Create a Virtual Private Cloud (VPC) on AWS

Project Overview

Built a custom Virtual Private Cloud (VPC) on AWS with subnets and internet connectivity. Completed both the AWS Console method and the AWS CLI method using CloudShell to demonstrate different infrastructure deployment approaches.

Duration: ~30 minutes | **Difficulty:** Easy | **Region:** eu-west-3 (Paris)

What I Built

A complete VPC infrastructure including:

- Custom VPC with IPv4 CIDR block (10.0.0.0/16)
- Public subnet in Availability Zone eu-west-3a
- Internet Gateway for internet connectivity
- Auto-assigned public IP addresses for resources

Technologies & Concepts

AWS Services:

- Amazon VPC (Virtual Private Cloud)
- AWS CloudShell
- AWS CLI

Key Concepts:

- VPCs and network isolation
- Subnets (public vs private)
- CIDR blocks and IP addressing
- Internet Gateways
- Availability Zones

Why VPCs Matter

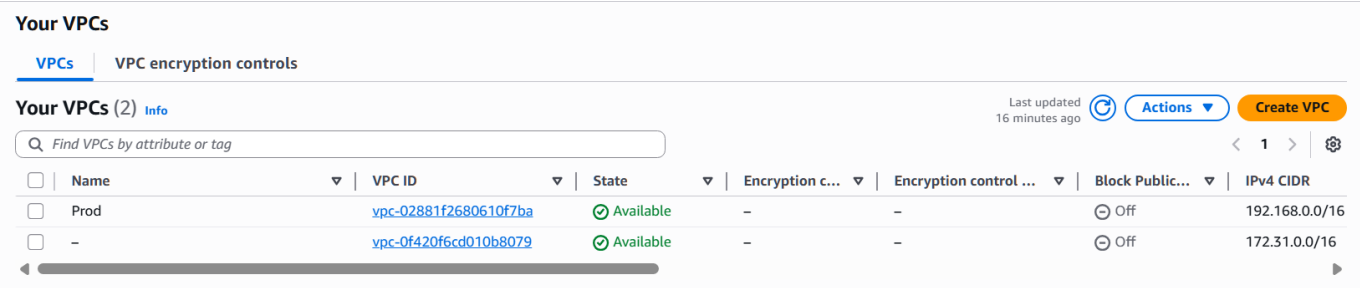
Without VPCs, every AWS resource would exist in one giant open space with no privacy or organization. VPCs provide:

- **Privacy** - Isolate your resources from others
- **Control** - Manage how resources communicate
- **Security** - Set up access rules and restrictions
- **Organization** - Group resources logically

Method 1: AWS Console (GUI)

Step 1: Create the VPC

Navigated to the VPC console to create a custom VPC.

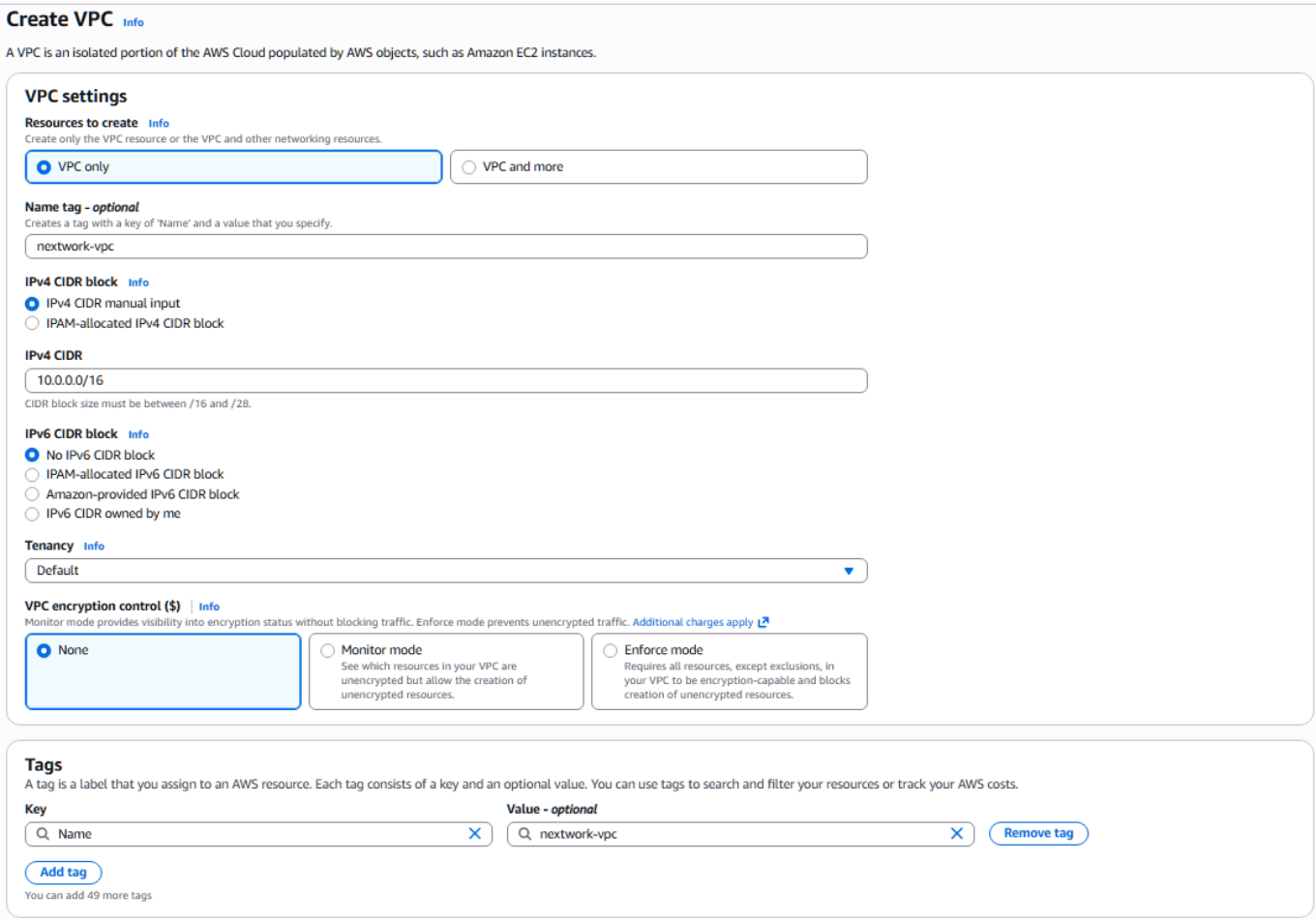


VPC Console showing existing default VPC

Configuration:

- **Name:** nextwork-vpc
- **IPv4 CIDR:** 10.0.0.0/16 (65,536 possible IP addresses)
- **Region:** eu-west-3 (Paris)
- **VPC only** (not VPC and more)

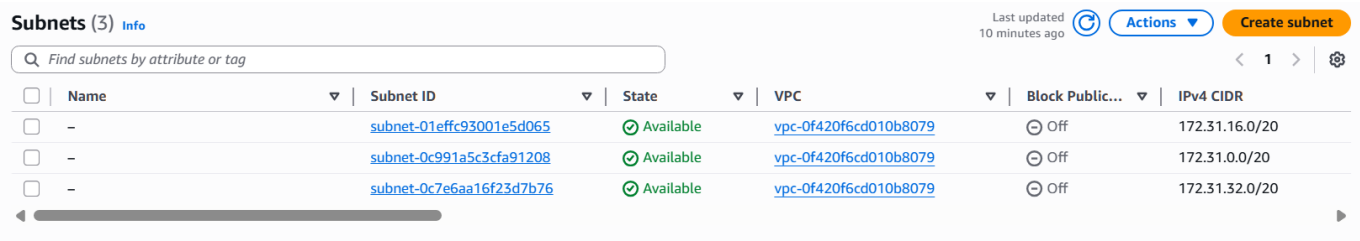
What's a CIDR block? It defines your IP address range. The `/16` means the first 16 bits are fixed (10.0), giving you 65,536 addresses (2^16).



Creating VPC with CIDR block 10.0.0.0/16

Step 2: Create a Public Subnet

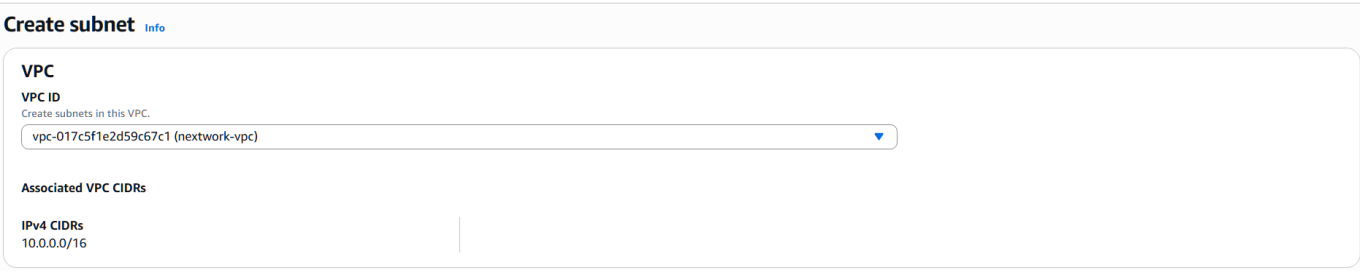
Created a subnet within the VPC - like creating a neighborhood in a city.



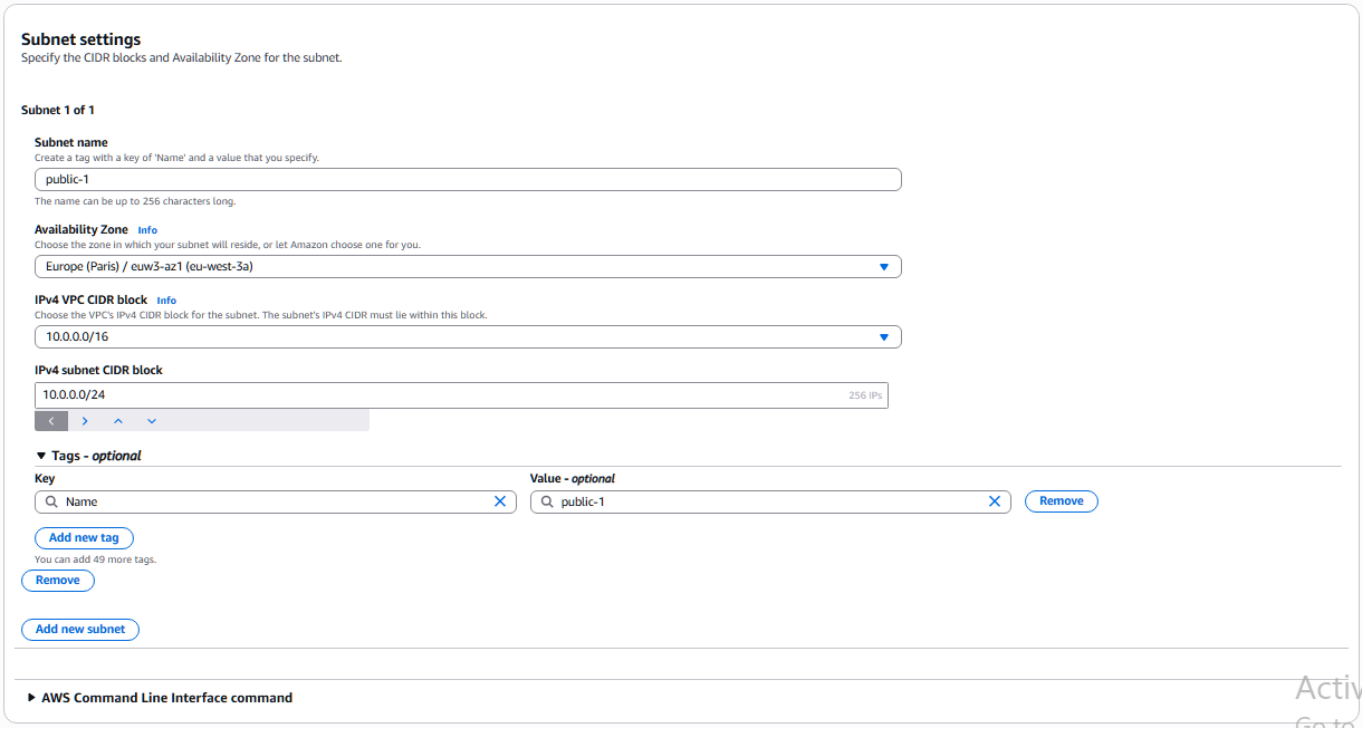
Subnet console showing 3 default subnets

Configuration:

- **VPC:** nextwork-vpc (vpc-017c5f1e2d59c67c1)
- **Name:** public-1
- **Availability Zone:** eu-west-3a (Europe Paris - first AZ)
- **IPv4 CIDR:** 10.0.0.0/24 (256 IP addresses)

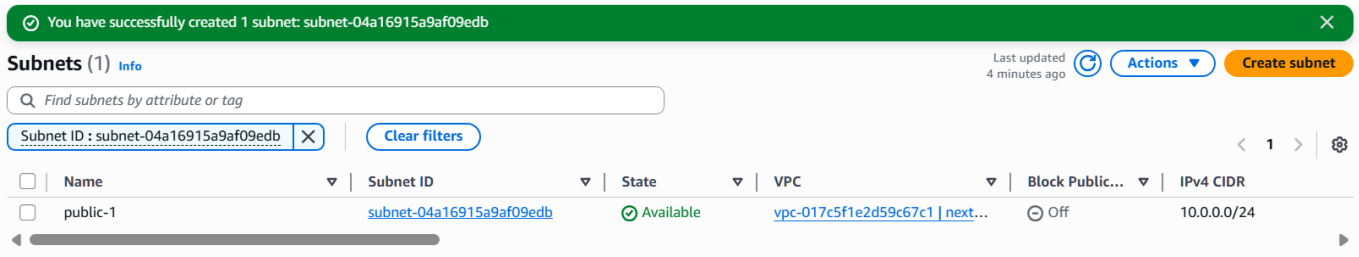


Selecting the VPC for the subnet



Configuring subnet settings with AZ and CIDR block

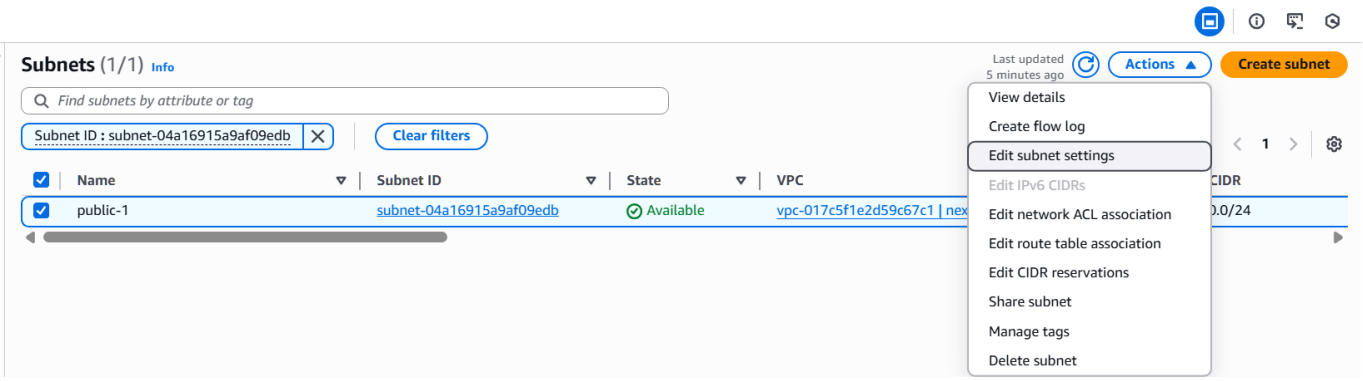
Why "Public"? This subnet will connect to the internet. Private subnets stay isolated for internal resources like databases.



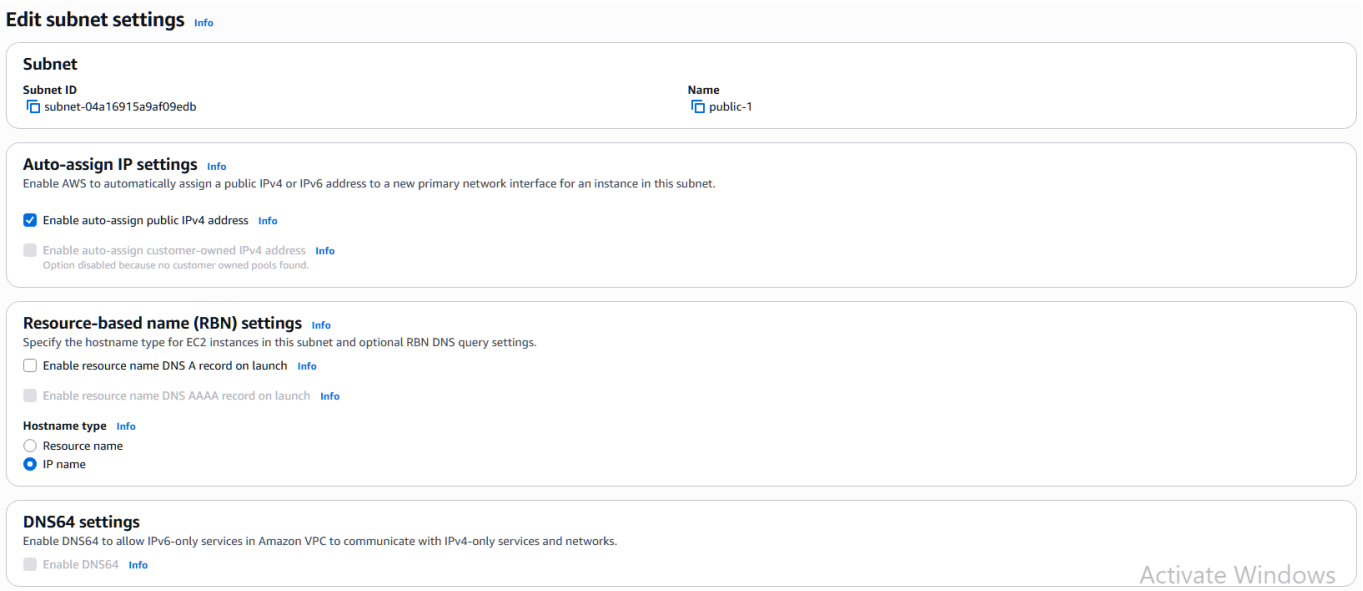
Subnet successfully created with 251 available IPs (AWS reserves 5)

Step 3: Enable Auto-Assign Public IP

Modified the subnet to automatically assign public IP addresses to resources launched in it.



Actions menu - Edit subnet settings



Enabling auto-assign public IPv4 address

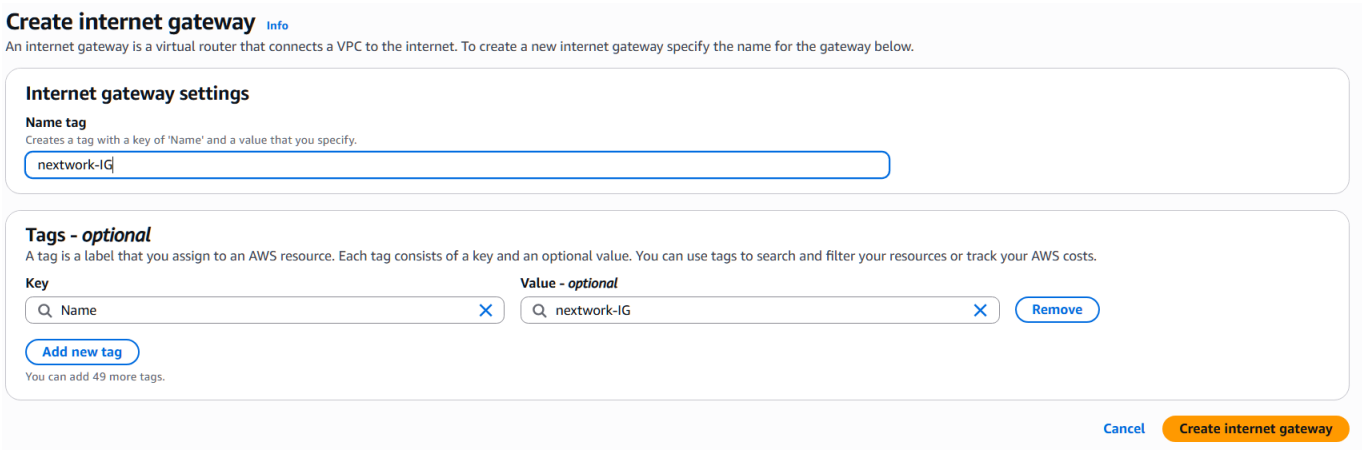
Why this matters: Without this, EC2 instances launched in this subnet won't get public IPs automatically, preventing internet access.

Step 4: Create Internet Gateway

Created an Internet Gateway - the bridge connecting the VPC to the internet.

Configuration:

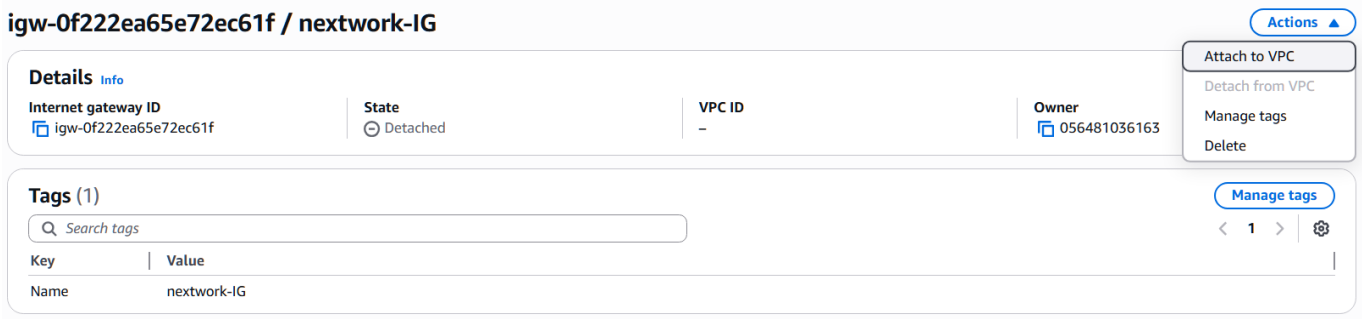
- **Name:** nextwork-IG



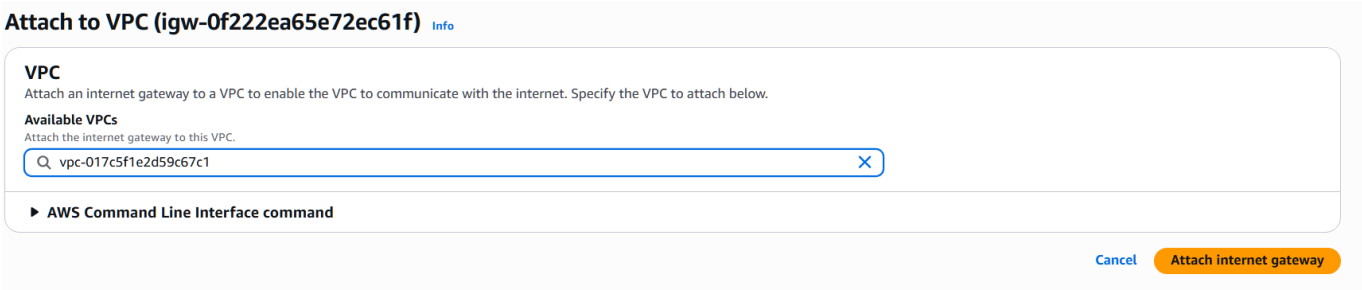
Creating the Internet Gateway

Step 5: Attach Internet Gateway to VPC

Connected the Internet Gateway to the VPC to enable internet connectivity.

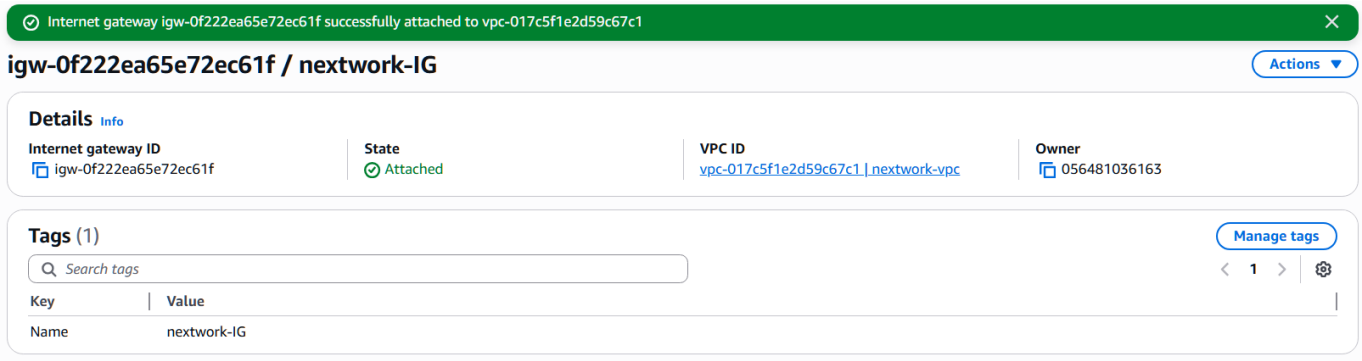


Selecting nextwork-vpc for attachment



Confirming the attachment

What does attaching do? It enables resources with public IPs in the VPC to access the internet and be accessible from the internet.



Internet Gateway successfully attached - State: Attached

Secret Mission: AWS CloudShell & CLI

Instead of clicking through the console, I recreated the entire VPC infrastructure using AWS CLI commands in CloudShell!

Step 6: Open AWS CloudShell

Accessed the browser-based terminal by clicking the >_ icon in the AWS Console top navigation bar.

AWS CloudShell - Pre-configured terminal with AWS CLI ready

What is CloudShell? A browser-based command-line interface with AWS CLI pre-installed. No setup or configuration needed!

Step 7: Create VPC with CLI

Used a single command to create the VPC:

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
'ResourceType=vpc,Tags=[{Key=Name,Value=NextWork-CLI-VPC}]'
```

Output received:

```
{
  "VpcId": "vpc-037197e4fc836955d",
  "State": "pending",
  "CidrBlock": "10.0.0.0/16",
  "IsDefault": false
}
```


Key Info:

- VPC ID: `vpc-037197e4fc836955d` ← Save this for next commands!
- State: pending → available in seconds
- 65,536 IP addresses available

Step 8: Create Subnet with CLI

Created the subnet using the VPC ID:

```
aws ec2 create-subnet \
  --vpc-id vpc-037197e4fc836955d \
  --cidr-block 10.0.0.0/24 \
  --availability-zone eu-west-3a \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-CLI-1}]'
```

Output received:


```
{
  "SubnetId": "subnet-0d91f172da7765724",
  "State": "available",
  "VpcId": "vpc-037197e4fc836955d",
  "CidrBlock": "10.0.0.0/24",
  "AvailableIpAddressCount": 251,
  "MapPublicIpOnLaunch": false
}
```


Key Info:

- Subnet ID: `subnet-0d91f172da7765724` ← Save for next command!
- 251 usable IPs (AWS reserves 5 for internal use)
- Public IP auto-assign is OFF - need to enable it

Step 9: Enable Auto-Assign Public IP

Modified the subnet attribute:

```
aws ec2 modify-subnet-attribute \
  --subnet-id subnet-0d91f172da7765724 \
  --map-public-ip-on-launch
```

Result: No output = Success! ☑

The subnet now automatically assigns public IPs, matching our console configuration.

Step 10: Create Internet Gateway with CLI

Created the internet gateway:

```
aws ec2 create-internet-gateway \
  --tag-specifications 'ResourceType=internet-gateway,Tags=
[{Key=Name,Value=NextWork-CLI-IG}]'
```

Output received:

```
{
  "InternetGatewayId": "igw-062ca4ceebe4d8ab2",
  "Attachments": [],
  "Tags": [{ "Key": "Name", "Value": "NextWork-CLI-IG" }]
}
```

Key Info:

- Internet Gateway ID: `igw-062ca4ceebe4d8ab2` ← Save for next step!
- Attachments empty = not connected yet

Step 11: Attach Internet Gateway to VPC

Connected the gateway to the VPC:

```
aws ec2 attach-internet-gateway \
  --internet-gateway-id igw-062ca4ceebe4d8ab2 \
  --vpc-id vpc-037197e4fc836955d
```

Result: No output = Success! ☑

The Internet Gateway is now attached and functional!

Key Learnings

Technical Skills Gained

- **VPCs isolate resources** - Essential for security and organization
- **CIDR notation** - /16 = 65,536 IPs, /24 = 256 IPs. Lower number = more addresses
- **Subnets organize VPCs** - Like neighborhoods in a city
- **Internet Gateways enable internet** - Must be attached to VPC to work
- **Availability Zones** - Each subnet exists in one AZ for redundancy
- **CLI is significantly faster** - 6 commands vs. many console clicks
- **JSON output structure** - IDs from one command become inputs for the next

Console vs CLI Comparison

Aspect	Console (GUI)	CLI (CloudShell)
Speed	Slower, many clicks	Fast, single commands
Learning Curve	Visual, beginner-friendly	Requires command knowledge
Automation	Manual, repetitive	Scriptable, repeatable
Documentation	Screenshots needed	Commands are self-documenting
Error Messages	Pop-ups	Detailed JSON responses
Reproducibility	Must click again	Re-run saved commands

My Verdict: CLI is definitely faster and more efficient once you learn the commands! Perfect for automation and Infrastructure as Code (IaC).

Best Practices Learned

- ☑ **Choose nearby regions** - Better performance for users
- ☑ **Enable auto-assign public IPs** - On public subnets only
- ☑ **Tag all resources** - Makes identification and cost tracking easier
- ☑ **Save resource IDs** - Essential when using CLI
- ☑ **Delete unused resources** - Avoid unexpected charges
- ☑ **Use CloudShell** - No need to install AWS CLI locally

Cleanup Process

Deleted all resources to avoid charges:

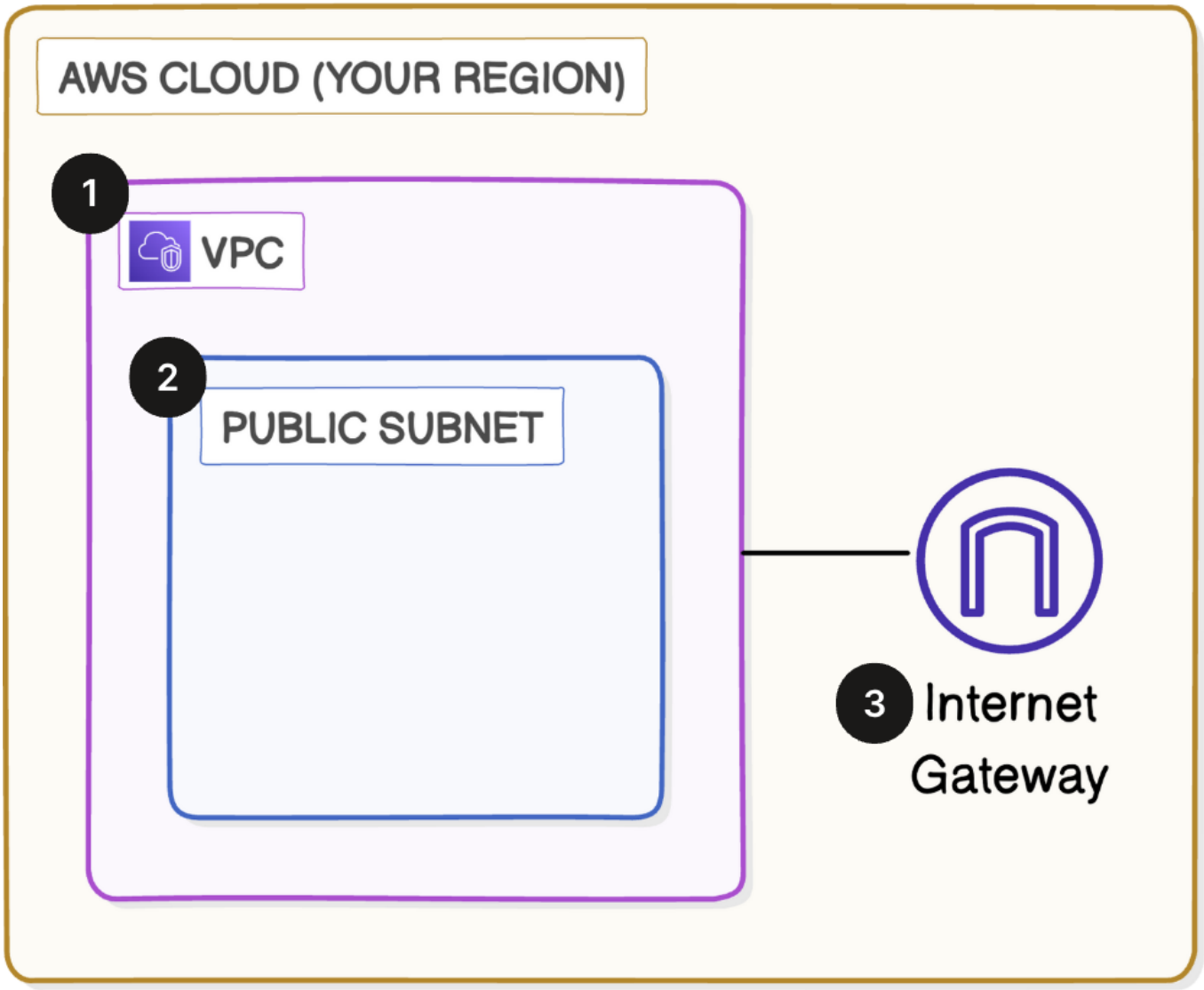
Console Method:

1. Navigate to VPC Dashboard
2. Select nextwork-vpc
3. Actions → Delete VPC
4. Subnet and Internet Gateway deleted automatically ☑

CLI Method: Same process for NextWork-CLI-VPC

Important: When you delete a VPC, AWS automatically removes associated subnets and internet gateways!

Architecture Diagram



What's Next?

This is **Part 1** of the NextWork VPC series! The foundation is built, now it's time to complete the networking setup.

Next Project: VPC Traffic Flow and Security

In the next project, I'll configure:

- **Route Tables** - Direct traffic to the internet gateway
- **Security Groups** - Firewall rules for EC2 instances
- **Network ACLs** - Subnet-level security controls

Conclusion

Successfully created a complete VPC infrastructure using both AWS Console and AWS CLI! The CLI approach proved significantly faster and is essential for automation and Infrastructure as Code practices.

Project Completed: December 2025
Author: YOUHAD AYOUB
Part of: NextWork 7 Day DevOps Challenge - Day 2

This project demonstrates foundational AWS networking skills essential for cloud architecture and DevOps practices.