

# Simple Web Deployment on AWS EC2 Project

Step-by-step guide deploy an index.html page using nginx

## Step 1 : Create VPC

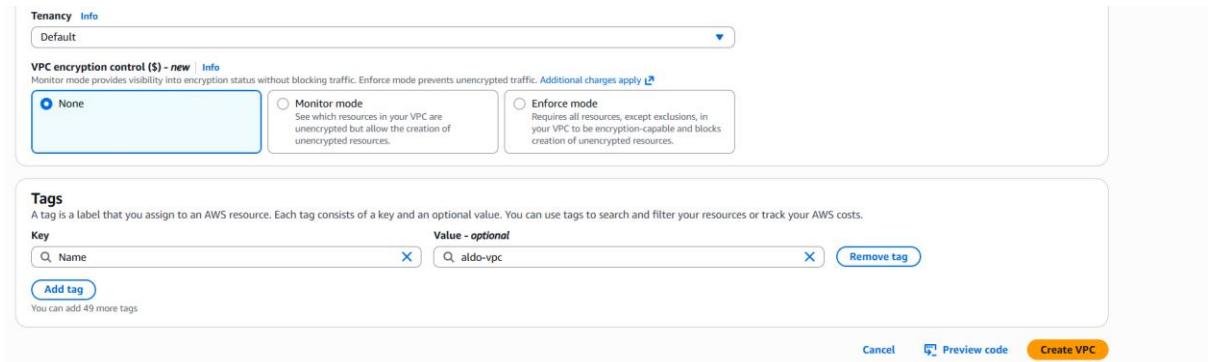
1. Head over to the VPC console
2. Click Create VPC

The screenshot shows the AWS VPC dashboard. On the left, there's a sidebar with navigation links for VPC dashboard, AWS Global View, Virtual private cloud (with options for Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, and Route servers), Security (Network ACLs, Security groups), and Private link and interface endpoints. The main area displays 'Resources by Region' for the Oregon region. It includes sections for VPCs (1), Subnets (4), Route Tables (1), Internet Gateways (1), NAT Gateways (0), VPC Peering Connections (0), Network ACLs (1), Security Groups (1), and Customer Gateways (0). Each section has a 'See all regions' link. On the right, there are boxes for Service Health (View complete service health details), Settings (Block Public Access, Zones, Console Experiments), Additional Information (VPC Documentation, All VPC Resources, Forums, Report an Issue), and AWS Network Manager (providing tools and features to help manage and monitor your network on AWS).

3. In VPC setting
  1. Chose VPC only
  2. In Name tag give your VPC unique name like “aldo-vpc”
  3. In IPv4 CIDR Block , Chose IPv4 Manual Input
  4. And in Column IPv4 CIDR input 10.0.0.0/16
  5. In IPv6 CIDR Block , Chose No IPv6 CIDR Block

The screenshot shows the 'Create VPC' configuration page. At the top, it says 'VPC > Your VPCs > Create VPC'. Below that is a 'Create VPC' button with an 'Info' link. A note states: 'A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.' Under 'VPC settings', there's a 'Resources to create' section with 'Info' and a note: 'Create only the VPC resource or the VPC and other networking resources.' Two radio buttons are shown: 'VPC only' (selected) and 'VPC and more'. The 'Name tag - optional' field contains 'aldo-vpc'. Under 'IPv4 CIDR block', there's an 'Info' link and two radio buttons: 'IPv4 CIDR manual input' (selected) and 'IPAM-allocated IPv4 CIDR block'. The 'IPv4 CIDR' field contains '10.0.0.0/16'. Under 'IPv6 CIDR block', there's an 'Info' link and three radio buttons: 'No IPv6 CIDR block' (selected), 'IPAM-allocated IPv6 CIDR block', 'Amazon-provided IPv6 CIDR block', and 'IPv6 CIDR owned by me'.

4. Then Scroll to the bottom and leave all default
5. The Click Create VPC
6. And you Successfully creat a VPC



## Step 2 : Create a Subnet

1. In VPC Dashboard you Click Subnets
2. And then Click Create Subnet

3. On the page create VPC , chose a column VPC ID and chose you VPC name the one that has been made earlier

VPC > Subnets > Create subnet

**Create subnet** Info

**VPC**

**VPC ID**  
Create subnets in this VPC.

Select a VPC

Q |  
vpc-0bf71d4a3c5a6bd1c (172.31.0.0/16)  
vpc-0dc49f44fc193681 (aldo-vpc) (10.0.0.0/16)

Select a VPC first to create new subnets.

Add new subnet

Cancel Create subnet

#### 4. In Subnets Setting

1. Create You subnet name in here iam using “Public Subnet”
2. Leave default Availability zone
3. And In column IPv4 Subnet CIDR Block input 10.0.0.0/24

**Subnet settings**  
Specify the CIDR blocks and Availability Zone for the subnet.

**Subnet 1 of 1**

**Subnet name**  
Create a tag with a key of 'Name' and a value that you specify.  
public subnet  
The name can be up to 256 characters long.

**Availability Zone** Info  
Choose the zone in which your subnet will reside, or let Amazon choose one for you.  
No preference

**IPv4 VPC CIDR block** Info  
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.  
10.0.0.0/16

**IPv4 subnet CIDR block**  
10.0.0.0/24  
256 IPs

#### 5. Scroll to the bottom and click create subnet

#### 6. Then you already create a subnet

**▼ Tags - optional**

Key	Value - optional
Name	public subnet

Add new tag  
Remove  
Add new subnet

Cancel Create subnet

7. Then you have to edit subnet setting
8. Click check box in your subnet then click Action
9. And chose Edit subnet settings

You have successfully created 1 subnet: subnet-0863b4561811c7951

**Subnets (1/1) Info**

Name	Subnet ID	State	VPC	Block
public subnet	subnet-0863b4561811c7951	Available	vpc-0dc49f44fca193681   aldo-vpc	Off

**subnet-0863b4561811c7951 / public subnet**

**Details**

Subnet ID subnet-0863b4561811c7951	Subnet ARN arn:aws:ec2:us-west-2:658059416348:subnet/subnet-0863b4561811c7951	State Available
IPv4 CIDR 10.0.0.0/24	Available IPv4 addresses 251	IPv6 CIDR -
Availability Zone usw2-az1 (us-west-2b)	Network border group -	VPC vpc-0dc49f44fca193681   aldo-vpc

**Block Public Access**  
Off

**IPv6 CIDR association ID**  
-

**Route table**  
-

10. On Page subnet settings you will see Auto-assign IP setting and Resource-base name (RBN) settings , chose check box enable auto-assign and enable resource name

11. Then click save

**Edit subnet settings**

**Subnet**

Subnet ID subnet-0863b4561811c7951	Name public subnet
---------------------------------------	-----------------------

**Auto-assign IP settings**

Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

Enable auto-assign public IPv4 address

Enable auto-assign customer-owned IPv4 address

**Resource-based name (RBN) settings**

Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

Enable resource name DNS A record on launch

Enable resource name DNS AAAA record on launch

**Hostname type**

Resource name

IP name

**DNS64 settings**

Enable DNS64 to allow IPv6-only services in Amazon VPC to communicate with IPv4-only services and networks.

Enable DNS64

**Save**

You have successfully changed subnet settings:

- Enable auto-assign public IPv4 address
- Enable resource name DNS A record on launch

**Subnets (1/1) Info**

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
public subnet	subnet-0863b4561811c7951	Available	vpc-0dc49f44fca193681   aldo-vpc	Off	10.0.0.0/24

**subnet-0863b4561811c7951 / public subnet**

**Details**

Subnet ID subnet-0863b4561811c7951	Subnet ARN arn:aws:ec2:us-west-2:658059416348:subnet/subnet-0863b4561811c7951	State Available
IPv4 CIDR 10.0.0.0/24	IPv6 CIDR -	Block Public Access Off

**IPv6 CIDR association ID**  
-

## Step 3 : Create Internet Gateway

1. In VPC Dashboard you Click Internet gateways
2. Click create internet gateways

The screenshot shows the AWS VPC Internet Gateways page. On the left, there's a navigation sidebar with 'Virtual private cloud' and 'Internet gateways' selected. The main area displays a table titled 'Internet gateways (1) Info'. The table has columns for Name, Internet gateway ID, State, VPC ID, and Owner. One row is listed: 'igw-02f0815b175cf5918' with 'Attached' state, 'vpc-0bf71d4a3c5a6bd1c' VPC ID, and '658059416348' owner. Below the table, a message says 'Select an internet gateway above'.

3. On Page create internet gateways
4. In column name tag give your name internet gateways , in here i using "igw"
5. Leave all default
6. Click create internet gateways

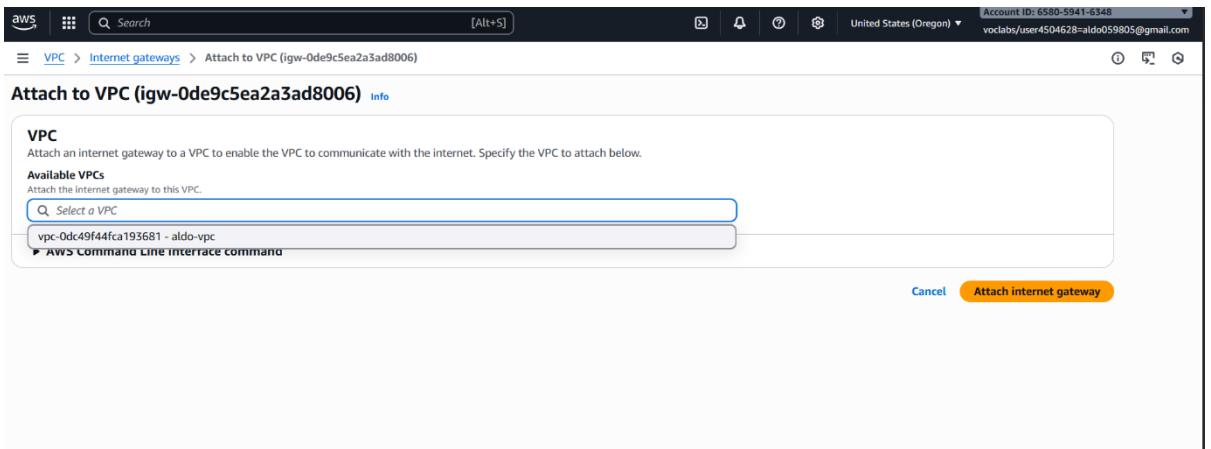
The screenshot shows the 'Create internet gateway' wizard. Step 1: Internet gateway settings. It asks for a 'Name tag' which is 'igw'. Below it, there's a section for 'Tags - optional' where a single tag 'igw' is added with key 'Name' and value 'igw'. At the bottom right are 'Cancel' and 'Create internet gateway' buttons.

7. After create internet gateway , we have attach internet gateways to VPC
8. On page internet gateways you click action and chose attach to VPC

The screenshot shows the AWS VPC Internet Gateways page again. The newly created gateway 'igw-0de9c5ea2a3ad8006 / igw' is listed with 'Attached' status and 'vpc-0bf71d4a3c5a6bd1c' VPC ID. A green message at the top says 'The following internet gateway was created: igw-0de9c5ea2a3ad8006 - igw. You can now attach to a VPC to enable the VPC to communicate with the internet.' On the right, there's an 'Actions' menu with options like 'Attach to VPC', 'Detach from VPC', 'Manage tags', and 'Delete'.

9. On page attach to vpc in available VPCs chose your VPC

## 10. An then click attach internet gateways



## Step 4 : Create Route Tables

1. In VPC dashboard click Route tables
2. You will see there only have 1 route tables
3. Then you click refresh button next to action button
4. Then you will see there now have 2 route tables

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
-	rtb-01d19288e4e4678dd	-	-	-	vpc-0bf71d4a3c5a6bd1c

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
-	rtb-01d19288e4e4678dd	-	-	-	vpc-0bf71d4a3c5a6bd1c
-	rtb-0efc140f48774f67c	-	-	-	vpc-0dc49f44fca193681   aldo-

5. In route tables column you can see at VPC column there is a vpc name you made before
6. In name column you can give a name to your route table

- Click name column and give your route table name , example “aldo-rtb” and then save

The screenshot shows the AWS VPC Route Tables page. On the left, there's a sidebar with 'Virtual private cloud' and 'Route tables' selected. The main area displays two route tables: 'rtb-01d19288e4e4678dd' and 'rtb-0efc140f48774f67c'. The second route table has its name being edited from 'rtb-0efc140f48774f67c' to 'aldo-rtb'. A modal dialog box is open with the 'Edit Name' field containing 'aldo-rtb' and a 'Save' button.

- After that you have setting routes and subnet association
- You can see in the page detail page there is a routes page
- Click route page and then click edit routes

The screenshot shows the 'Routes' tab for the 'aldo-rtb' route table. It lists a single route: 'Destination' 10.0.0.0/16, 'Target' local, 'Status' Active, 'Propagated' No, and 'Route Origin' CreateRouteTable. There is a 'Edit routes' button at the top right of the routes table.

- On page edit routes you click add route
- In destination chose 0.0.0.0/0
- And target column chose internet gateways and then chose your internet gateway which has been made before
- And click save changes

The screenshot shows the 'Edit routes' dialog for the 'rtb-0efc140f48774f67c' route table. It lists a route with 'Destination' 10.0.0.0/16, 'Target' local, 'Status' Active, 'Propagated' No, and 'Route Origin' CreateRouteTable. A new route is being added with 'Destination' 0.0.0.0/0, 'Target' Internet Gateway, 'Status' Active, 'Propagated' No, and 'Route Origin' CreateRoute. The 'Add route' button is highlighted.

- Then we go to page Subnet associations
- Click edit subnet associations

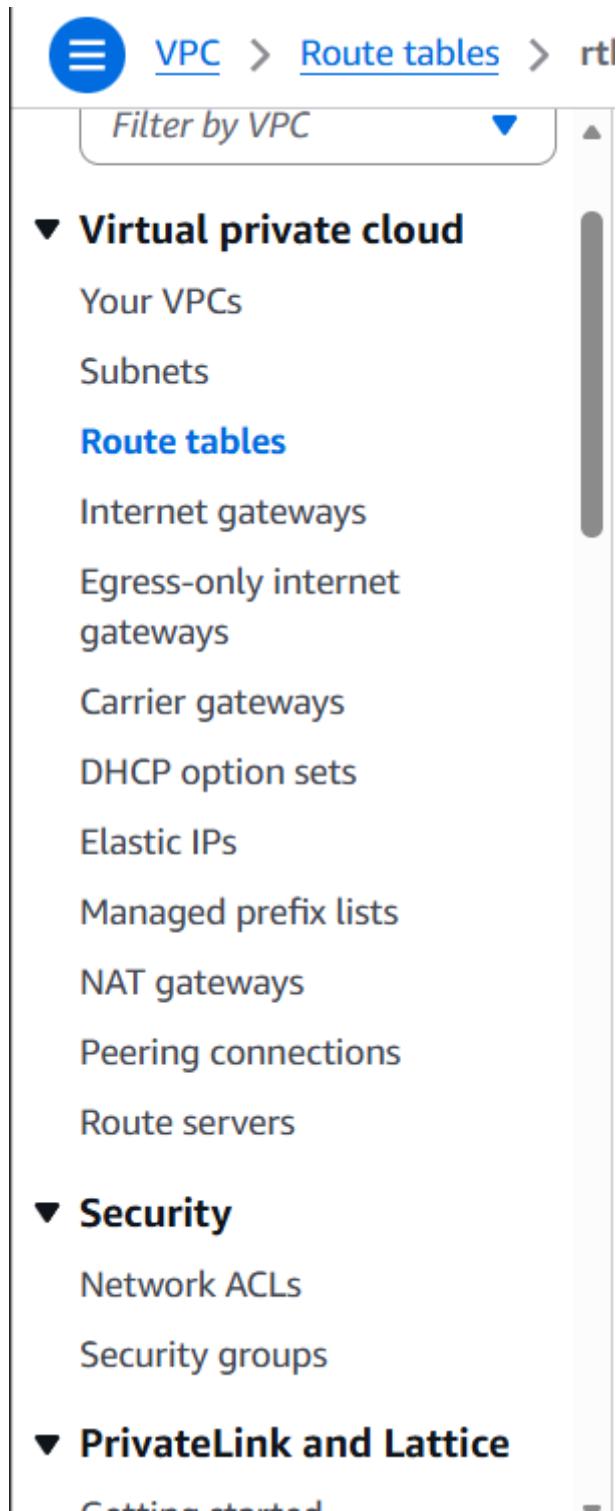
17. On page edit subnet associations you will see you subnet which has been made before

18. Click check box

19. And click save associations

## Step 5 : Create Security Group

1. In AWS console you scroll down until you find Security , and then click security groups



2. On page security group , click create security group

The screenshot shows the AWS VPC > Security Groups interface. On the left sidebar, there are several navigation options: Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers, Security (Network ACLs, Security groups), PrivateLink and Lattice (Getting started, Endpoints). The main content area displays a table titled "Security Groups (2) Info" with columns: Name, Security group ID, Security group name, VPC ID, and Description. Two rows are listed: one for "sg-099348fff85bbe4bf" and another for "sg-046d88b56c64f27cd", both under the "default" security group name and associated with the "vpc-0bf71d4a3c5a6bd1c" VPC ID. A search bar at the top allows finding security groups by attribute or tag. Action buttons include "Actions", "Export security groups to CSV", and a prominent orange "Create security group" button.

3. On page create security group
4. In basic details
5. Create you security group name , example “aldoSG”
6. Description you can empty this or you can write Allow SSH
7. In vpc click and then chose you vpc name

**Create security group** Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

**Security group name** Info  
aldoSG  
Name cannot be edited after creation.

**Description** Info  
Allow ssh

**VPC info**  
vpc-0bf71d4a3c5a6bd1c

**Inbound rules**

**Add rule**

8. In inbound rules click add rules
9. In type chose http and source chose anywhere IPv4
10. In type chose SSH and source chose anywhere IPv4
11. In type chose https and source chose anywhere IPv4

Type	Protocol	Port range	Source	Description - optional	Action
HTTP	TCP	80	Anyw...	0.0.0.0/0 X	Delete
SSH	TCP	22	Anyw...	0.0.0.0/0 X	Delete
HTTPS	TCP	443	Anyw...	0.0.0.0/0 X	Delete

**Add rule**

12. And leave outbound rules default and then click create security group

**Outbound rules**

Type: All traffic | Protocol: All | Port range: All | Destination: Custom | Description - optional: 0.0.0.0/0

**Tags - optional**

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.

No tags associated with the resource.

**Details**

Security group name: aldoSG | Security group ID: sg-0f64eca6b35553612 | Owner: 658059416348 | Description: Allow ssh | VPC ID: vpc-0dc49f44fc193681

**Inbound rules (3)**

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-07b59c859d1b2c77a	IPv4	HTTPS	TCP	443
-	sgr-05137ec1d190593cf	IPv4	HTTP	TCP	80
-	sqr-052a8466bf9d3e882	IPv4	SSH	TCP	22

## Step 6 : Create EC2 Instance

### 1. On aws console search for EC2

**Services**

- EC2** Virtual Servers in the Cloud
- EC2 Image Builder** A managed service to automate build, customize and deploy OS images
- Recycle Bin** Protect resources from accidental deletion

**Features**

- EC2 Instances** ■ CloudWatch feature
- EC2 Resource Health** ■ CloudWatch feature

**Dashboard**

Were these results helpful? **Yes** **No**

2. Click EC2
3. On EC2 dashboard page click launch instance

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with navigation links like 'Dashboard', 'Instances', 'Images', and 'Elastic Block Store'. The main area has a 'Resources' summary table and a 'Launch instance' section. The 'Launch instance' section contains a large orange 'Launch instance' button, which is the focal point of the screenshot.

4. On page launch an instance you can create and add your instance name , example “aldo Server”

The screenshot shows the 'Launch an instance' configuration page. It has sections for 'Name and tags', 'Image', 'Instance type', 'Networking & security', 'Block device mapping', and 'Advanced details'. In the 'Name and tags' section, there's a table with one row: 'Key' (Name) and 'Value' (aldo Server). Below the table is a note: 'You can add up to 49 more tags.'

5. In application and OS Images (Amazon Machine Image) chose Amazon Linux then leave default the rest

**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 Linux](#) [macOS](#) [Ubuntu](#) [Windows](#) [Red Hat](#) [SUSE Linux](#) [Debian](#)

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

Amazon Linux 2023 kernel-6.1 AMI  
ami-0ebf411a80b6b2cb (64-bit (x86), uefi-preferred) / ami-0e723566181f273cd (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 2023.9.20251208.0 x86\_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username   <a href="#">Edit</a>
64-bit (x86) ▾	uefi-preferred	ami-0ebf411a80b6b2cb	2025-12-03	ec2-user <a href="#">Verified provider</a>

6. In instance Type chose t3.micro
7. And for key pair login chose Proceed without a key pair

**Instance type [Info](#) | [Get advice](#)**

**Instance type**

**t3.micro** Free tier eligible  
Family: t3 2 vCPU 1 GiB Memory Current generation: true On-Demand SUSE base pricing: 0.0104 USD per Hour  
On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour On-Demand Windows base pricing: 0.0196 USD per Hour  
On-Demand RHEL base pricing: 0.0392 USD per Hour On-Demand Linux base pricing: 0.0104 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**

Proceed without a key pair (Not recommended) Default value ▾ [Create new key pair](#)

8. In network setting , you can click edit

**Network settings [Info](#)**

[Edit](#)

**Network [Info](#)**  
vpc-0bf71d4a3c5a6bd1c

**Subnet [Info](#)**  
No preference (Default subnet in any availability zone)

**Auto-assign public IP [Info](#)**  
Enable  
Additional charges apply when outside of free tier allowance

**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

We'll create a new security group called 'launch-wizard-1' with the following rules:

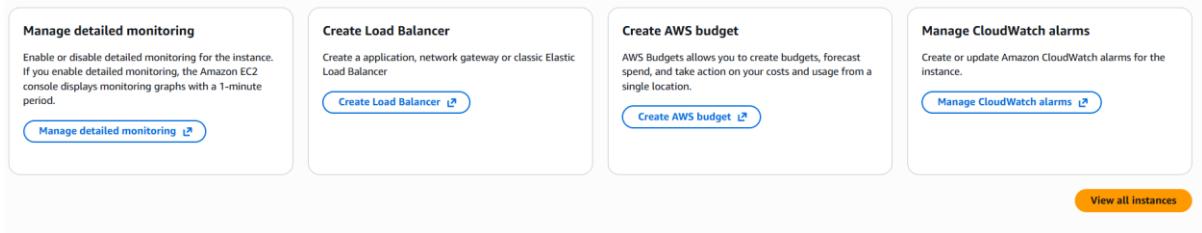
Allow SSH traffic from Anywhere  
Helps you connect to your instance 0.0.0.0/0

9. In network setting

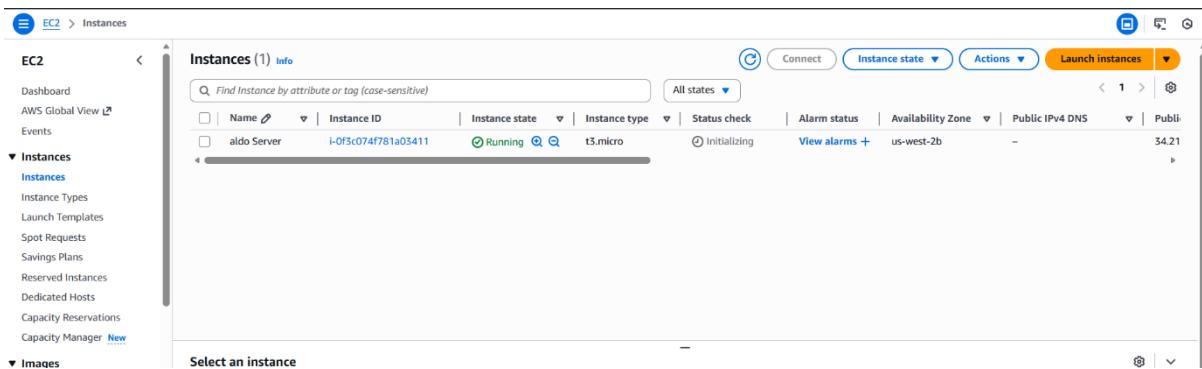
10. In vpc column you chose your vpc has been made before
11. And subnet will automatically fill itself or you can chose you subnet has been made before incase you subnet column not automatically fill itself
12. Auto-assign public IP chose ENABLE
13. Firewall / Security group chose select existing security group , then chose you security group you made before

14. Leave default the rest and click launch instance

15. At page launch successfully just scroll down and then click view all instances



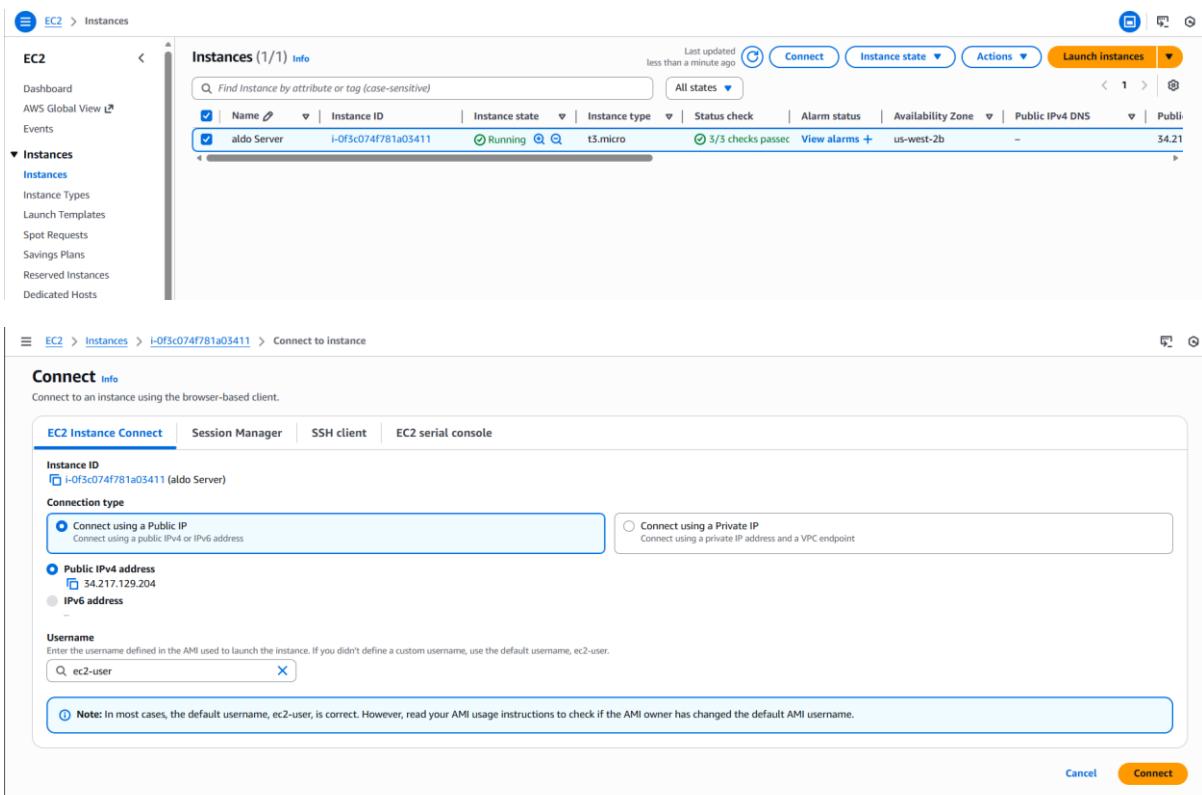
16. On page instace dashboard you will see you instance , then you see at status check column there is Initializing



17. Just wait 1 – 5 menit then you can click refresh until status check become check passed

18. Then you click check box instance and then click connect , you will redirect to page coonect and then you chose EC2 instance connect and click connect

19. And then you will redirect to amazon linux



## Step 7 : Install nginx and create a html.index for simple website

- Run following command

Sudo yum update -y

```
[ec2-user@ip-10-0-0-227 ~]$ sudo yum update -y
amazon Linux 2023 Kernel Livepatch repository
Last metadata expiration check: 0:00:01 ago on Tue Dec 9 16:44:45 2025.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-0-227 ~]$
```

- Install nginx

Sudo yum install nginx -y

```
[ec2-user@ip-10-0-0-227 ~]$ sudo yum install nginx -y
Last metadata expiration check: 0:00:46 ago on Tue Dec 9 16:44:45 2025.
Dependencies resolved.
=====
Package           Architecture      Version        Repository      Size
=====
Installing:
nginx            x86_64          1:1.28.0-1.amzn2023.0.2
=====
Installing dependencies:
generic-logos-httd
generic-tools-libs
libunwind
nginx-core
nginx-filesystem
nginx-mimetypes
=====
Transaction Summary
Install 7 Packages
=====
Total download size: 1.1 M
Installed size: 3.7 M
Downloading Packages:
(1/7): libunwind-1.4.0-5.amzn2023.0.3.x86_64.rpm
(2/7): generic-logos-httd-18.0.0-12.amzn2023.0.3.noarch.rpm
(3/7): generic-tools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm
(4/7): nginx-1.28.0-1.amzn2023.0.2.x86_64.rpm
(5/7): nginx-core-1.28.0-1.amzn2023.0.2.x86_64.rpm
(6/7): nginx-filesystem-1.28.0-1.amzn2023.0.2.noarch.rpm
(7/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm
=====
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing:
    Running scriptlet: nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch
  Installing : nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch
  Installing : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
  Installing : libunwind-1.4.0-5.amzn2023.0.3.x86_64
  Installing : generic-tools-libs-2.9.1-1.amzn2023.0.3.x86_64
  Installing : nginx-1.28.0-1.amzn2023.0.2.x86_64
  Installing : generic-logos-httd-18.0.0-12.amzn2023.0.3.noarch
  Installing : nginx-1.28.0-1.amzn2023.0.2.x86_64
=====
  generic-logos-httd-18.0.0-12.amzn2023.0.3.noarch generic-tools-libs-2.9.1-1.amzn2023.0.3.x86_64 libunwind-1.4.0-5.amzn2023.0.3.x86_64 nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
=====
Complete!
[ec2-user@ip-10-0-0-227 ~]$
```

- Star and enable nginx

Sudo systemctl start nginx

Sudo systemctl enable nginx

```
[ec2-user@ip-10-0-0-227 ~]$ sudo systemctl start nginx
[ec2-user@ip-10-0-0-227 ~]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-10-0-0-227 ~]$
```

#### 4. Create index html

Sudo nano /usr/share/nginx/html/index.html

```
[ec2-user@ip-10-0-0-227 ~]$ sudo nano /usr/share/nginx/html/index.html
```

#### 5. Create or copy paste you website code using html

```
GNU nano 2.3                                         /usr/share/nginx/html/index.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DevOps Portfolio Project</title>
    <style>
        body {
            margin: 0;
            font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
            background: #0e0eff;
            color: #f2f2f2;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            text-align: center;
        }

        h1 {
            font-size: 2.8rem;
            margin-bottom: 10px;
            color: #4eaaff;
        }

        p {
    
```

File Name to Write: /usr/share/nginx/html/index.html

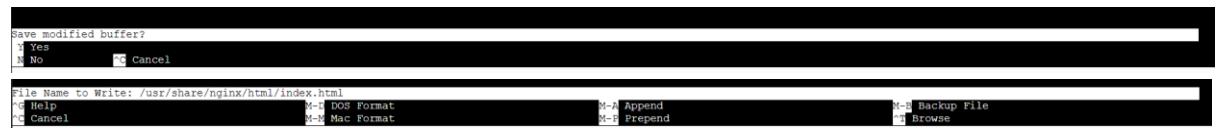
Save modified buffer?  
Y Yes  
N No       Cancel

M-DOS Format      M-A Append  
M-M Mac Format      M-P Prepend  
 Help       Backup File  
 Cancel       Browse

#### 6. Then press CTRL + X

#### 7. Then press Y

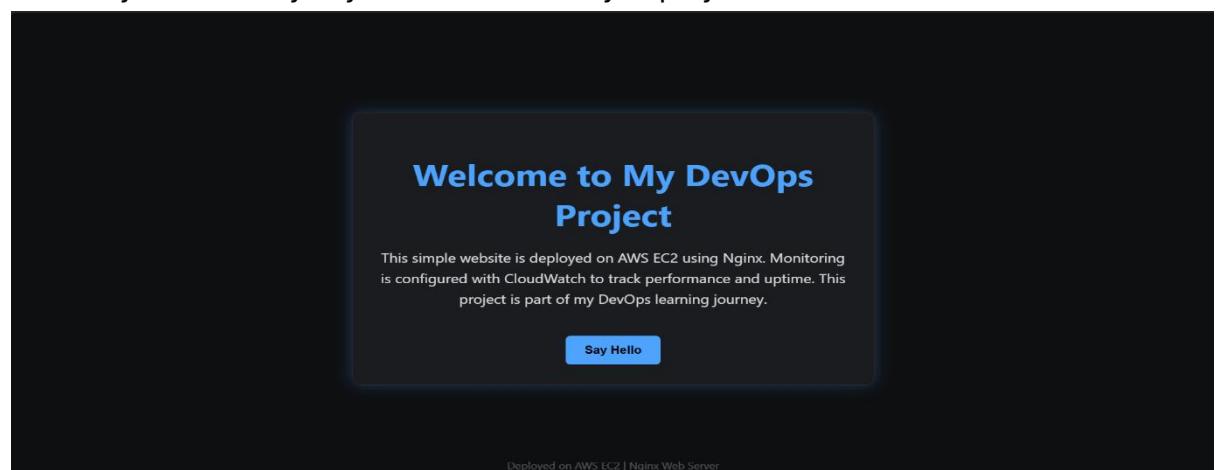
#### 8. Then enter



#### 9. Go back to page AWS console then go to EC2 instance in detail page you will see IP public

#### 10. Click ip public then the new page will be open

#### 11. And you can see your website already deployed



**By completing this project, I learned:**

- How to launch and configure an EC2 instance
- How to manage Linux services (systemctl)
- How to install and configure a web server
- How to deploy static content
- How to configure firewall/security groups
- How to work with Nginx directory structure