САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 4
“Docker, docker compose”

Выполнил:

Коротин А.М.

K33392

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.

Ход работы

Для выполнения лабораторной работы необходимо выполнить следующие задачи:

1) Написание Dockerfile для каждого сервиса для их контейнеризации,
2) Написание общего docker-compose файла, включающего все необходимые сервисы,
3) Обеспечение сетевого взаимодействия между сервисами, объявленными в пункте 2.

Начнем с 1 пункта. Dockerfile для обоих сервисов будут аналогичными, их можно увидеть на рисунке 1.

```
FROM node:alpine
USER root
LABEL authors="Alexey Korotin K33392"
WORKDIR /app
COPY package.json package-lock.json /app/

RUN npm install
RUN mkdir "/app/src"
COPY src /app/src
COPY tsconfig.json .eslintrc.json .env /app/

RUN npm run build

ENTRYPOINT ["npm", "run", "runApp"]
```

Рисунок 1 — Dockerfile

В качестве entrypoint используется npm-скрипт runApp, он запускает собранное ранее приложение с использованием .env файла с переменными окружения. Скрипты можно увидеть на рисунке 2.

```
"scripts": {
  "start": "npx tsc && node --env-file ./.env build/index.js",
  "build": "npx tsc",
  "runApp": "node --env-file ./.env build/index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
}
```

Рисунок 2 — Npm-скрипты сервиса

Далее приступим к файлу docker-compose. Он будет содержать в себе 3 сервиса:

— сервис СУБД PostgreSQL,

— сервис аутентификации,

— основной сервис приложения (smart_devices).

Сервисы приложения зависят от сервиса СУБД (если точнее — от готовности БД принимать подключения). В связи с этим сделаем healthcheck для сервиса СУБД (рисунок 3).

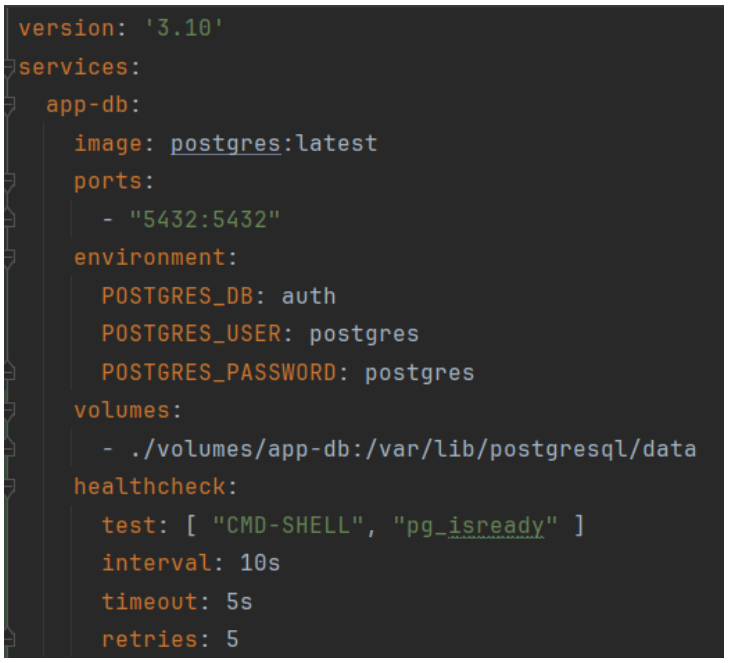

Рисунок 3 — Сервис СУБД PostgreSQL

Код для остальных сервисов приложения приведен на рисунках 4 и 5.

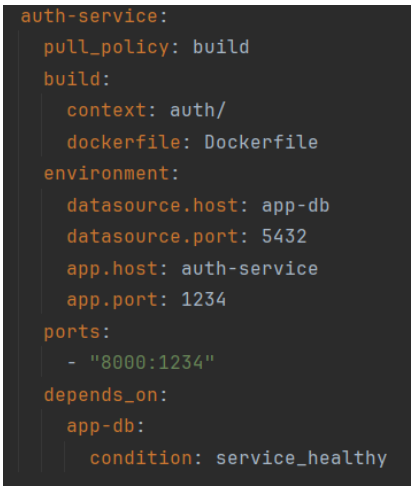

Рисунок 4 — Сервис аутентификации

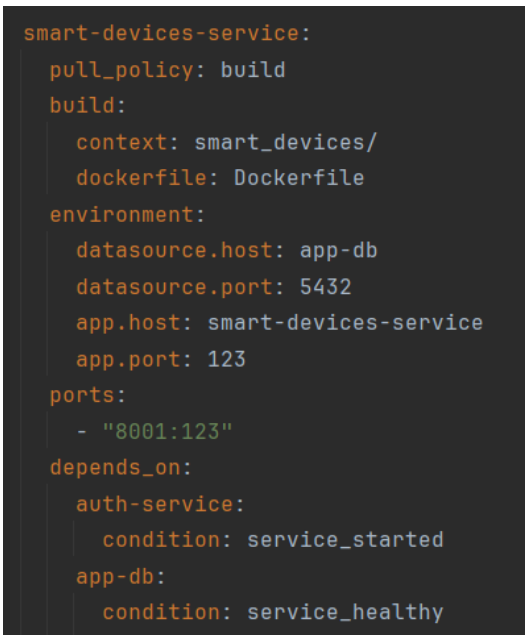

Рисунок 5 — Основной сервис smart_devices

Стоит отметить, что при каждом запуске системы будет производиться сборка docker-образов сервисов приложения с использованием ранее написанных Dockerfile’ов.

Далее перейдем к обеспечению сетевого взаимодействия между сервисами. Оно уже обеспечено в приведенных выше объявлениях сервисов, сейчас будет приведено пояснение.

В качестве хостов сервисов используется доменные имена во внутренней сети docker (docker internal network). Данная информация передается в приложения через переменные окружения.

Таким же образом происходит и подключение к БД — в качестве хоста указывается доменное имя во внутренней сети docker. При этом проброс порта для подключения не требуется.

Далее проверим работу реализованного сетевого взаимодействия. Выполним команду docker compose up -d для развертывания системы сервисов (рисунок 6).

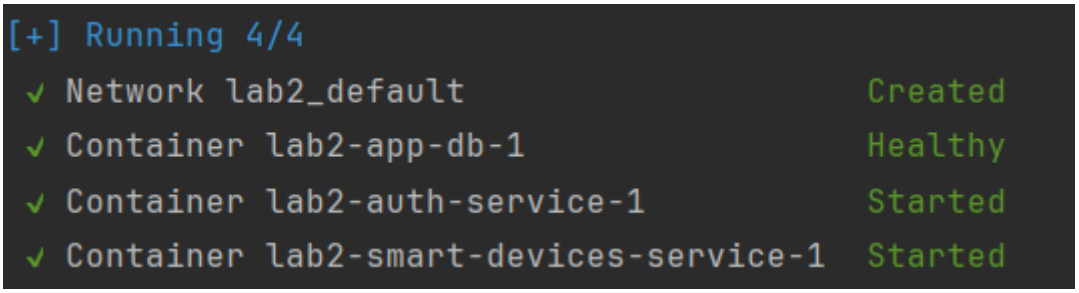

Рисунок 6 — Результат развертывания приложения

На рисунках 4 и 5 можем видеть, что auth-service проброшен на порт 8000, а smart-devices-service на порт 8001 хост-машины. Попробуем получить доступ к swagger-документации сервисов по адресам localhost:8000 и localhost:8001 (рисунки 7 и 8).

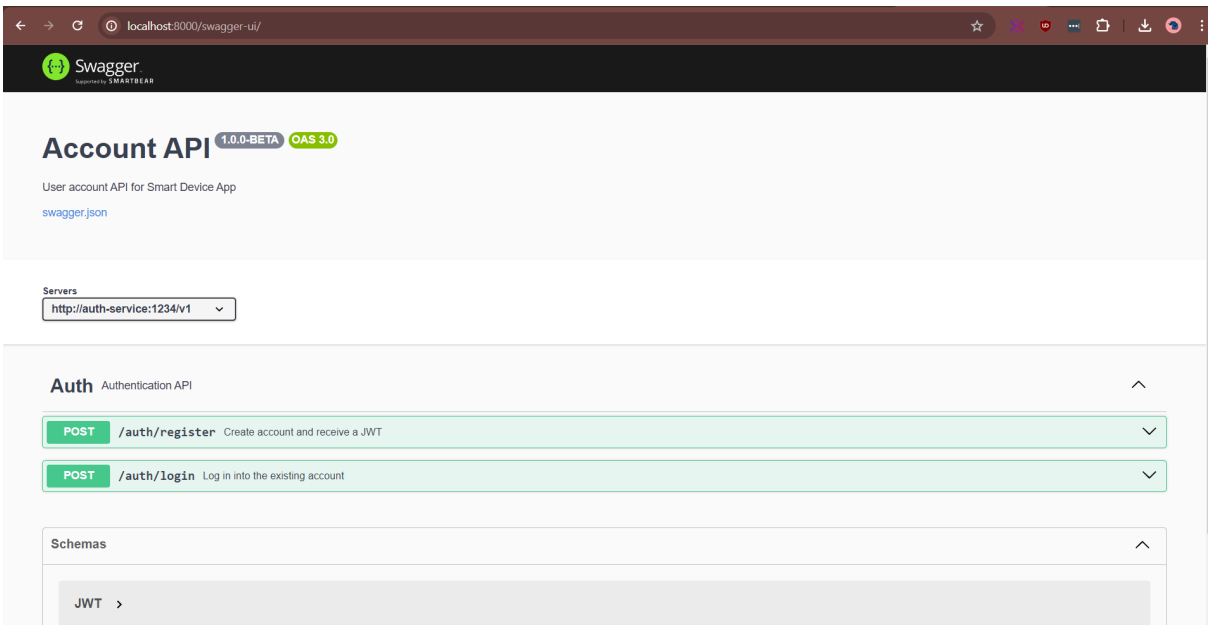

Рисунок 7 — Swagger-документация auth-service

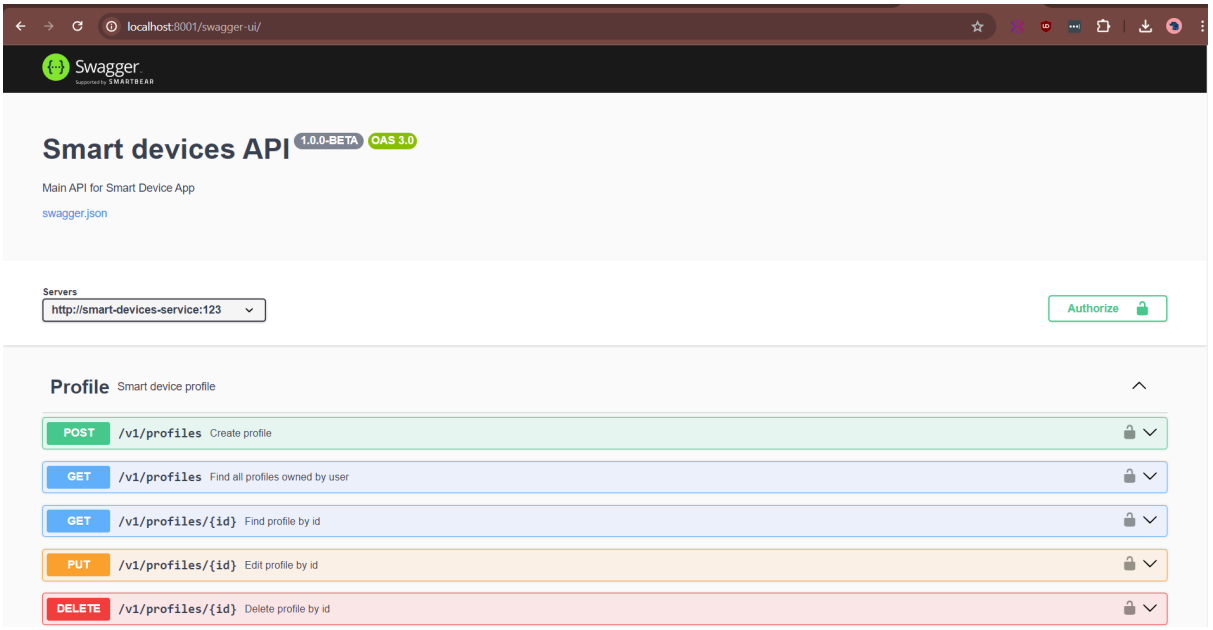

Рисунок 8 — Swagger-документация smart-devices-service

Вывод

В ходе выполнения лабораторной работы были изучены основы контейнеризации и сетевого взаимодействия контейнеров средствами docker и docker compose.

Также была произведена контейнеризация ранее написанного приложения, использующего микросервисную архитектуру.