**Tutorial – Load Balancing with Spring Clout Netflix Zuul and Eureka**

**Overview**

Client-side service discovery allows services to find and communicate with each other without hard-coding the hostname and port. The only ‘fixed point’ in such an architecture is the service registry, with which each service has to register.

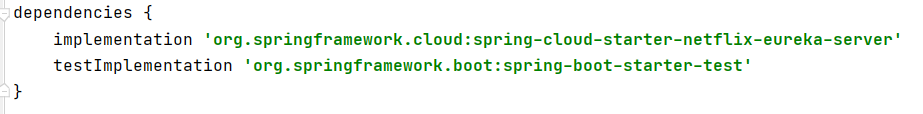
With Netflix Eureka, each client can simultaneously act as a server to replicate its status to a connected peer. In other words, a client retrieves a list of all connected peers in a service registry and makes all further requests to other services through a load-balancing algorithm.

This tutorial presents how to route requests to a REST Service discovered by Spring Cloud Eureka through Zuul Proxy.

**1. Eureka Server**

- Create a new Gradle project. The JDK version should be 8 or higher.

- Add the proper dependencies in build.gradle file:



- Create the main application class (make sure you annotate it with @EnableEurekaServer):

A computer code with black text

Description automatically generated

- Configure the parameters in application.properties. We need to specify the server port, together with 2 other parameters that tell the built-in *Eureka Client* not to register with itself because our application should be acting as a server.

A close-up of a computer code

Description automatically generated

After configuring the Eureka Server, browsing to [http://localhost:8761](http://localhost:8761/) should display the Eureka dashboard.

**2. Eureka Client**

- Create a new Gradle project representing a new microservice. It will be the client of the Eureka Server. In order to enable that, we need to import the following dependencies:

A close up of words

Description automatically generated

- Configure the properties of the eureka client:

A computer screen shot of a code

Description automatically generated

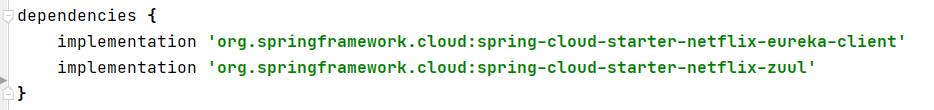
- Enable discovery client in the main application class:

A computer screen shot of a program code

Description automatically generated

**3. Zuul Gateway**

- Create a new Gradle project representing the Zuul Api Gateway service. Import the following dependencies in build.gradle:



- Set the properties referring to Zuul being a client of Eureka Server:

A computer screen shot of a computer code

Description automatically generated

- Set the gateway properties, mapping the paths to the right services:

A computer screen shot of a program

Description automatically generated

- Enable Eureka Client and Zuul Proxy on the main application class:

A computer code with black text

Description automatically generated

**4. Test**

Now the setup is complete. We should run the Eureka Server, the Zuul gateway and the microservice. We can view the registered clients accessing <http://localhost:8761>

A screenshot of a computer

Description automatically generated

Also, when making a request to the zuul service, it should do a proper redirection to the microservice (ex. calling <http://localhost:8765/api/user>).