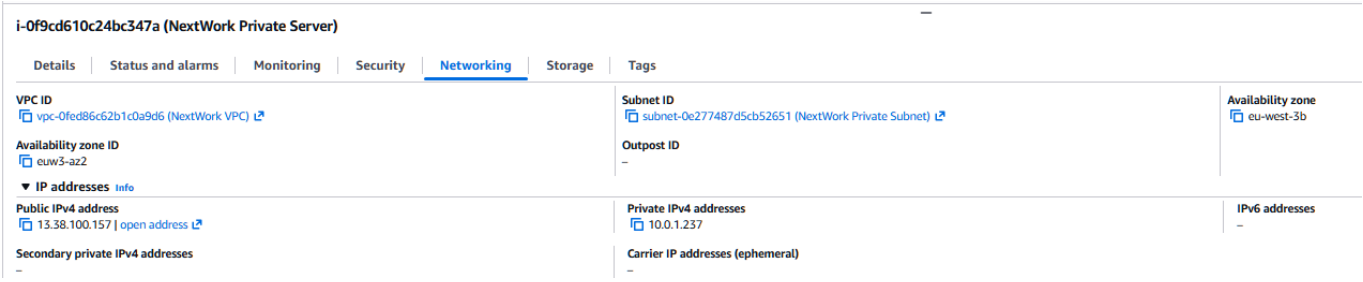
Defense in Depth:

1. Private subnet has no internet route (architectural)
2. Network ACL controls subnet-level traffic
3. Security group allows SSH only from public security group
4. Even if public server is compromised, attacker can't SSH from their own computer directly to private server

Storage Configuration



Private Instance Networking Details



⚠ **Important Discovery:**

Even though I launched in a private subnet, the instance got a public IP because I left "Auto-assign public IP" enabled!

However, because the private subnet's route table has NO route to the Internet Gateway, this public IP is effectively useless - the instance still can't communicate with the internet.

Best practice: For private instances, disable auto-assign public IP to avoid confusion.

Success! My private server is running and secured with proper SSH restrictions! 🌓

Step 4: Use VPC Wizard

What I did:

After manually creating VPCs in previous projects (which took hours!), I discovered the VPC Wizard - AWS's automated tool that creates complete VPC architectures in minutes.

Why Use the VPC Wizard?

Manual VPC Creation (Parts 1-3):

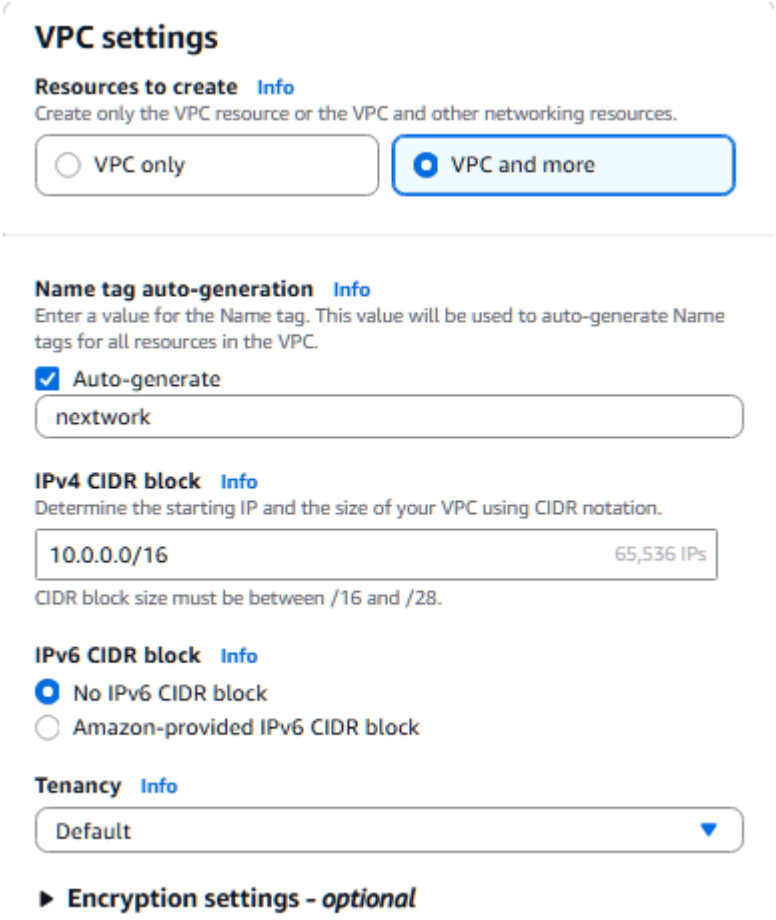
- ☑ Learn every component deeply
- ☑ Full control over configuration
- ❌ Time-consuming (multiple steps across projects)
- ❌ Easy to make mistakes
- ❌ Must manually name and tag everything

VPC Wizard:

- ☑ Creates VPC in minutes
- ☑ Follows AWS best practices
- ☑ Auto-generates consistent resource names
- ☑ Perfect for testing/development environments
- ❌ Less control over individual components

Wizard Configuration

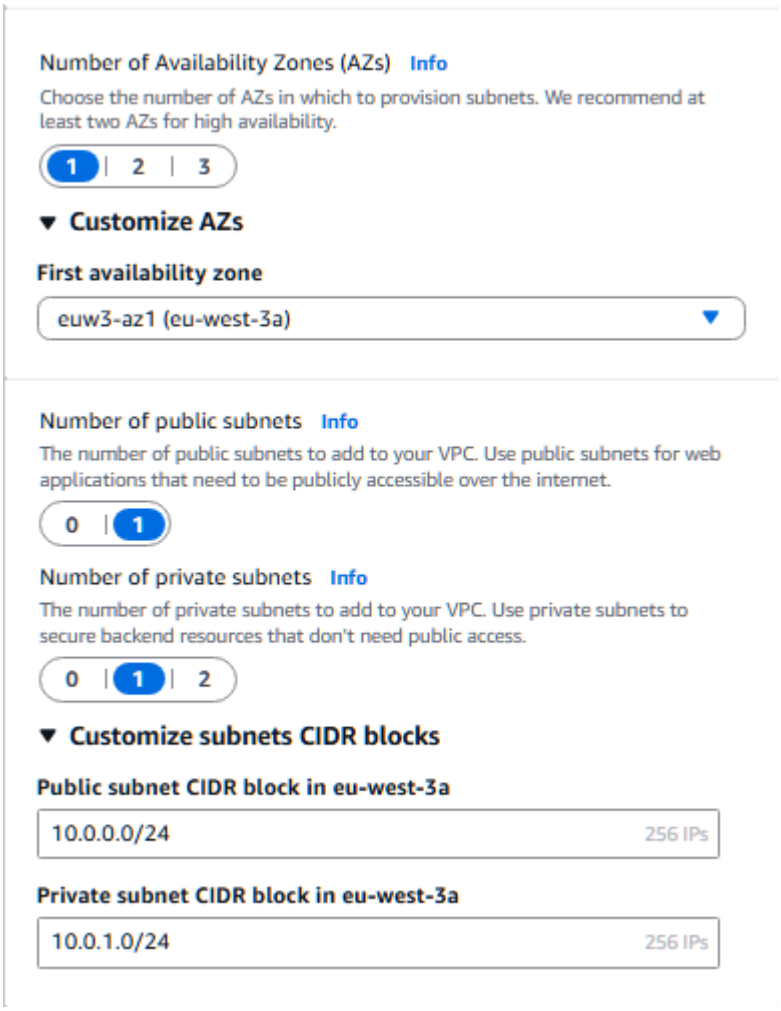
Main VPC Settings:



Why auto-generate names?

AWS will automatically prefix all resources with "nextwork-" (e.g., nextwork-vpc, nextwork-subnet-public1, nextwork-rtb-public), making it easy to identify which resources belong together!

Availability Zones and Subnets

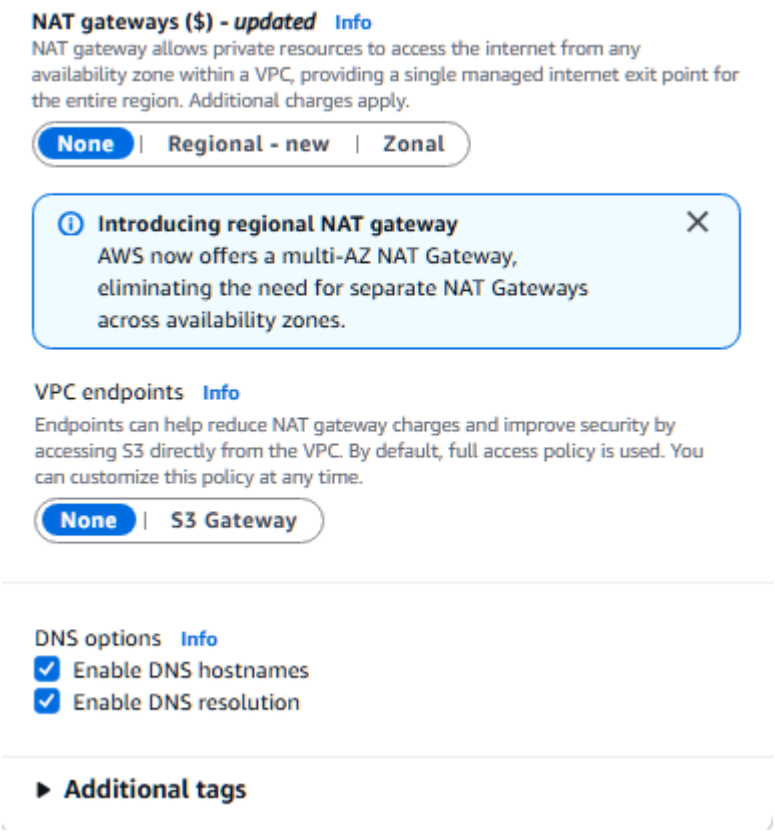


AWS enforces best practices! When you select 2 AZs, AWS requires at least 1 public subnet per AZ for high availability. You can't create just 1 public subnet across 2 AZs.

Why /24 instead of /20?

Default wizard uses /20 (4,096 IPs per subnet), but I changed to /24 (256 IPs) to match my manual VPC configuration and conserve IP space.

NAT Gateways and VPC Endpoints



What are NAT Gateways?

NAT (Network Address Translation) Gateways allow private subnet resources to access the internet for updates/patches while blocking inbound traffic from the internet.

Cost consideration: NAT Gateways cost ~$0.045/hour + data transfer charges. For learning, I skipped this.

What are VPC Endpoints?

VPC Endpoints allow private connectivity to AWS services (like S3) without going through the internet. The wizard offers S3 Gateway endpoint by default.

Why skip for now?

We'll explore VPC endpoints in detail in a later project of this series!

DNS Options

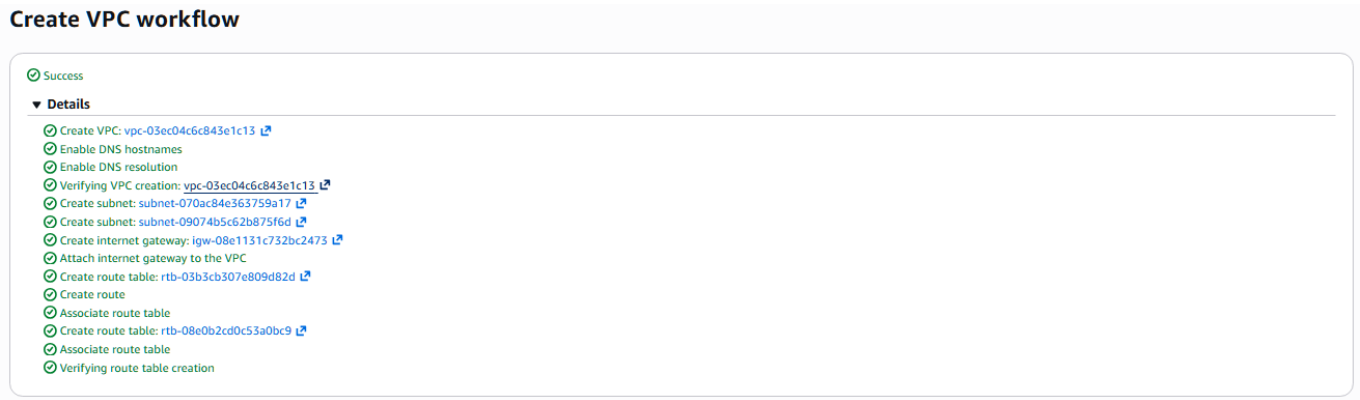
What do these do?

- **DNS hostnames**: Instances get human-readable names like `ec2-13-38-100-157.eu-west-3.compute.amazonaws.com`
- **DNS resolution**: AWS translates these hostnames to IP addresses automatically

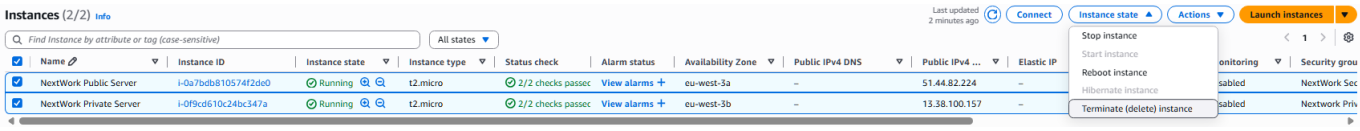
Resource Map Preview

(Single AZ):

Wizard Execution



Both VPCs Running



I now have EC2 instances running in both VPCs:

- **NextWork Public Server** (i-0a7bdb810574f2de0) - Running in NextWork VPC, eu-west-3a
- **NextWork Private Server** (i-0f9cd610c24bc347a) - Running in NextWork VPC, eu-west-3b

Both instances are healthy with 2/2 status checks passed! ☑

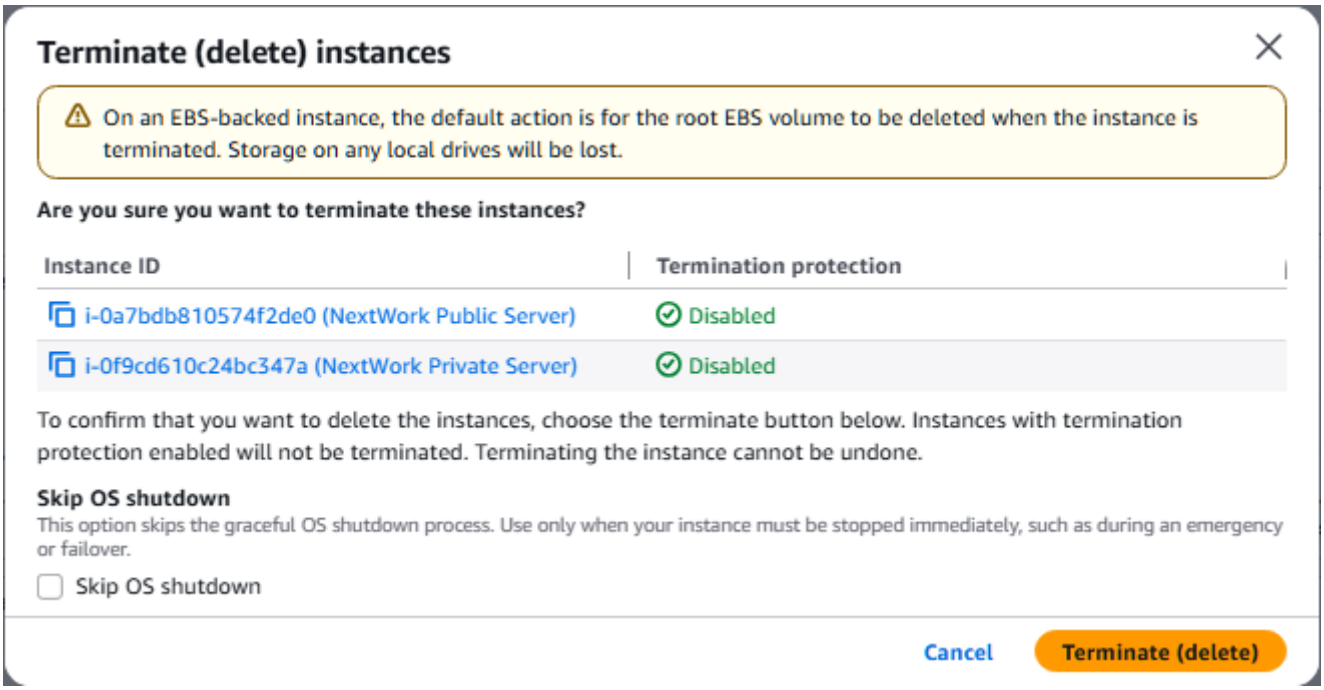
Cleanup

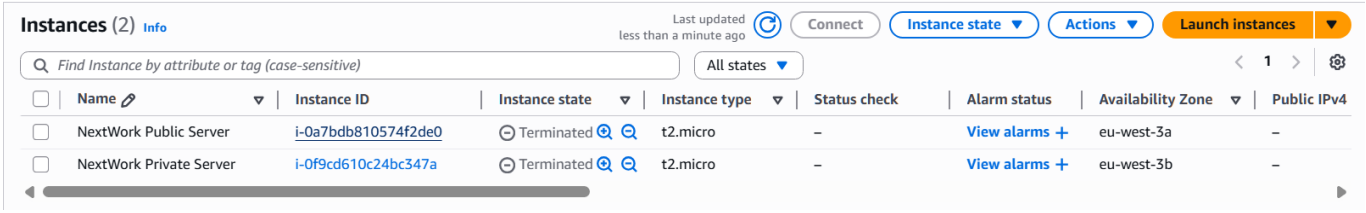
Deletion Order

Important: Delete resources in the correct order to avoid dependency issues!

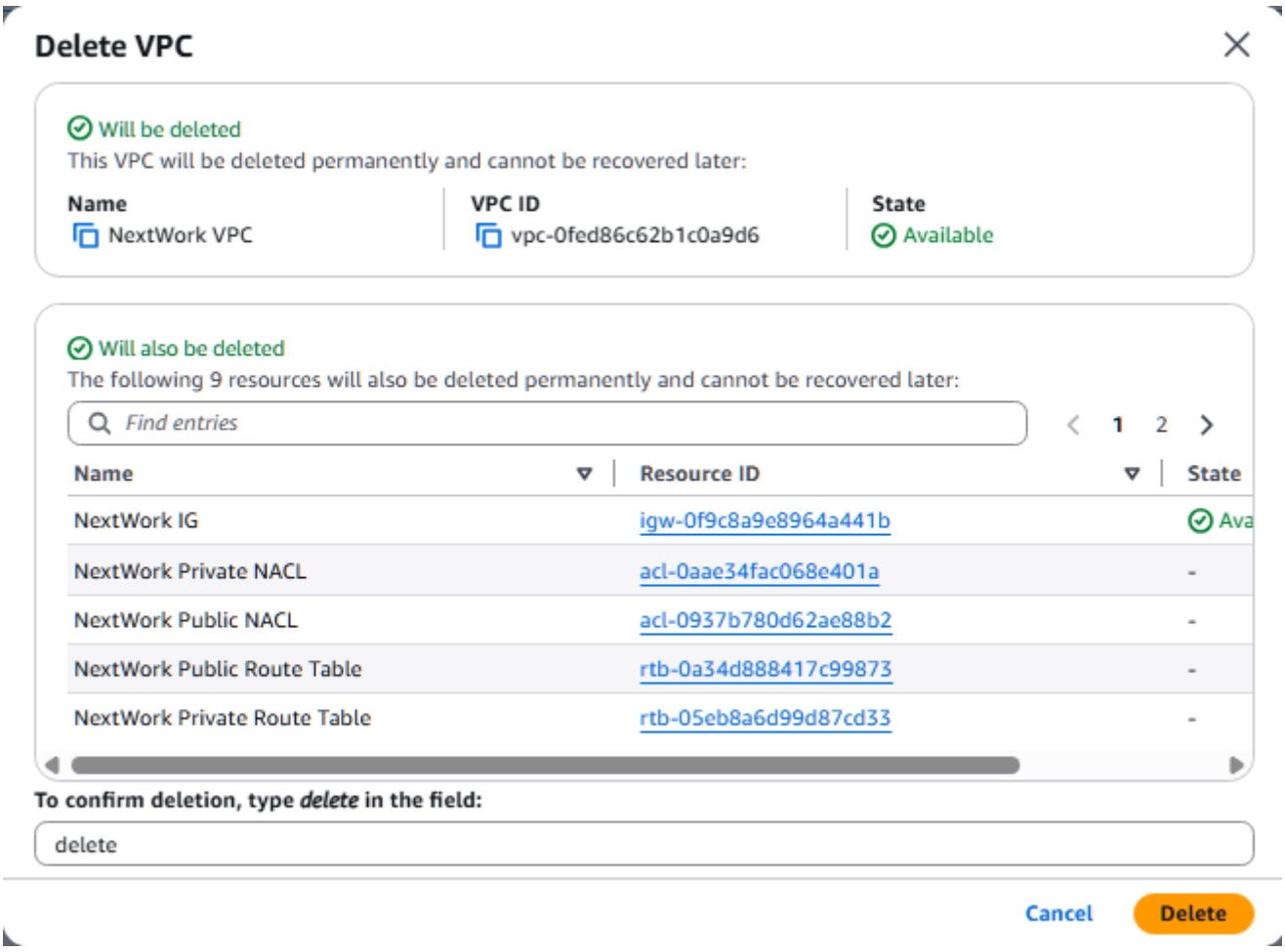
1. **Terminate EC2 Instances** (they're attached to VPCs)
2. **Delete NextWork VPC** (cascade deletes subnets, route tables, etc.)
3. **Delete nextwork-vpc** (wizard-created VPC)

Step 1: Terminate EC2 Instances





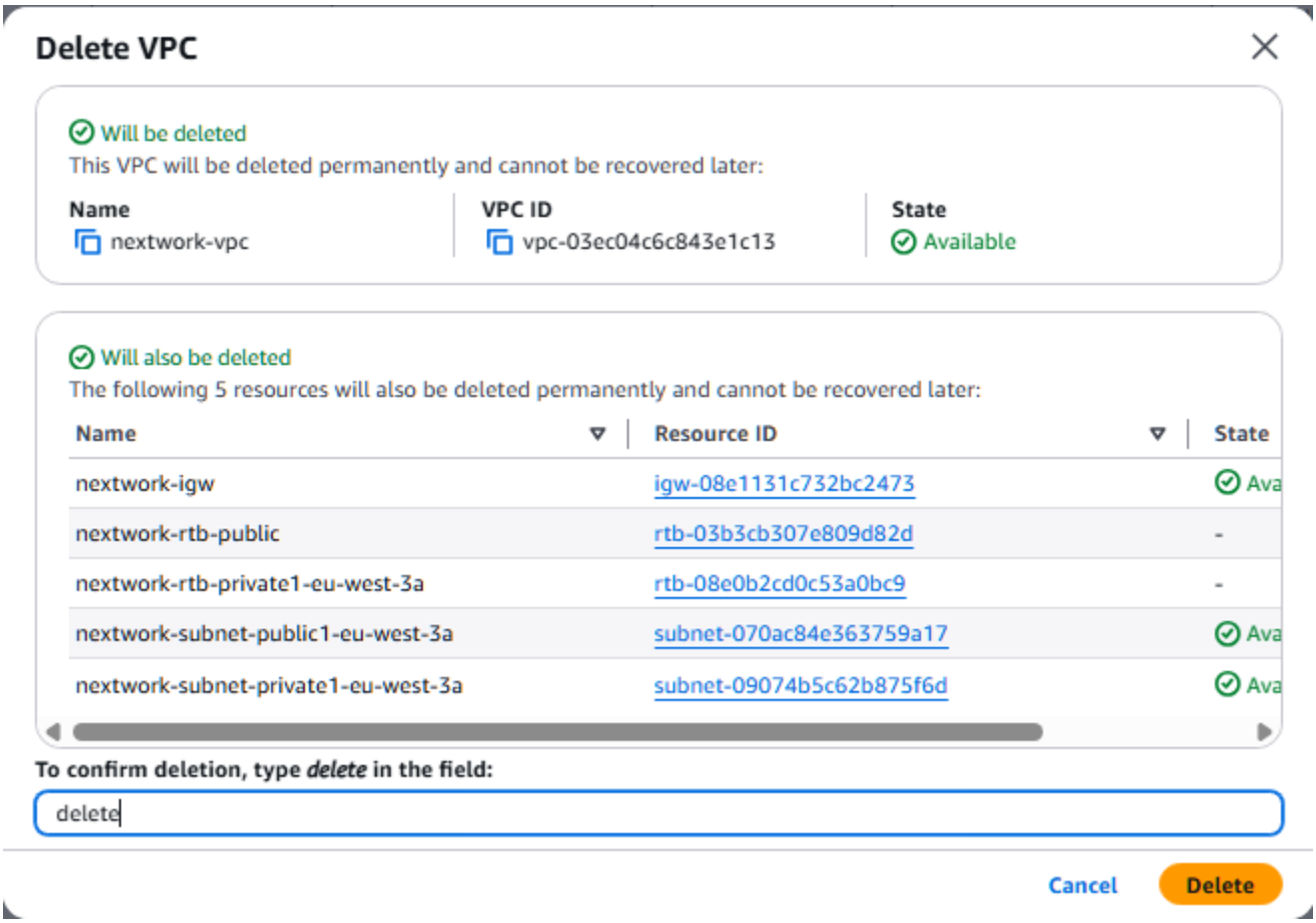
Step 2: Delete NextWork VPC



Why so many resources?

Deleting a VPC cascades to all dependent resources. This is why we manually created these in Projects 1-3 - to understand what the VPC actually contains!

Step 3: Delete nextwork-vpc



Conclusion

This project marked a significant milestone in my AWS learning journey. After spending Parts 1-3 building the networking foundation (VPC, subnets, route tables, internet gateway, security groups, and NACLs), I finally got to **deploy actual resources** into that infrastructure. In **Part 5: Testing VPC Connectivity**, I'll test the network architecture by connecting to instances and verifying that the security configuration works as designed.

Project Completed: January 2026
Author: YOUHAD AYOUB
Region: eu-west-3 (Paris)
NextWork Challenge: AWS Beginners Challenge - Project 7