МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЁТ

по лабораторной работе 2

по дисциплине

Методология разработки DevOps

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Симанов В.С.

(подпись) (фамилия, и.,о.)

21-ИС\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2024

Цель: научиться работать с Docker и Docker-Compose.

Описание л.р.:

1)Создать новый репозиторий с названием DevOps.

2)Использовать версии 2.0 ранее разработанных микросервисов (frontend и backend) из лабораторной работы № 1.

3)Написать Dockerfile для каждого компонента — frontend и backend, который будет содержать инструкции для создания Docker-образа.

4)В репозитории DevOps создать файл docker-compose.yml, в котором реализовать следующую логику:

- Сборка образов с использованием названия и версии в соответствии с репозиторием микросервиса. Образы должны собираться автоматически на основе файлов

- Dockerfile для frontend’а и backend’а.

- Сохранения данных файла data.txt вне контейнера, чтобы данные сохранялись даже после перезагрузки контейнера.

- Создание отдельной сети для взаимодействия между контейнерами frontend’a и backend’а. Контейнеры должны общаться между собой через эту сеть без проброса портов наружу.

- У frontend’а должна быть зависимость от backend’а

- Контейнеры автоматически перезапускаются после перезагрузки сервера (ноды).

Результат выполнения:

1)В каждом репозитории: frontend и backend имеется свой Dockerfile.

2)В репозитории DevOps имеется docker-compose.yml, который обеспечивает сборку и запуск микросервисов.

3)Все микросервисы запускаются и корректно взаимодействуют друг с другом с использованием Docker-образов и Docker Compose.

4)Функциональная реализация для frontend’а и backend’а не нарушена.

ЯП: Java с использованием springboot. Frontend и backend уже написаны

Ход работы:

Создаем репозиторий DevOps: mkdir DevOps

Переходим в него: cd DevOps

Загружаем в него back и front: git clone <Ссылка на back или front по очереди> v3.0 (последняя версия репозиторий)

Добавляем в каждый репозиторий Dockerfile.

Содержимое Dockerfile для backend:

FROM openjdk:21-jdk-slim  
  
WORKDIR /app  
  
COPY target/backend-2.0.jar /app/backend.jar  
  
EXPOSE 8081  
  
ENTRYPOINT ["java", "-jar", "backend.jar"]

Используем образ 21 jdk

Устанавливаем рабочую директорию (/app) внутри контейнераа

Копируем приложения для backend’а

Открываем порт 8081

И запускаем

Содержимое Dockerfile для frontend:

FROM openjdk:21-jdk-slim  
  
WORKDIR /app  
  
COPY target/frontend-2.0.jar /app/frontend.jar  
  
EXPOSE 8080  
  
ENTRYPOINT ["java", "-jar", "frontend.jar"]

Для frontend’а описание такое же что и для backend с минимальными различиями что порт открывается на 8080

Далее в директорию DevOps добавляем docker-compose

Его содержимое:

version: '3'

services:

  frontend:

    build:

      context: ./frontend

      dockerfile: Dockerfile

    image: frontend

    container\_name: frontend

    depends\_on:

      - backend

    ports:

      - "8080:8080"

    networks:

      - app-network

    restart: always

  backend:

    build:

      context: ./backend

      dockerfile: Dockerfile

    image: backend

    container\_name: backend

    volumes:

      - ./data:/app/data

    ports:

      - "8081:8081"

    networks:

      - app-network

    restart: always

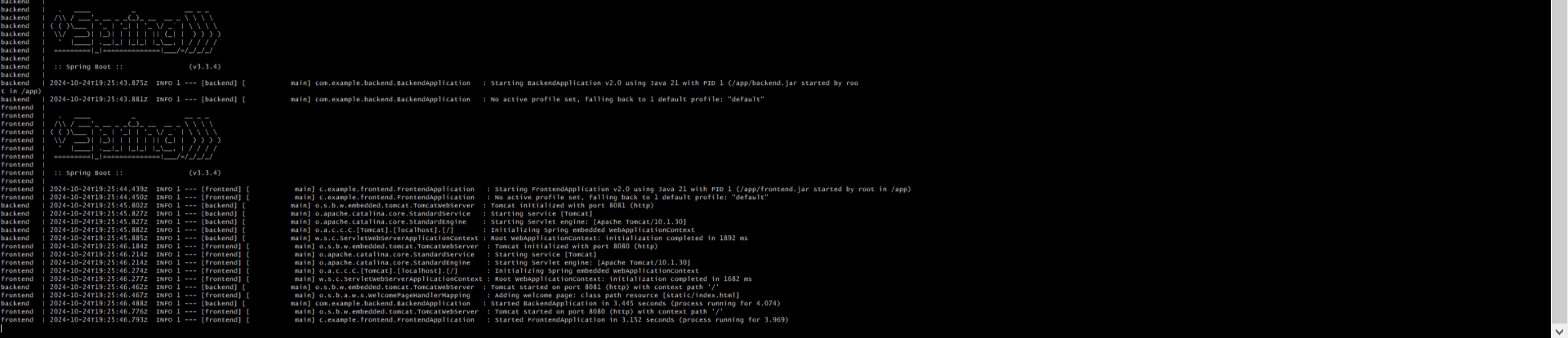
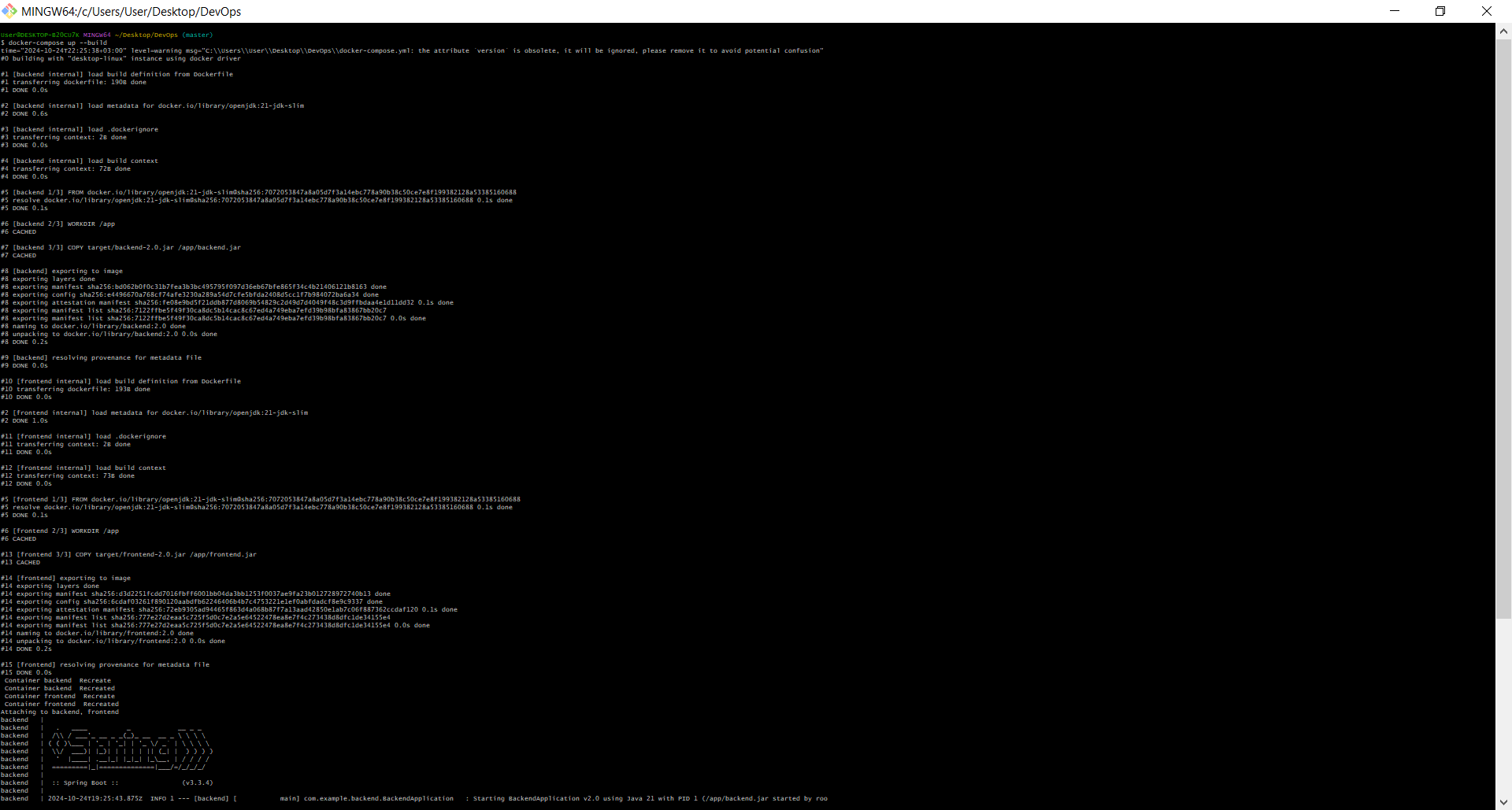
networks:

  app-network:

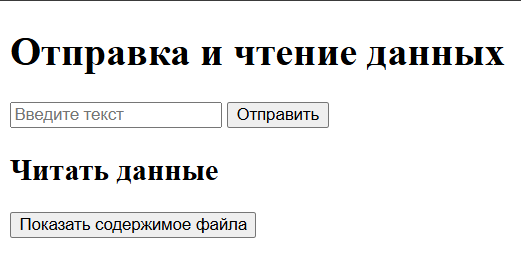
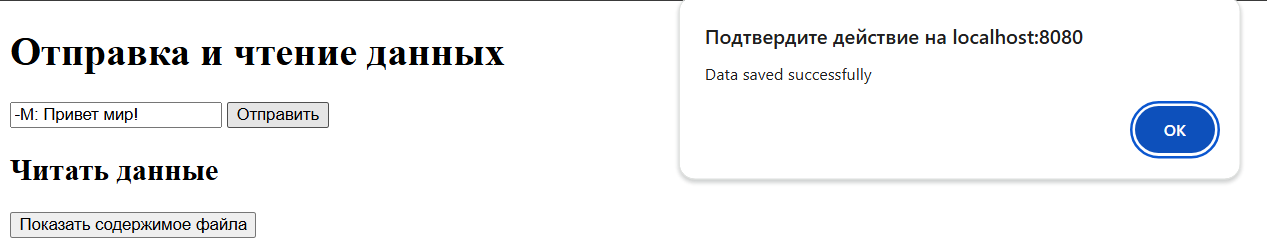
    driver: bridge

Далее собираем образы командой: docker-compose up –build

Gitbash:



Работа микросервисов:



Вывод: Я научился работать с Docker и Docker-Compose.