САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 4
«Docker, docker compose»

Выполнила:
Коник А. А.
Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения.

Ход работы

Пропишем докерфайлы для микросервисов и gateway:

```
FROM node:19.8.1-alpine

WORKDIR /events_microservice

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 9000

CMD [ "npm", "start" ]
```

```
FROM node:19.8.1-alpine

WORKDIR /users_microservice

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 9001

CMD [ "npm", "start" ]
```

```
FROM node:19.8.1-alpine

WORKDIR /gateway

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8000

CMD [ "npm", "start" ]
```

Обеспечим сетевое взаимодействие между частями приложения с помощью docker compose, в docker-compose.yml:

```
version: '3.9'

services:
  gateway:
    container_name: gateway
    build:
      context: ./gateway
    volumes:
      - ./gateway:/gateway
      - /gateway/node_modules
    depends_on:
      - events_microservice
      - users_microservice
    ports:
      - "127.0.0.1:8000:8000"
    restart: always
```

```yaml
events_microservice:
  container_name: events_microservice
  build:
    context: ./events_microservice
  volumes:
    - ./events_microservice:/events_microservice
    - /events_microservice/node_modules
  ports:
    - "9000"
  restart: always

users_microservice:
  container_name: users_microservice
  build:
    context: ./users_microservice
  volumes:
    - ./users_microservice:/users_microservice
    - /users_microservice/node_modules
  ports:
    - "9091"
  restart: always
```

Сбилдим образы с помощью команды docker-compose build и запустим их для создания контейнеров. Порт контейнера gateway забиндим с localhost, чтобы через него осуществлялась связь с клиентом.

Образы и контейнеры:

Name	Tag	Status
lr_4-gateway d9542c5da444	latest	In use
lr_4-users_microservice 7c9f022cc2f6	latest	In use
lr_4-events_microservice 00eb4533a4a3	latest	In use

lr_4	-	Running (3/3)	
elated_cori 60f50d652397	lr_4-gateway	Running	8000:8000
ecstatic_benz 0ed3aa189cc5	lr_4-users_microservice:latest	Running	
charming_borg 8f54312ec9ff	lr_4-events_microservice:latest	Running	

```
2023-05-30 14:48:55 > gateway@1.0.0 prestart
2023-05-30 14:48:55 > npm run build
2023-05-30 14:48:55
2023-05-30 14:48:55
2023-05-30 14:48:55 > gateway@1.0.0 build
2023-05-30 14:48:55 > npx tsc
2023-05-30 14:48:55
2023-05-30 14:48:59
2023-05-30 14:48:59 > gateway@1.0.0 start
2023-05-30 14:48:59 > nodemon dist/index.js
2023-05-30 14:48:59
2023-05-30 14:48:59 [nodemon] 2.0.22
2023-05-30 14:48:59 [nodemon] to restart at any time, enter `rs`
2023-05-30 14:48:59 [nodemon] watching path(s): src/**/*
2023-05-30 14:48:59 [nodemon] watching extensions: ts
2023-05-30 14:48:59 [nodemon] starting `ts-node ./src/index.ts dist/index.js`
2023-05-30 14:49:00 Running gateway on port 8000
```

```
2023-05-30 14:46:11 Running server on port 9001
2023-05-30 14:46:11 Executing (default): SELECT name FROM sqlite_m
2023-05-30 14:46:11 Executing (default): SELECT 1+1 AS result
2023-05-30 14:46:11 Executing (default): CREATE TABLE IF NOT EXIST
, `email` VARCHAR(255) UNIQUE, `password` VARCHAR(255) NOT NULL, `
2023-05-30 14:46:11 Connection has been established successfully.
2023-05-30 14:46:11 Executing (default): PRAGMA INDEX_LIST(`Users`
2023-05-30 14:46:11 Executing (default): PRAGMA INDEX_INFO(`sqlite
2023-05-30 14:46:11 Executing (default): SELECT name FROM sqlite_m
2023-05-30 14:46:11 Executing (default): CREATE TABLE IF NOT EXIST
, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL);
2023-05-30 14:46:11 Executing (default): PRAGMA INDEX_LIST(`Refres
2023-05-30 14:46:11 Executing (default): PRAGMA INDEX_INFO(`sqlite
2023-05-30 14:46:11 synced models
```

```
2023-05-30 14:46:33 Running server on port 9000
2023-05-30 14:46:33 Executing (default): SELECT name FROM sqlite_master WHERE type='tabl
2023-05-30 14:46:33 Executing (default): SELECT 1+1 AS result
2023-05-30 14:46:33 Executing (default): CREATE TABLE IF NOT EXISTS `Events` (`id` INTEG
NOT NULL, `district` VARCHAR(255) NOT NULL, `ev_type` VARCHAR(255) NOT NULL, `date` VARC
5) NOT NULL, `website` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME
2023-05-30 14:46:33 Connection has been established successfully.
2023-05-30 14:46:33 Executing (default): PRAGMA INDEX_LIST(`Events`)
2023-05-30 14:46:33 Executing (default): PRAGMA INDEX_INFO(`sqlite_autoindex_Events_1`)
2023-05-30 14:46:33 Executing (default): SELECT name FROM sqlite_master WHERE type='tabl
2023-05-30 14:46:33 Executing (default): CREATE TABLE IF NOT EXISTS `UserEnrolledEvents`
DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL);
2023-05-30 14:46:33 Executing (default): PRAGMA INDEX_LIST(`UserEnrolledEvents`)
2023-05-30 14:46:33 synced models
```

Вывод

В ходе работы я познакомилась с развертыванием приложения в докере с использованием docker compose и упаковала в docker-контейнеры свое приложение.