



Build a Virtual Private Cloud (VPC)



Anas Oubassou

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

☒ VPC only ☐ VPC and more

Name tag - optional [Info](#)
Creates a tag with a key of 'Name' and a value that you specify.

NextWork VPC

IPv4 CIDR block [Info](#)
☒ IPv4 CIDR manual input
☐ IPAM-allocated IPv4 CIDR block

IPv4 CIDR
10.0.0.0/16
CIDR block size must be between /16 and /28.

IPv6 CIDR block [Info](#)
☒ No IPv6 CIDR block
☐ IPAM-allocated IPv6 CIDR block
☐ Amazon-provided IPv6 CIDR block
☐ IPv6 CIDR owned by me

Tenancy [Info](#)
Default

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional	
<input type="text" value="Name"/>	<input type="text" value="NextWork VPC"/>	Remove tag

[Add tag](#)

You can add 49 more tags.

[Cancel](#) [Preview code](#) [Create VPC](#)



Introducing Today's Project!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) is a fundamental AWS service that enables you to create isolated virtual networks within the AWS cloud environment. What makes it particularly valuable is the comprehensive control it provides over your network infrastructure. You can define your own IP address ranges, configure subnets according to your specific needs, and customize routing tables to direct traffic exactly where you want it. This level of control significantly enhances security by allowing you to isolate resources and implement multiple layers of access controls, creating a foundation for secure cloud deployments.

How I used Amazon VPC in this project

For today's project, I built a custom VPC with a public subnet to host internet-accessible resources. I began by defining a 10.0.0.0/16 CIDR block for my VPC, which provided a generous pool of IP addresses. Within this VPC, I created a subnet with a more specific CIDR range (10.0.0.0/24) to organize my resources efficiently. To ensure resources in this subnet could communicate with the internet, I enabled the auto-assign public IPv4 setting and attached an internet gateway to my VPC. This configuration established the necessary pathway for my resources to be accessible from the public internet while maintaining the security benefits of a controlled VPC environment.

Virtual Private Clouds (VPCs)

VPCs serve as isolated virtual networks within AWS that create a private, secure environment for your cloud resources. They provide the flexibility to organize resources logically, control traffic flows between different application tiers, and establish connection rules without exposing your infrastructure to the public internet.

I discovered that my AWS account already contained a default VPC since creation. AWS implements this automatically to ensure users can immediately begin using services like EC2 without first needing to learn complex VPC configuration details. When setting up my custom VPC, I needed to define an IPv4 CIDR block, which specifies the range of IP addresses available within the network. The 10.0.0.0/16 block I selected provides 65,536 possible address (ranging from 10.0.0.0 to 10.0.255.255), offering ample capacity for various resources and proper network segmentation.

The screenshot displays the 'Create VPC' wizard in the AWS Management Console. The 'VPC settings' section is active, showing options for 'Resources to create' (VPC only selected), 'Name tag' (NextWork-VPC), 'IPv4 CIDR block' (10.0.0.0/16), 'IPv6 CIDR block' (No IPv6 CIDR block selected), and 'Tenancy' (Default). The 'Tags' section below shows a single tag with key 'Name' and value 'NextWork-VPC'. At the bottom, there are buttons for 'Cancel', 'Preview code', and 'Create VPC'.

VPC settings

Resources to create info
Creates only the VPC resource or the VPC and other networking resources.

☒ VPC only ☐ VPC and more

Name tag - optional info
Creates a tag with a key of 'Name' and a value that you specify.

NextWork-VPC

IPv4 CIDR block info
☒ IPv4 CIDR manual input
☐ IPAM-allocated IPv4 CIDR block

IPv4 CIDR
10.0.0.0/16
CIDR block size must be between /16 and /28.

IPv6 CIDR block info
☒ No IPv6 CIDR block
☐ IPAM-allocated IPv6 CIDR block
☐ Amazon-provided IPv6 CIDR block
☐ IPv6 CIDR owned by me

Tenancy info
Default

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional	
Name	NextWork-VPC	<input type="button" value="Remove tag"/>

You can add 49 more tags.

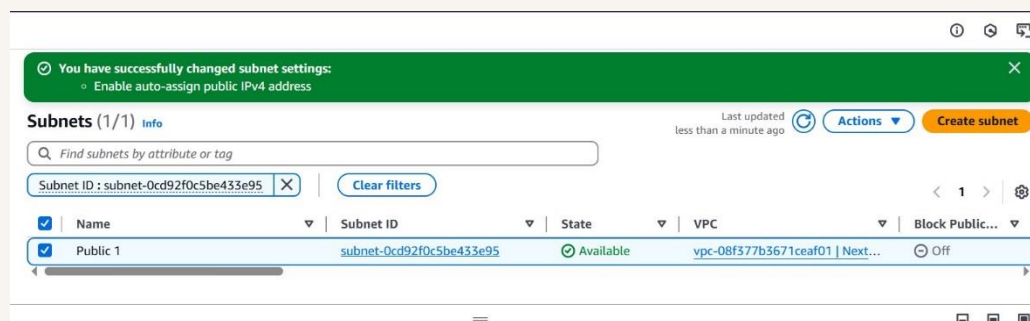


Subnets

Subnets are subdivisions within a VPC that segment your network. There are already subnets existing in my account, one for every Availability Zone in my region. Each has its own CIDR block within a specific AZ.

Once I created my subnet, I enabled auto-assign public IPv4 addresses. This setting ensures EC2 instances launched here automatically receive public IPs to communicate with the internet.

The difference between public and private subnets are their internet access. For a subnet to be considered public, it has to have a route to an internet gateway. Without this route, it remains isolated.



Internet gateways

Internet gateways are essential VPC components that establish connectivity between your VPC and the public internet. They serve dual purposes: functioning as targets for internet-bound traffic in route tables and performing Network Address Translation (NAT) for instances with public IP addresses, enabling bidirectional communication between your VPC resources and the internet.

Attaching an internet gateway to a VPC is a crucial step that enables resources to communicate with the internet, provided they have public IP addresses and appropriate route table configurations. Had I omitted this step in my implementation, my resources would remain unable to reach the internet even if assigned public IP addresses, as the gateway provides the actual connection point to the external network.

