Setup overview:

A screenshot of a computer

Description automatically generated with low confidence

In the load balancing measurement setup, we aim to simulate the situation of multiple concurrent users accessing the same network resource as a webpage to the exact location. Then we will install a load balance strategy such as HAproxy, EC2 load balancer, and Nginx onto the frontend server. Based on the technologies and methods that we choose; the frontend server will forward the request to one of the agent servers to distribute the user load. We expect that using this can achieve a higher performance than a single node server. In addition, we will compare the technologies we mentioned above and decide which is the best and justify the reason.

1. **Install a web server to accept user request in backend**

We can build a simple website with a web page that accepts user requests and returns the result using Spring.

Graphical user interface

Description automatically generated with medium confidence

Then we build the web container file

Graphical user interface, text, application

Description automatically generated

Create a t2.micro Instant on Amazon cloud, then upload the web container file just build to the t2.micro Instant directory.

Graphical user interface, text

Description automatically generated

To start up the server on one of the backend servers, we will require to install java JDK.

Type "sudo apt-get update" first if first time running the EC2 instant.

Then Type “sudo apt install openjdk-11-jdk”

Text

Description automatically generated

After that, we can start the web server by typing “sudo java -jar loadbalancetestweb-0.0.1-SNAPSHOT.war”

Text

Description automatically generated

Now the server should be up and running. We can access the testing website with the ec2 instant IP with port 80

\*Remember to update the inbound rules for accepting internet connection from outside

A screenshot of a computer

Description automatically generated

Perform this step to set up more instants of serval more time in each backend server will be used in load balancing.

1. **Setup frontend server to accept requests from the user and perform load balance strategy**

In this step, we will be required to set up the environment for running the load balancing solution. Each of them needs to configure differently. Therefore, the following step will be separated into three subsections.

2.1 HAproxy load balancer setup procedure.

Create a t2.micro Instant on Amazon cloud, but this time we will use it to install different types of load balancing strategies.

Graphical user interface, text, application, email

Description automatically generated

Remote into the EC2 instant and type "sudo apt-get update" if first startup

Then Type “sudo apt install haproxy” to install haproxy.

Text

Description automatically generated

Haproxy should run automatically, to check the service status, type "sudo systemctl status haproxy"

Text

Description automatically generated

After that, we will need to configure our loadbalancer. Location the directory location /etc/haproxy/ and find haproxy.cfg

Graphical user interface, application

Description automatically generated

Add the following line to setup the load balance strategy and backend server load we just set in step 1.

一張含有 文字 的圖片

自動產生的描述

Load balancing strategy

The EC2 backend instances running the web application

Example in text

frontend haproxy\_inbound

bind :80

default\_backend haproxy\_httpd

backend haproxy\_httpd

balance roundrobin

server webapp1 172.31.86.115:80/loadtest1 check

server webapp2 172.31.93.219:80/loadtest1 check

To make the new setting take effect, we will need to restart the HAproxy service by typeing "sudo systemctl stop haproxy" then "sudo systemctl start haproxy"

Text

Description automatically generated

Then the load balancer should be working correctly. Access the load balancer public IP address to test the result.

Graphical user interface, text, application, email

Description automatically generated

You can check the result of the webpage

A picture containing text, cellphone, phone, displayed

Description automatically generated

Since it is round robin, you can refresh the webpage and see each backend server taking turns serving the page.

Graphical user interface, application, website

Description automatically generated

2.2 Nginx load balancer setup procedure.