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: CHANDRU S

Department: INFORMATION TECHNOLOGY

Introduction

This Proof of Concept (POC) focuses on setting up a cloud-based Load Balancer using AWS to distribute traffic across multiple virtual machines (EC2 instances). Load Balancers are essential in modern cloud architectures, ensuring high availability, fault tolerance, and scalability for web applications.

In this POC, an AWS Application Load Balancer (ALB) is configured to distribute traffic evenly between two EC2 instances running simple web servers. This hands-on setup demonstrates the core principles of load balancing in the cloud.

Overview

The POC involves the following key steps:

1. **Creating EC2 Instances:** Launching two virtual machines—WebServer1 and WebServer2—using the AWS Free Tier.
2. **Configuring Web Servers:** Installing 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 the setup by accessing the Load Balancer's DNS name and ensuring traffic alternates between the two servers.

Objectives

This POC aims to achieve the following:

1. Understand how to create and configure EC2 instances in AWS.
2. Install and configure Apache HTTP Server on Linux-based EC2 instances.
3. Set up an Application Load Balancer to distribute traffic across multiple servers.
4. Validate the Load Balancer's functionality by testing with unique responses from each server.
5. Build a foundational understanding of cloud-based load balancing for real-world use cases.

Importance

1. **Scalability:** Load balancing facilitates seamless scaling by adding or removing servers based on traffic demands.
2. **Fault Tolerance:** If one server fails, the Load Balancer redirects traffic to healthy instances, ensuring reliability.
3. **Cost Efficiency:** This POC leverages AWS Free Tier services, making it a low-cost solution for testing and learning.
4. **Hands-On Experience:** It provides practical experience in configuring essential AWS services, a crucial skill for cloud professionals.

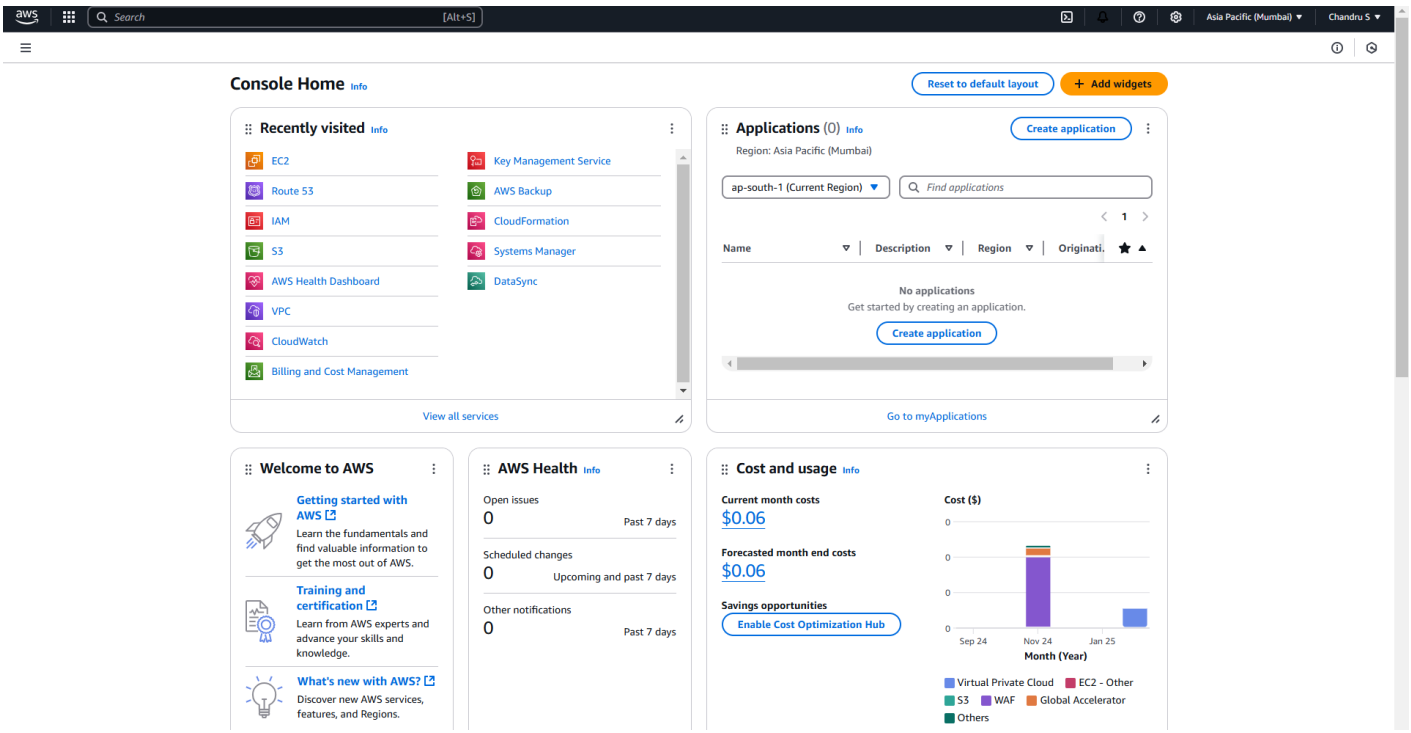
5. **Foundation for Advanced Concepts:** This setup serves as a stepping stone for more complex implementations, such as auto-scaling and secure traffic distribution.

Step-by-Step Overview

Step 1:

Access AWS Management Console

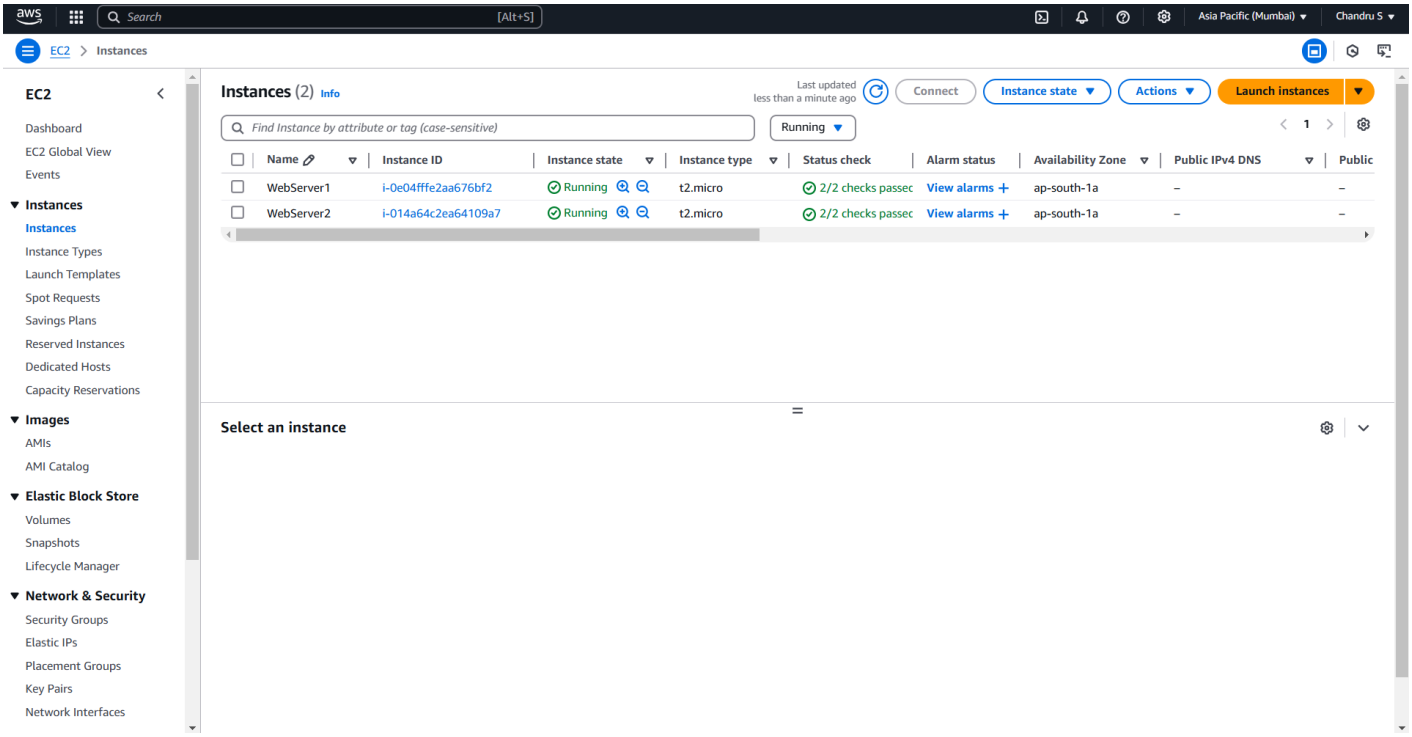
- Log in to the AWS Management Console with your credentials.



Step 2:

Launch EC2 Instances

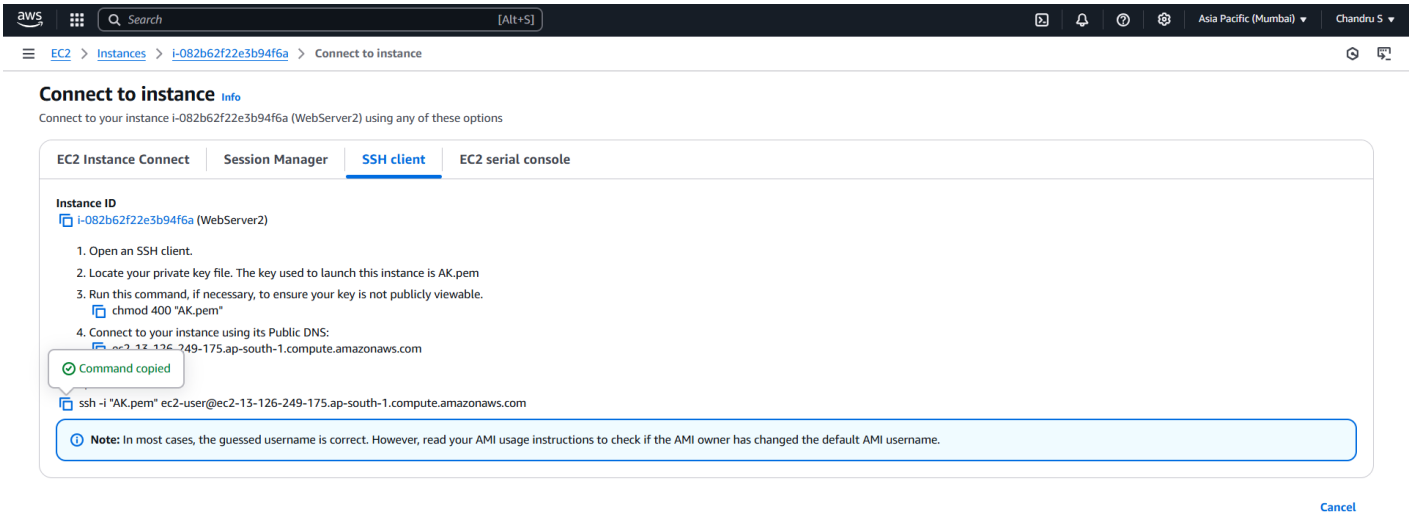
- In the **EC2 Dashboard**, click Launch Instance.
- Name the first instance **WebServer1** and select **Amazon Linux 2 AMI** (Free Tier eligible) as the OS.
- Choose **t2.micro** as the instance type.
- For the **Key Pair**, either select an existing key or create a new one for SSH access.
- Under **Network Settings**, click **Edit** and enable **Allow HTTP traffic** from the internet.
- Keep the default storage size (8 GB) and launch the instance.
- Repeat the same steps for the second instance, naming it **WebServer2**.



Step 3:

Connect to EC2 Instances

- In the **EC2 Dashboard**, select **WebServer1**, click **Connect**, and follow the SSH instructions.
- Use the **terminal** to connect to the instance.



Step 4:

Install and Configure Web Servers

- Run the following commands to install and start Apache on each instance:

```
PS C:\Users\chandru> cd "C:\Users\chandru\CHAN\AWS Cloud\30 days"
PS C:\Users\chandru\CHAN\AWS Cloud\30 days> ssh -i "AK.pem" ec2-user@ec2-13-126-249-175.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-126-249-175.ap-south-1.compute.amazonaws.com (13.126.249.175)' can't be established.
ED25519 key fingerprint is SHA256:8GGBV3OQ8PQKay/BDwUm/7WxryKhkrdUfXS/UktF/PY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-126-249-175.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
```

sudo yum update -y

```
[ec2-user@ip-10-0-4-44 ~]$ sudo yum update -y
```

sudo yum install httpd -y

```
[ec2-user@ip-10-0-4-44 ~]$ sudo yum install httpd -y
```

sudo systemctl start httpd

```
[ec2-user@ip-10-0-4-44 ~]$ sudo systemctl start httpd
```

sudo systemctl enable httpd

```
[ec2-user@ip-10-0-4-44 ~]$ sudo systemctl enable httpd
```

echo "Hello from WebServer1!" > /var/www/html/index.html

```
[ec2-user@ip-10-0-4-44 ~]$ echo "Hello from WebServer1!" | sudo tee /var/www/html/index.html
Hello from WebServer1!
```

exit

```
[ec2-user@ip-10-0-4-44 ~]$ exit
logout
Connection to ec2-13-126-249-175.ap-south-1.compute.amazonaws.com closed.
```

- Repeat the same steps for **WebServer2**, changing the message in the last command to:

```
PS C:\Users\chandru\CHAN\AWS Cloud\30 days> ssh -i "AK.pem" ec2-user@ec2-13-234-78-41.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-234-78-41.ap-south-1.compute.amazonaws.com (13.234.78.41)' can't be established.
ED25519 key fingerprint is SHA256:x9k9FUSzP2cWJ0SklphvZsSZe8iaKqCnnNo76iNhee4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-234-78-41.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

echo "Hello from WebServer2!" > /var/www/html/index.html

```
[ec2-user@ip-10-0-10-19 ~]$ echo "Hello from WebServer2!" | sudo tee /var/www/html/index.html
Hello from WebServer2!
```

exit

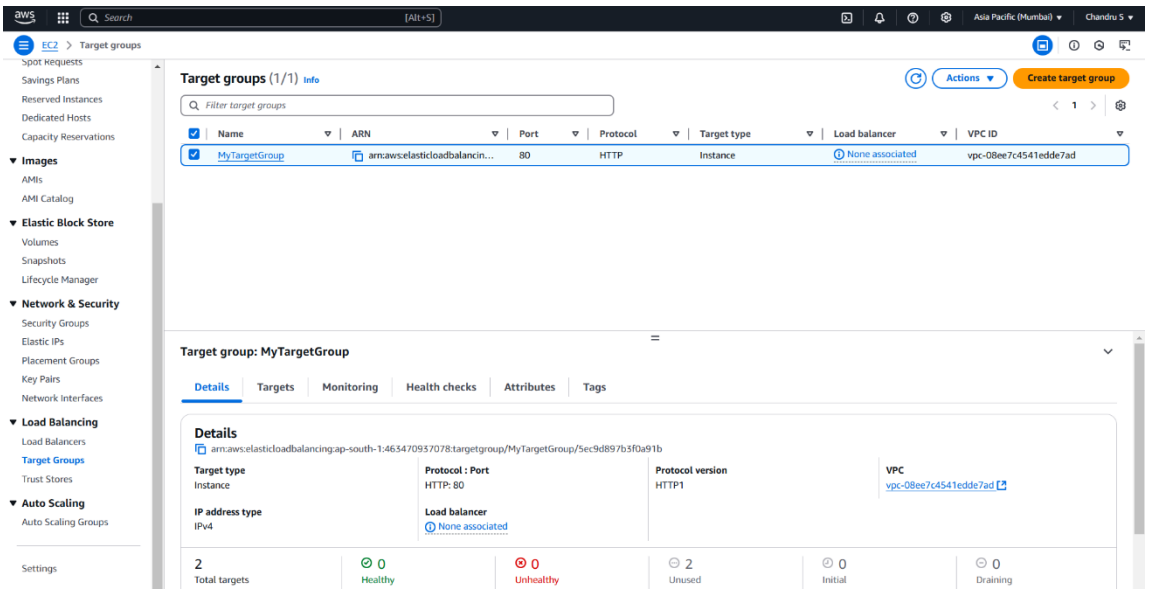
```
[ec2-user@ip-10-0-10-19 ~]$ exit
logout
Connection to ec2-13-234-78-41.ap-south-1.compute.amazonaws.com closed.
```

Step 5:

Create a Target Group

1. In the AWS Management Console, go to **EC2 Dashboard → Load Balancing → Target Groups.**
2. Click **Create Target Group** and configure the following:
   - **Target Type**: Instances
   - **Name**: MyTargetGroup
   - **Protocol**: HTTP, Port: 80
   - **VPC**: Same as EC2 instances (usually the default VPC)
   - **Health Check Path**: /

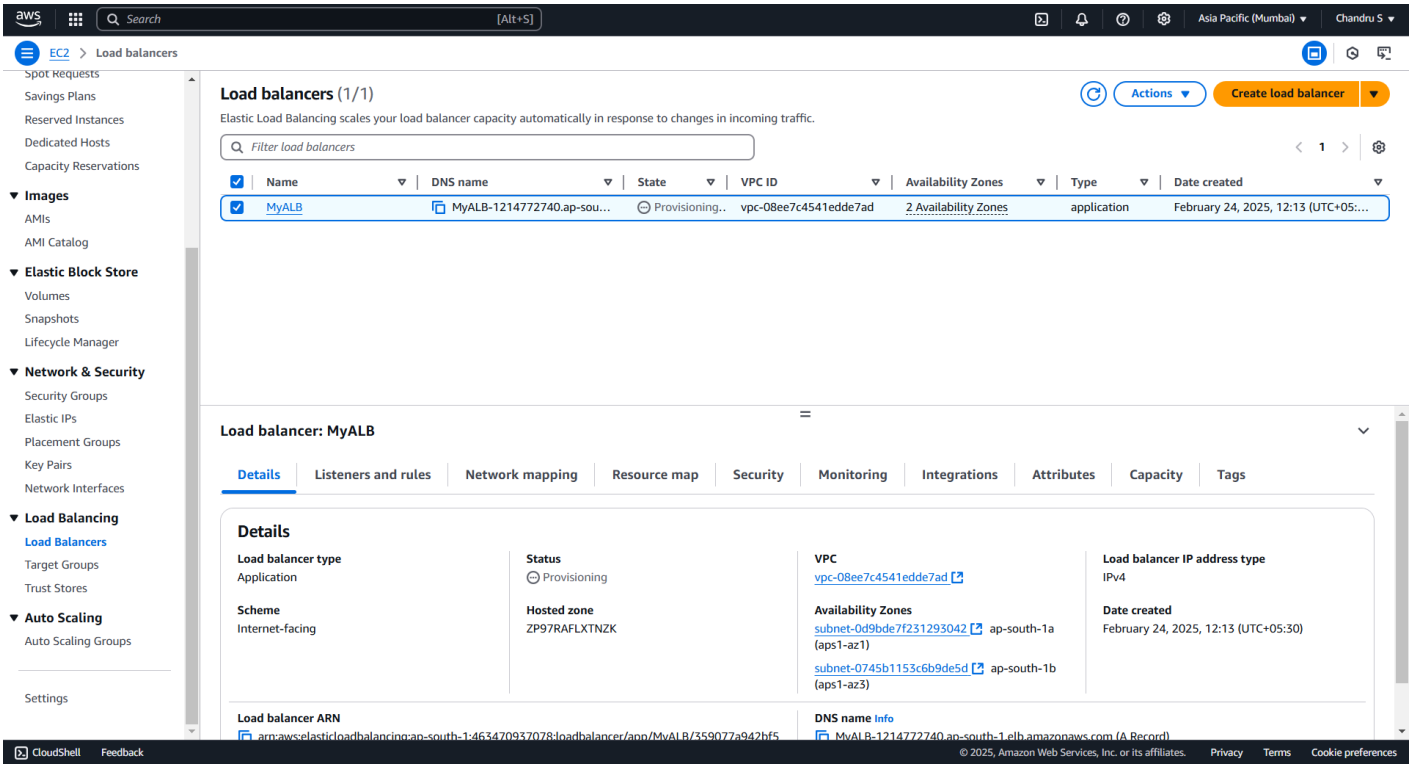
3. Click **Next**, select both **WebServer1** and **WebServer2**, include them as pending targets, and create the target group.



Step 6:

Create an Application Load Balancer (ALB)

1. In the EC2 Dashboard, go to **Load Balancers** and click **Create Load Balancer**.
2. Select **Application Load Balancer** (Free Tier eligible) and configure the following:
    - **Name:** MyALB
    - **Scheme:** Internet-facing
    - **IP Address Type:** IPv4
    - **Listener:** HTTP, Port 80
    - **VPC:** Default VPC, select at least two subnets.
3. On the **Security Groups** page, select or create a security group that allows HTTP traffic.
4. On the **Routing** page, select the previously created target group (*MyTargetGroup*) and finish the setup.



Step 7:

Verify Load Balancer Functionality

1. In the **Load Balancers** section, select **MyALB** and find its **DNS name** under the **Description** tab.
2. Copy the DNS name and open it in your browser.
3. Refresh the page multiple times—you should see the messages **"Hello from WebServer1!"** and **"Hello from WebServer2!"** alternately.

This confirms that the Load Balancer is correctly distributing traffic between the two EC2 instances.

Outcome

Upon completing this POC, you will:

1. Successfully launch and configure two EC2 instances running Apache web servers.
2. Set up an Application Load Balancer to distribute traffic between the instances.
3. Verify traffic distribution through the Load Balancer's DNS endpoint.
4. Gain practical insights into cloud-based load balancing and its role in enhancing web application availability and fault tolerance.