# Hands-On with AWS Load Balancer: EC2 Setup, Target Groups & Health Checks

## Introduction

A **Load Balancer (LB)** in AWS helps distribute incoming traffic across multiple EC2 instances. This ensures:

- No single instance is overloaded
- Higher **availability**
- Improved **fault tolerance**

## How Load Balancer Works

Load Balancer automatically routes client requests to healthy targets (EC2s) based on various **strategies** like:

- **Round Robin**
- **Least Outstanding Requests**

LB performs **health checks** on targets and only routes traffic to **healthy** ones. It listens on specific ports (e.g., 80 or 443) and routes the requests to a **Target Group** of EC2 instances.

## Types of Load Balancers in AWS

1. Application Load Balancer (ALB)
2. Network Load Balancer (NLB)
3. Classic Load Balancer (CLB)
4. Gateway Load Balancer (GWLB)

| Type | Layer | Protocols | Use Case |
|---|---|---|---|
| **ALB** | Layer 7 | HTTP/HTTPS | Application routing, content-based |
| **NLB** | Layer 4 | TCP/UDP | High performance, low latency |
| **CLB** | Both 4 & 7 | HTTP, TCP | Legacy systems |
| **GWLB** | Layer 3/4 | Custom flows | Used with firewalls, inspection tools |

## TCP vs UDP

- **TCP**: Reliable, ordered, but slower (e.g., Email)
- **UDP**: Faster, connectionless, some data loss possible (e.g., video/voice calls)

## OSI Model :

The Open Systems Interconnection (OSI) model defines a framework for network communication. It consists of 7 layers:

Layer 7 to Layer 1:
Application → Presentation → Session → Transport → Network → Data Link → Physical

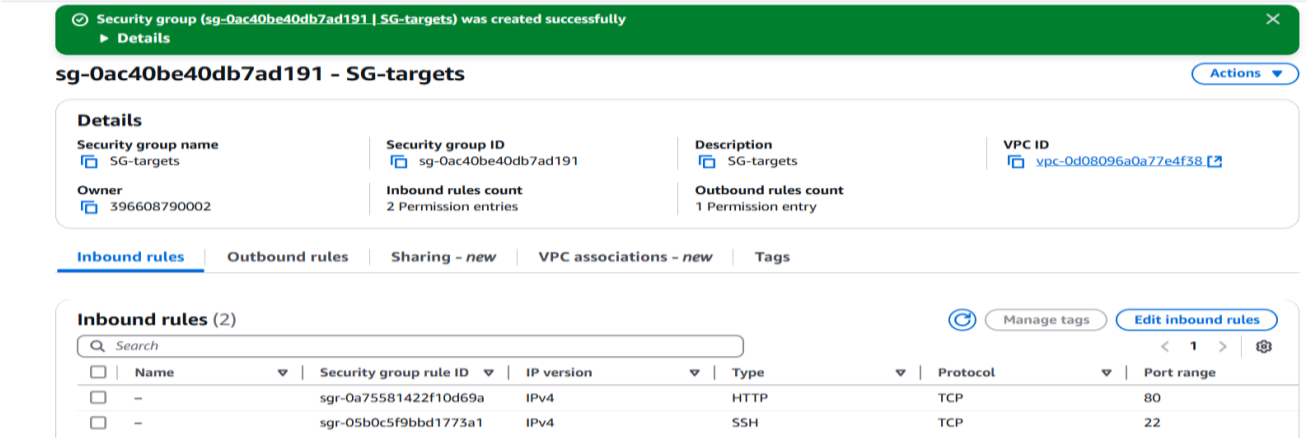
LB primarily works at **Layer 4 (NLB)** and **Layer 7 (ALB)**.

## Step-by-Step Practical: Load Balancer Setup in AWS

### 1. Create a Security Group

Go to EC2 > Security Groups

- Create a new SG allowing:
    - Inbound: TCP 22, TCP 80
    - Outbound: All traffic



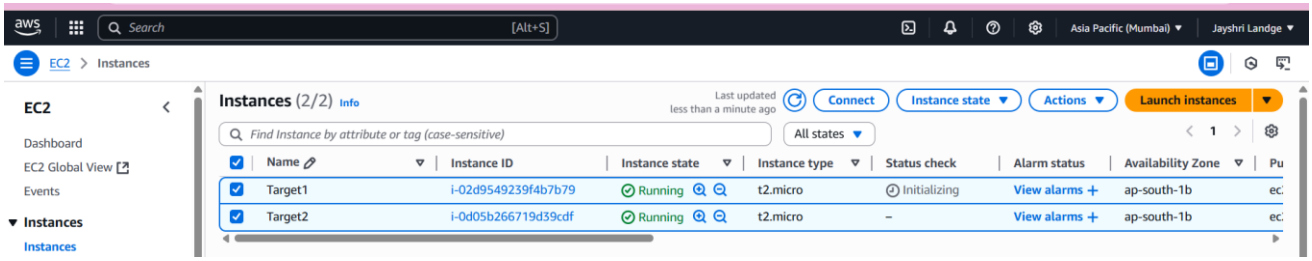

**Screenshot 1: Security Group allowing ports 22 (SSH) and 80 (HTTP)**

### 2. Launch Two Web Servers

- Launch 2 EC2 instances using Amazon Linux or Ubuntu
- SSH into each instance
- Install Apache or Nginx:
- sudo yum install httpd -y
- sudo systemctl start httpd
- sudo systemctl enable httpd

- Modify index.html with unique content:
- echo "This is Web Server 1" > /var/www/html/index.html

And similarly for Server 2.




***Screenshot 2: Running EC2 Instances***

```
[root@ip-172-31-15-32 ~]# cat /var/www/html/server1/index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Server 1 – DevOps Projects</title>
  <style>
    body {
      margin: 0;
      font-family: 'Segoe UI', sans-serif;
      color: #ffffff;
      text-align: center;
      height: 100vh;
      /* Purple to sky blue gradient background */
      background: linear-gradient(135deg, #6a11cb, #2575fc);
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
    }
    h1 {
      font-size: 3em;
      text-shadow: 2px 2px 10px #000000aa;
      margin-bottom: 0.3em;
    }
    p {
      font-size: 1.2em;
      text-shadow: 1px 1px 5px #00000099;
      margin-top: 0;
    }
  </style>
</head>
```

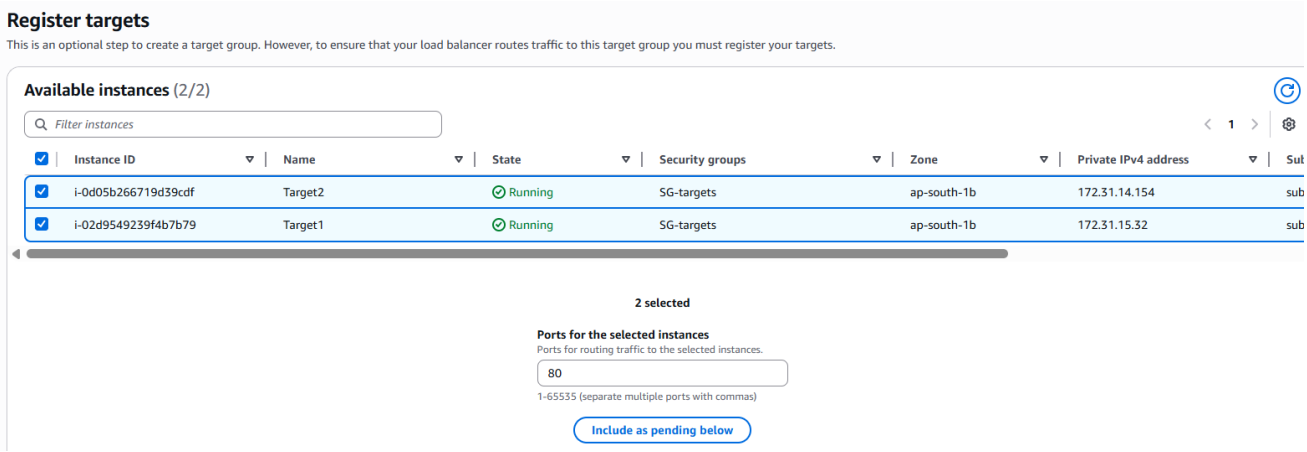
**Screenshot 3: index.html content for Web Server 1**

```
[root@ip-172-31-15-32 ~]# cat > /var/www/html/server2/index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Server 2 – DevOps Projects</title>
  <style>
    body {
      margin: 0;
      font-family: 'Segoe UI', sans-serif;
      color: #ffffff;
      text-align: center;
      height: 100vh;
      /* Purple to sky blue gradient background */
      background: linear-gradient(135deg, #6a11cb, #2575fc);
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
    }
    h1 {
      font-size: 3em;
      text-shadow: 2px 2px 10px #000000aa;
      margin-bottom: 0.3em;
    }
    p {
      font-size: 1.2em;
      text-shadow: 1px 1px 5px #00000099;
      margin-top: 0;
    }
  </style>
</head>
```

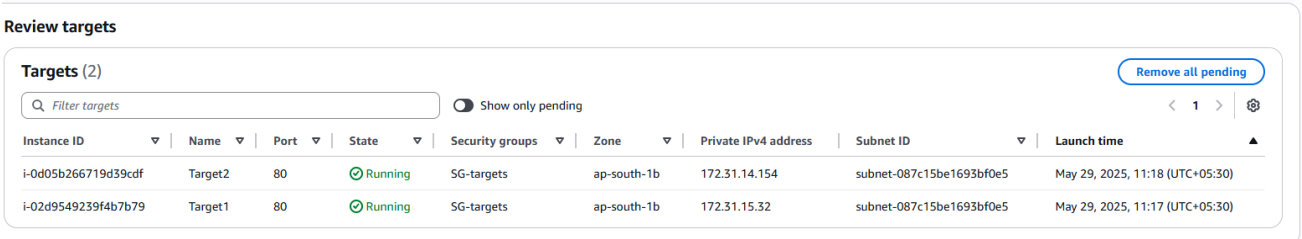
**Screenshot 4: index.html content for Web Server 2**

## 3. Create a Target Group

- Navigate to EC2 > Target Groups
- Create a new target group (choose instance-based, protocol: HTTP)
- Add both EC2 instances as **targets**



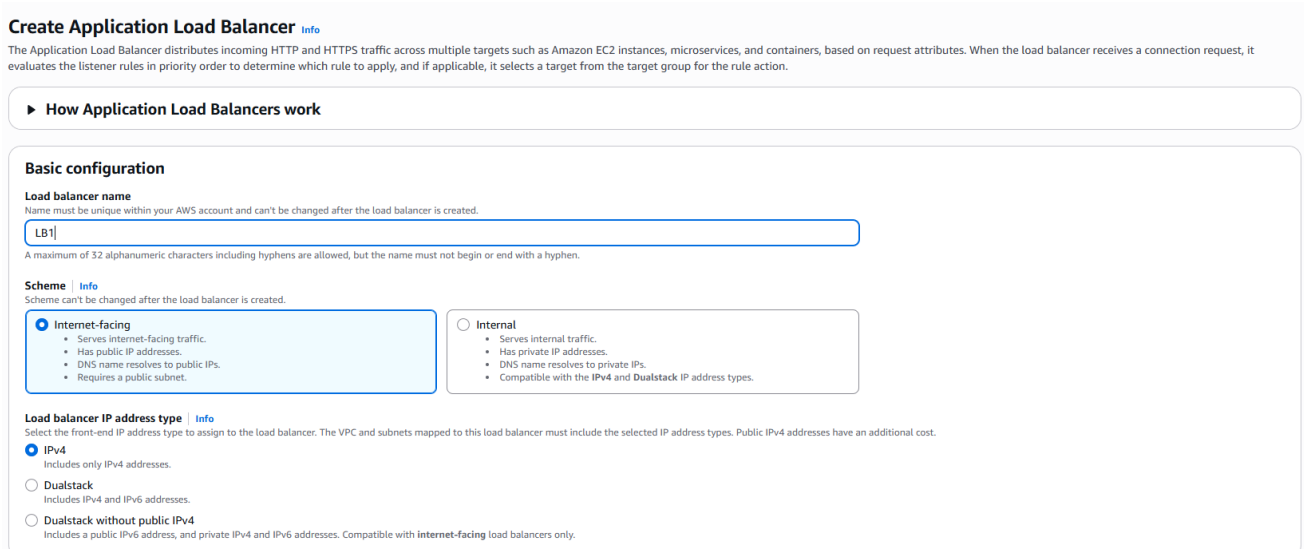

**Screenshot 5: Target Group creation**



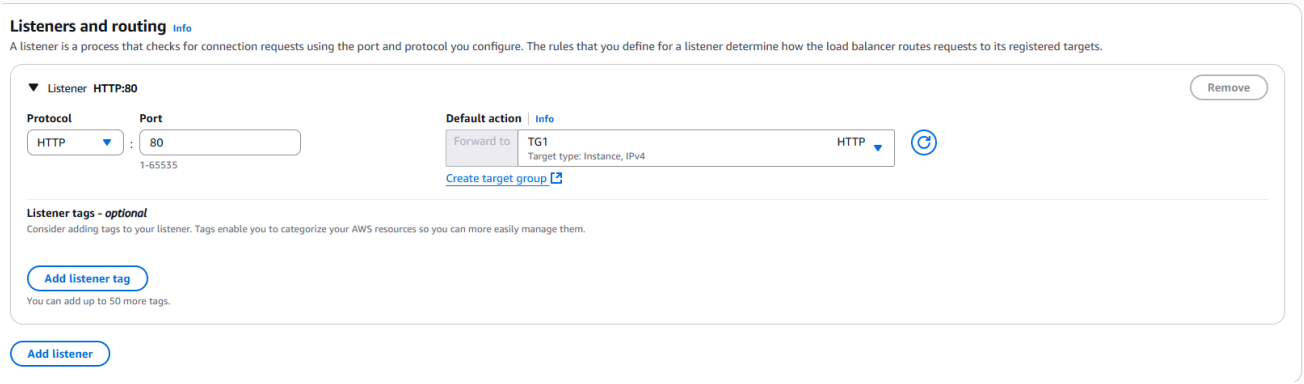

**Screenshot 6: Both EC2s registered as targets**

## 4. Create a Load Balancer

- Navigate to EC2 > Load Balancers
- Create **Application Load Balancer**
- Select **internet-facing**, port 80
- Choose the Security Group created in step 1
- Assign the **Target Group** from step 3



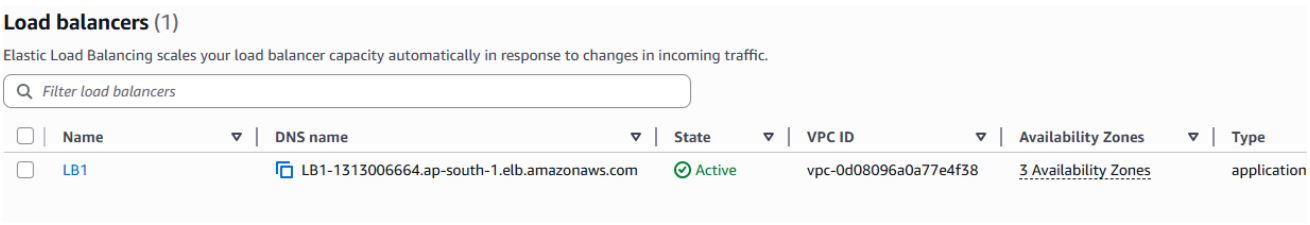

**Screenshot 7: Creating an Application Load Balancer (ALB)**



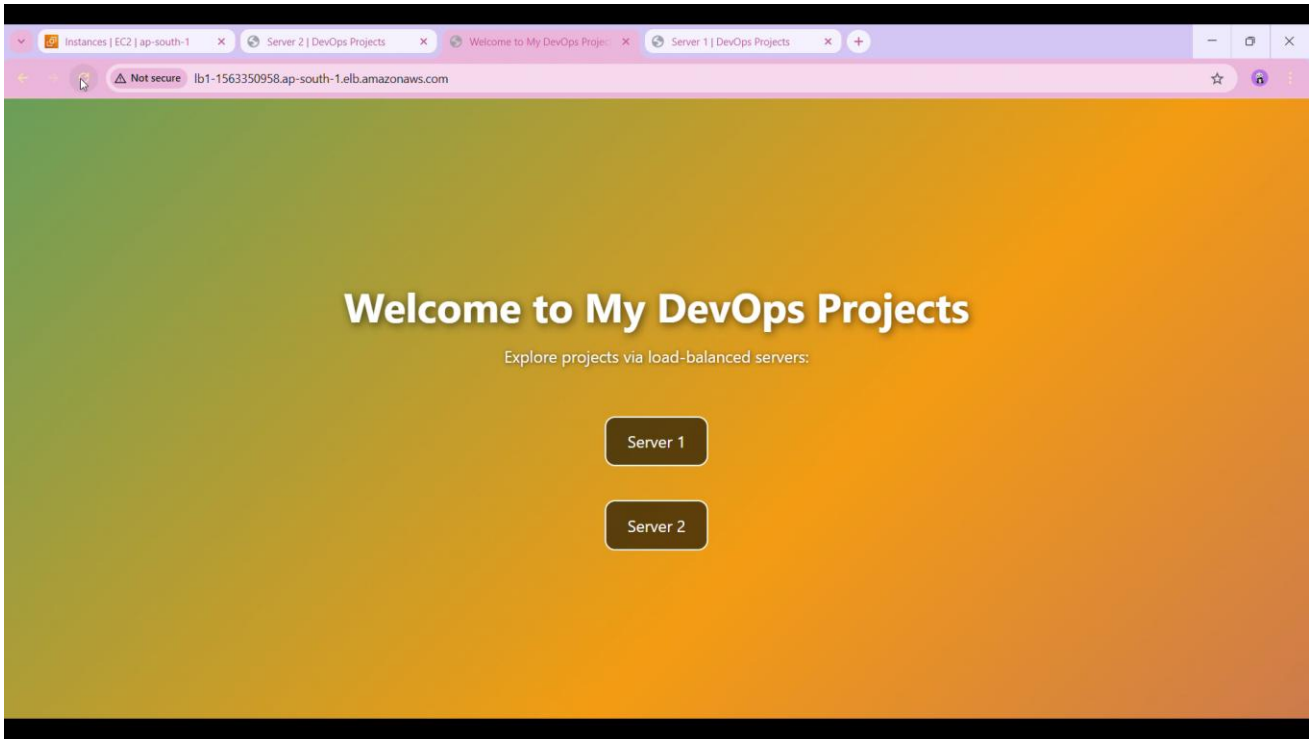

**Screenshot 8: Listener configuration and Target Group selection**

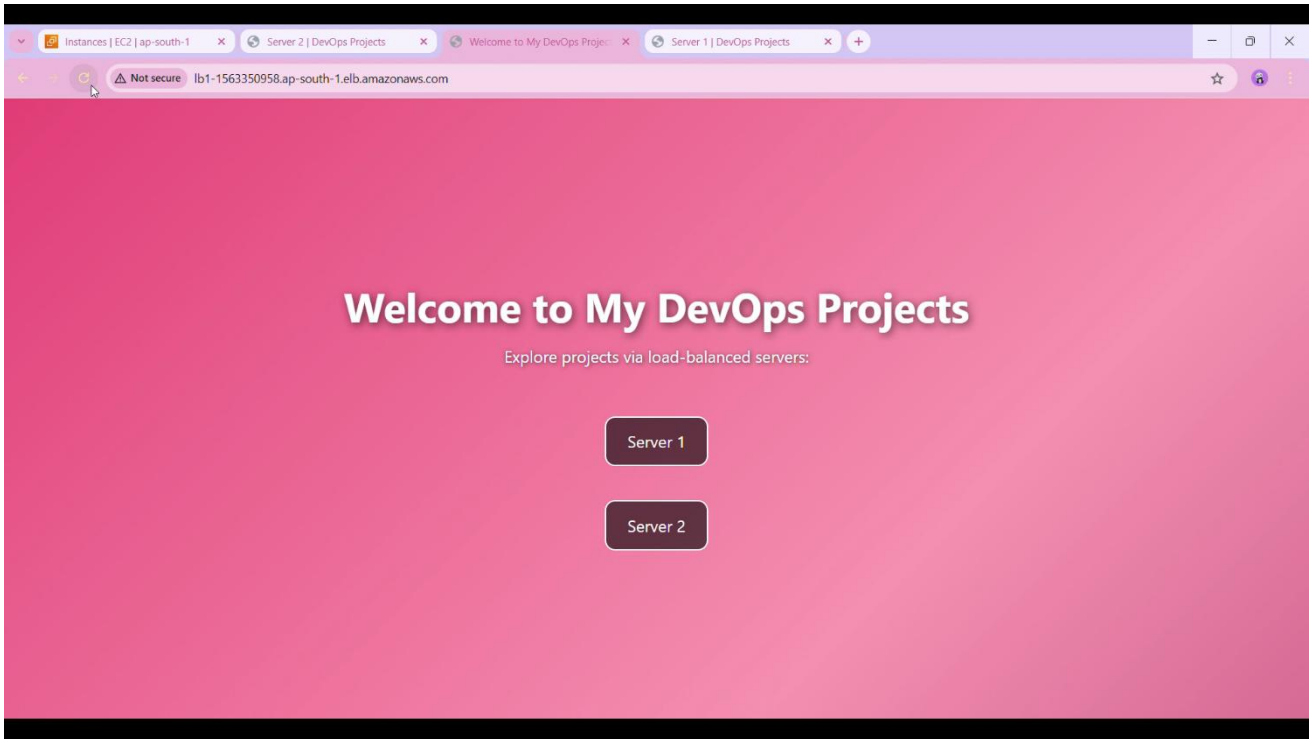
## 5. Access the Application via DNS

- Go to Load Balancer > Description
- Copy the **DNS name**
- Paste in browser → You should see content from Web Server 1 or 2 randomly (due to round robin)




**Screenshot 9: DNS Name of Load Balancer in EC2 Console**



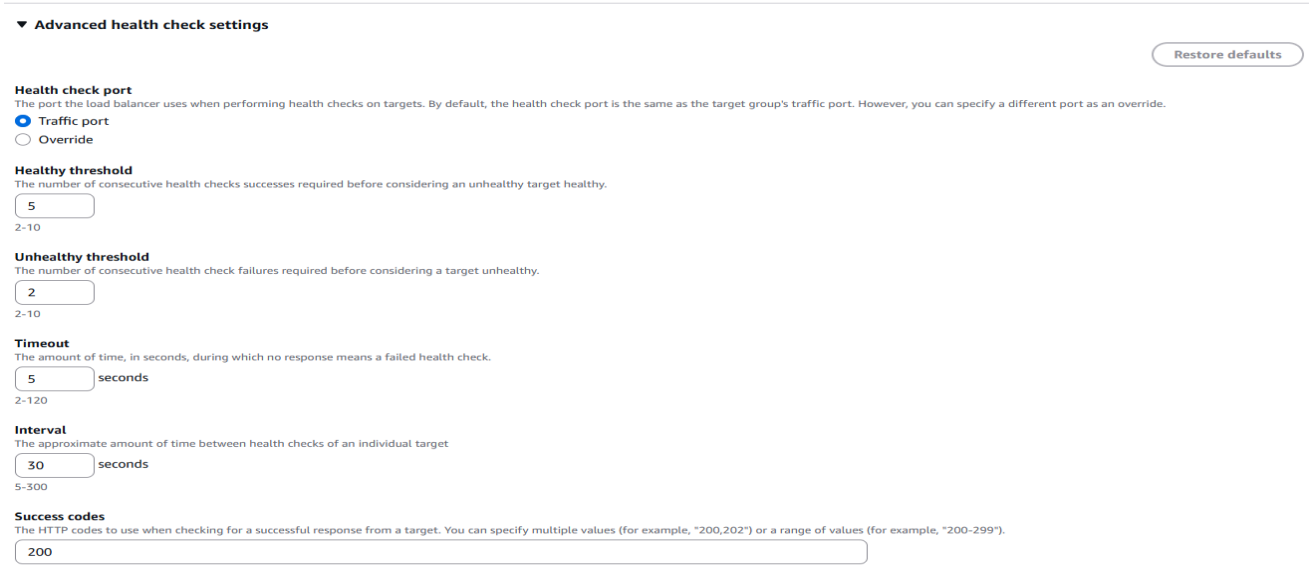




**Screenshot 10: Browser output showing Web Server 1 and 2 responses after refresh**

## Load Balancer Health Check Parameters

- **Timeout**: Wait time for instance to respond (e.g., 5 sec)
- **Interval**: Time between two health checks (e.g., 30 sec)
- **Healthy Threshold**: Min. success attempts to be considered healthy
- **Unhealthy Threshold**: Min. failures before marking unhealthy

The LB continuously **pings** instances. If it gets a 200 OK response within timeout, the instance is healthy.




**Screenshot 11: Health check configuration page**

## Load Balancing Strategies

### 1.Round Robin:

Request 1 → Server 1

Request 2 → Server 2

Request 3 → Server 1 (repeat)

### 2.Least Outstanding Requests:

Sends traffic to server with **least active requests**

## Conclusion

With just a few clicks and commands, you can set up a **robust, fault-tolerant Load Balancer** in AWS. This hands-on:

- Reinforces OSI Layer understanding
- Covers real-world deployment
- Teaches scaling strategies with ALB/NLB