САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 4

Выполнил:

Рыбалко Олег

Группа K33392

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

Необходимо упаковать приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями приложения, а также настроить общение микросервисов между собой посредством RabbitMQ.

Ход работы

1. Создадим Dockerfile для каждого сервиса

```
FROM node:18.15 as build

WORKDIR /app

COPY package.json .
RUN npm i

COPY . .

RUN npm run build

ENTRYPOINT [ "npm", "run", "serve" ]

```

2. Создадим сервис, который будет получать и обрабатывать сообщения из RabbitMQ. В нашем случае сервис будет записывать логи другого сервиса в базу данных.

```
channel.consume(
  queue,
  function (msg: Message | null) {
    if (!msg) return
    const log = JSON.parse(msg.content.toString()) as Log
    console.log(`Received: ${msg.content.toString()}`)
    db.run(
      `INSERT INTO logs (path, code, method) VALUES (?, ?, ?)`,
      [log.path, log.code, log.method],
      function (error: Error) {
        if (error) {
          console.log('Failed to save a log: %s', error)
        }
      }
    )
  },
  {
    noAck: true,
  }
)
```

3. Добавим middleware в основной сервис, который будет отправлять сообщение в очередь после каждого запроса

```
connection.createChannel(function (error1: Error, channel: Channel) {
  if (error1) {
    throw error1
  }

  const queue = 'logs_queue'
  var msg = JSON.stringify({
    code: res.statusCode,
    path: req.path,
    method: req.method,
  })

  channel.assertQueue(queue, {
    durable: false,
  })
  channel.sendToQueue(queue, Buffer.from(msg))

  console.log(' [x] Sent %s', msg)
})
```

4. Создадим docker-compose.yaml, в который добавим все сервисы и RabbitMQ.

```
backend:
  build:
    context: ./lab1
  environment:
    - "AUTH_URL=http://auth:9091"
  ports:
    - "9090:9090"
  depends_on:
    - auth
    - rabbitmq

auth:
  build:
    context: ./lab3

logs:
  build:
    context: ./lab4
  depends_on:
    rabbitmq:
      condition:
        service_healthy
```

```
gateway:
  image: caddy
  ports:
    - "80:80"
  volumes:
    - ./Caddyfile:/etc/caddy/Caddyfile
  depends_on:
    - auth
    - backend

rabbitmq:
  image: "rabbitmq:3"
  healthcheck:
    test: rabbitmq-diagnostics -q ping
    interval: 5s
    timeout: 5s
    retries: 5
```

Вывод

В данной лабораторной работе удалось создать микросервис для работы с логами, создать Dockerfile для создания образа из каждого сервиса, подключить RabbitMQ для общения с сервисом логов и развернуть все сервисы в docker-compose.