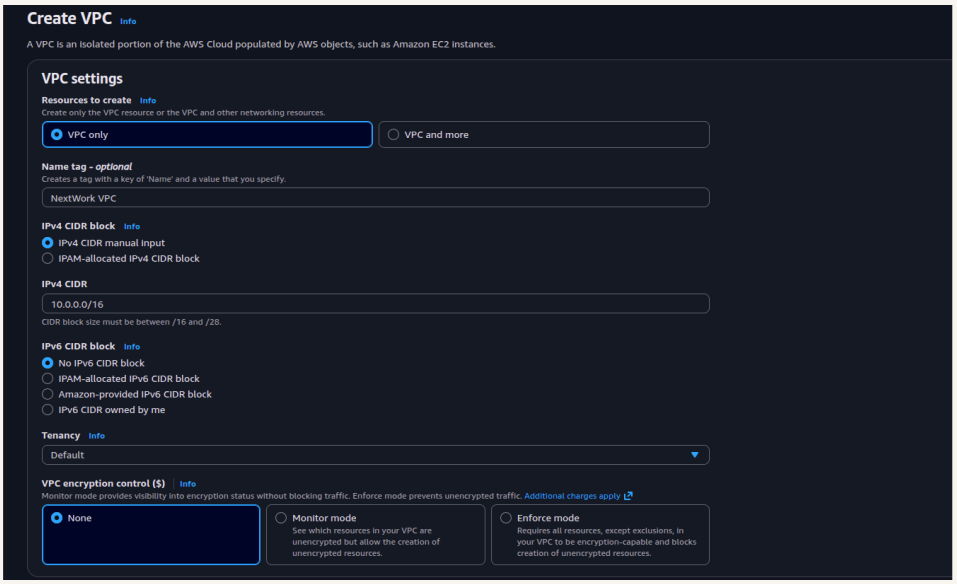
Build a Virtual Private Cloud (VPC)

Clive Maduke



Clive Maduke
NextWork Student

Introducing Today's Project!

In this project, I will learn how to set up my own private network in the AWS cloud. It is like building a secure digital neighborhood where my future applications and websites can live. I am doing this project to understand the basics of cloud networking. I want to learn what a VPC, subnets, and gateways are, and how to connect things safely to the internet. This is the first, most important step for hosting anything on AWS!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) is your own logically isolated, configurable network space within the AWS cloud. It's like having a private data center in the cloud that you control. It is useful because it provides the foundation for security and control. It allows you to: Launch resources in a defined virtual network. Control inbound and outbound traffic with security groups and network ACLs. Segment your network into public and private subnets for layered security. Connect securely to your on-premises data centers or other VPCs. Without a VPC, your AWS resources would be in a shared, flat network. With it, you build secure, production-ready architectures.

placeholder

In today's project, I used Amazon VPC to build a secure and structured network foundation from the ground up. I created a custom VPC with a defined IP address range, then subdivided it into public and private subnets across Availability Zones for high availability. I attached an Internet Gateway to enable public internet access for resources in the public subnet, while keeping backend resources like databases isolated in the private subnet.

This hands-on setup gave me complete control over the network environment, allowing me to implement core AWS networking and security best practices in a practical, portfolio ready way.

Personal reflection

This project took me approximately 2-3 hours to complete from start to finish, including the console-based setup and the CloudShell/CLI extension. The time was well spent on understanding the relationships between components (VPC, subnets, gateways, route tables) rather than just the clicks. The initial console workflow helped with visualization, while the CLI portion reinforced the precise, repeatable nature of Infrastructure as Code. Breaking it into clear, documented steps made the process manageable and educational.

placeholder

One thing I did not expect in this project was how fundamental and interconnected every component is. For instance, simply creating a subnet does not make it "public"; it requires the separate, explicit step of creating an Internet Gateway and correctly editing the route table.

Clive Maduke
NextWork Student

This taught me that cloud networking is like building with precise, independent blocks each step must be intentional and correctly linked for the whole structure to work.

Clive Maduke
NextWork Student

Virtual Private Clouds (VPCs)

What I did in this step

In this step, I will access the AWS VPC Dashboard and create my core Virtual Private Cloud. This VPC will be my project's dedicated, isolated network space in the AWS cloud, defined by a specific IP address range. To create it, I will navigate to the VPC service from the AWS Console, click "Create VPC," and provide a name (like "MyPortfolio-VPC") and the IPv4 CIDR block (for example, 10.0.0.0/16). This simple action lays the secure foundation for all the subsequent networking components I will build.

How VPCs work

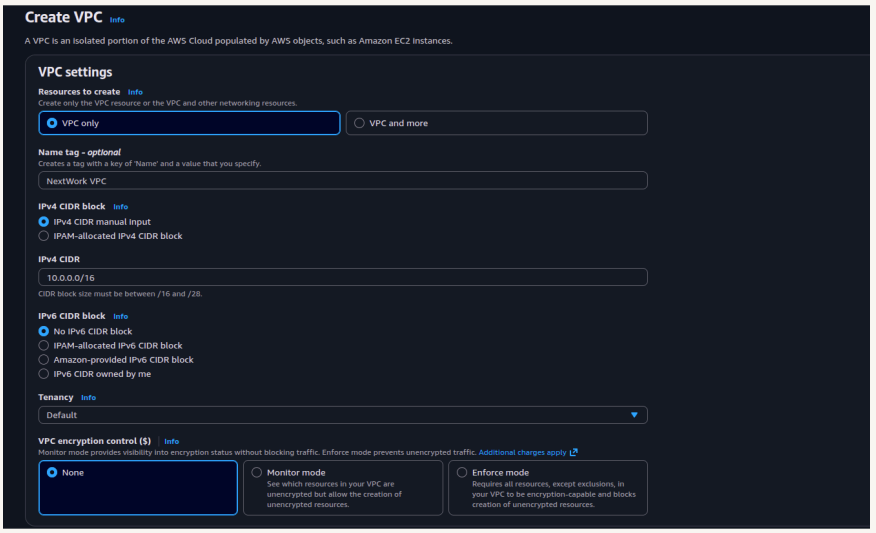
A VPC (Virtual Private Cloud) is your own logically isolated section of the AWS cloud. I think of it as our private data centers within AWS, where you have complete control over your virtual networking environment. You decide its IP address range, create subnets, and configure route tables and network gateways. It's the secure, foundational network where you launch resources like EC2 instances, keeping them separate from other AWS customers and the public internet until you choose to connect them.

Why there is a default VPC in AWS accounts

There was already a default VPC in my account ever since my AWS account was created. This is because AWS provides it for convenience, allowing the users to launch resources like EC2 instances immediately without needing to first configure networking. It comes pre-configured with a public subnet, an internet gateway, and default security settings.

Clive Maduke
NextWork Student

This "ready-to-use" network is great for quick testing and learning, but for production or specific architectures, it is best practice to build a custom VPC from scratch for full control and security.



Clive Maduke
NextWork Student

Defining IPv4 CIDR blocks

To set up my VPC, I had to define an IPv4 CIDR block, which is essentially the blueprint for my private network’s address space. It consists of two parts: a base IP address (like 10.0.0.0) that acts as the network’s starting point, and a prefix (like /16) that determines its size. This prefix specifies how many unique addresses will be available for my cloud resources. For my project, I chose 10.0.0.0/16. This block creates a large, private range that can host over 65,000 resources, giving me plenty of room for subnets and future expansion while keeping the network logically isolated.

Clive Maduke
NextWork Student

Subnets

What I did in this step

In this step, I will create a subnet inside my VPC. Think of the VPC as my entire private cloud neighborhood—the subnet is like a specific, organized street within it. I'm doing this because I need to group my resources logically and control their access. For instance, I can create a public subnet (like a street with direct access to the main road) for web servers that need internet access, and a private subnet (like a quiet cul-de-sac) for secure databases. This separation is a fundamental security and organizational practice in cloud architecture.

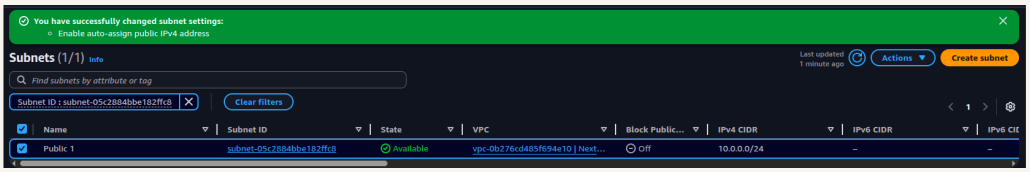
Creating and configuring subnets

Subnets are subdivisions of your VPC’s IP address range, allowing you to group resources within specific Availability Zones for better organization and security. There are already subnets existing in my account, one for every Availability Zone in my region, placed inside the default VPC. These are pre-configured as public subnets to allow quick resource launches. However, for my custom project VPC, I am intentionally creating my own subnets to design a proper network layout with isolated public and private sections, aligning with best practices for control and security.

Public vs private subnets

The difference between public and private subnets lies in their internet access. A public subnet has a direct route to the internet via an Internet Gateway, allowing resources to be publicly reachable.

A private subnet does not have this direct route, keeping its resources isolated. For a subnet to be considered public, it has to have its traffic routed through an Internet Gateway. Currently, my subnet's route table does not have this gateway configured, making it private by default. In the next step, I will create and attach an Internet Gateway to create a true public subnet.



Clive Maduke
NextWork Student

Auto-assigning public IPv4 addresses

Once I created my public subnet, I enabled the "Auto-assign public IPv4 address" setting. This configuration ensures that any EC2 instance I launch directly into this subnet will automatically receive a public IP address from AWS. This is a crucial convenience for resources that need to be directly accessible from the internet, like a web server. Without this setting, I would have to manually assign an Elastic IP address to each new instance, adding an extra step. By enabling auto-assign, I streamline the process of making instances publicly reachable as soon as they start.

Clive Maduke
NextWork Student

Internet gateways

What I did in this step

In this step, I will attach an Internet Gateway (IGW) to my VPC. This gateway acts as the front door between my private cloud network and the public internet. I am doing this because my resources in public subnets like a web server need a way to communicate with users online. Without this gateway, the VPC is completely isolated. By creating and attaching the IGW, and then updating the route table of my public subnet to point traffic to it, I enable inbound and outbound internet access for the resources I place there.

Setting up internet gateways

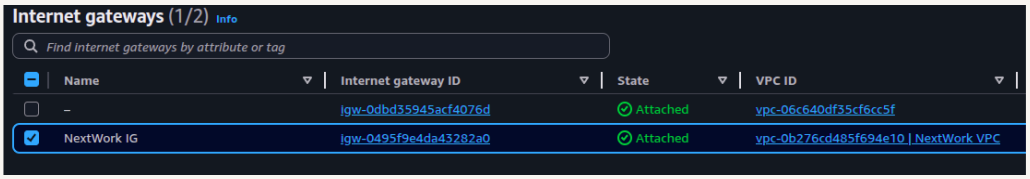
Internet gateways are the main doorway between your VPC and the public internet. They serve two key purposes: they allow resources within your public subnets (like a web server) to connect out to the internet to fetch updates, and they enable the internet to connect in to those resources, like when a user visits your website. Think of an Internet Gateway as a secure, managed translator that connects your private network addressing to public IP addresses. It is a highly available and scalable component built by AWS, and it’s the foundational piece that makes a subnet truly "public."

placeholder

Attaching an internet gateway to a VPC means establishing a controlled, two-way connection to the global internet.

Clive Maduke
NextWork Student

It provides the essential pathway for network traffic between your public subnets and the outside world, enabling features like public web hosting and software updates. If I missed this step, my VPC would remain a completely isolated private network. Any instances in it, even in what I've designated as a "public" subnet, would be unable to reach the internet for updates or be reached by users. My resources would be stranded, making the VPC unusable for any application that requires online connectivity. This gateway is the non-negotiable link that brings a cloud network to life.



Clive Maduke
NextWork Student

Using the AWS CLI

What I'm doing in this extension

In this project extension, I will use AWS CloudShell and the AWS Command Line Interface (CLI) to build my VPC infrastructure entirely through code. I'll run commands to programmatically create the VPC, subnets, and internet gateway. I am doing this because it demonstrates a crucial real-world skill: Infrastructure as Code (IaC). Using the CLI automates the process, ensures consistency, and allows for easy replication or modification. This adds a valuable technical layer to my portfolio, showing I can manage AWS resources beyond just the visual console.

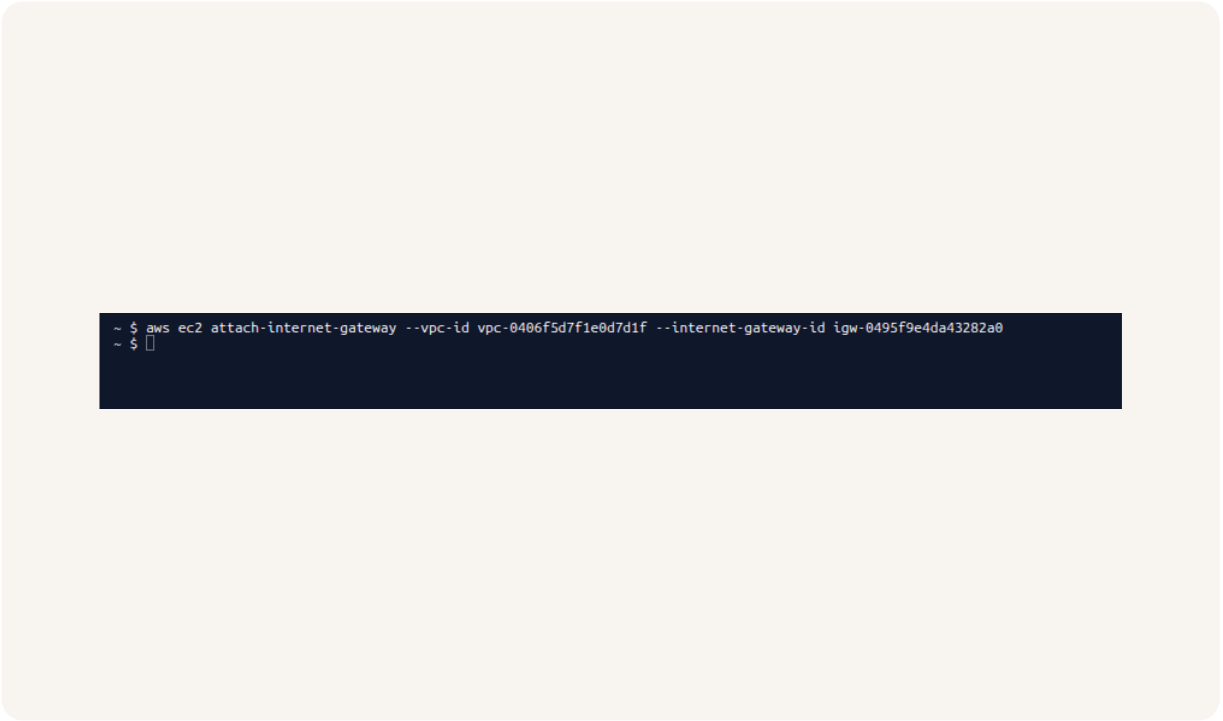
Exploring CloudShell and CLI

CloudShell is a browser-based terminal built directly into the AWS Console. It gives me instant, authenticated command-line access to my AWS account without needing to install or configure any software on my own computer. The CLI (Command Line Interface) is a powerful tool that lets me control almost all AWS services by typing text commands. Instead of clicking buttons in the web console, I write scripts (like aws ec2 create-vpc) to build and manage my infrastructure quickly and repeatably. Together, they allow for fast, programmatic control of the cloud.

Debugging my setup

To set up a VPC, you use the aws ec2 create-vpc command with a --cidr-block parameter, like: aws ec2 create-vpc --cidr-block 10.0.0.0/16 Make sure to avoid errors by including the mandatory --cidr-block and ensuring its format is correct (e.g., a valid private range like 10.0.0.0/16).

Also, remember to attach the correct region to your command if you're not using the default. A common mistake is omitting this required parameter, which causes an immediate syntax error.



Comparing CloudShell vs AWS Console

Compared to using the AWS Console, an advantage of using commands is automation and repeatability.

Clive Maduke
NextWork Student
nextwork.org

I can script the entire setup, save it, and recreate my VPC identically with a single script, which is perfect for documentation and deployment pipelines. An advantage of using the Console is its visual clarity and discoverability. It's easier to explore relationships between resources, see immediate visual feedback, and learn what options are available through intuitive menus. Overall, I preferred starting with the Console to learn the concepts visually, then using CloudShell/CLI to master the precise, repeatable commands combining the best of both approaches for a complete understanding.

The place to learn & showcase your skills

Check out nextwork.org for more projects