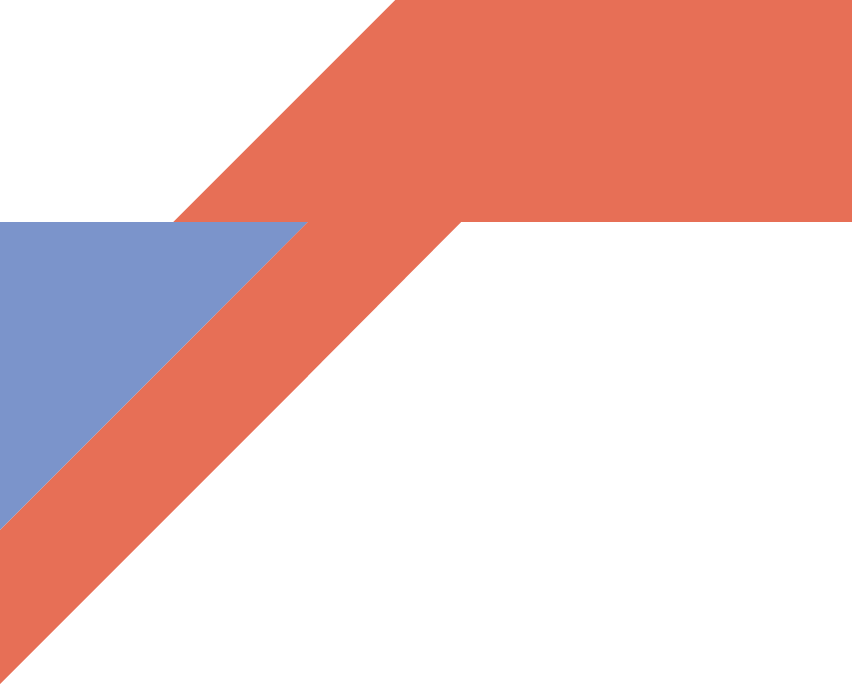


Project made by Rishabh Jha



VPC Implementation

AWS Project





Summary

**Project Summary: AWS VPC Infrastructure Management**

This project focuses on designing, implementing, and managing a Virtual Private Cloud (VPC) within the AWS environment to ensure secure, scalable, and efficient network infrastructure. The VPC is configured to host applications and services with both public and private subnets, providing a controlled and isolated environment.

Key components include:

**VPC Design:** Creation of a custom VPC with multiple availability zones for high availability and fault tolerance.

**Subnet Configuration:** Establishment of public and private subnets to segment application layers effectively.

**Security Management:** Implementation of security groups and network access control lists (NACLs) to regulate inbound and outbound traffic.

**Routing and Connectivity:** Configuration of internet gateways, NAT gateways, and route tables to manage external and internal connectivity.

**Monitoring and Maintenance:** Integration with AWS CloudWatch for real-time monitoring and management of network resources.

The project ensures robust network security, optimized performance, and seamless communication between application instances, demonstrating best practices in AWS cloud architecture.

Contents

Summary .........................................................................1

Architechture Diagram......................................................................... 3

Step by Step Instructions .........................................................................4

1.1 VPC Creation and Configuration…………………………………............... 4

1.2 Create an Auto Scaling Group……....................................................9

1.2.1 Create Launch Template .............................................................. 11

1.3 Install Application in Server................................................................18

1.3.1 Create Bastion Host .…………….………………………………........................................18

1.3.2 Add Key Pair to Bastion Host for Connecting to ASG Instances .………. 23

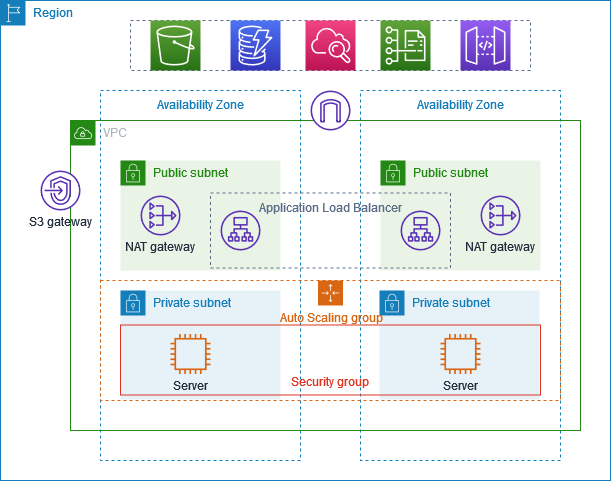
1.4Create Load Balancer……………………………………………………………...............30

1.4.1 Create Target Group…………………………….…………………………………............... 34

1.4.2 Troubleshooting - Error……..………………………………………………............... 36

Conclusion .........................................................................39

Architechture Diagram



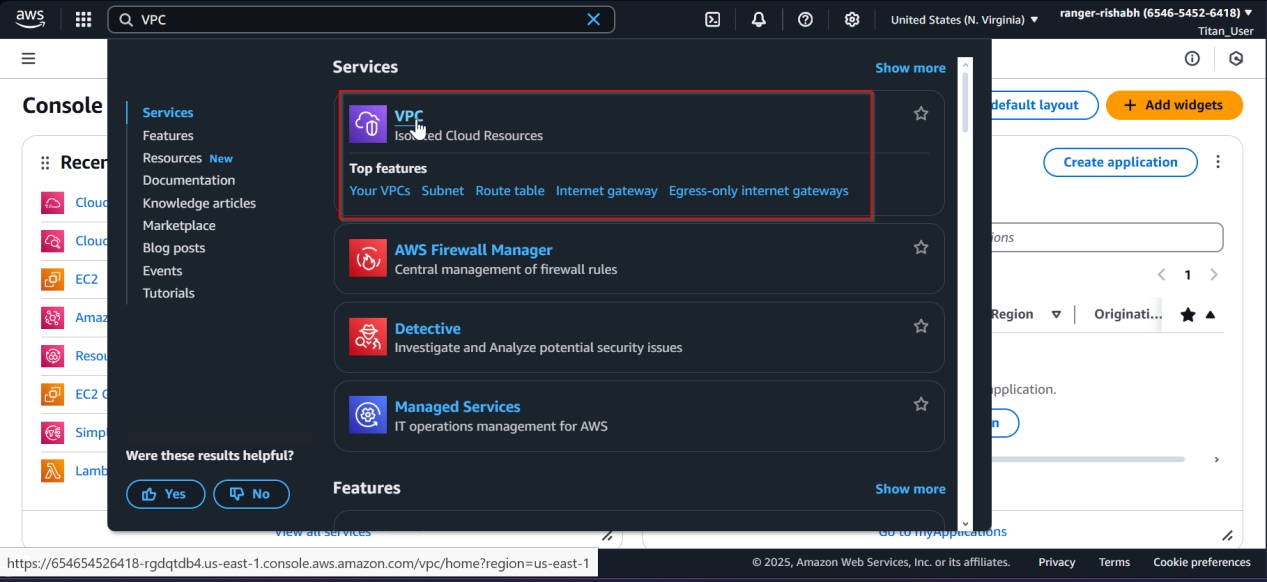
Step by Step Instructions

**Step 1: VPC Creation and Configuration**:

1. **Search for VPC**

In the AWS Management Console, navigate to the search bar at the top.

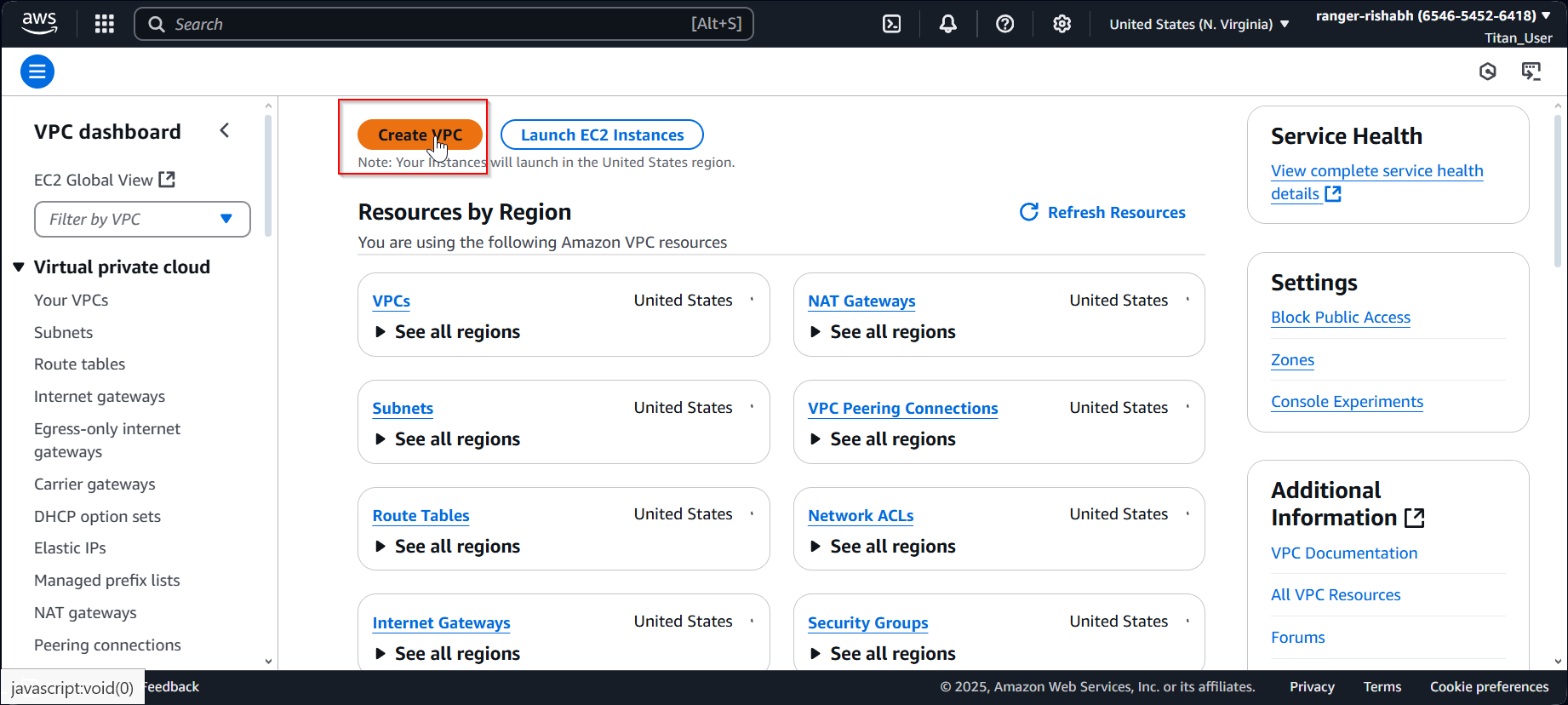
Type **"VPC"** and select **"VPC"** from the search results.



**Initiate VPC Creation**

Click on **"Create VPC"** in the VPC dashboard.

Select **"VPC and more"** to access advanced configuration options.



**Configure VPC Settings**

Enter the **VPC Name** as **"aws-prod"** to label your VPC.

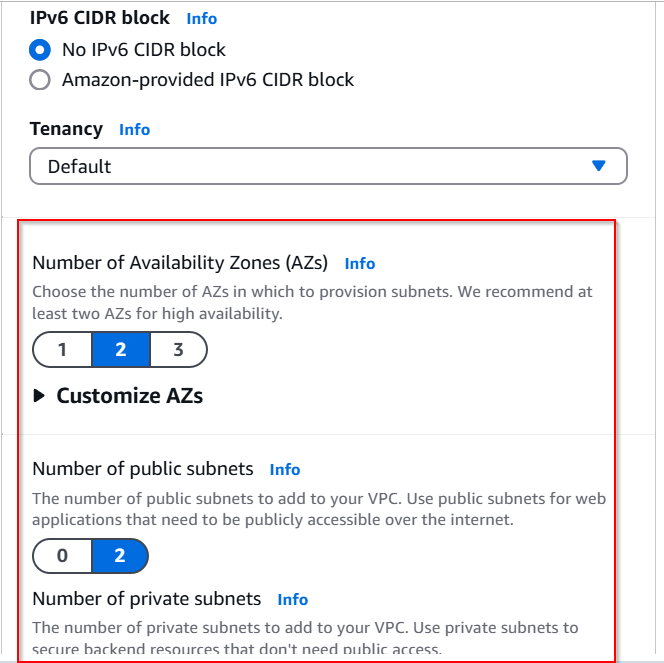
Keep the **same subnet settings** as the default VPC for consistency.

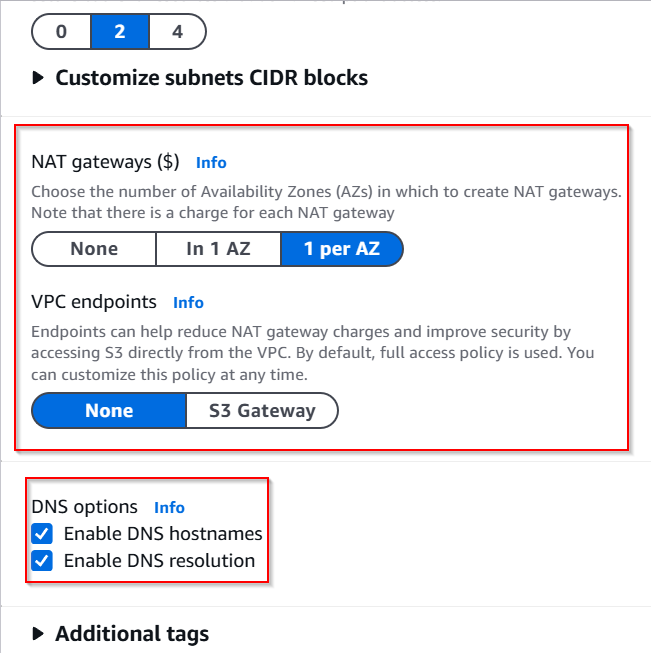
Set **IPv6 addressing** to **None** (disable IPv6).

**Define Availability Zones and Subnets**

Select **2 Availability Zones (AZs)** for high availability.

Configure **2 Public Subnets** and **2 Private Subnets**, ensuring each AZ has one of each.





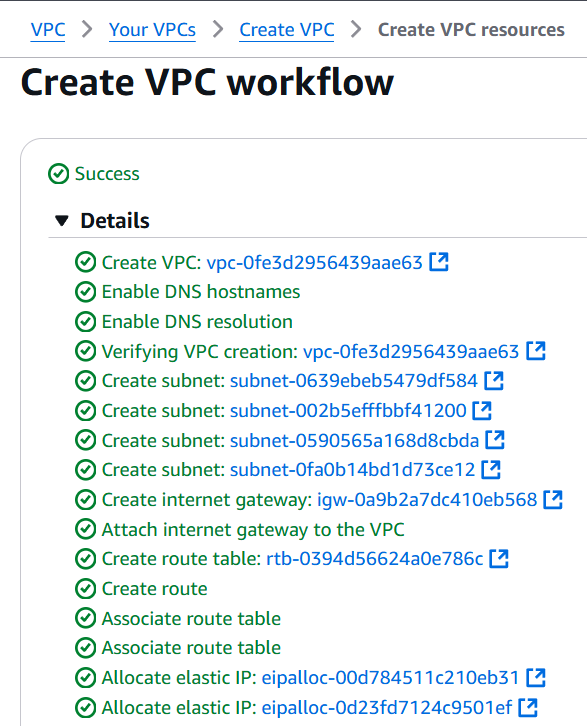
**Additional Configuration**

Do **not** create a **VPC endpoint** at this stage.

**Create the VPC**

Review the settings to ensure accuracy.

Click **"Create VPC"** to complete the process

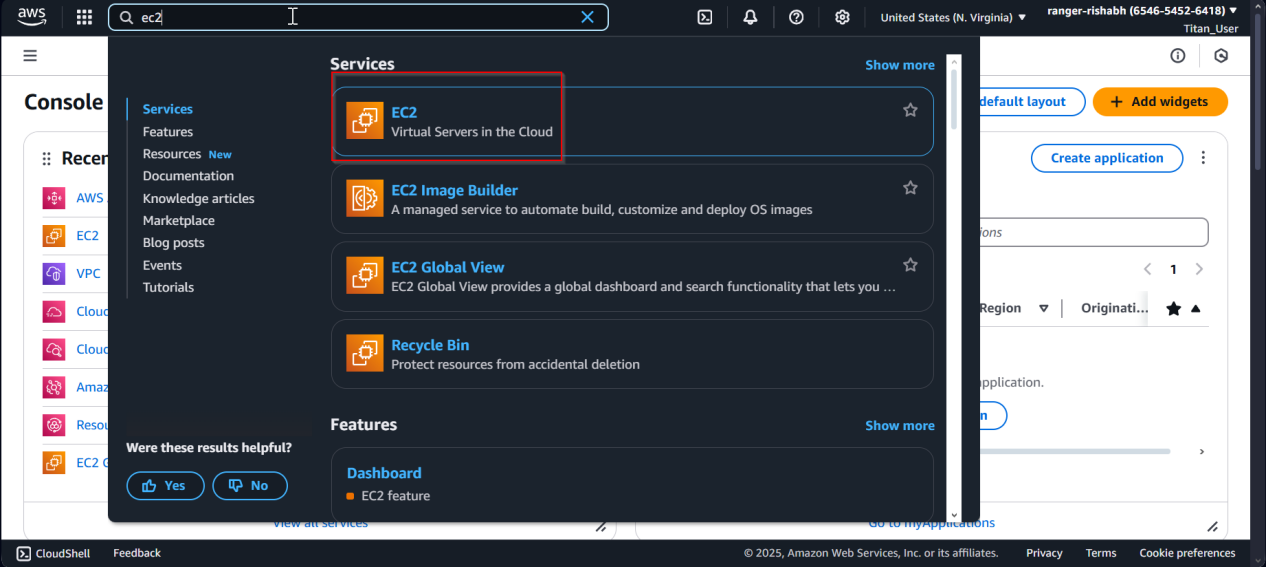


**Step 2: Create an Auto Scaling Group**

**Navigate to EC2 Dashboard**

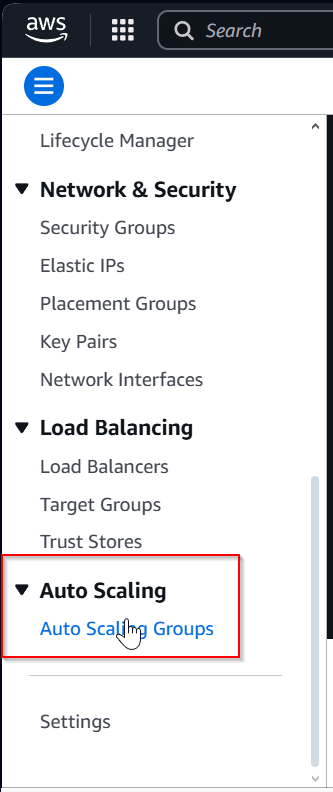
Go to the **AWS Management Console**.

Search for **"EC2"** and select **"EC2"** from the search results.



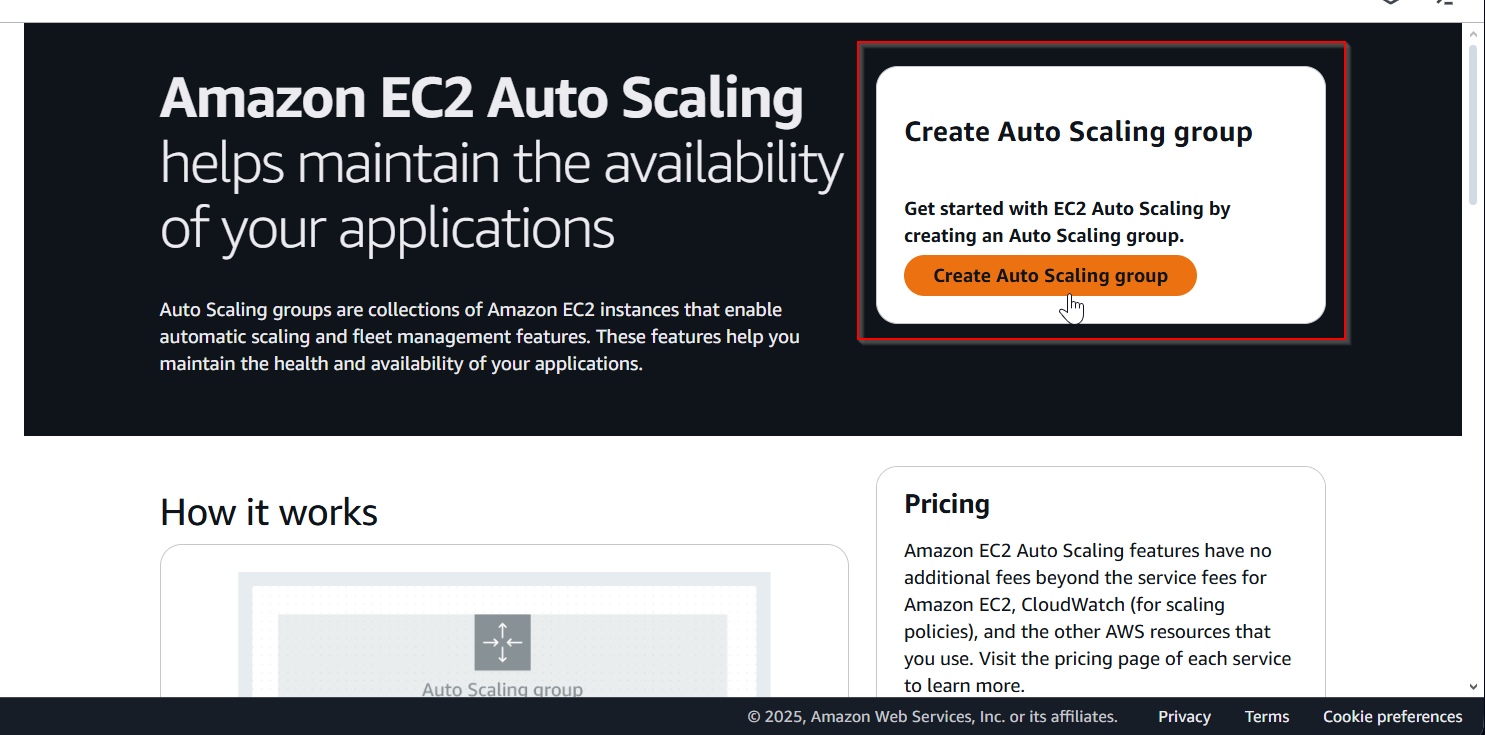
**Access Auto Scaling Groups**

In the left-hand panel, find and click on **"Auto Scaling Groups"** under the **Instances** section.



**Create Auto Scaling Group**

Select **"Create Auto Scaling Group"** to start the process.

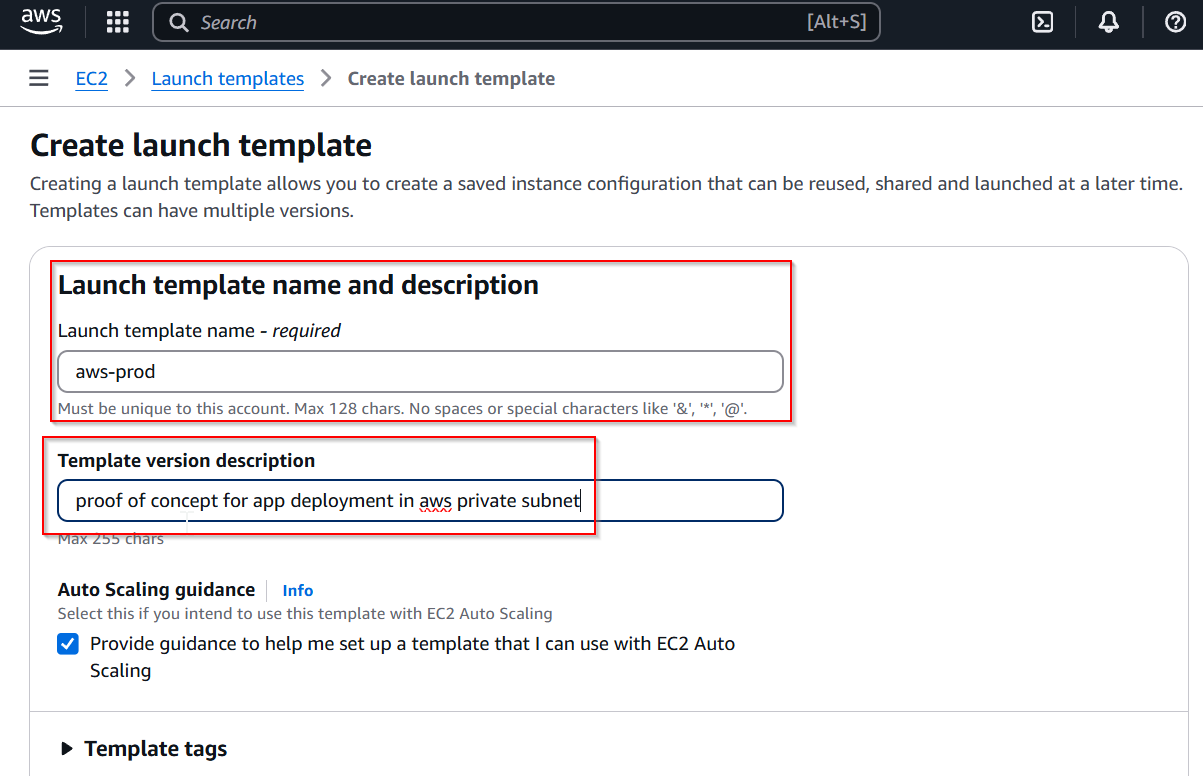


**Step 2 (Subsection): Create Launch Template**

### ****Provide Basic Information****

**Name**: Enter aws-prod.

**Description**: Add Proof of concept for app deployment in AWS private subnet.



### ****Select Amazon Machine Image (AMI)****

Choose **Ubuntu** as the AMI for the instances.



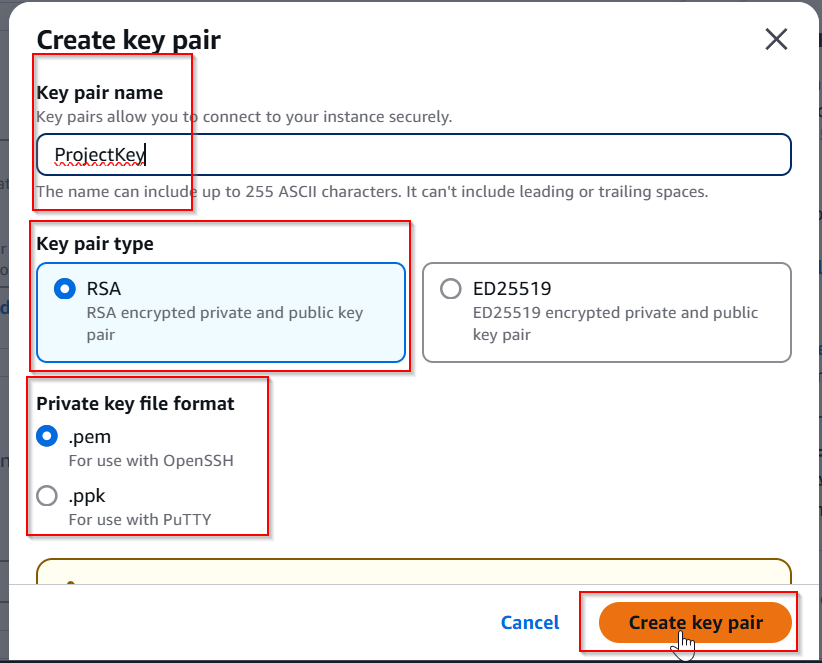
### 3. ****Choose Instance Type****

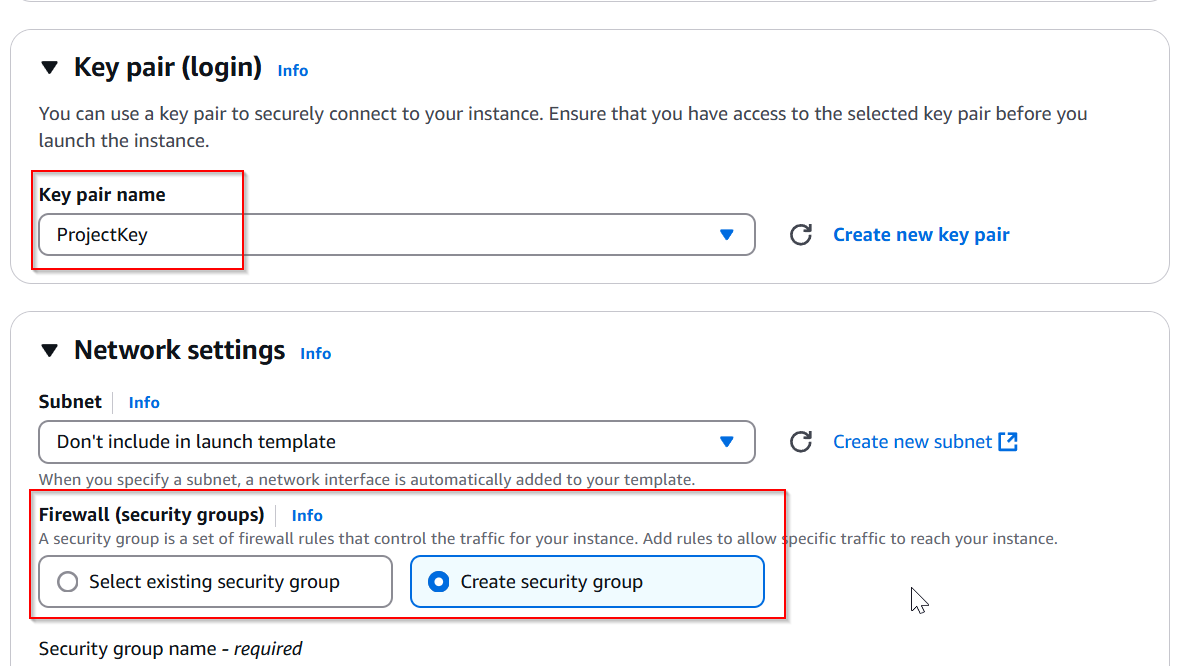
Select t2.micro for the instance type, which is suitable for testing purposes under the AWS Free Tier.

### 4. ****Configure Key Pair****

Choose an existing **Key Pair** for SSH access.

If a key pair doesn't exist, click **"Create Key Pair"** to generate one, download it securely, and store it safely.





**Create New Security Group**

Under the **Network Settings** section, click on **"Create Security Group"**.

**Provide Security Group Details**

**Name**: Enter prod-example.

**Description**: Enter Allow SSH access.

Select the **VPC** created in **Step 1** (aws-prod).

**Add Inbound Rules**

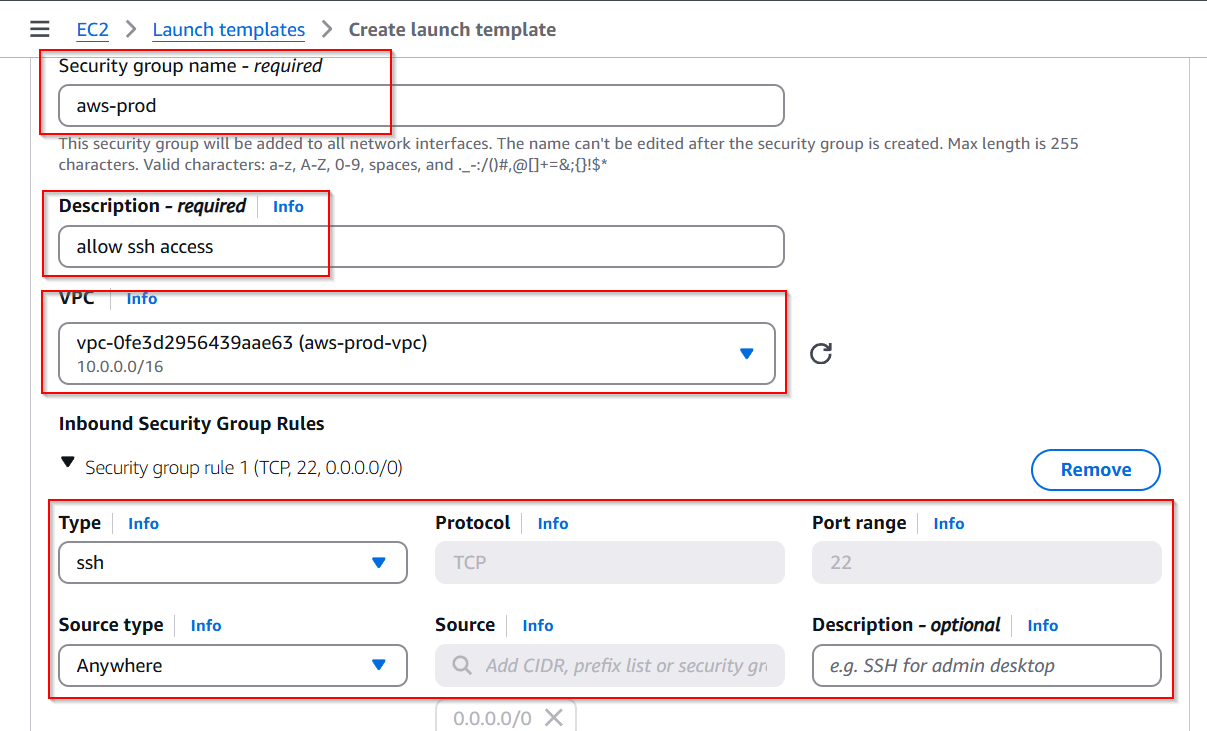
**Rule 1: SSH Access**

Protocol: **SSH**

Port Range: **22**

Source: **Anywhere (0.0.0.0/0)**

This allows secure remote access via SSH.



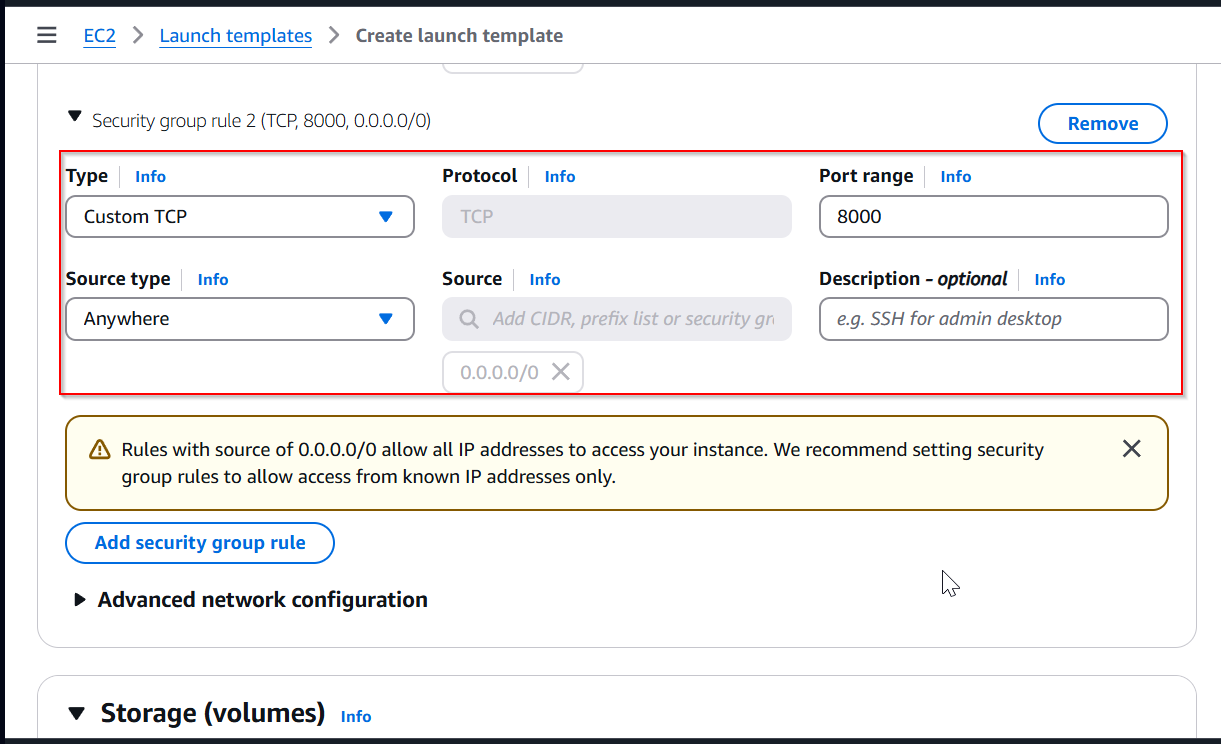
**Rule 2: Custom TCP Access**

Protocol: **TCP**

Port Range: **8000**

Source: **Anywhere (0.0.0.0/0)**

This enables access to the application running on port **8000**.



**Create Launch Template**

Review all the configurations.

Click **"Create Launch Template"** to complete the process.

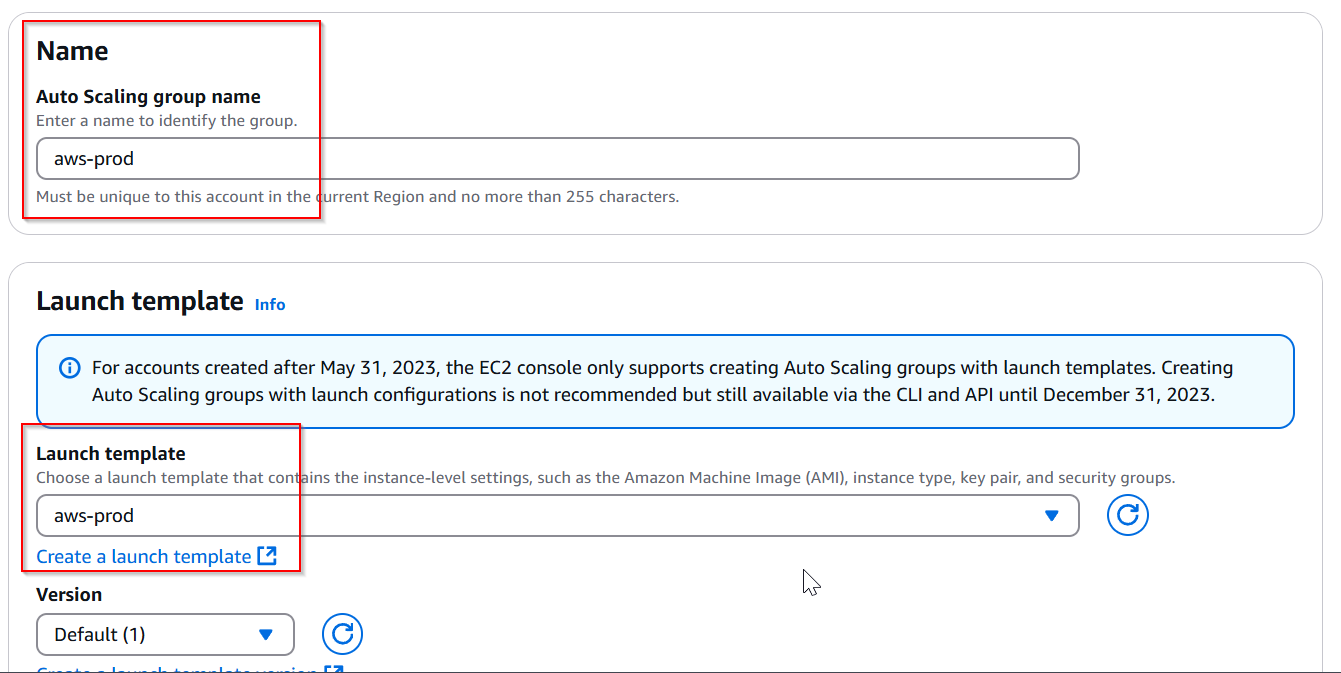
**Auto Scaling Group (ASG)**:

### 1. ****Select Launch Template****

After creating the ASG, you will be prompted to select a **Launch Template**.

Choose the template named aws-prod (created in the previous step).

Click on **"Next"** to proceed.



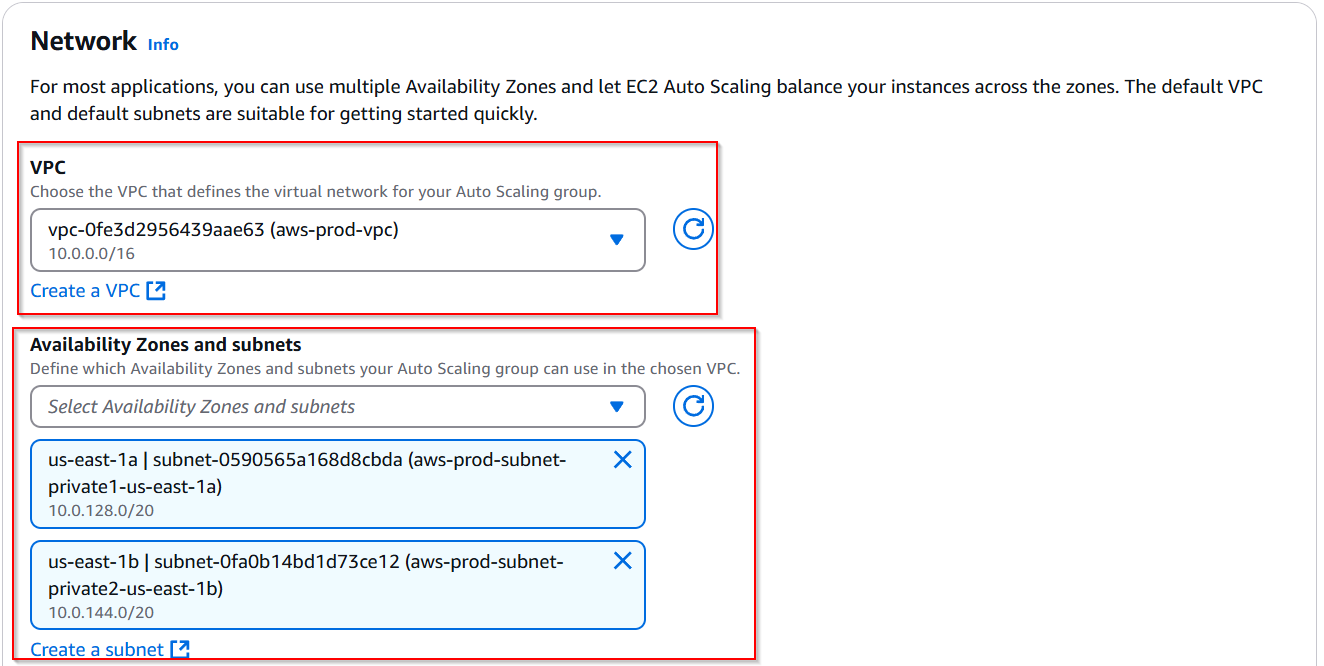
### ****Configure Network Settings****

**Select VPC**: Choose the VPC named aws-prod created in **Step 1**.

**Select Availability Zones (AZs)**: Choose the **2 private subnets** associated with the VPC across both AZs.

This ensures the instances are deployed in both zones for better availability and fault tolerance.

Click on **"Next"** to continue.



### ****Skip Load Balancer Configuration****

On the **"Configure Load Balancer"** page, select **"No load balancer"** since this step does not involve using a load balancer.

Click on **"Next"** to move forward.

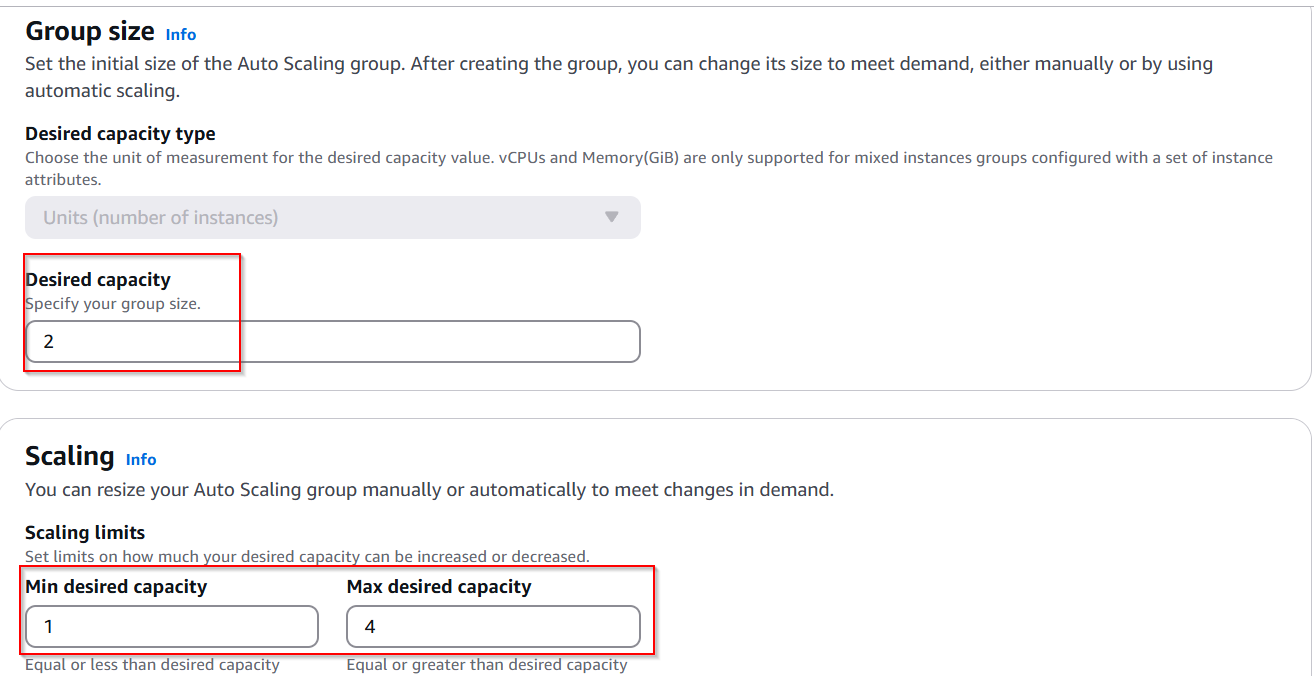
### ****Set Desired, Minimum, and Maximum Capacity****

**Desired Capacity**: Set to 2 instances (the number of instances you want running under normal conditions).

**Minimum Capacity**: Set to 1 instance (ensuring at least one instance is running).

**Maximum Capacity**: Set to 4 instances (allowing the ASG to scale up to this limit if needed).

Click on **"Next"** to proceed.



### ****Configure Scaling Policy****

On the **"Configure Scaling Policies"** page, select **"None"** as no specific scaling policy will be applied for this proof of concept.

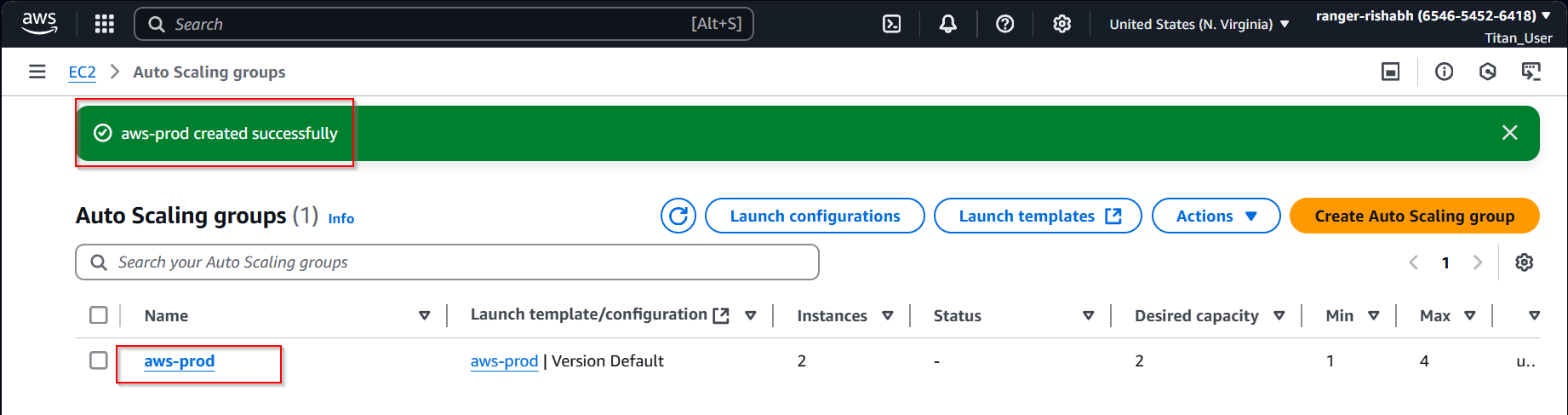
Click on **"Next"** to proceed.

### ****Review and Launch Auto Scaling Group****

Review all the settings you have configured.

Ensure the launch template, VPC, subnets, instance capacity, and scaling policies are correctly set.

Click on **"Launch Auto Scaling Group"** to complete the setup.



**Step 3: Install Application in Server**

**Subsection: Create Bastion Host**

A Bastion Host acts as a secure gateway that allows you to access instances in the private subnet using SSH. Follow these steps to create a Bastion Host in the AWS environment.

### ****Navigate to EC2 Dashboard****

Go to the **AWS Management Console**.

Search for **EC2** in the search bar and open the **EC2 Dashboard**.

Click on **Instances** from the left-hand menu.

### ****Launch a New Instance****

Select **Launch Instances**.

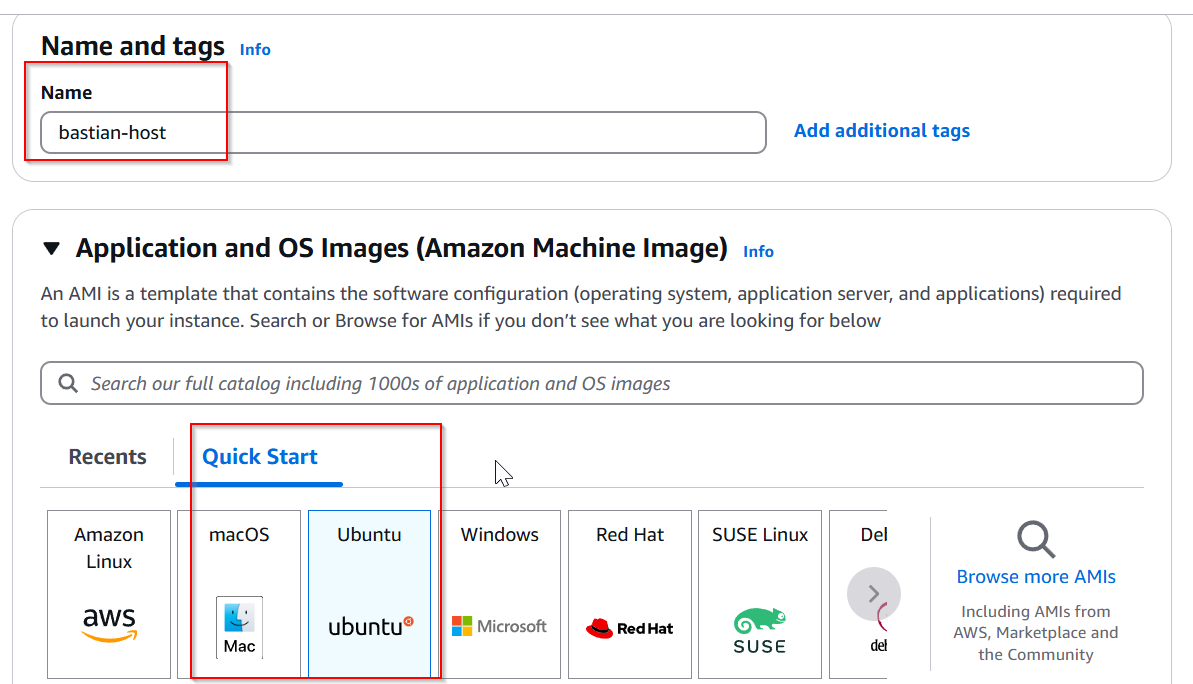
Provide a name for the instance, e.g., Bastion Host.

### 

### ****Choose an Amazon Machine Image (AMI)****

Select **Ubuntu** as the AMI.

Ensure it is an appropriate version like **Ubuntu Server 20.04 LTS (HVM), SSD Volume Type**.



### ****Select Instance Type****

Choose t2.micro (suitable for small-scale applications and included in the AWS Free Tier).

### 

### ****Configure Key Pair****

Select an existing **Key Pair** or create a new one.

Ensure you download and securely store the private key (**.pem** file) for SSH access.

### ****Configure Network Settings****

Under **Network Settings**:

Choose the VPC created in **Step 1** (aws-prod).

Select one of the **Public Subnets**.

Enable **Auto-assign Public IP** to ensure external connectivity.

### 

### ****Set Security Group****

Choose **Create a new Security Group** or select an existing one that allows SSH access.

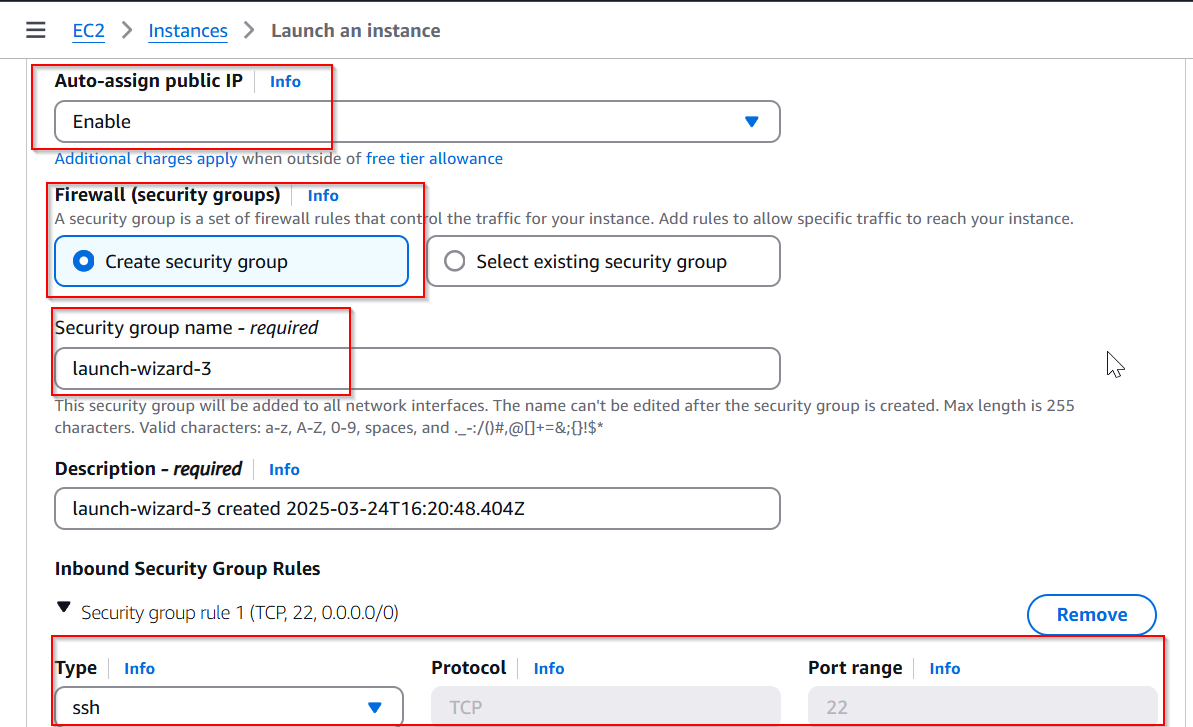
Configure the following inbound rule:

**Type**: SSH

**Protocol**: TCP

**Port Range**: 22

**Source**: Anywhere (0.0.0.0/0) or restrict it to your IP for better security.

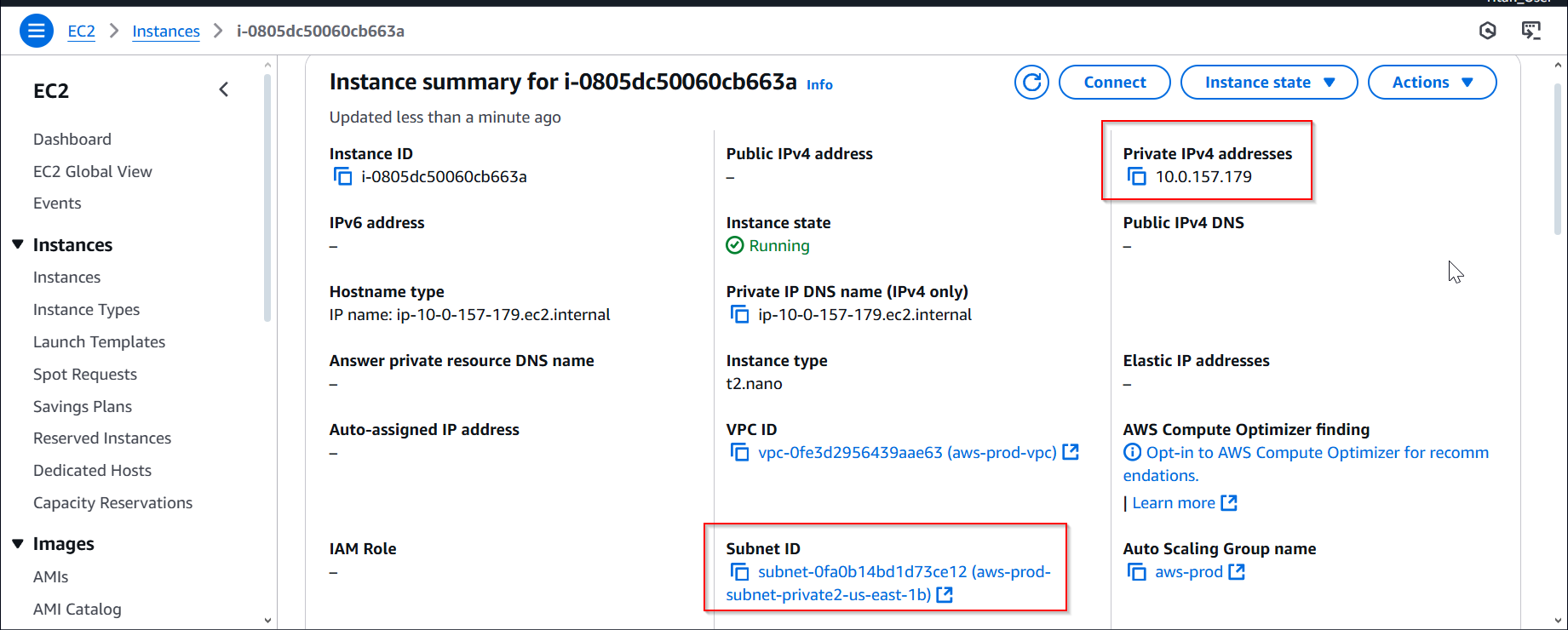


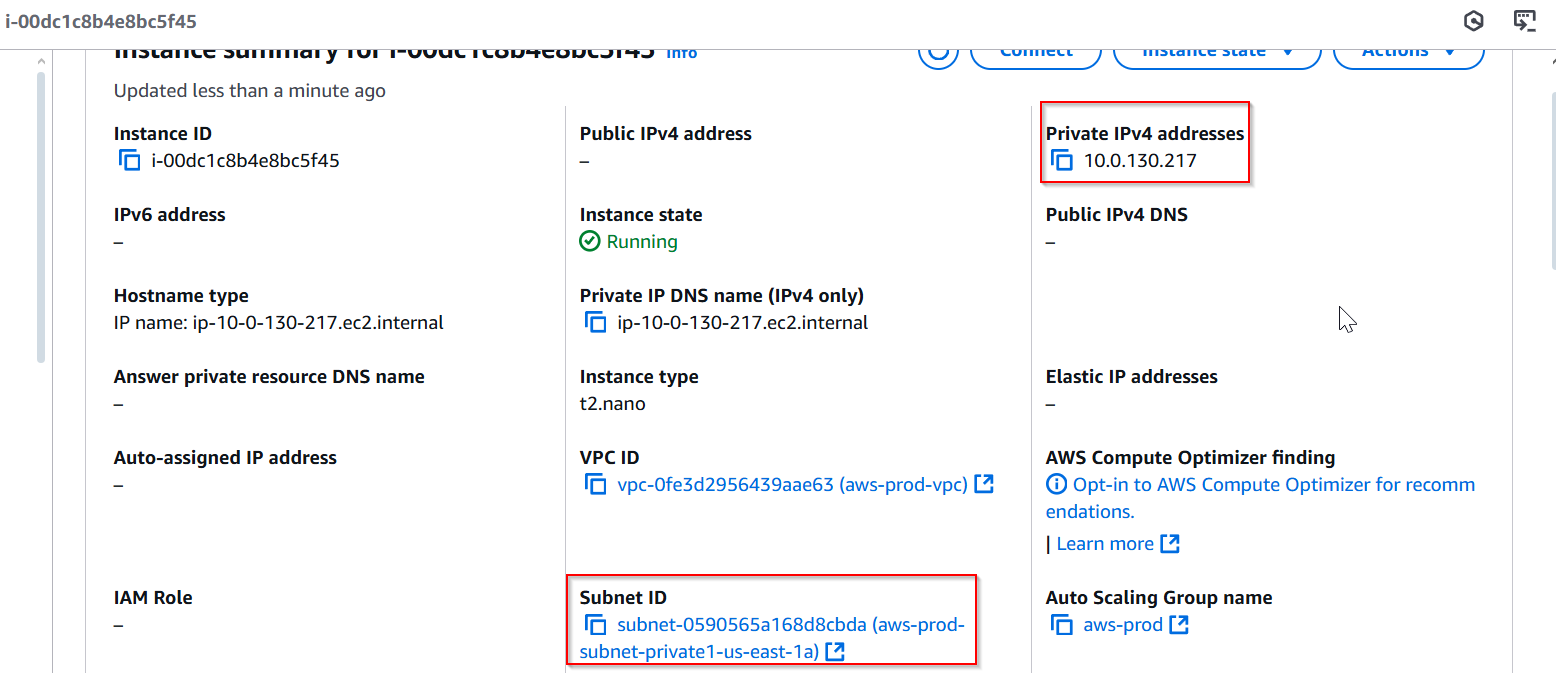
### ****Launch the Instance****

Review all configurations.

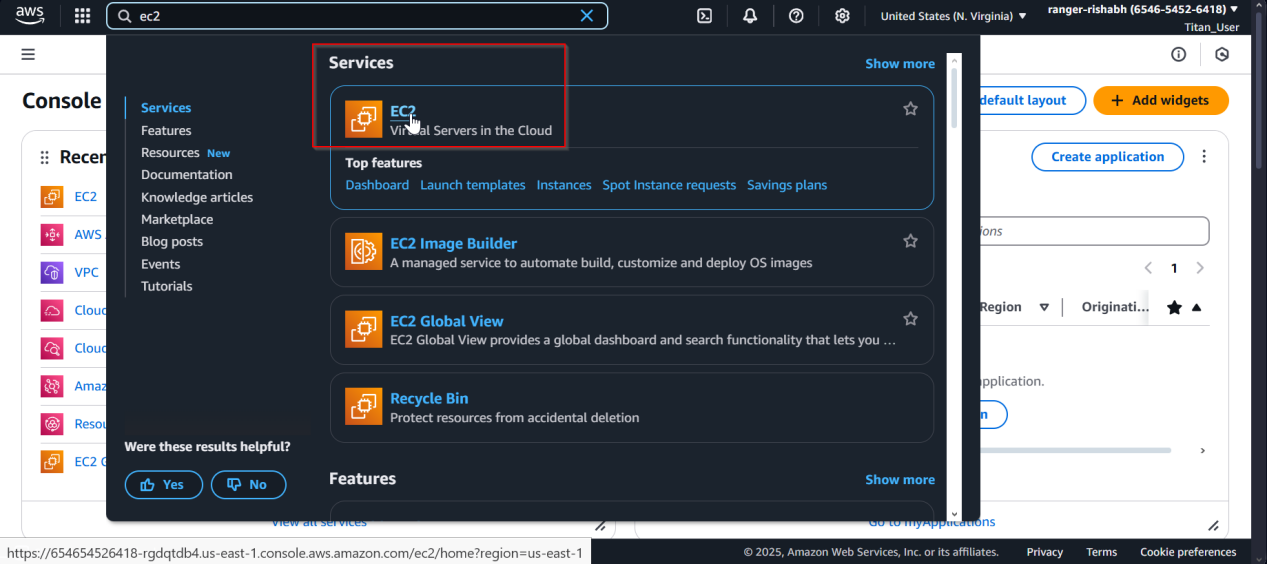
Click on **Launch Instance**.

Monitor the instance status in the **EC2 Dashboard** and wait for it to reach the Running state.









**Subsection: Add Key Pair to Bastion Host for Connecting to ASG Instances**

To securely connect to the Auto Scaling Group (ASG) instances using the Bastion Host, follow these steps to transfer your key pair and establish an SSH connection.

### ****Locate the SSH Key Pair****

Ensure the private key (.pem file) used for creating the ASG instances is available on your local machine.

Note the full path of the **.pem** file for further use.

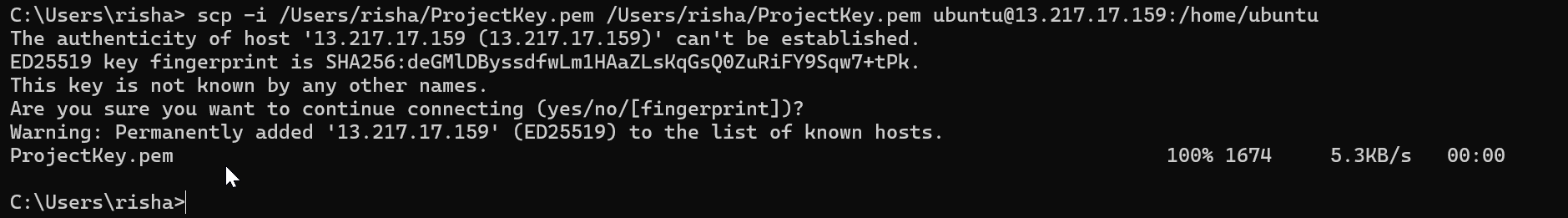
### ****Transfer the Key Pair to the Bastion Host****

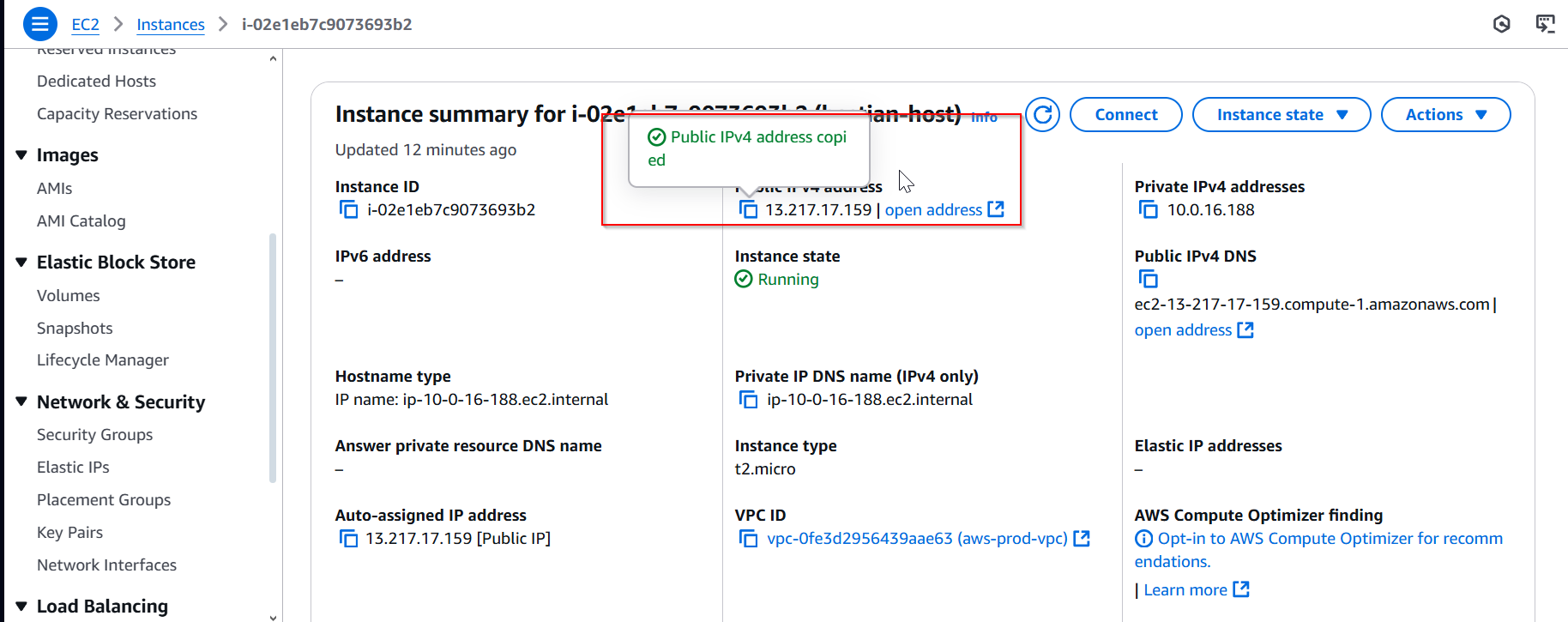
Use the **scp** (Secure Copy Protocol) command to securely transfer the key pair file from your local system to the Bastion Host.

Open your terminal or command prompt and run the following command:scp -i <path\_to\_your\_pem\_file> <path\_to\_pem\_file\_on\_pc> ubuntu@<Bastion\_Host\_Public\_IP>:/home/ubuntu

<path\_to\_your\_pem\_file>: Path to the private key file on your local system.

<Bastion\_Host\_Public\_IP>: The public IP address of the Bastion Host (visible in the AWS EC2 dashboard).





### ****SSH into the Bastion Host****

After transferring the key pair, establish an SSH connection to the Bastion Host using the following command:

ssh -i <path\_to\_your\_pem\_file> ubuntu@<Bastion\_Host\_Public\_IP>

<path\_to\_your\_pem\_file>: Path to your key pair used for the Bastion Host.

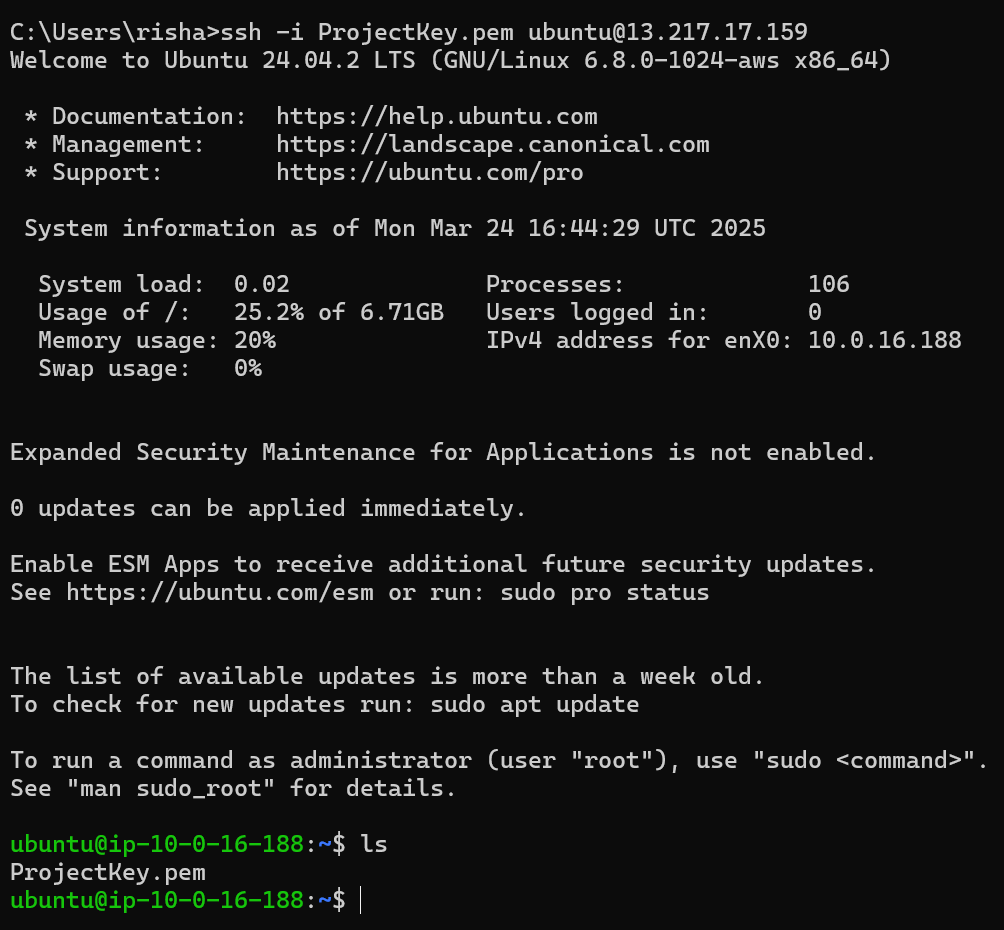
<Bastion\_Host\_Public\_IP>: The public IP address of the Bastion Host.

### ****4. Verify the Key Pair****

Once logged into the Bastion Host, check if the key pair has been successfully transferred:

ls

You should see your .pem file in the /home/ubuntu directory.



### ****Connect to the ASG Instance****

Go to the AWS Management Console → **EC2 Dashboard** → **Instances**.

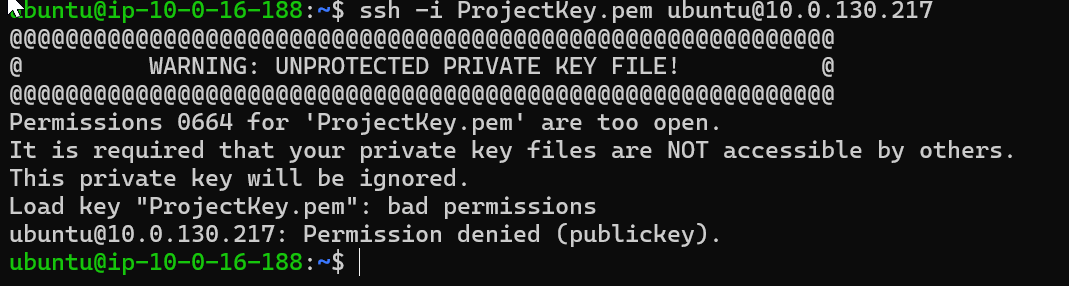
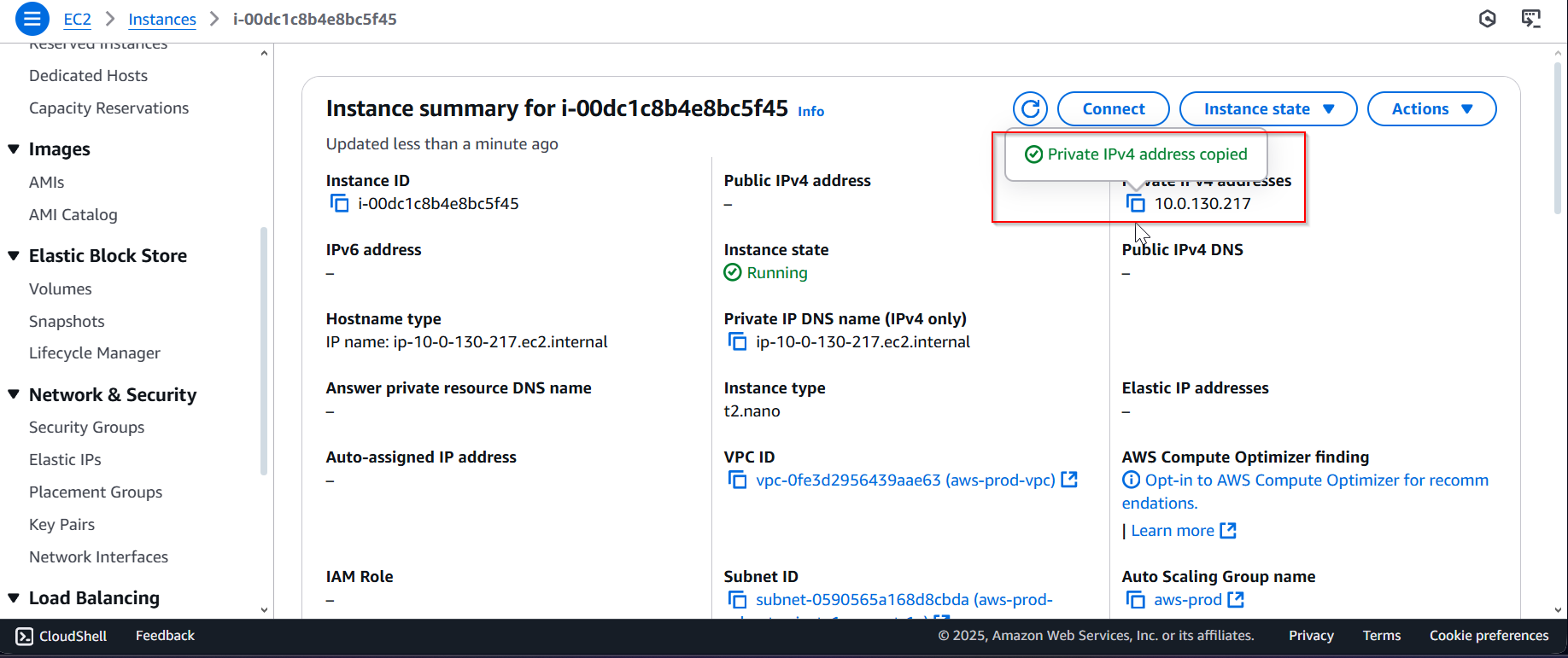
Copy the **Private IP Address** of one of the ASG instances.

Use the following SSH command to connect to the instance using the Bastion Host:

ssh -i <path\_to\_pem\_file\_on\_bastion> ubuntu@<ASG\_Instance\_Private\_IP>

<path\_to\_pem\_file\_on\_bastion>: Path to the key pair on the Bastion Host (e.g., /home/ubuntu/key.pem).

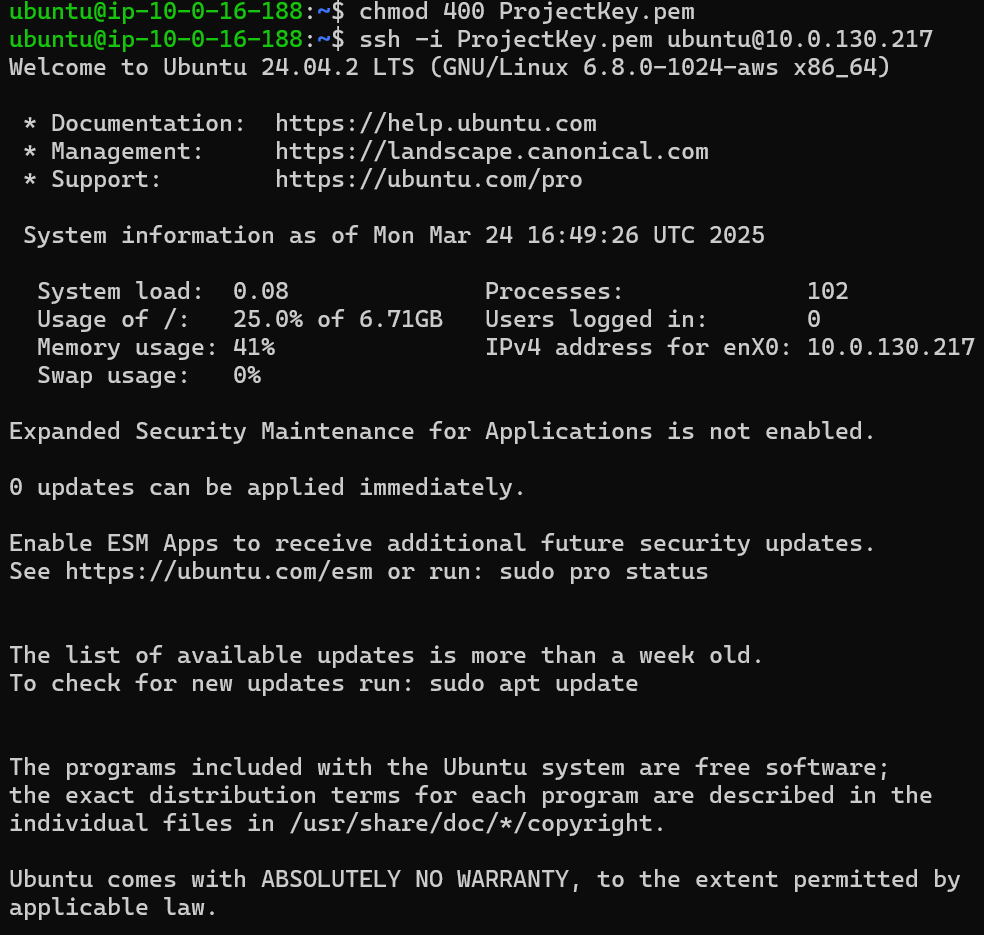
<ASG\_Instance\_Private\_IP>: The private IP address of the target ASG instance.



### ****Login Complete****

Upon successful connection, you should have access to the ASG instance.

You are now logged into the instance using the Bastion Host as a gateway.



In this step, you will create a simple web application using a basic HTML file, host it using Python's built-in HTTP server, and verify its accessibility. Follow these instructions:

### ****Get Sample HTML from W3Schools****

Open your browser and visit [W3Schools](https://www.w3schools.com/" \t "_new) to find a simple HTML example.

Copy the sample HTML code of your choice.

Example:

<!DOCTYPE html>

<html>

<head>

<title>Welcome</title>

</head>

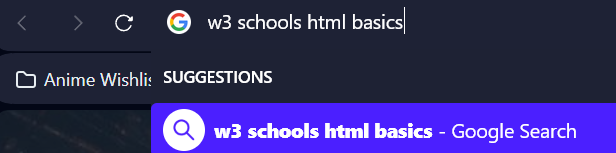
<body>

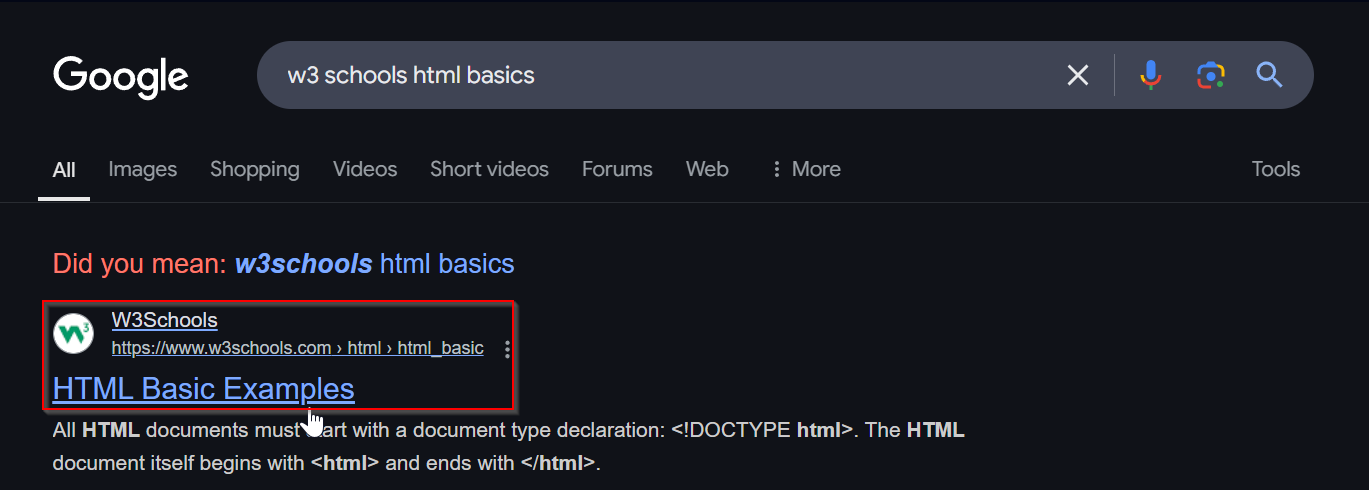
<h1>Welcome to AWS!</h1>

<p>This is a simple HTML page hosted on an AWS EC2 instance.</p>

</body>

</html>





### 

### ****Create an**** index.html ****File****

After logging into the ASG instance, navigate to the home directory using the following command:

cd /home/ubuntu

Create a new HTML file using the **nano** text editor:

nano index.html

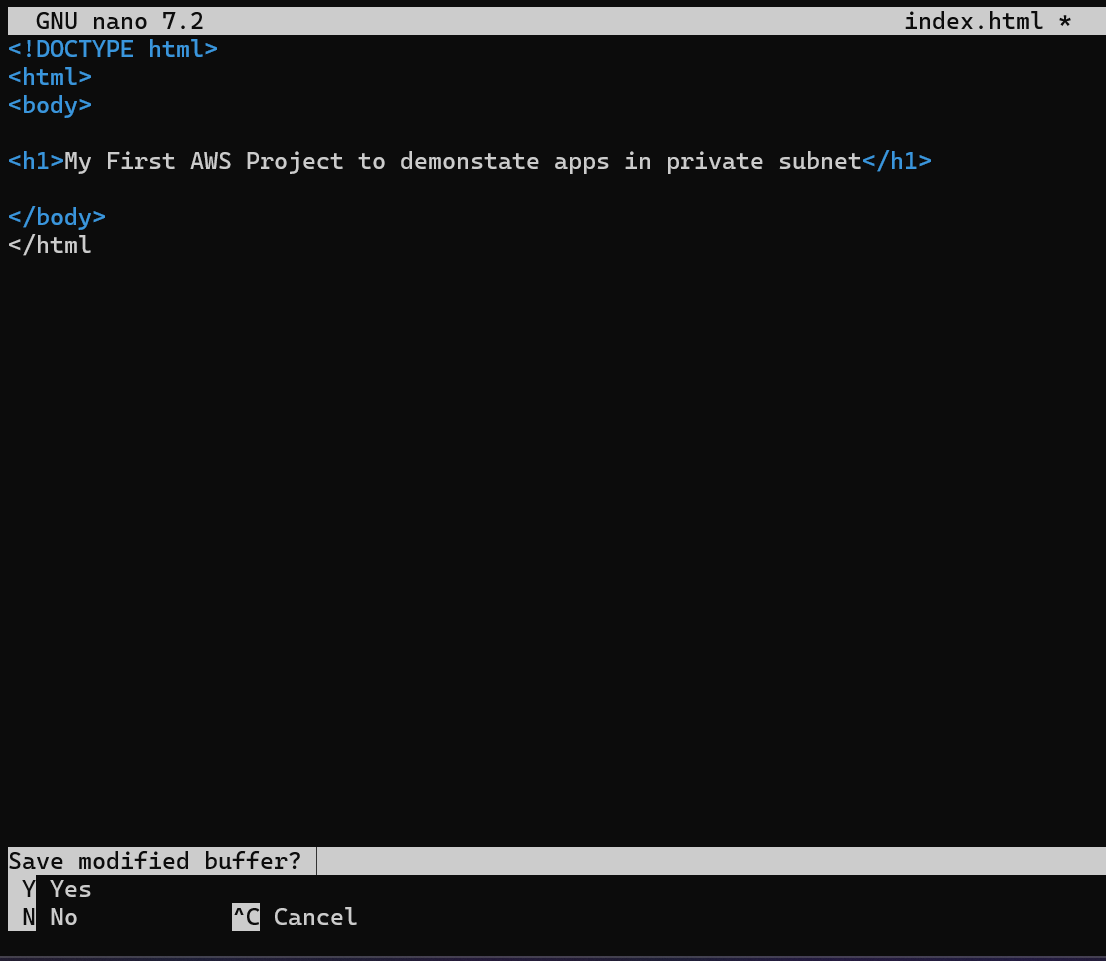
### ****Paste and Save the Code****

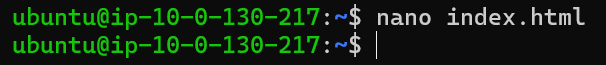
Paste the copied HTML code into the index.html file by right-clicking or using Ctrl + Shift + V in the terminal.

After pasting, press Ctrl + O to write the changes.

Press Enter to confirm.

Exit the editor using Ctrl + X.





### ****Start the Python HTTP Server****

Run the following command to launch a simple HTTP server using Python:

python3 -m http.server 8000

This will start the server on **port 8000** and host the index.html file.

Ensure port **8000** is allowed in your security group for access.



## ****Step 4: Create Load Balancer****

In this step, you will create an **Application Load Balancer (ALB)** to distribute incoming traffic across your EC2 instances in the private subnet using a target group.

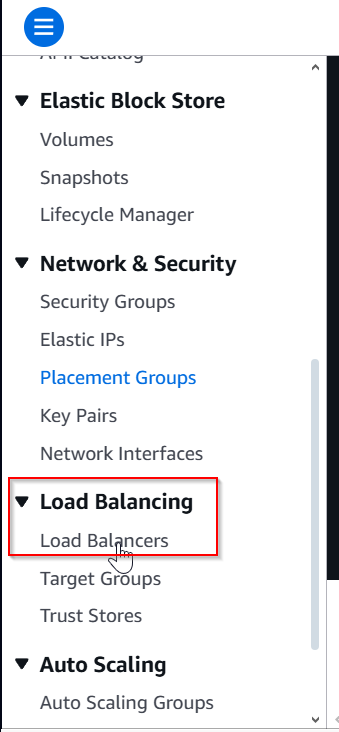
### ****Search for EC2****

Go to the **AWS Management Console**.

Search for **"EC2"** in the search bar and select it.

### ****Navigate to Load Balancers****

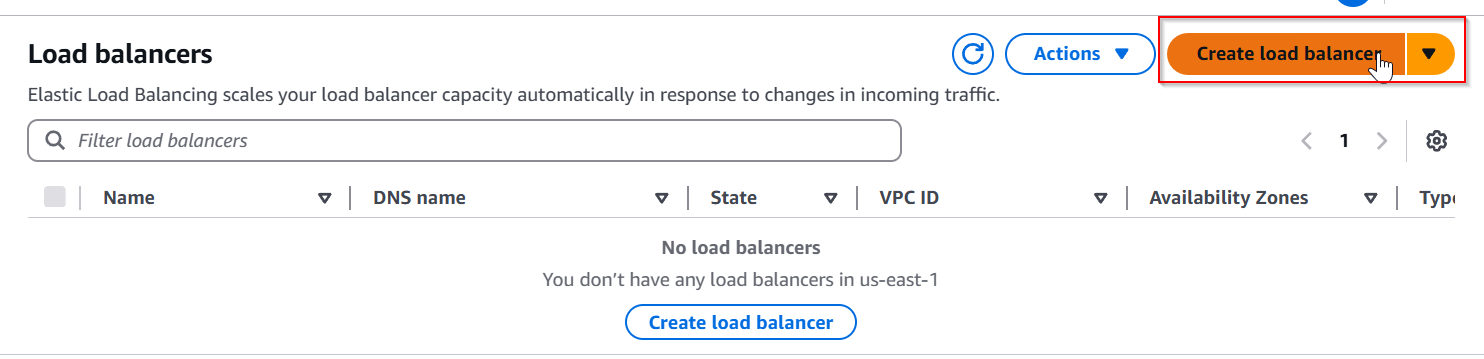
In the left-hand menu, select **Load Balancers** under the **Load Balancing** section.

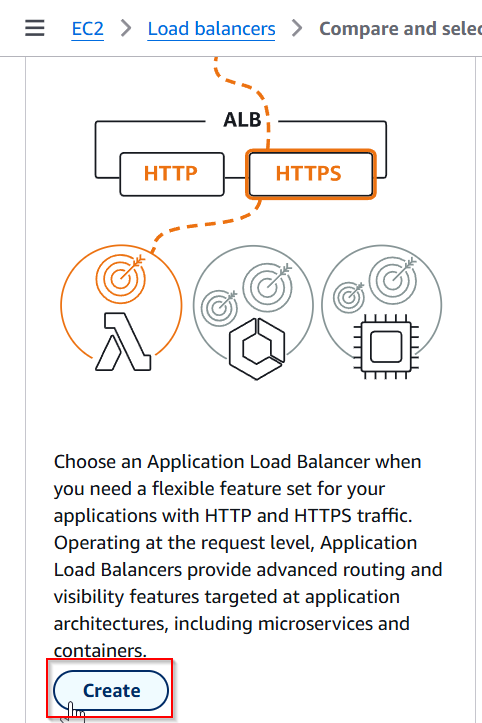


### ****Create a Load Balancer****

Click on **Create Load Balancer**.

Select **Application Load Balancer**.



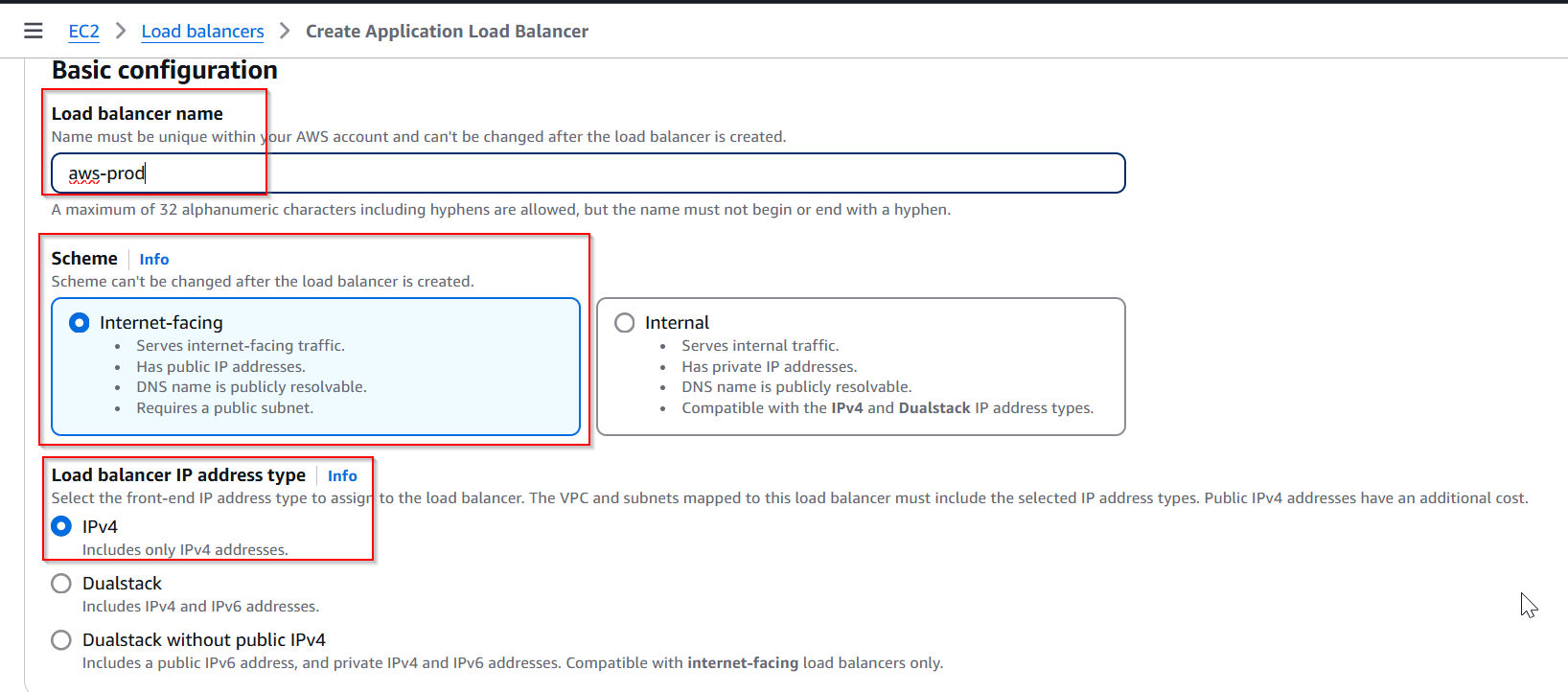


### ****Configure the Load Balancer****

**Name**: Enter aws-prod.

**Scheme**: Choose **Internet-facing**.

**IP address type**: Select **IPv4**.



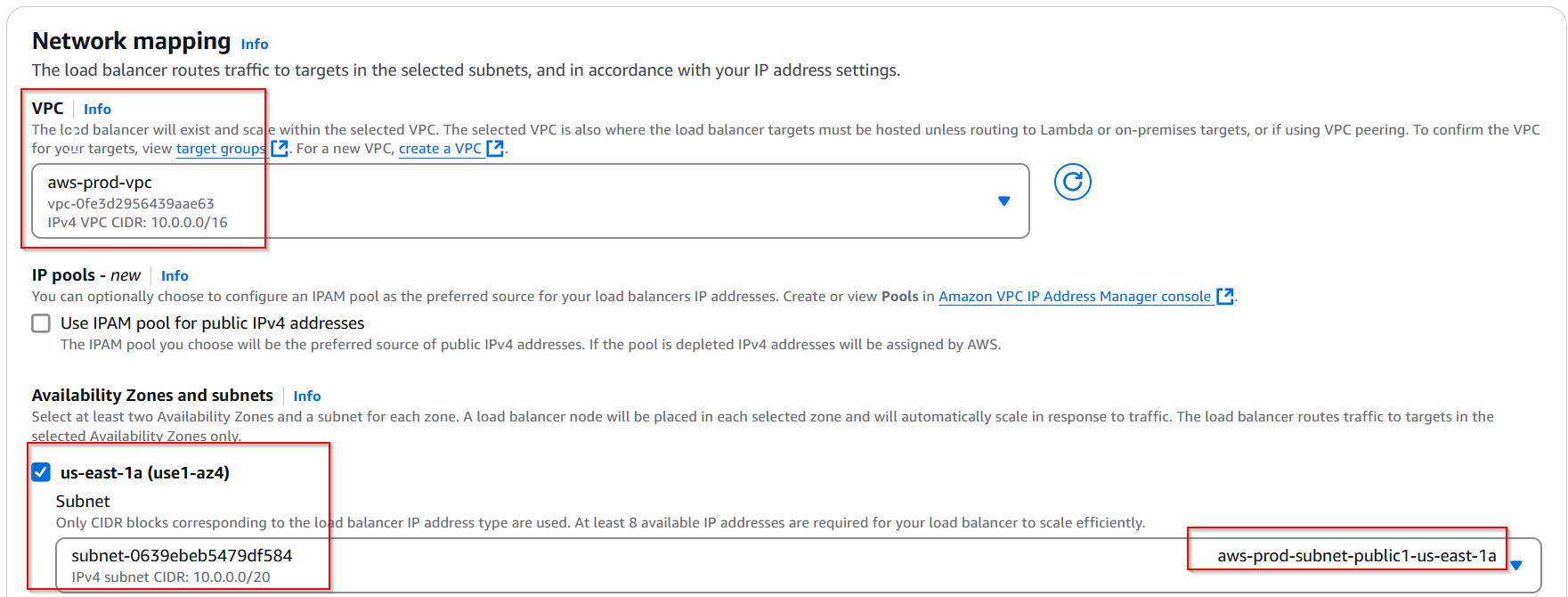
### ****Choose Network and Subnets****

Under **Network mapping**:

Select the **VPC** you created in **Step 1** (e.g., aws-prod).

Choose **Both Availability Zones (AZs)**.

Select the **Public Subnets** (one for each AZ).

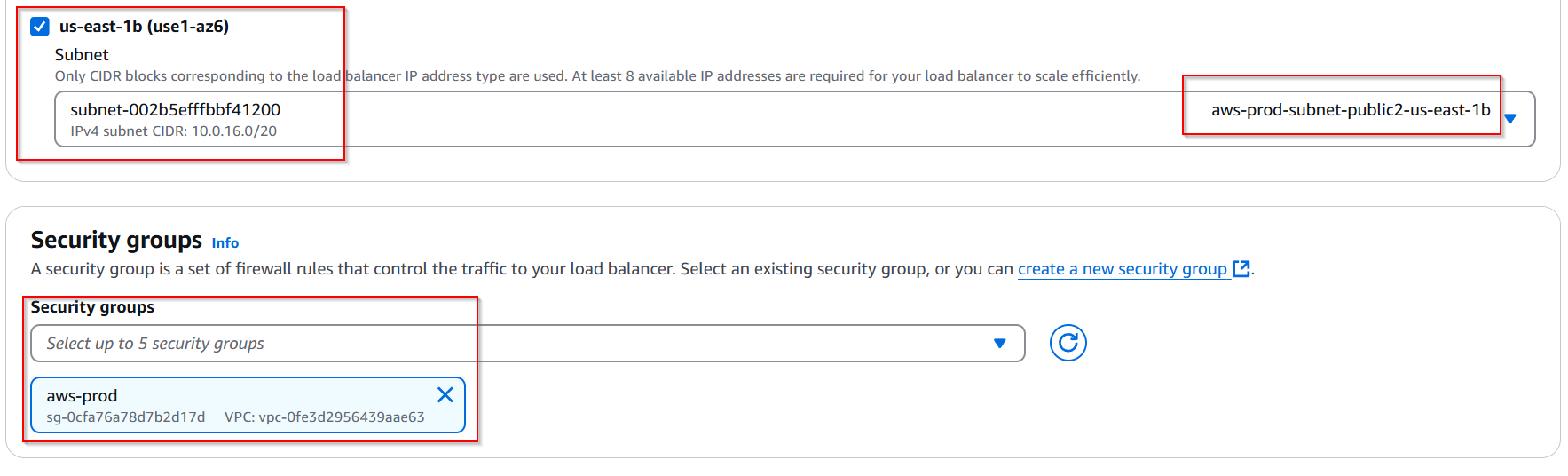


### ****Configure Security Groups****

Choose **Create a new Security Group**.

Name it vpc for aws-prod.

Allow inbound traffic on **HTTP (Port 80)** from anywhere (0.0.0.0/0).



## ****Subsection: Create Target Group****

A **Target Group** routes traffic to one or more EC2 instances. Follow these steps to create a target group:

### ****Navigate to Target Groups****

In the left-hand menu under **Load Balancing**, select **Target Groups**.

Click on **Create Target Group**.

### ****Configure Target Group****

**Target type**: Select **Instances**.

**Name**: Enter aws-prod.

**Protocol**: Choose **HTTP**.

**Port**: Set to **80**.

Select the same VPC created earlier (**aws-prod VPC**).



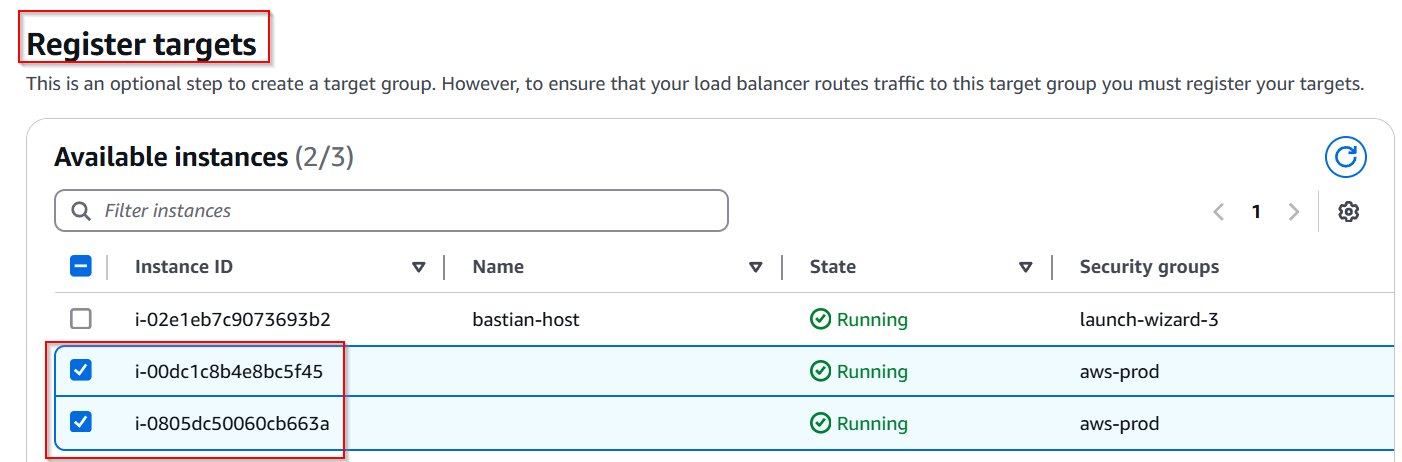
### ****Register Targets****

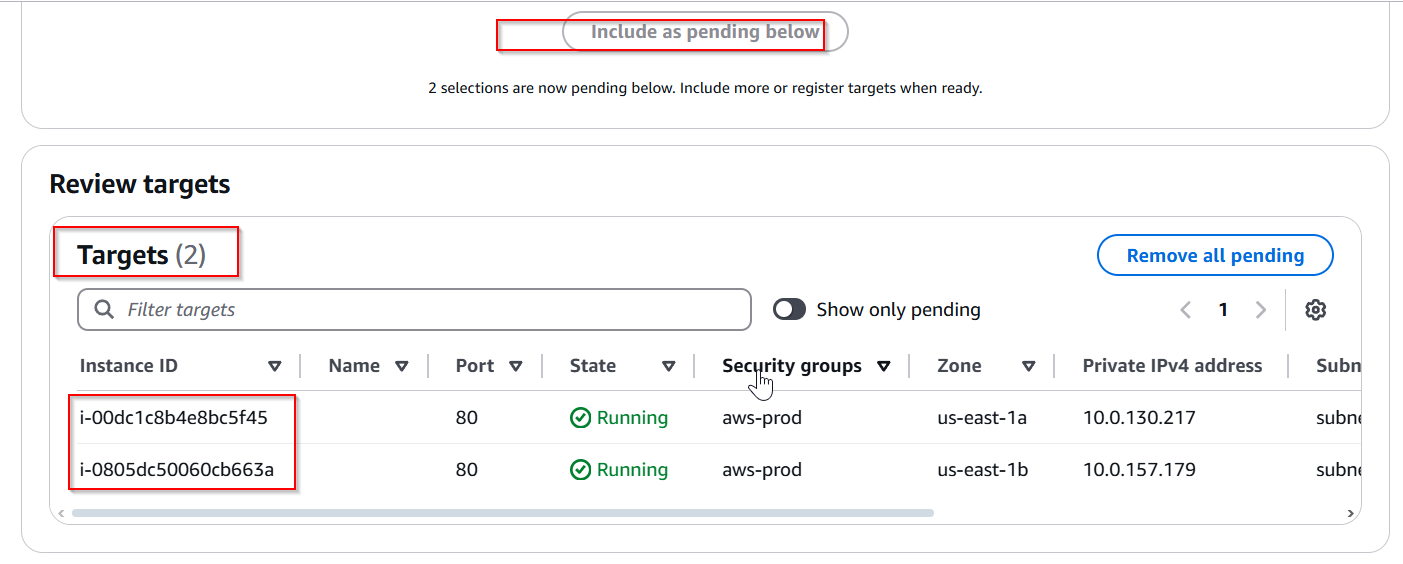
Click **Next**.

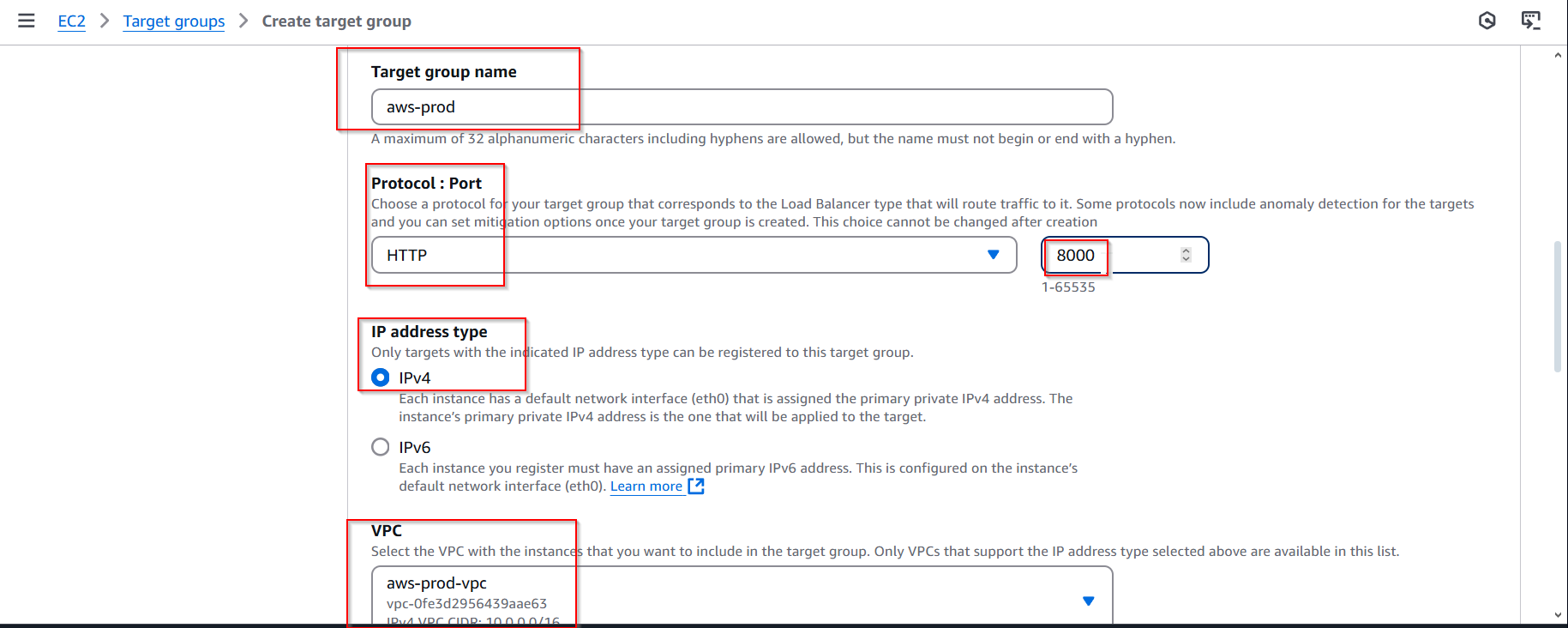
Select the two EC2 instances created by the Auto Scaling Group (ASG).

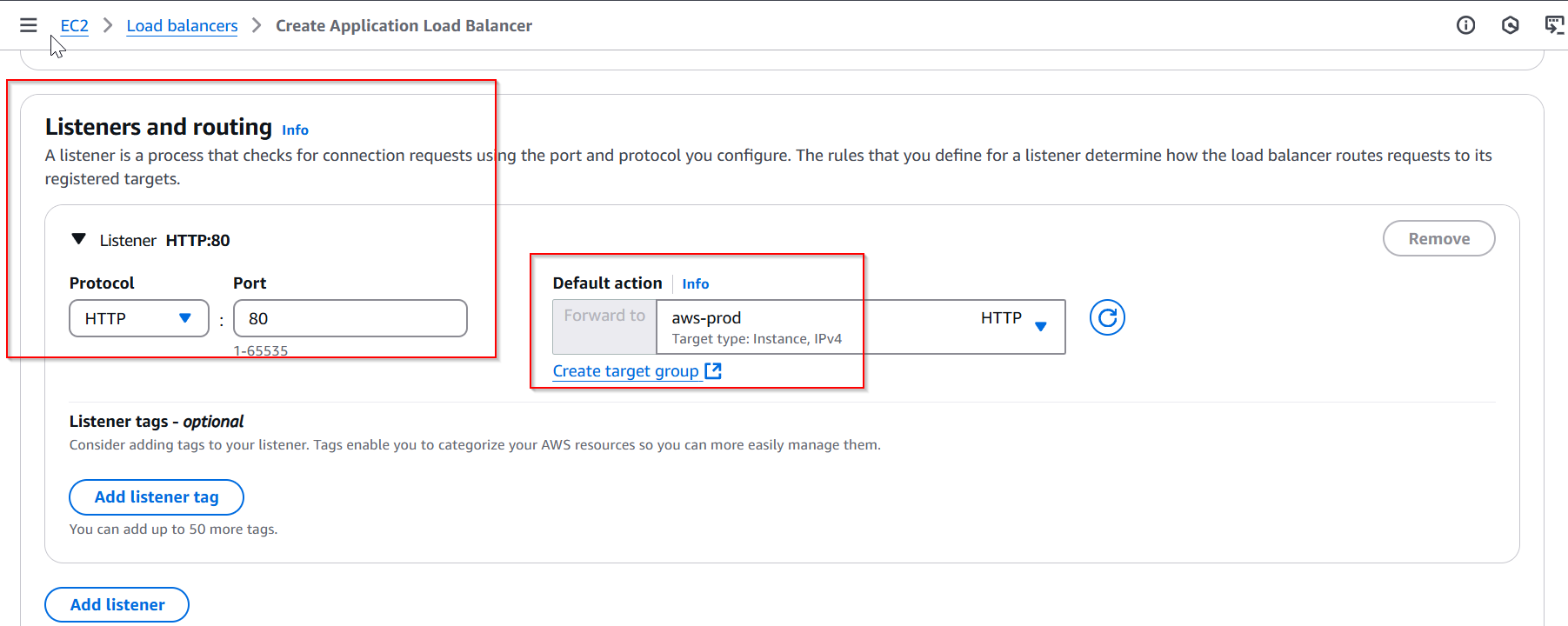
Click on **Include as pending below**.

Confirm and select **Create Target Group**.



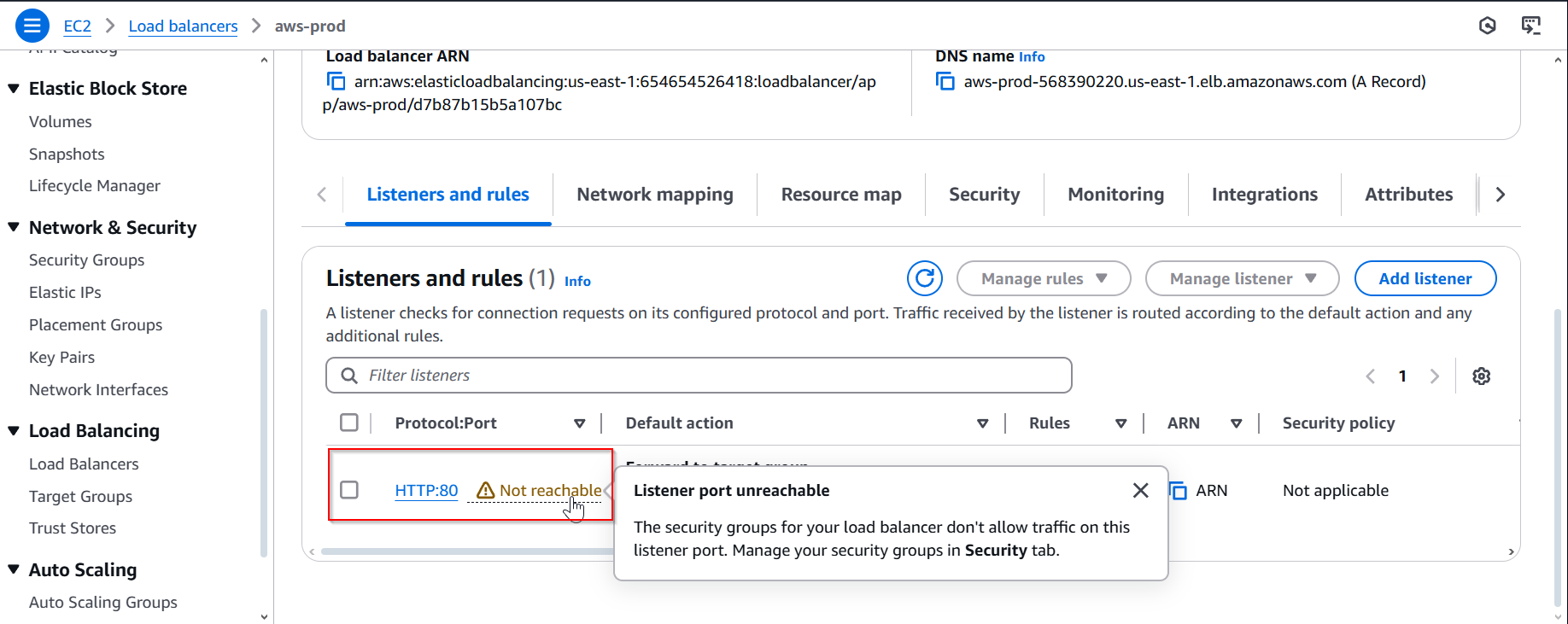






## ****Subsection: Troubleshooting - Error (Port 80 is Not Reachable)****

If you encounter an error indicating that **Port 80 is not reachable**, follow these steps to troubleshoot and resolve the issue:



### ****Access Security Group Settings****

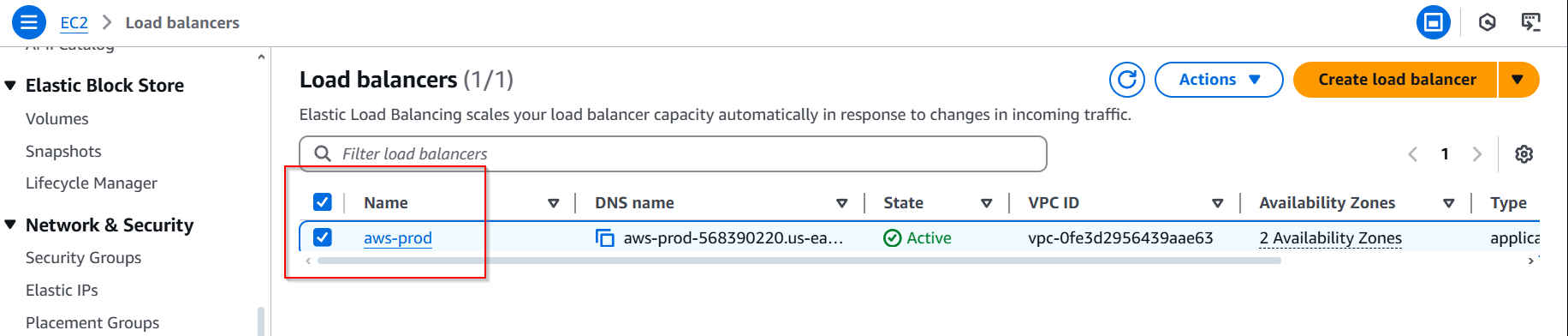
Go to the **AWS Management Console**.

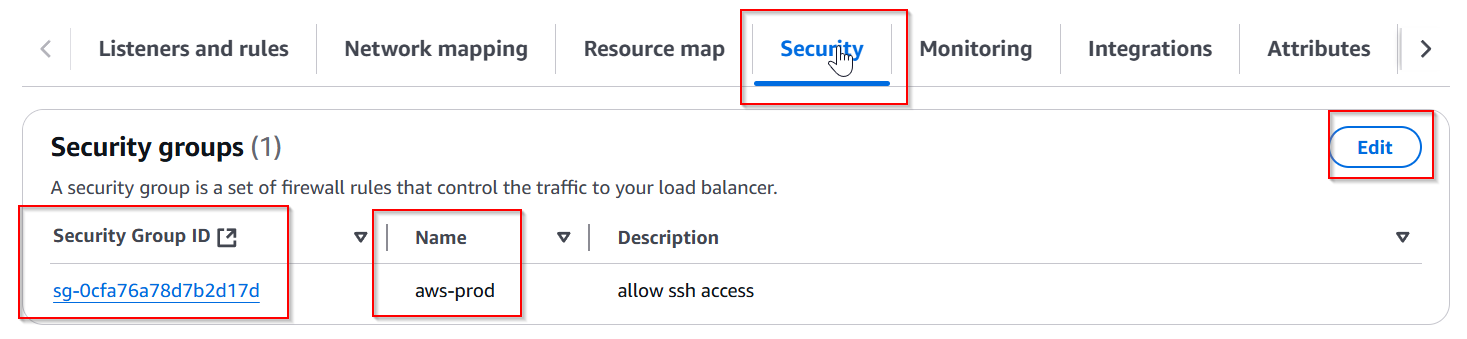
Search for **EC2** and select it.

In the left-hand menu, click on **Security Groups** under the **Network & Security** section.

Locate the security group associated with the load balancer (aws-prod).

Click on the security group name to view its details.

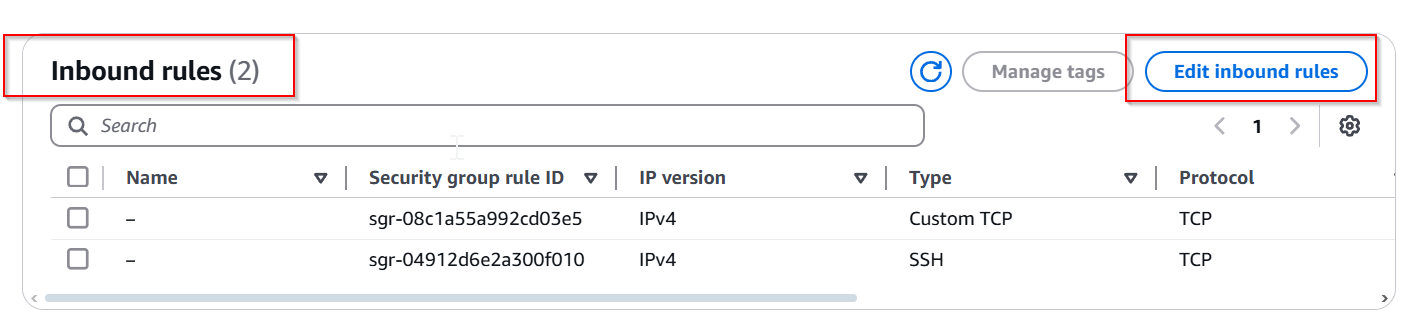




### ****Edit Inbound Traffic Rules****

In the **Inbound rules** tab, click on **Edit inbound rules**.

Click on **Add Rule**.



#### ****Add the Following Rule:****

**Type**: HTTP

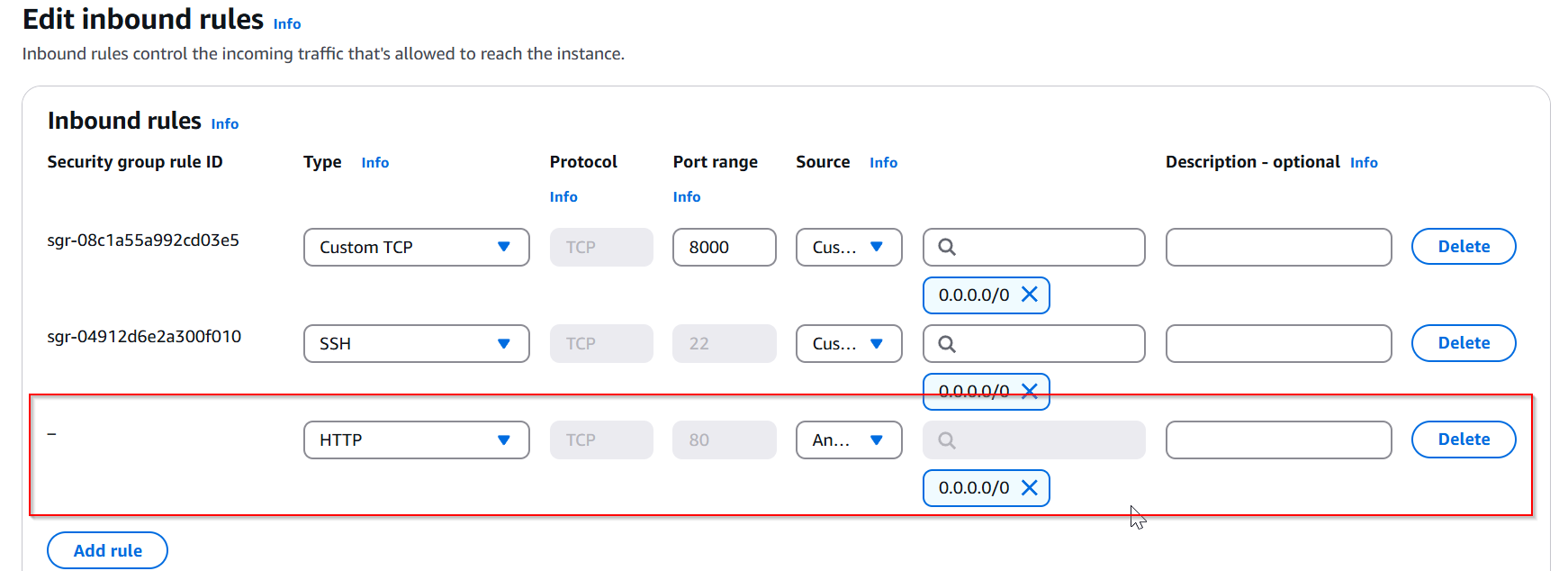
**Protocol**: TCP

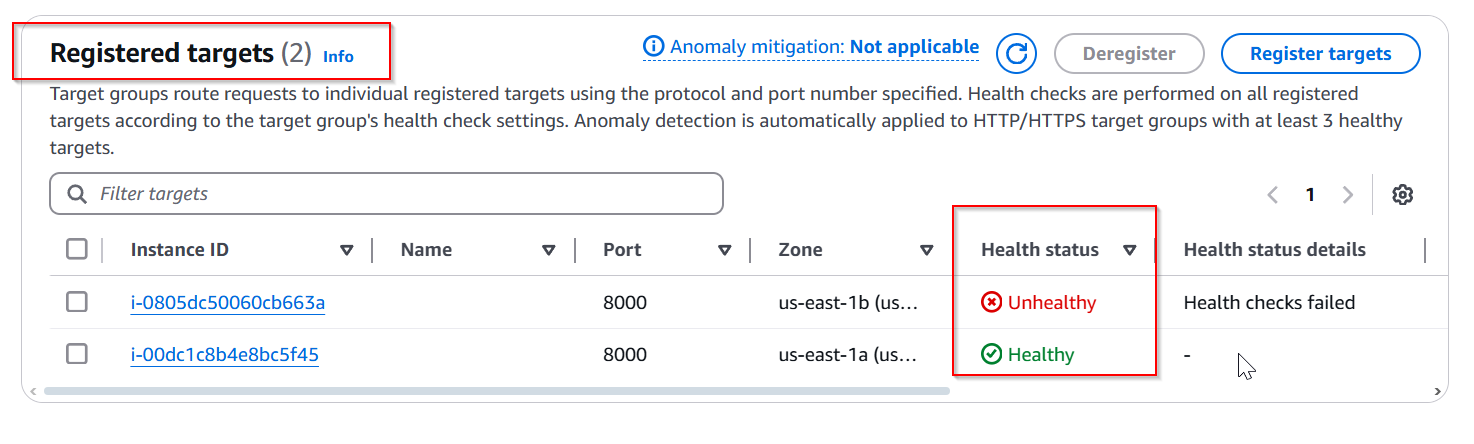
**Port Range**: 80

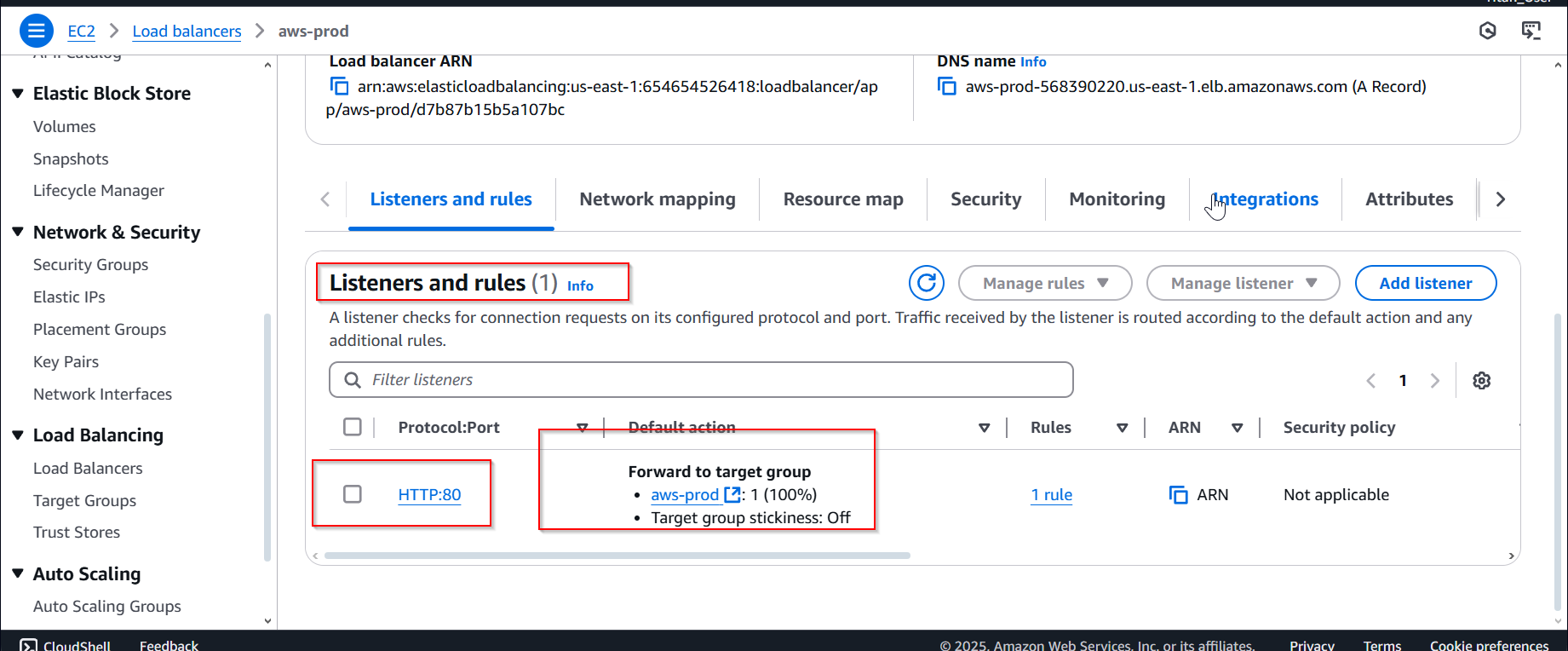
**Source**: Select **Anywhere (0.0.0.0/0)**

(If IPv6 access is required, add **::/0** as well.)

Click on **Save Rules**.







### ****Verify Connectivity****

Copy the **DNS Name** of the load balancer from the **Load Balancers** tab.

Open a browser and navigate to http://<load-balancer-dns>.

You should see the HTML page created earlier.

Conclusion

This project successfully demonstrates the deployment of a web application within a secure and scalable AWS environment using VPC, EC2, Auto Scaling Groups, and an Application Load Balancer.

Through the structured creation of a custom VPC with both public and private subnets, we established a secure network infrastructure. The application was deployed on EC2 instances within private subnets, ensuring greater security. Access to these instances was managed through a bastion host, enabling secure SSH access.

An Auto Scaling Group (ASG) was implemented to ensure high availability and scalability of the application. The ASG automatically adjusted the number of instances based on demand, ensuring optimal performance. Furthermore, an Application Load Balancer was configured to distribute incoming traffic across multiple instances, providing fault tolerance and load management.

The troubleshooting steps performed during the setup, including resolving connectivity issues and adjusting security group rules, highlighted the importance of effective monitoring and management within a cloud environment.

In conclusion, this project serves as a comprehensive proof of concept for deploying a resilient web application on AWS. It offers practical insights into cloud networking, security management, and application scalability — key components for building production-grade cloud solutions. This foundational knowledge can be further expanded to deploy more complex applications with additional AWS services in future projects.