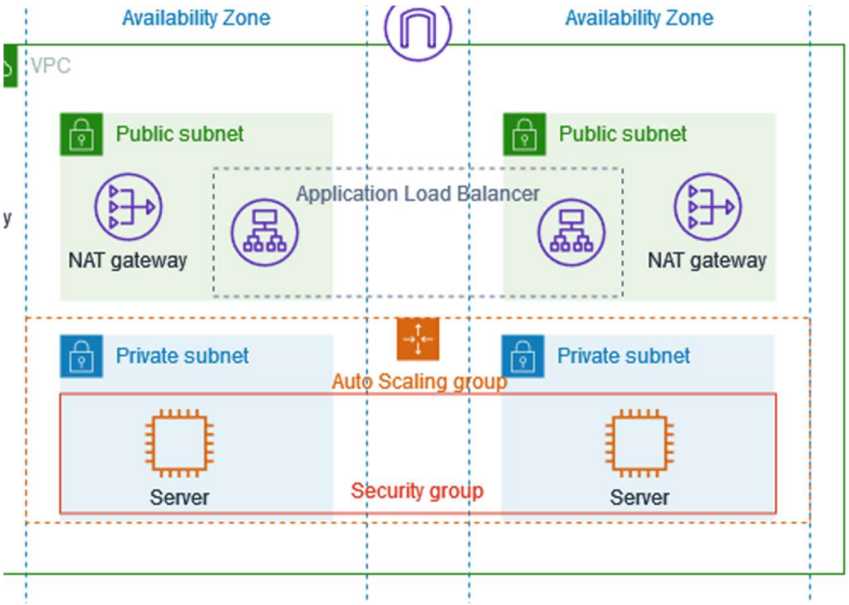
AWS PROJECT- *VPC with Public-Private Subnet.*



Main Goal:-

The VPC creates a private network for your instances. Although the Application Load Balancer (ALB) is deployed in a public subnet, the instances are placed in private subnets, making them inaccessible directly from the internet. This setup ensures that only the ALB can communicate with the instances.

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.

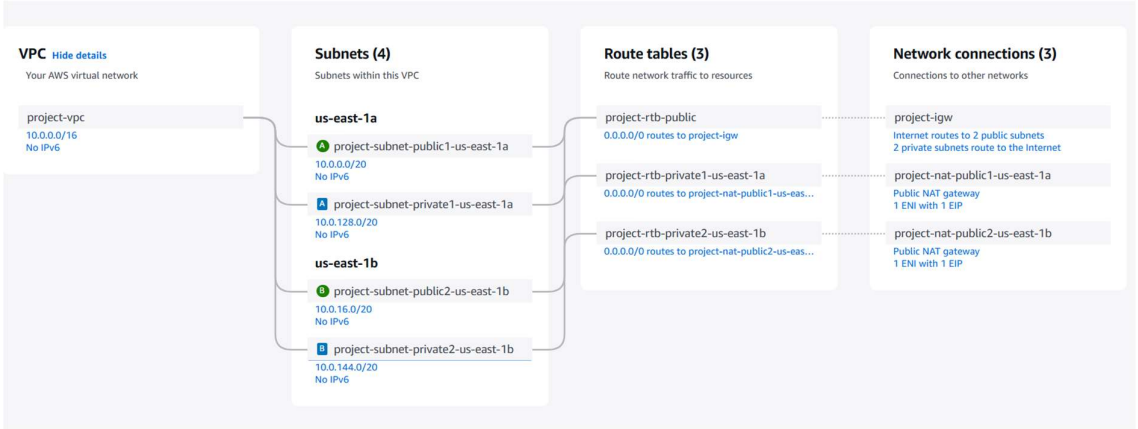
WORKING STEPS-

STEP1-

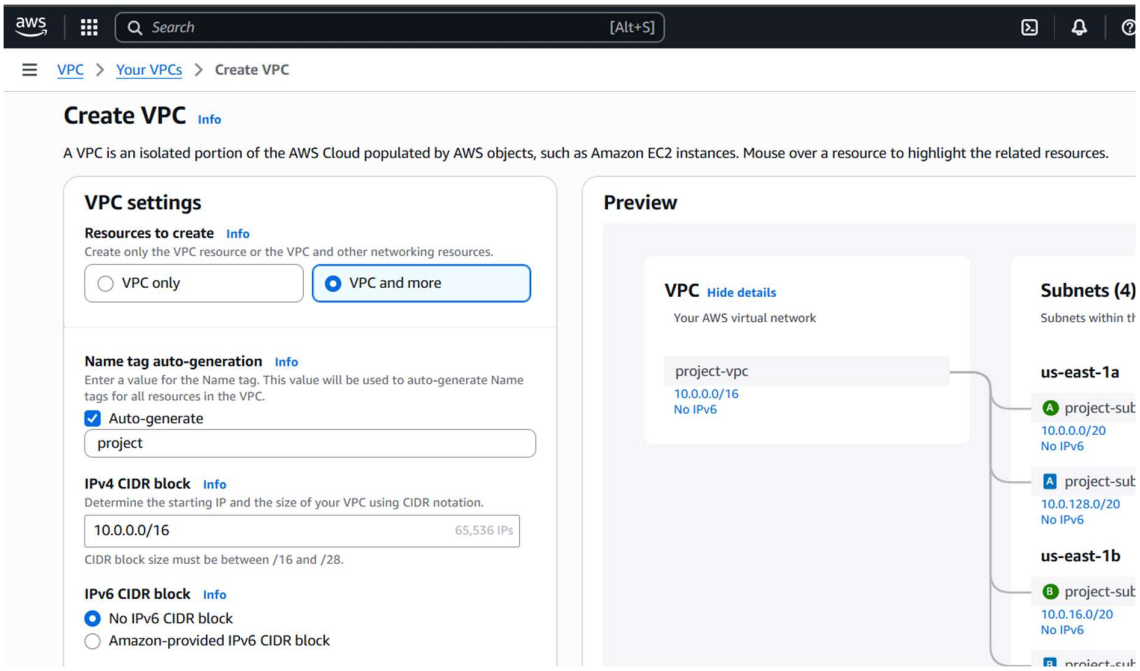
VPC and Subnet Setup

- **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.

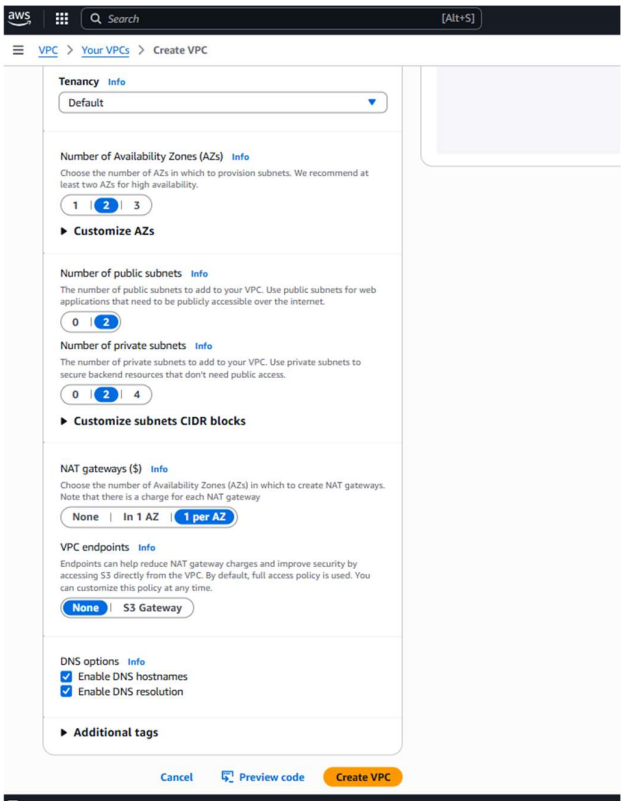
A)



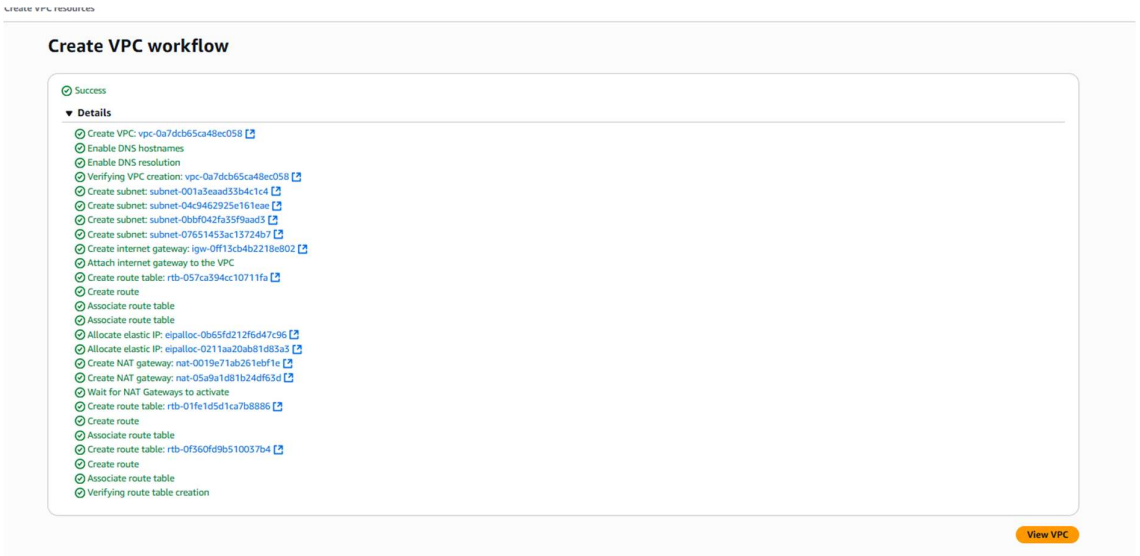
B)


aws
Search
[Alt+S]
VPC > Your VPCs > Create VPC
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
project
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
Preview
VPC Hide details
Your AWS virtual network
project-vpc
10.0.0.0/16
No IPv6
Subnets (4)
us-east-1a
project-sub
10.0.0.0/20
No IPv6
project-sub
10.0.128.0/20
No IPv6
us-east-1b
project-sub
10.0.16.0/20
No IPv6

C)



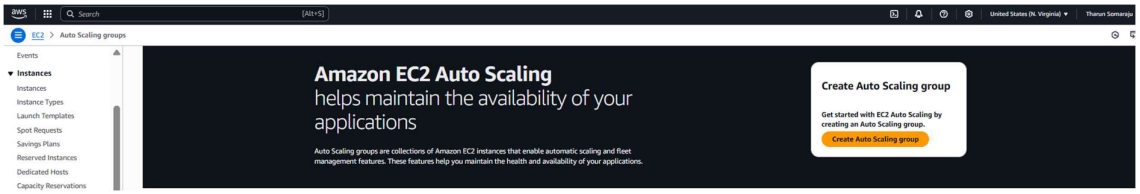
Create VPC



STEP 2-

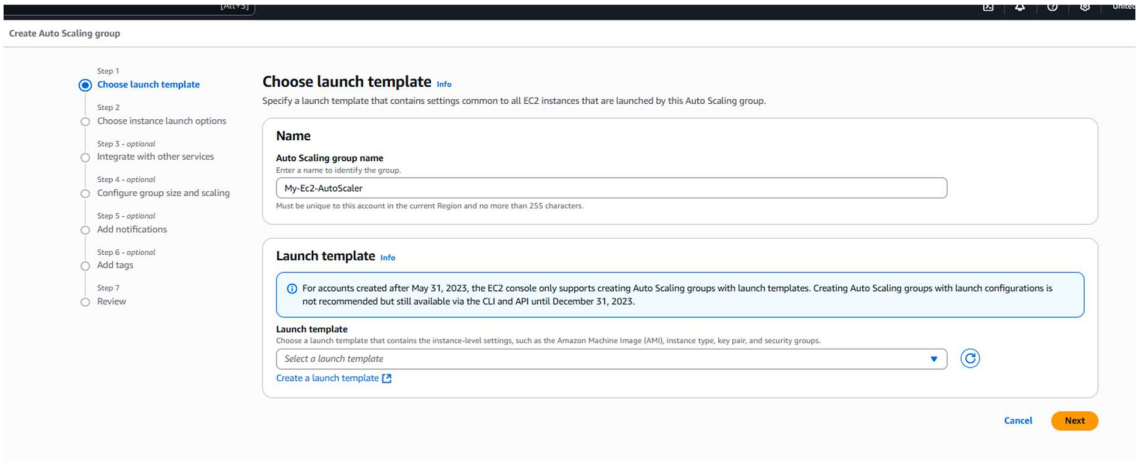
A)

Launch Ec2 Ubuntu Instance in each Private Subnets, by Auto Scaling



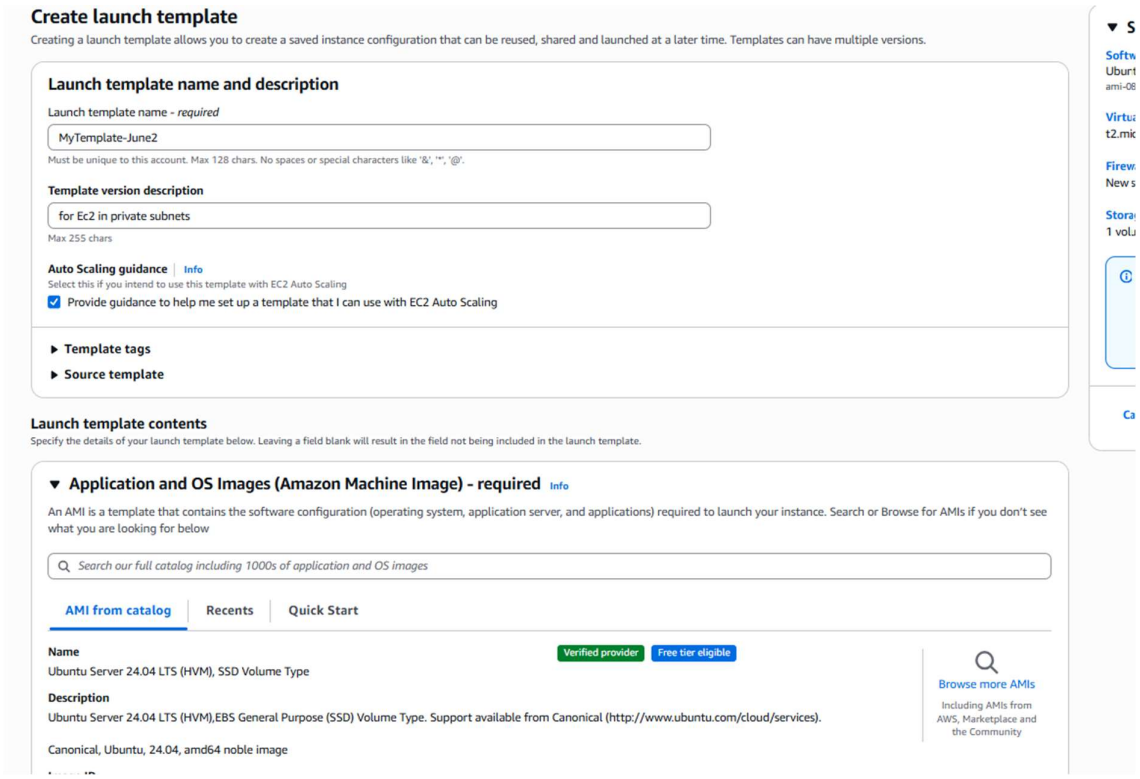
B)

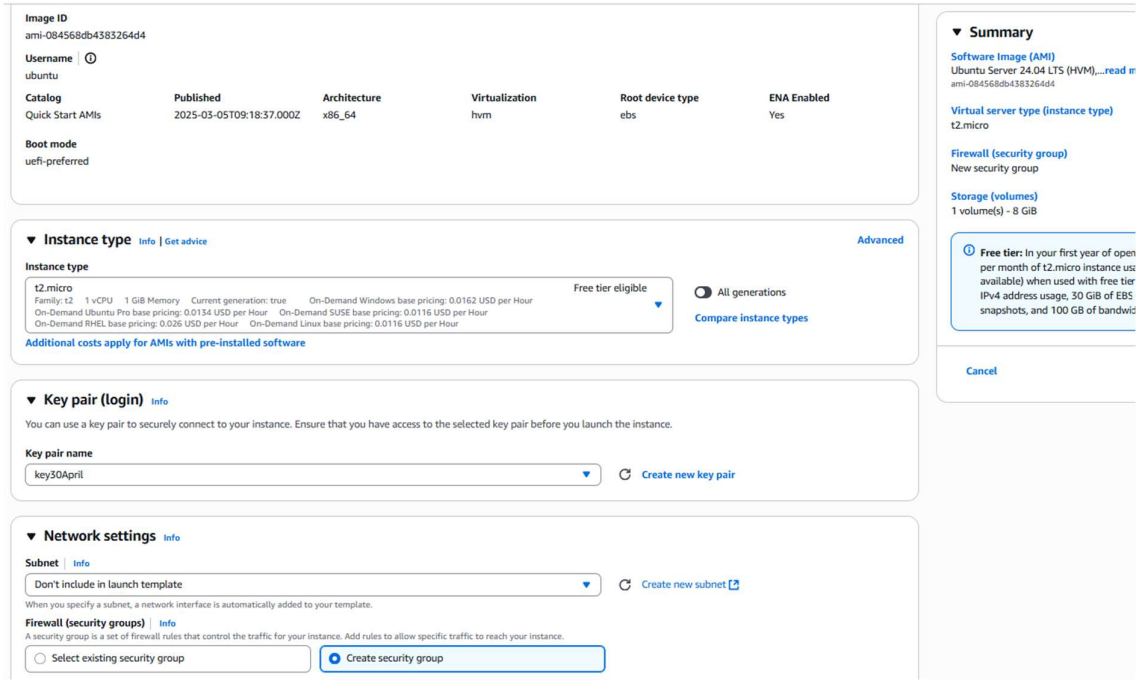
Create Launch Template for Ec2 Specifications (which we use in the Future for any Creations)



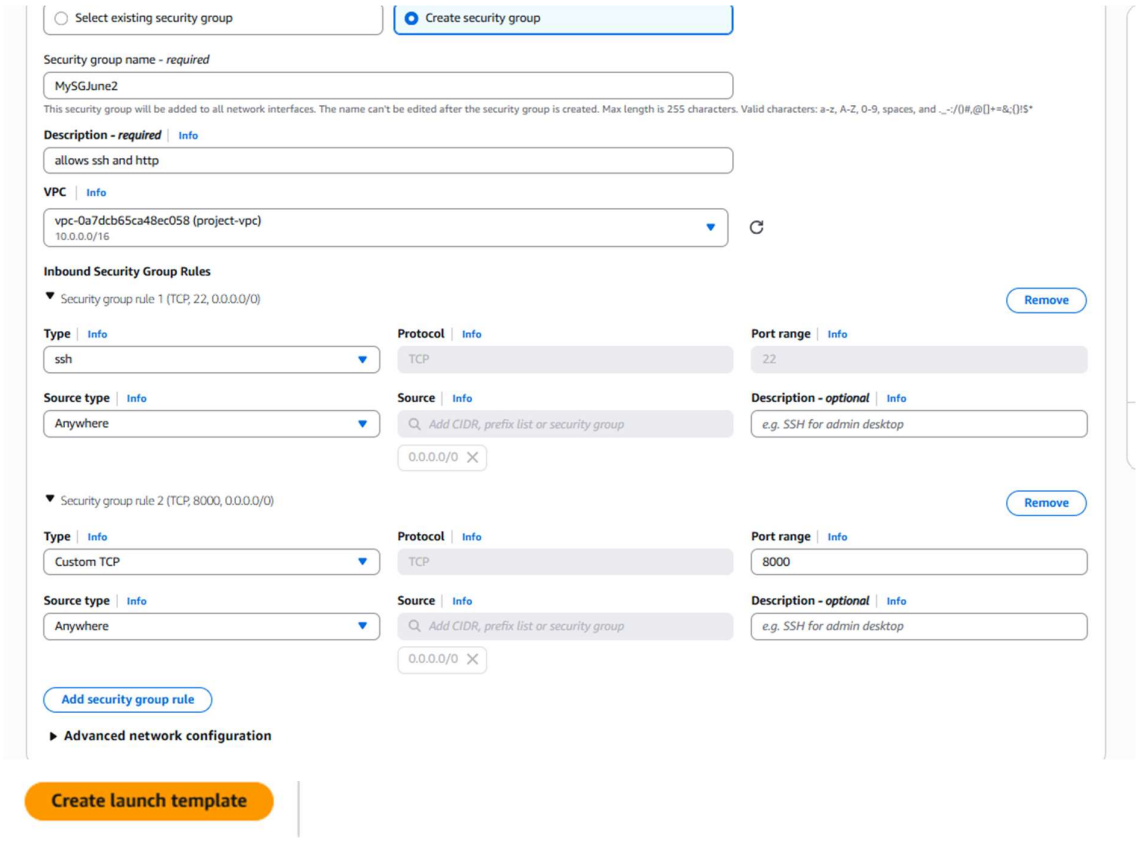
Create the Launch Template as it automatically loads the page in other Tab

Give the Specifications for the Ec2 Instances ..



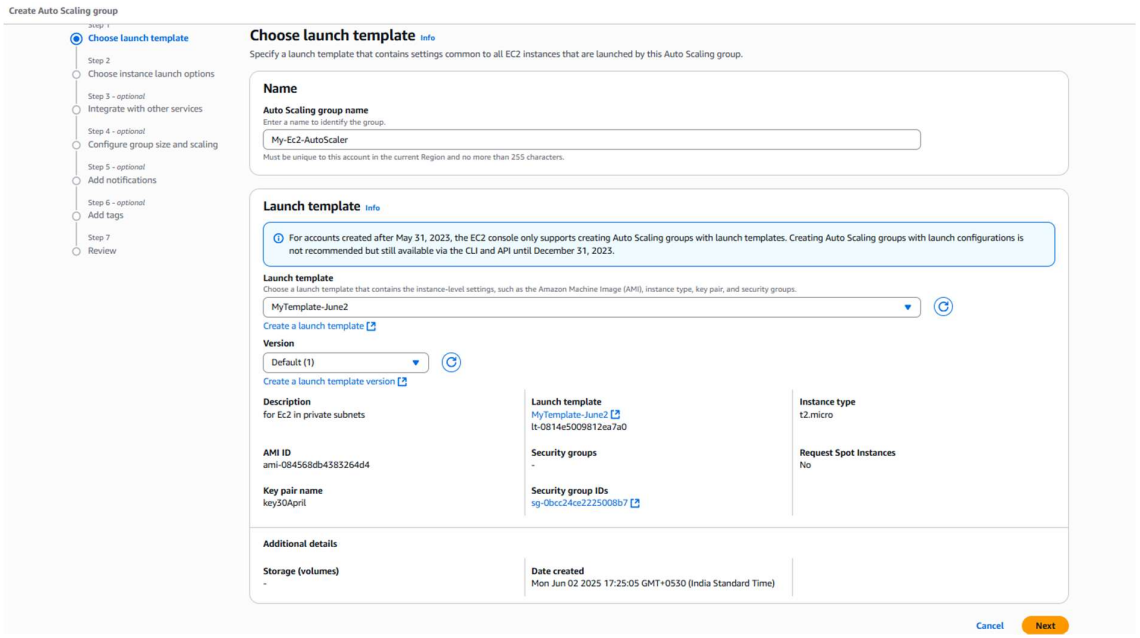


Here Create a Security Group that allows only from PORT 22, PORT 8000(i.e Our Load Balancer only Can Access these Private Ec2 Instaces)



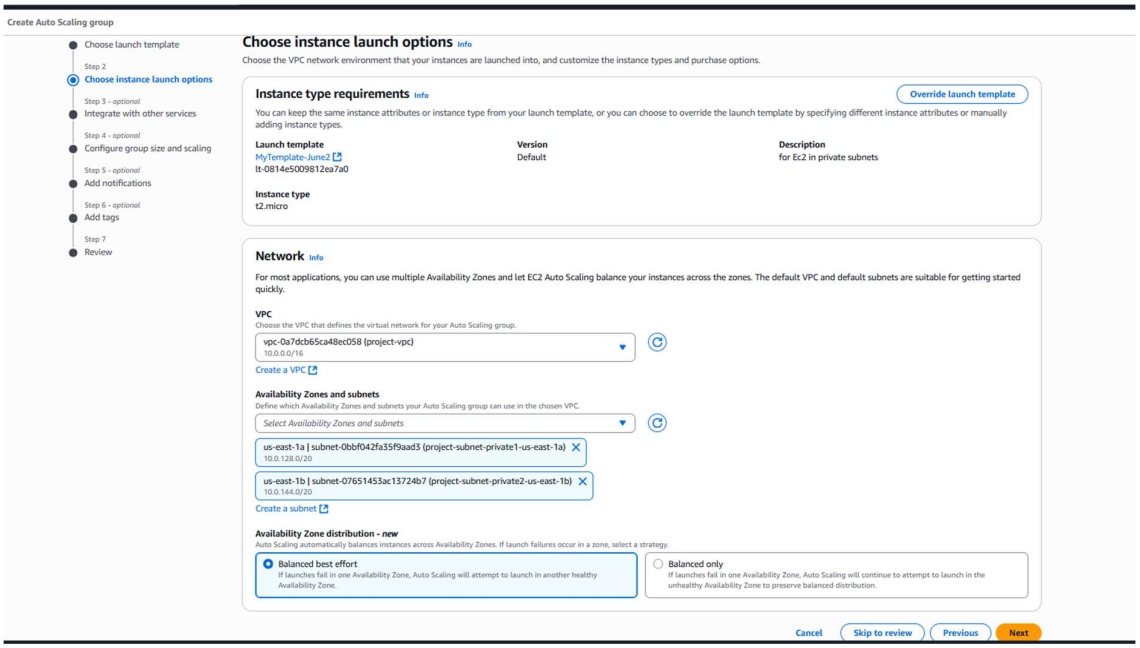
C)

Load the Created Launch Template in the Auto Scaling Section

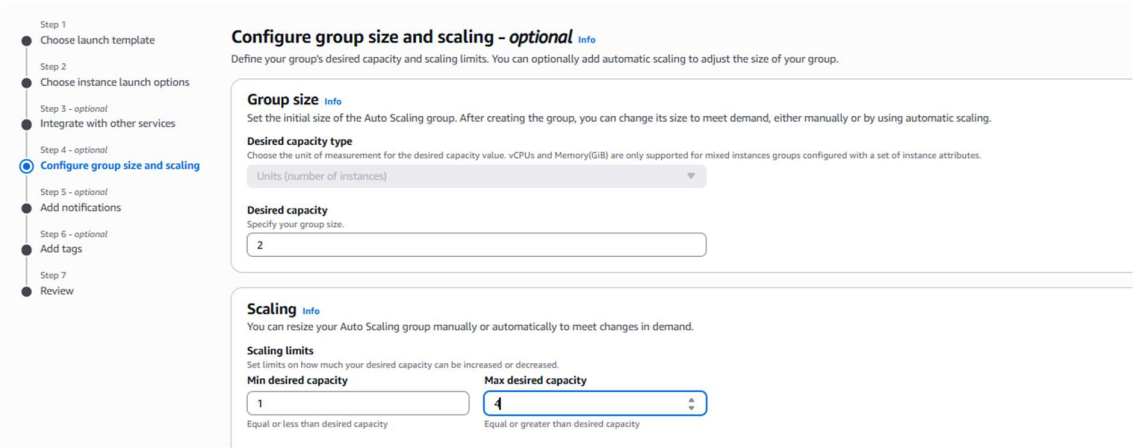


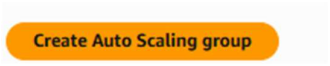
D)

Create AutoScaling in the Private Subnet of Created VPC- private1A and private 1B



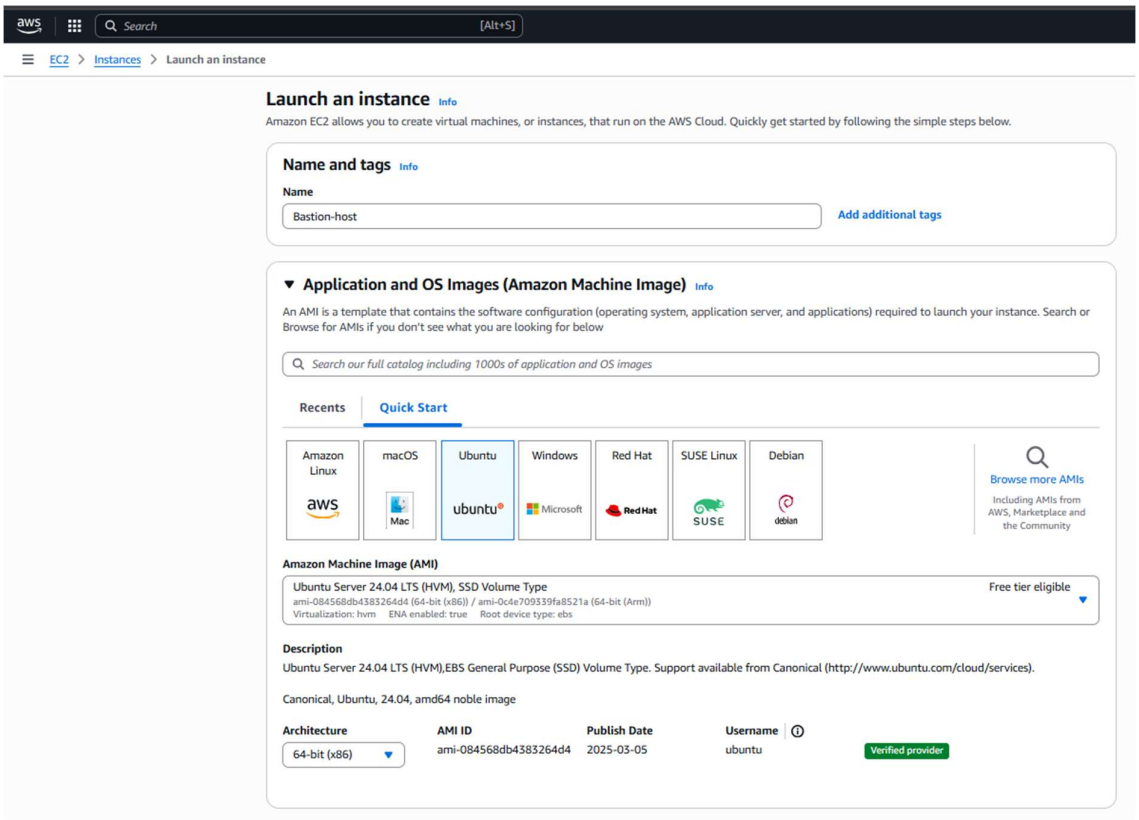
E)Configure group size and Auto Scaling of the Ec2 Instances



F) 

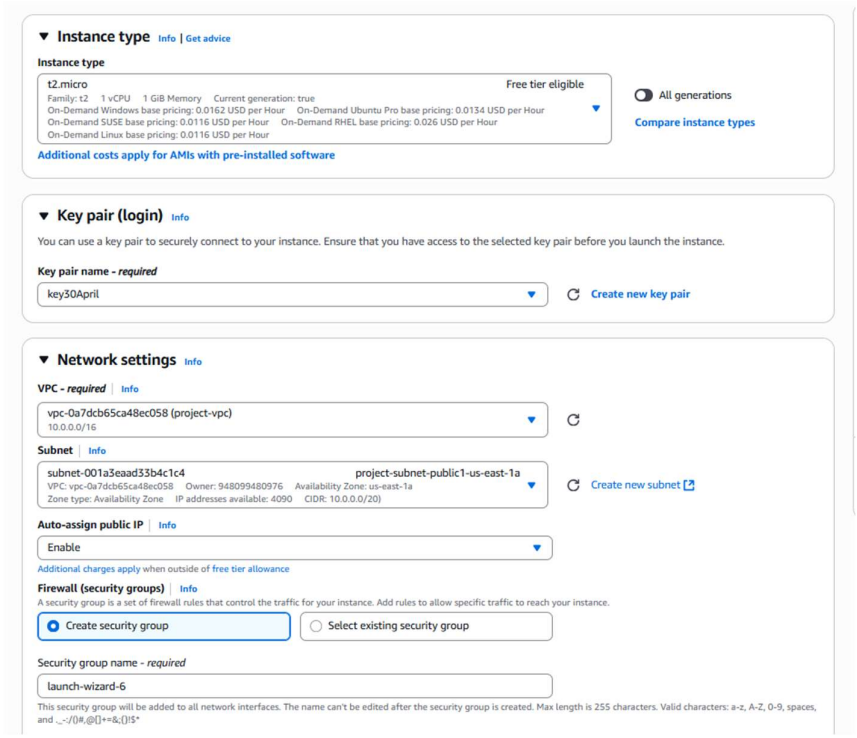
G)

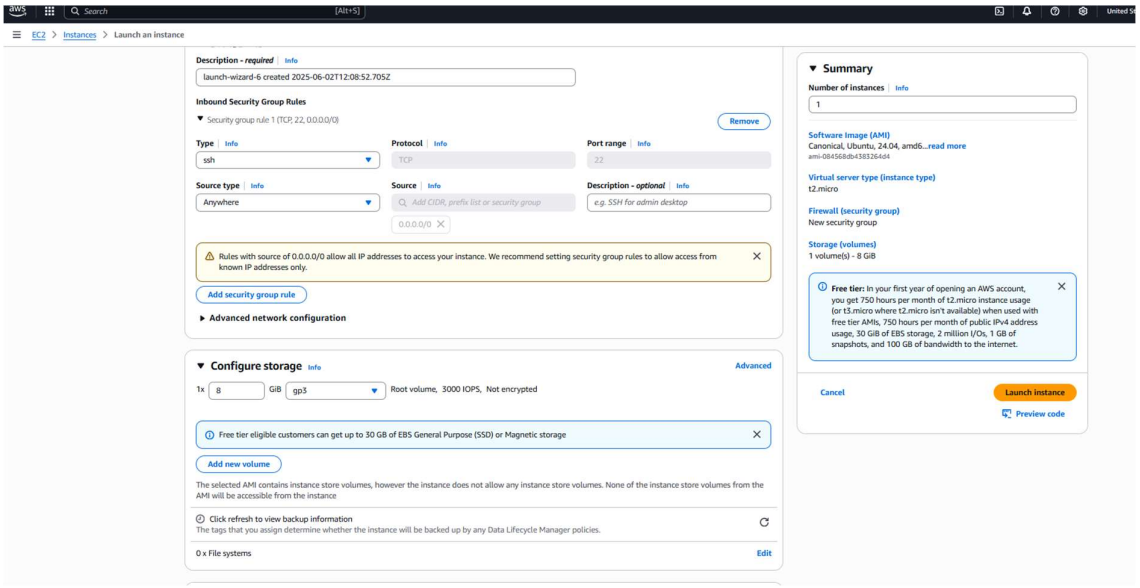
Create another Ec2 Instance - Bastion-Host(Jump Server) in the **Public Subnet** of Our VPC, as from our own PC Cmd-line we need to SSH to Bastion



Be Careful, as now we are putting the Bastion-Host in **OUR VPC** that is Created , and in **the PUBLIC Subnet of that VPC.**

And PUT Auto-Assign public IP - **ENABLE**



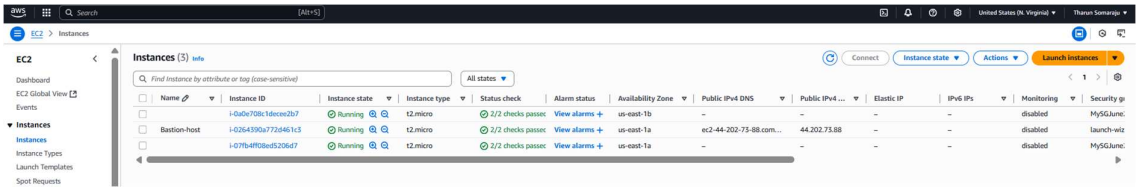


LAUNCH THE Instance.

STEP 3-

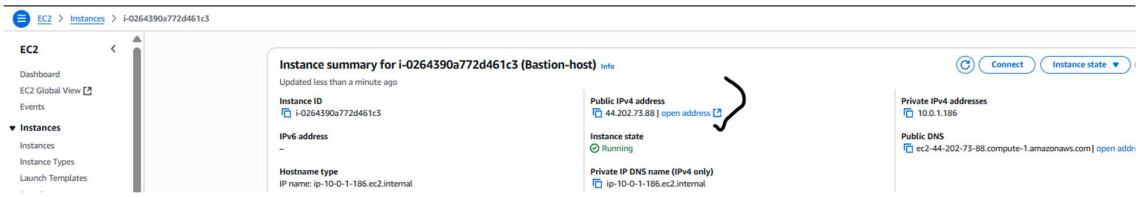
A)

We Can see 3 instance – One is Bastion and two are Private Subnet Instances.



B)

Go to Bastion-host instance and note the Public IP of it.



C)

FOR copying key **key30April.pem** from the Local Pc to Bastion (as we want to login to private subnet from bastion) in the Command Prompt of the PC

scp -i "C:\Users\Somaraju Tharun\Downloads\key30April.pem" "C:\Users\Somaraju Tharun\Downloads\key30April.pem" ubuntu@44.202.73.88:/home/ubuntu/

```
C:\Users\Somaraju Tharun>scp -i "C:\Users\Somaraju Tharun\Downloads\key30April.pem" "C:\Users\Somaraju Tharun\Downloads\
key30April.pem" ubuntu@44.202.73.88:/home/ubuntu/
The authenticity of host '44.202.73.88 (44.202.73.88)' can't be established.
ED25519 key fingerprint is SHA256:u0wpy+nyWoDzTBFeJ4rFQ2sbm+z3n1H4XAfFMx+P0Q8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added '44.202.73.88' (ED25519) to the list of known hosts.
key30April.pem                                                100% 1678     5.3KB/s   00:00

C:\Users\Somaraju Tharun>
```

SUCCESSFULLY COPIED THE Key-Pair to Bastion from the Local PC

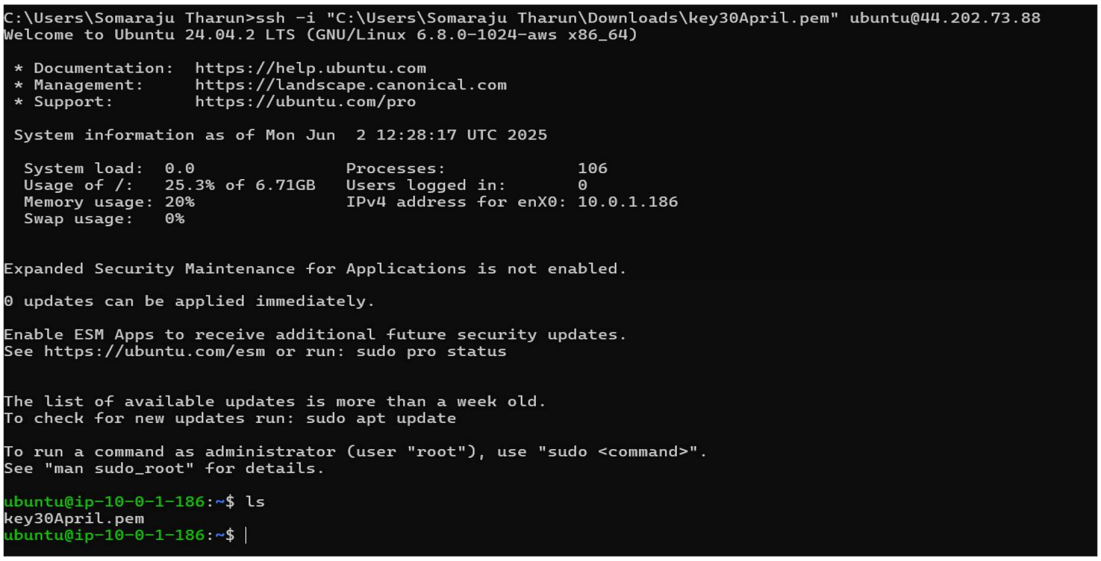
D)

To verify if the key got copied or not

-ssh -i "C:\Users\Somaraju Tharun\Downloads\key30April.pem" ubuntu@44.202.73.88

-$ ls

You will see the key



E)

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

F)

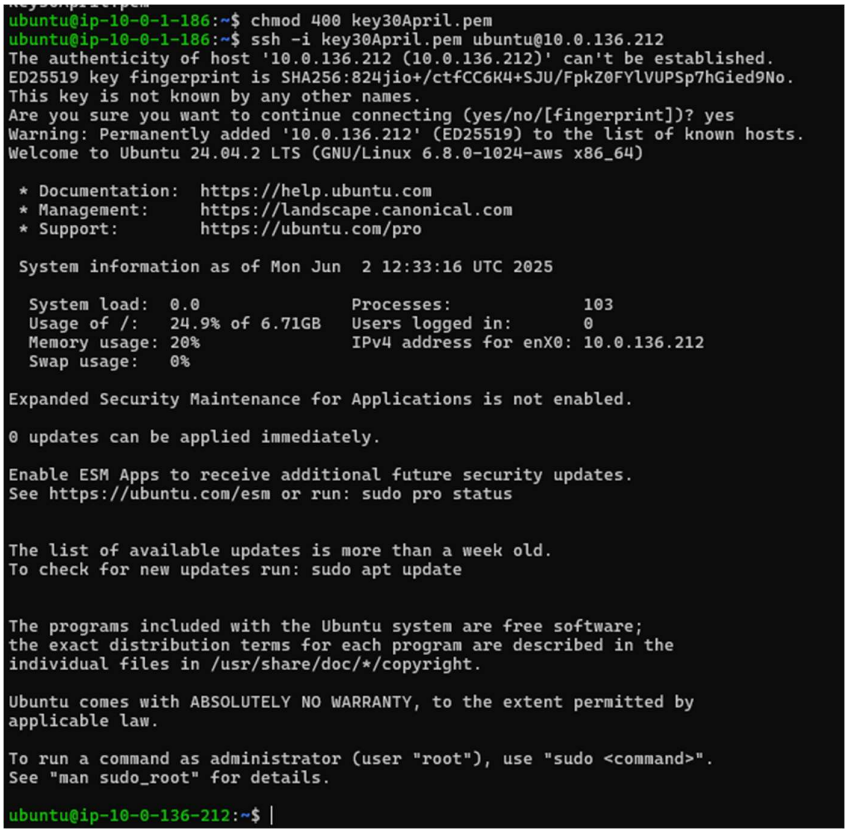
To SSH from the **Bastion Host** into the **private EC2 instance (10.0.151.236)**, use the following command **on the Bastion terminal**:

$ssh -i key30April.pem ubuntu@10.0.136.212

G)

You are Finally in Ec2 instace Terminal –

ubuntu@ip-10-0-136-212:~$



H)

CREATE A HTML PAGE IN THE EC2 TERMINAL

1. Create the file

Run this on your EC2 instance:

bash

CopyEdit

$nano index.html

Paste the following code:

html

CopyEdit

```
<!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>

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

</body>

</html>

3. 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

🌐 **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:**

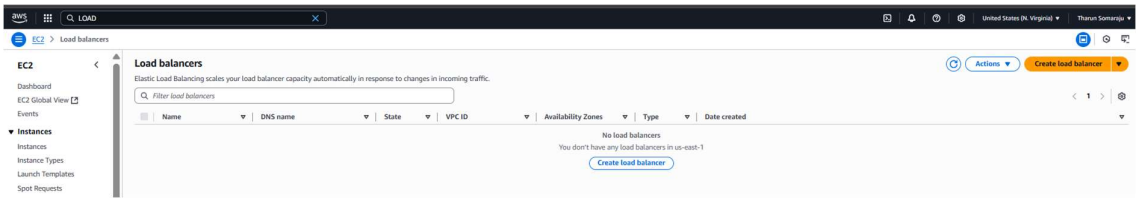
How does EC2 recognize the key-pair?

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.

STEP 4-

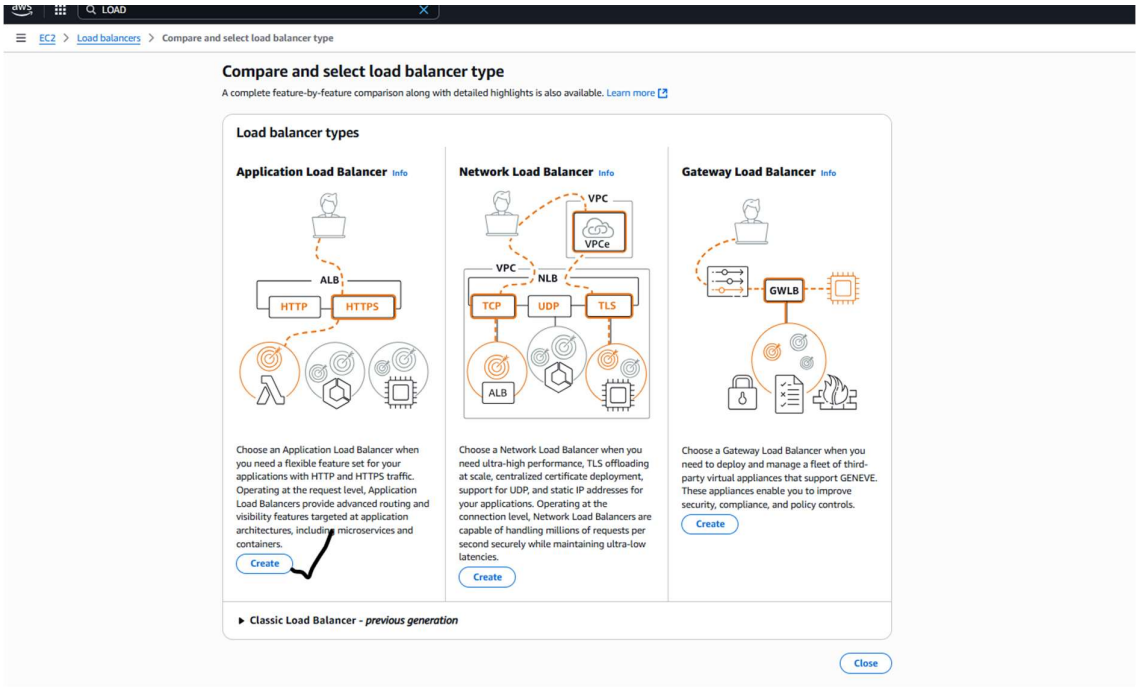
A)

Create LOAD BALANCER

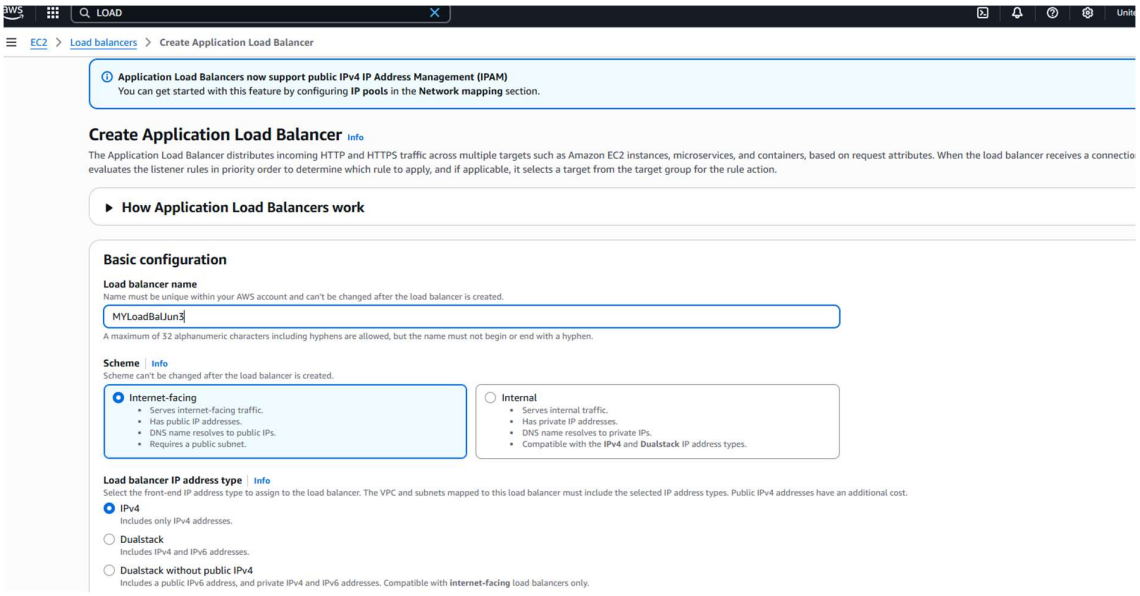


B)

Create APPLICATION LOAD BALANCER

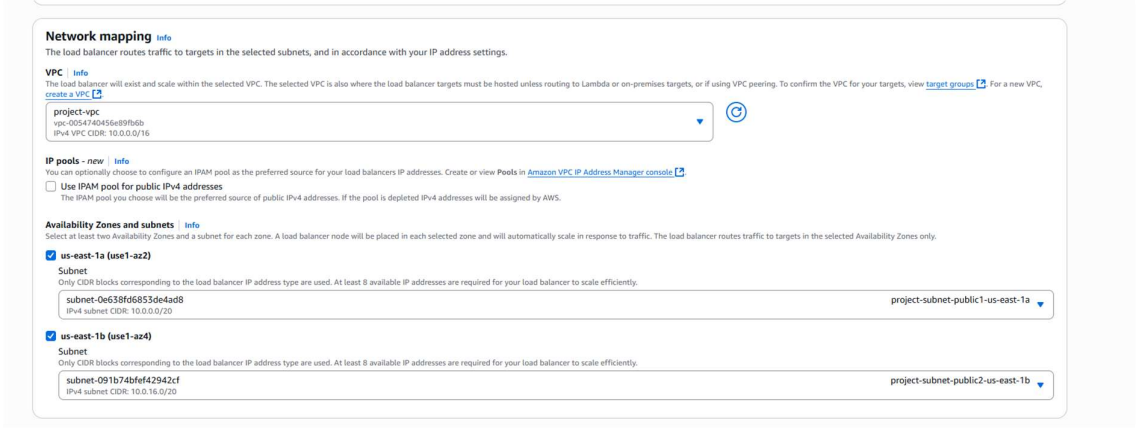


C)



D)

Make sure that Load Balancer in the Public Subnets of OUR VPC



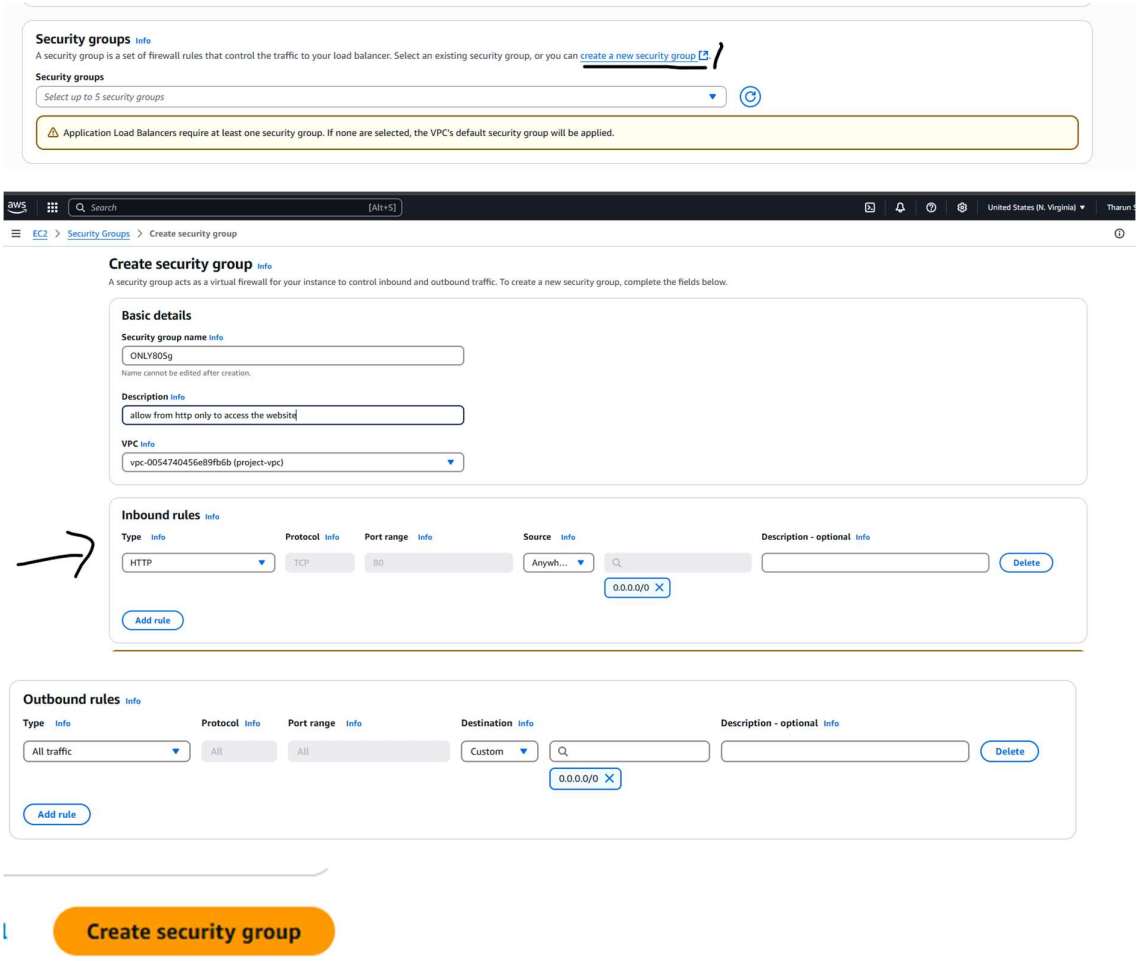
E)

Create A new Security Group for Load Balancer allowing only **PORT 80** because

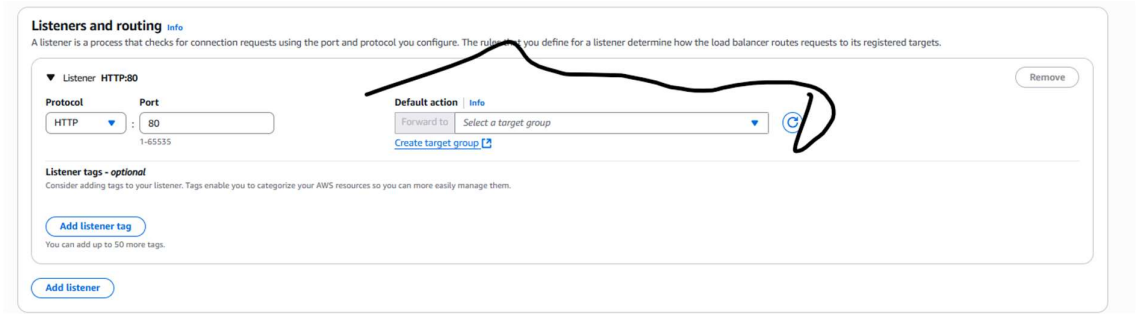
🔒 Why only allow port 80 on LB's Security Group?

- **Port 80 (HTTP)** is used by users to access the website via the Load Balancer.
- The LB **never needs SSH (port 22)** or **app ports (like 8000)** from the public.

- Limiting to port 80 reduces your **exposure to unnecessary attacks**.



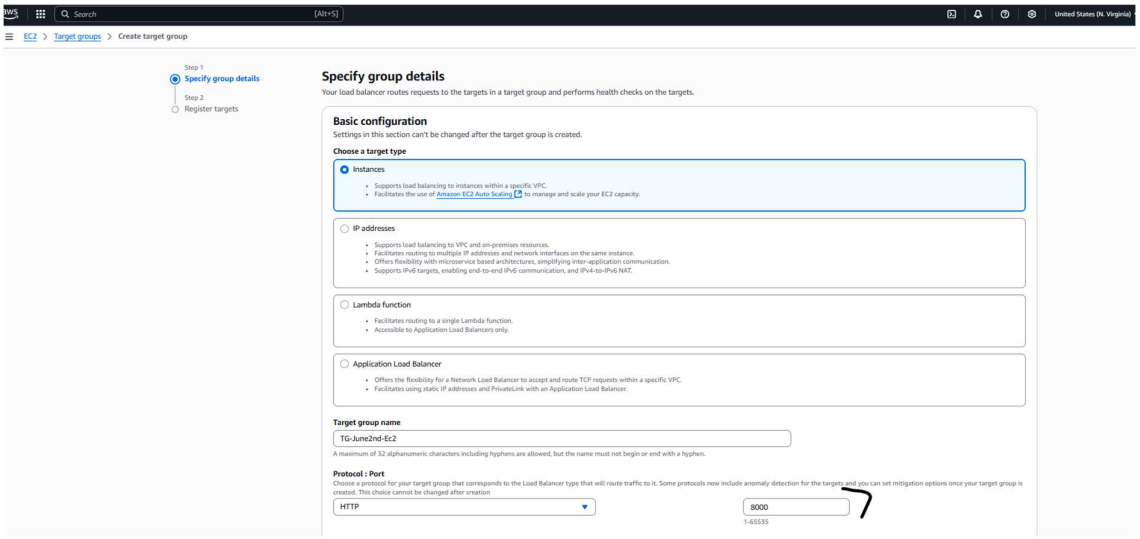
F) Add the Security Group that is Created i.e PORT 80



G) Specify the Target Groups(Our Private Subnet Ec2 Instances)

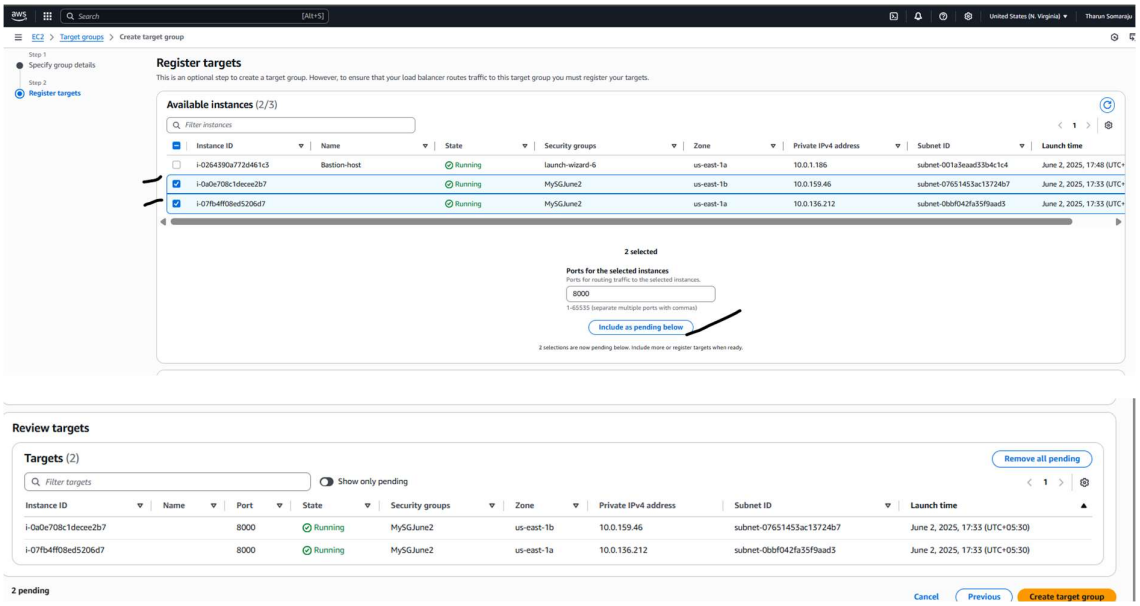
And We are Specifying the PORT 8000 For the Target Group Because , The Inbound Security Group for Instances are **PORT 8000** AND PORT 22..

So Therefore we put the PORT 8000 here for LB to Contact Ec2



H)

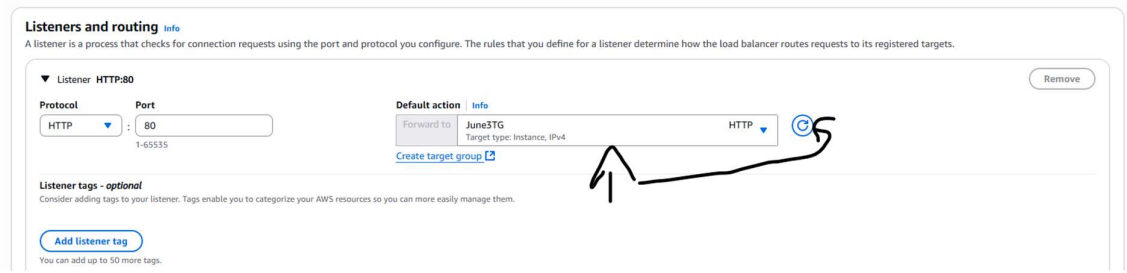
Put the Targets as our Private Instances and Press include as Pending Below button



PRESS **CREATE TARGET GROUP**

I)

Now add the Created **Target Group** to the Load-Balancer

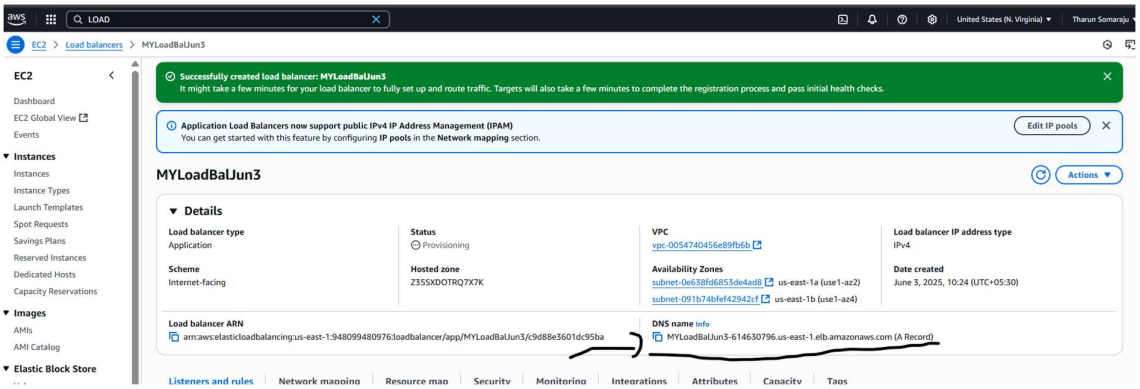


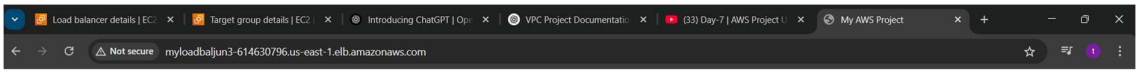
PRESS **CREATE THE LOAD-BALANCER**

J)

After Successful creation of Load Balancer

Now Go Copy the URL of Load Balancer for testing..and check it in the Chrome or any browser ..





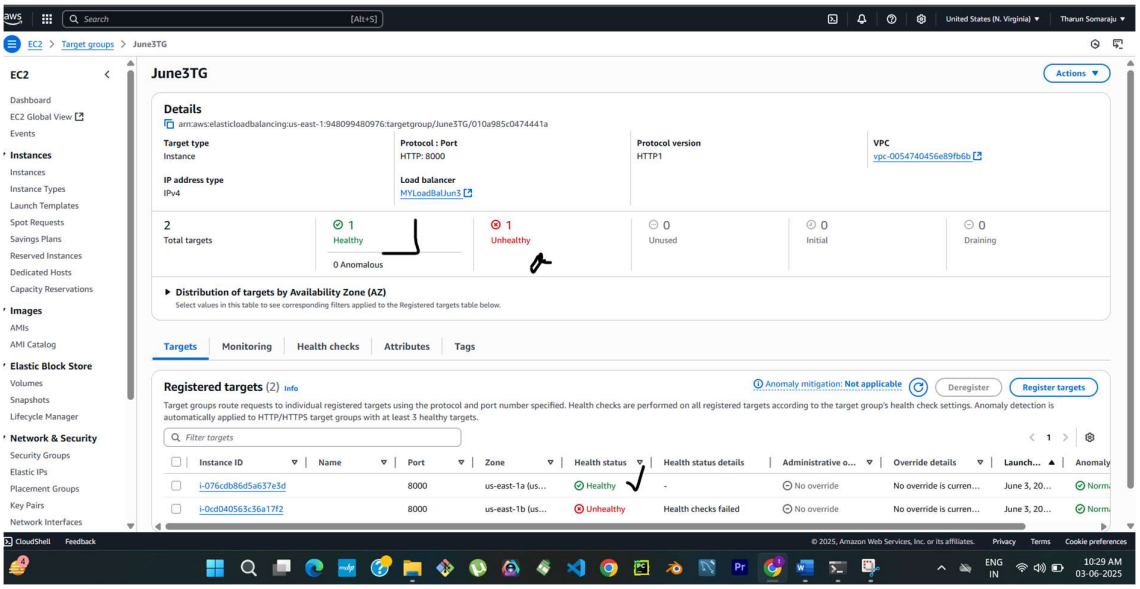
Welcome to My AWS Created on June 3rd -2025 BY THARUN SOMARAJU

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

Finally we can see the Website ,

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



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……