

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

Ситьков Александр Вячеславович БД-241м

Практическая работа 3-2. Docker-compose

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

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

Создать docker-compose.yml файл с минимум тремя сервисами, настроенными в соответствии с заданными требованиями. Это позволит лучше понять, как работает Docker Compose для координации нескольких контейнеров.

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 возвращает 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
    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
access_token_expire_minutes=10
refresh_token_expire_days=10

```

```
secret_key=my_key
POSTGRES_PASSWORD=password
postgres_user=user
postgres_db=database
postgres_host=host
postgres_port=1234
server_port=8000
client_port=1111CLIENT_PORT=3000
```

3. Настройка Docker Compose.

Настройка Docker Compose

Этот файл конфигурации задаёт несколько сервисов.

Описание сервисов

`init` — сервис для подготовки базы данных, выполняющий миграции командой `alembic upgrade head` для актуализации структуры данных.

`server` — основной сервис, разворачивающий приложение FastAPI и организующий взаимодействие с базой данных.

`db` — сервис управления данными, использующий образ PostgreSQL.

Жёсткое именование контейнеров

У каждого контейнера задано фиксированное имя с помощью параметра `container_name`, например, `container_name: db`, `container_name: server`, и `container_name: init`.

Использование `depends_on`, `volume`, проброса порта и команд `command` / `entrypoint`

`depends_on`: Параметр `depends_on` обеспечивает последовательный запуск сервисов. Сначала запускается `db`, за ним `init`, и, после его завершения, — `server`.

`volume`: Volume монтирует папку `/server` с локальной машины в контейнер `server`, позволяя контейнеру видеть изменения в локальной папке без пересборки.

Проброс порта: Порт 8000 в контейнере `server` проброшен наружу (`8000:${SERVER_PORT:-8000}`), что даёт возможность подключения к серверу через этот порт.

`command`: В `init` выполняется миграция базы данных командой `alembic upgrade head`, а для `server` приложение запускается через FastAPI с помощью `uvicorn`.

Описание Healthcheck и сети

Healthcheck: В `server` используется для проверки доступности сервера по URL `http://localhost:${SERVER_PORT:-8000}/health`.

Сеть: Все сервисы подключены к `my_network`, что позволяет им взаимодействовать друг с другом.

4. Процесс запуска.

Для запуска используется команда

`sudo docker-compose up --build`

Сначала запускаются `db`, затем `init`, и `server`.

```
alex@alex-VirtualBox:~/lab3_2$ sudo docker-compose up --build
[sudo] password for alex:
Building init
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 118.3kB
Step 1/6 : FROM python:3.10
--> 9ec1cdbafe6e
Step 2/6 : WORKDIR /app
--> Using cache
--> 226606a1120d
Step 3/6 : COPY ./requirements.txt /app/server/requirements.txt
--> Using cache
--> 7e98ab4ad483
Step 4/6 : RUN pip install --no-cache-dir --upgrade -r /app/server/requirements.txt
--> Using cache
--> b2c18fdf65d2
Step 5/6 : COPY ./alembic.ini /app/alembic.ini
--> Using cache
--> 1bb4f516fc05
Step 6/6 : COPY ./src /app/server/src
--> b0cb68e78f3f
Successfully built b0cb68e78f3f
Successfully tagged lab3_2_init:latest
Building server
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/
```

Рисунок 1.1 Запуск docker-compose

```
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 118.3kB
Step 1/6 : FROM python:3.10
--> 9ec1cdbafe6e
Step 2/6 : WORKDIR /app
--> Using cache
--> 226606a1120d
Step 3/6 : COPY ./requirements.txt /app/server/requirements.txt
--> Using cache
--> 7e98ab4ad483
Step 4/6 : RUN pip install --no-cache-dir --upgrade -r /app/server/requirements.txt
--> Using cache
--> b2c18fdf65d2
Step 5/6 : COPY ./alembic.ini /app/alembic.ini
--> Using cache
--> 1bb4f516fc05
Step 6/6 : COPY ./src /app/server/src
--> Using cache
--> b0cb68e78f3f
Successfully built b0cb68e78f3f
Successfully tagged lab3_2_server:latest
db is up-to-date
Recreating init ... done
Recreating server ... done
Attaching to db, init, server
init      | Traceback (most recent call last):
init      |   File "/usr/local/lib/python3.10/site-packages/sqlalchemy/engine/base.py", line 145, in __init__
init      |     self._dbapi_connection = engine.raw_connection()
init      |   File "/usr/local/lib/python3.10/site-packages/sqlalchemy/engine/base.py", line 3292, in raw_connection
init      |     return self.pool.connect()
```

Рисунок 1.2 Запуск docker-compose

5. Результат работы.

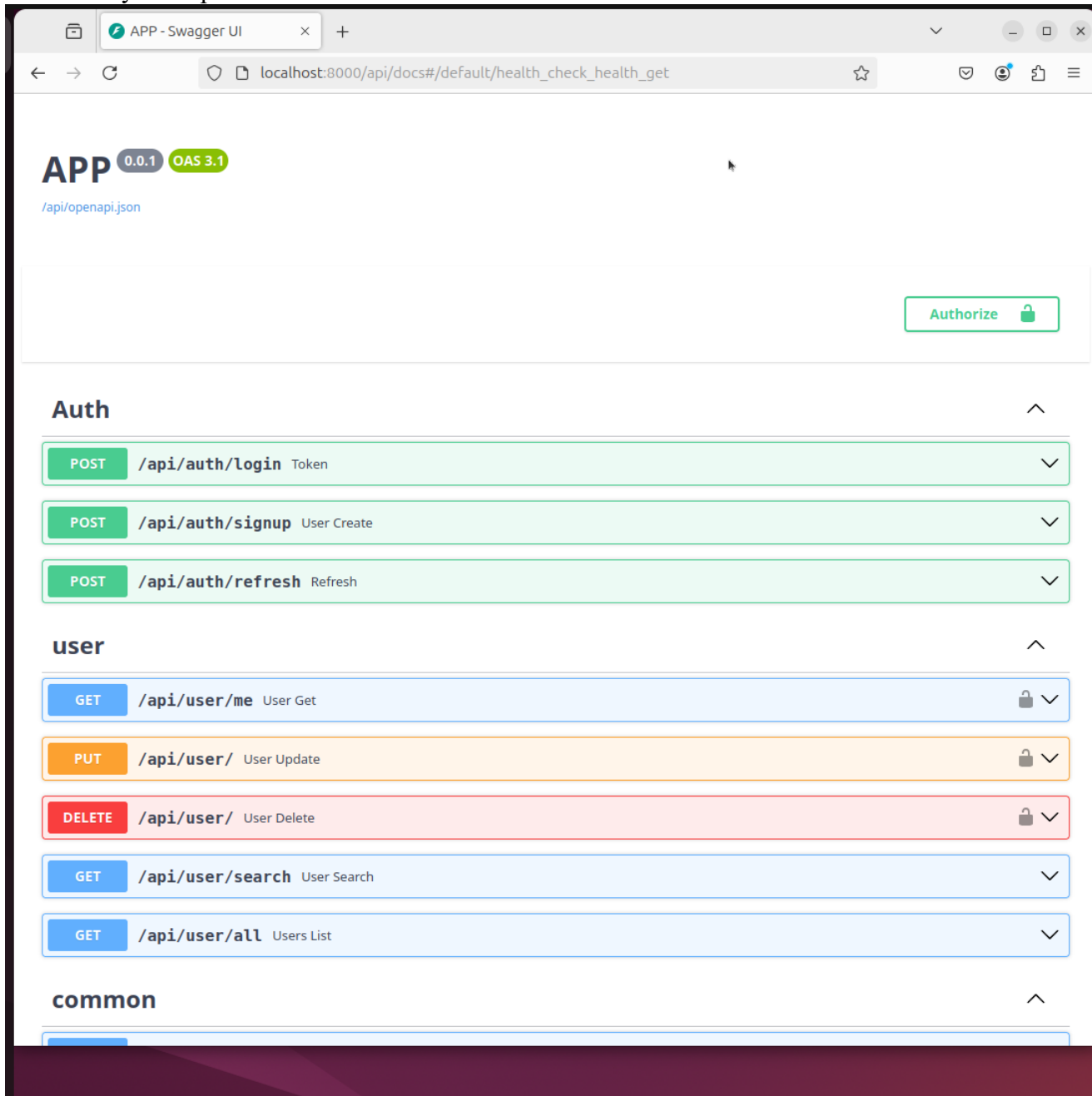


Рисунок 1.3 Проверка работы контейнеров в браузере

Результаты тестирования healthcheck.

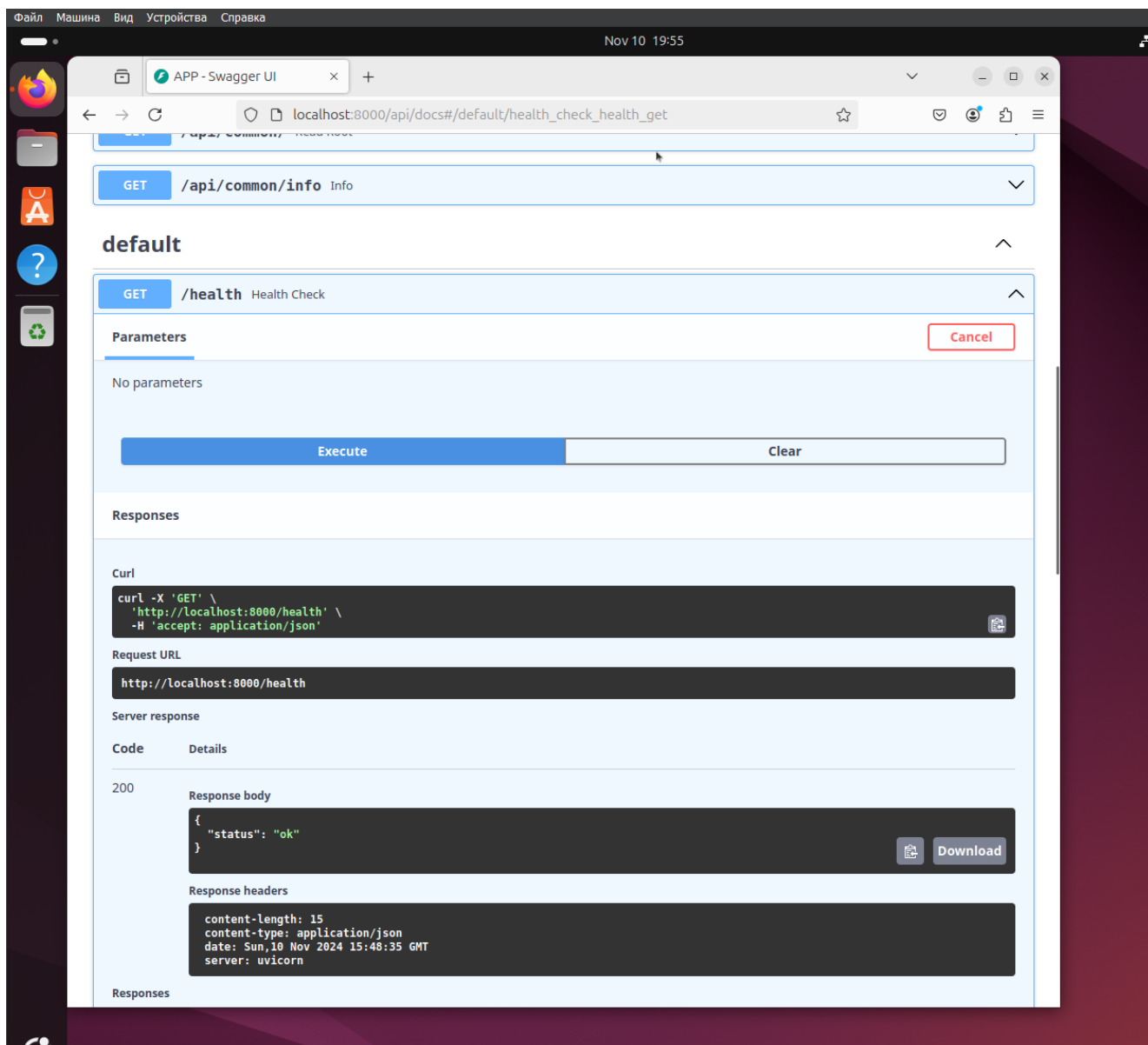


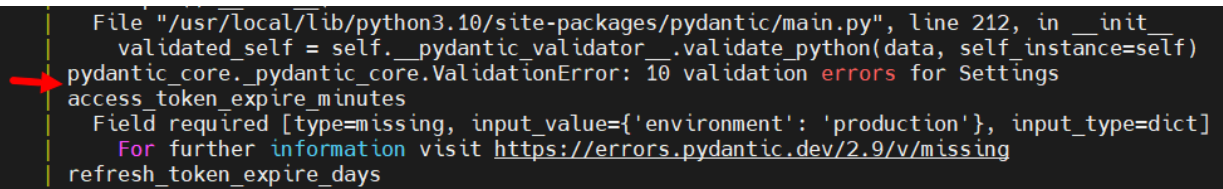
Рисунок 1.4 Результаты тестирования healthcheck

6. Ошибки и исправления.

При запуске возникал ряд ошибок, а именно нет необходимых переменных в файле. env, а именно

- access_token_expire_minutes
- refresh_token_expire_days
- secret_key
- POSTGRES_PASSWORD
- postgres_user
- postgres_db
- postgres_host
- postgres_port
- server_port
- client_port

В терминале это выглядело следующим образом:



```
File "/usr/local/lib/python3.10/site-packages/pydantic/main.py", line 212, in __init__
    validated_self = self.__pydantic_validator__.validate_python(data, self_instance=self)
pydantic_core._pydantic_core.ValidationError: 10 validation errors for Settings
  access_token_expire_minutes
    Field required [type=missing, input_value={'environment': 'production'}, input_type=dict]
  For further information visit https://errors.pydantic.dev/2.9/v/missing
  refresh_token_expire_days
```

Рисунок 1.5 – Вывод ошибок при запуске

После заполнения этих полей в файле .env все заработало.

Заключение

В ходе выполнения практической работы, был создан uml файл с тремя сервисами, которые были настроены в соответствии с заданными требованиями. Это позволило понять, как работает Docker Compose для координации нескольких контейнеров. А также были исправлены возникшие ошибки.