

-:AWS PROJECT:-

Name - Binku Kumar Sinha

Institute's Name - ISOEH SILIGURI

Batch Name - AWS-17MAY-7JUNE-3pm-5pm

Project Details:

Objective: Create a web application and deploy it on an EC2 instance using different Linux distributions.

The web application must include at least 3 HTML pages.

Load Balancing must be implemented for the deployed application.

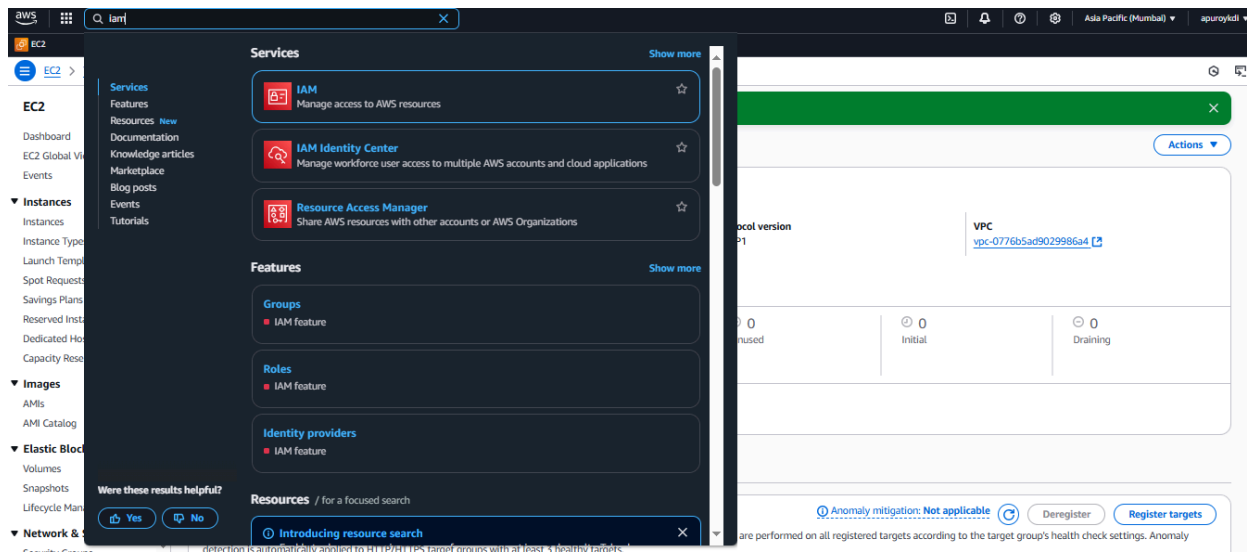
The deployment process should involve different AWS IAM users—each user should deploy a separate instance of the web application.

The website should be set up and managed by these different IAM users accordingly.

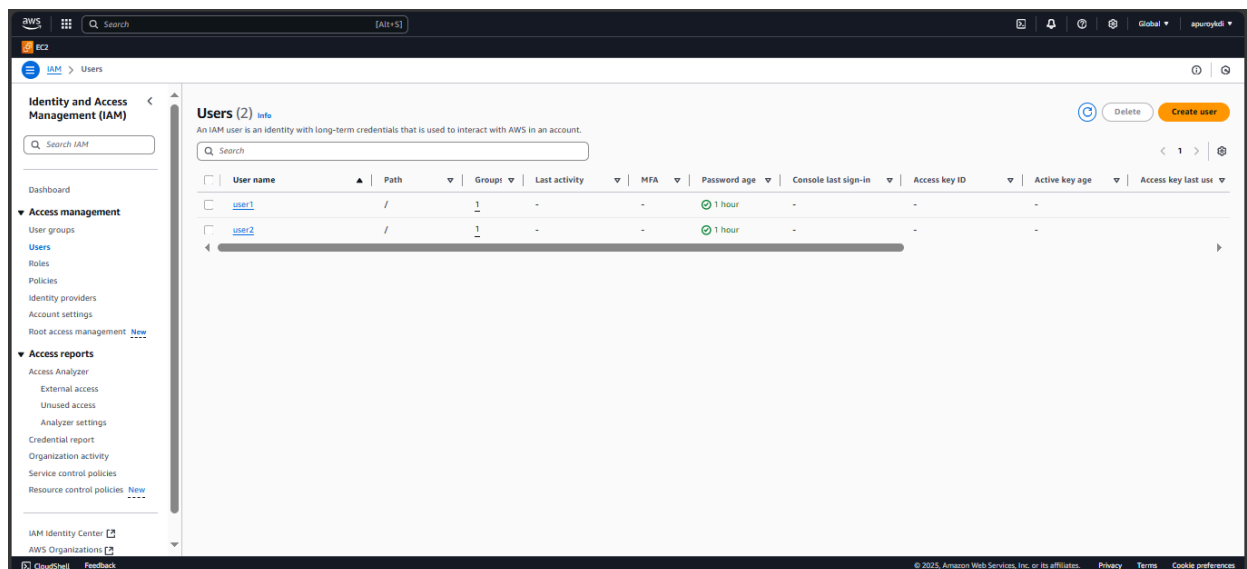
Deployment:

To do this project we need to login into **AWS Console** with the **Root Id**. Then need to go to The **IAM Dashboard** to create the Users and the groups. As bellow:

IAM Dashboard:



Users list:



We created two users mentioned above. And these users are added with groups for their specified permissions..

Group Name and Permissions With user name:

AdminG group:

The screenshot displays the AWS IAM console interface for the 'AdminG' user group. The left sidebar shows the navigation menu with 'Access management' and 'Access reports' sections. The main content area is titled 'AdminG' and includes a 'Summary' section with the following details:

- User group name:** AdminG
- Creation time:** June 12, 2025, 17:30 (UTC+05:30)
- ARN:** arn:aws:iam::157476309840:group/AdminG

Below the summary, the 'Permissions' tab is active, showing a list of 'Permissions policies (9)'. The table lists the following policies:

Policy name	Type	Attached entities
AmazonAppFlowFullAccess	AWS managed	1
AmazonAppStreamFullAccess	AWS managed	1
AmazonConnect_FullAccess	AWS managed	1
EC2InstLaunchFullAccess	AWS managed	1
ElasticLoadBalancingFullAccess	AWS managed	1
ElasticLoadBalancingFullAccess	AWS managed	1
ElasticLoadBalancingFullAccess	AWS managed	1
GlobalAcceleratorFullAccess	AWS managed	1
IAMFullAccess	AWS managed	1
IAMFullAccess	AWS managed	1

This is the **Admin group** named **AdminG** and **user1** is added to this group who has the full permission to maintain all the services as EC2 Instance, IAM

DevOps group:

The screenshot displays the AWS IAM console interface for the 'DevOps' user group. The left sidebar shows the navigation menu with 'Access management' and 'Access reports' sections. The main content area is titled 'DevOps' and includes a 'Summary' section with the following details:

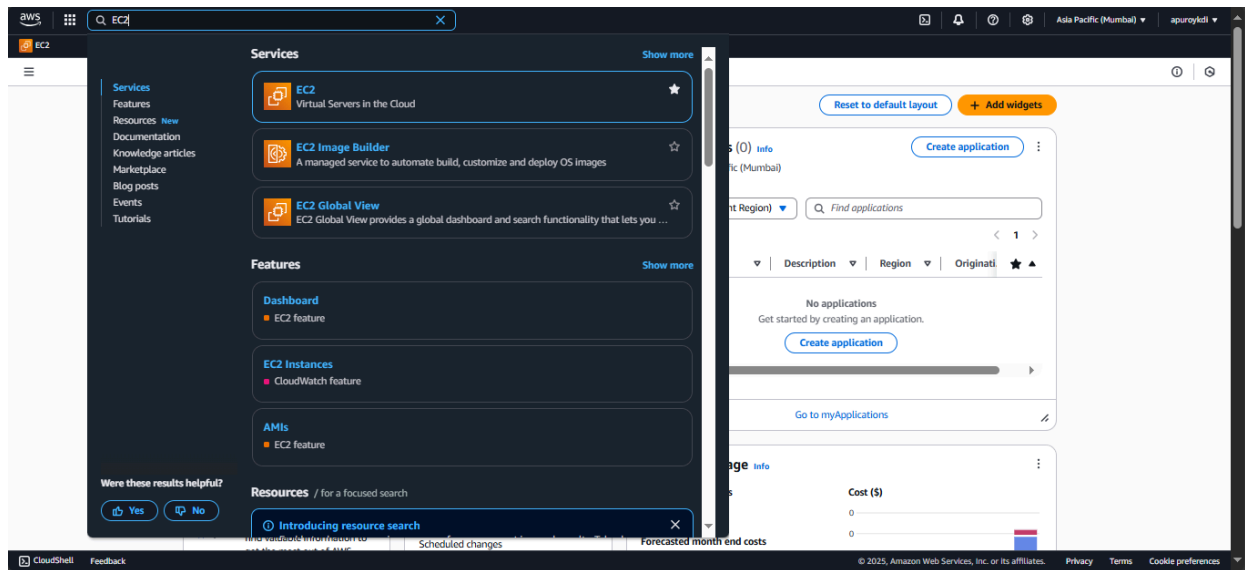
- User group name:** DevOps
- Creation time:** June 12, 2025, 17:31 (UTC+05:30)
- ARN:** arn:aws:iam::157476309840:group/DevOps

Below the summary, the 'Permissions' tab is active, showing a list of 'Permissions policies (1)'. The table lists the following policy:

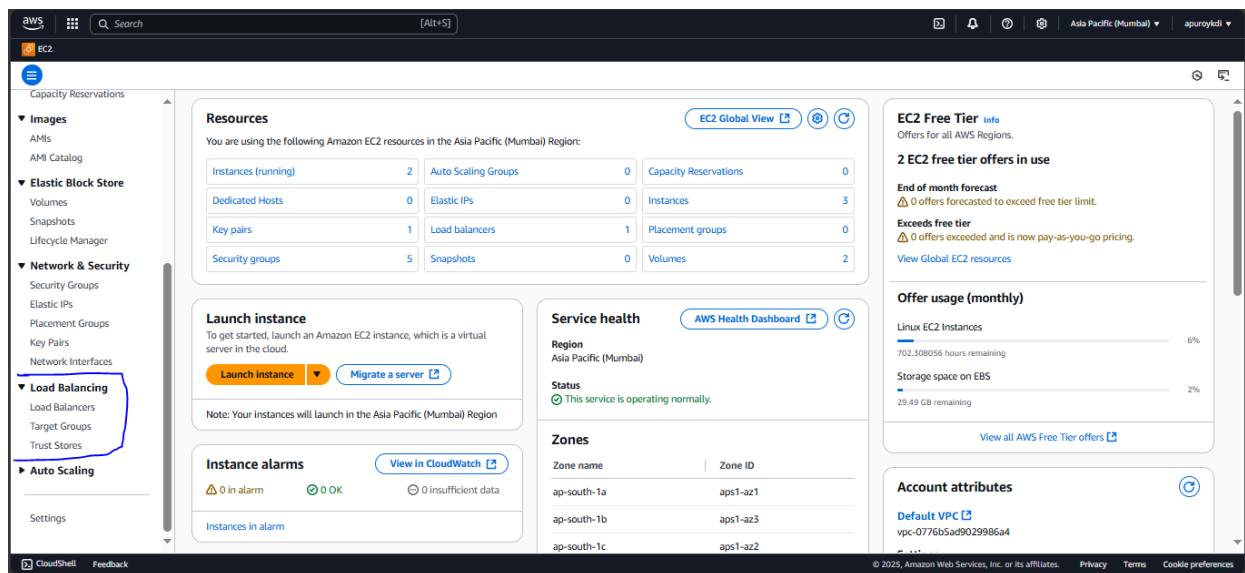
Policy name	Type	Attached entities
ReadOnlyAccess	AWS managed - job function	1

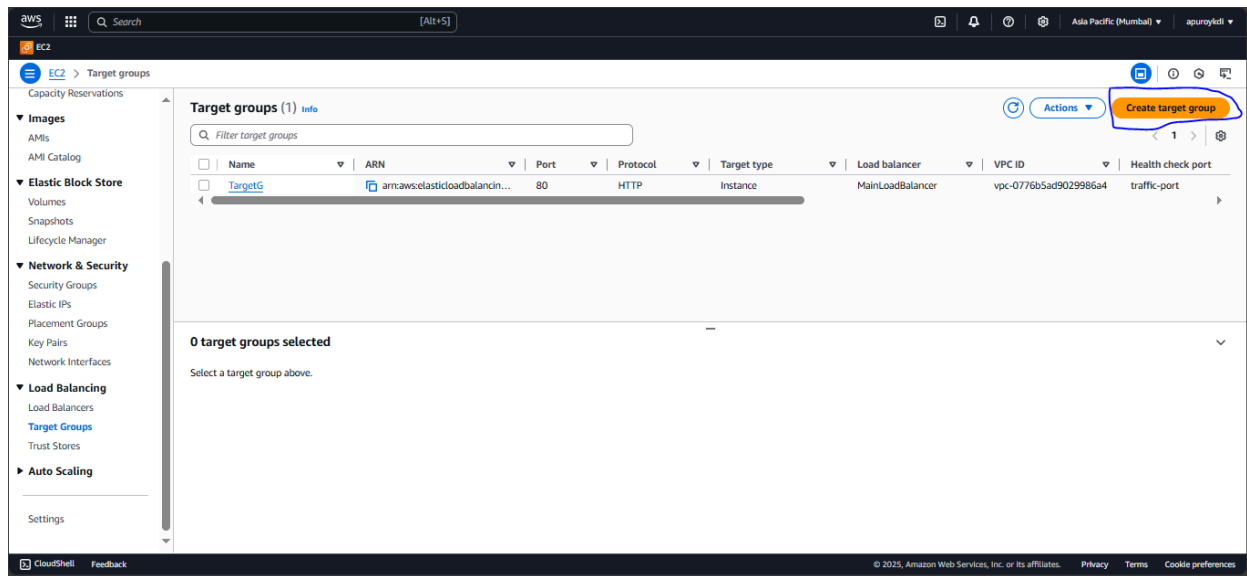
This is the **Normal group** named **DevOps** and **user2** is added to this group who has only the **read-only** permission.

For the rest of the process we need to go into **EC2 Service** by searching in the search bar. As bellow:



Now at first we have to configure the **Load Balancers** and the **Target Groups** from the **Load Balancing** option which is in the left side and in the third position from bottom of EC2 Options menu. As bellow:



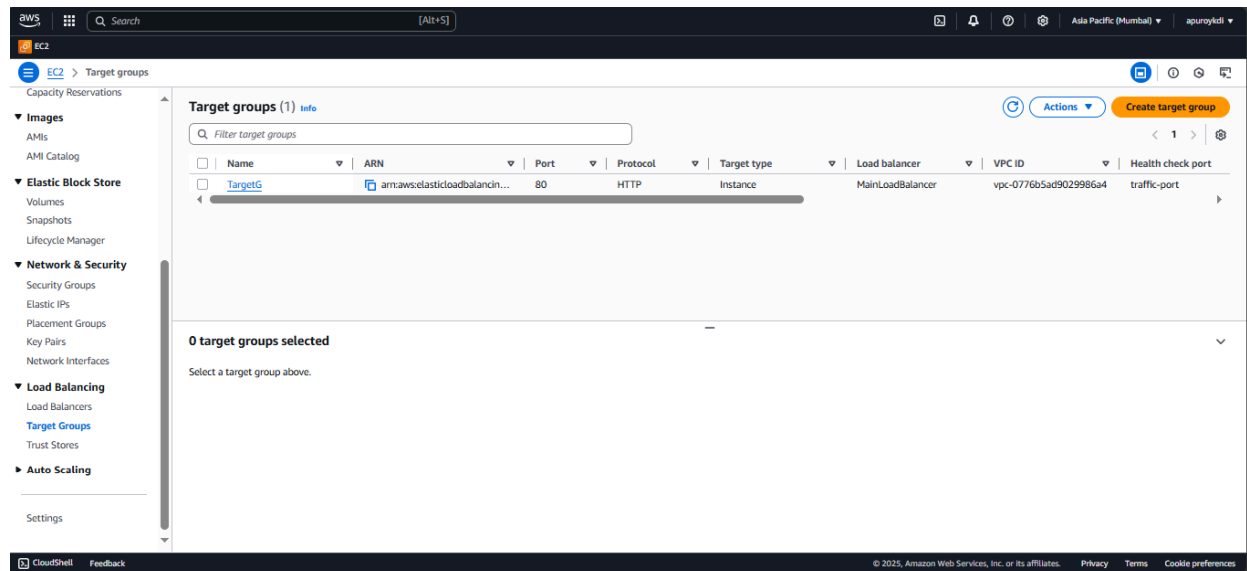


Here we can see the three options as listed on the above picture:

1. **Load Balancers**
2. **Target Groups**
3. **Trust Stores**

Before configuring the **Load Balancers** , we need to configure the **Target Groups** first. So we click the 2nd option for **Target Groups** .

Target Group Dashboard:



If we don't have any created group, then we need to create groups by clicking the **Create target group** (as shown in the above picture) . But we have created a group named **TargetG**.

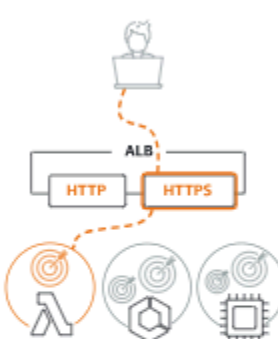
As we have created the **Target Group**. Now I need to configure the **Load Balancers**. After clicking the Load Balancers we would see the menu as **Create load balancer** . By clicking that menu there will be displayed three new menus to create the load balancer as **1.HTTP, HTTPS, 2.TCP, UDP, TLS, 3. GWLB** respectively . But we need the 1st one for this project .

Compare and select load balancer type

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

Load balancer types

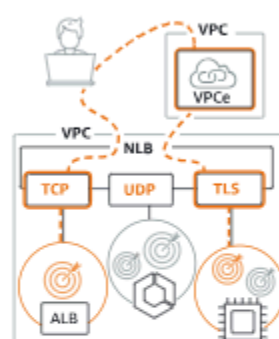
Application Load Balancer Info



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.

Create


Network Load Balancer Info



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.

Create

Gateway Load Balancer Info



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.

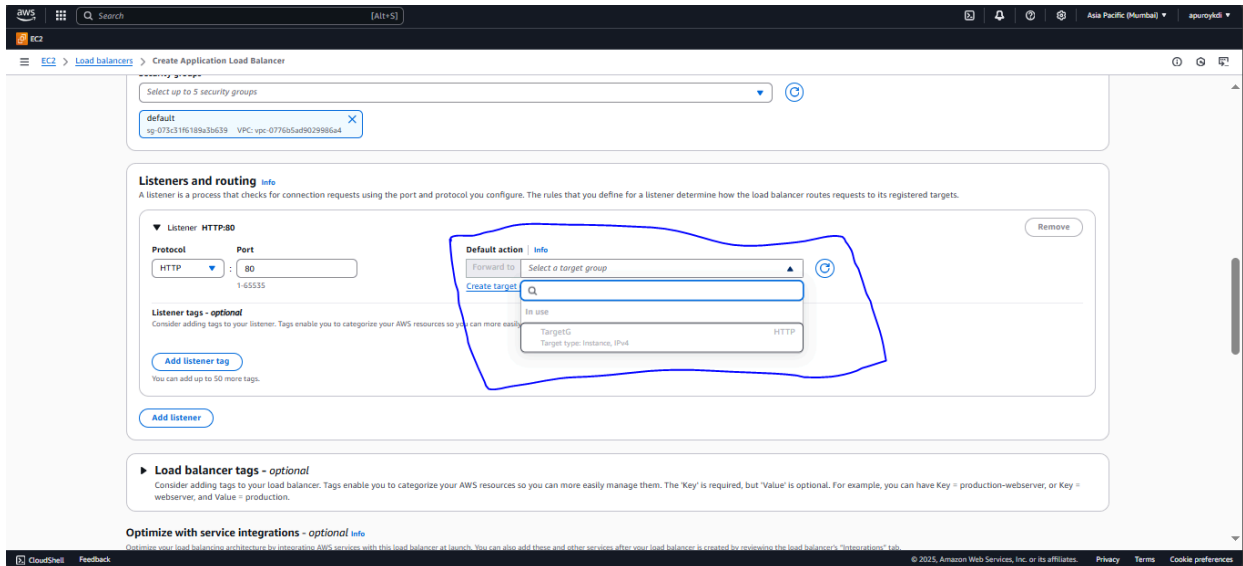
Create

► Classic Load Balancer - previous generation

Close

After this, there will be some queries that we need to fill properly to create a loader.

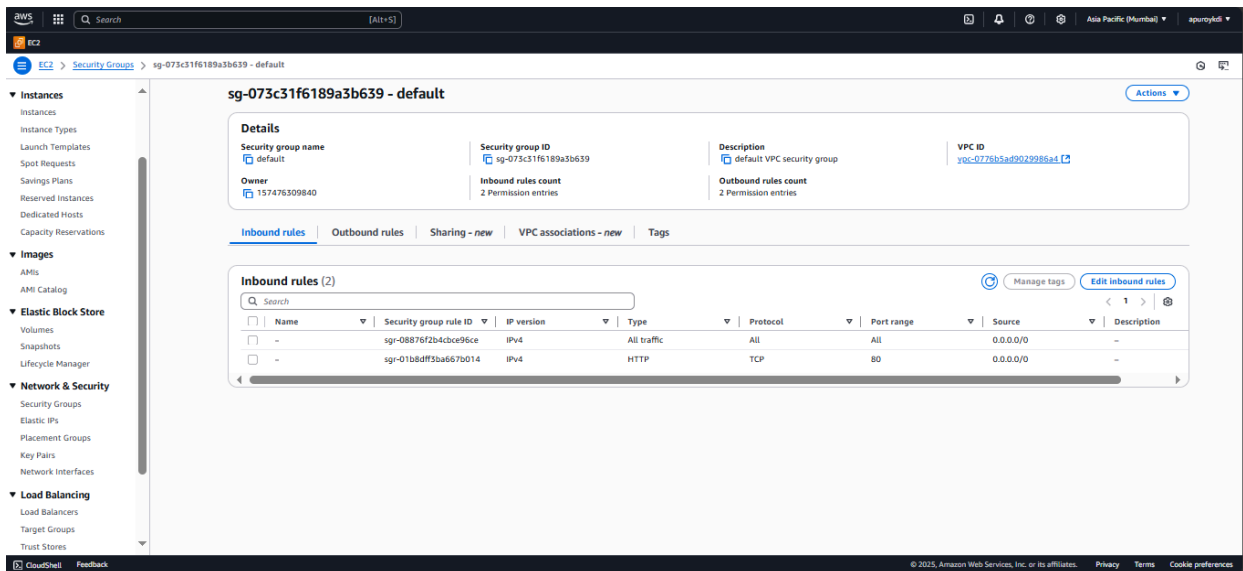
Select the Target Group:



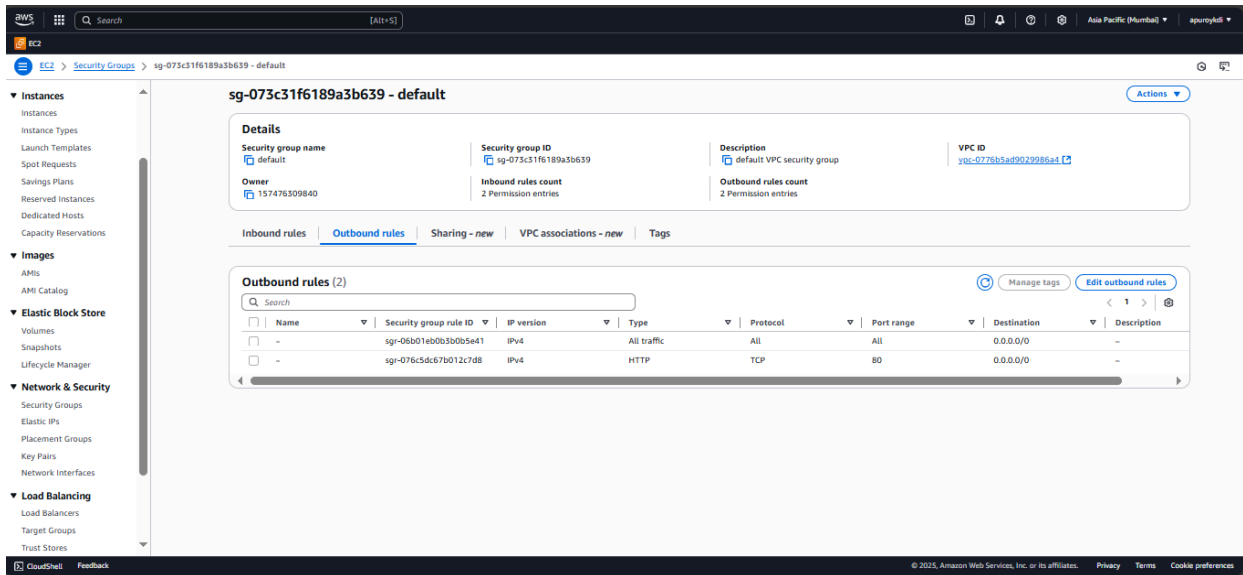
This is why we need to create the **Target Group** first. Here we have to select or create the **Target Group** to fulfill the criteria to create the **Load Balancer** .

Now we need to configure the **Security Groups for Inbound rules Outbound rules**. Only allow the **HTTP** connections for the Port 80. As below:

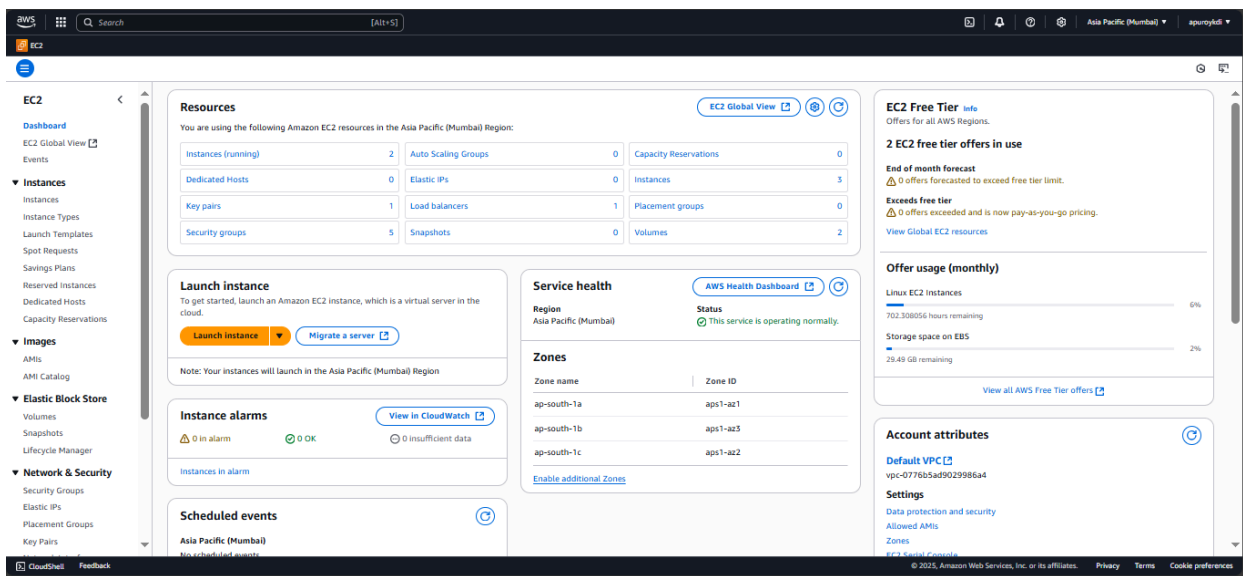
Inbound Rules:



Outbound Rules:



Now we will go for creating the EC2 Instances and click the **Launch instance** as showing below:



Then we will type the instance or the server name as our choice . And also select the **Amazon Machine Image (AMI)** according to our needs or requirements . Need to remember that the **Firewall (Security Groups)** always be the same as shown below in the picture for accessing multiple **Apache Servers**. Need to check all the boxes as it may be needed in the future (both HTTP and HTTPS) to secure our server or website by applying the **SSL Certificate**.

Network | Info
vpc-0776b5ad9029986a4

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-5' with the following rules:

- ☒ Allow SSH traffic from
Helps you connect to your instance Anywhere
0.0.0.0/0
- ☒ Allow HTTPS traffic from the Internet
To set up an endpoint, for example when creating a web server
- ☒ Allow HTTP traffic from the Internet
To set up an endpoint, for example when creating a web server

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

▼ Configure storage Info Advanced
1x GiB Root volume, 3000 IOPS, Not encrypted

🔗 Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage ✕

After creating the EC2 Instance need to wait a few seconds or minutes as it will take time to initialize the instance to use . Then login into the server by connecting.

Instances (2) Info
Last updated less than a minute ago

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	ELB
<input type="checkbox"/>	server-1	i-03f55c831088e269f	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-15-207-221-15.ap-...	15.207.221.15	-
<input type="checkbox"/>	server-2	i-082574e8d1fbc56d4	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-13-200-222-252.ap...	13.200.222.252	-

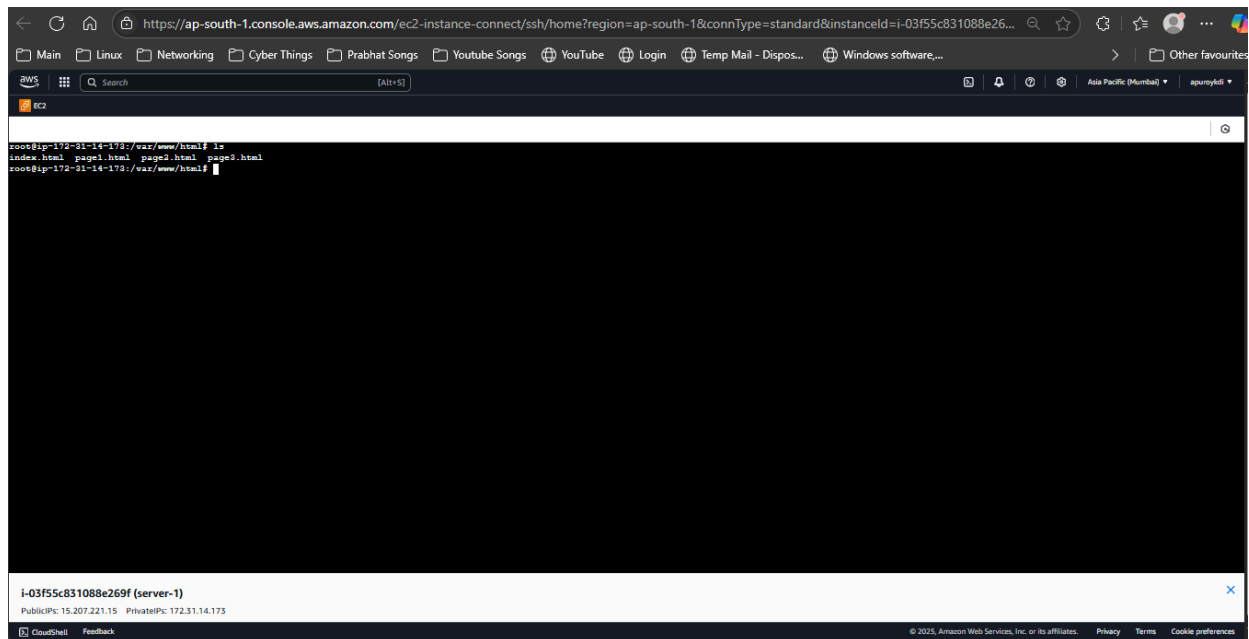
Select an instance

After successfully login into the server, need to configure the apache2 service by some commands . The commands given below:

Redhat based:

```
$sudo su ----- (For root access)
#yum update -y ---- (To update the system . )
#yum install httpd -y ----- (To install apache2 server into the system)
#systemctl status httpd ----(To check the status the service is running or not)
#systemctl start httpd ---- (To start the service if not started)
#cd /var/www/html ----- (The default location of apache server where we need to add the web
pages )
#echo "<head>This is the page 1 from the server 1</head>" > page1.html ----- (First page with
the extension of .html)
#echo "<head>This is the page 2 from the server 1</head>" > page2.html ----- (Second page
with the extension of .html)
#echo "<head>This is the page 3 from the server 1</head>" > page3.html ----- (Third page with
the extension of .html)
#systemctl restart apache2 ---- (To restart the service to the updation and execution of the
pages into the web server)
#
```

1st Server is Amazon Linux:



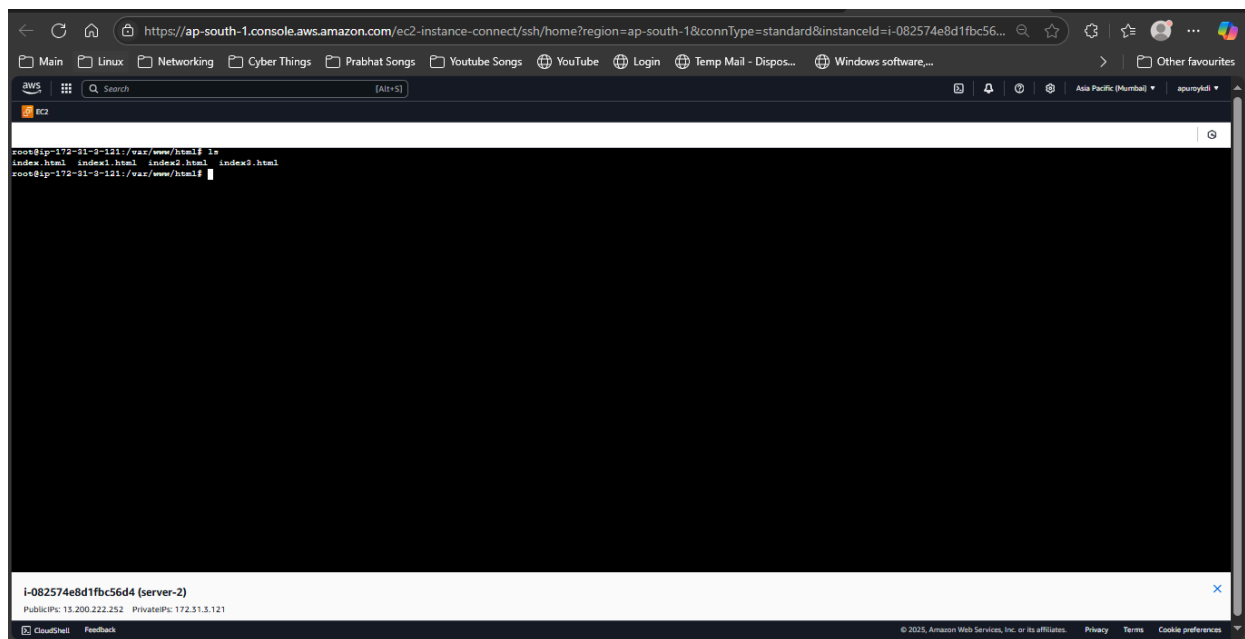
15.207.221.15 This is the Public address of server 1

Debian or Ubuntu based:

```
$sudo su ----- (For root access)
```

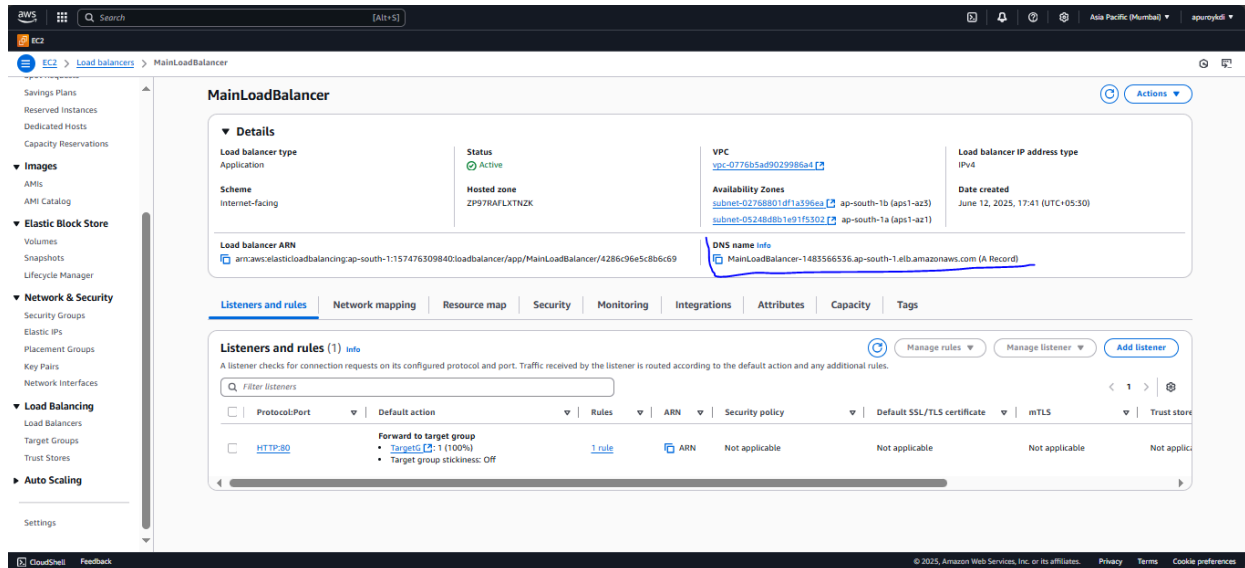
```
#apt update -y ---- (To update the system . )
#apt install apache2 ----- (To install apache2 server into the systemc)
#stemctl status apache2 ----(To check the status the service is running or not)
#systemctl start apache2 ---- (To start the service if not started)
#cd /var/www/html ----- (The default location of apache server where we need to add the web
pages )
#echo "<head>This is 1st page from the server 1</head>" > index1.html ----- (First page with
the extension of .html)
#echo "<head>This is 2nd page from the server 1</head>" > index2.html ----- (Second page
with the extension of .html)
#echo "<head>This is 3rd page from the server 1</head>" > index3.html ----- (Third page with
the extension of .html)
#systemctl restart apache2 ---- (To restart the service to the updation and execution of the
pages into the web server)
#
```

2nd Server is Ubuntu:



13.200.222.252 This is the Public address of Server 2

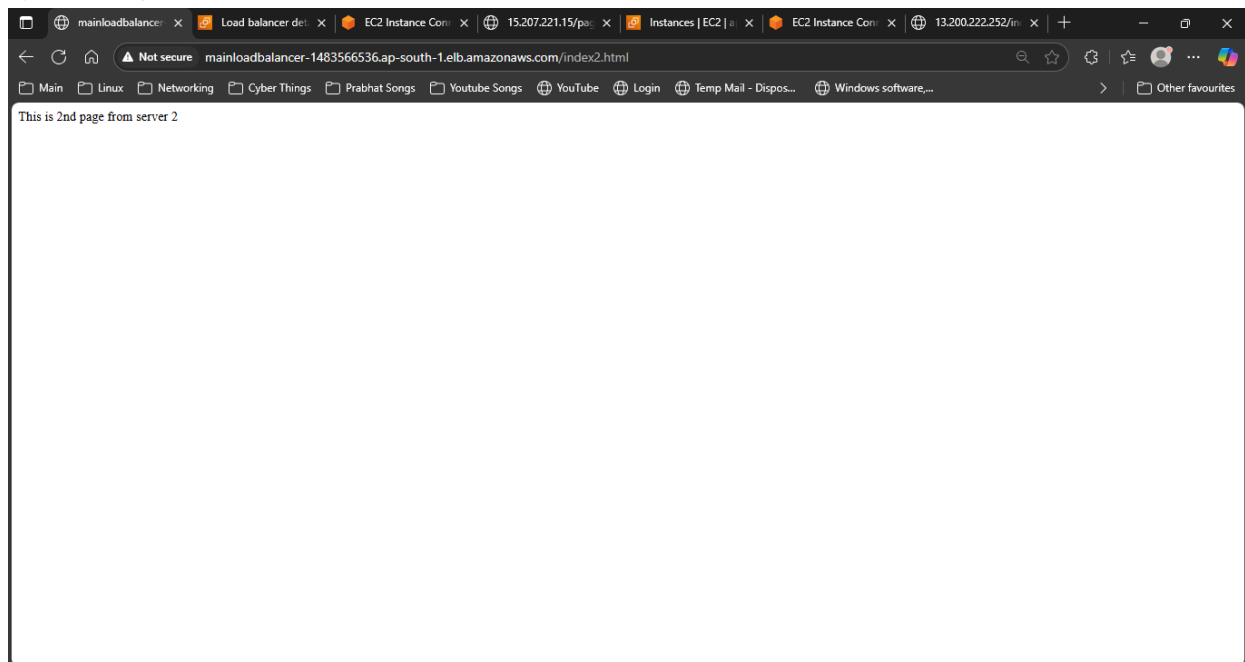
The Dashboard of Load Balancers:



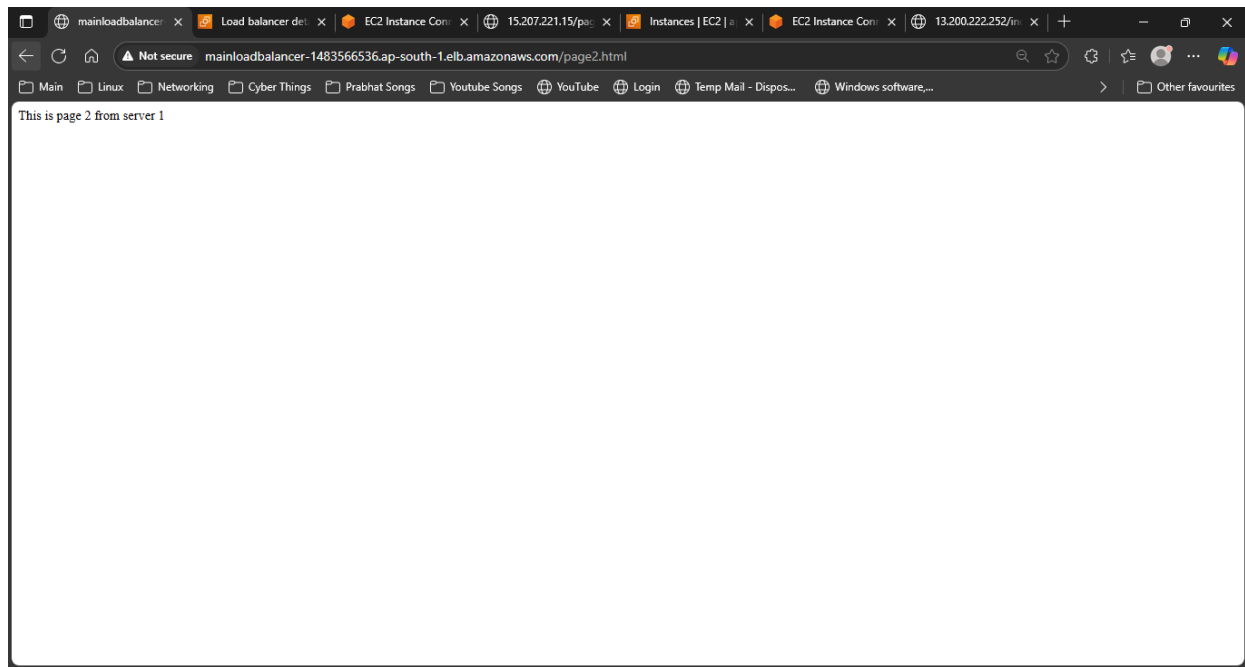
Here in the above picture , we can see lots of Links , each link is used for individual purposes. So for our case , we just take the **DNS** for balancing the loader .

MainLoadBalancer-1483566536.ap-south-1.elb.amazonaws.com By this link we can now access each page of that two servers's web pages . just suffixing with the name of the page we want to access . as shown below:

By suffixing with '/index2.html' we can access data from server-2.

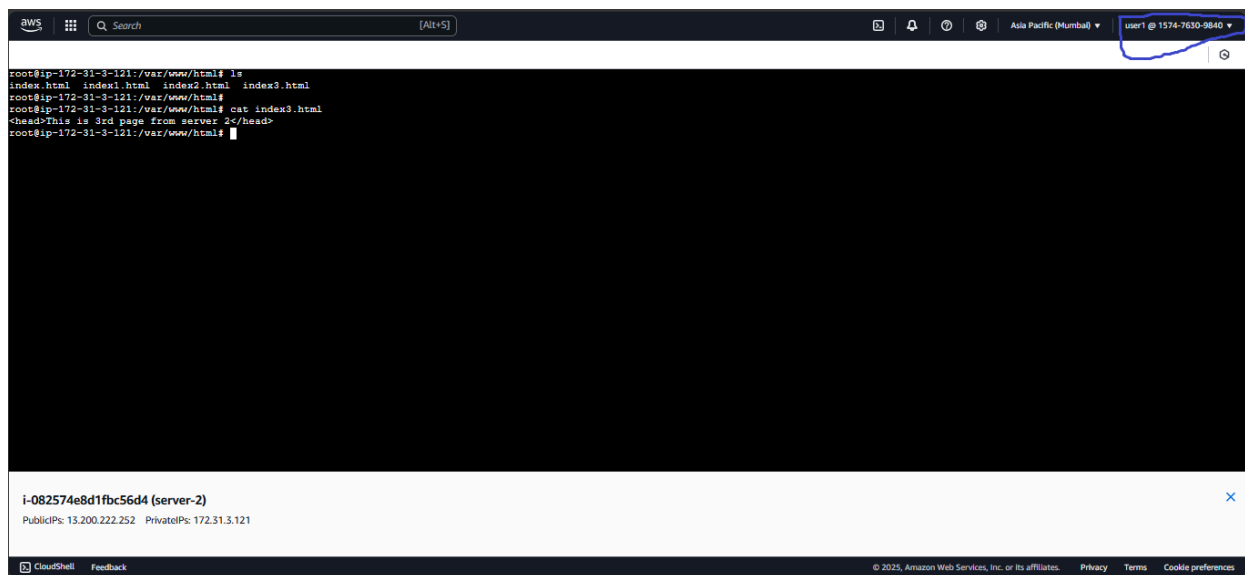


By suffixing with '/page2.html' we can access the data from server-1

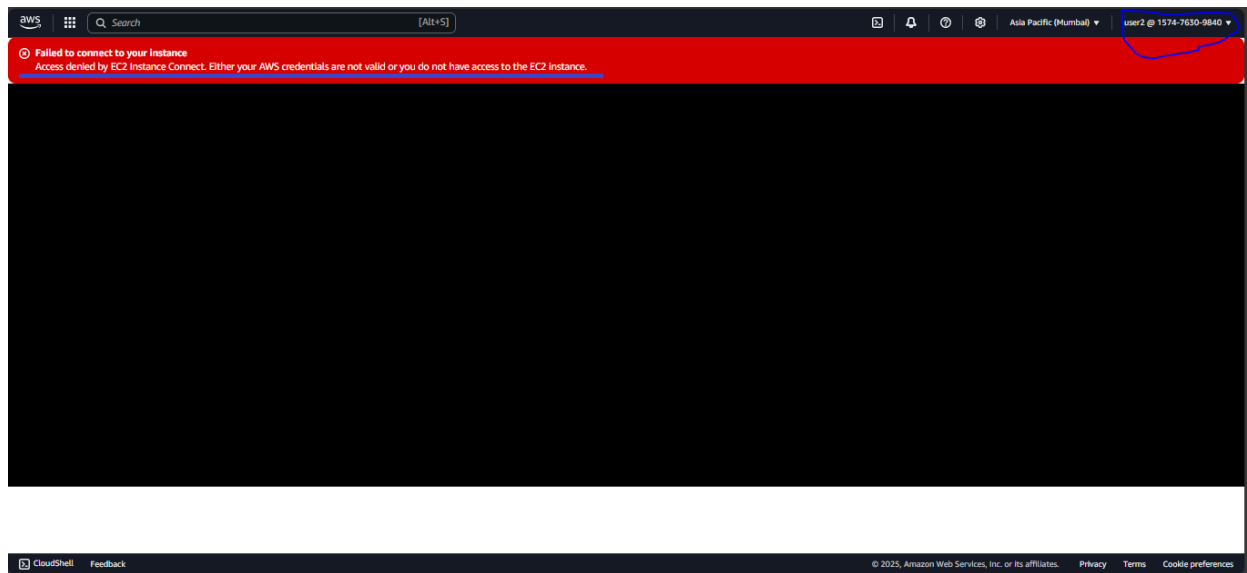


Every configuration is completed now let's check the permission by the **user1** and the **user2**.

User1: user1 has successfully accessed the terminal of the server-1. user1 has the full permission as Admin.



User2: user2 has no permission to access the terminal of any server, but has the read-only permission.



The End

