

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет Инфокоммуникационных технологий

Образовательная программа 09.03.03 Мобильные и сетевые технологии

О Т Ч Е Т

по лабораторной работе № 4
по дисциплине «Бэк-энд разработка»

Выполнил:

Мосин З. И.

Проверил:

Добряков Д. И.



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2024

Цели работы

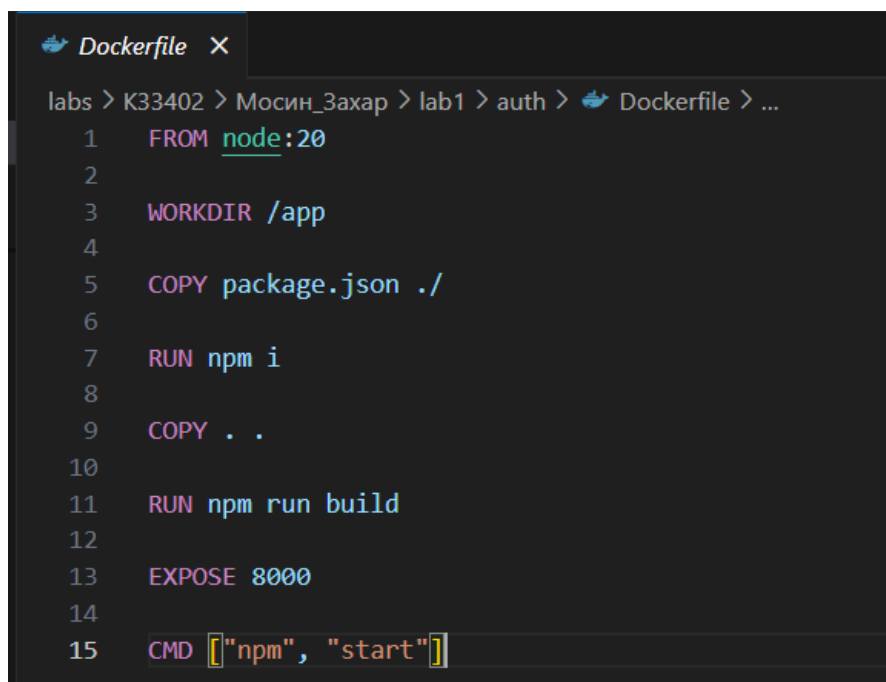
Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой посредством RabbitMQ.

Делать это можно как с помощью docker compose так и с помощью docker swarm.

Ход работы

Для данной работы необходимо было создать docker - контейнеры для каждой части нашего приложения. В файле docker-compose была описана работа со всеми частями приложения: контейнеры для основного сервера и микросервиса авторизации, образ PostgreSQL для работы с базой данных, образ для работы с RabbitMQ.

Имеющиеся контейнеры были описаны с помощью Dockerfile. При этом для предотвращения ошибок и улучшения производительности контейнеров были созданы. dockerignore файлы, описывающие директории, которые не нужно использовать при сборке контейнеров.



```
Dockerfile X
labs > K33402 > Мосин_Захар > lab1 > auth > Dockerfile > ...
1 FROM node:20
2
3 WORKDIR /app
4
5 COPY package.json ./
6
7 RUN npm i
8
9 COPY . .
10
11 RUN npm run build
12
13 EXPOSE 8000
14
15 CMD ["npm", "start"]
```

Содержимое docker compose:

```
version: '3.8'

services:
  auth:
    container_name: auth
    build:
      context: ./auth
      dockerfile: Dockerfile
    environment:
      - PORT=8000
      - DB_NAME=postgres
      - DB_USERNAME=postgres
      - DB_PASSWORD=1234
      - DB_HOST=postgresdb
    depends_on:
      - rabbitmq
      - postgresdb
    ports:
      - '8000:8000'
    networks:
      - mynetwork

  app:
    container_name: app
    build:
      context: ./app
      dockerfile: Dockerfile
    environment:
      - PORT=5000
      - AUTH_PORT=8000
      - DB_NAME=postgres
      - DB_USERNAME=postgres
      - DB_PASSWORD=1234
      - DB_HOST=postgresdb
      - HOST=auth
    depends_on:
      - rabbitmq
      - postgresdb
    networks:
      - mynetwork
    ports:
      - '5000:5000'

  rabbitmq:
    image: rabbitmq:3-management
```

```

ports:
  - '5672:5672'
  - '15672:15672'
networks:
  - mynetwork

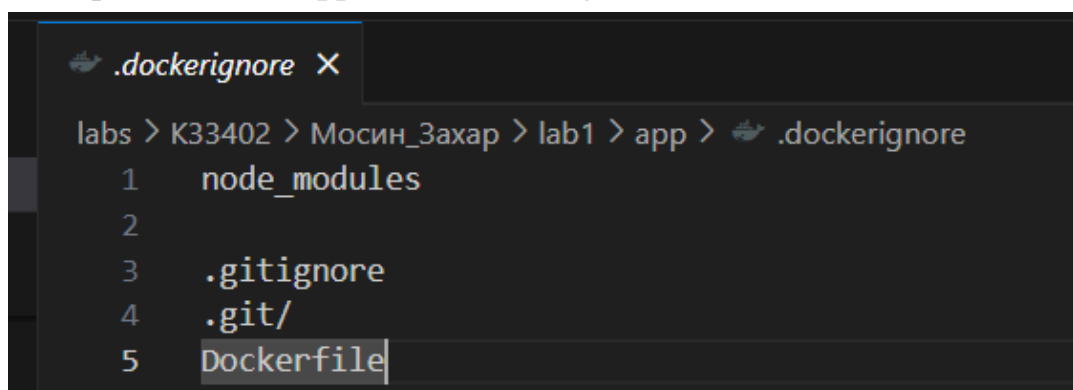
postgresdb:
  image: postgres:16
  restart: always
  environment:
    - POSTGRES_DB=postgres
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=1234
    - POSTGRES_HOST=postgresdb
  ports:
    - '5432:5432'
  volumes:
    - database_data:/var/lib/postgresql/data
  networks:
    - mynetwork

volumes:
  database_data:
    driver: local

networks:
  mynetwork:
    driver: bridge

```

Настроенные для app и auth dockerignore



The screenshot shows a file explorer window with a tab titled ".dockerignore". The breadcrumb path is "labs > K33402 > Мосин_Захар > lab1 > app". The file list contains five items, with "5 Dockerfile" selected:

- 1 node_modules
- 2
- 3 .gitignore
- 4 .git/
- 5 Dockerfile

Также в отдельной директории была настроена работа с RabbitMQ для более удобного взаимодействия с процессом авторизации и регистрации.

```
TS rabbitmq.ts X
labs > K33402 > Мосин_Захар > lab1 > auth > src > config > TS rabbitmq.ts > ...
1  import dotenv from 'dotenv'
2
3  import amqp from 'amqplib/callback_api'
4
5  dotenv.config()
6
7  const rabbitmqHost = process.env.RABBITMQ_HOST || 'localhost'
8
9  amqp.connect(`amqp://${rabbitmqHost}`, function (error0, connection) {
10     if (error0) {
11         throw error0
12     }
13     connection.createChannel(function (error1, channel) {
14         if (error1) {
15             throw error1
16         }
17
18         const queue = 'authorize'
19
20         channel.assertQueue(queue, {
21             durable: false,
22         })
23
24         channel.sendToQueue(queue, Buffer.from('готово'))
25         console.log(" [x] Sent 'готово'")
26     })
27 })
28
```

По итогу полученное приложение может быть запущено с помощью docker compose одной командой, а взаимодействие его частей регулируется с помощью RabbitMQ.

ВЫВОД

В процессе работы упростили запуск и сборку приложения с помощью docker и docker-compose, а также настроили взаимодействие отдельных частей приложения, с помощью RabbitMQ. Получили навыки работы с данными инструментами.