AWS Load Balancer

Overview

AWS Load Balancer is a fully managed service that distributes incoming application traffic across multiple targets, such as EC2 instances, in different Availability Zones. It ensures high availability, fault tolerance, and scalability.

Why Use a Load Balancer?

- Distributes traffic efficiently to prevent overloading a single resource.
- Enhances application availability by routing traffic to healthy instances.
- Provides automatic failover in case of server failure.
- Supports security features like TLS termination and integrated authentication.
- Enables flexible routing and advanced request handling with listener rules.

Connection Between AWS Load Balancer and OSI Model

- **Layer 4 (Transport Layer)**: Network Load Balancer (NLB) operates at this layer, handling TCP/UDP traffic.
- **Layer 7 (Application Layer)**: Application Load Balancer (ALB) operates here, managing HTTP/HTTPS requests.
- **Layer 3 (Network Layer)**: Gateway Load Balancer (GLB) processes network packets between appliances.
- **Layer 2 (Data Link Layer)**: Classic Load Balancer (CLB) provides basic load balancing capabilities.
- Load balancers interact with lower OSI layers via security groups and networking configurations.

Types of Load Balancers

1) Application Load Balancer (ALB)

- Clients send requests to your application.
- Listeners match requests based on protocol and port configuration.
- Incoming requests are evaluated against rules and routed to the appropriate target group.
- HTTPS listeners can offload TLS encryption and decryption.
- Healthy targets receive traffic based on routing rules and load balancing algorithms.

2) Network Load Balancer (NLB)

- Clients send requests to your application.
- The load balancer receives requests directly or via AWS PrivateLink.
- Listeners match protocol and port, routing requests based on the default action.
- TLS listeners offload encryption and decryption.
- Traffic is distributed to healthy targets using the flow hash algorithm.

3) Gateway Load Balancer (GLB)

- Routes requests based on VPC route table, Internet Gateway, or Transit Gateway.
- Directs traffic to a target group of virtual appliances (e.g., firewalls, deep packet inspection systems).
- Virtual appliances process and forward or drop traffic based on configurations.
- Functions as a bump-in-the-wire for inline traffic inspection.

4) Classic Load Balancer (CLB)

- Routes requests based on VPC route table, Internet Gateway, or Transit Gateway.
- Uses a fleet of appliances for traffic filtering and deep packet inspection.
- Processes traffic flows before forwarding or dropping traffic.
- Functions as a basic load balancer for legacy applications.

Load Balancer Schemes

- **Internet Facing**: Exposes a public IP to route traffic from the internet.
- **Internal**: Routes traffic only within a private VPC, restricting external access.

Target Group in Load Balancers

A Target Group defines a group of registered targets (such as EC2 instances) that receive traffic from the load balancer. It supports different protocols and health checks to determine target availability.

Project :- AWS Load Balancer Path-Based Routing Project For EC2 Hosted Web Applications.

Step 1: Create Two EC2 Instances in Different Availability Zones

We are launching two EC2 instances in the same AWS region but in separate Availability Zones (AZs). This setup improves high availability and fault tolerance.

Instance 1: "Test EC2 1a"

- **AMI Selection**: Choose an appropriate Amazon Machine Image (AMI) based on the required operating system.
- **Key Pair (KP)**: Select a key pair for secure SSH access.
- **Networking**:
    - Place the instance in a specific subnet belonging to **Availability Zone ap-south-1a** (e.g., `subnet-0603aa0dd1018f970`).
    - Use the **security group** (`ec2sg`) to allow necessary inbound/outbound traffic.

- **User Data**: The following script executes automatically upon instance launch:

```
#!/bin/bash
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
echo "Hello World from $(hostname -f)" > /var/www/html/index
```

- **Launch the instance** named **"Test EC2 1a"**.

Instance 2: "Test EC2 1b"

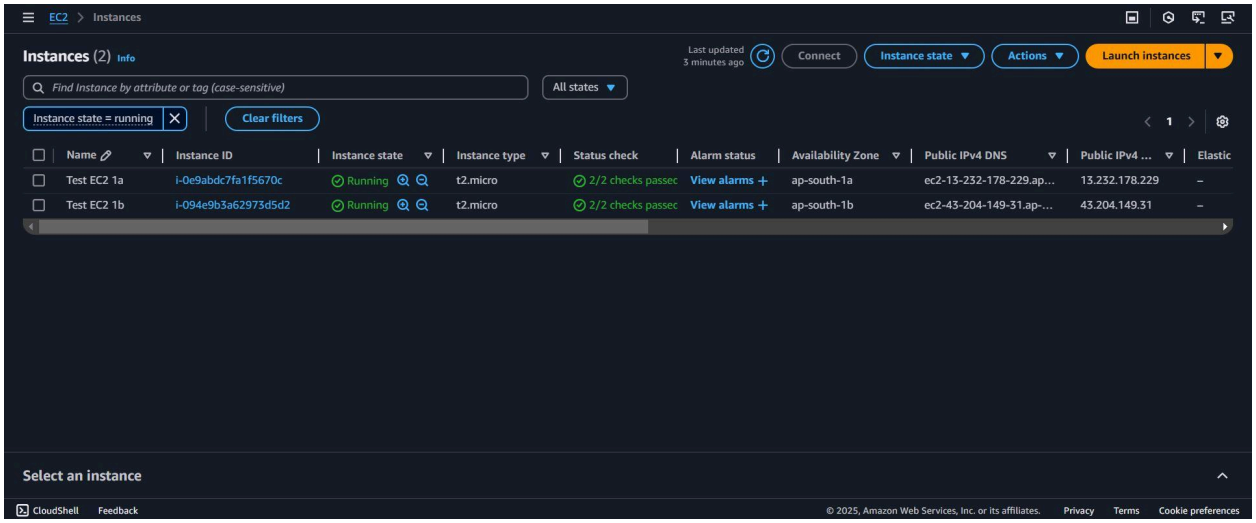
- **Follow the same steps as "Test EC2 1a"** but change:
    - **Subnet**: `subnet-006d8c52d39e28865` (Availability Zone: `ap-south-1b`).
- **Launch the instance** named **"Test EC2 1b"**.

Step 2: Launch the Instance Called "ap-south-1b"

Follow the same procedure while ensuring the instance is created in the correct **subnet and AZ**.

Step 3: Verify Instances Are Running

1. Navigate to the **EC2 Dashboard** in the AWS Management Console.
2. Check the status of the instances to ensure they are in a "running" state.



Step 4: Test Web Server Accessibility

1. Copy the **public IP address** of each instance.
2. Open a web browser and paste the IP address into the address bar.

3.You should see a webpage displaying:

Hello World from <hostname>

Hello World from Mumbai 1a ip-172-31-43-97.ap-south-1.compute.internal

Hello World from Mumbai 1b ip-172-31-9-249.ap-south-1.compute.internal

Step 2: Create an Application Load Balancer

1. Navigate to **Load Balancers** in the **EC2 Dashboard**.
2. Click **Create Load Balancer** and select **Application Load Balancer**.
3. Configure the Load Balancer:
    - **Name**: `DemoALB`
    - **Scheme**: `Internet Facing`
    - **IP Address Type**: `IPv4`
    - **Network Mapping**: Keep **VPC Default** and select **all Availability Zones and subnets**.
    - **Security Group**: Select `launch-wizard-1` (we will configure it later).

Listeners and Routing

Listeners: A listener checks for connection requests using the specified protocol and port. Here, we set:

- **Protocol**: HTTP
- **Port**: 80 (A port is a communication endpoint that allows data transfer between devices.)
- **Default Action**: Forward requests to a **Target Group**.

What is a Target Group in Load Balancers?

A **Target Group** is a logical group of registered targets (EC2 instances) that receive traffic from the Load Balancer. It defines how traffic is directed based on routing rules.

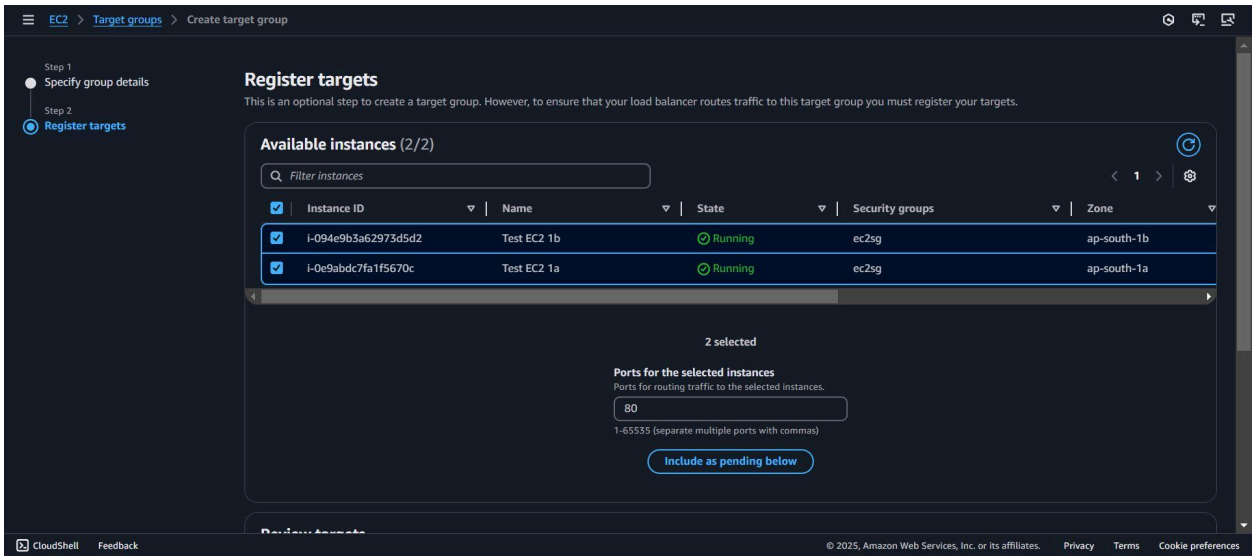
1. Click **Create Target Group**.
2. **Target Group Configuration**:
    - **Target Type**: `Instances`
    - **Name**: `TG1`
    - **Protocol**: HTTP, **Port**: 80
    - **VPC**: Select the appropriate VPC
    - **Protocol Version**: HTTP1

What are Health Checks?

Health checks monitor the status of registered targets to ensure traffic is only sent to healthy instances. It includes:

- **Health Check Protocol**: HTTP
- **Health Check Path**: `/`

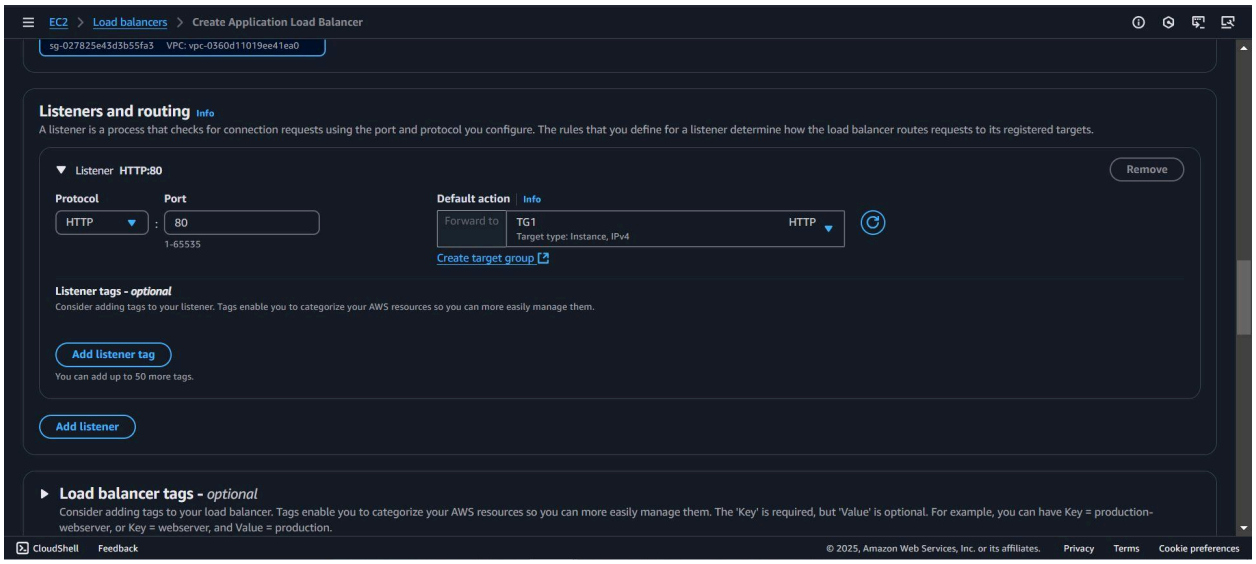
3. Click **Next**.
4. **Register Targets**:
    - Select the two previously created EC2 instances.
    - **Ports for the selected instances**: These define the communication channel for routing traffic to the instances.
    - Click **Include as pending below**.



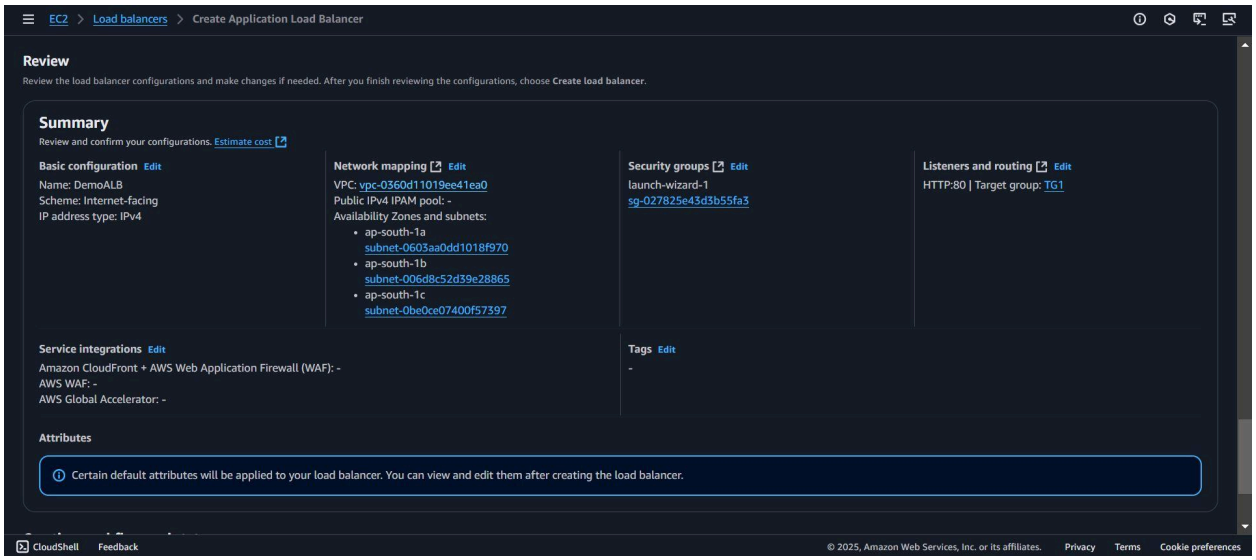
5. Click **Create Target Group**.

tep 3: Assign Target Group to Load Balancer

1. Go to **Listeners and Routing** in the Load Balancer settings.
2. Refresh the page and **add TG1 as the target group for HTTP**.



3. Review the **Summary** to check all Load Balancer configurations.



4. Click **Create Load Balancer**.

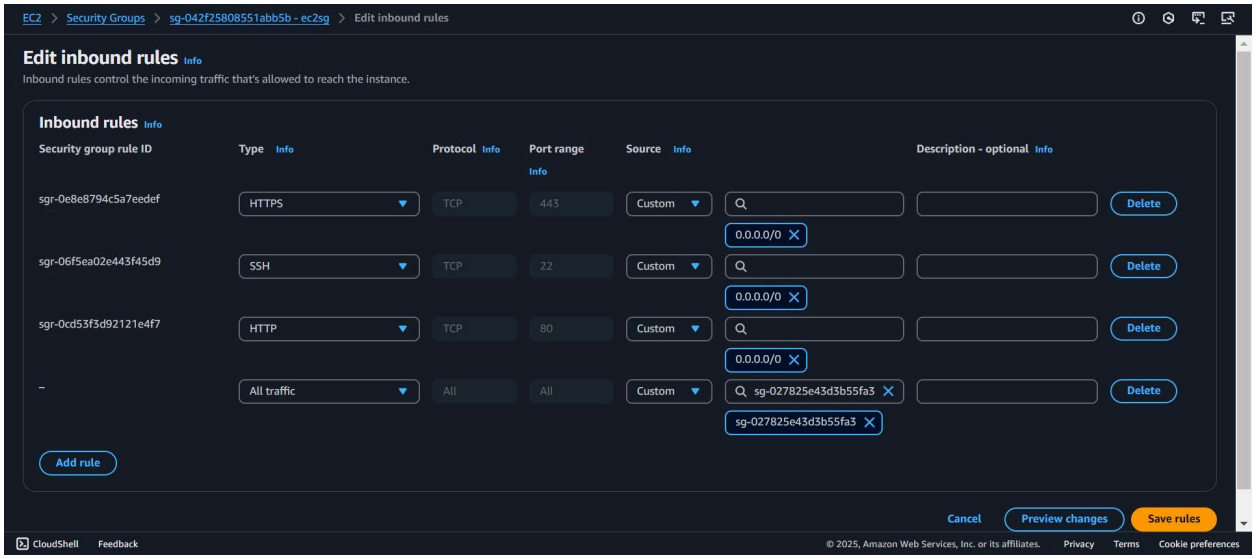
Step 4: Security Group Configuration

Now we have two **Security Groups (SGs)**:

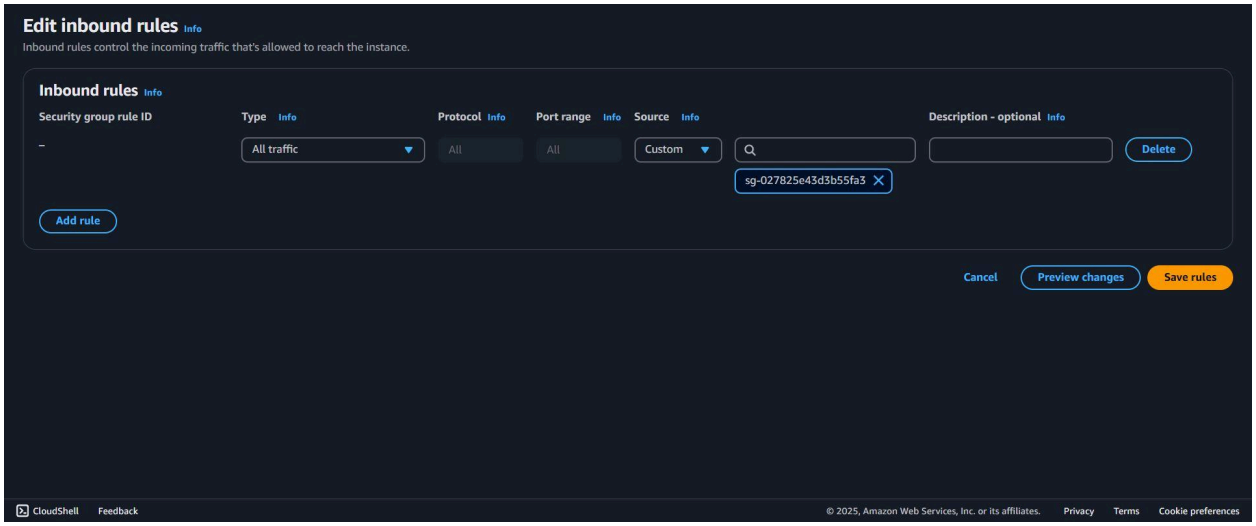
1. **Load Balancer Security Group**: Controls incoming traffic to the Load Balancer.
2. **Instance Security Group**: Controls traffic to EC2 instances.

To ensure secure communication, configure the **instance's security group** to allow traffic only from the **Load Balancer's security group**:

1. Navigate to **EC2 Dashboard**.
2. Select **Security Groups** and choose `ec2sg`.
3. Click **Edit Inbound Rules**.
4. **Modify the inbound rules**:
    - **Add Rule**:
        - **Type**: `All Traffic`
        - **Source**: `Custom`
        - **Select Security Group of Load Balancer (`launch-wizard-1`)**`.
        -



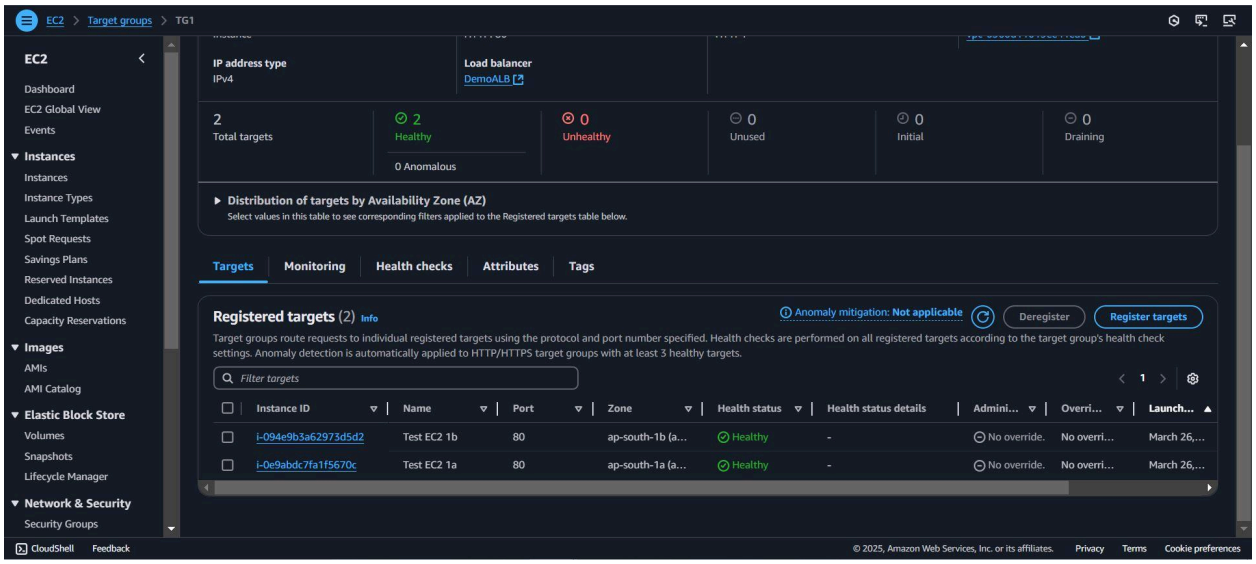
5. **Delete previous three inbound rules** (Reason: Removing public access ensures that instances only receive traffic from the Load Balancer, enhancing security).



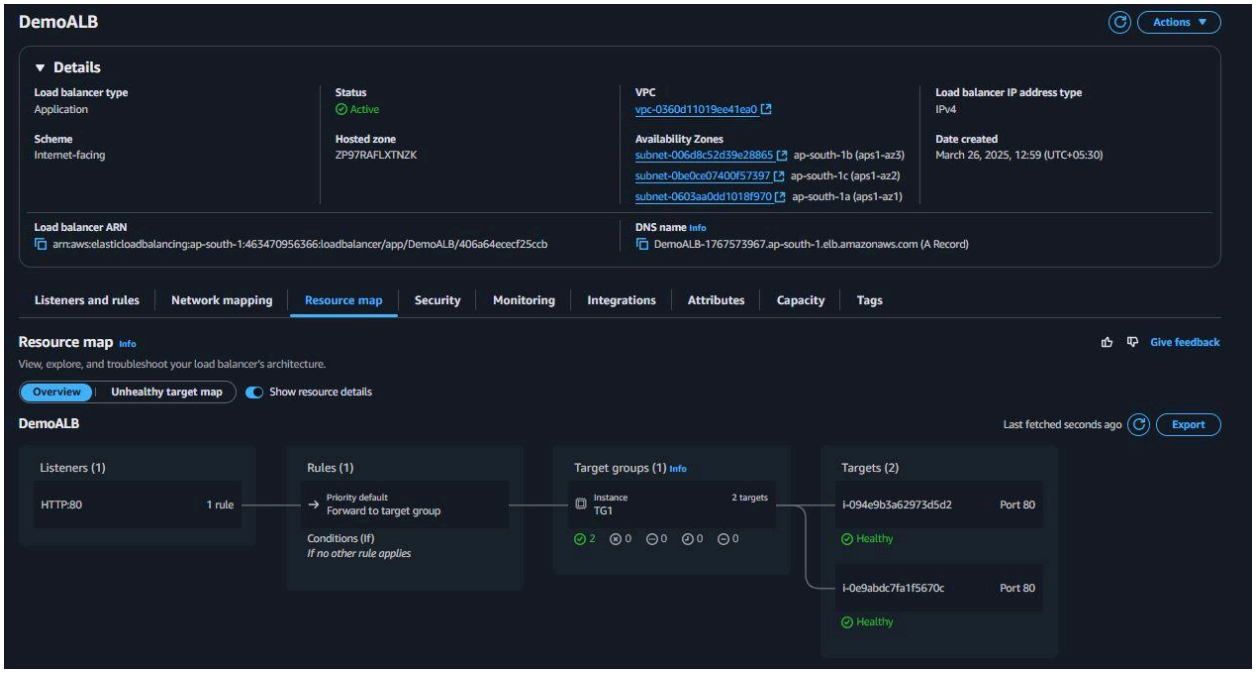
6. Click **Save Rules**.

Step 5: Verify Target Group and Load Balancer Status

1. **Check Target Group Details**:
   - Navigate to **Target Groups**.
   - Verify that both instances appear as **Healthy**.



2. **Check Load Balancer Status**:
    - Navigate to **Load Balancers**.
    - Verify that the `DemoALB` status is `Active` or `Provisioning`.



Step 6: Verify Load Balancer Traffic Distribution

1. **Copy the "DNS Name" of the Load Balancer**.
2. Open a browser and paste the **DNS Name** to check the webpage.

Hello World from Mumbai 1a ip-172-31-43-97.ap-south-1.compute.internal

3. **Refresh the page multiple times** to see the traffic distribution between the two instances.

Now, you can see that the traffic is being **distributed equally** between the two EC2 instances. This demonstrates **high availability**, a key feature of AWS Load Balancer.

To allow users to access the application, simply share the **DNS Name of `DemoALB`**.

Currently, the Load Balancer has only one **Target Group** (`TG1`) with the following **Listener Rule**:

- **Forward to Target Group**
  - `TG1` (100% traffic distribution)
  - **Target group stickiness**: Off

Understanding Listener Rule Configuration

Currently, the Load Balancer has only one **Target Group** (`TG1`) with the following **Listener Rule**:

- **Forward to Target Group**
  - `TG1` (100% traffic distribution)
  - **Target group stickiness**: Off

What does this mean?

- **"Forward to Target Group TG1: 1 (100%)"** → All incoming traffic is directed to the `TG1` target group.
- **"Target group stickiness: Off"** → The Load Balancer does not persistently direct requests from the same client to the same instance; instead, it distributes traffic evenly across all healthy targets.

This setup ensures efficient load balancing and redundancy within the AWS region.

Step 7 :Advanced Use Cases for Load Balancer

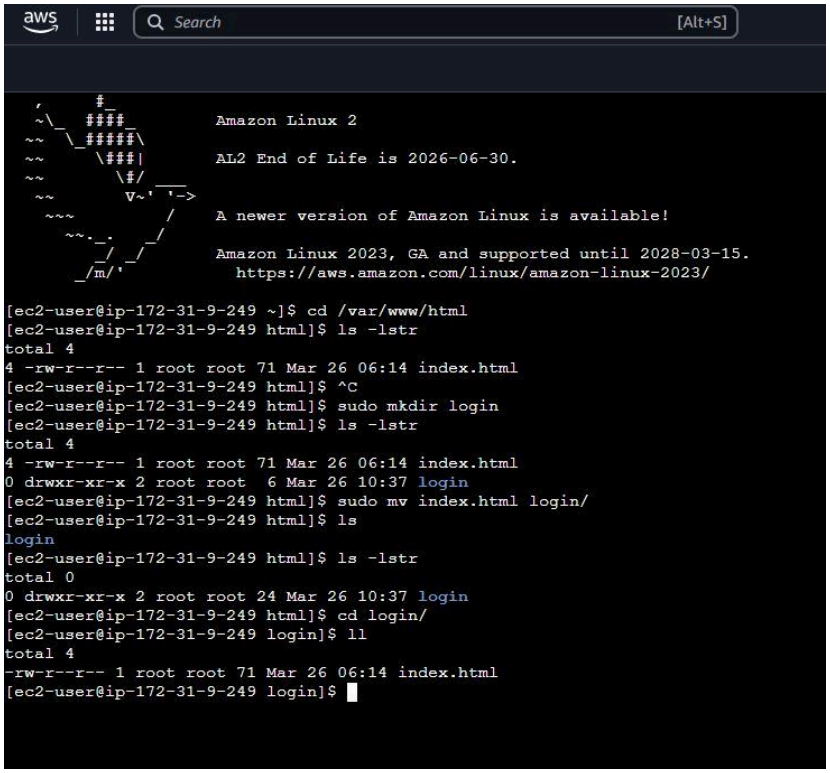
Modify the Path of `index.html`

1. Go to **EC2 Dashboard**.
2. Select **EC2 Instance "Test EC2 1b"**.
3. Click **Connect** to access the instance (Ensure SSH access is enabled in the Security Group, as it was previously removed).

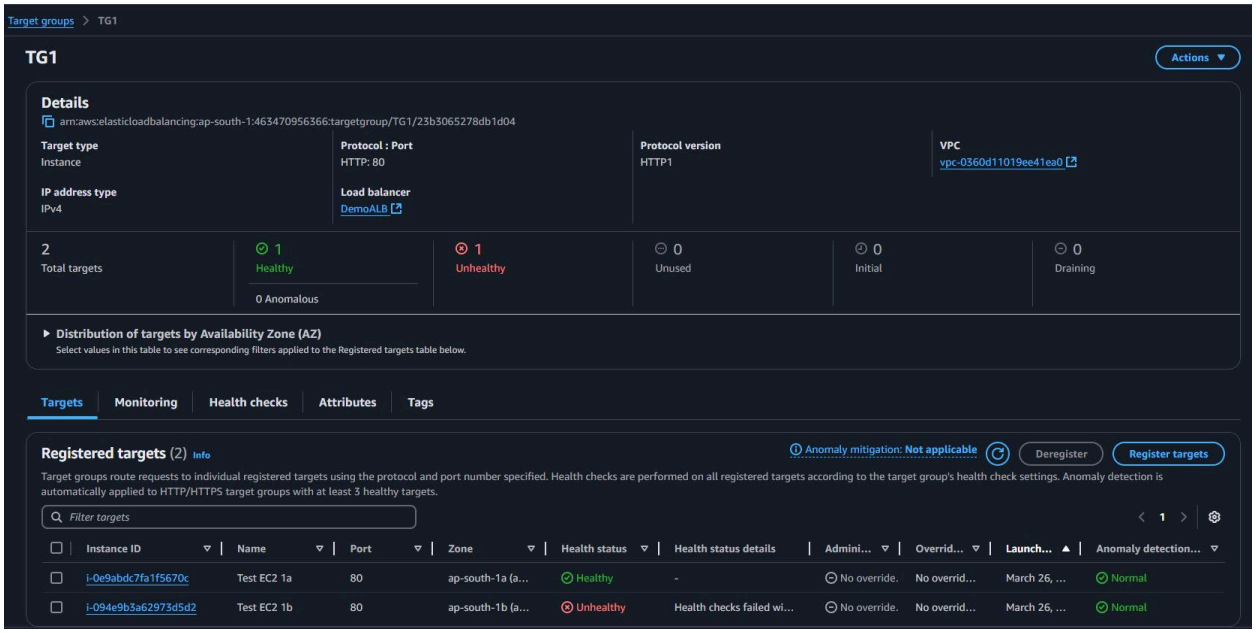
4. Run the following commands:

```
cd /var/www/html
ls -lstr
sudo mkdir login
sudo mv index.html login/
```

This changes the path of ***`index.html`*** *on "Test EC2 1b".*

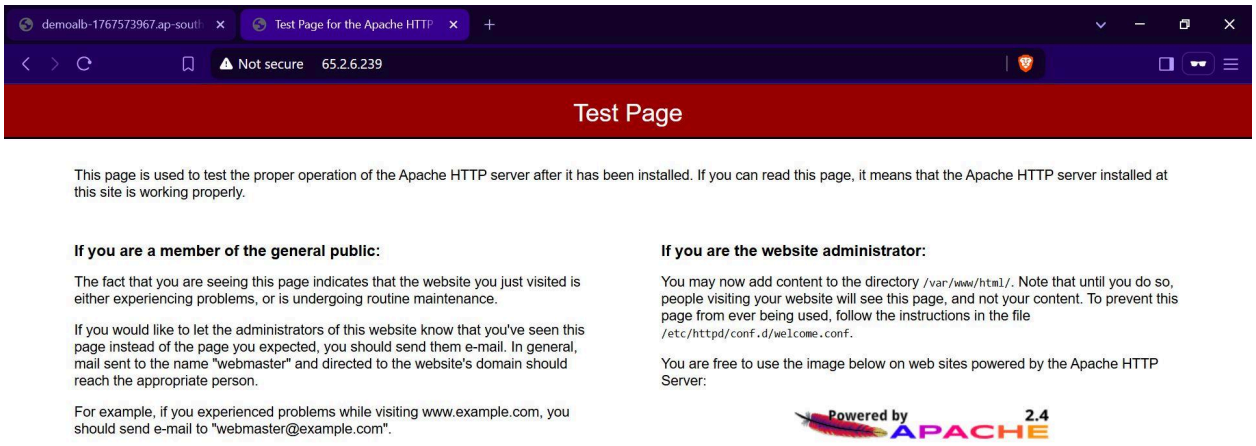


1. Navigate to the **Target Group Dashboard**.
2. Locate **Target Group "TG1"**.
3. Notice that the health check will fail for **EC2 Instance "Test EC2 1b"** because the default `/index.html` path no longer exists at the expected location.



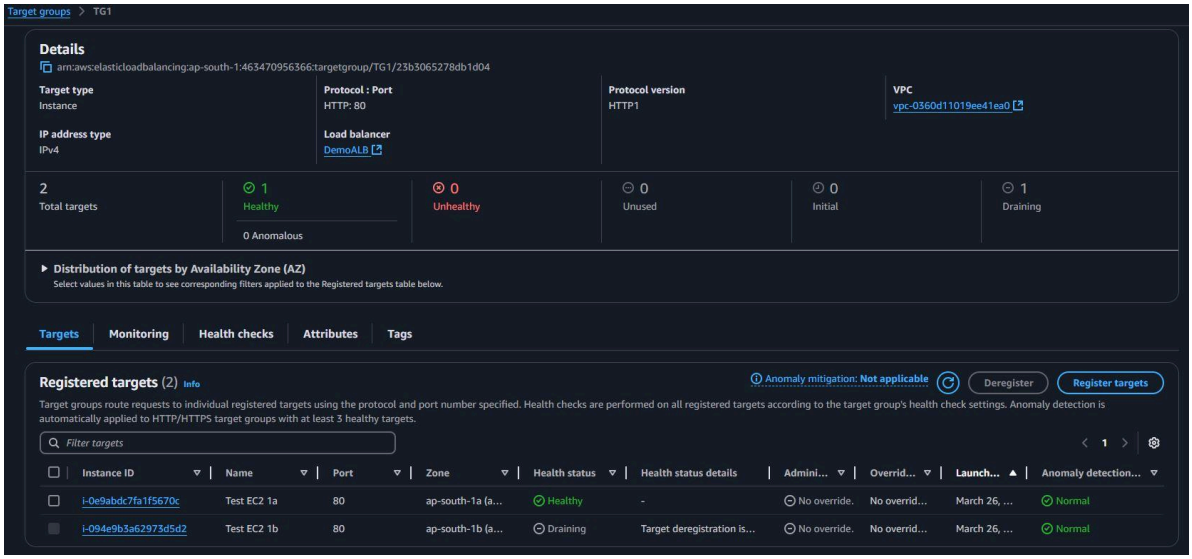
Load Balancer Path Awareness Issue

1. You can see in the below image that the **Load Balancer is unaware** that we changed the path of our web application (`index.html`) to `/var/www/html/login/index.html` for EC2 "Test EC2 1b".
2. However, it still redirects requests to the old path `/var/www/html`.



Fix Load Balancer Path Routing

1. We need to add **intelligence** to the Load Balancer so that when the web application is accessed from the new path, it correctly routes to the appropriate EC2 instance.
2. Navigate to **TG1 Target Group**.
3. **Deregister** the unhealthy EC2 instance **"Test EC2 1b"** from **TG1**.



Draining in AWS Load Balancer (Short Explanation)

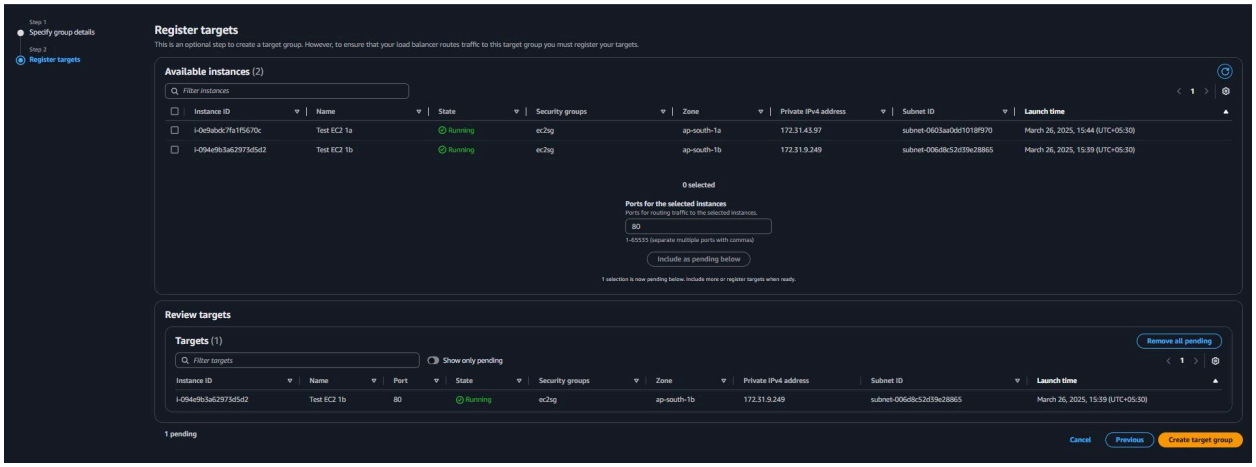
Connection draining (or deregistration delay) ensures that active connections to an instance are **allowed to complete** before the instance is **removed** from the load balancer. This prevents ongoing requests from being abruptly terminated.

Step 8 : Create a New Target Group

1. Go to **Target Groups** in the AWS Console.
2. Click **Create Target Group**.
3. Set the **Name** as **"loginTG"**.
4. Keep **IP address type** as **IPv4**.
5. Select **VPC = HTTP1**.
6. Configure **Health Checks**:
    - The associated load balancer periodically sends requests to the registered targets to test their status.
    - **Health check protocol**: **HTTP**.
    - Use the default path `/` to perform health checks on the root, or specify a custom path.
    - Set the new health check path as **`/login/`**.
7. Click **Next**.

Register EC2 Instance to New Target Group

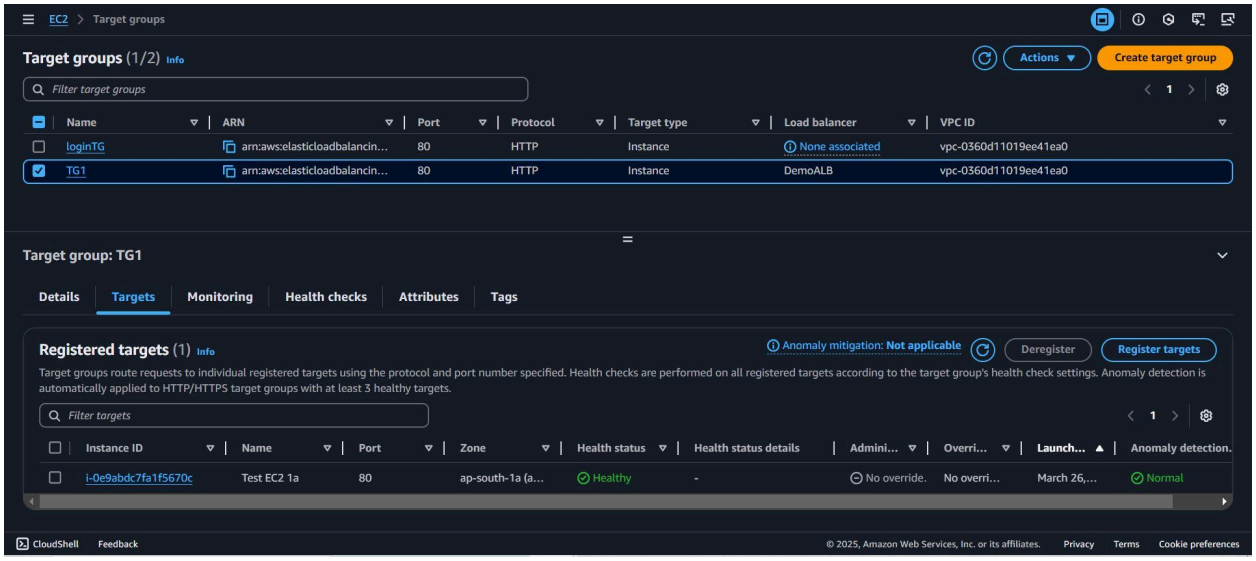
1. Include EC2 **"Test EC2 1b"** under **Include as pending below**.
2. Click **Create Target Group**.

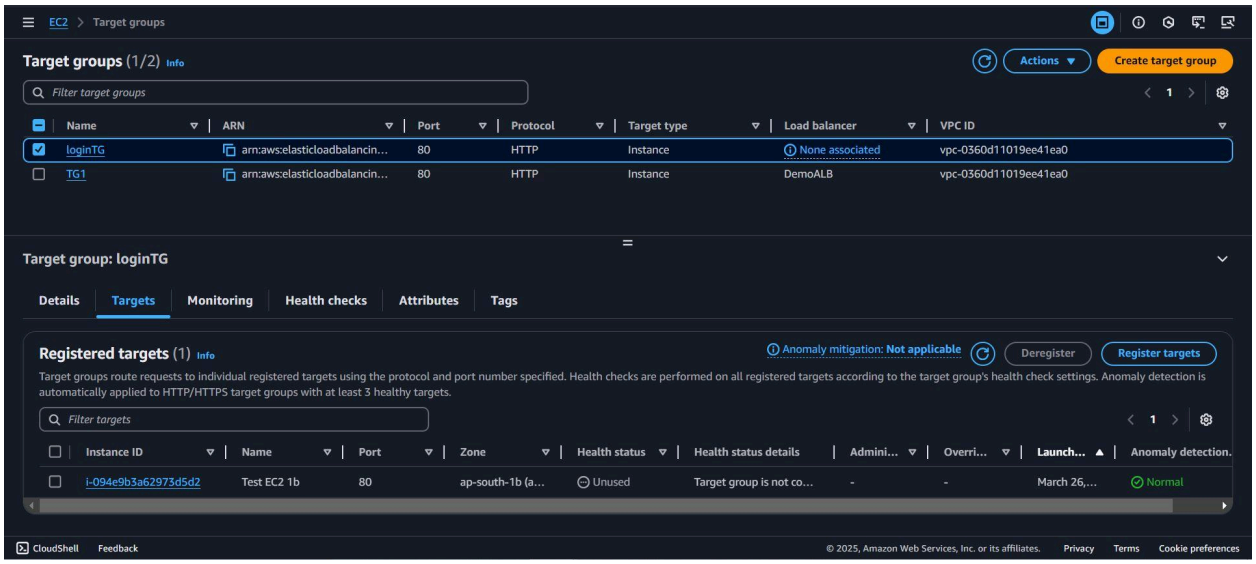


Target Group Overview

Now, you have the following Target Groups:

1. **TG1** – Includes **EC2 "Test EC2 1a"**.
2. **loginTG** – Includes **EC2 "Test EC2 1b"**.

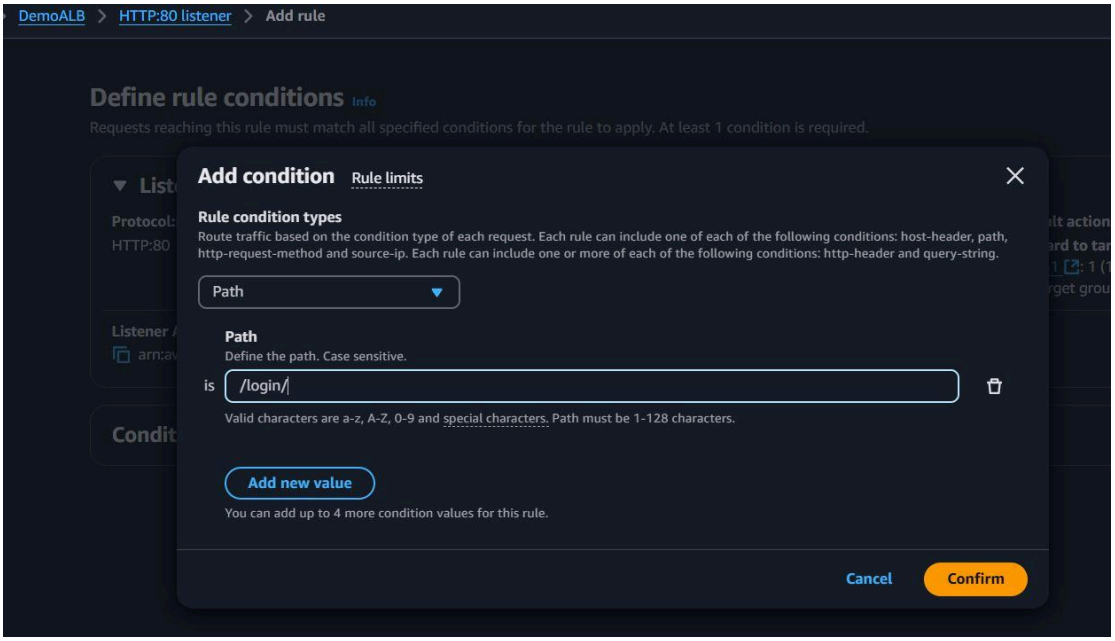




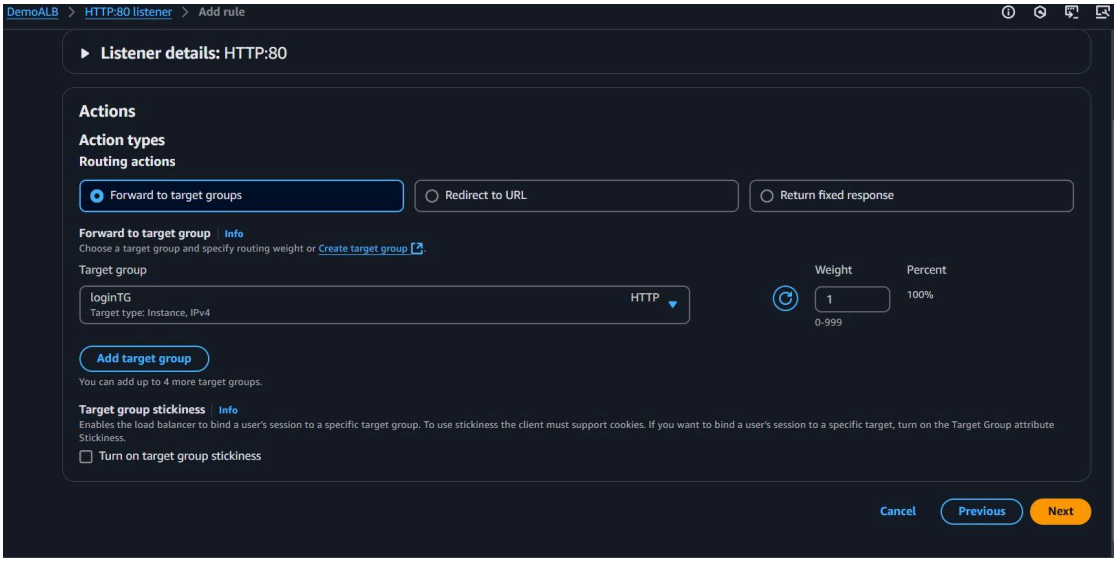
Step 9: Creating a New Listener Rule for Load Balancer

In this step, we will create a new listener rule for our Load Balancer to handle traffic for the path "/login/".

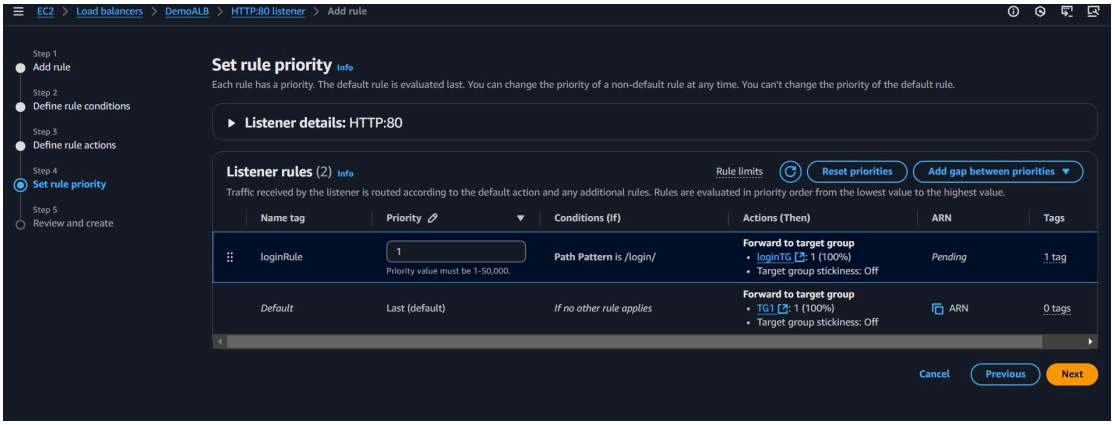
1. **Navigate to Listener Settings:**
    - Go to the Listener section of your Load Balancer.
    - Click on Add Rule.
2. **Add a Name to the Rule:**
    - Provide a meaningful name for the rule.
3. **Define Rule Conditions:**
    - Click on Add Condition.
    - Select "Path" as the condition type.
    - Click Add Path and enter the desired path "/login/".



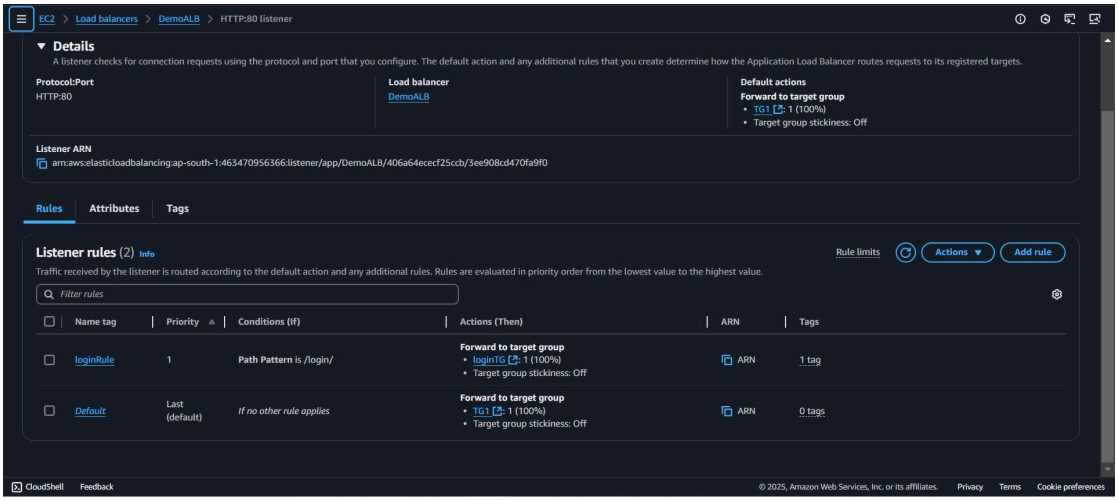
    - Click Next.
4. **Define Rule Actions:**
    - Under Actions, choose Forward to Target Group.
    - Select the Target Group as "loginTG".
    - Click Next.



5. **Set Rule Priority:**
    - Specify the rule priority. In this case, set it to 1.
    - Click Next.



6. **Review and Create:**
    - Verify all configurations.
    - Click Create Rule.



Now, the listener will have two rules, and they will be executed in a top-to-bottom order based on priority.

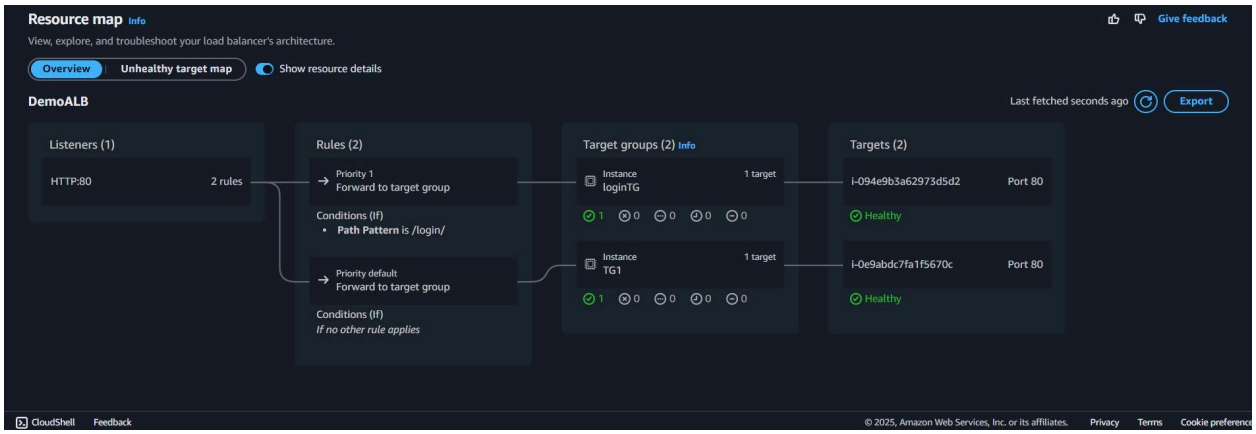
Step 10 :- Verifying the Load Balancer Routing

1. **Test Default Load Balancer Behavior:**
    - Hit the URL of the Load Balancer "DemoALB".
    - You should see the load balancer redirecting traffic to EC2 "Test EC2 1a".

2. **Test Path-Based Routing:**
    - Hit the URL of the Load Balancer "DemoALB" and append "/login/" at the end.
    - You should see the load balancer redirecting traffic to EC2 "Test EC2 1b".

Hello World from Mumbai 1b ip-172-31-9-249.ap-south-1.compute.internal

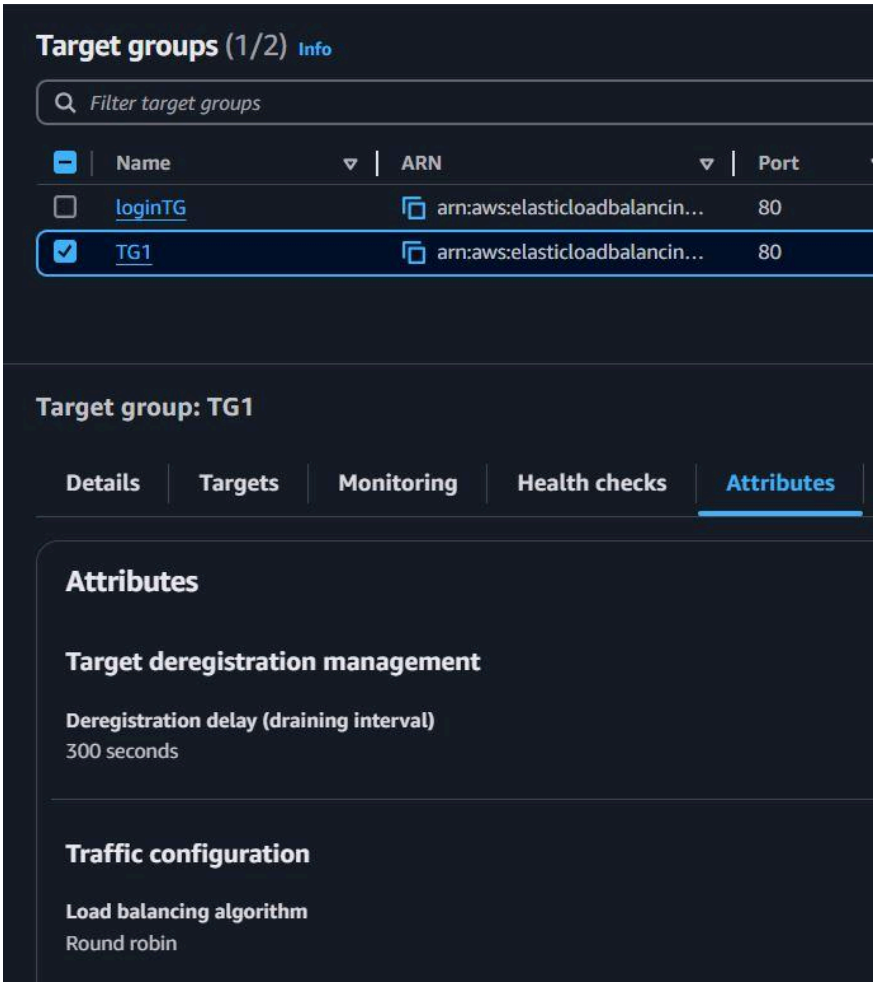
3. **Resource Mapping for Clear Understanding:**
    - To get a better understanding, you can check the Resource Map for the "DemoALB" Load Balancer.



Understanding the Round Robin Algorithm in AWS ALB:

The Application Load Balancer (ALB) in AWS uses the Round Robin Algorithm to distribute incoming traffic across multiple registered targets within a target group.

- **How It Works: Each new request is sent to the next available target in a circular order, ensuring even distribution of traffic.**
- **Example in ALB: If there are two EC2 instances (Test EC2 1a and Test EC2 1b) in the target group, the first request will be routed to Test EC2 1a, the second to Test EC2 1b, the third back to Test EC2 1a, and so on.**
- **Load Balancing Benefits: This approach ensures that no single instance is overwhelmed while others remain idle, providing efficient resource utilization and improved application availability.**


Target groups (1/2) Info
Filter target groups
Name
ARN
Port
loginTG
arn:aws:elasticloadbalancin...
80
TG1
arn:aws:elasticloadbalancin...
80
Target group: TG1
Details
Targets
Monitoring
Health checks
Attributes
Attributes
Target deregistration management
Deregistration delay (draining interval)
300 seconds
Traffic configuration
Load balancing algorithm
Round robin