Launching Resources

By Haroon Zaman | December 2025

Introduction

In this project, I aimed to learn how compute resources (EC2 instances) interact with AWS networking components. My goal was to fully understand how public and private EC2 instances work, how they communicate, and how to build a complete network environment using the **“VPC and More”** option in AWS.

To achieve this, I performed three major tasks:

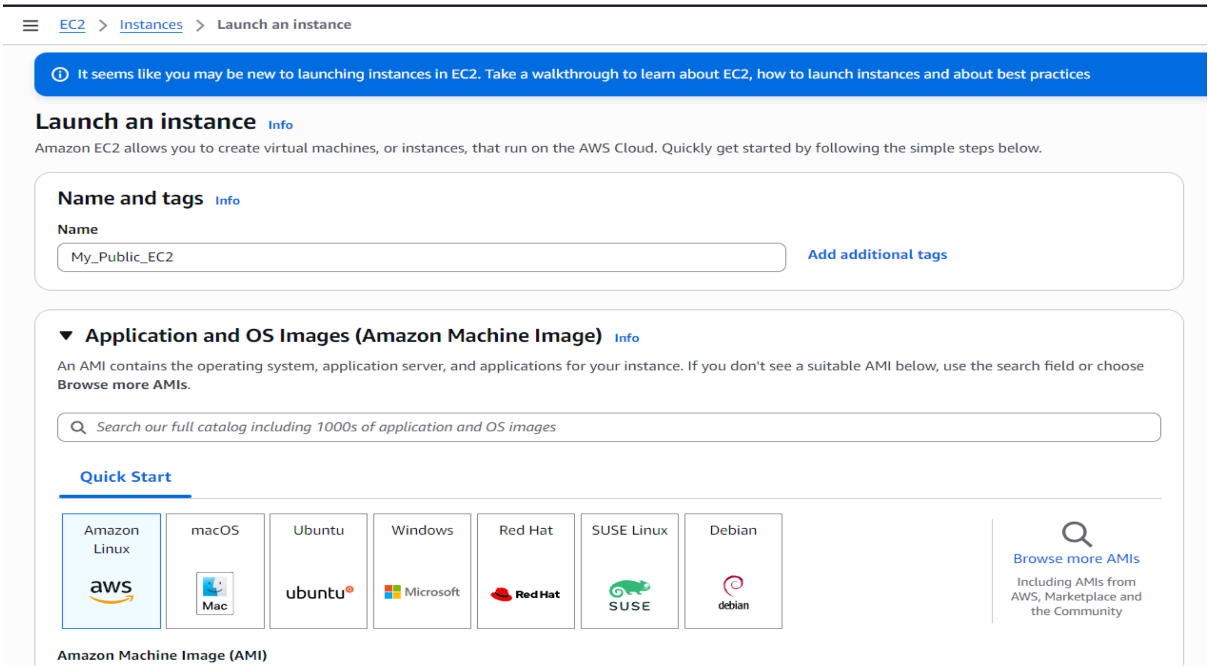
- Created a Public EC2 Instance
- Created a Private EC2 Instance
- Created a VPC Using “VPC and More”

Launching a Public EC2 Instance

A public EC2 instance is a virtual machine that can be accessed from the internet. It sits inside a **public subnet**, gets a **public IP**, and uses a security group that allows incoming connections like SSH or HTTP.

Public EC2s are used when you need direct access from outside AWS—such as testing servers, hosting small applications, or practicing networking.

With this understanding, I proceeded to launch my public EC2 instance.



EC2 > Instances > Launch an instance

ⓘ It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: My_Public_EC2

Add additional tags

▼ **Application and OS Images (Amazon Machine Image)** Info

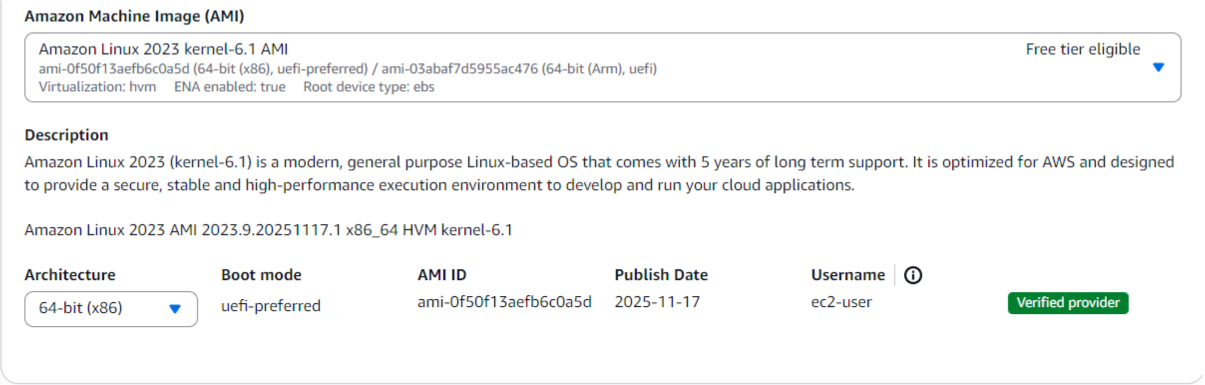
An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Machine Image (AMI)

- First, I searched for **EC2** in the AWS search bar and opened the EC2 console.
- Then I clicked on **Instances** from the left-side panel.
  (I could also access the same page from the right side by clicking **Launch Instance**.)
- After clicking **Launch Instance**, the setup page opened.
- I gave my instance a name: **My_Public_EC2**.
- Under **Amazon Machine Image (AMI)**, I selected **Amazon Linux**.
  There were other operating system options available, but I chose Linux because it is lightweight, fast to boot, and widely used for AWS practice and cloud projects.



Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI
ami-0f50f13aefb6c0a5d (64-bit (x86), uefi-preferred) / ami-03abaf7d5955ac476 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

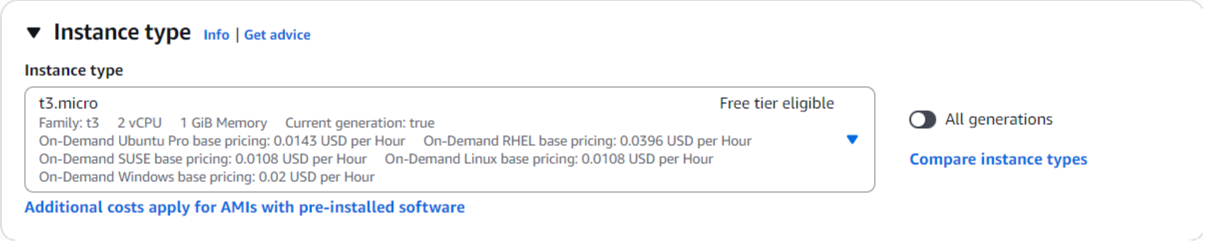
Free tier eligible

Description

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.9.20251117.1 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username
64-bit (x86)	uefi-preferred	ami-0f50f13aefb6c0a5d	2025-11-17	ec2-user



Instance type

t3.micro
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0143 USD per Hour On-Demand RHEL base pricing: 0.0396 USD per Hour
On-Demand SUSE base pricing: 0.0108 USD per Hour On-Demand Linux base pricing: 0.0108 USD per Hour
On-Demand Windows base pricing: 0.02 USD per Hour

Free tier eligible

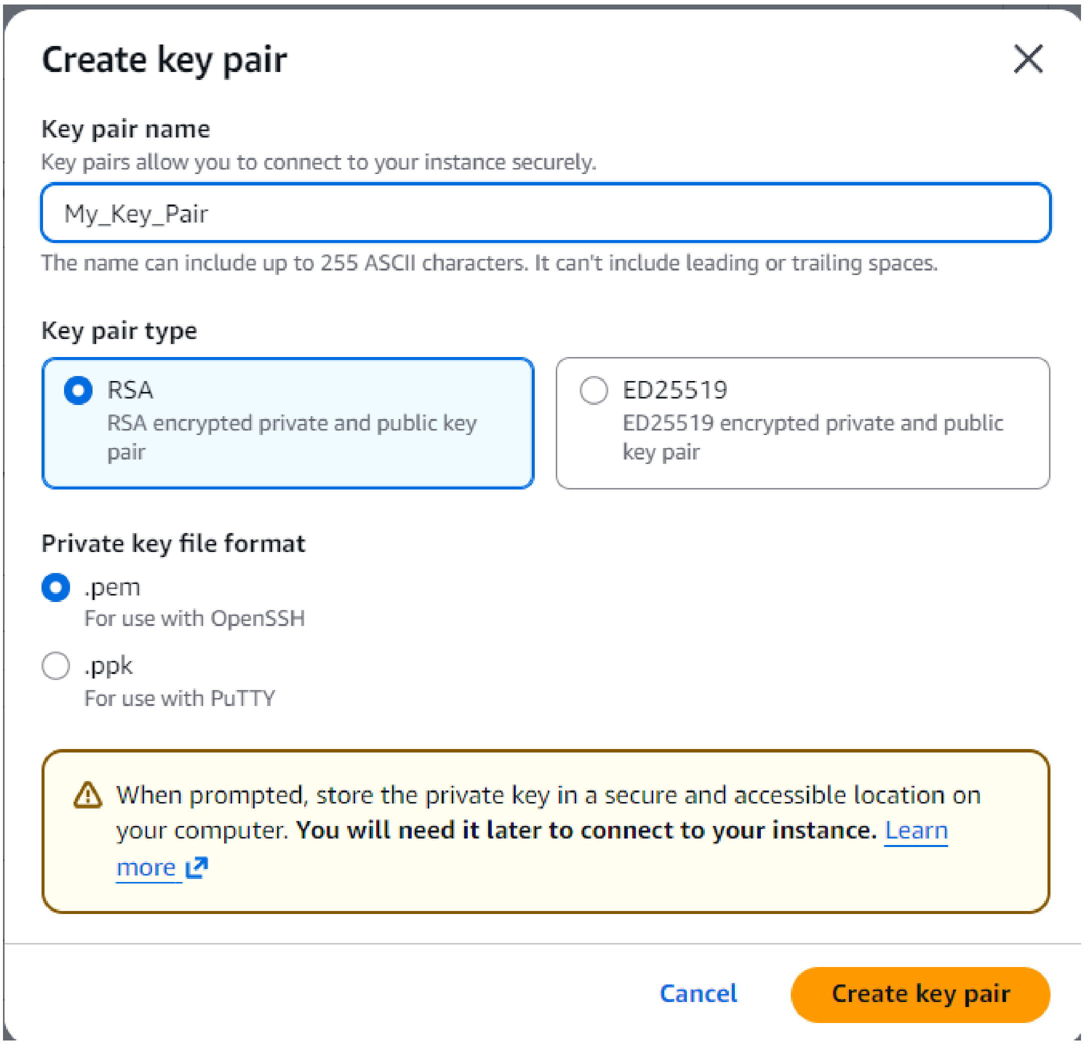
Additional costs apply for AMIs with pre-installed software

- There was an option to select the **architecture**, and I chose **64-bit (x86)**.
  This is the standard architecture used for most servers and applications, ensuring compatibility and good performance.
- Next, I had to choose an **Instance Type**.
  An **instance type** determines the hardware power of your EC2 instance — such as CPU, RAM, and network performance.
  Simply put: *it decides how strong your virtual machine will be.*
- I selected **t3.micro**.
  The **t3** family is a *burstable performance instance*, meaning it provides enough power for general tasks and can temporarily boost performance when needed.
  The **micro** size is the smallest and most cost-efficient, perfect for testing and learning environments.

Key Pair Configuration

- Next was the **Key Pair** section. A **key pair** is a secure login credential used to connect to your EC2 instance.
  Instead of using a password, AWS gives you a private key file that proves *you* are the owner of the instance.
- I could either select an already created key pair or proceed without one, but I chose to create a new key pair.
- I named it **My_Key_Pair**.

- Then I had to choose the key type. There were two options:
    - **RSA** – the most common and widely supported key type; works with almost all SSH clients.



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

My_Key_Pair

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

● RSA
RSA encrypted private and public key pair

○ ED25519
ED25519 encrypted private and public key pair

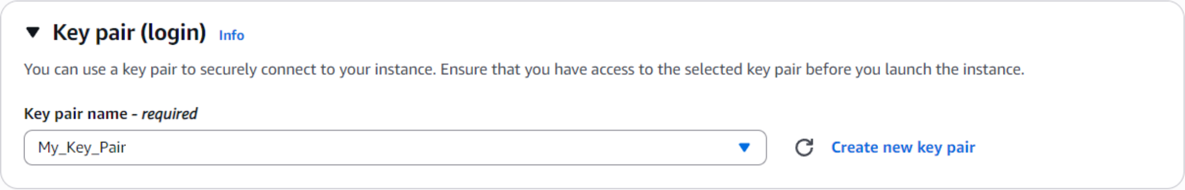
Private key file format

● .pem
For use with OpenSSH

○ .ppk
For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more ↗

-
    - **ED25519** – a newer, faster, and more secure algorithm, but not supported everywhere.
      I selected **RSA** for maximum compatibility.
- Next was choosing the file format. I selected **.pem**.
  The other option was **.ppk**, which is mainly used for Windows with PuTTY.
  Since I preferred the standard Linux/SSH format, I used **.pem**.



▼ **Key pair (login)** Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

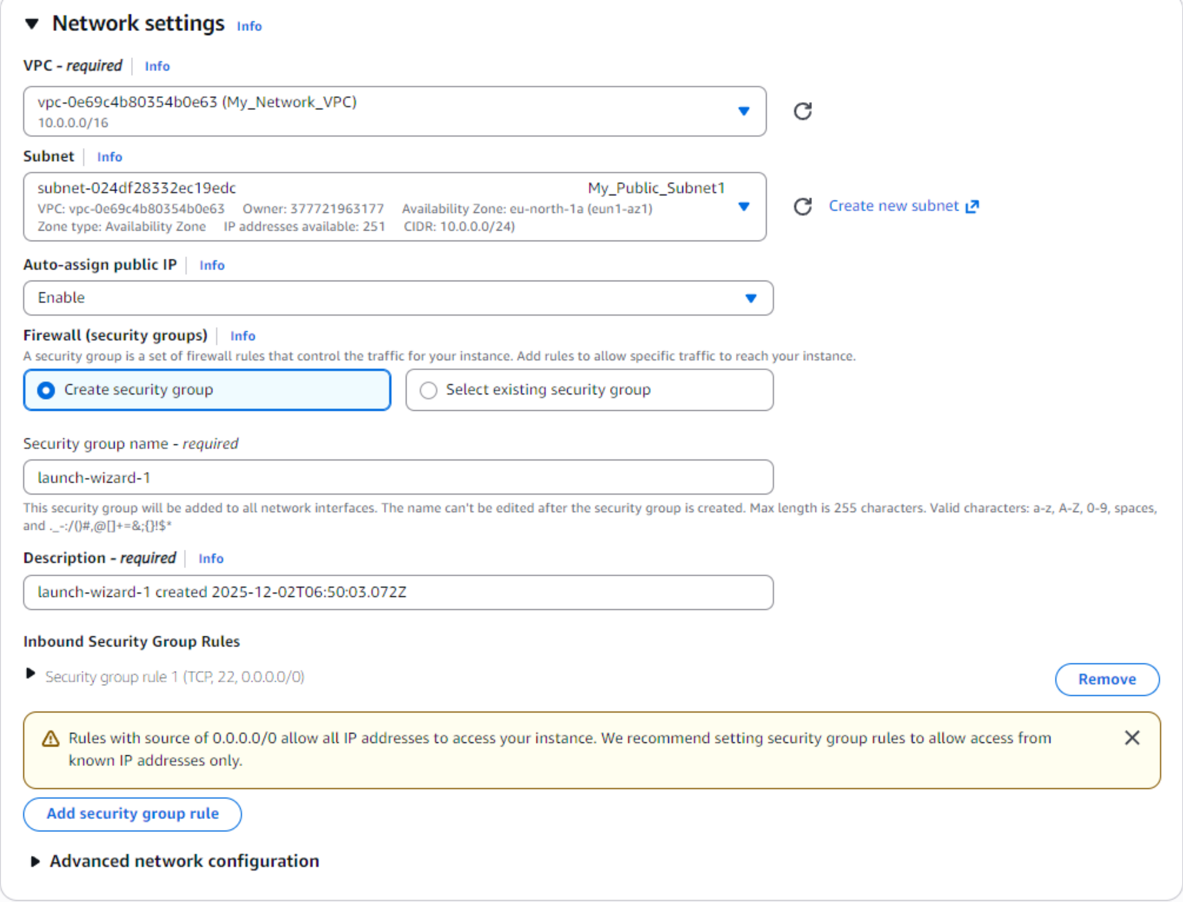
Key pair name - *required*

My_Key_Pair ▼

- After clicking **Create Key Pair**, a private key file was automatically downloaded to my computer. This file is essential for connecting to my EC2 instance later.

Network Settings

- Next was the Network Settings section. Here I had to configure how my EC2 instance would connect within my VPC.
- First, I selected the VPC I created earlier called **My_Network_VPC**. This ensures the instance launches inside my own custom network instead of the default VPC.
- Then I selected the subnet **My_Public_Subnet** because this EC2 instance is meant to be publicly accessible from the internet.



▼ Network settings

VPC - *required* |

vpc-0e69c4b80354b0e63 (My_Network_VPC)
10.0.0.0/16

Subnet |

subnet-024df28332ec19edc My_Public_Subnet1
VPC: vpc-0e69c4b80354b0e63 Owner: 377721963177 Availability Zone: eu-north-1a (eun1-az1)
Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.0.0/24)

Auto-assign public IP |

Enable

Firewall (security groups) |

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

Security group name - *required*

launch-wizard-1

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* |

launch-wizard-1 created 2025-12-02T06:50:03.072Z

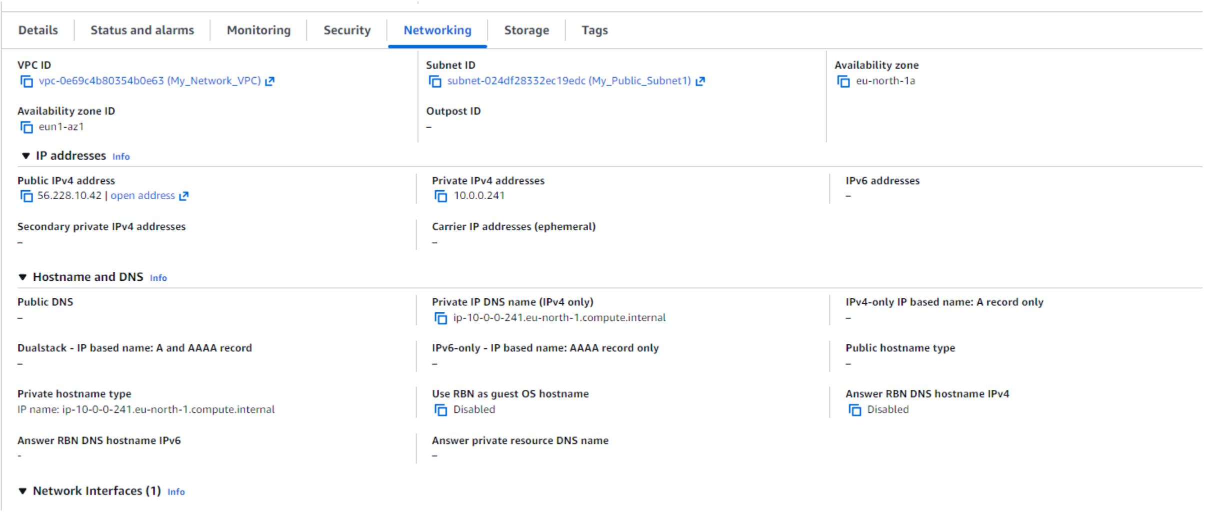
Inbound Security Group Rules

▶ Security group rule 1 (TCP, 22, 0.0.0.0/0)

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

▶ Advanced network configuration

- After that, I enabled **Auto-assign Public IP**. This setting tells AWS to automatically give my instance a public IP address so it can be accessed from outside the VPC. Without this, the instance would only have a private IP.
- Next was the Firewall (Security Group) section. Here, I selected my previously created security group called **My_Public_Security_Group**. This security group acts like a virtual firewall that controls what inbound and outbound traffic is allowed for the instance.
- I left the rest of the settings as default, and after reviewing everything, I clicked **Launch Instance** to create my EC2.
- After the instance was created, I opened the networking panel to review the details. Here I could clearly see all the network-related settings that were automatically applied or selected during the launch process.
- The panel showed the **VPC** as **My_Network_VPC**, confirming that the instance was launched inside the correct custom network.
- The **Subnet** was listed as **My_Public_Subnet**, which matched my intention of placing this instance in the public section of my VPC.



VPC ID
vpc-0e69c4b80354b0e63 (My_Network_VPC)

Subnet ID
subnet-024df28332ec19edc (My_Public_Subnet1)

Availability zone
eu-north-1a

Availability zone ID
eun1-az1

Outpost ID
–

▼ IP addresses Info

Public IPv4 address
56.228.10.42 | open address

Private IPv4 addresses
10.0.0.241

IPv6 addresses
–

Secondary private IPv4 addresses
–

Carrier IP addresses (ephemeral)
–

▼ Hostname and DNS Info

Public DNS
–

Private IP DNS name (IPv4 only)
ip-10-0-0-241.eu-north-1.compute.internal

IPv4-only IP based name: A record only
–

Dualstack - IP based name: A and AAAA record
–

IPv6-only - IP based name: AAAA record only
–

Public hostname type
–

Private hostname type
IP name: ip-10-0-0-241.eu-north-1.compute.internal

Use RBN as guest OS hostname
Disabled

Answer RBN DNS hostname IPv4
Disabled

Answer RBN DNS hostname IPv6
-

Answer private resource DNS name
–

▼ Network Interfaces (1) Info

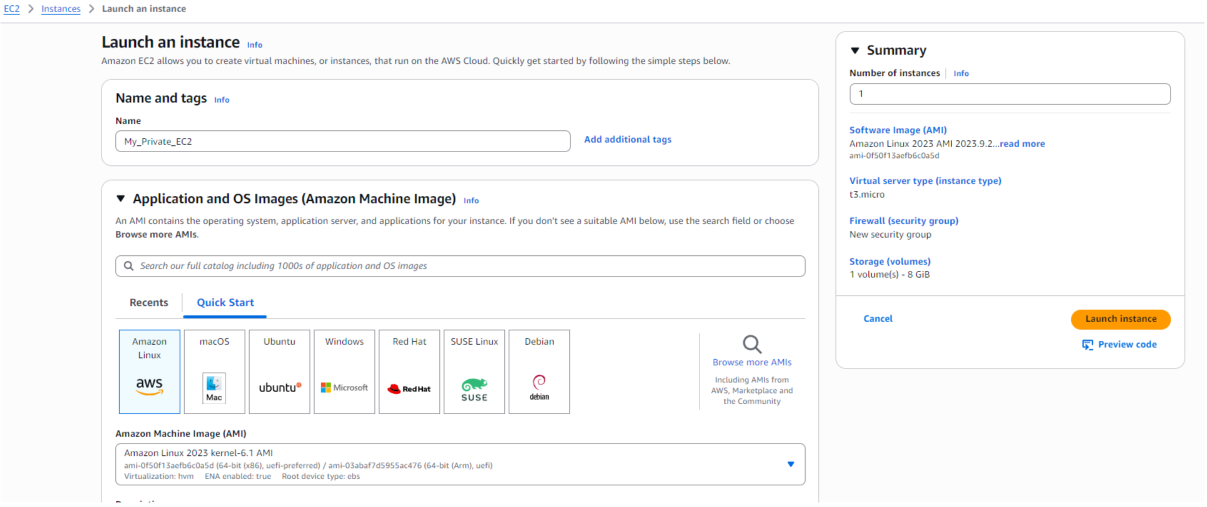
- I could also see that **Auto-assign Public IP** was enabled, and a public IP address had been assigned to the instance. This means the instance is reachable from the internet (as long as the security group allows access).

Launching a Private EC2 Instance

A private EC2 instance is a virtual machine that **cannot be accessed directly from the internet**. It is placed inside a **private subnet** and only has a private IP, which allows communication within the VPC.

Private EC2s are used for **backend services, databases, or any resources that need to stay secure and isolated** from public access. They rely on routing, NAT gateways, or other public instances if they need temporary internet access.

With this understanding, I proceeded to launch my private EC2 instance inside my private subnet.



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name
My_Private_EC2

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose Browse more AMIs.

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)
Amazon Linux 2023 kernel-6.1 AMI
ami-0f50f13aefb6c0a5d (64-bit (x86), uefi-preferred) / ami-03abaf7d5955ac476 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

▼ Summary

Number of instances | Info
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2...read more
ami-0f50f13aefb6c0a5d

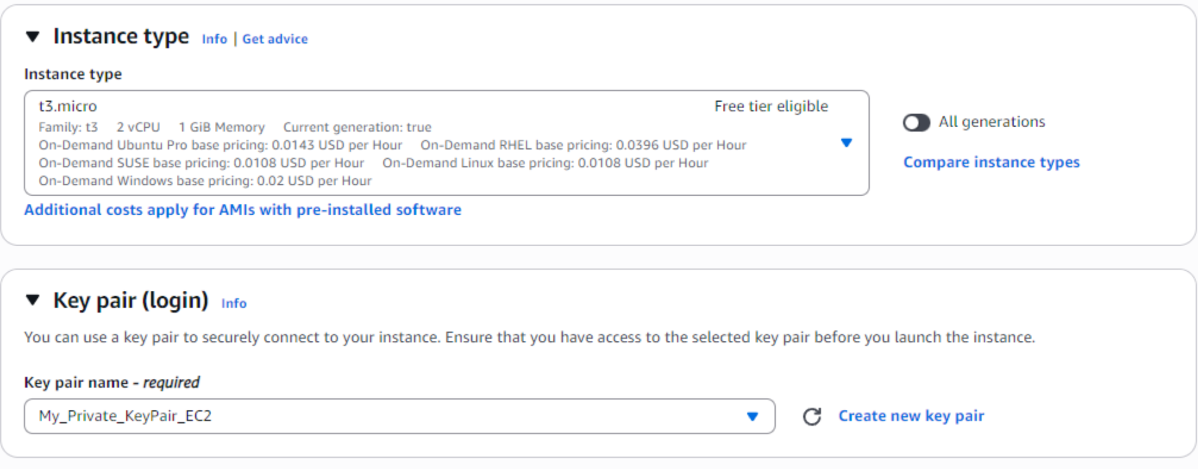
Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

- First, I clicked on **Launch Instance** to start creating my private resource.
- I gave the instance a name: **My_Private_EC2**.

- Next, I selected the same operating system as before: **Linux (Amazon)**.
  I chose Linux because it is lightweight, widely supported, and ideal for learning and testing in AWS.



▼ **Instance type** Info | Get advice

Instance type

t3.micro — Free tier eligible
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0143 USD per Hour On-Demand RHEL base pricing: 0.0396 USD per Hour
On-Demand SUSE base pricing: 0.0108 USD per Hour On-Demand Linux base pricing: 0.0108 USD per Hour
On-Demand Windows base pricing: 0.02 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ **Key pair (login)** Info

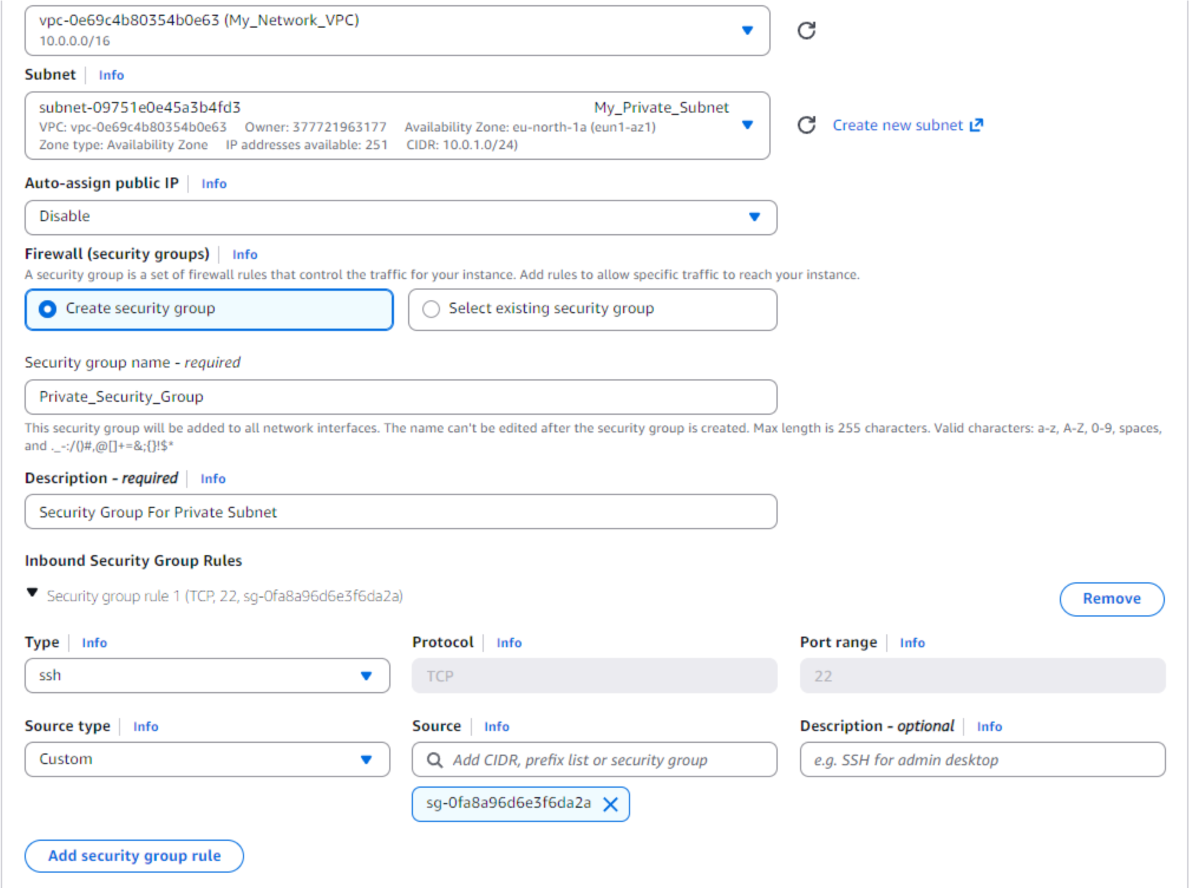
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

My_Private_KeyPair_EC2

Create new key pair

- I chose the **same instance type** as my public EC2, which is **t3.micro**, to keep it consistent and lightweight for testing purposes.
- Next, I created a **new key pair** called **My_Private_Keypair_EC2**.
  This key pair is essential to securely connect to the private instance later, just like with the public EC2.



vpc-0e69c4b80354b0e63 (My_Network_VPC)
10.0.0.0/16

Subnet | Info

subnet-09751e0e45a3b4fd3 — My_Private_Subnet
VPC: vpc-0e69c4b80354b0e63 Owner: 377721963177 Availability Zone: eu-north-1a (eun1-az1)
Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.1.0/24)

Create new subnet

Auto-assign public IP | Info

Disable

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

Security group name - *required*

Private_Security_Group

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* | Info

Security Group For Private Subnet

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, sg-0fa8a96d6e3f6da2a)

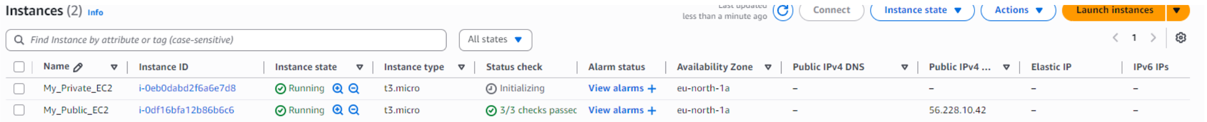
Remove

Type | Info	**Protocol** | Info	**Port range** | Info
ssh	TCP	22
Source type | Info	**Source** | Info	**Description - *optional*** | Info
Custom	Add CIDR, prefix list or security group sg-0fa8a96d6e3f6da2a	e.g. SSH for admin desktop

Add security group rule

- Next, I configured the networking settings for my private EC2 instance.

- I selected the same VPC I created earlier, **My_Network_VPC**, to ensure it is part of my custom network.
- For the subnet, I chose **My_Private_Subnet**, which is designed to be isolated from the internet.
- I **disabled Auto-assign Public IP** because this instance should remain private and not be directly reachable from the internet.
- In the Firewall (Security Group) section, I created a new security group called **Private_Security_Group**.
  I also added a description indicating it is for the private subnet.
- For the **Inbound rule**, I selected **SSH** as the type, set the source type to **Custom**, and chose my **Public_Security_Group** as the source.
  This ensures that only resources that are part of the public security group—like my public EC2 instance—can communicate with this private EC2.
  In other words, it allows internal communication between specific resources while keeping the instance isolated from external access.



Instances (2) Info

Last updated less than a minute ago

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs
□	My_Private_EC2	i-0eb0dabd2f6a6e7d8	Running	t3.micro	Initializing	View alarms +	eu-north-1a	–	–	–	–
□	My_Public_EC2	i-0df16bfa12b86b6c6	Running	t3.micro	3/3 checks passec	View alarms +	eu-north-1a	–	56.228.10.42	–	–

- After configuring the network settings and security group, I left all the remaining options at their default values.
- Once I reviewed everything to ensure it was correct, I clicked **Launch Instance** to create my private EC2.

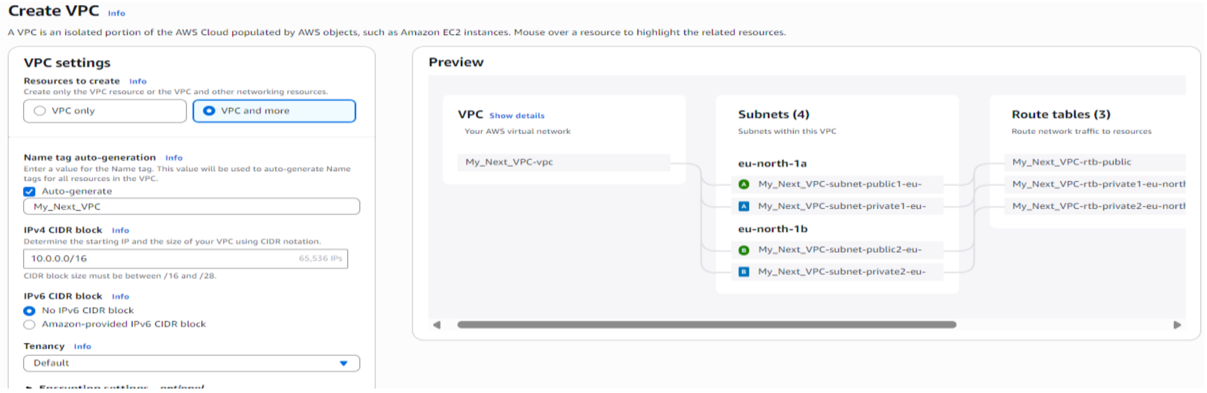
Creating a VPC Using "VPC and More"

The **"VPC and More"** option in AWS is a shortcut that lets you create an entire network setup in one go.

Instead of manually creating a VPC, subnets, route tables, internet gateways, and NAT gateways one by one, this option does it all automatically for you.

It's useful when you want a **complete, production-ready network** quickly, with public and private subnets, proper routing, and internet access already configured.

Using this option saves time and ensures that all the networking components are correctly connected and ready to use.



Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances. Mouse over a resource to highlight the related resources.

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 auto-generation Info
Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.
☑ Auto-generate
My_Next_VPC

IPv4 CIDR block Info
Determine the starting IP and the size of your VPC using CIDR notation.
10.0.0.0/16 65,536 IPs
CIDR block size must be between /16 and /28.

IPv6 CIDR block Info
● No IPv6 CIDR block
○ Amazon-provided IPv6 CIDR block

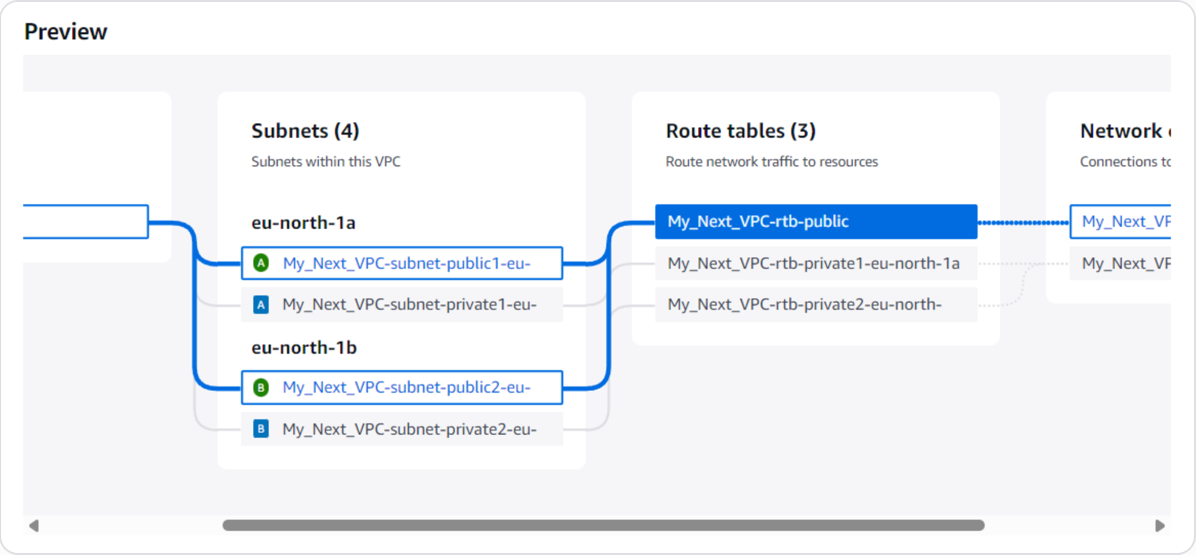
Tenancy Info
Default

Preview

VPC Show details
Your AWS virtual network
My_Next_VPC-vpc

Subnets (4)
Subnets within this VPC
eu-north-1a
My_Next_VPC-subnet-public1-eu-
My_Next_VPC-subnet-private1-eu-
eu-north-1b
My_Next_VPC-subnet-public2-eu-
My_Next_VPC-subnet-private2-eu-

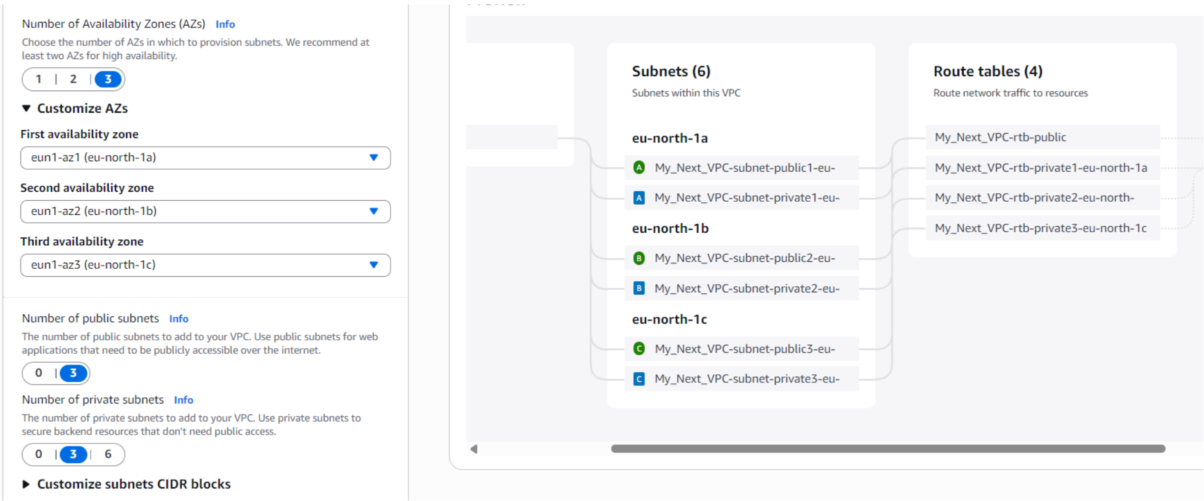
Route tables (3)
Route network traffic to resources
My_Next_VPC-rtb-public
My_Next_VPC-rtb-private1-eu-nortl
My_Next_VPC-rtb-private2-eu-nortl


Preview
Subnets (4)
Subnets within this VPC
eu-north-1a
A My_Next_VPC-subnet-public1-eu-
A My_Next_VPC-subnet-private1-eu-
eu-north-1b
B My_Next_VPC-subnet-public2-eu-
B My_Next_VPC-subnet-private2-eu-
Route tables (3)
Route network traffic to resources
My_Next_VPC-rtb-public
My_Next_VPC-rtb-private1-eu-north-1a
My_Next_VPC-rtb-private2-eu-north-
Network
Connections tc
My_Next_VF
My_Next_VF

In the preview window on the right side, I noticed that when I moved the cursor over different components, it **highlighted the relationships between the network elements**.

This made it easy to see how the VPC, subnets, route tables, and internet gateways were connected.

It helps to **visualize the network layout** before creating it, ensuring that everything is properly linked and organized.


Number of Availability Zones (AZs) Info

Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.

1 | 2 | **3**

▼ Customize AZs

First availability zone: eun1-az1 (eu-north-1a) ▼

Second availability zone: eun1-az2 (eu-north-1b) ▼

Third availability zone: eun1-az3 (eu-north-1c) ▼

Number of public subnets Info

The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.

0 | **3**

Number of private subnets Info

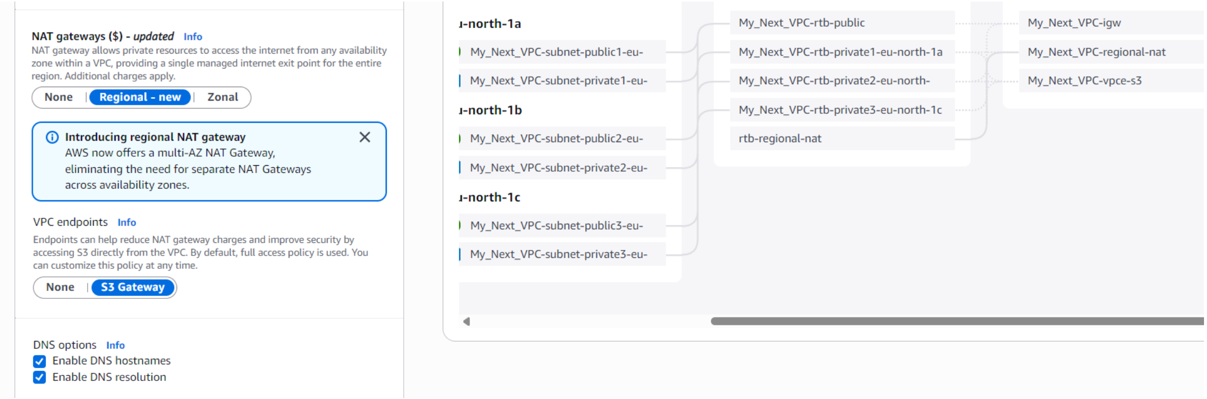
The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.

0 | **3** | 6

▶ Customize subnets CIDR blocks

Subnets (6)
Subnets within this VPC
eu-north-1a
A My_Next_VPC-subnet-public1-eu-
A My_Next_VPC-subnet-private1-eu-
eu-north-1b
B My_Next_VPC-subnet-public2-eu-
B My_Next_VPC-subnet-private2-eu-
eu-north-1c
C My_Next_VPC-subnet-public3-eu-
C My_Next_VPC-subnet-private3-eu-
Route tables (4)
Route network traffic to resources
My_Next_VPC-rtb-public
My_Next_VPC-rtb-private1-eu-north-1a
My_Next_VPC-rtb-private2-eu-north-
My_Next_VPC-rtb-private3-eu-north-1c

- Any changes I made on the left side were reflected immediately on the right-side preview.
- For example, choosing **Availability Zones**, selecting the **number of public subnets**, or even changing the **name of the VPC** updated the diagram in real time.
- This feature makes it easy to **see the effect of each setting** and ensures that the network layout matches my intended design before creating the VPC.



NAT gateways ($) - *updated* Info

NAT gateway allows private resources to access the internet from any availability zone within a VPC, providing a single managed internet exit point for the entire region. Additional charges apply.

None | [✓ Regional - new] | Zonal

Introducing regional NAT gateway
AWS now offers a multi-AZ NAT Gateway, eliminating the need for separate NAT Gateways across availability zones.

VPC endpoints Info

Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.

None | [✓ S3 Gateway]

DNS options Info

☑ Enable DNS hostnames
☑ Enable DNS resolution

ı-north-1a
My_Next_VPC-subnet-public1-eu-
My_Next_VPC-subnet-private1-eu-
ı-north-1b
My_Next_VPC-subnet-public2-eu-
My_Next_VPC-subnet-private2-eu-
ı-north-1c
My_Next_VPC-subnet-public3-eu-
My_Next_VPC-subnet-private3-eu-
My_Next_VPC-rtb-public
My_Next_VPC-rtb-private1-eu-north-1a
My_Next_VPC-rtb-private2-eu-north-
My_Next_VPC-rtb-private3-eu-north-1c
rtb-regional-nat
My_Next_VPC-igw
My_Next_VPC-regional-nat
My_Next_VPC-vpce-s3

While configuring the VPC, I noticed several optional settings:

- **NAT Gateway** – This allows private subnets to access the internet. Options include:
    - **None** – No NAT, private instances cannot reach the internet.
    - **Regional** – A single NAT gateway for the entire region.
    - **Zonal** – One NAT gateway per Availability Zone.
- **VPC Endpoints** – These let your VPC privately connect to AWS services without going through the internet. Options include:
    - **None** – No private connection.
    - **S3 Gateway** – Provides a private route to Amazon S3 directly from your VPC.
- **DNS Settings** – You can enable or disable **DNS Hostnames** and **DNS Resolution**. Enabling these allows instances in the VPC to resolve domain names and access AWS services using hostnames.

These options give more control over **internet access, private connectivity, and name resolution** within the VPC.

Conclusion

Through this project, I learned how to create and configure EC2 instances both publicly and privately, as well as how to build a complete VPC using the “VPC and More” option.

I now understand how subnets, route tables, security groups, and NAT gateways work together to control network traffic and access.

This hands-on experience strengthened my knowledge of AWS networking, showing me how to design secure and functional cloud environments, manage internal and external communication, and visualize complex network layouts effectively.