

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 4

Выполнил:

Котовщиков Андрей

К3339

Проверил:

Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Научиться работать с инструментами для контейнеризации.

- реализовать Dockerfile для каждого сервиса;
- написать общий docker-compose.yml;
- настроить сетевое взаимодействие между сервисами.

Ход работы

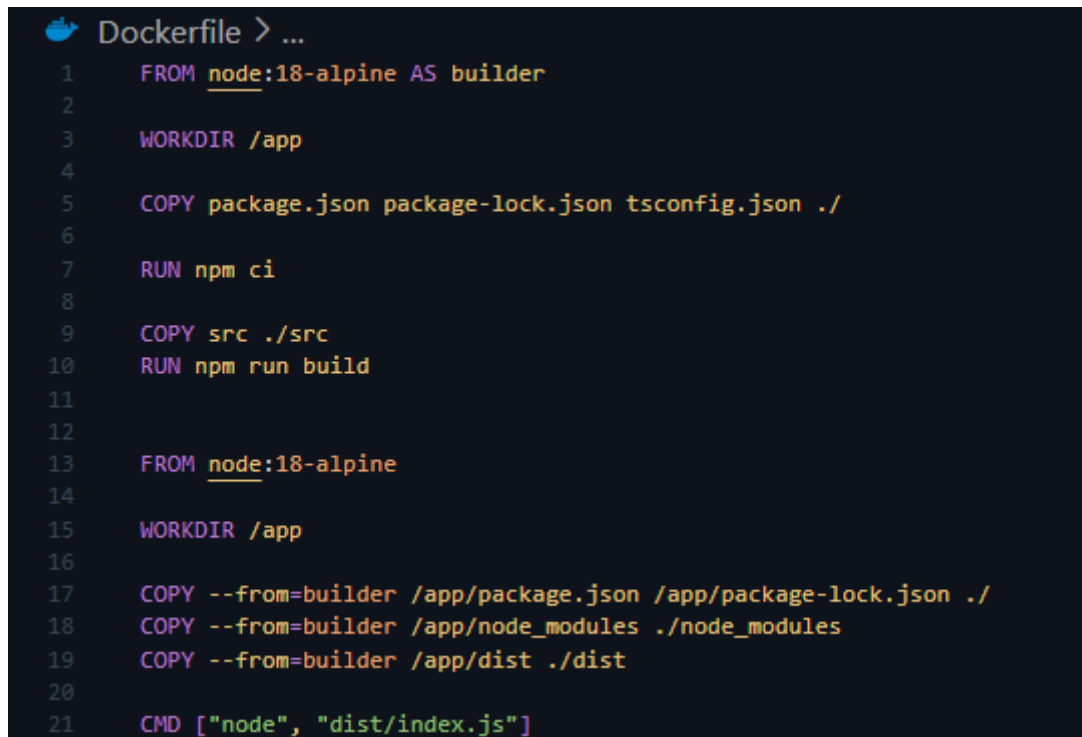
1. Создание Dockerfile

Для создания образа Python сервиса с фильмами был написан Dockerfile с использованием multi-stage сборки с целью минимизации размера итогового образа (рисунок 1). Далее из этого образа будет запущен контейнер с сервисом. Dockerfile для сервиса авторизации полностью идентичен.

```
Dockerfile > ...
1  FROM python:3.11-slim AS build
2
3  WORKDIR /app
4
5  COPY pyproject.toml uv.lock /app/
6
7  RUN apt-get update && \
8      pip install --upgrade pip && \
9      pip install --upgrade uv && \
10     pip install --upgrade wheel && \
11     uv pip compile pyproject.toml -o requirements.txt && \
12     pip wheel \
13     --no-deps \
14     --no-cache-dir \
15     --wheel-dir /app/wheels \
16     -r requirements.txt
17
18
19 FROM python:3.11-slim
20
21 WORKDIR /app
22
23 COPY --from=build /app/requirements.txt .
24 COPY --from=build /app/wheels /app/wheels
25
26 COPY ./src /app/src
27 COPY ./alembic.ini .
28
29 ENV PYTHONPATH=/app/src
30
31 RUN pip install \
32     --no-deps \
33     --no-cache-dir \
34     --no-index \
35     --find-links=/app/wheels \
36     -r requirements.txt
37
38 CMD [ "uvicorn", "src.main:app", "--host", "0.0.0.0" ]
```

Рисунок 1 – Dockerfile для сервиса фильмов

Для сервиса отправки писем аналогично был создан Dockerfile, но предназначенный для сборки приложения на Node.js вместо Python (рисунок 2).

A screenshot of a code editor showing a Dockerfile. The file is titled 'Dockerfile > ...'. It contains 21 lines of code, numbered 1 to 21. The code is written in a dark theme with syntax highlighting. The Dockerfile defines a multi-stage build. Stage 1 (builder) starts with 'FROM node:18-alpine AS builder', sets 'WORKDIR /app', copies 'package.json', 'package-lock.json', and 'tsconfig.json' from the current directory, runs 'npm ci', copies 'src' from the current directory, and runs 'npm run build'. Stage 2 (runtime) starts with 'FROM node:18-alpine', sets 'WORKDIR /app', copies files from the builder stage: 'package.json', 'package-lock.json', 'node_modules', and 'dist', and finally runs 'CMD ["node", "dist/index.js"]'.

```
1 FROM node:18-alpine AS builder
2
3 WORKDIR /app
4
5 COPY package.json package-lock.json tsconfig.json ./
6
7 RUN npm ci
8
9 COPY src ./src
10 RUN npm run build
11
12
13 FROM node:18-alpine
14
15 WORKDIR /app
16
17 COPY --from=builder /app/package.json /app/package-lock.json ./
18 COPY --from=builder /app/node_modules ./node_modules
19 COPY --from=builder /app/dist ./dist
20
21 CMD ["node", "dist/index.js"]
```

Рисунок 2 – Dockerfile для сервиса отправки писем

2. Создание единого docker-compose файла

Для удобства развертывания каждого сервиса на сервере (или локально) был написан единый docker-compose.yml файл со всей необходимой инфраструктурой (рисунок 3 и 4).

```

1  ---
2  D>Run All Services
3  services:
4    D>Run Service
5    database:
6      image: postgres
7      container_name: database
8      restart: unless-stopped
9      ports:
10       - "5435:5432"
11      healthcheck:
12        test: ["CMD-SHELL", "pg_isready -U ${DB_USER} -d ${DB_NAME}"]
13        interval: 5s
14        timeout: 5s
15        retries: 5
16      environment:
17        - PGDATA=/var/lib/postgresql/data/postgres
18        - POSTGRES_USER=${DB_USER}
19        - POSTGRES_PASSWORD=${DB_PASSWORD}
20        - POSTGRES_DB=${DB_NAME}
21      volumes:
22        - database_data:/var/lib/postgresql/data/postgres
23        - ./dataset.csv:/app/dataset.csv
24      networks:
25        - backend_net
26
27  D>Run Service
28  api:
29    container_name: api
30    env_file: .env
31    restart: unless-stopped
32    command: bash -c "alembic upgrade head && uvicorn src.main:app --host 0.0.0.0 --port 8000"
33    build:
34      context: .
35      dockerfile: Dockerfile
36    ports:
37      - "8000:8000"
38    environment:
39      - DB_HOST=database
40      - DB_PORT=5432
41    volumes:
42      - ./src:/app/src # DEV ONLY
43    depends_on:
44      database:
45        condition: service_healthy
46    networks:
47      - backend_net
48      - mail_net
49
50  volumes:
51    database_data:
52
53  networks:
54    backend_net:
55      name: "backend_net"
56    mail_net:
57      external: true

```

Рисунок 3 – docker-compose файл для сервиса фильмов

```

1  services:
2    > Run Service
3    zookeeper:
4      image: confluentinc/cp-zookeeper:7.4.0
5      container_name: zookeeper
6      ports:
7        - "2181:2181"
8      environment:
9        ZOOKEEPER_CLIENT_PORT: 2181
10     networks:
11       - mail_net
12
13   > Run Service
14   kafka:
15     image: confluentinc/cp-kafka:7.4.0
16     container_name: kafka
17     healthcheck:
18       test: ["CMD", "bash", "-c", "echo > /dev/tcp/localhost/9092"]
19       interval: 5s
20       timeout: 5s
21       retries: 5
22     depends_on:
23       - zookeeper
24     ports:
25       - "9092:9092"
26     environment:
27       KAFKA_BROKER_ID: 1
28       KAFKA_ZOOKEEPER_CONNECT: "zookeeper:2181"
29       KAFKA_LISTENERS: "PLAINTEXT://0.0.0.0:9092"
30       KAFKA_ADVERTISED_LISTENERS: "PLAINTEXT://kafka:9092"
31       KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
32   networks:
33     - mail_net
34
35   > Run Service
36   mail-sender:
37     build:
38       context: .
39       dockerfile: Dockerfile
40     container_name: mail-sender
41     depends_on:
42       kafka:
43         condition: service_healthy
44     env_file:
45       - .env
46     restart: unless-stopped
47     networks:
48       - mail_net
49
50 networks:
51   mail_net:
52     name: "mail_net"

```

Рисунок 4 – docker-compose файл для сервиса отправки писем

3. Настройка сетевого взаимодействия между контейнерами

Для взаимодействия сервисов внутри сети docker была создана bridge сеть под названием mail_net. С ее помощью сервисы могут коммуницировать друг с другом.

Вывод

В ходе выполнения лабораторной работы 4 были изучены инструменты для контейнеризации приложений такие как Docker и docker-compose. Все сервисы теперь работают в изолированных контейнерах и общаются между собой при помощи внутренней сети docker.