Experiments

In this lesson, we'll look at a few experiments that can be done with the example we previously looked at.

We'll cover the following

- Examine logs
- Create new microservice
- Other experiments

Examine logs

Try the following experiments in the coding environment given below!

- Start the system and examine the logs of *microservice-order-invoicing* and *microservice-order-shipping* with `docker logs -f msatom_invoicing_1` respectively `docker logs -f msatom_shipping_1`.
- The **microservices log messages when they poll data** from the Atom feed, because there are new orders.
- If you start additional instances of a microservice with `docker compose up --scale`, these **new instances will** collect orders via the Atom feed and **log information about them**. In doing so, only one instance writes at a time; the other ones ignore the data.
- **Create orders and notice this behavior based on the log messages**.
- Explore the code to find out what the log messages mean and where

they are put out.

```
version: '3'
services:
  apache:
    image: educative1/mapi_msatom_apache
    links:
     - order
     - shipping
     - invoicing
    ports:
     - "8080:80"
  postgres:
    image: educative1/mapi_msatom_postgres
    environment:
      POSTGRES_PASSWORD: dbpass
      POSTGRES_USER: dbuser
  order:
    image: educative1/mapi_msatom_order
    links:
     - postgres
  shipping:
    image: educative1/mapi_msatom_shipping
    links:
     - order
     - postgres
  invoicing:
    image: educative1/mapi_msatom_invoicing
    links:
     - order
     - postgres
```

Create new microservice

Supplement the system with an additional microservice.

- As an example, a microservice can be used that credits the customer with a bonus depending on the value of the order or that counts the orders.
- Of course, you can copy and modify one of the existing microservices.
- Implement a microservice which polls the URL `http://order:8080/feed`.
- In addition, the microservice should display an HTML page with

some information (customer bonus or number of calls).

- Package the microservice in a Docker image and reference it in `docker-compose.yml`. There you can also determine the name of the Docker container.
- Create a link from the container `apache` to the container with the new service in `docker-compose.yml` and from the container with the new service to the container `order`.
- The microservice has to be accessible via the homepage. For this purpose, you have to create a load balancer for the new Docker container in the file `000-default.conf` in the Docker container `apache`. Use the name of the Docker container for this. Then, add a link to the new load balancer in `index.html`.
- Optional: Add HTTP caching format.

Other experiments

- Currently, it is only possible to request all orders at once in the Atom feed. **You can implement paging so that only a subset of the orders is returned**.

- At the moment, the system runs with Docker compose. However, it could also run on a different infrastructure. **Port the system to one of these platforms**:

  - On a microservices platform (chapter 12 (https://www.educative.io/collection/page/10370001/5441945024331776/5768973159235584)).
  - On Kubernetes. Chapter 13 (https://www.educative.io/collection/page/10370001/5441945024331776/4922985196552192) discusses Kubernetes in more detail.
  - On Cloud Foundry. chapter 14 (https://www.educative.io/collection/page/10370001/5441945024331776/5727581686988800) deals with Cloud Foundry.

- Instead of using the Atom format, you could also deliver **your own representation of a feed**.
    - For example, as a JSON document. Change the implementation in the example so that it uses its own custom data.

We'll conclude this chapter in the next lesson.

← Back

Variations

Next →

Introduction

☑ Mark as Completed

Report an Issue