САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая/Лабораторная работа

Выполнил:

Пластун Елизавета Олеговна

Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

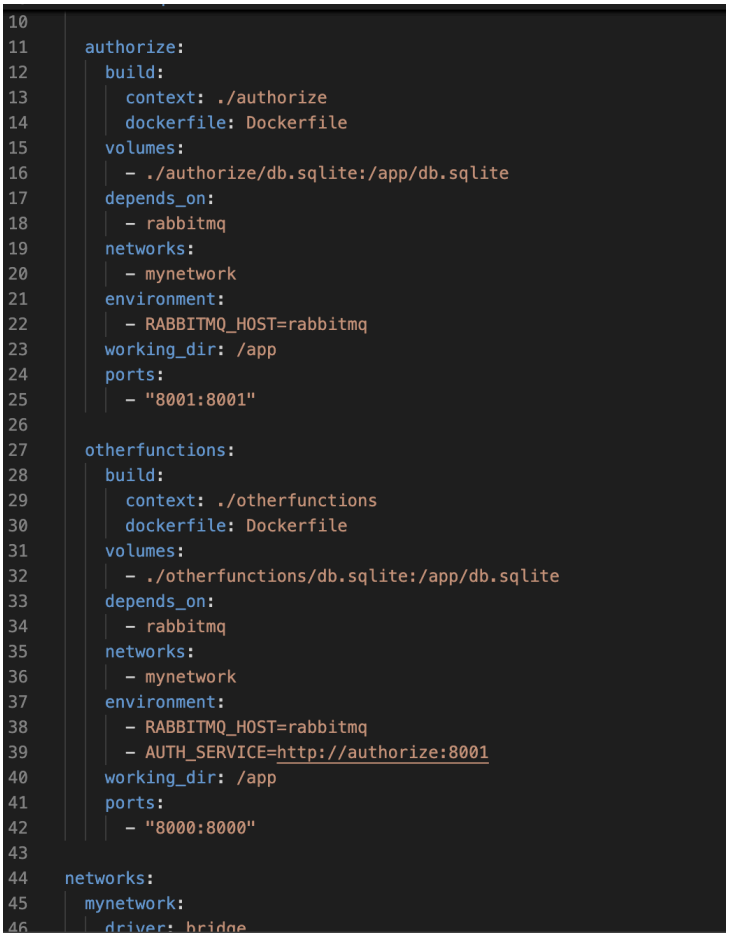
Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой посредством RabbitMQ. Делать это можно как с помощью docker-compose так и с помощью docker swarm. При разумном использовании swirl вы получите дополнительные баллы.

Ход работы

Сначала создаем Dockerfile для микросервисов

```
FROM node:18.15

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]

```

Также создаем docker compose

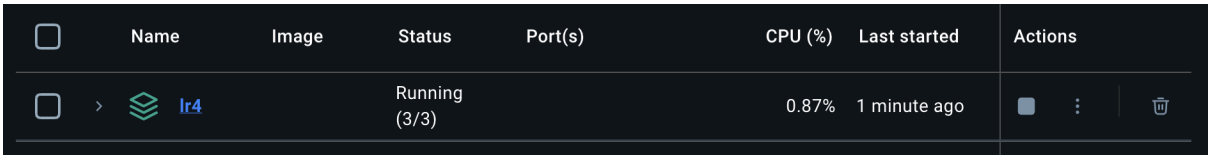


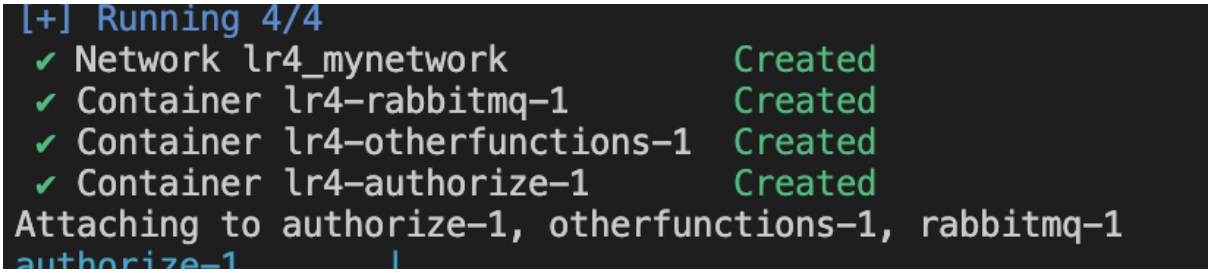
```
10
11    authorize:
12      build:
13        context: ./authorize
14        dockerfile: Dockerfile
15      volumes:
16        - ./authorize/db.sqlite:/app/db.sqlite
17      depends_on:
18        - rabbitmq
19      networks:
20        - mynetwork
21      environment:
22        - RABBITMQ_HOST=rabbitmq
23      working_dir: /app
24      ports:
25        - "8001:8001"
26
27    otherfunctions:
28      build:
29        context: ./otherfunctions
30        dockerfile: Dockerfile
31      volumes:
32        - ./otherfunctions/db.sqlite:/app/db.sqlite
33      depends_on:
34        - rabbitmq
35      networks:
36        - mynetwork
37      environment:
38        - RABBITMQ_HOST=rabbitmq
39        - AUTH_SERVICE=http://authorize:8001
40      working_dir: /app
41      ports:
42        - "8000:8000"
43
44  networks:
45    mynetwork:
46      driver: bridge
```

В config микросервисов создаем rabbitmq

```
import amqp from 'amqplib/callback_api';
import dotenv from 'dotenv';

dotenv.config();

const rabbitmqHost = process.env.RABBITMQ_HOST || 'localhost';

amqp.connect(`amqp://${rabbitmqHost}`, function(error0, connection) {
  if (error0) {
    throw error0;
  }
  connection.createChannel(function(error1, channel) {
    if (error1) {
      throw error1;
    }

    const queue = 'authorize';

    channel.assertQueue(queue, {
      durable: false
    });

    channel.sendToQueue(queue, Buffer.from('готово'));
    console.log(" [x] Sent 'готово'");
  });
});

```

Далее собираем и запускаем контейнеры





Сетевое взаимодействие между сервисами обеспечивается с помощью Docker-сети, что позволяет контейнерам взаимодействовать друг с другом по именам хостов.

Вывод

В четвертой лабораторной работе удалось упаковать приложение в docker-контейнеры и обеспечить сетевое взаимодействие между

различными частями приложения с использованием RabbitMQ и docker-compose.