Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики управления и технологий

Ли Александр Андреевич БД-241м

Практическая работа 3-2. Docker-compose

Направление подготовки/специальность
38.04.05 - Бизнес-информатика
Бизнес-аналитика и большие данные
(очная форма обучения)
Вариант 12

Москва
2024

Введение

Практическая работа нацелена на знакомство студентов с основами работы в Linux, установку системы, проведение предварительной настройки системы и настройка SSH на Ubuntu 24.

Цель

Целью практической работы заключается в создании проекта с использованием Docker-compose для взаимодействия с несколькими контейнерами, что позволяет создать среду, в которой три отдельных сервиса (init , app ,db) взаимодействуют друг с другом, что упрощает их развертывание, настройку и управление.

Init – используется для обновления базы данных до последней версии.
Db – база данных
App – создает api приложение

Взаимодействие

1.Сначала запускается база данных

2.После того, как запустилась bd, активируется init , задачей которого является обновление и настройка структуры базы данных, чтобы она была готова для использования основным приложением.

3.После всего вышесказанного, запускается app, который взаимодействует с базой данных, получая запрос от пользователей и разворачивает страницу через fast api.

1.bd →init (настраивает db)→app (взаимодействует с db)

2.Конфигурационные файлы:

docker-compose.yml

```
version: '3'
services:
  init:
    container_name: init
    build:
      context: ./server
      dockerfile: Dockerfile
    env_file: .env
```

```
  environment:
    - PYTHONUNBUFFERED=0
    - PYTHONPATH=/app/server
    - ENVIRONMENT=production
  command:
    - sh
    - -c
    - 'alembic upgrade head'
  depends_on:
    - db

server:
  container_name: server

  build:
    context: ./server
    dockerfile: Dockerfile
  env_file: .env
  environment:
    - PYTHONUNBUFFERED=0
    - PYTHONPATH=/app/server
    - ENVIRONMENT=production
    - SERVER_PORT=${SERVER_PORT}
     - CLIENT_PORT=${CLIENT_PORT}

  restart: always
  command:
    - sh
    - -c
    - 'uvicorn server.src.main:app --host 0.0.0.0 --port ${SERVER_PORT:-8000}'
  ports:
    - '8000:${SERVER_PORT:-8000}'
  volumes:
    - ./server:/app/server
  depends_on:
    - db
    - init
  healthcheck:
```

```
    test: ["CMD-SHELL", "curl -f http://localhost:${SERVER_PORT:-8000}/health || exit 1"]
    interval: 30s
    timeout: 10s
    retries: 3
  networks:
    - my_network

db:
  container_name: db
  image: postgres
  restart: always
  ports:
    - '5432:5432'
  env_file: .env
  healthcheck:
    test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER:-postgres}"]
    interval: 30s
    timeout: 10s
    retries: 5
  networks:
    - my_network

networks:
  my_network:
    driver: bridge
```

.env.

```
APP_PORT=8080

DB_USER=user
DB_PASSWORD=pass
POSTGRES_PASSWORD=password
ACCESS_TOKEN_EXPIRE_MINUTES=15
REFRESH_TOKEN_EXPIRE_DAYS=7
SECRET_KEY=your_secret_key
POSTGRES_USER=postgres
POSTGRES_DB=your_database
POSTGRES_HOST=db
POSTGRES_PORT=5432
SERVER_PORT=8000
CLIENT_PORT=3000
```

3. Настроека Docker Compose.

1. Описание сервисов.

   -Init предназначен для подготовки работы базы данных. Выполняет миграцию командой alembic upgrate head, чтоб структура базы была актуальной

   - server основное приложение , которое разворачивает страницу через fast api и взаимодействие с базой данных
   - db управляет хранилищем данных на основе образа postgres

2. Жесткое именование контейнеров.

   Каждый контейнер имеет жесткое именование , заданное с помощью **container_name**

   container_name: db
   container_name: server
   container_name: init

3. Применение depends_on, volume, прокидка порта наружу, command и/или entrypoint.
   -depends_on
   Так как init и server зависят от db, а server от init, необходимо использовать depends_on, чтобы один сервис должен запускаться после другого. Сделаем что, db запускалась перед init, а init завершал работу перед запуском server.

   - volume
   Используется для того, чтобы данные из локальной машины были доступны внутри контейнера. Volume монтирует локальную папку /server в контейнер server. Это удобно, так как любые изменения файлов в локальной папке сразу видны внутри контейнера без необходимости пересборки.

   -прокидка порта наружу
   В контейнере server пробрасываем порт наружу ('8000:${SERVER_PORT:-8000}'), для того чтобы, подключаться к серверу извне, используя порт 8000.

   -command

   В init с помощью команды alembic upgrade head выполняется миграция базы данных,а для server запускает выполнения приложения на FastAPI с помощью uvicorn

4. Описание healthcheck и сети.
   Healthcheck используется в server для того, чтобы проверять доступность сервера по URL
   http://localhost:${SERVER_PORT:-8000}/health.

   Все сервисы подключены к сети my_network, что дает возможность сервисам взаимодействовать друг с другом.

4.Процесс запуска.

Для запуска используется команда

Sudo docker-compose up –build

docker-compose запускает сервисы в порядке, указанном в docker-compose.yml. Сначала запускаются db, затем init, и server.

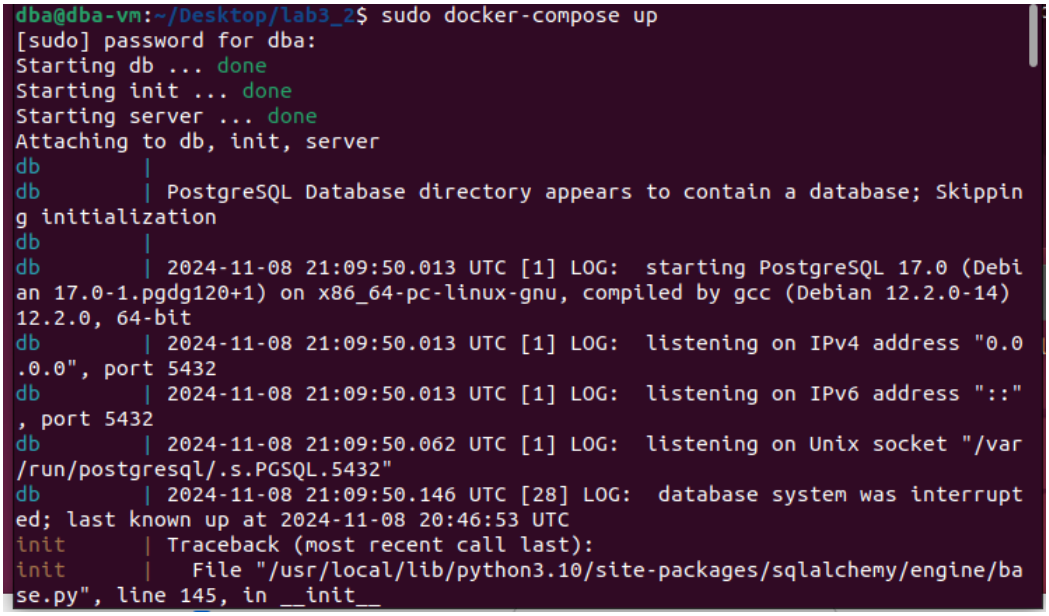

Рис.1 Запуск docker-compose

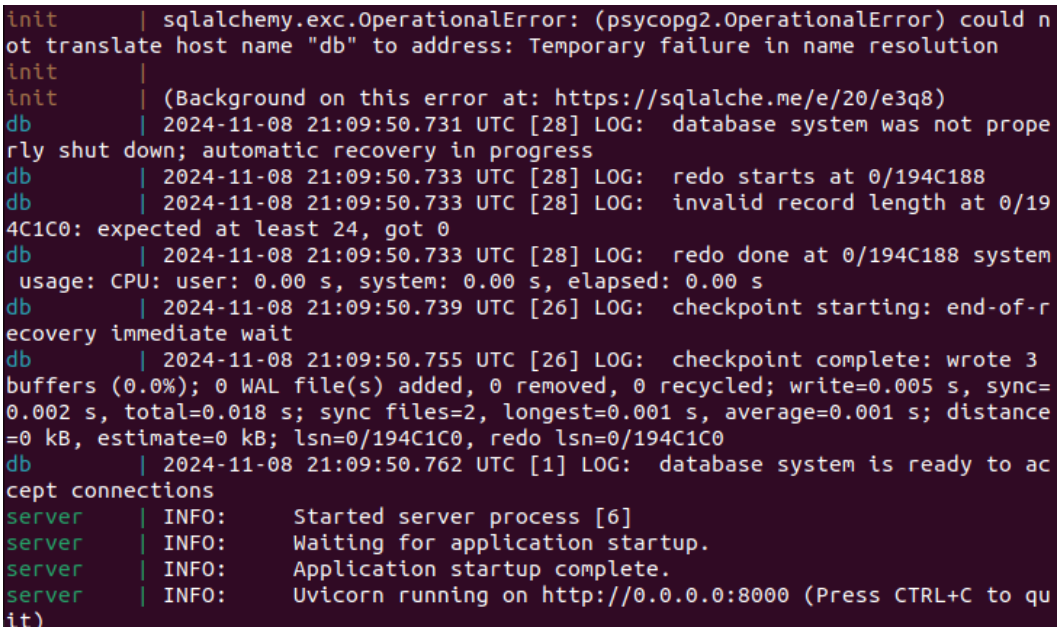

Рис.2 Запуск docker-compose

5.Результат работы.

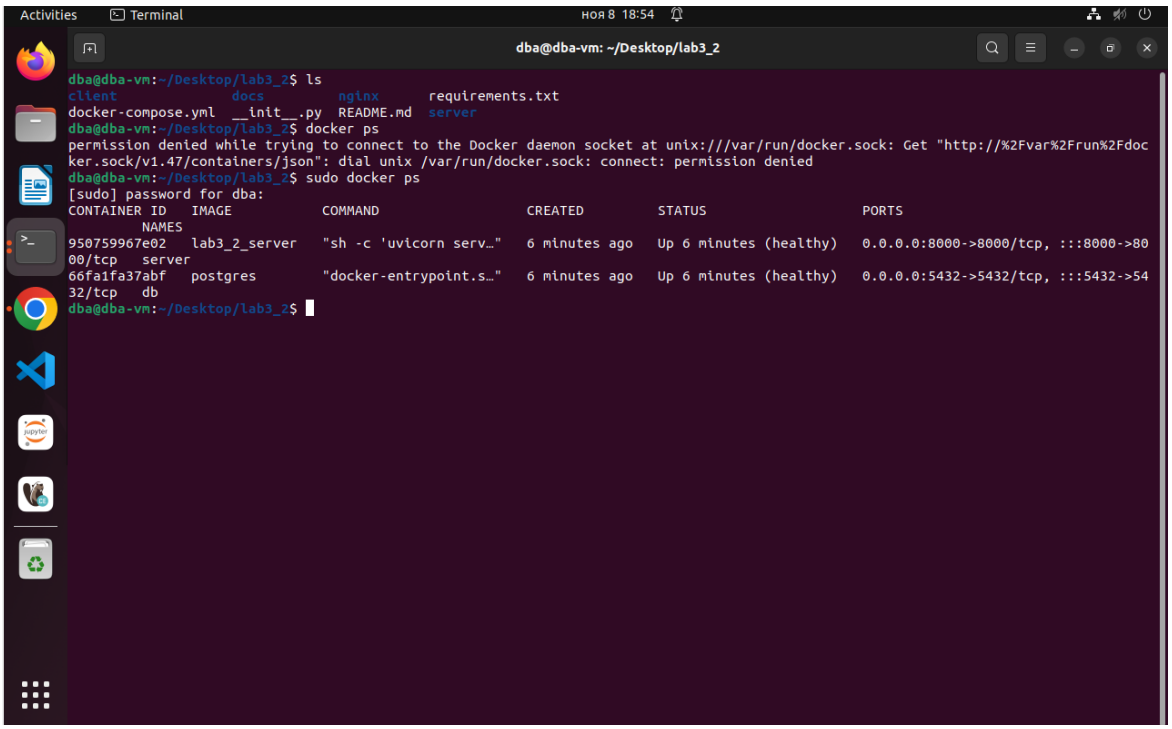

Рис.3 Проверка работы контейнеров командой

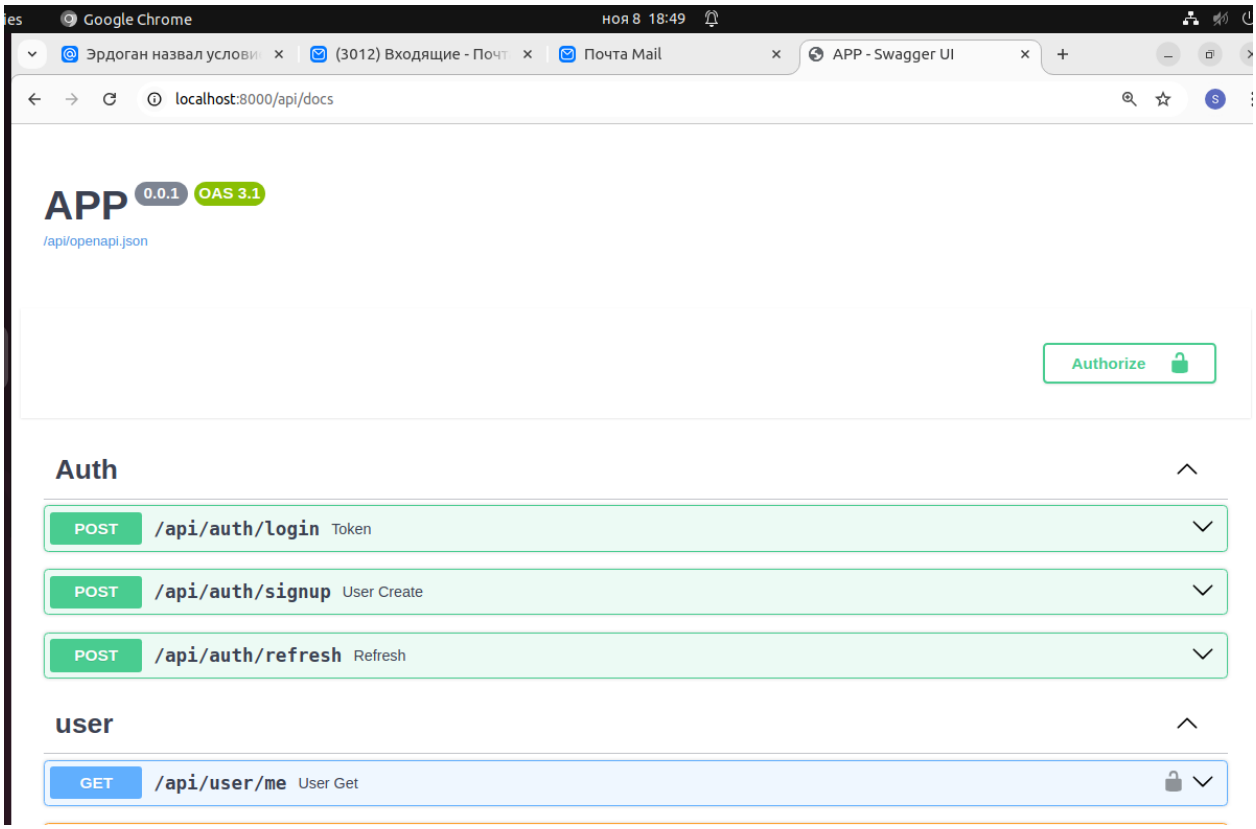

Рис.4 Проверка работы контейнеров в браузере

- Результаты тестирования healthcheck.

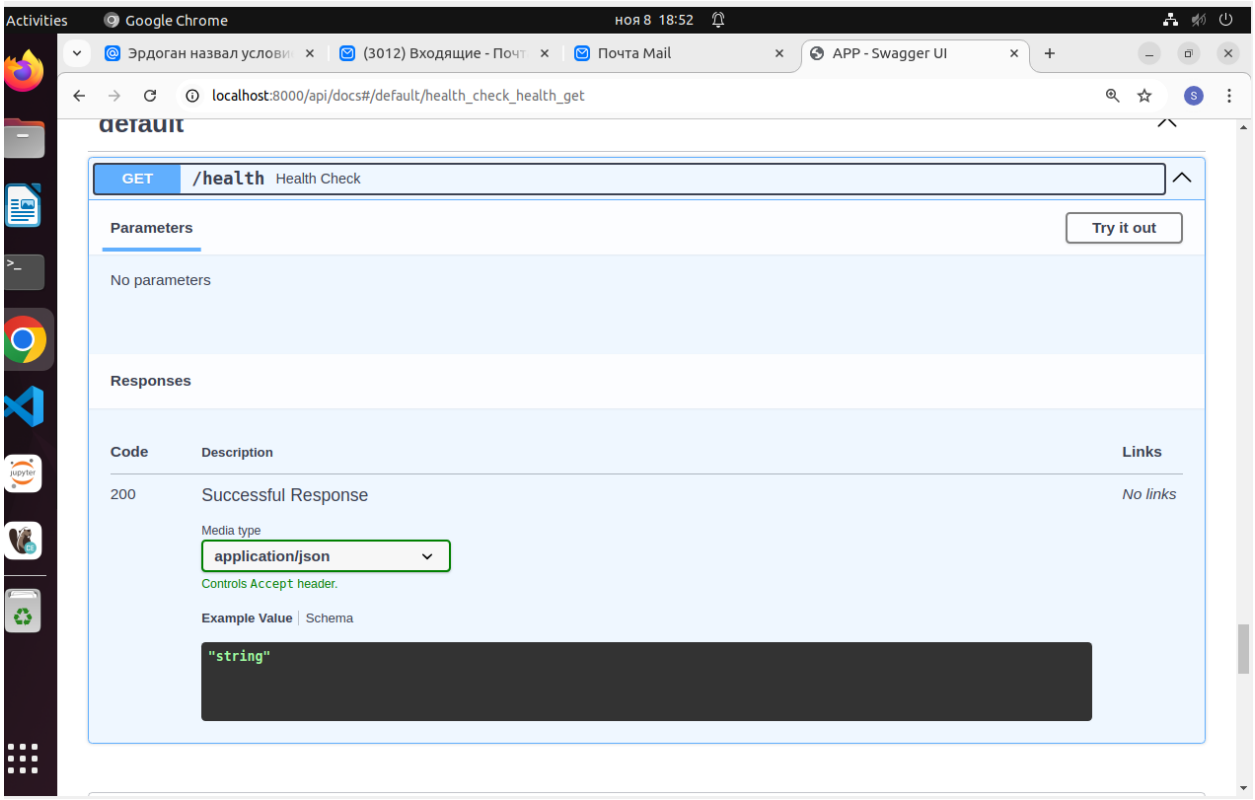

Рис.5 Результаты тестирования healthcheck

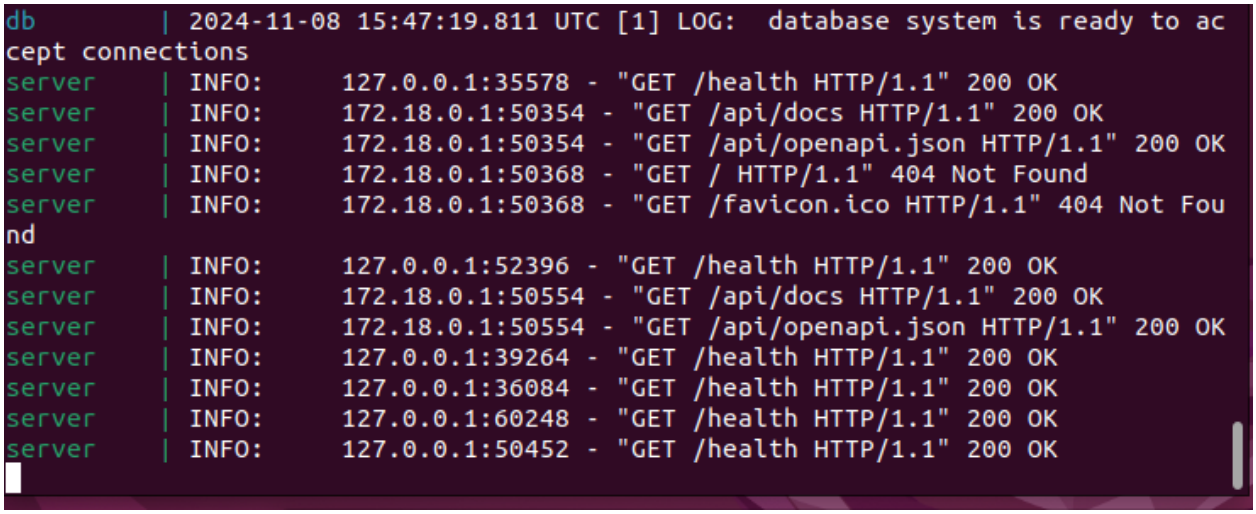

Рис.6

6.Ошибки и исправления.

В процессе запуска была ошибка связанная с postgres, не были заданы имя, пароль для postgress и другими параметрами. Чтобы решить проблему в .env и .yml были добавлены данные строчки

.yml в контейнер сервера

- SERVER_PORT=${SERVER_PORT}

- CLIENT_PORT=${CLIENT_PORT} Эти переменные задают порты для сервера и клиента

.env

POSTGRES_PASSWORD=password
Пароль для входа в базу данных PostgreSQL.

ACCESS_TOKEN_EXPIRE_MINUTES=15
Время действия токена доступа

REFRESH_TOKEN_EXPIRE_DAYS=7
Время действия токена обновления

SECRET_KEY=your_secret_key
Ключ для шифрования данных и токенов в приложении.

POSTGRES_USER=postgres
Имя пользователя для подключения к базе данных PostgreSQL

POSTGRES_DB=your_database
Название базы данных

POSTGRES_HOST=db
имя хоста для подключения к базе данных

POSTGRES_PORT=5432
Порт для подключения к базе данных PostgreSQL

SERVER_PORT=8000
Порт, на котором запускается сервер приложения

CLIENT_PORT=3000
Порт, на котором работает клиентское приложени

Заключение

В ходе выполнения практической работы, был создан docker-compose.yml файл с тремя сервисами, настроенны в соответствии с заданными требованиями.Что позволило понять, как работает Docker Compose для координации нескольких контейнеров. Также были встречены ошибки, и приведено их решение