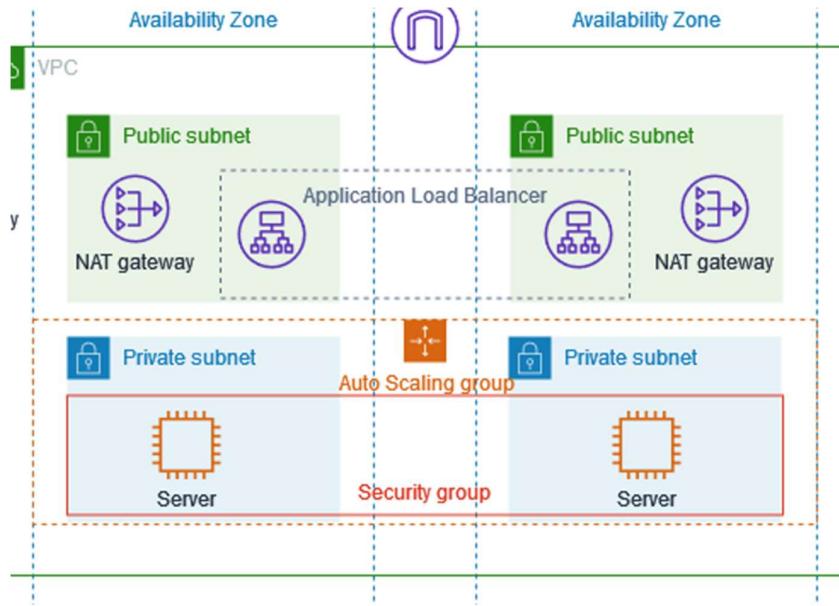


# AWS PROJECT- VPC with Public-Private Subnet.



## Main Goal:-

To securely deploy a web application where the application servers are hosted in a private subnet and can only be accessed through a public-facing Application Load Balancer (ALB) in a public subnet, ensuring the backend remains isolated from direct internet access while still being reachable via the ALB.

For example, think of the VPC as a gated community—while the ALB acts as the front gate accessible to the public, the instances (your application or website) reside securely inside. When an external user accesses the application through the ALB's public DNS, the ALB forwards the request to the instances in the private subnet and returns the response.

Thus, users do not directly access the instances; they interact with the ALB, which securely handles communication with the backend instances within the same VPC.

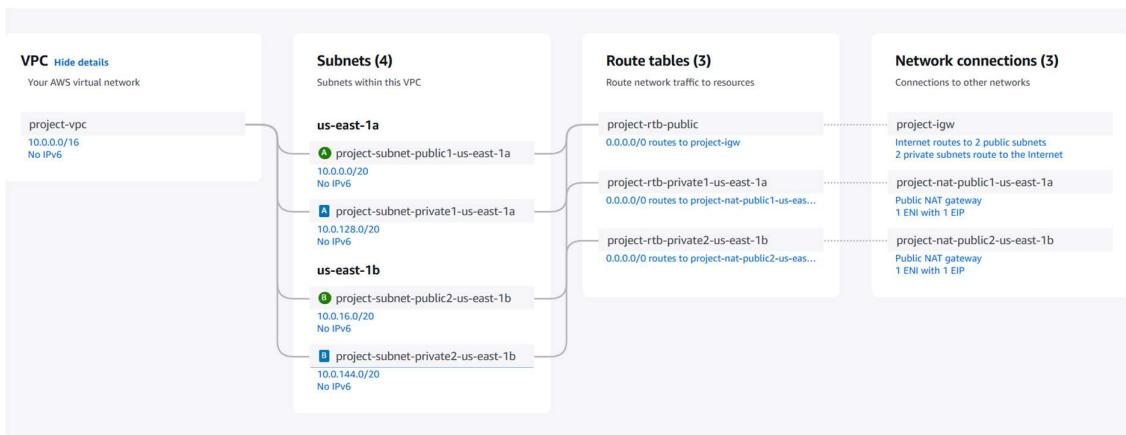
## **WORKING STEPS-**

STEP1-

### **VPC and Subnet Setup**

- **VPC Name:** project-vpc
- **CIDR Block:** 10.0.0.0/16
- Created **4 subnets** (2 public and 2 private) across **2 Availability Zones** (us-east-1a & us-east-1b).
  - **Public Subnets:**
    - project-subnet-public1-us-east-1a
    - project-subnet-public2-us-east-1b
  - **Private Subnets:**
    - project-subnet-private1-us-east-1a
    - project-subnet-private2-us-east-1b
- **Internet Gateway (IGW)** attached for public access.
- **NAT Gateways** created in each public subnet to allow outbound internet access from private subnets.
- **Route Tables** properly configured:
  - Public route table routes to the IGW.
  - Private route tables route to respective NAT gateways.

A)



B)

This screenshot shows the "Create VPC" page in the AWS Management Console.

The top navigation bar includes the AWS logo, search bar, and other navigation links.

The main title is "Create VPC" with an "Info" link.

A descriptive text states: "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."

The page is divided into two main sections:

- VPC settings** (left):
  - Resources to create**: A dropdown menu with "VPC only" (unchecked) and "VPC and more" (checked, highlighted in blue).
  - Name tag auto-generation**: A section where you can enter a name tag for the VPC. It includes a checkbox for "Auto-generate" and a text input field containing "project".
  - IPv4 CIDR block**: A section to determine the starting IP and size of the VPC using CIDR notation. It shows "10.0.0.0/16" and "65,536 IPs". A note says "CIDR block size must be between /16 and /28."
  - IPv6 CIDR block**: A section to choose an IPv6 CIDR block. It has two options: "No IPv6 CIDR block" (selected) and "Amazon-provided IPv6 CIDR block".
- Preview** (right):
  - VPC**: Shows basic details like CIDR (10.0.0.0/16) and IPv6 status (No IPv6).
  - Subnets (4)**: Lists four subnets under "us-east-1a" and one subnet under "us-east-1b".
    - us-east-1a**:
      - project-sut: CIDR 10.0.0.0/20, No IPv6
      - project-sut: CIDR 10.0.128.0/20, No IPv6
    - us-east-1b**:
      - project-sut: CIDR 10.0.16.0/20, No IPv6
      - project-sut: CIDR 10.0.144.0/20, No IPv6

C)

The screenshot shows the 'Create VPC' wizard in the AWS VPC service. The configuration steps are as follows:

- Tenancy**: Set to 'Default'.
- Number of Availability Zones (AZs)**: Set to 2 (selected from 1, 2, 3).
- Number of public subnets**: Set to 2 (selected from 0, 2).
- Number of private subnets**: Set to 4 (selected from 0, 2, 4).
- NAT gateways (\$)**: Set to '1 per AZ'.
- VPC endpoints**: Set to 'S3 Gateway'.
- DNS options**: Both 'Enable DNS hostnames' and 'Enable DNS resolution' are checked.
- Additional tags**: None specified.

At the bottom, there are 'Cancel', 'Preview code', and 'Create VPC' buttons.

## Create VPC

The screenshot shows the 'Create VPC workflow' page, indicating a 'Success' status. The workflow details are as follows:

- Success: Create VPC: vpc-0a7dc65ca48ec058 (status: Succeeded)
- Details:
  - Create VPC: vpc-0a7dc65ca48ec058 (status: Succeeded)
  - Enable DNS hostnames (status: Succeeded)
  - Enable DNS resolution (status: Succeeded)
  - Verifying VPC creation: vpc-0a7dc65ca48ec058 (status: Succeeded)
  - Create subnet: subnet-001a3ead53b1c1c4 (status: Succeeded)
  - Create subnet: subnet-04e9462925e161eae (status: Succeeded)
  - Create subnet: subnet-0bbf042fa35f9aa03 (status: Succeeded)
  - Create subnet: subnet-07651453ac13724b7 (status: Succeeded)
  - Create internet gateway: igw-0ff13cb4b2218e002 (status: Succeeded)
  - Attach internet gateway to the VPC (status: Succeeded)
  - Create route table: rtb-057ca394cc10711fa (status: Succeeded)
  - Create route (status: Succeeded)
  - Associate route table (status: Succeeded)
  - Associate route table (status: Succeeded)
  - Allocate elastic IP: eipalloc-0665fd212f6d47c96 (status: Succeeded)
  - Allocate elastic IP: eipalloc-0211aa20ab81d8s5s (status: Succeeded)
  - Create NAT gateway: nat-0019e71ab261ebf1e (status: Succeeded)
  - Create NAT gateway: nat-059a91d1b24df63d (status: Succeeded)
  - Wait for NAT Gateways to activate (status: Succeeded)
  - Create route table: rtb-01fe1d5d1ca7b8846 (status: Succeeded)
  - Create route (status: Succeeded)
  - Associate route table (status: Succeeded)
  - Create route table: rtb-0f360fd9b510037b4 (status: Succeeded)
  - Create route (status: Succeeded)
  - Associate route table (status: Succeeded)
  - Verifying route table creation (status: Succeeded)

At the bottom right, there is a 'View VPC' button.

## STEPS AFTER CREATION OF VPC

---

# A)Create Security Groups

Set up Security Groups **before** creating the Bastion Host, Load Balancer, and Private EC2s.

---

### Step-by-Step Instructions

---

#### **Step 1: Create Security Group for Load Balancer (SG-ALB)**

1. Go to **EC2 Dashboard** → **Security Groups** → **Create security group**
2. Name: SG-ALB
3. Description: Allow HTTP from public to Load Balancer
4. VPC: Select your VPC (e.g., project-vpc)
5. Inbound Rule:
  - Type: HTTP
  - Port: 80
  - Source: Anywhere (0.0.0.0/0)
6. Outbound Rule: leave default (allow all)
7. Click **Create security group**

**sg-0f12ba679a07e6ad4 - SG-ALB**

**Inbound rules (1)**

Name	Security group rule ID	Type	Protocol	Port range	Source
-	sgr-02c26410bf46bb768	HTTP	TCP	80	0.0.0.0/0

## Step 2: Create Security Group for Bastion Host (SG-Bastion)

1. Go to **Security Groups** → **Create security group**
2. Name: SG-Bastion
3. Description: Allow SSH from my IP
4. VPC: Select your VPC
5. Inbound Rule:
  - Type: SSH
  - Port: 22
  - Source: My IP (or Anywhere if we are doing ssh from different Devices )
6. Outbound Rule: leave default
7. Click **Create security group**

**Basic details**

**Description**

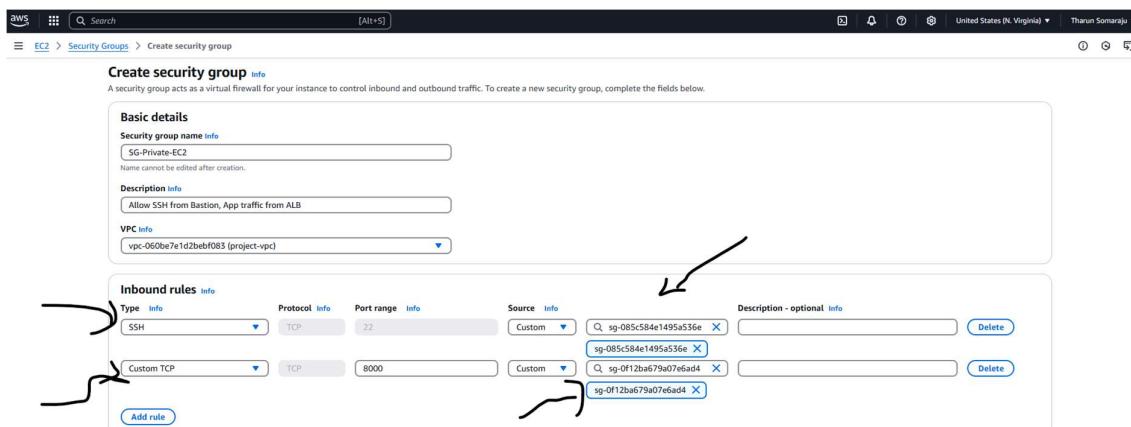
**VPC info**

**Inbound rules**

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	Anywhere	0.0.0.0/0

### Step 3: Create Security Group for Private EC2s (SG-Private-EC2)

1. Go to **Security Groups** → **Create security group**
2. Name: SG-Private-EC2
3. Description: Allow SSH from Bastion, App traffic from ALB
4. VPC: Same VPC as above
5. Inbound Rules:
  - o Rule 1:
    - Type: SSH
    - Port: 22
    - Source: **Custom** → choose **SG-Bastion**
  - o Rule 2:
    - Type: Custom TCP
    - Port: 8000
    - Source: **Custom** → choose **SG-ALB**
6. Outbound Rule: leave default
7. Click **Create security group**



#### THE MAIN IMPORTANT POINTS TO BE NOTED

-So only the Bastion Host instance (that has SG-Bastion) can access your Private EC2, not the whole world.

**-So only the Load Balancer (with SG-ALB) can send traffic to the private EC2 on port 8000, even though the Load Balancer is open to the public.**

---

 Done! You Now Have:

SG Name	Use	Inbound Rules
SG-ALB	Load Balancer	HTTP (80) from Anywhere
SG-Bastion	Bastion Host	SSH (22) from your IP or Anywhere
SG-Private-EC2 Private EC2/ASG		SSH (22) from SG-Bastion TCP 8000 from SG-ALB

---

 Final Step:

When you launch:

- **Load Balancer** → Attach SG-ALB
- **Bastion Host** → Attach SG-Bastion
- **Private EC2s or Auto Scaling Group** → Attach SG-Private-EC2

Now your architecture is secure and properly set up without using “Anywhere” for private EC2s. 

## B) Add Auto Scaling Ec2 in Private Subnet

Go to **Ec2 Auto Scaling**, to create instances in the Private Subnet.

-Create a **Launch Template** for Ec2's , so we don't need to create this template again and again.

So when we want Auto Scale, again, we simply use this Template.

**Choose launch template**

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

**Name**

Auto Scaling group name  
Enter a name to identify the group.  
**AutoScaleMAIN**

Must be unique to this account in the current Region and no more than 255 characters.

**Launch template**

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

Launch template  
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Select a launch template  
[Create a launch template](#)

**Create launch template**

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

**Launch template name and description**

Launch template name - required  
**MyTemplateEc2222**

Must be unique to this account. Max 128 chars. No spaces or special characters like '/\/\*`@'.

Template version description  
A template for ec2

Max 255 chars

**Auto Scaling guidance** | [Info](#)  
Select this if you intend to use this template with EC2 Auto Scaling  
 Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

**► Template tags**  
**► Source template**

**Launch template contents**

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

**Application and OS Images (Amazon Machine Image) - required** | [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

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

**AMI from catalog** | Recents | Quick Start

**Name**  
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

**Description**  
Ubuntu Server 24.04 LTS (HVM).EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

**Verified provider** | **Free tier eligible**

**Browse more AMIs**  
Including AMIs from AWS, Marketplace and the Community

Now SELECT THE Key-Pair, Security Group as (SG-Private-Ec2)

**Instance type**

t2.micro  
Family: t2 - 1 vCPU - 1.6 GB Memory - Current generation - Free tier eligible

**Key pair (login)**

key30April

**Network settings**

Subnet: Don't include in launch template

Firewall (security groups): Select existing security group (SG-Private-EC2)

Security groups: SG-Private-EC2 sg-07422896d67a22915

**Summary**

Software Image (AMI): Ubuntu Server 24.04 LTS (HVM) ... more  
ami-084568db4383264d4

Virtual server type (instance type): t2.micro

Firewall (security group): SG-Private-EC2

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

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs. 750 hours per month of public IPv4 address usage. 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

**Create launch template**

## CREATE LAUNCH TEMPLATE

### ADD THE LAUNCH TEMPLATE TO AUTOSCALING

**Launch template**

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

**Launch template**

Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

MyTemplateEc22222

**Description**

A template for ec2

**AMI ID**

ami-084568db4383264d4

**Key pair name**

key30April

**Launch template**

MyTemplateEc22222  
lt-027f170cf0d85509c

**Security groups**

-

**Security group IDs**

sg-07422896d67a22915

**Instance type**

t2.micro

**Request Spot Instances**

No

**Additional details**

**Storage (volumes)**

-

**Date created**

Tue Jun 03 2025 18:40:03 GMT+0530 (India Standard Time)

**Cancel** **Next**

### ADD THE AutoScaler to VPC, i.e in the Private Subnets ONLY

Step 1  
Step 2  
**Choose instance launch options**  
Step 3 - optional  
Step 4 - optional  
Step 5 - optional  
Step 6 - optional  
Step 7 - Review

**Choose instance launch options** Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

**Instance type requirements** Info

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

<b>Launch template</b> MyTemplateEc22222 <small>Info</small> lt-027f170cf085509c	<b>Version</b> Default	<b>Description</b> A template for ec2
--	---------------------------	--

**Instance type**  
t2.micro

**Network** Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**  
Choose the VPC that defines the virtual network for your Auto Scaling group.  
vpc-060be7e1d2bebfb083 (project-vpc) Info  
10.0.0.0/16 Info

**Create a VPC** Info

**Availability Zones and subnets**  
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets Info

us-east-1a | subnet-0b04645d294261d23 (project-subnet-private1-us-east-1a) Info  
10.0.128.0/20 Info

us-east-1b | subnet-0aac54b923c87b3b3 (project-subnet-private2-us-east-1b) Info  
10.0.144.0/20 Info

**Create a subnet** Info

**Availability Zone distribution - new**

## Specify Auto Scaling Scale Size

Step 1  
Step 2  
Step 3 - optional  
Step 4 - optional  
**Configure group size and scaling - optional** Info  
Step 5 - optional  
Step 6 - optional  
Step 7 - Review

**Configure group size and scaling - optional** Info

Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

**Group size** Info

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances)

**Desired capacity**  
Specify your group size.  
2

**Scaling** Info

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

**Scaling limits**  
Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity  Max desired capacity

Equal or less than desired capacity  Equal or greater than desired capacity

**Automatic scaling - optional**  
Choose whether to use a target tracking policy Info

You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies  
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

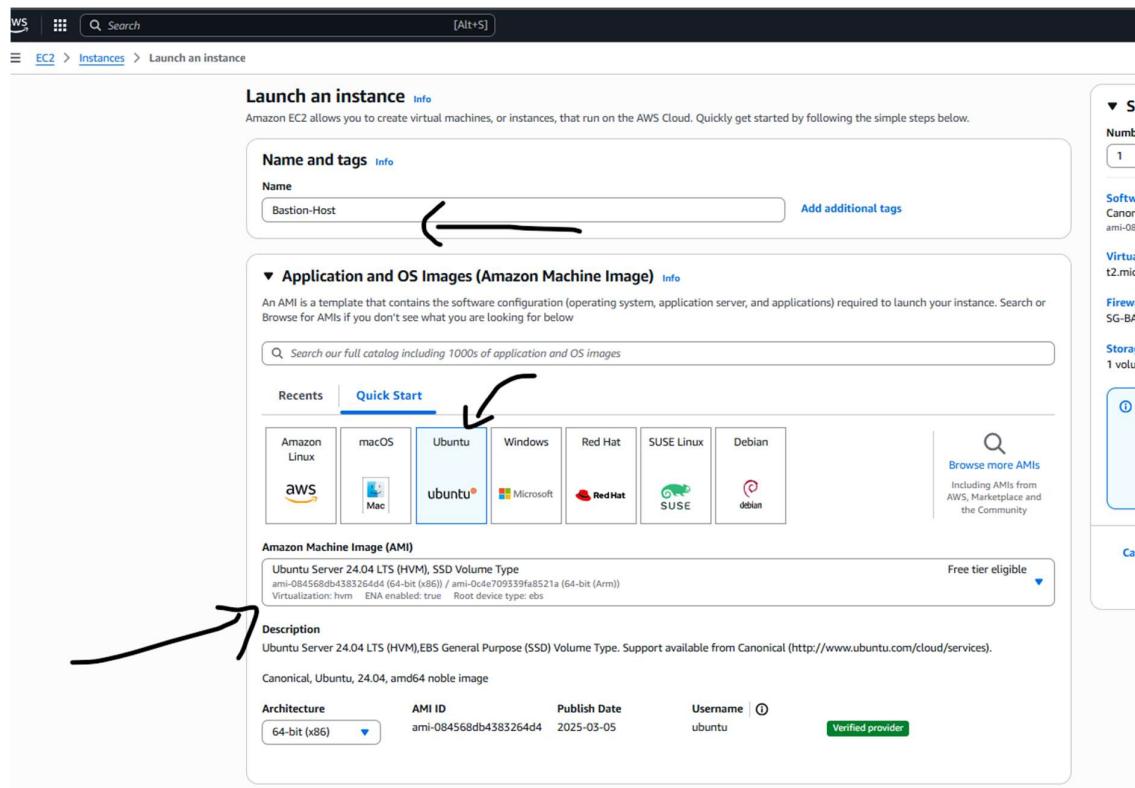
NOW PRESS CREATE AUTO SCALING GROUP

**Create Auto Scaling group**

C)

# Create a Bastion Host in Public Subnet(for contacting private ec2 via SSH)

Launch a Bastion-Host Ec2 Instance



NOW

Specify the Required terms such as

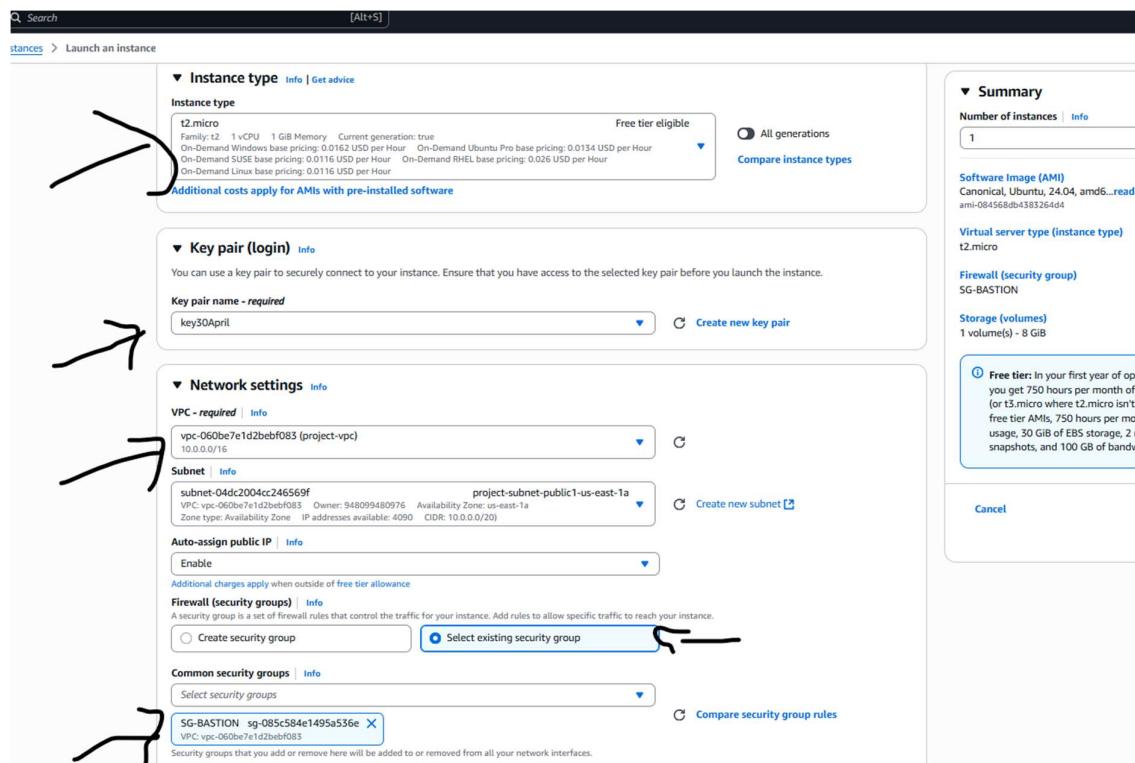
**Key-pair,**

**VPC,**

and the Subnet i.e **PUBLIC SUBNET** of VPC

AND Auto Assign IP- **Enable**

**Add the Security Group that we created for Bastion (SG-Bastion)**



**PRESS LAUNCH INSTANCE**

**Launch instance**

## D)HOST THE WEBSITE IN A PRIVATE INSTANCE

- We first make contact with Bastion, and from Bastion we contact Private Instance to Deploy a HTML Code and Python server for Hosting 8000

- FOR copying key **key30April.pem** from the Local Pc to Bastion (as we want to login to private subnet from bastion)

```
scp -i "C:\Users\Somaraju Tharun\Downloads\key30April.pem" "C:\Users\Somaraju Tharun\Downloads\key30April.pem" ubuntu@54.87.183.33:/home/ubuntu/
```

```
Microsoft Windows [Version 10.0.26100.4202]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Somaraju Tharun>scp -i "C:\Users\Somaraju Tharun\Downloads\key30April.pem" "C:\Users\Somaraju Tharun\Downloads\key30April.pem" ubuntu@54.87.183.33:/home/ubuntu/
The authenticity of host '54.87.183.33 (54.87.183.33)' can't be established.
ED25519 key fingerprint is SHA256:It2A6/paGh0AwdJ2lDAtpw3L/njBoOkn/zM0Yis8j3I.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added '54.87.183.33' (ED25519) to the list of known hosts.
key30April.pem                                         100% 1678      5.7KB/s   00:00

C:\Users\Somaraju Tharun>
```

- To verify if the key got copied or not

```
-ssh -i "C:\Users\Somaraju Tharun\Downloads\key30April.pem" ubuntu@54.87.183.33
```

```
-$ ls
```

### You will see the key

```
C:\Users\Somaraju Tharun>ssh -i "C:\Users\Somaraju Tharun\Downloads\key30April.pem" ubuntu@54.87.183.33
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Tue Jun  3 13:42:28 UTC 2025

 System load:  0.0          Processes:           104
 Usage of /:   25.3% of 6.71GB   Users logged in:     0
 Memory usage: 21%
 Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-3-25:~$ ls
key30April.pem
ubuntu@ip-10-0-3-25:~$ |
```

**-chmod 400** is used to secure the .pem file by allowing only the owner to read it, which SSH requires for safe access.

```
$chmod 400 key30April.pem
```

To SSH from the **Bastion Host** into the **private EC2 instance (10.0.137.234)**, use the following command **on the Bastion terminal**:

```
$ssh -i key30April.pem ubuntu@10.0.137.234
```

```
ubuntu@ip-10-0-3-25:~$ ssh -i key30April.pem ubuntu@10.0.137.234
The authenticity of host '10.0.137.234 (10.0.137.234)' can't be established.
ED25519 key fingerprint is SHA256:5b9PNRHbjd5LKYY494qwLwPNinpmRVeiYoKl1D0rJhM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.137.234' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Tue Jun  3 13:45:57 UTC 2025

  System load:  0.0          Processes:      103
  Usage of /:   25.0% of 6.71GB  Users logged in:    0
  Memory usage: 19%           IPv4 address for enX0: 10.0.137.234
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-137-234:~$ |
```

-

- 1) CREATE A HTML PAGE IN THE EC2 TERMINAL

#### Create the file

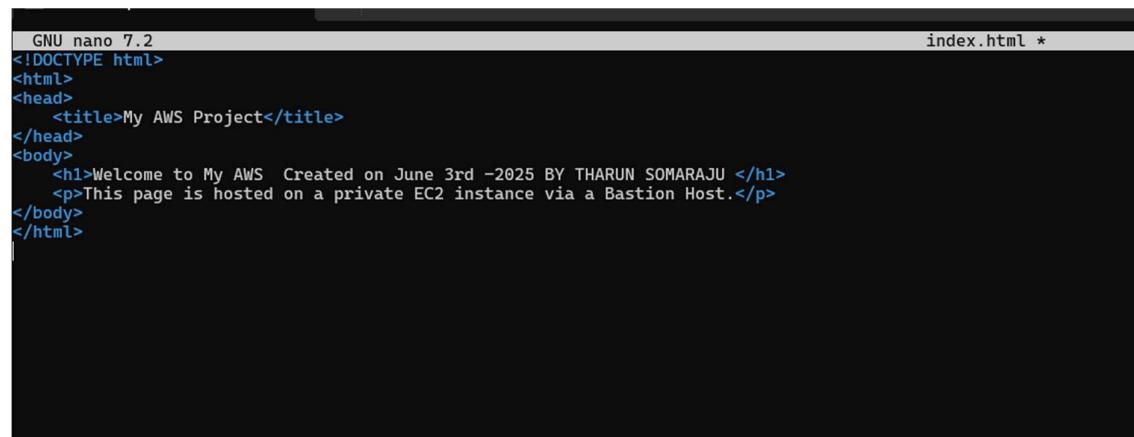
Run this on your EC2 instance:

\$nano index.html

```
ubuntu@ip-10-0-137-234:~$ nano index.html
```

Paste the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>My AWS Project</title>
</head>
<body>
    <h1>Welcome to My AWS Created on June 3rd -2025 BY THARUN SOMARAJU </h1>
    <p>This page is hosted on a private EC2 instance via a Bastion Host.</p>
</body>
</html>
```



The screenshot shows a terminal window with the nano text editor open. The file is named 'index.html'. The code inside the file is identical to the one provided in the previous text block, containing HTML tags for a title, head, body, h1, and p elements.

Save and exit:

- Press **Ctrl + O** → **Enter** to save
- Then **Ctrl + X** to exit Nano

Your HTML page is ready as index.html.

Create a Python server

```
$ python3 -m http.server 8000
```

```
See 'man python3' for details.

ubuntu@ip-10-0-137-234:~$ nano index.html
ubuntu@ip-10-0-137-234:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```



#### What happens when user visits Load Balancer?

1. The request goes to the Load Balancer.
2. Load Balancer sends the request to your private EC2.
3. The Python server running on that EC2 responds with your HTML page.
4. User sees your website!

## IMPORTANT POINT TO REMEMBER ABOUT-

*How does an EC2 instance recognize the key-pair?*

When we

SSH FROM Local PC → Bastion-Host(Public Subnet)

Bastion-Host(Public Subnet) → Ec2 (Private Subnet)

Here we are copying KeyPair.pem from local pc to bastion , as we want this KeyPair.pem to be verified with private ec2 when we are doing SSH from bastion to Private ec2.

***BELOW IS THE MAIN AND IMPORTANT DESCRIPTION about the ABOVE:-***

When you launch an EC2 instance, you attach a key-pair (like key30April.pem) to it. Here's what happens:



Behind the Scenes:

1. When the EC2 is created with a key-pair:

- AWS puts the **public part** of the key into a file called:

**/home/ubuntu/.ssh/authorized\_keys**

2. When you try to connect using:

**ssh -i key30April.pem ubuntu@<ec2-ip>**

- Your **private key** (key30April.pem) is used by your SSH client.
  - The EC2 instance checks:
    - “Does this private key match the public key I have in authorized\_keys?”
3.  If it matches:
- You are allowed to login.
4.  If it doesn't match:
- Access is denied.

---

### Simple Analogy:

Think of it like a **lock and key**:

- EC2 has the **lock** (public key).
- You have the **key** (private key .pem).
- Only the **right key** can unlock it.

---

### So to answer your question:

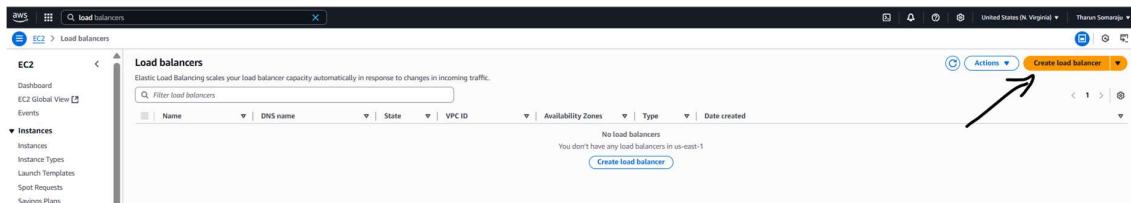
**How does EC2 recognize the key-pair?**

Because **when the instance is launched, AWS automatically saves the public part of the key-pair** into the EC2's login system. When you try to connect using the private key, it matches it with the stored public key.

---

# E)CREATE THE LOAD BALANCER in the Public Subnet And its Targets (Private instances )

- Create the Load balancer in the Public Subnet because , A user Can get Only LB URL to access the website present in the Private Ec2 Instance..

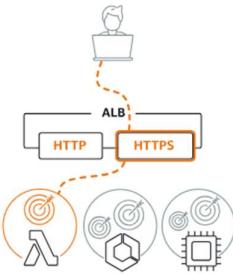
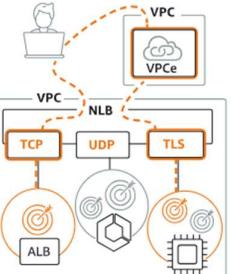


Select Application Load balancer

EC2 > Load balancers > Compare and select load balancer type

## Compare and select load balancer type

A complete feature-by-feature comparison along with detailed highlights is also available. [Learn more](#)

Load balancer types
<b>Application Load Balancer</b> <a href="#">Info</a>

Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.
<a href="#">Create</a>
<b>Network Load Balancer</b> <a href="#">Info</a>

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.
<a href="#">Create</a>
<b>Gateway Load Balancer</b> <a href="#">Info</a>

Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.
<a href="#">Create</a>

WS load instances

EC2 > Load balancers > Create Application Load Balancer

Application Load Balancers now support public IPv4 IP Address Management (IPAM). You can start with this feature by configuring IP pools in the Network mapping section.

How Application Load Balancers work

Basic configuration

Load balancer name: LoadBalancerJune1

Scheme: Internal

Load balancer IP address type: IPv4

Select the VPC we created , Select the AZ's i.e Public Subnets, and Add the **Security Group(SG-ALB)** the we created At initial Working

**Network mapping**

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC**

The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view [target groups](#) For a new VPC, click [Create a new VPC](#).

**IP pools - new**

You can optionally choose to configure an IPAM pool as the preferred source for your load balancer's IP addresses. Create or view Pools in [Amazon VPC IP Address Manager console](#).

Use IPAM pool for public IPv4 addresses

The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted, IPv4 addresses will be assigned by AWS.

**Availability Zones and subnets**

Select at least two Availability Zones and a subnet for each zone. A load balancer will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

us-east-1a (use1-a2z)

**Subnet**

Only CGR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.

subnet-04c2004c246569f (IPv4 subnet CDR: 10.0.0.0/20) project-subnet-public1-us-east-1a

us-east-1b (use1-a2z)

**Subnet**

Only CGR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.

subnet-095cb11da03ab5 (IPv4 subnet CDR: 10.0.16.0/20) project-subnet-public2-us-east-1b

**Security groups**

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

**Security groups**

Select up to 5 security groups

SG-ALB sg-0f123a79a07e6ad4 VPC vpc-060be71d27ebfb83

Create a Target Group that Load balancer has to pass the Traffic to..

**Listeners and routing**

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

**Listener HTTP:80**

Protocol: HTTP Port: 80 Default action: Select a target group

**Create target group**

**Step 1 Specify group details**

**Specify group details**

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

**Basic configuration**

Settings in this section can't be changed after the target group is created.

**Choose a target type**

- Instances
  - Supports load balancing to instances within a specific VPC.
  - Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.
- IP addresses
  - Supports load balancing to VPC and on-premises resources.
  - Facilitates routing to multiple IP addresses and network interfaces on the same instance.
  - Offers flexibility with microservice-based architectures, simplifying inter-application communication.
  - Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
- Lambda function
  - Facilitates routing to a single Lambda function.
  - Accessible to Application Load Balancers only.
- Application Load Balancer
  - Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
  - Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

**TG-JUNE**

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Protocol**

Protocol for load balancer-to-target communication. Can't be modified after creation.

HTTP Port: 80

Keep the Port Number as 8000, as the Instance allowed the Inbound as 8000(SG-ALB)

>Create target group

Step 1  Specify group details

Step 2  Register targets

### Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

#### Basic configuration

Settings in this section can't be changed after the target group is created.

##### Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates the use of multiple network interfaces on the same instance.
- Offers flexibility with microservice-based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

**Target group name**

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Protocol**

Protocol for load balancer-to-target communication. Can't be modified after creation.

HTTP

**Port**

Port number where targets receive traffic. Can be overridden for individual targets during registration.

T-65535

## Specify the VPC

**IP address type**

Only targets with the indicated IP address type can be registered to this target group.

IPv4

Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

IPv6

Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

**VPC**

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

project-vpc  
vpc-060be7e1d2bebf083  
IPv4 VPC CIDR: 10.0.0.0/16

**Protocol version**

HTTP1

Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

HTTP2

Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

gRPC

Send requests to targets using gRPC. Supported when the request protocol is gRPC.

## Add the Targets

Create target group

**Register targets**

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

**Available instances (2/3)**

Instance ID	Name	State	Security groups	Zone	Private IPv4 address	Subnet ID	Launch time
i-020e20d7bedff6bc2f	Bastion-Host	Running	SG-BASTION	us-east-1a	10.0.3.25	subnet-04dc2004cc246569f	June 3, 2025, 19:00 UTC+
i-0cd5b6bb15fecf0b		Running	SG-Private-EC2	us-east-1b	10.0.157.120	subnet-0aac54b923c87b3b3	June 3, 2025, 18:48 UTC+
i-03a20dc5a8bf2fd090		Running	SG-Private-EC2	us-east-1a	10.0.137.234	subnet-0b04645d294261d23	June 3, 2025, 18:48 UTC+

2 selected

Ports for the selected instances

Ports for routing traffic to the selected instances.

1-65535 (separate multiple ports with commas)

**Review targets**

Targets (2)

Instance ID	Name	Port	State	Security groups	Zone	Private IPV4 address	Subnet ID	Launch time
i-0c05b68b135fee0b	8000	Running	SG-Private-EC2	us-east-1b	10.0.157.120	subnet-0aac54b923c87b3b3	June 3, 2025, 18:48 (UTC+05:30)	
i-05a20dc3a8bf2d090	8000	Running	SG-Private-EC2	us-east-1a	10.0.157.234	subnet-0b04645d294261d23	June 3, 2025, 18:48 (UTC+05:30)	

2 pending

Cancel Previous Create target group

## Add the Target Group to the Load Balancer

**Listeners and routing**

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

**Listener HTTP:80**

Protocol: HTTP Port: 80 Default action: TG-JUNES

**Listener tags - optional**

**Add listener tag**

**Add listener**

**PRESS CREATE LOAD-BALANCER BUTTON**

**Copy the URL of Load Balancer AND HAPPILY ACCESS THE WEBSITE THAT IS PRESENT IN THE PRIVATE SUBNET Ec2 Instance..**

**LoadBalancerJune3**

**Details**

Load balancer type: Application Status: Provisioning VPC: vpc-060be7e1d2bebfb083

Scheme: Internet-facing Hosted zone: Z35SXDOTRQ7X7K Availability Zones: subnet-0a95ch11cdca03abec us-east-1b (use1-az4) subnet-004cc246569f us-east-1a (use1-az2)

Date created: June 3, 2025, 19:39 (UTC+05:30)

**Listeners and rules (1)**

Protocol: HTTP:80 Default action: TG-JUNES ARN: arn:aws:elasticloadbalancing:us-east-1:948099480976:loadbalancer/app/LoadBalancerJune3/50aebe3e0466456f64

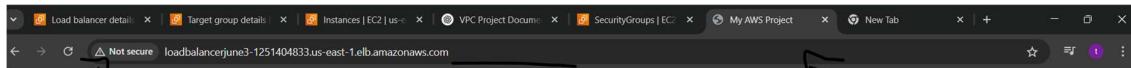
**Listeners and rules (1)**

Protocol: HTTP:80 Default action: TG-JUNES ARN: Not applicable

**Now We can see the Website that we Hosted on Private Subnet..**

**Here The user is Not Directly accessing the Private Ec2 Instances , instead Via Load Balancer User can Access the Website that is created in the Private Ec2 ..**

**Therefore, Now Our Main Goal is Achieved**



Welcome to My AWS Created on June 3rd -2025 BY THARUN SOMARAJU

This page is hosted on a private EC2 instance via a Bastion Host.



A screenshot of the AWS CloudWatch Metrics console. The left sidebar shows 'Metrics' under the 'CloudWatch Metrics' section. The main area displays a chart for a metric named 'Lambda Function Invocations' over a 1-hour period. The chart shows a single data series with a value of 1. Below the chart, there's a table with columns: Metric Name, Namespace, Unit, and Value. The table shows the same data point: Metric Name is 'Lambda Function Invocations', Namespace is 'AWS/Lambda', Unit is 'Count', and Value is 1. At the bottom, there are tabs for 'Metrics' (selected), 'Logs', 'CloudWatch Metrics Insights', and 'CloudWatch Metrics Analytics'.

**Therefore , One instance is Healthy and another one not healthy because , We Hosted the Website in only One Ec2 Instance, So the Request Goes for only One instance..**

---

## **STEPS TO DISMANTLE THE RESOURCES**

Here's a simple checklist to **safely dismantle each AWS tool** from your project and avoid charges:

---

### **✓ 1. Terminate EC2 Instances**

- Go to **EC2 > Instances**
  - Select all running instances (including Bastion and private ones)
  - Click "**Instance State**" > **Terminate**
- 

### **✓ 2. Delete Auto Scaling Groups**

- Go to **EC2 > Auto Scaling Groups**
  - Delete your Auto Scaling Group (ASG)
- 

### **✓ 3. Delete Load Balancer**

- Go to **EC2 > Load Balancers**
  - Select and **delete** your Application Load Balancer (ALB)
- 

### **✓ 4. Delete Target Groups**

- Go to **EC2 > Target Groups**
  - Select and **delete** them (if not auto-removed)
-

 **5. Delete NAT Gateway**

- Go to **VPC > NAT Gateways**
- Select and **delete**

 NAT Gateway can **incur charges even when idle** — make sure to remove it!

---

 **6. Release Elastic IP (if any)**

- Go to **VPC > Elastic IPs**
  - Disassociate and then **release** unused IPs
- 

 **7. Delete Subnets and Route Tables**

- Go to **VPC > Subnets and Route Tables**
  - Delete custom ones **after** EC2s and NAT are gone
- 

 **8. Delete VPC**

- Go to **VPC > Your VPCs**
- Select and **delete your custom VPC**

-----THE END-----

