Student: Condrea Adrian
Group: 258-1

Configuring Load Balancers using NGINX

Tutorial

Introduction

In this tutorial, we'll walk through the process of configuring load balancers for a system with redundancy, using NGINX. Load balancing is a critical component for distributing incoming network traffic across multiple servers, ensuring no single server bears too much demand. By spreading the load, it enhances the responsiveness and availability of your system.

Prerequisites

- A system set up with multiple instances. For this example, we will use a booking system with three services (bookingService, userManagementService, propertyService), each of them having two instances.
- NGINX installed

Installing NGINX

MacOS installation

For installing NGINX on MacOS, we will use Homebrew.
Homebrew can be installed by running the following command in a terminal:
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

After Homebrew is installed, we can install NGINX through the following command:
brew install nginx

Then, we can start NGINX with the command:
sudo nginx

Verify the installation by opening a browser and navigating to http://localhost:8080 (port can be configured, see step below)

Configuring NGINX

Locate nginx config using command **nginx -t** (e.g: /usr/local/etc/nginx/nginx.conf)

Configure port

In the **server** block, we can specify the port on which to run NGINX.
Observation: the default port 8080 clashes with zookeper's default port (which is needed for Kafka). Thus, I have set the port for NGINX to 8081.

Configure Upstream:

Inside the http block, define an upstream block. This will contain the list of your system servers.

Example:

```
upstream property_service {
      server localhost:3001;
      server localhost:3004;  # Second instance of Property Service
   }

   upstream booking_service {
      server localhost:3002;
      server localhost:3005;  # Second instance of Booking Service
   }

   upstream user_management_service {
      server localhost:3003;
      server localhost:3006;  # Second instance of User Management Service
   }
```

Create a server block

Within the same http block, define a server block to handle incoming requests.

Example:

```
server {
      listen 80;

      location /api/properties {
         proxy_pass http://property_service;
      }

      location /api/bookings {
         proxy_pass http://booking_service;
      }

      location /api/users {
         proxy_pass http://user_management_service;
      }

      location / {
         proxy_pass http://localhost:3000;
      }
```

This configuration directs all traffic coming to port 80 (default http port) to the servers defined in the upstream block.

Load balancing techniques

NGINX supports various methods of load balancing like round-robin (default), least-connected, and ip-hash. You can specify the method in the upstream block.

Example:

```
upstream booking_servers {
   least_conn; # Uses the least-connected method
   server backend1.example.com;
   server backend2.example.com;
}
```

Testing the load balancing

Load balancing can be tested by running the following shell script:

```
#!/bin/bash

# NGINX server URL
SERVER_URL="http://localhost/api/properties"  # Replace with your actual NGINX server URL

# Number of requests to send
REQUEST_COUNT=10

echo "Sending $REQUEST_COUNT requests to $SERVER_URL..."

for ((i=1; i<=REQUEST_COUNT; i++))
do
  echo "Request $i:"
  curl -s "$SERVER_URL"
  echo -e "\n"
done

echo "Test completed."
```