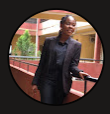
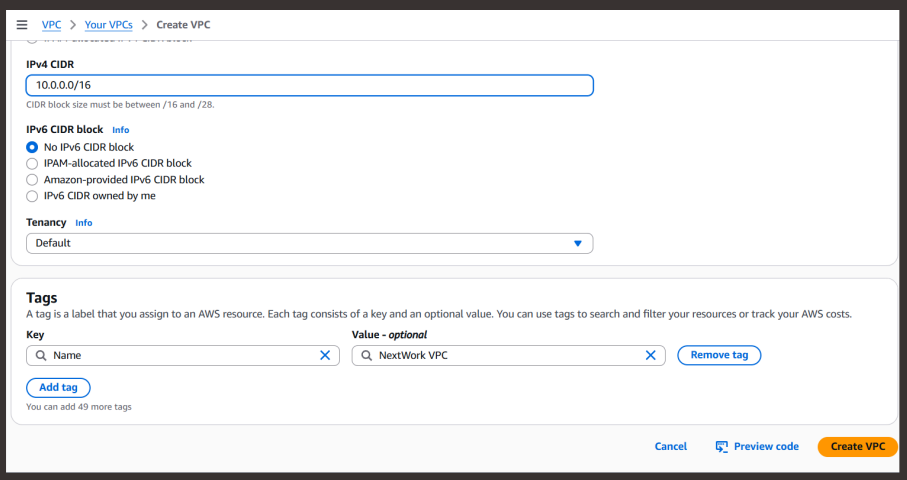
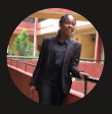
nextwork.org

Build a Virtual Private Cloud (VPC)



Joba Amunigun
https://www.linkedin.com/in/dvoice




Joba Amunigun
NextWork Student

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) is a private, isolated section of the AWS cloud where I can launch and organize resources like EC2 instances, databases, and subnets. It’s useful because it gives me full control over how my resources are connected, secured, and accessed — including IP ranges, routing, and internet access. let's think of it like building your own private space in the cloud — with walls, doors, and keys — instead of tossing everything into a big open internet room.

How I used Amazon VPC in this project

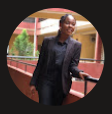
In today’s project, I used Amazon VPC to create a secure and customizable network space in the cloud — using both the AWS Console and CloudShell with the AWS CLI. With the Console, I visually explored how VPC components like subnets, route tables, and gateways fit together. Then, I used CloudShell to practice creating those same resources through commands, which helped me understand what’s happening behind the scenes.

One thing I didn't expect in this project was...

One thing I didn’t expect was how easy it is to miss a step and get stuck — especially when working in the CLI. Something as small as forgetting a CIDR block or misplacing a route table setting can break everything. But that’s where the real learning happens — not just fixing it, but understanding why it broke

This project took me...

This project took me about 90 minutes — including the secret mission, CLI troubleshooting, and most importantly, taking time to truly understand each term and concept. I didn’t want to just finish the task — I wanted to know what I was doing, and why it mattered. That made the extra time completely worth it.


Joba Amunigun
NextWork Student

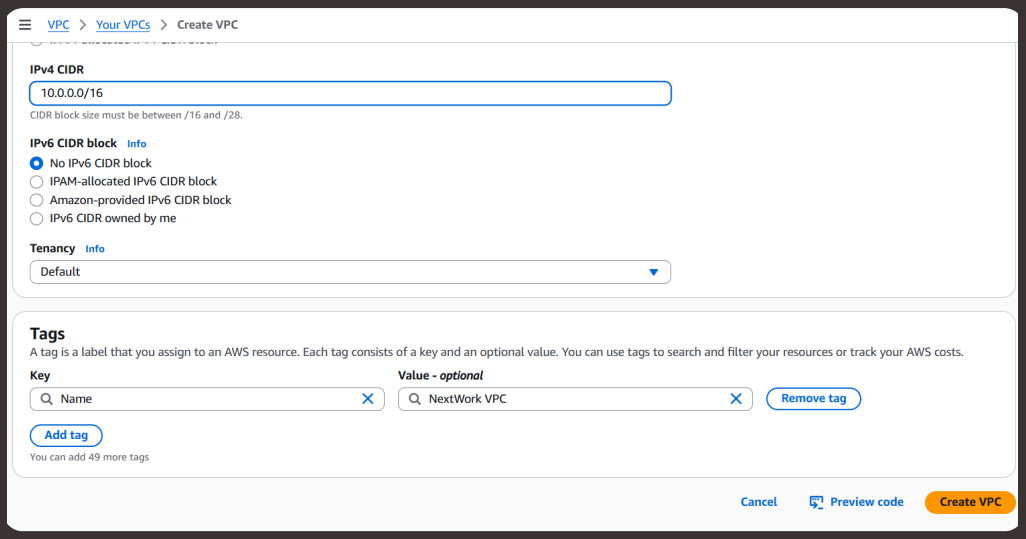
Virtual Private Clouds (VPCs)

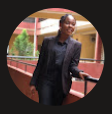
VPCs are like your own private neighborhood in the cloud — a secure space where you control what resources live there, how they connect, and who can access them. To put it simply - without VPCs, every AWS resource would exist in one giant, open space in the cloud, like a country without cities or districts.

There was already a default VPC in my account ever since my AWS account was created. This is because AWS automatically gives every new user a basic, ready-to-use VPC — so you can launch services like EC2 without needing to set up networking from scratch. Think of it like getting a free starter apartment when you sign up — it's move-in ready, but later on, you might want to design your own place to fit your exact needs.

To set up my VPC, I had to define an IPv4 CIDR block, which is a range of IP addresses that my resources can use to communicate within the network.IPv4 (Internet Protocol version 4) is the standard format for writing IP addresses, using four numbers separated by dots — like 192.168.0.1. Each number ranges from 0 to 255, which means there are over 4 billion possible combinations. In a network like a VPC, each device needs its own unique IP address, and IPv4 makes that possible. The CIDR block (Classless Inter-Domain Routing) determines how big or small that range of addresses is. The number after the slash (e.g., /16) controls how many IPs I get. For example, 10.0.0.0/16 means I can assign addresses from 10.0.0.0 to 10.0.255.255, which is 65,536 possible addresses. Think of it like claiming a section of a city and then assigning house numbers — the CIDR block defines how big your section is, and IPv4 tells you the exact "address" of each house - ensuring no 2 house use the same IP.

Joba Amunigun
NextWork Student




Joba Amunigun
NextWork Student

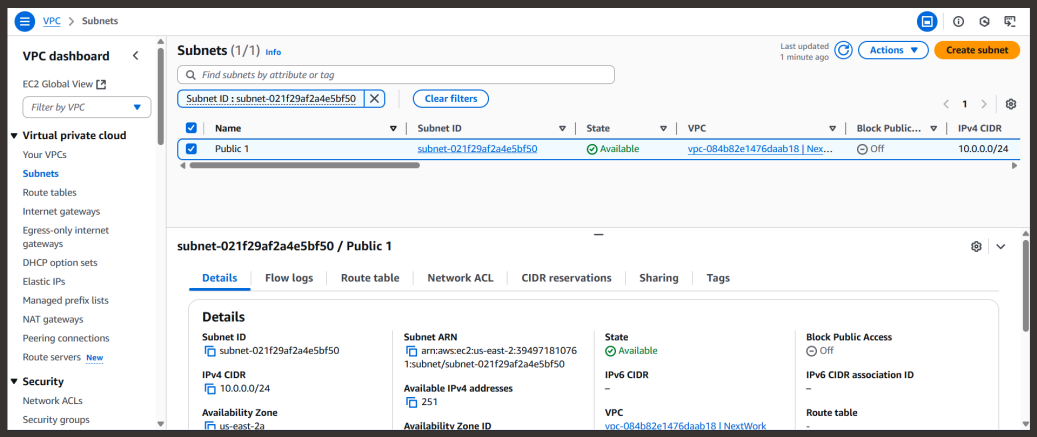
Subnets

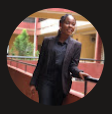
Subnets are smaller sections within a VPC that help organize and separate resources in a network. There are already subnets existing in my account — one for every Availability Zone in the region. This setup makes sure resources are spread out across zones for better availability and fault tolerance. Think of a VPC as a city, and subnets as neighborhoods inside that city — each with its own boundaries and rules for who can enter or communicate within it.

Once I created my subnet, I enabled auto-assign public IPv4 addresses. This setting makes sure that any new instance launched into the subnet automatically gets a public IP address — so that it can be accessed from the internet without needing to create one manually - a huge time saver! □ It's like giving every house in a neighborhood its own mailbox number on the main street — so mail (or traffic) can find it directly.

The difference between public and private subnets is how they connect to the internet. For a subnet to be considered public, it has to be connected to an Internet Gateway, and its route table must direct traffic to that gateway. Right now, my subnet doesn't have that route set up — so even if it exists inside my VPC, it's not reachable from the internet and is still considered private. Think of it like a house without a driveway — it exists, but no one from the outside can reach it unless you build the road (Internet Gateway) and add directions (route table).

Joba Amunigun
NextWork Student



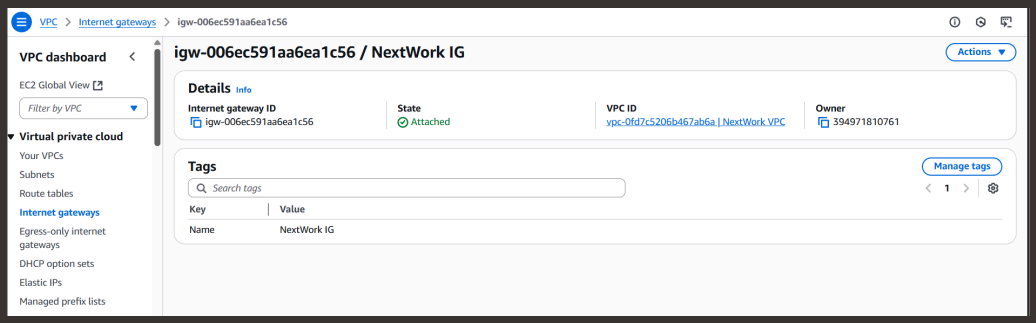

Joba Amunigun
NextWork Student

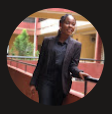
Internet gateways

Internet gateways are components in a VPC that allow resources (like EC2 instances) to send and receive traffic from the internet. They act as the bridge between your private AWS network and the public internet — without them, your instances can't be reached or communicate outside the VPC. Think of an internet gateway like the front gate of a gated estate — it's the entry and exit point that lets your resources talk to the outside world.

Attaching an internet gateway to a VPC means I'm allowing resources inside my VPC (like EC2 instances) to connect to the internet — and also to receive traffic back from it. If I missed this step, even if my subnet and instance were set up correctly, the instance wouldn't be able to access the internet or be reached from it. Internet access just wouldn't work — because there would be no gateway to route traffic in or out of the VPC. Think of it like setting up a house with lights, furniture, and Wi-Fi routers — but forgetting to connect it to the main power grid. The setup exists, but nothing gets through.

Joba Amunigun
NextWork Student




Joba Amunigun
NextWork Student

Using the AWS CLI

VPC resources could also be created with CloudShell, which is a browser-based terminal in the AWS Console that gives you instant access to a secure command-line environment — without installing anything. CLI (Command Line Interface) is the tool used inside CloudShell to run commands that interact with AWS services. Instead of using the AWS Console's buttons and menus, the CLI lets you create and manage resources like VPCs using typed commands. Using the CLI is like sending instructions directly to AWS — it's faster, more precise, and helps you learn how cloud infrastructure really works behind the scenes. Let me know if you'd like to follow this up with an actual command explanation or a breakdown of what aws ec2 create-vpc does!

To set up a VPC or a subnet, you can use the command "aws ec2 create-subnet --vpc-id [VPC-ID] " in CloudShell — which I did. But when I ran it without including a "--cidr-block", I got a MissingParameters error. Turns out, AWS CLI needs to know what range of IP addresses the subnet should use — and without the CIDR block, it can't create one. That's when it clicked: I can't build part of a network without telling AWS how much "space" to assign. Now I know — always include the CIDR block when using the create-subnet command! (i.e. aws ec2 create-subnet --vpc-id [VPC-ID] --cidr-block [ADD-CIDR-BLOCK-HERE]) It's like asking AWS to build a room without telling it how big the floor plan should be.

Compared to using the AWS Console, an advantage of using commands is that it's faster and gives you more control — especially when repeating tasks or automating setups. You start to feel more like a real cloud engineer when you're working directly with the AWS CLI. An advantage of using the Console is that it's more visual and beginner-friendly.

It helps you understand what you're building by letting you see each step and double-check settings more easily. Overall, I preferred using the CLI in CloudShell once I got the hang of it — because it pushed me to really understand what each part of the VPC setup does, instead of just clicking through.

nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects