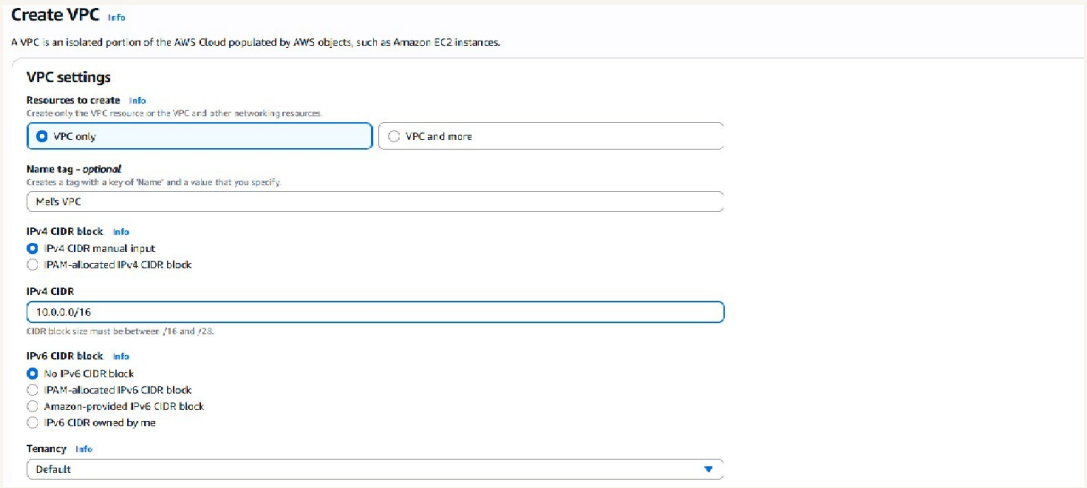
Build a Virtual Private Cloud (VPC)

Melvin green



Introducing Today's Project!

In this project, I will demonstrate how to set up and configure a VPC from scratch, as setting up and managing a VPC is a vital skill for looking to master cloud infrastructure. I'm doing this project to learn the core of AWS networking by creating my very own Virtual Private Cloud (VPC).

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) is your own private, isolated section of the AWS cloud where you have full control over your network environment. It allows you to define your own IP address ranges, create subnets, configure routing rules, and control exactly what traffic gets in and out of your resources. This isolation is what makes VPC such a critical security and architectural tool; sensitive resources like databases can be kept completely private while public-facing resources like web servers remain accessible. Almost every company running on AWS uses a VPC, making it one of the most fundamental concepts in cloud computing and a core skill for any cloud support engineer.

In this project, I built an Amazon VPC from the ground up to understand how cloud networking works in a real environment. I started by defining an IPv4 CIDR block to assign a range of IP addresses to my private network, then divided that network into subnets to organize where different resources would live, separating public-facing resources from private backend ones. I attached an internet gateway to open the connection between my private network and the public internet, allowing my public subnet to handle real traffic. I also explored AWS CloudShell and the AWS CLI, where I ran commands like aws EC2 create subnets to create and configure resources directly from the terminal instead of clicking through the console. Throughout the project, I learned the importance of correct CIDR block formatting to avoid configuration errors and gained a deeper understanding of how each component works. The CIDR block, subnets, and internet gateway connect to form a fully functioning cloud network

I assumed the console would always be the easier way to manage resources, but once I started running commands inside AWS CloudShell, I realized how much faster and more precise the CLI is. What really surprised me was that CloudShell already had the CLI pre-installed, no downloads or setup needed.
It completely changed how I think about working in the cloud.

Virtual Private Clouds (VPCs)

What I did in this step

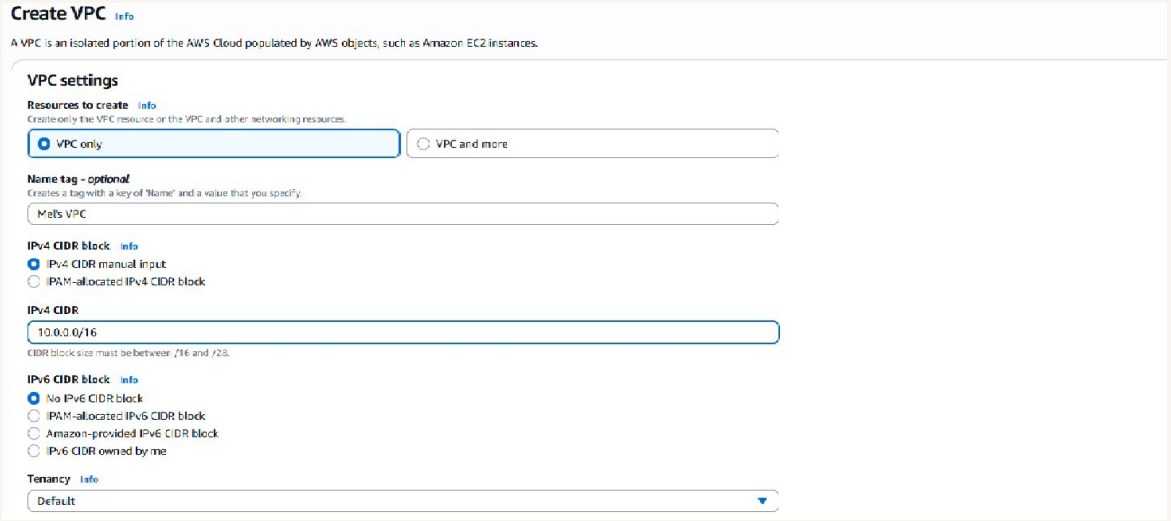
In this step, access the Amazon VPC console in Amazon Web Services and create a new VPC by choosing a setup option, naming it, assigning an IP range, and clicking Create VPC to establish your private cloud network.

How VPCs work

VPC (Virtual Private Cloud) is essentially your own private section of the AWS cloud. I think of it as my own personal network within Amazon's massive infrastructure. The simplest way to think about it: Imagine AWS is a huge apartment building. A VPC is your individual apartment; you have your own space, your own rules, and control over who gets in and out.

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 has set up a default VPC to allow me to deploy resources like EC2/RDS databases right away - without having to create my own VPC from scratch.



Defining IPv4 CIDR blocks

To set up my VPC, I had to define an IPv4 CIDR block, which is a way to assign a whole block of IP addresses to my private network. This determines how many resources, such as EC2 instances and services, can exist within the VPC.

Subnets

What I did in this step

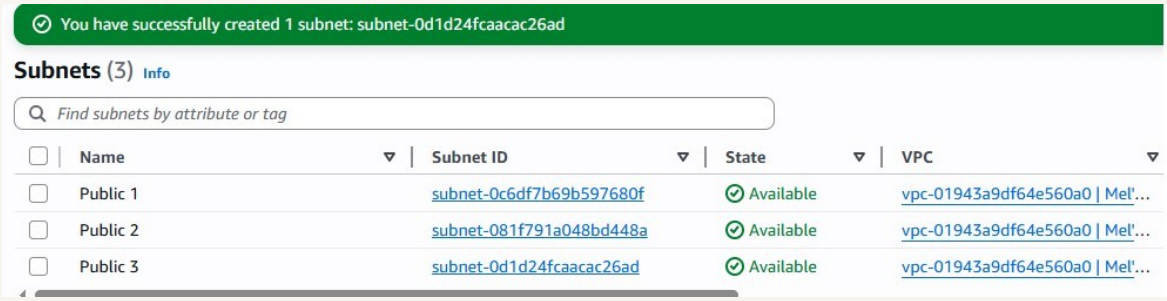
In this step, I will divide my VPC into subnets to start organizing where different resources would live.

Creating and configuring subnets

Subnets are subsections of my VPC, just like how neighborhoods are subsections of a city. There are already subnets in my account, one for every Availability zone in the region that I've set up my VPC in.

Public vs private subnets

I named my subnet public 1, but that doesn't automatically make my subnet a public subnet. For a subnet to be considered public, it must be connected to an internet gateway.



Auto-assigning public IPv4 addresses

Once I created my subnet, I enabled auto-assign public IPv4 address. This setting makes sure. Any EC2 instance launched in that subnet will instantly get a public IP address, so I won't have to create one manually.

Internet gateways

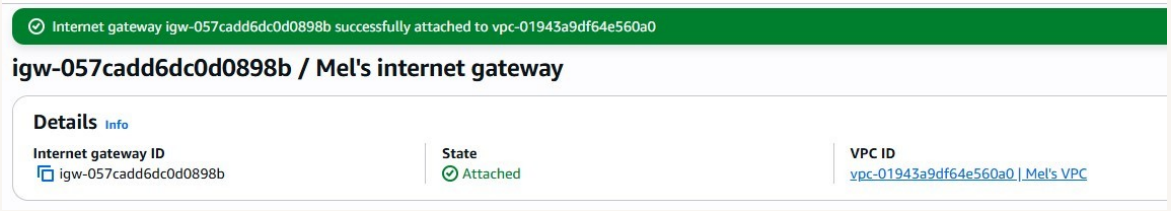
What I did in this step

In this step, I will be connecting my VPC to an internet gateway because a VPC is completely isolated by default and has no way to communicate with the outside world on its own. Without an internet gateway, any EC2 instances or resources running inside my VPC would be unreachable from the internet and unable to send or receive any outside traffic.

Setting up internet gateways

An Internet Gateway is a component that allows internet access for resources in my VPC/subnet. An Internet gateway is also how users in the public can access my resources in a public subnet.

Attaching an internet gateway to a VPC means resources in the VPC can now access the internet. The EC2 instances with public IP addresses also become accessible to users, so the applications hosted on those servers become public too.


Internet gateway igw-057cadd6dc0d0898b successfully attached to vpc-01943a9df64e560a0
igw-057cadd6dc0d0898b / Mel's internet gateway
Details Info
Internet gateway ID
igw-057cadd6dc0d0898b
State
Attached
VPC ID
vpc-01943a9df64e560a0 | Mel's VPC

Using the AWS CLI

What I'm doing in this extension

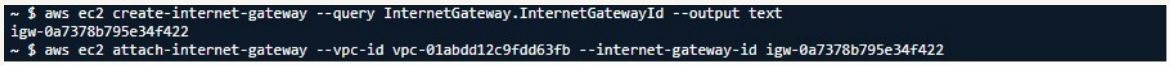
In this project extension, I will use the AWS CLI to launch a VPC's resources because I want to report back on whether it was a faster, more efficient way to do this project.

Exploring CloudShell and CLI

VPC resources could also be created with CloudShell, is a shell in your AWS Management Console, which means it's a space for you to run code. The awesome thing about AWS CloudShell is that it already has AWS CLI preinstalled. AWS CLI (Command Line Interface) is a software that lets you create, delete, and update AWS resources with commands instead of clicking through your console.

Debugging my setup

To set up a VPC or a subnet, you can use the command aws ec2 create-subnet -vpc-id vpc-01abdd12c9fdd63fb --cidr-block 10.0.0.0/24. To avoid errors, make sure the CIDR block is formatted correctly. This means ensuring the subnet CIDR block falls within the VPC's CIDR block, does not overlap with the VPC's CIDR block, and is not larger than the VPC's CIDR block



Comparing Cloud Shell vs AWS Console

Compared to using the AWS Console, an advantage of using the CLI is that it provides the speed and versatility that professionals need for complex tasks. You can also use CLI to automate tasks using scripts, making it essential for managing cloud environments efficiently. An advantage of using the Console is that it provides a visual, point-and-click interface that is much easier for beginners to navigate and understand what's happening inside their environment. Overall, I preferred starting with the Console to build my foundational understanding, but I can see myself relying more heavily on CLI as my skills grow and my tasks become more complex.