Load Balancers-Task

1. Configure Classic Load balancer.

➢ Classic Load Balancer is an AWS service that distributes incoming traffic across multiple EC2 instances to improve:

- Availability
- Fault tolerance
- Scalability

It works at Layer 4 (TCP) and Layer 7 (HTTP/HTTPS).

➢ **Key Features**

- Supports **HTTP, HTTPS, TCP, SSL**
- Performs **health checks** on instances
- Routes traffic only to **healthy instances**
- Works with **EC2 Auto Scaling**
- Provides **DNS name** for access
- Can be **internet-facing** or **internal**

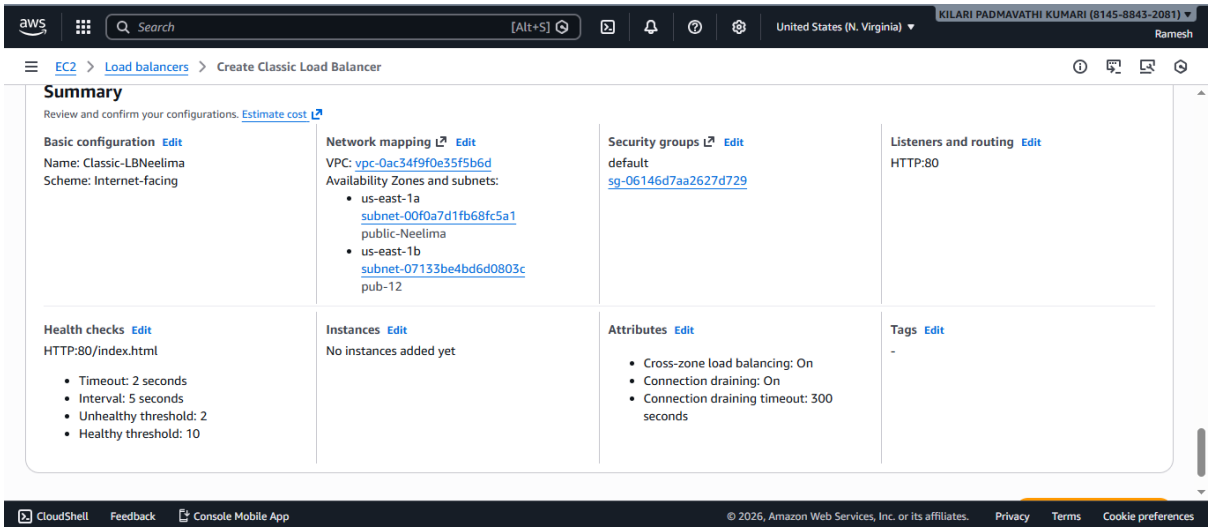
➢ **CLB Workflow**

1. User sends request to CLB DNS.
2. CLB receives request.
3. Health check verifies instances.
4. CLB forwards traffic to **healthy EC2**.
5. If one fails, traffic is sent to others.

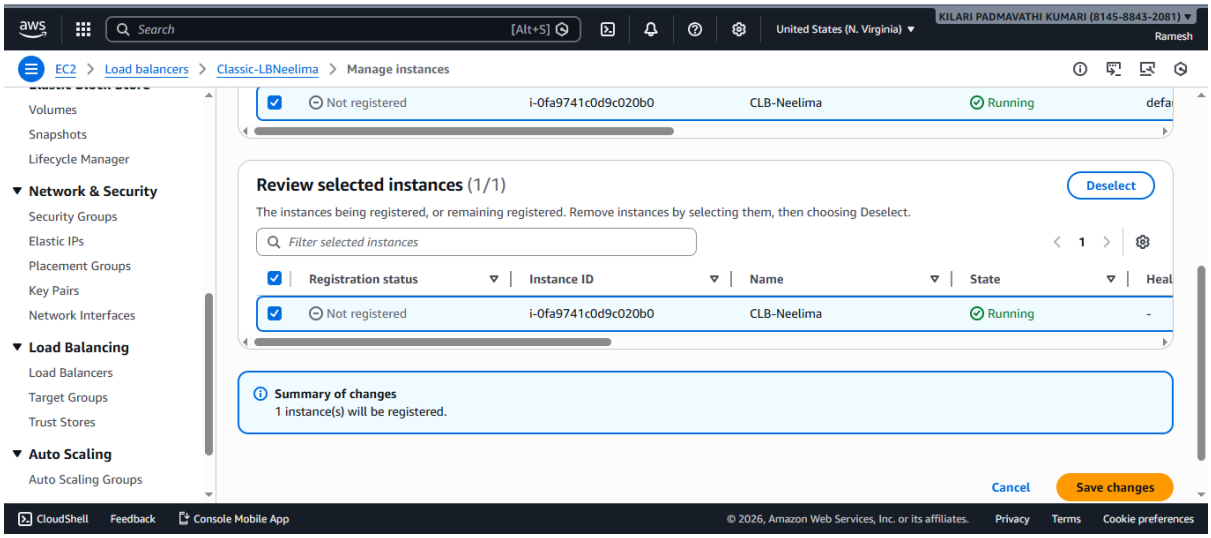
➢ **Classic Load Balancer – Configuration Steps**

- **Step 1: Open CLB**
- AWS Console → **EC2** → **Load Balancers** → **Create Load Balancer**
  Choose **Classic Load Balance.**

Load Balancers-Task

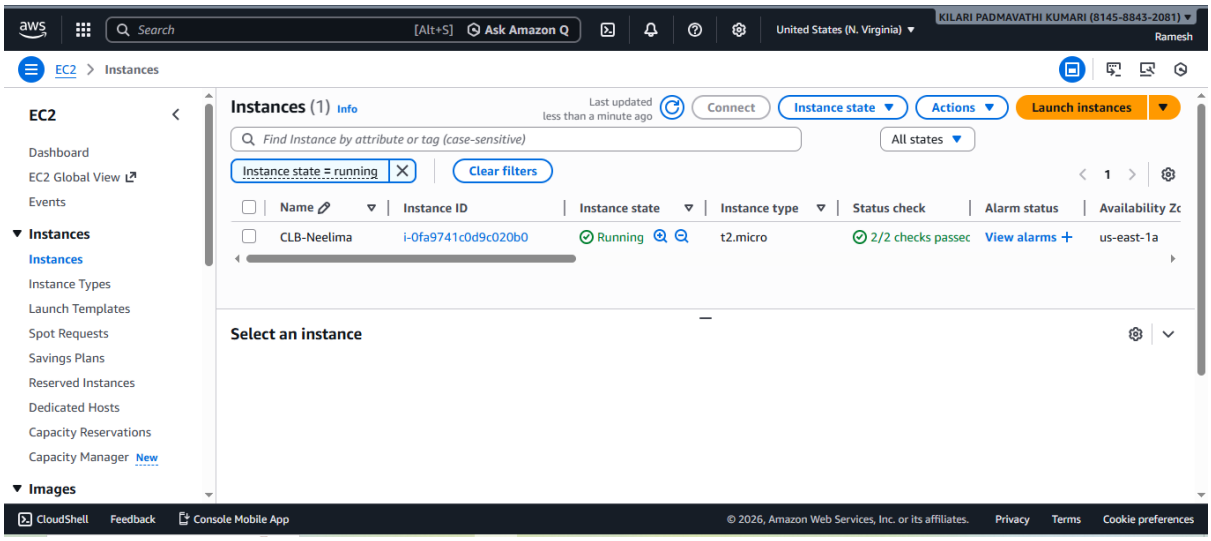


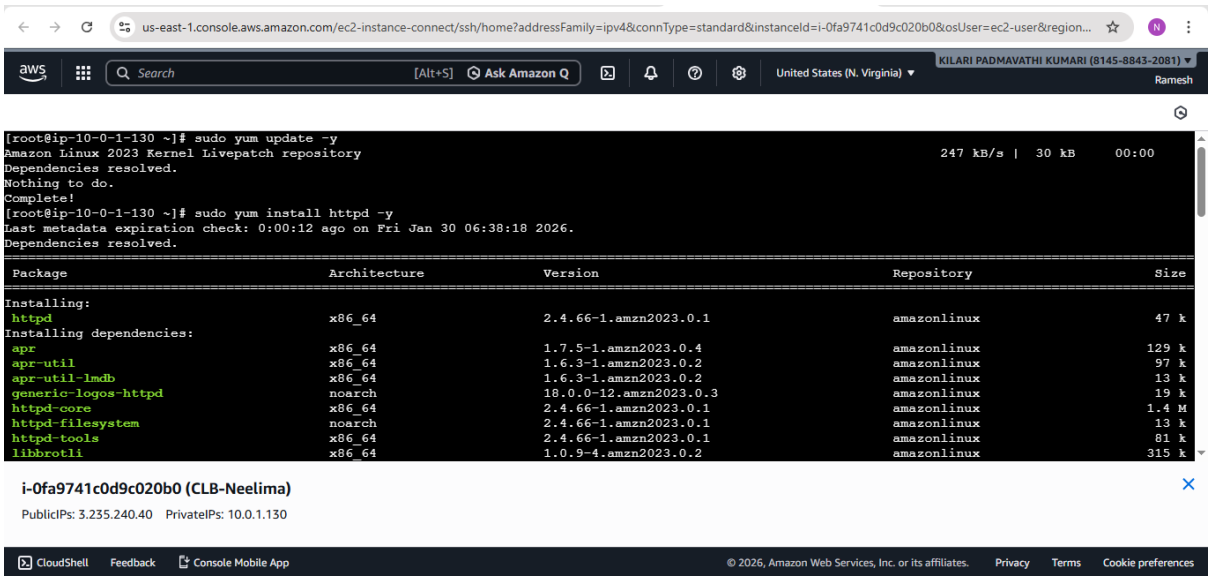
- **Add EC2 Instances**
- Select your running EC2
- Click **Add to Registered**



Load Balancers-Task

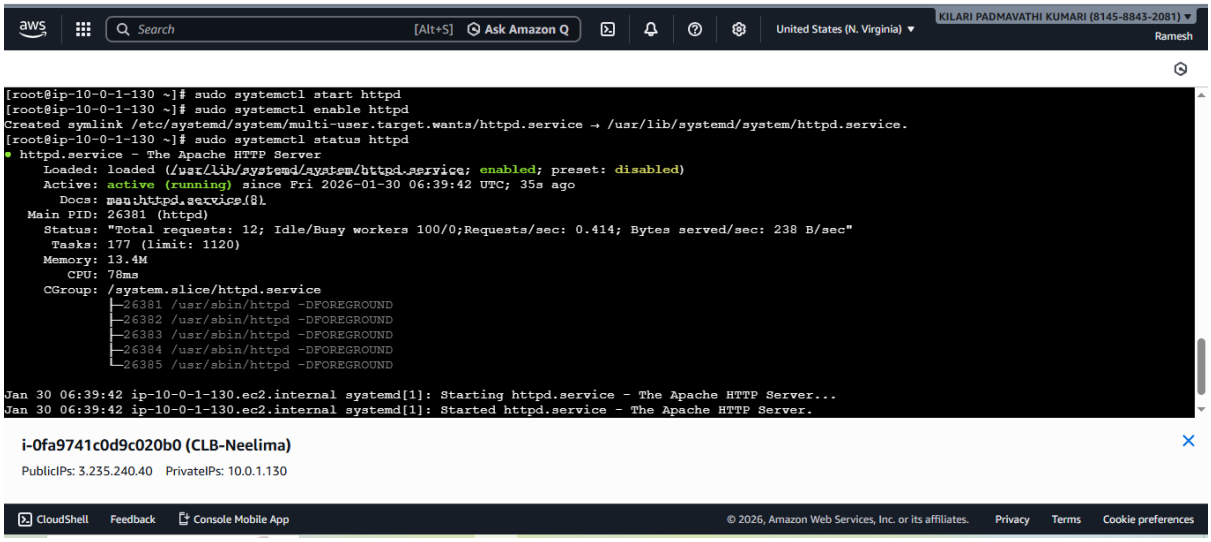
- Go to AWS console
- Create a EC2 instance





- Connect instance through ssh
- Switch to root user for more privileges
- Install httpd service in EC2
- Commands are as follows:
- Sudo yum update -y
- Sudo yum install httpd -y

Load Balancers-Task



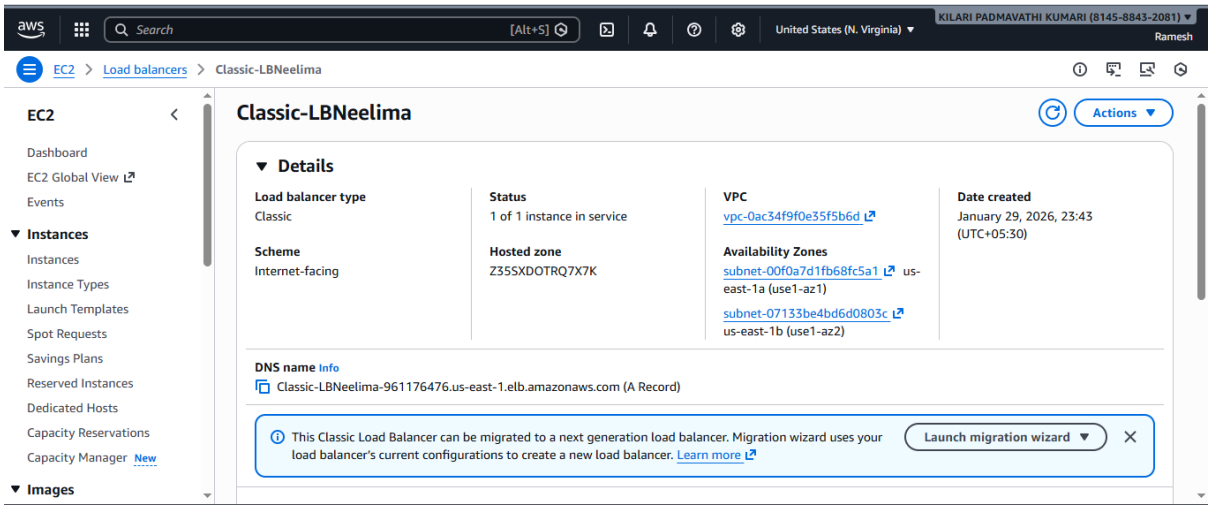
- Sudo systemctl start httpd
- Sduo systemctl enable httpd
- Sudo systemctl status httpd (check it is running or not)

```
[root@ip-10-0-1-130 ~]# echo "<h1>CLB Working - Neelima</h1>" | sudo tee /var/www/html/index.html
<h1>CLB Working - Neelima</h1>
[root@ip-10-0-1-130 ~]# echo "<h1>CLB Working - Neelima</h1>" | sudo tee /var/www/html/index.html
<h1>CLB Working - Neelima</h1>
```

- Create a static webpage in ec2 command as follows
- echo “<h1>CLB working – Neelima</h1> |sudo tee /var/www/html/index.html
- check whether it is running in local machine with public ip
- Locally it is working

CLB Working - Neelima

Load Balancers-Task



- Go ec2→ Load balancers
- Click on created load balancer
- Copy DNS name paste link in browser and check for output.

- “The Classic Load Balancer DNS acts as a single public endpoint. It receives client requests, performs health checks, and forwards traffic to healthy EC2 instances based on the configured listener rules.”

Load Balancers-Task

2. Configure Application Load balancer.

An ALB is a type of AWS Load Balancer that operates at the Layer 7 (Application Layer) of the OSI model. This means it can route traffic based on content of the request, such as:

- Hostname (Host header)
- URL path (/images, /api)
- HTTP headers, query strings, or source IP

Key features:

- Supports HTTP and HTTPS (web traffic).
- Content-based routing: Sends requests to different target groups based on URL or hostname.
- Works with EC2, ECS, Lambda, IP addresses as target

When to use ALB:

- You have multiple web applications on the same domain or server.
- You want to route /api requests to one service and /web requests to another.
- You need SSL termination and security policies at the load balancer level.

Key Components of ALB

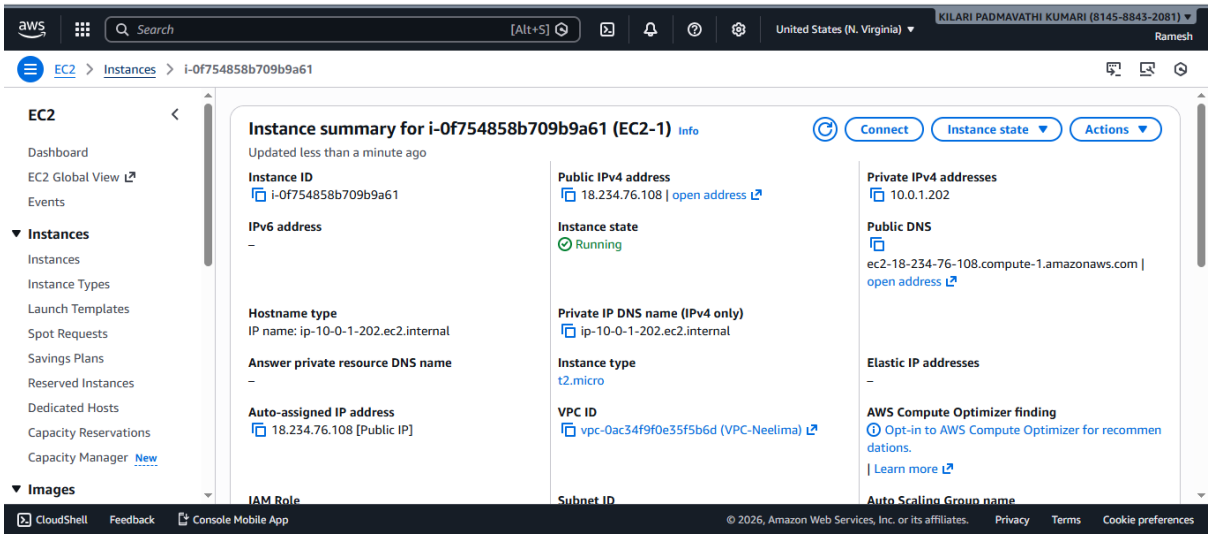
1. **Load Balancer** – The front-end endpoint that client's access.
2. **Listeners** – Check for incoming traffic on a port (usually 80 for HTTP or 443 for HTTPS).
3. **Target Groups** – Logical group of backend servers (EC2 instances, IPs, Lambda).
4. **Rules** – Define how requests are routed to different target groups.
5. **Health Checks** – Ensure traffic is sent only to healthy targets.

Load Balancers-Task

Configuration Steps in AWS Console

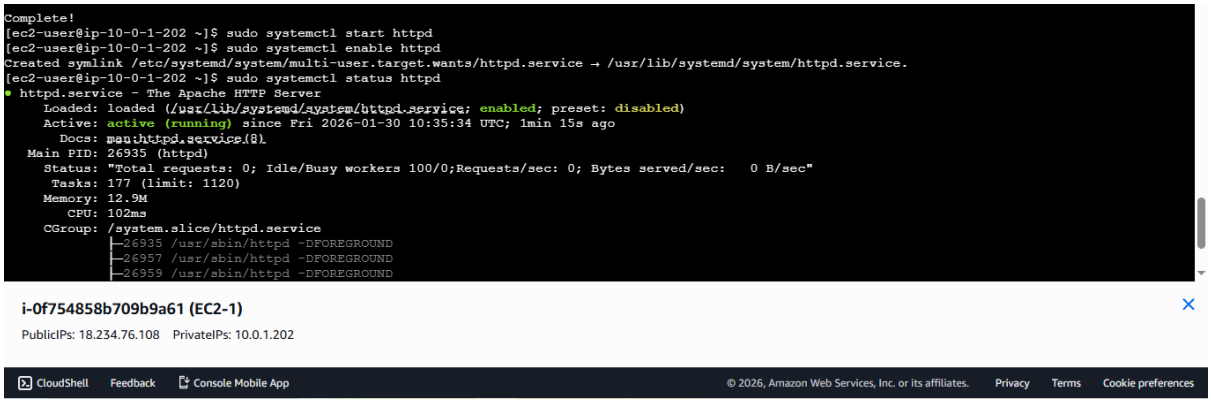
Step 1: Launch EC2 Instances

- Launch **2 or more EC2 instances** with a web server (Apache/Nginx).
- Make sure they are in **different subnets** (preferably in different AZs for high availability).
- Go to AWS search Ec2 instance
- Launch instance with public ip

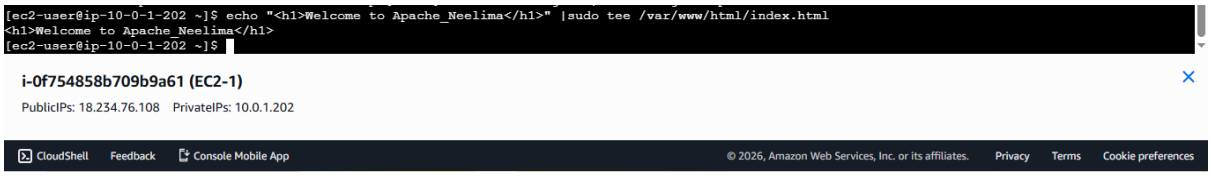


- Connect to instance through ssh connection
- Install apache in EC2-1 instance commands are as follows
    - Sudo yum update -y
    - sudo yum install httpd -y

Load Balancers-Task



- sudo systemctl start httpd
- sudo systemctl enable httpd
- sudo systemctl status httpd

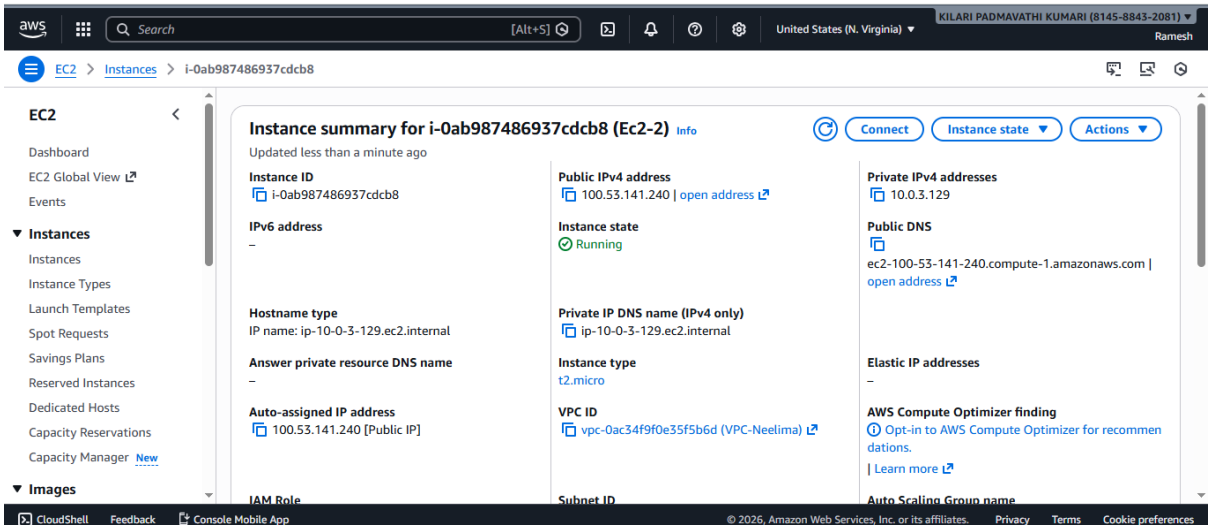


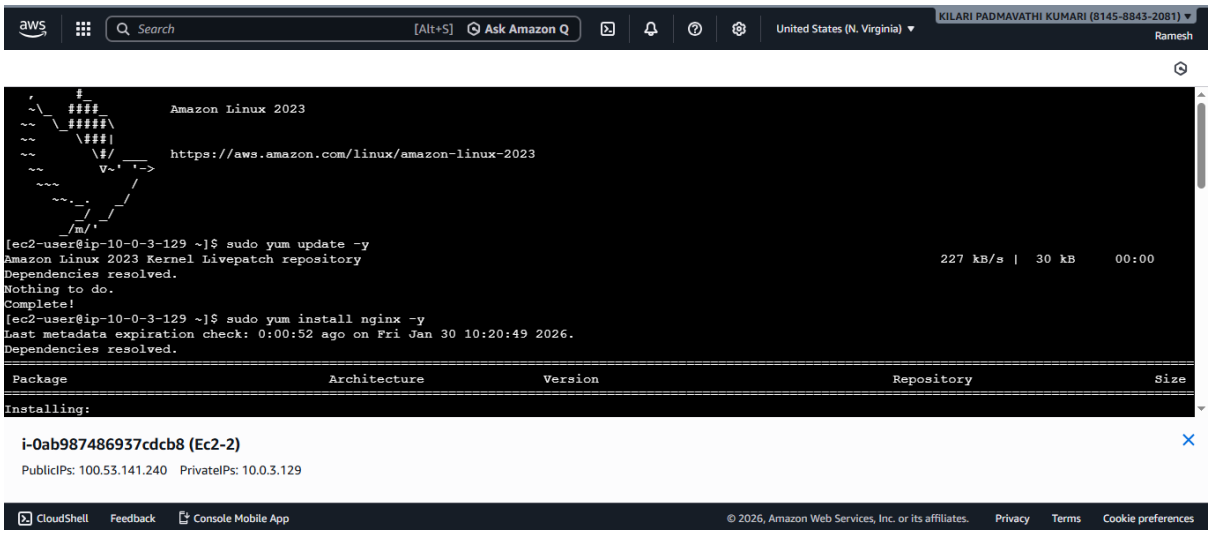
- Create webpage in Instance using html
- echo “<h1> message </h1> |sudo tee /var/www/html/index.html

- After webpage creation check locally using public ip

Load Balancers-Task

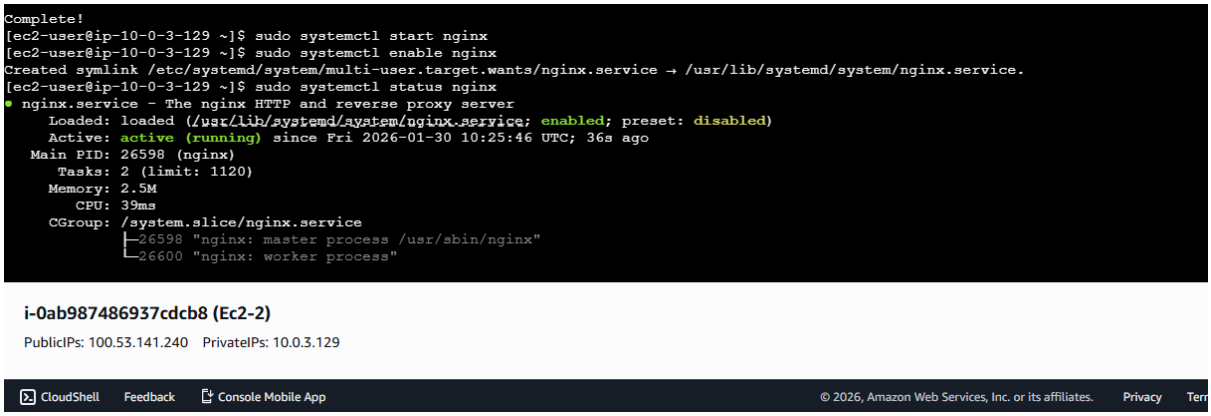
- Go to AWS search Ec2 instance
- Launch instance(EC2-2) with public ip



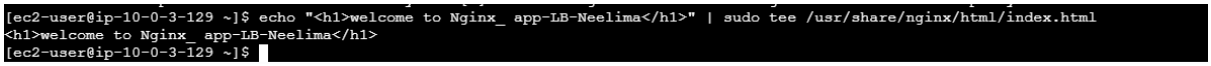


- Connect to instance using ssh connection
- Install Nginx in ec2 instance and commands are as follows
  - sudo yum update -y
  - sudo yum install nginx -y

Load Balancers-Task



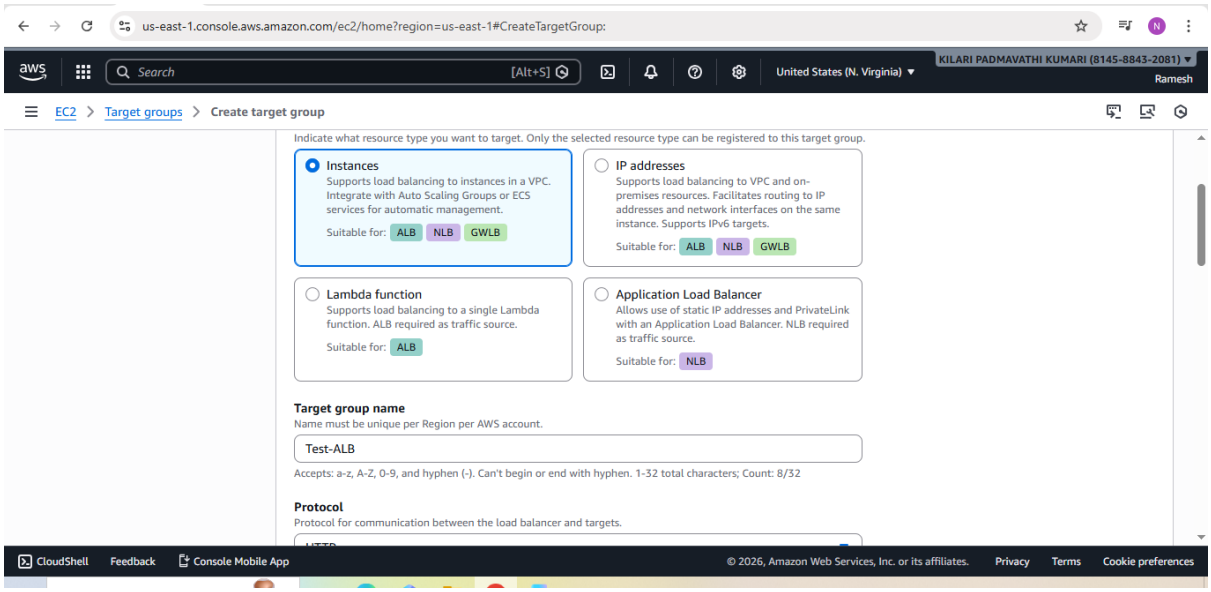
- sudo systemctl start nginx
- sudo systemctl enable nginx
- sudo systemctl status nginx



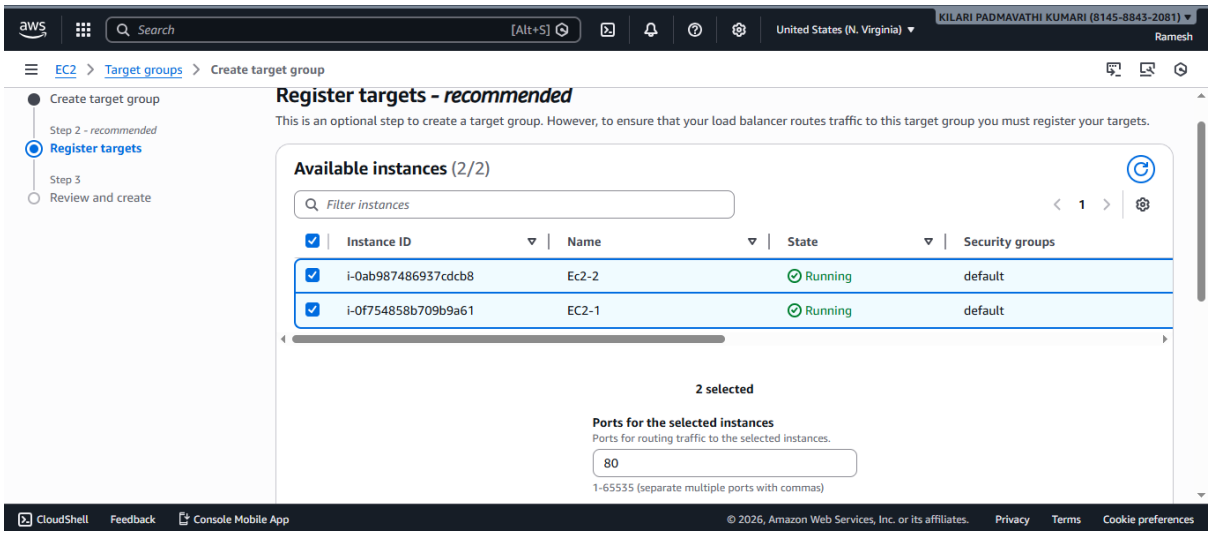
- Create a webpage in ece instance using html
- echo “<h1>Message</h1>” | sudo tee /usr/share/nginx/html/index.html

- Test webpage using public ip in local machine.

Load Balancers-Task

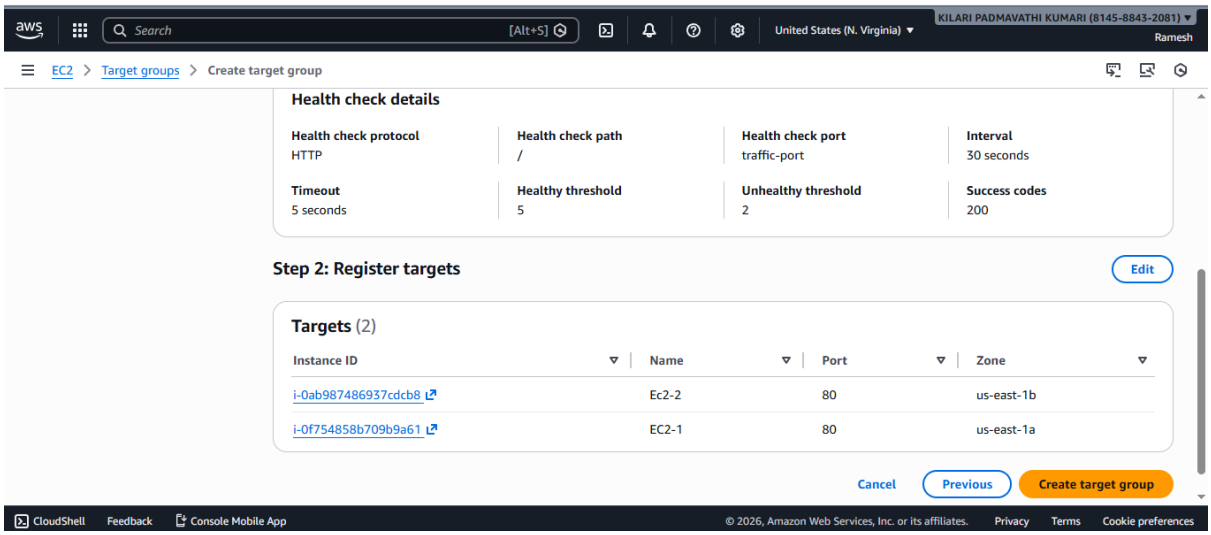


- Now Go to EC2 →Target Groups →create target groups
- Select instances
- Enter Target group Name and enter details and
- click on Next

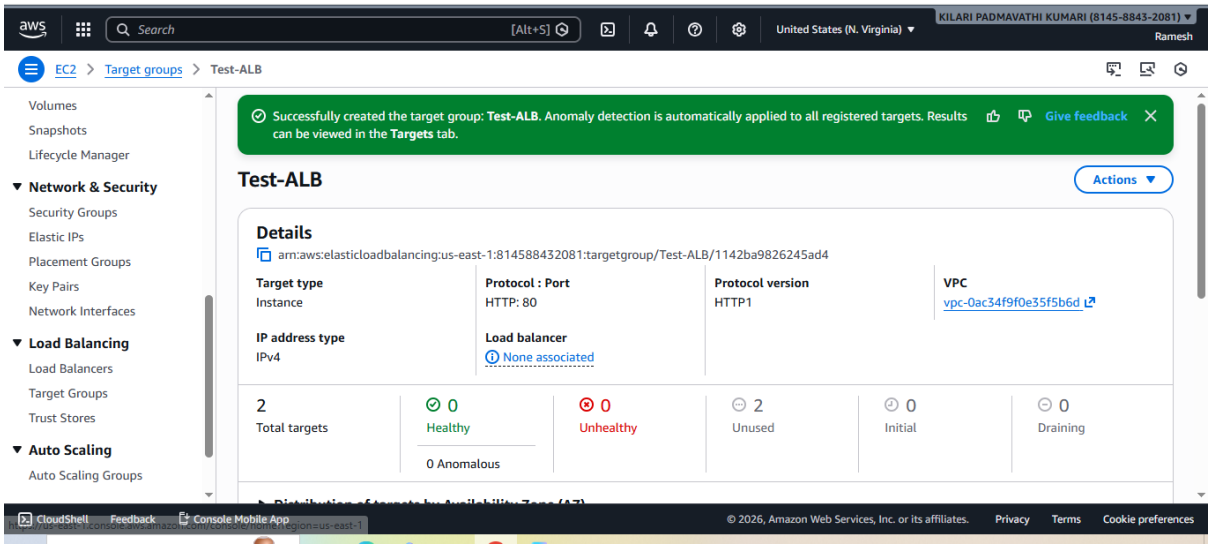


- In Register targets Add Instances we required.
- Ports for the selected instances click on included as

Load Balancers-Task

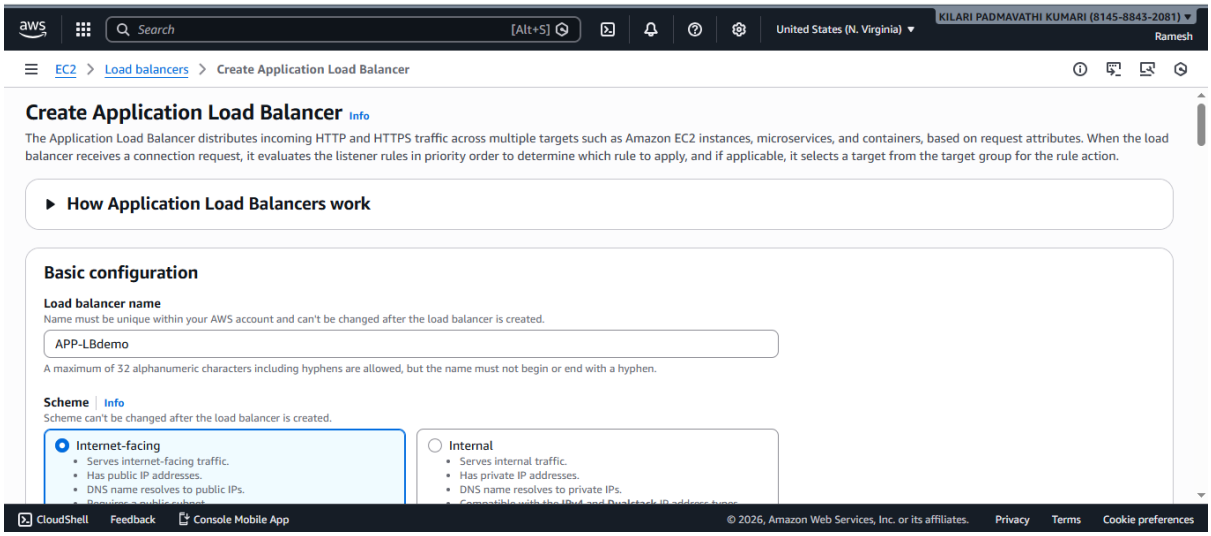


- Click on Create target group

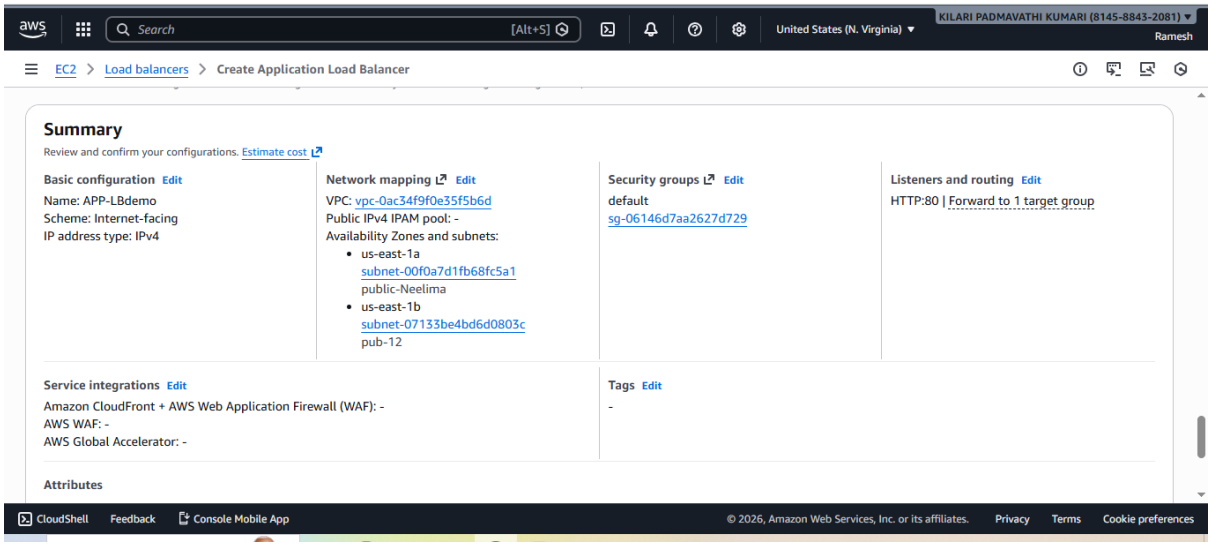


- The above image shows Target groups created successfully and we can information in details.

Load Balancers-Task

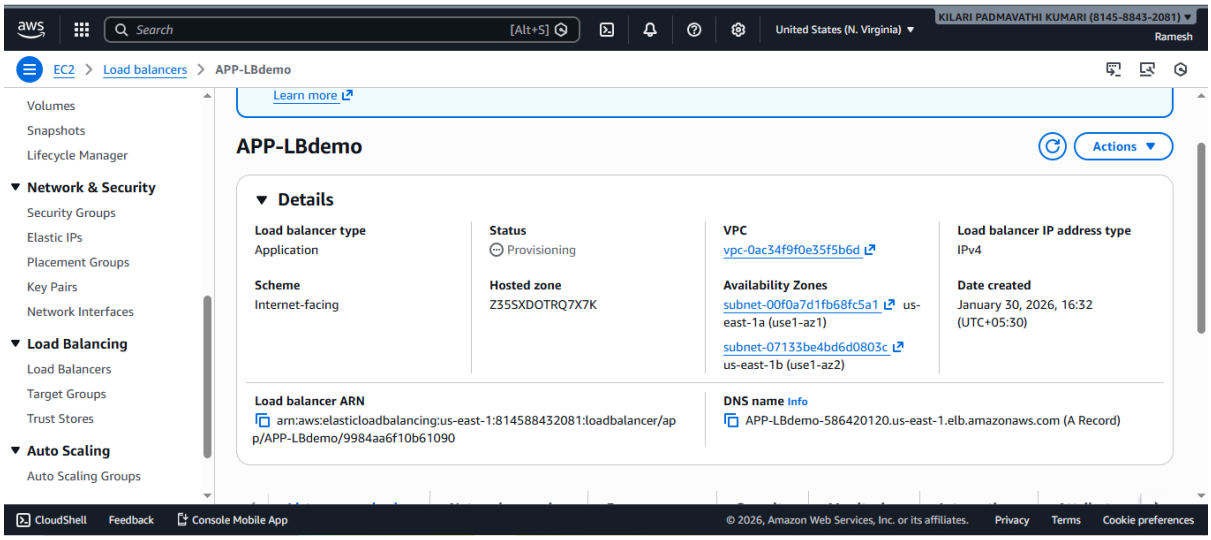


- Now, Create Application Load Balancer
- EC2→ Load Balancers → create Application Load Balancer
- In Basic Configuration→ Enter Load Balancer Name
- Scheme→Internet-facing
- Add VPC and Subnets,Security groups, listeners and routing



- In summary we see complete configurations.
- Click on create.

Load Balancers-Task



- Click on App-LBdemo create before.
- In details we have DNS name,copy it
- Paste DNS name in Browser and check output.

Conclusion

An Application Load Balancer (ALB) allows us to distribute incoming web traffic to multiple backend servers. In this setup, we successfully configured:

- One EC2 instance running Apache other EC2 instance running Nginx
- Both accessed through a single Application Load Balancer

Load Balancers-Task

- The ALB forwards each request to **only one server at a time**, using a round-robin method. That is why refreshing the ALB URL shows either the Apache page or the Nginx page.
- This proves that the **Application Load Balancer efficiently manages and routes traffic**, improves availability, and allows multiple applications to run behind a single endpoint.

Load Balancers-Task

3. Configure Network Load balancer.

A Network Load Balancer (NLB) operates at Layer 4 (TCP/UDP) of the OSI model.It forwards traffic based on IP address and port, not on HTTP content.

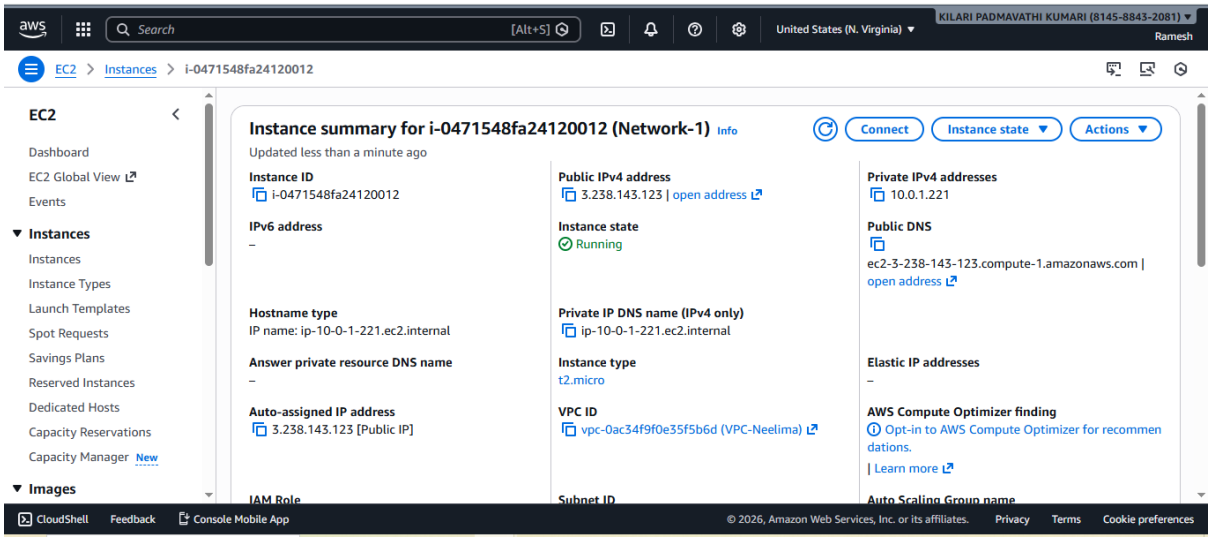
➢ It is used for:

- Very high traffic
- Low latency
- TCP, UDP, TLS based applications
- When you need static IP addresses

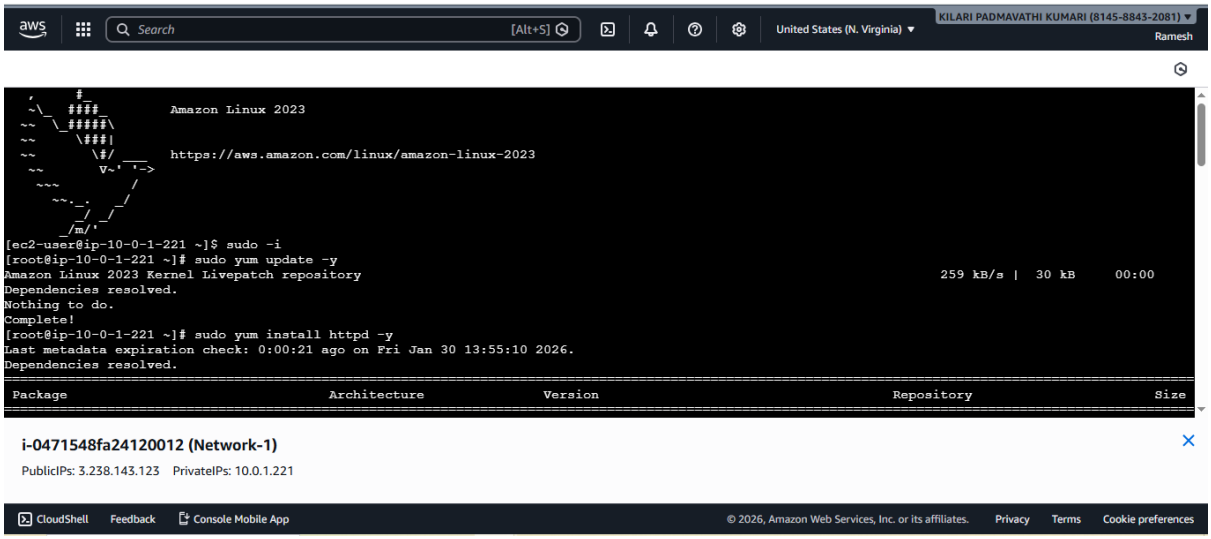
Configuration Steps

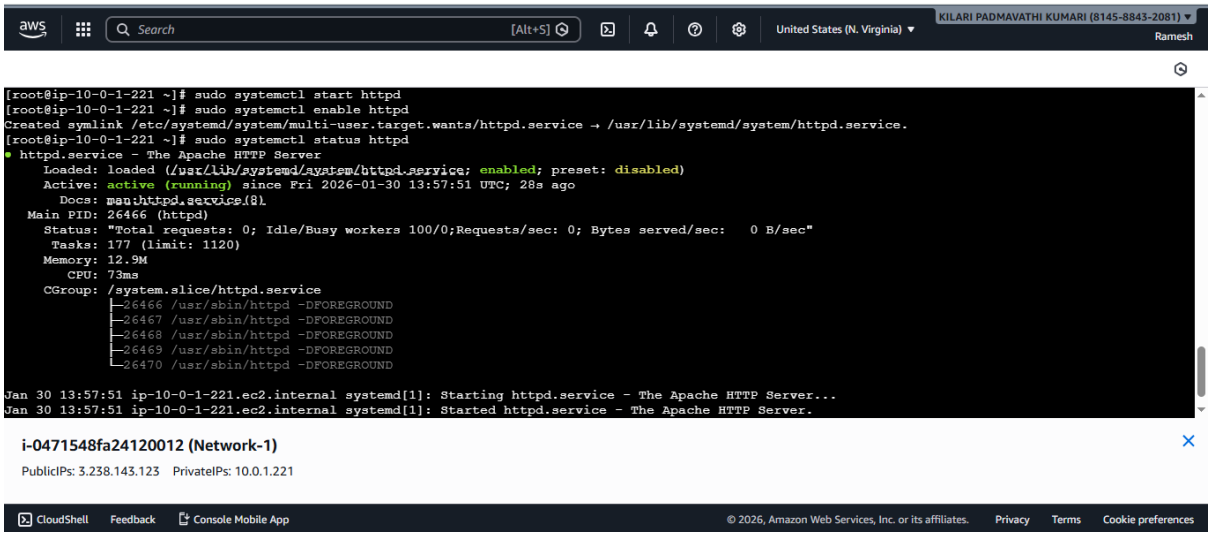
Step 1 – Launch EC2 Instances

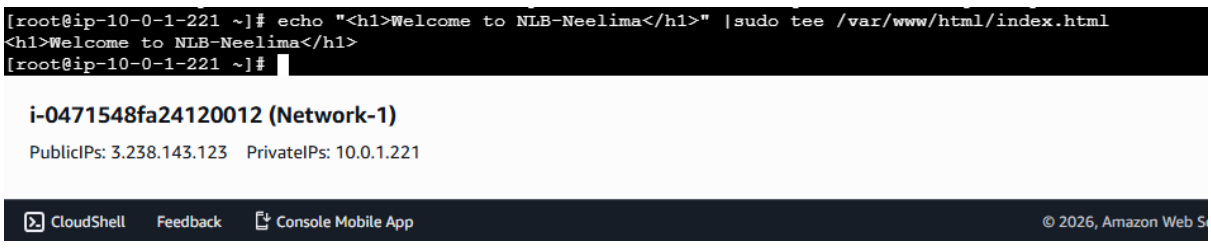
- Create 2 EC2 instances
- Install any server (Apache / Nginx / Tomcat)
- Make sure both are in the same VPC

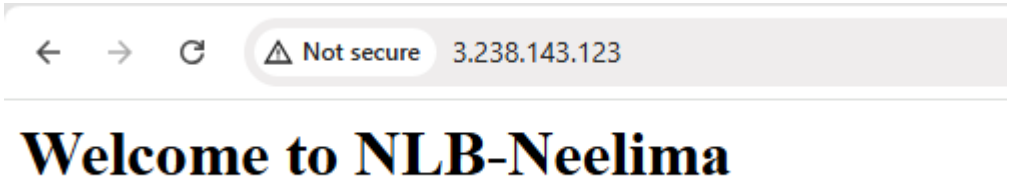


Load Balancers-Task

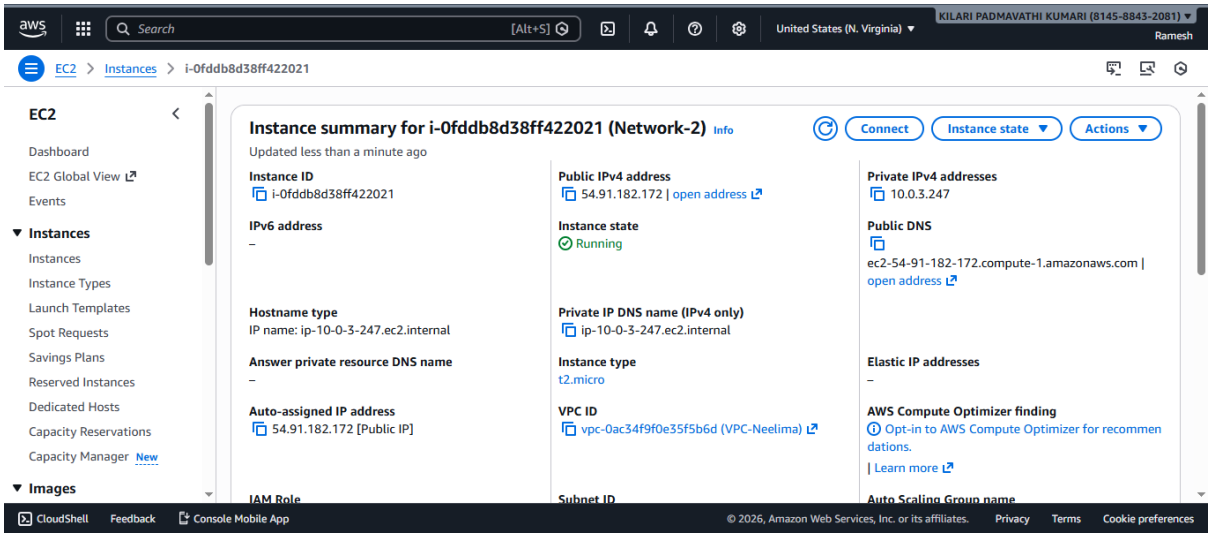


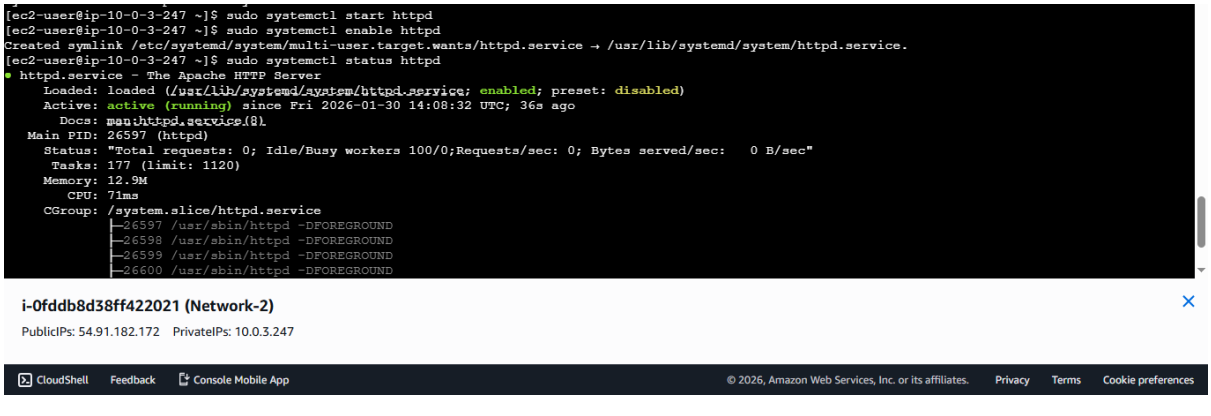




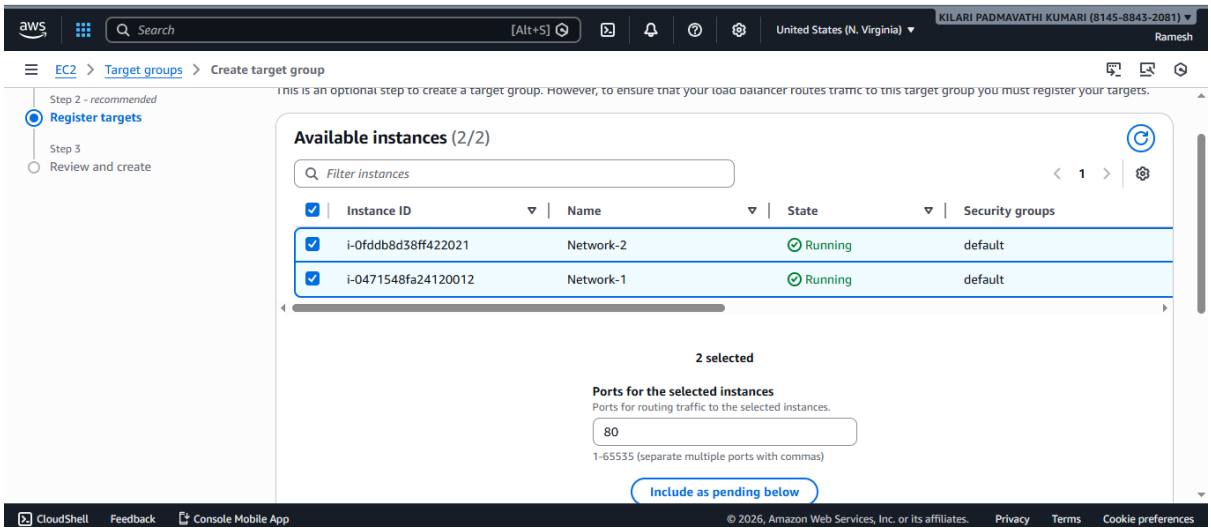


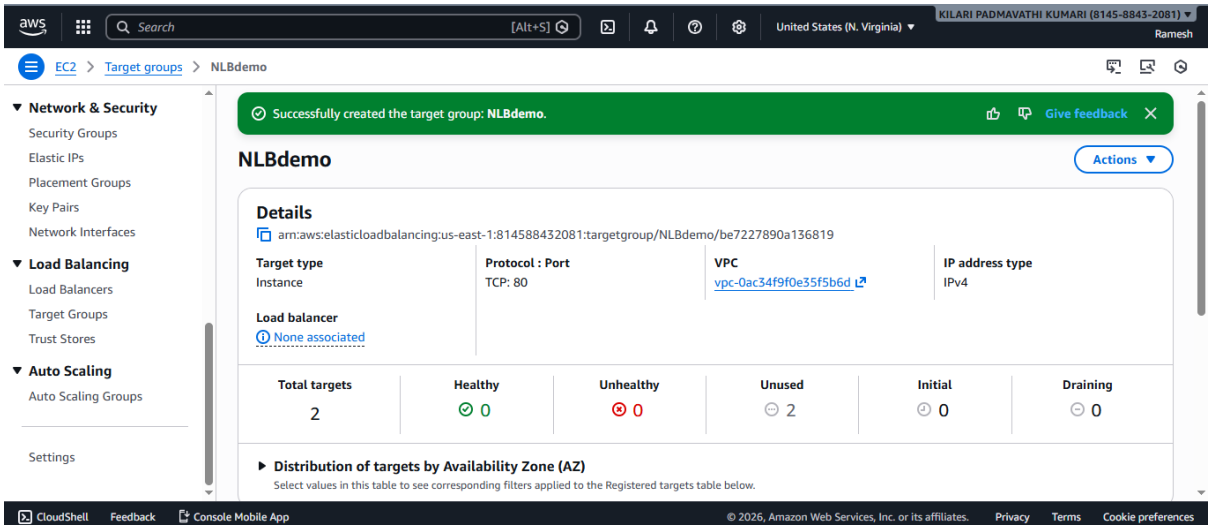
Load Balancers-Task



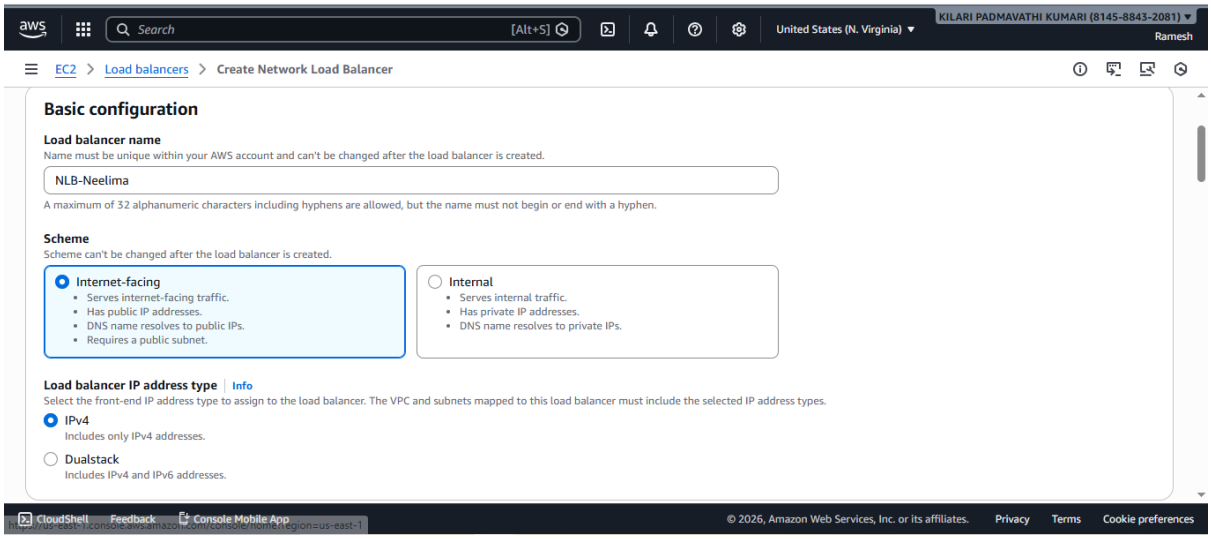


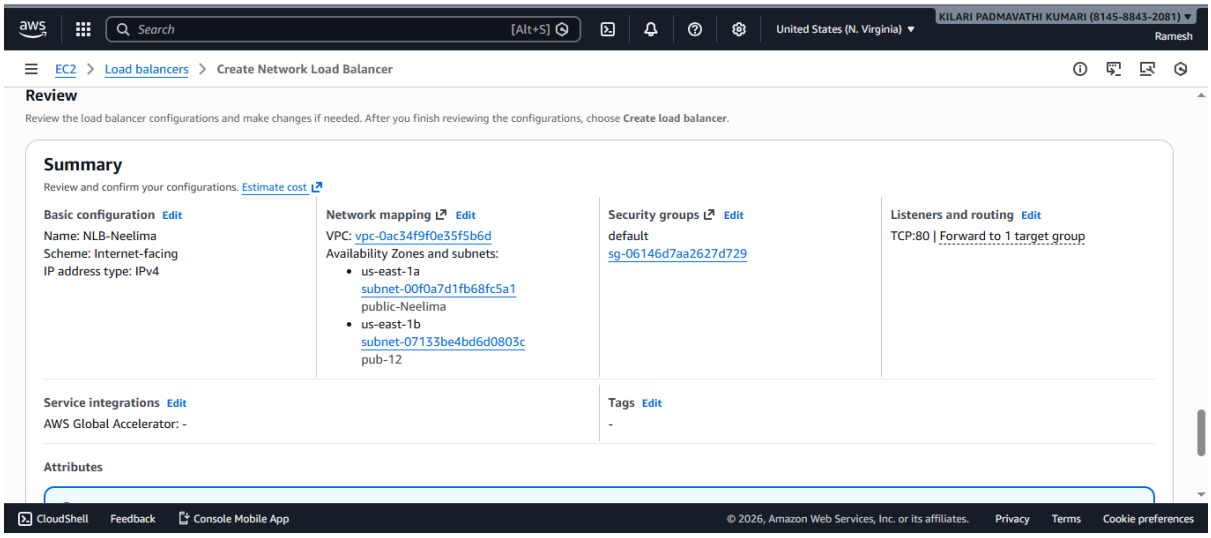
Load Balancers-Task

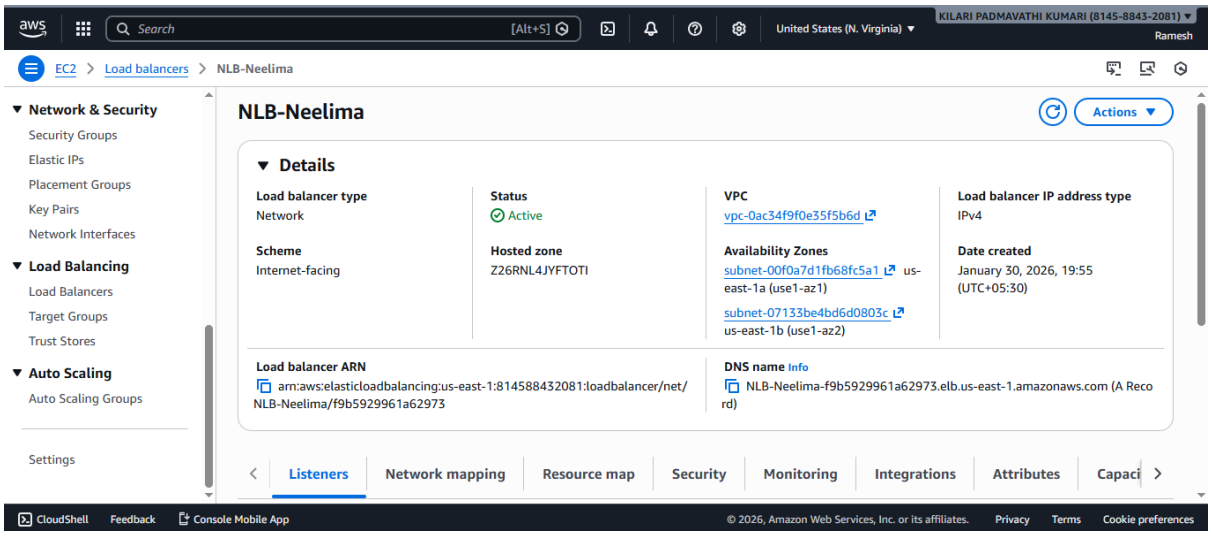




Load Balancers-Task



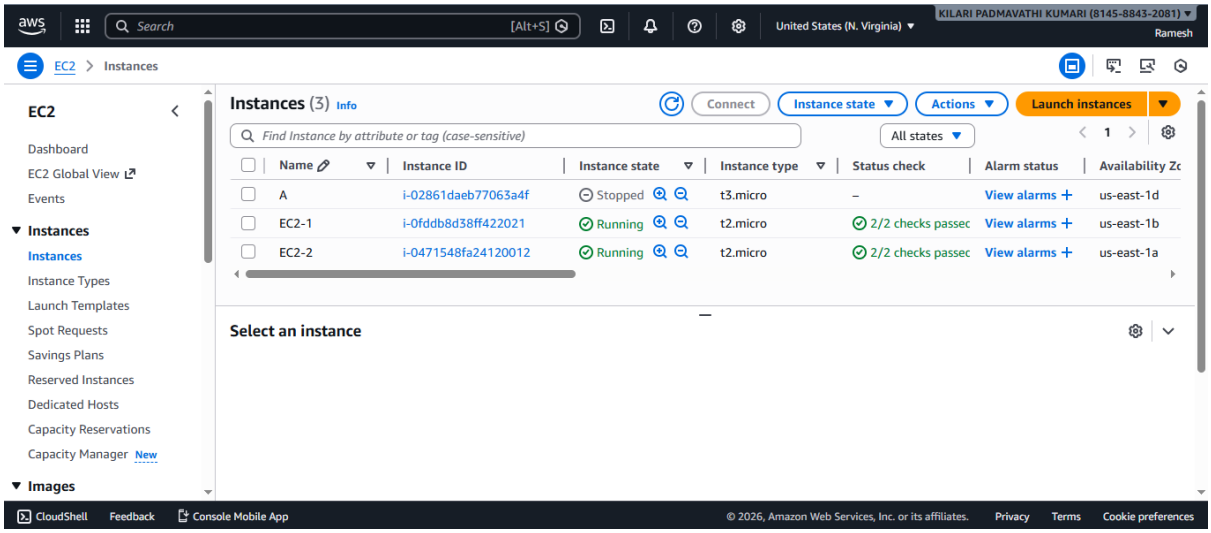




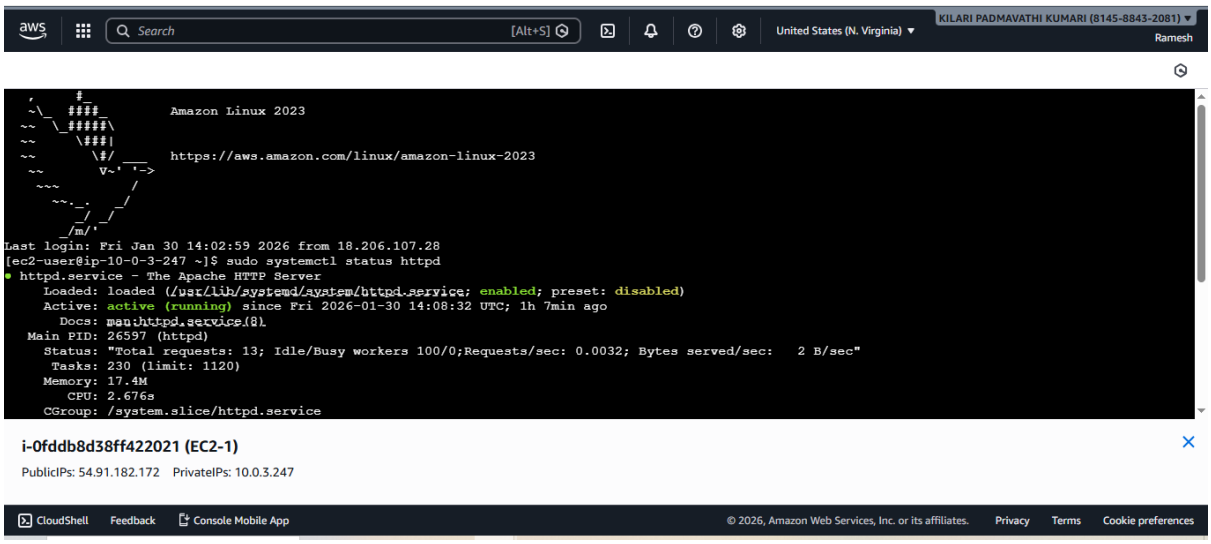
Load Balancers-Task

4. Attach SSL for application load balancer.

- Create Instances with public ip
- Make sure instances are running or not



Load Balancers-Task

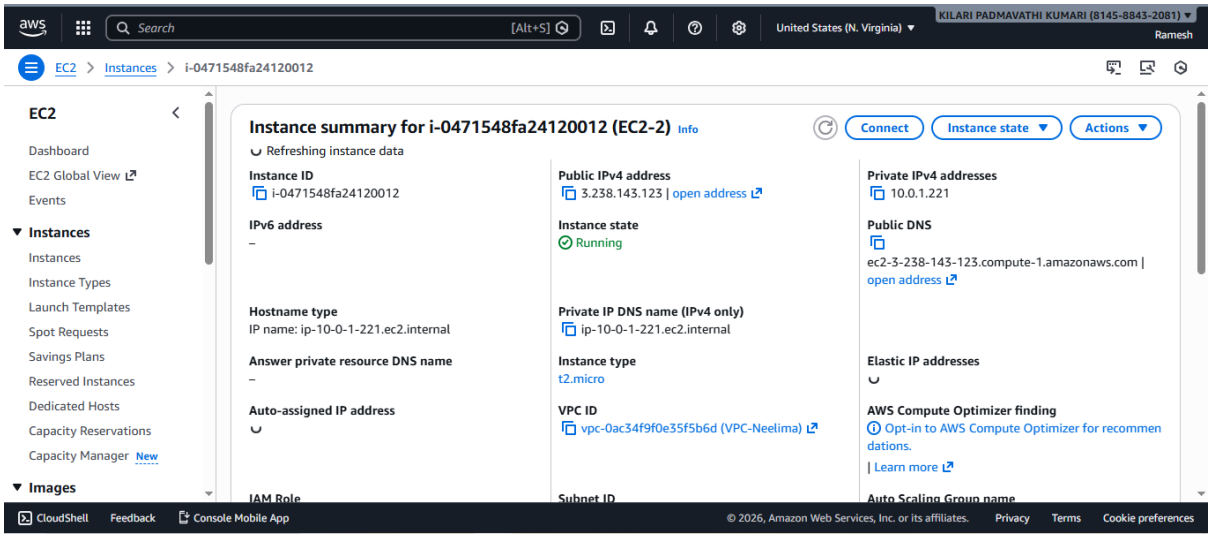


- Connect instance with ssh through public ip
- Instance must contain apache.
- Make apache is active and running.

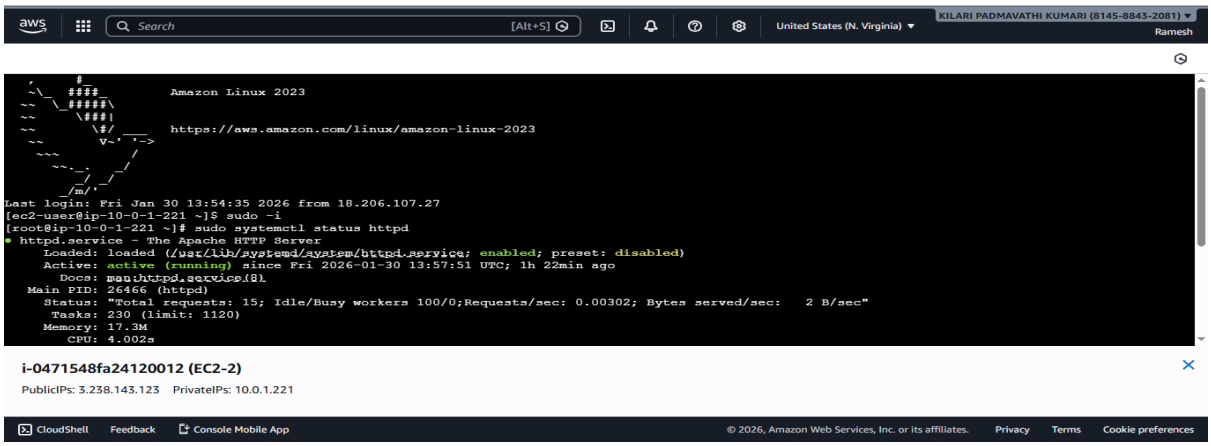
```
[ec2-user@ip-10-0-3-247 ~]$ echo "<h1>Welcome to App_LB-Neelima</h1>" |sudo tee /var/www/html/index.html
<h1>Welcome to App_LB-Neelima</h1>
[ec2-user@ip-10-0-3-247 ~]$
```

- Create a webpage
- Test it in local machine with public ip.

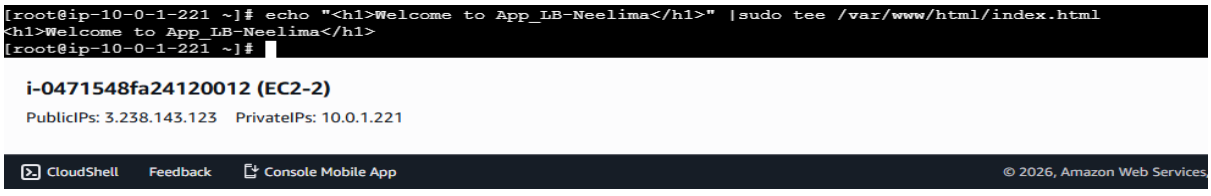
Load Balancers-Task



- Create other instance with public ip



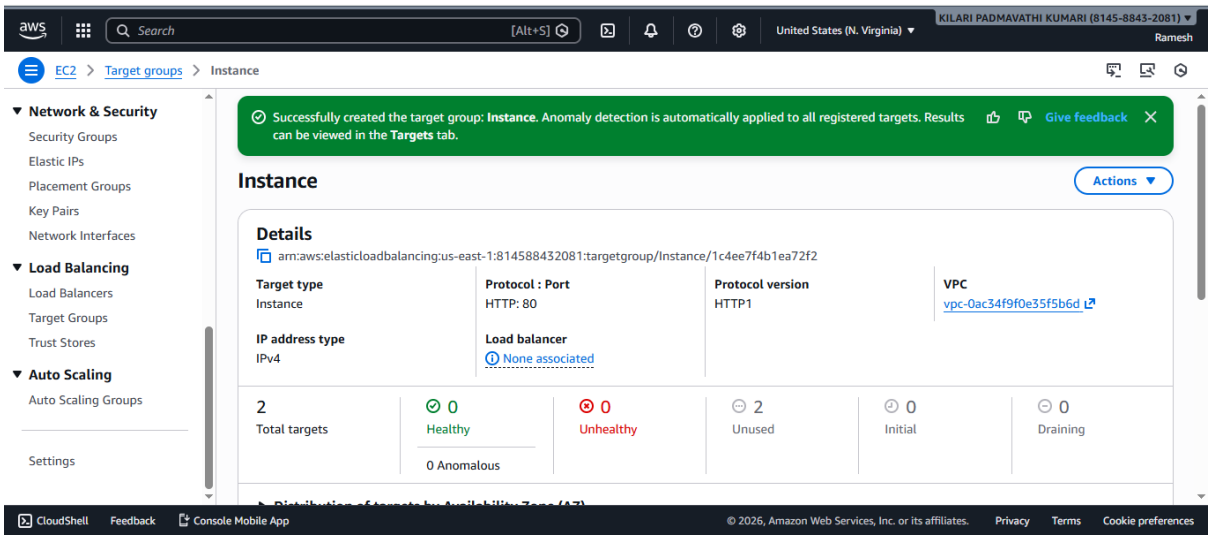
- Connect instance with public ip through ssh
- Check whether apache is active and running



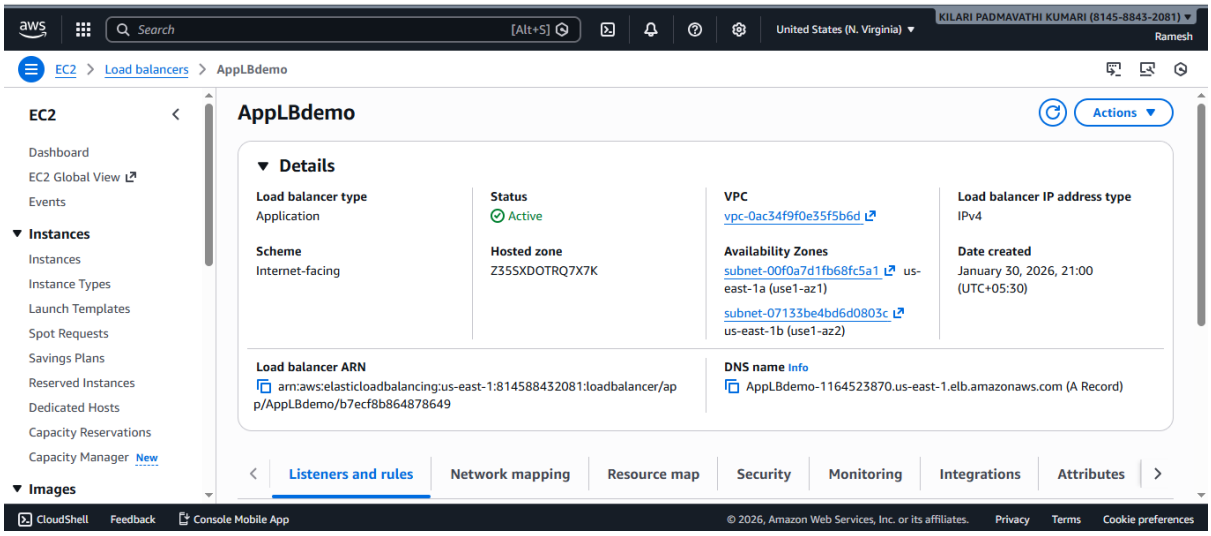
- Create a webpage within the instance
- Test it in browser within local machine.

Welcome to App_LB-Neelima

Load Balancers-Task



- Now go to Ec2→Target groups
- Create Target group with instances



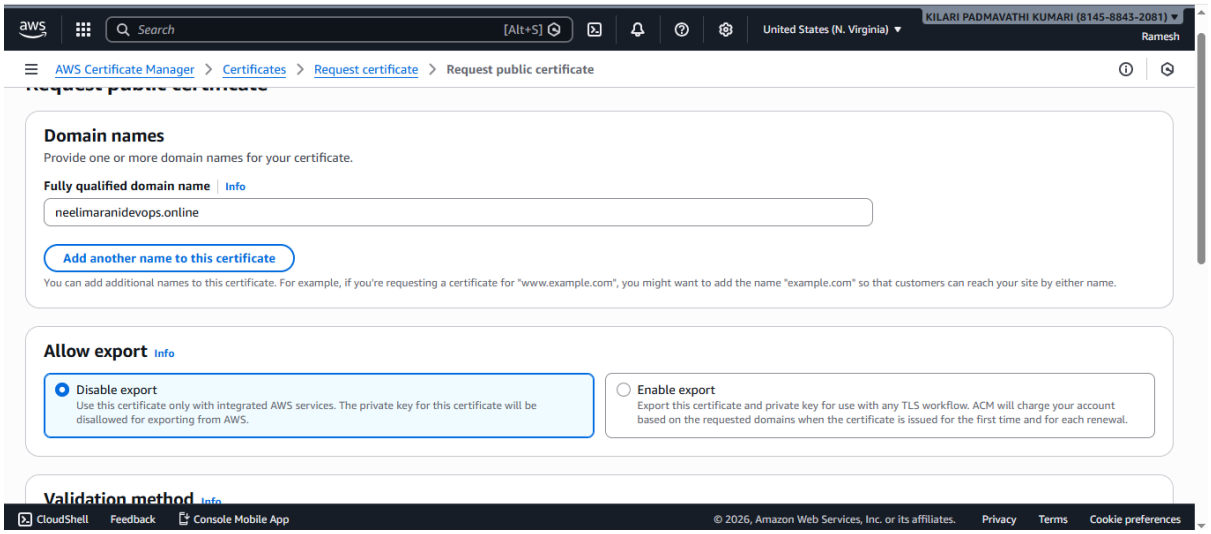
- EC2→load balancers→Application LB
- Create Application LB with VPC and subnet
- Internet facing
- Target groups.

Load Balancers-Task

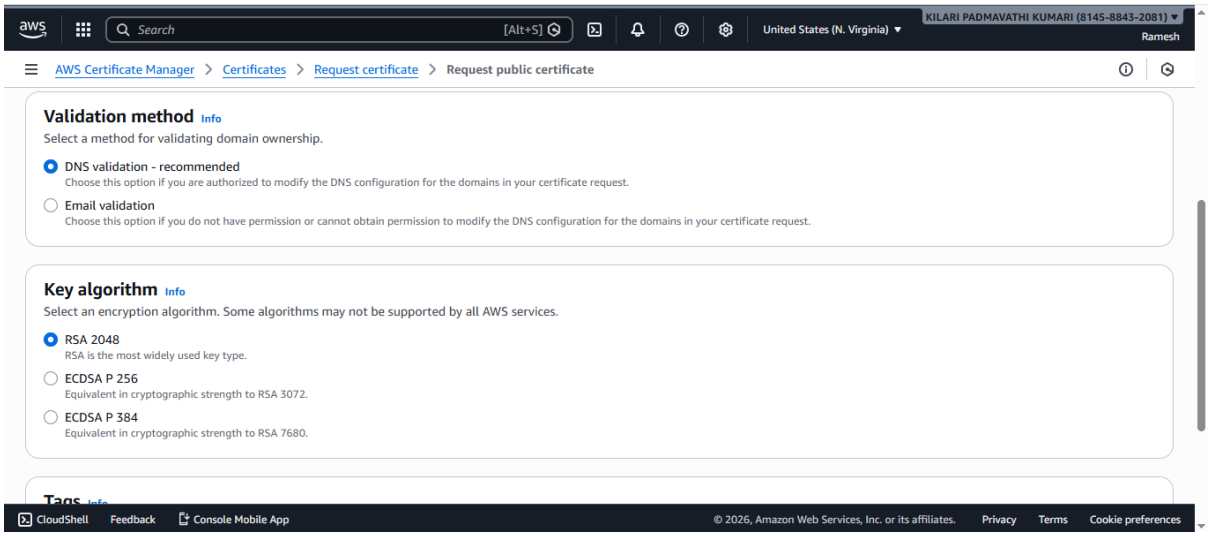
- To verify Copy DNS name and browse it

Step 1: Request SSL Certificate (ACM)

1. Open AWS Console → Certificate Manager (ACM)
2. Click Request certificate
3. Select Public certificate → Next
4. Enter your domain name
   Example:neelimaranidevops.online



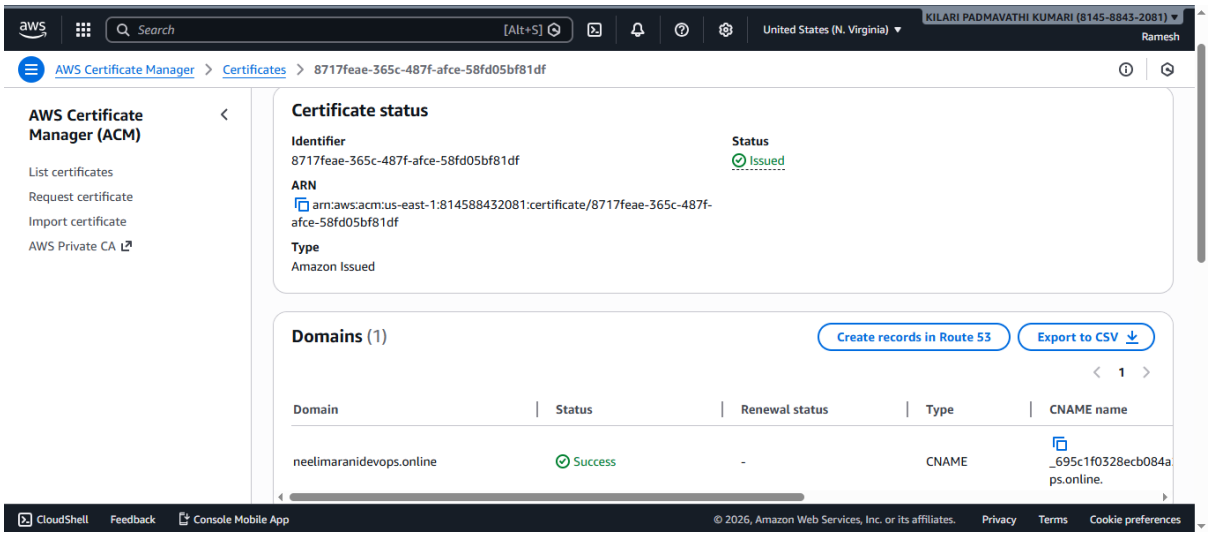
Load Balancers-Task



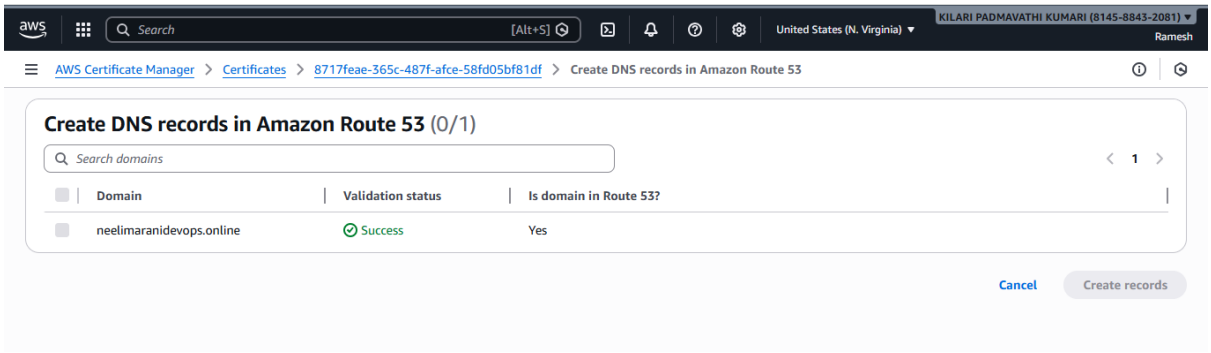
- Choose **DNS validation**
- Click **Request**

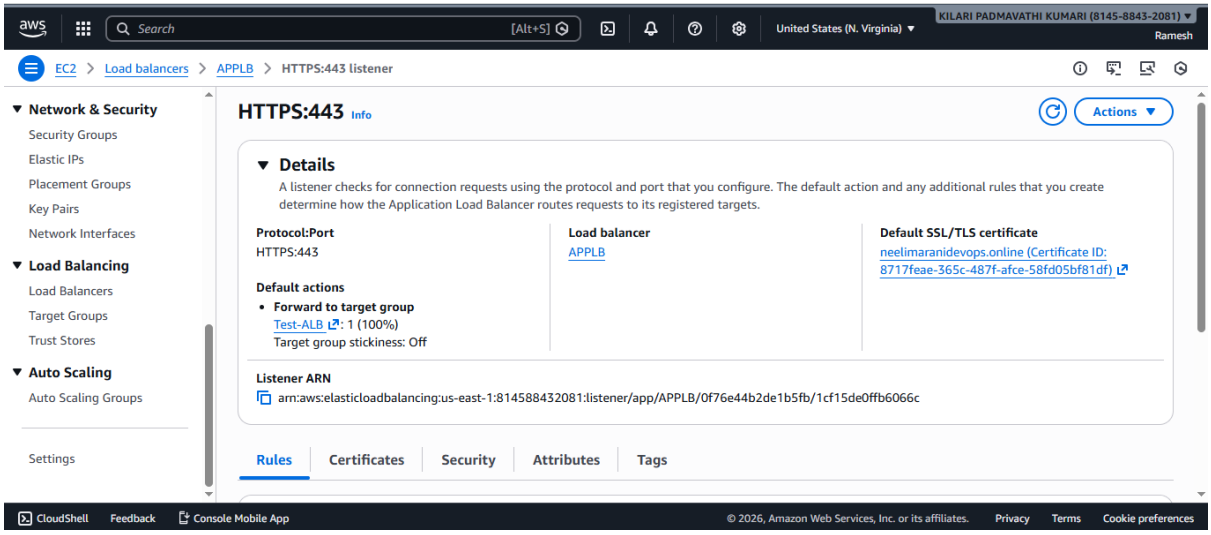
validate Domain

1. Open the certificate → click **Create record in Route53**
2. If domain is not in Route53 → copy CNAME and add in your DNS provider
3. Wait until **Status = Issued**



Load Balancers-Task

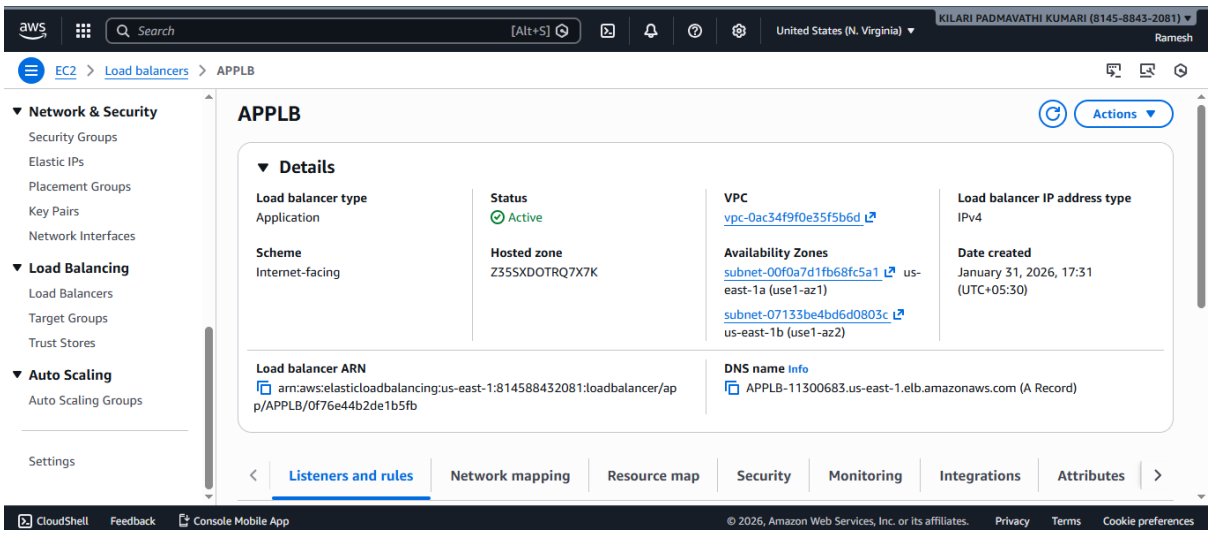




5. Map Application load balancer to R53

Step 1: Go to your ALB

1. Open **EC2 → Load Balancers**
2. Select your **Application Load Balancer**

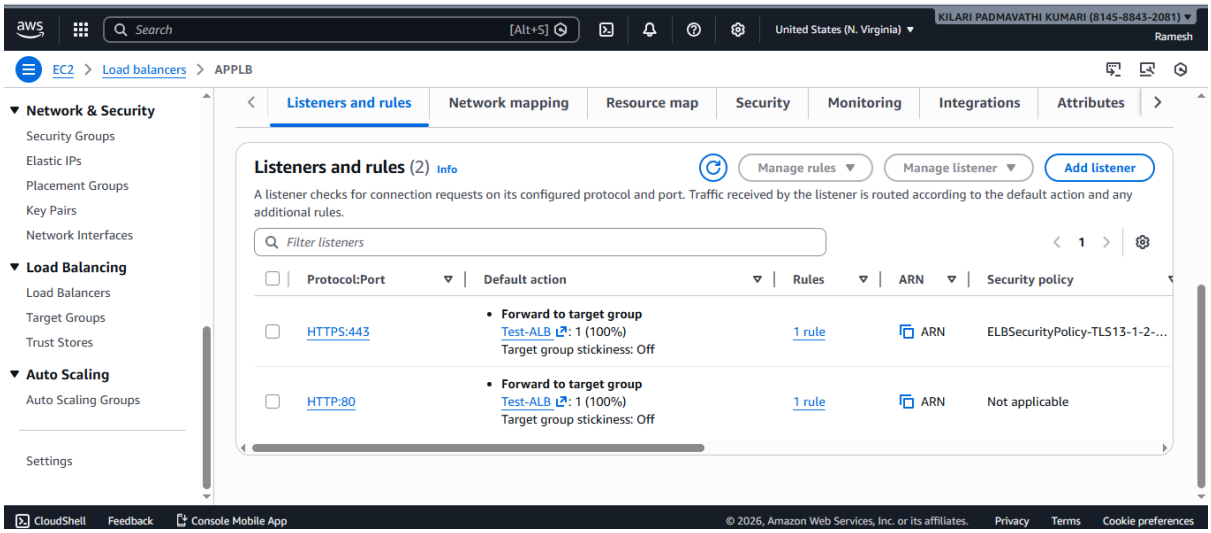


Load Balancers-Task

Step 2: Add HTTPS Listener

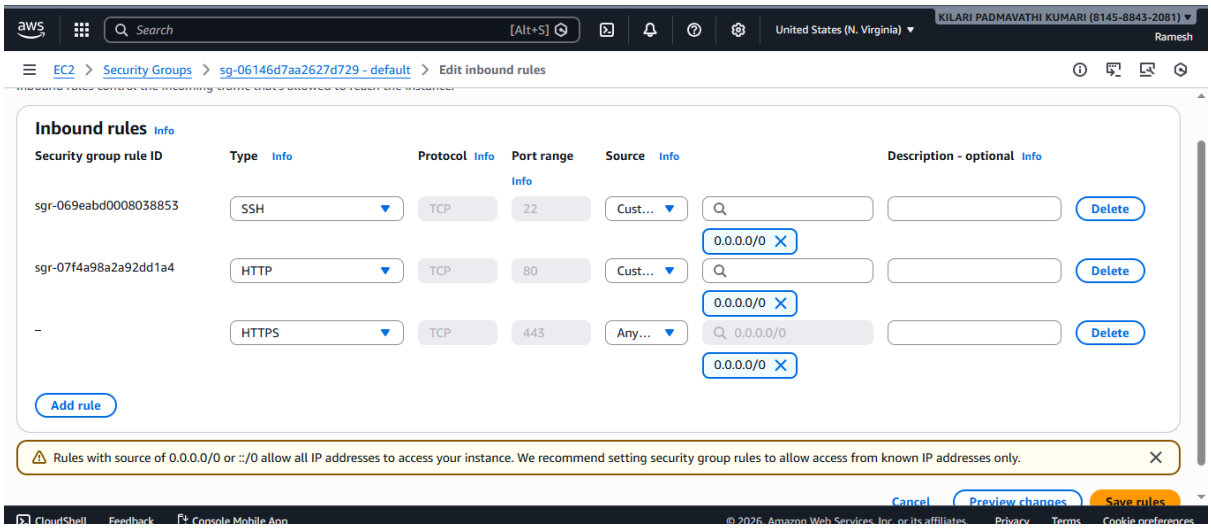
Go to **Listeners → Add listener**

- Protocol: **HTTPS**
- Port: **443**
- Default action: **Forward to your target group**

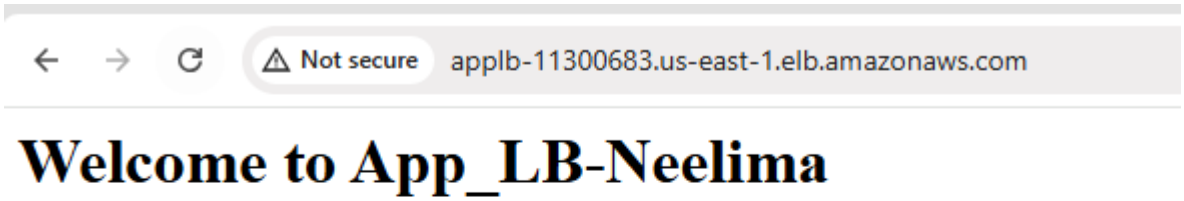


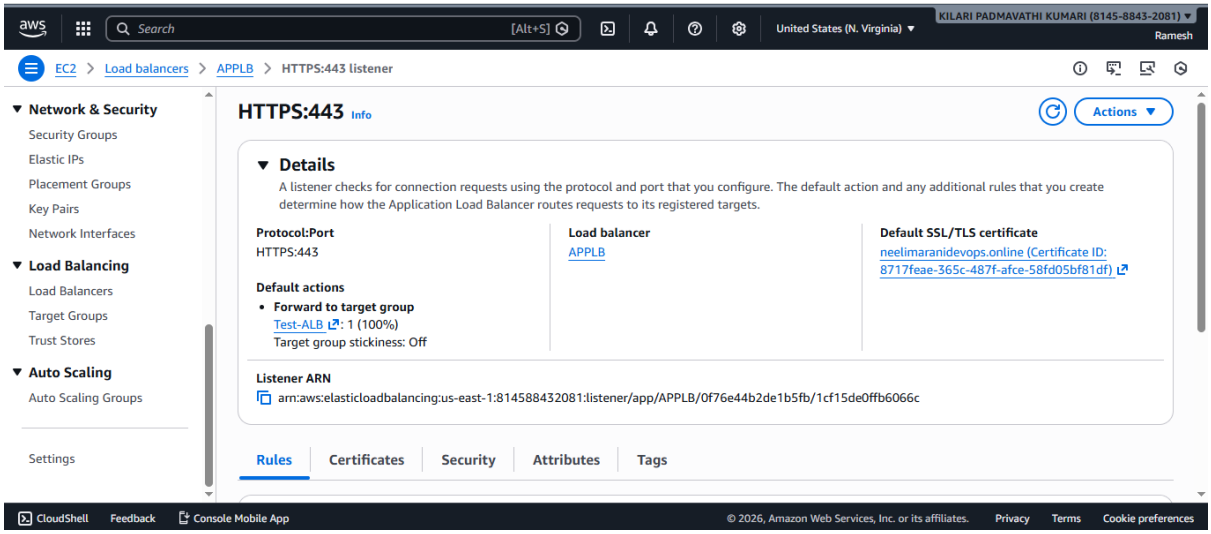
Step 4: Update Security Group

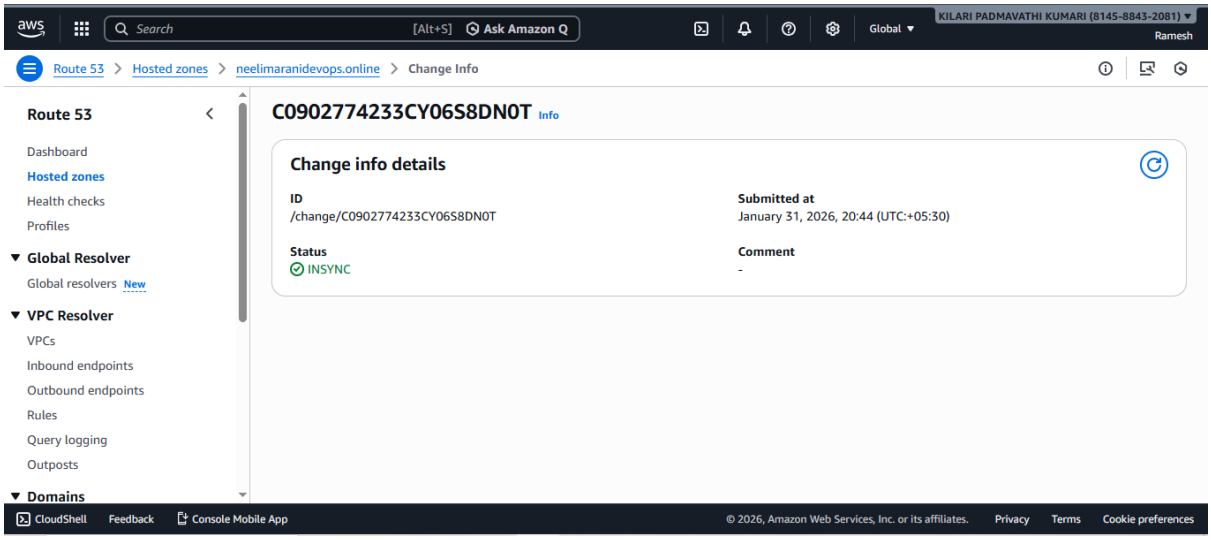
- Allow **Inbound port 443** for **0.0.0.0/0** in ALB security group



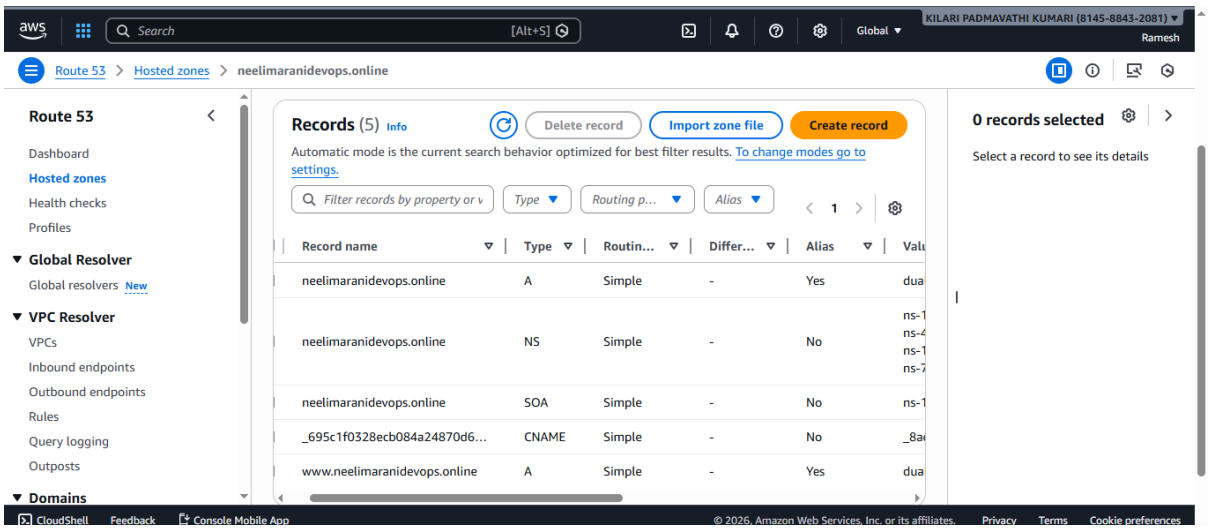
Load Balancers-Task



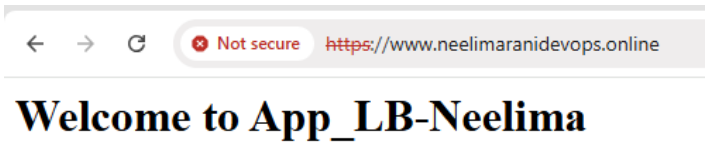




Load Balancers-Task



- Add Records in Route53 as shown above.
- Browser with our domain name :neelimaranidevops.online



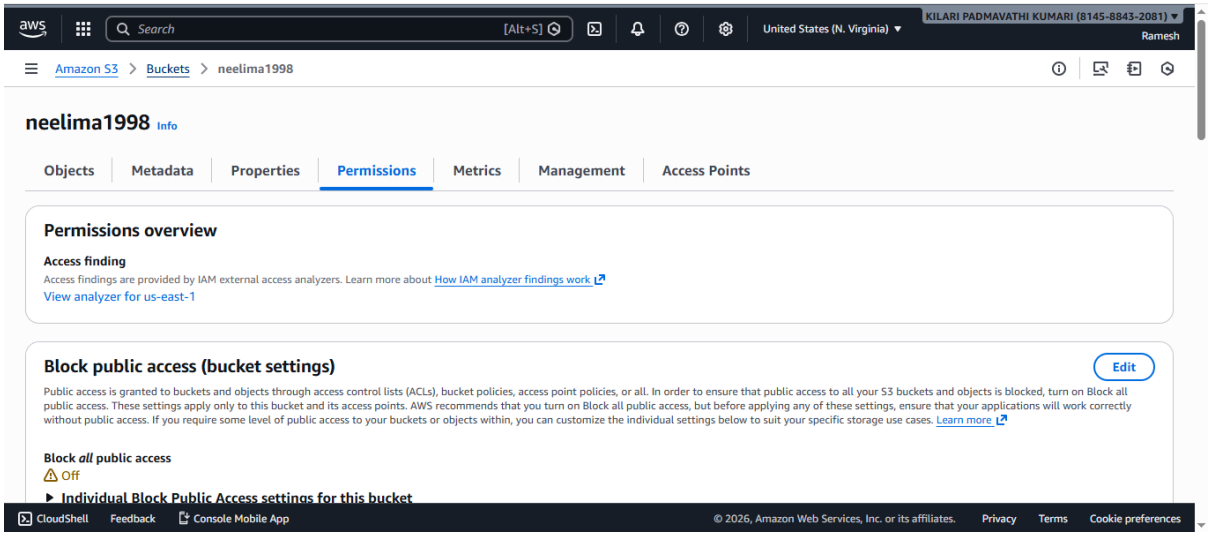
Welcome to App_LB-Neelima

Load Balancers-Task

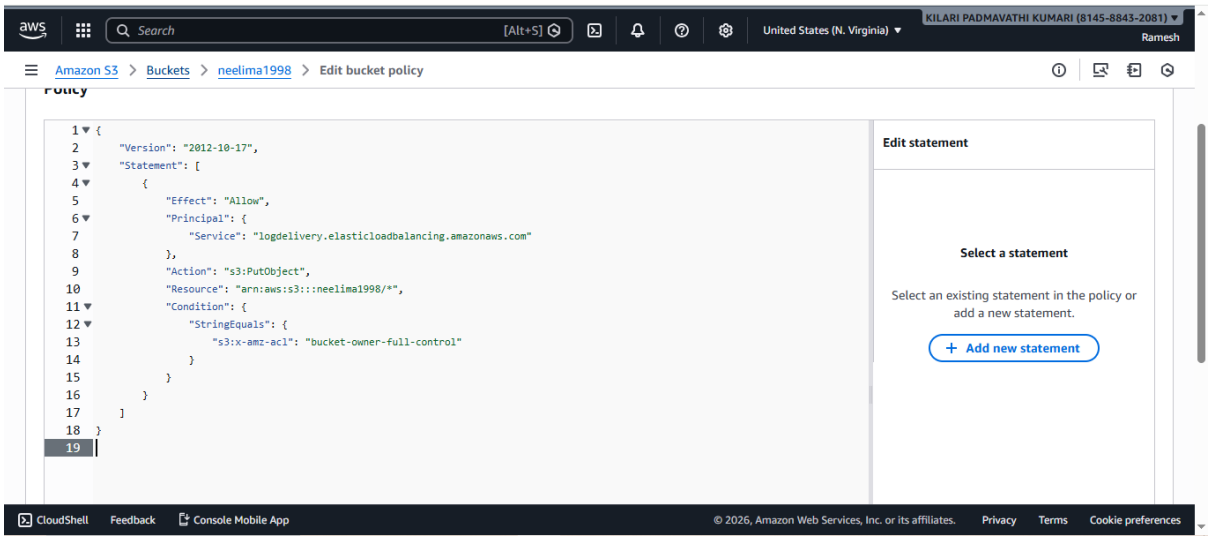
6. Push the application load balancer logs to S3.

step 1: Create an S3 bucket for ALB logs

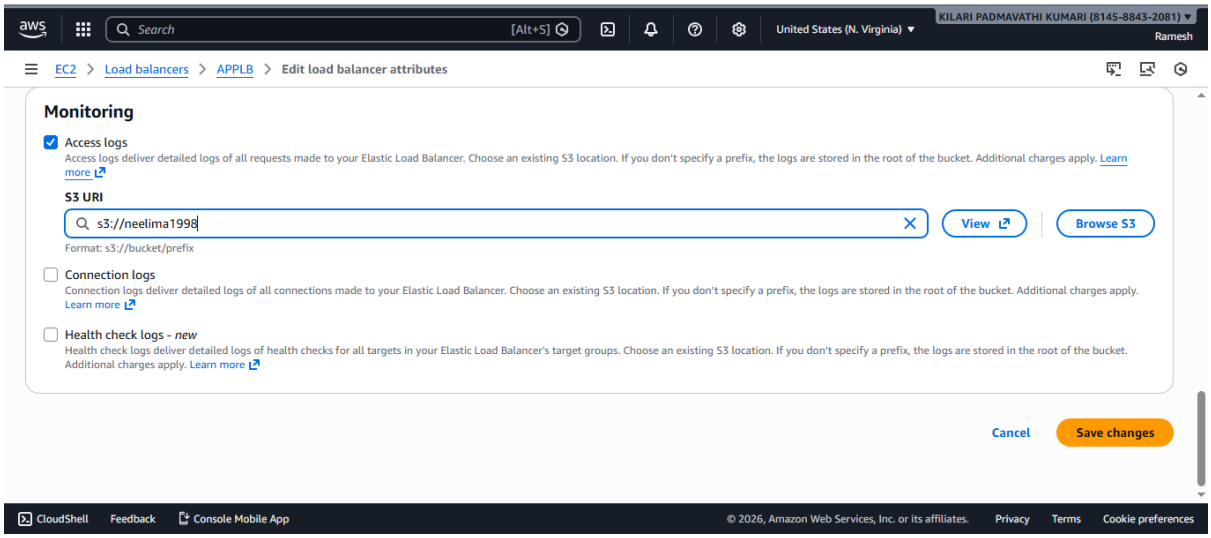
1. Go to AWS S3 in the console.
2. Click Create bucket.
3. Give it a name like: neelima1998



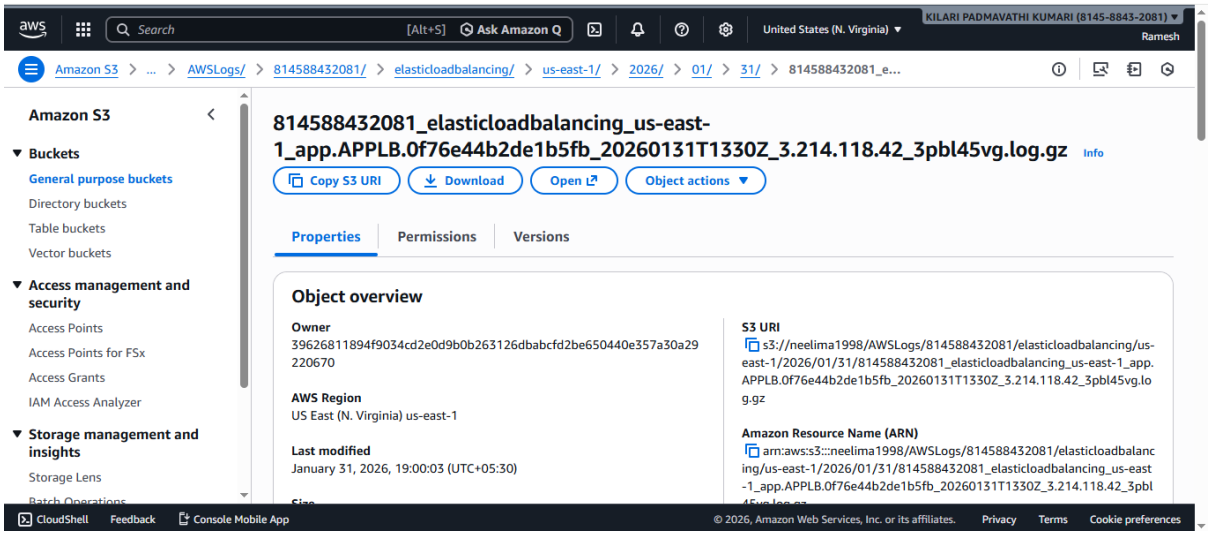
- Go to permission tab
- Click on Bucket policy
- Edit Bucket policy
- Click on save



Load Balancers-Task



- Go to Ec2 → Load Balancers
- Select Application load Balancers
- Edit Load balancers Attributes
- Monitoring →enable Access logs
- Browser s3 bucket where we want to store logs.
- Click on save changes.



- To verify Go to s3 bucket →AWSlogs .