Experiments

In this lesson, we'll discuss some experiments you can do on the Netflix stack.

We'll cover the following

- Try the experiments in the following widget!
- Additional microservice
- Scaling and load balancing
- Simulate failure
- Extend access using Hystrix
- Extend Zuul setup
- Filter with Zuul
- Create your own microservice

Try the experiments in the following widget!

```
version: '3'
services:
  eureka:
    image: educative1/mapi_ms_eureka
    ports:
      - "8761:8761"
  customer:
    image: educative1/mapi_ms_customer
    links:
     - eureka
  catalog:
    image: educative1/mapi_ms_catalog
    links:
     - eureka
  order:
    image: educative1/mapi_ms_order
    links:
     - eureka
  zuul:
    image: educative1/mapi_ms_zuul
    links:
     - eureka
    ports:
      - "8080:8080"
  turbine:
    image: educative1/mapi_ms_turbine
    links:
     - eureka
    ports:
      - "8989:8989"
```

Additional microservice

Supplement the system with an additional microservice.

- A microservice that is used by a call center agent to create notes for a call can be used as an example. The call center agent should be able to select the customer.
- You can copy and modify one of the existing microservices. microservices.
- Register the microservice in Eureka.

- The customer microservice must be called via Ribbon. The microservice will be found automatically via Eureka, otherwise, the microservice must be looked up explicitly in Eureka.
- Package the microservice in a Docker image and add the image to `docker-compose.yml`. There you can also determine the name of the Docker container.
- Create a link in `docker-compose.yml` from the container with the new service to the container `eureka`. That way the microservice can register at the Eureka server.
- The microservice must be accessible from the homepage. To do this, you have to create a link similar to the other links in the file `index.html` in the Zuul project. Zuul automatically sets up the routing for the microservice as soon as the microservice is registered in Eureka.

Scaling and load balancing

Try scaling and load balancing.

- Increase the number of instances of a service with `docker-compose up --scale customer=2`.
- Use the Eureka dashboard to determine whether two customer microservices are running. It is available at port 8761, at http://localhost:8761/ (http://localhost:8761/) when Docker is running on the local computer.
- Observe the logs of the order microservice with `docker logs -f ms_order_1` and have a look at whether different instances of the customer microservice are called. This should be the case because

Ribbon is used for load balancing. For this, you have to trigger requests to the order application, e.g., a simple reload of the starting page.

Simulate failure

Simulate the failure of a microservice.

- Watch the logs of the order microservice with `docker logs -f ms_order_1` and have a look at how the catalog microservice is called. For this, you have to trigger requests to the order application. For example, you can reload the starting page.
- Find the IP address of the order microservice with the help of the Eureka dashboard at port 8761, http://localhost:8761/ (http://localhost:8761/) when Docker is running locally.
- Open the Hystrix dashboard at port 8989 on the Docker host, at http://localhost:8989/ (http://localhost:8989/) when Docker is running on the local computer.
- Using this IP address enter the URL of the Hystrix JSON data stream in the Hystrix dashboard. This can be http://172.18.0.6:8080/actuator/hystrix.stream (http://172.18.0.6:8080/actuator/hystrix.stream). The Hystrix dashboard should show closed circuit breakers.
- Shut down all catalog instances with `docker-compose up --scale catalog=0`.
- Watch the log of the order microservices during the next calls.
- Also observe the Hystrix dashboard. The circuit breaker will only open when multiple calls have failed.

open when multiple calls have failed.

- When the circuit breaker is open, the order microservice should work again since the fallback is activated, then a cached value is used.

Extend access using Hystrix

Only access to the catalog microservice is safeguarded with Hystrix.

In the order microservice the class `CustomerClient` in package `com.ewolff.microservice.order.clients` implements the access to the customer microservice.

Extend the access to the customer microservice using Hystrix. For this, use the class `CatalogClient` from the same package as an example.

Extend Zuul setup

Extend the Zuul setup by a fixed route. In `application.yml`, in directory `src/main/resource`, in project `microservice-demo-zuul-server` add for example the following to make the INNOQ homepage appear at http://localhost:8080/innoq (http://localhost:8080/innoq):

```
zuul:
  routes:
    innoq:
      path: /innoq/**
      url: http://innoq.com/
```

Filter with Zuul

Add a filter to the Zuul configuration.

A tutorial dealing with Zuul filters can be found at https://spring.io/guides/gs/routing-and-filtering/ (https://spring.io/guides/gs/routing-and-filtering/).

Create your own microservice

- Create your own microservice that only returns simple HTML.
- Integrate it into Eureka and deploy it as part of the Docker compose environment.
- If it is registered in Eureka, it can be addressed immediately from the Zuul proxy at a URL like http://localhost:8080/mymicroservice (http://localhost:8080/mymicroservice) .

We'll conclude this chapter with the next lesson.

← Back
Variations

Next →
Introduction

Mark as Completed


Report an issue