Placement Empowerment Program

Cloud Computing and DevOps Centre

Set Up a Load Balancer in the Cloud Configure a load balancer to distribute traffic across multiple VMs hosting your web application.

Name: SAMRAJ K Department: CSE

Introduction

In this Proof of Concept (POC), the focus is on setting up a cloud-based Load Balancer using AWS to distribute traffic across multiple virtual machines (EC2 instances). Load Balancers play a crucial role in modern cloud architectures by ensuring high availability, fault tolerance, and scalability for web applications. This POC demonstrates the basic setup of an AWS Load Balancer, allowing traffic to be distributed between two EC2 instances running simple web servers.

Overview

The POC covers the following:

1. **Creating EC2 Instances**: Setting up two virtual machines (WebServer1 and WebServer2) in the AWS Free Tier.

2. **Configuring Web Servers**: Installing and configuring Apache HTTP Server on each instance to host simple HTML web pages.

3. **Setting Up a Load Balancer**: Creating an Application Load Balancer (ALB) to distribute incoming traffic evenly between the two EC2 instances.

4. **Testing the Load Balancer**: Verifying that the Load Balancer works by checking the DNS name and ensuring it alternates traffic between the two servers.

Objectives

1. To understand the process of creating and configuring EC2 instances in AWS.

2. To install and configure a web server (Apache HTTP Server) on Linux-based EC2 instances.

3. To set up an Application Load Balancer to distribute traffic across multiple servers.

4. To validate that the Load Balancer works as intended by testing it with unique responses from each server.

5. To build a foundational understanding of cloud-based load balancing for real-world use cases.

Importance

1. Scalability: Demonstrates how load balancing allows scaling applications by adding or removing servers as traffic demands change.

2. Fault Tolerance: Ensures that if one server goes down, the Load Balancer redirects traffic to the healthy server, improving reliability.

3. Cost Efficiency: Explores how to leverage AWS Free Tier services to test and deploy cloud-based solutions with minimal cost.

4. Hands-On Experience: Provides practical experience in configuring essential AWS services, an important skill for cloud computing professionals.

5. Foundation for Advanced Concepts: Sets the stage for more complex setups, such as auto-scaling, secure traffic distribution, and monitoring solutions.

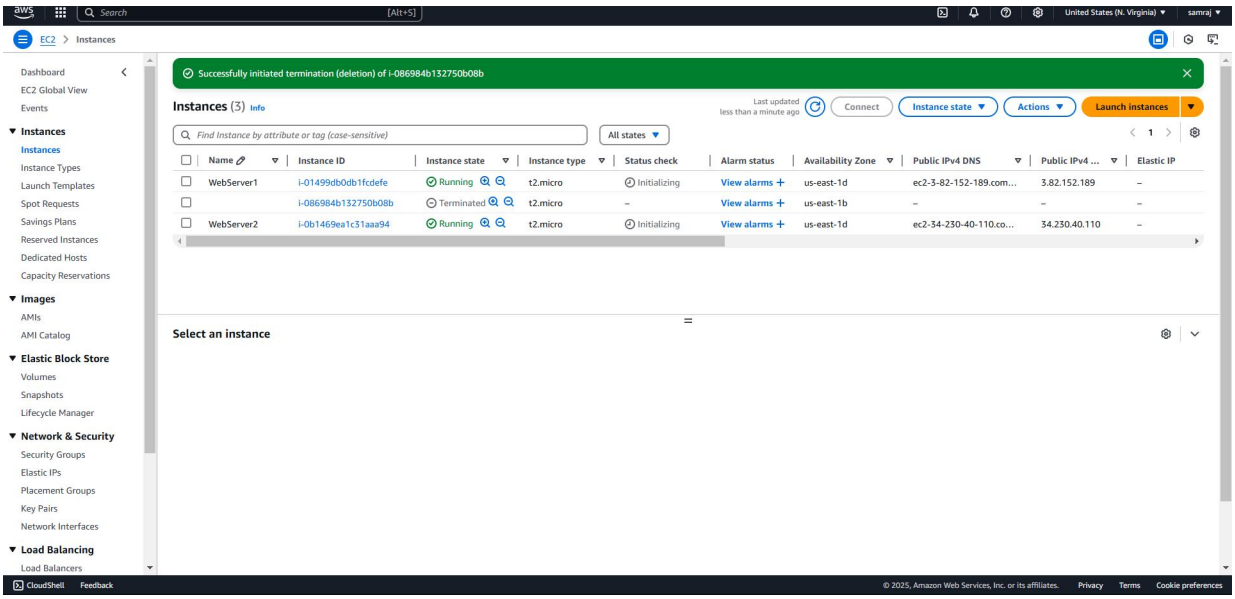
Step-by-Step Overview

Step 1:

1. Go to AWS Management Console.
2. Enter your username and password to log in.

Step 2:

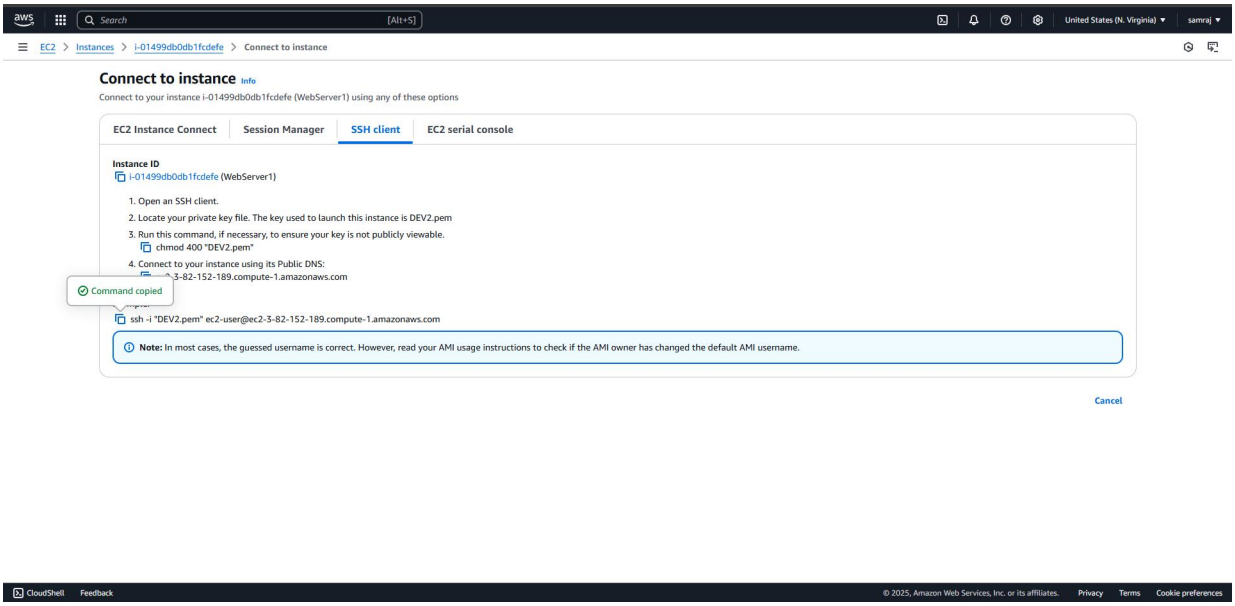
To create your instances, click **Launch Instance** and fill in the details: name the first instance "WebServer1," select **Amazon Linux 2 AMI (Free Tier eligible)** as the OS, and choose the **t2.micro** instance type. For the Key Pair, either select an existing one or create a new key pair to use for SSH access. Under **Network Settings**, click "Edit" and ensure "Allow HTTP traffic from the internet" is checked to enable web traffic. Keep the storage size at the default 8 GB, then click **Launch Instance**. Repeat the same steps for the second instance, naming it "WebServer2."



Step 3:

Click on **WebServer1**, then click **Connect**.

Use the instructions under **SSH client** to connect to your instance via terminal.



Step 3:

Run the following commands to install and start a web server

```
PS C:\Users\samni> cd downloads
PS C:\Users\samni\downloads> ssh -i "DEV2.pem" ec2-user@ec2-3-82-152-189.compute-1.amazonaws.com
[ec2-user@ip-172-31-91-108 ~]$ echo "Hello from Webserver1" | sudo tee /var/www/html/index.html
Hello from Webserver1
```

```
[ec2-user@ip-172-31-91-108 ~]$ sudo yum update -y
```

```
[ec2-user@ip-172-31-91-108 ~]$ sudo yum install httpd -y
```

```
[ec2-user@ip-172-31-91-108 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-91-108 ~]$ sudo systemctl enable httpd
```

```
[ec2-user@ip-172-31-91-108 ~]$ echo "Hello from Webserver1" | sudo tee /var/www/html/index.html
Hello from Webserver1
```

```
[ec2-user@ip-172-31-91-108 ~]$ exit
logout
Connection to ec2-3-82-152-189.compute-1.amazonaws.com closed.
```

Step 4:

Repeat these steps for **WebServer2** but change the message in the last command to:

```
PS C:\Users\samni> cd downloads
PS C:\Users\samni\downloads> ssh -i "DEV2.pem" ec2-user@ec2-34-230-40-110.compute-1.amazonaws.com
```

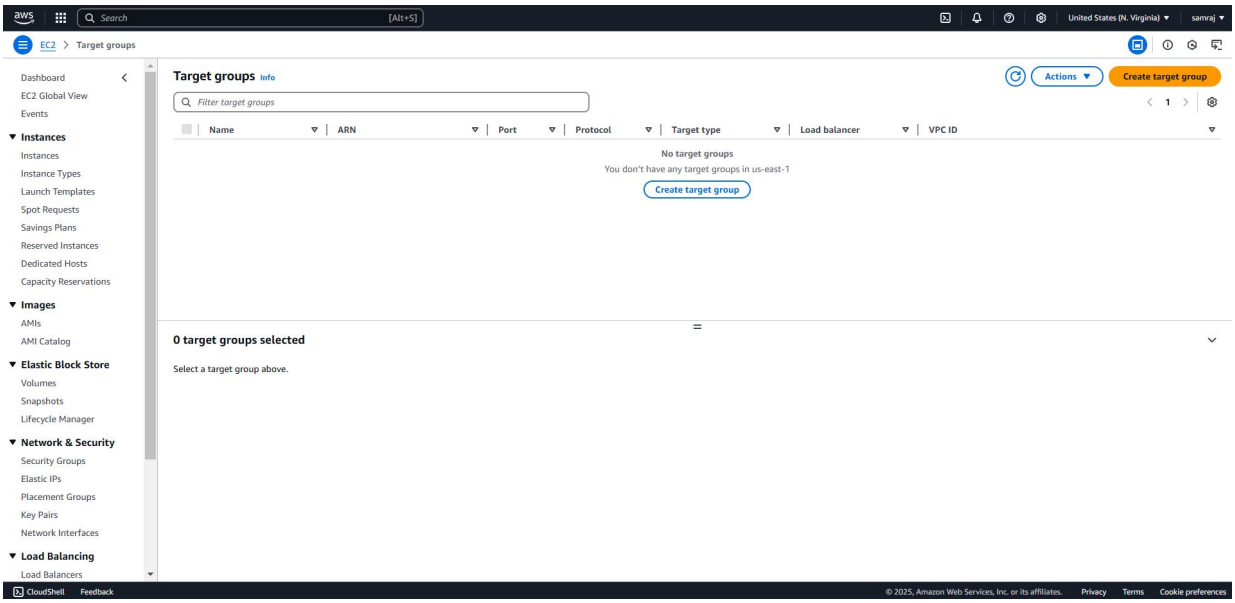
```
[ec2-user@ip-172-31-93-44 ~]$ echo "Hello from webserver2" | sudo tee /var/www/html/index.html
Hello from webserver2
```

Step 5:

1. In the **AWS Management Console**, go to the **EC2 Dashboard**.

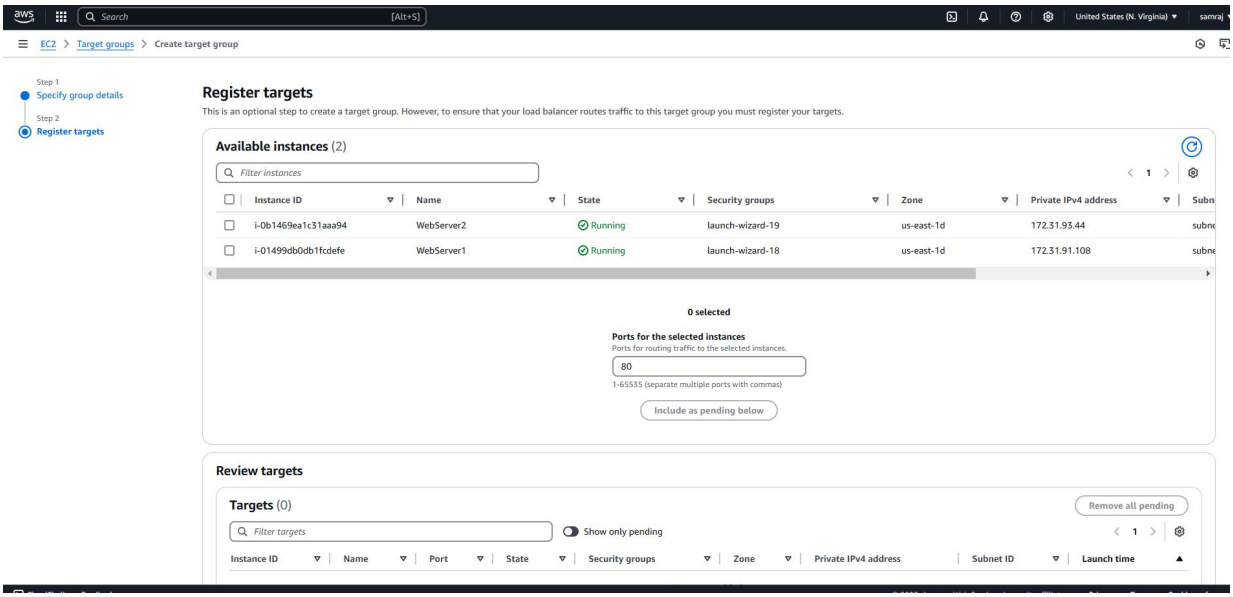
2. Scroll down and click on **Target Groups** under "Load Balancing."

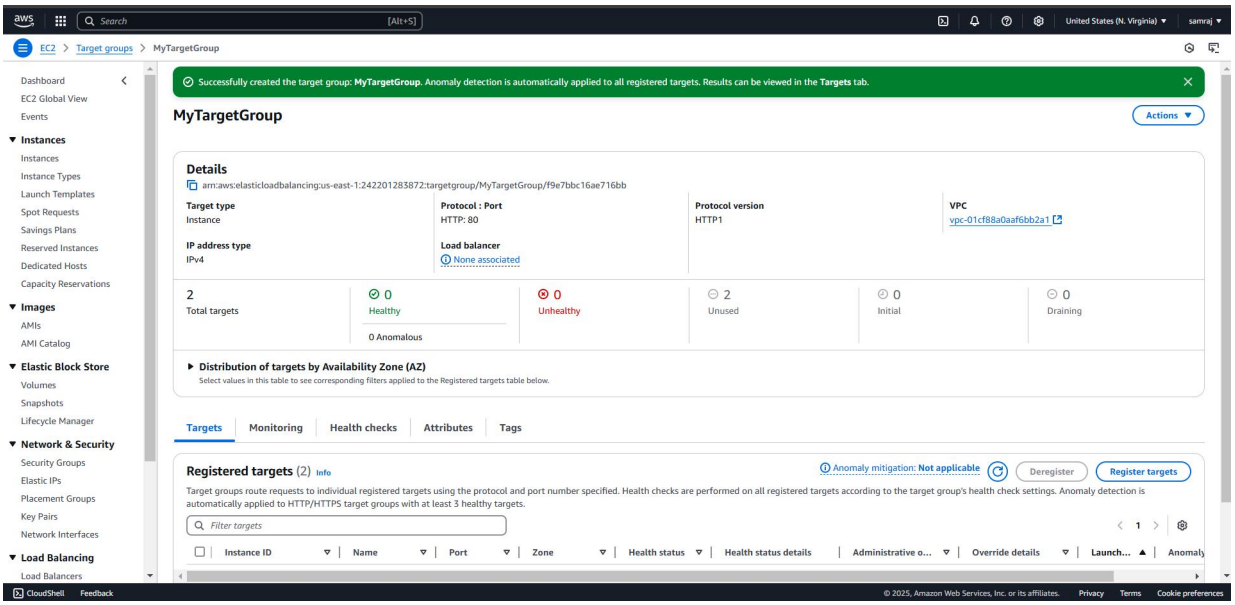
3. Click **Create Target Group**.



Step 6:

To create a target group, select **Instances** as the target type, name it (e.g., "MyTargetGroup"), set the **Protocol** to HTTP and **Port** to 80, and choose the same VPC as your EC2 instances (usually the default VPC). Keep the **Health Check Path** as / to verify the web server's status. Click **Next**, select both WebServer1 and WebServer2 under "Register Targets," click **Include as pending below**, and then create the target group.

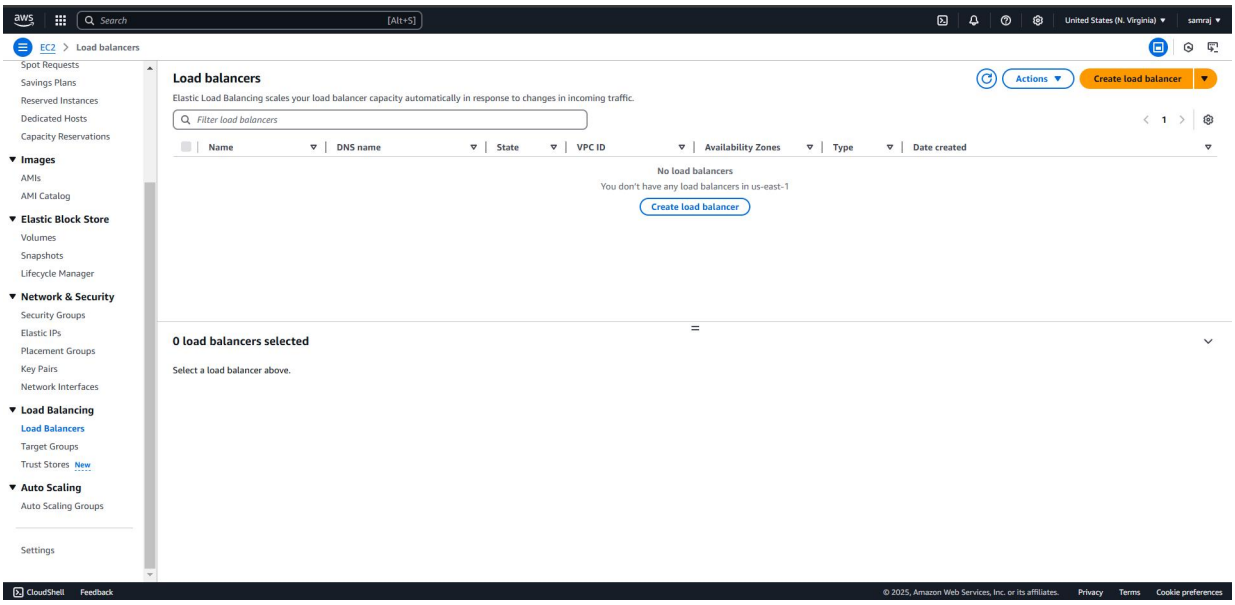


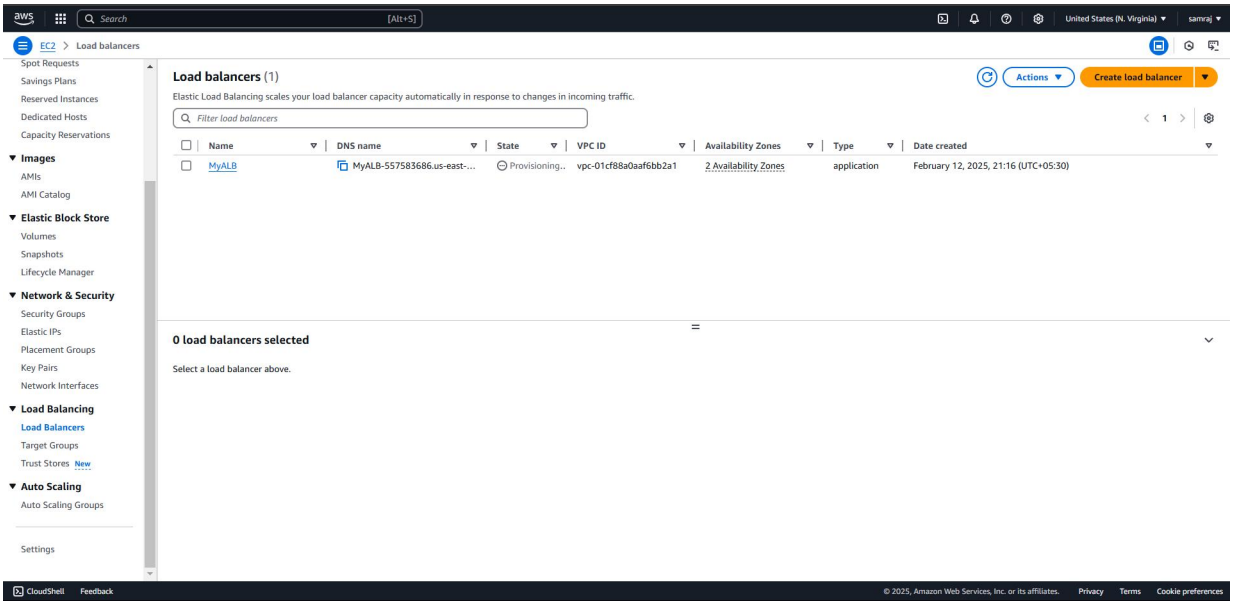


Step 7:

In the EC2 Dashboard, go to **Load Balancers** under "Load Balancing" and click **Create Load Balancer**. Select **Application Load Balancer (free tier eligible)** and configure it: name it (e.g., "MyALB"), set the **Scheme** to Internet-facing, **IP Address Type** to IPv4, and ensure the listener is HTTP on port 80. Select the VPC and at least two subnets for high availability. Skip the security settings since this is HTTP. On the **Security Groups** page, choose or create a security group that allows HTTP traffic. On the **Routing** page, select

the previously created target group (e.g., "MyTargetGroup") and click **Create Load Balancer**.





Step 8:

To verify the functionality of your Load Balancer:

1. Go to the **Load Balancers** section in the AWS Management Console.

2. Select your Load Balancer and find its **DNS name** under the **Description** tab.

3. Copy the DNS name and open it in your browser.

4. Refresh the page to confirm that traffic is being alternated between the two EC2 instances. You should see the messages **"Hello from WebServer1!"** and **"Hello from WebServer2!"** displayed alternately.

This confirms that the Load Balancer is correctly distributing traffic and ensuring high availability.

Outcome

By completing this POC of setting up an Application Load Balancer in AWS, you will:

1. Launch and configure two EC2 instances with Amazon Linux 2, each hosting a simple web server with unique content.

2. Create and configure an Application Load Balancer to distribute incoming traffic between the two EC2 instances.

3. Verify the functionality of the Load Balancer by accessing the DNS name and observing traffic alternation between the two web servers.

4. Understand the importance of Load Balancers in ensuring high availability and fault tolerance for web applications.