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

ДИСЦИПЛИНА:

Интеграция и развертывание программного обеспечения с помощью контейнеров

**Лабораторная работа 3.1**

**Docker Compose для мультиконтейнерных приложений**

Выполнила: Шведова С.С., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

**Цель работы:** научиться создавать Dockerfile и собирать образы Docker для приложений.

**Задачи:**

1. Создать Dockerfile для указанного приложения.
2. Собрать образ Docker с использованием созданного Dockerfile.
3. Запустить контейнер из собранного образа и проверить его работоспособность.
4. Выполнить индивидуальное задание.

**Вариант 13.** Создайте Dockerfile для приложения на Java, которое использует библиотеку JDBC для подключения к базе данных PostgreSQL и выполнения простого запроса.

Для того, чтобы было подключение к базе данных PostgreSQL, надо его настроить. Сначала надо запустить контейнеры на основе настроек, определенных в файле docker-compose.yml (рисунок 1).

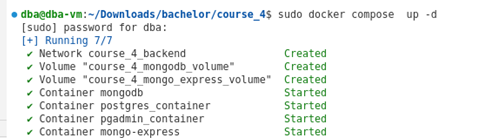


Рисунок 1. Запуск контейнеров

Затем с помощью команды sudo docker inspect id\_conteiner надо узнать IP address сервера PgAdmin и после этого уже перейти по нему (рисунок 2). Так как индивидуальное задание требует выполнение простого запроса, то нужно создать таблицу и заполнить ее значениями.

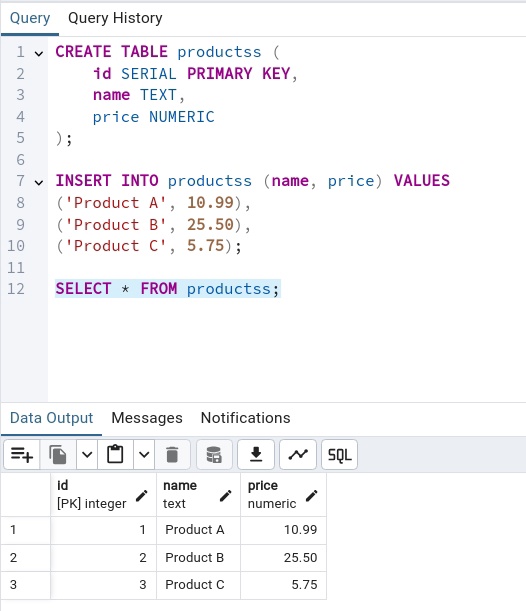


Рисунок 2. Подключение к PostreSQL

После этого надо в папке проекта my-java-app создать докерфайл и исходный код программы, который по заданию должен быть написан на языке Java (рисунок 3).



Рисунок 3. Создание файлов

Докерфайл выглядит следующим образом (рисунок 4).

1. **FROM openjdk:11-jdk-slim.** Указывает базовый образ, на основе которого будет строиться новый образ. Она используется облегченный образ OpenJDK версии 11 с установленным JDK (Java Development Kit). Он будет служить средой для выполнения Java-приложения.

2. **WORKDIR /app**: Эта команда устанавливает рабочую директорию внутри контейнера. Все последующие команды будут выполняться из этой директории.

3. **COPY . /app**: Эта команда копирует все файлы из текущей директории хоста (где находится Dockerfile) в директорию /app внутри контейнера.

4. **RUN javac -cp postgresql-42.7.5.jar App.java**: Эта команда выполняет компиляцию файла App.java, используя Java-компилятор javac. Параметр -cp postgresql-42.7.5.jar указывает класс-путь (classpath) для компилятора, так что он может находить классы из библиотеки Postgres, необходимой для приложения.

5. **CMD ["java", "-cp", ".:postgresql-42.7.5.jar", "App"]** определяет команду, которая будет выполнена при запуске контейнера. Она запускает Java-программу, задавая класс-путь (classpath), чтобы включить текущую директорию (.) и файл библиотеки postgresql-42.7.5.jar. App — это имя класса, который будет запущен.



Рисунок 4. Dockerfile

Чтобы включить файл библиотеки postgresql-42.7.5.jar, надо его установить и переместить в рабочий каталог. На рисунке 5 показаны файлы в папке проекта, где библиотека установлена.



Рисунок 5. Папка проекта

Затем нужно создать исходный код программы, написанный на Java (рисунок 6).

Сначала происходит импорт необходимых классов, то есть код импортирует классы из пакета java.sql, которые необходимы для работы с базой данных, включая Connection, DriverManager, PreparedStatement, ResultSet, и SQLException. Затем в классе App определяются параметры подключения и создается вывод простого запроса. В данном случае выбирает все столбцы из таблицы products и, используя цикл while, программа проходит по каждой строке результата, и выводит значение столбца price.



Рисунок 6. Код программы

После этого надо собрать образ (рисунок 7).

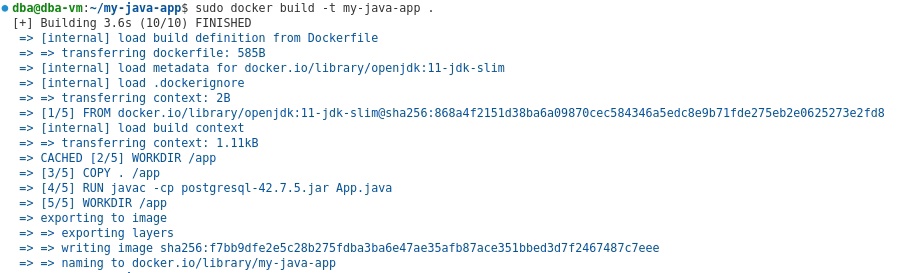


Рисунок 7. Сборка образа

Теперь можно запустить приложение. Как можно увидеть на рисунке 8, оно действительно вывело значения столбца price.



Рисунок 8. Запуск приложения

На рисунке 9 показано удаление докера.

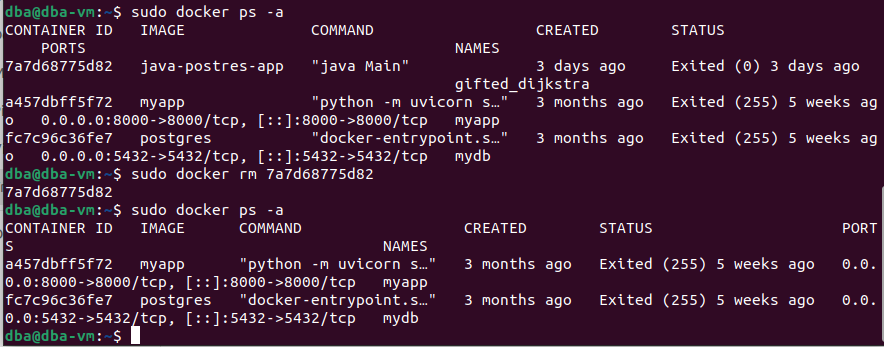


Рисунок 9. Удаление докера

**Выводы**

1. Был создан Dockerfile для приложения на Java;

2. Собран образ Docker с использованием созданного Dockerfile;

3. Был запущен контейнер из собранного образа и была проверена его работоспособность.

**Контрольные вопросы**

**1. Что такое Dockerfile и для чего он используется?**

Dockerfile — это текстовый файл, содержащий инструкции для сборки исходного кода. Он позволяет автоматизировать процесс построения контейнеров, используя базовый образ.

**2. Какие основные инструкции используются в Dockerfile?**

* FROM. Указывает базовый образ, на основе которого нужно создать новый. Чаще всего FROM используется для образов с операционной системой и предустановленными компонентами.
* RUN. Указывает, какие команды необходимо выполнить внутри контейнера во время сборки образа. Так можно установить зависимости или обновить пакеты до нужной версии.
* COPY и ADD. Копирует файлы из локальной файловой системы в контейнер. Чаще всего копируют исходный код приложения.
* WORKDIR. Устанавливает рабочую директорию для последующих инструкций. Так можно последовательно работать с файлами в разных директориях.
* CMD. Определяет аргументы по умолчанию при запуске контейнера.
* ENTRYPOINT. Задает команду, которая будет выполнена при запуске контейнера.

**3. Как выполняется сборка образа Docker с использованием Dockerfile?**

Сборка образа Docker с использованием Dockerfile выполняется с помощью команды docker build.

**4. Как запустить контейнер из собранного образа?**

Чтобы запустить контейнер из собранного образа, нужно выполнить команду docker run

**5. Каковы преимущества использования Dockerfile для создания образов Docker?**

* Позволяет создавать одинаковое окружение для работы программы, независимо от машины, которая собирает образ.
* Оптимизирует распределение ресурсов по контейнерам.
* Управляет жизненным циклом приложения — запускает, останавливает, масштабирует и обновляет контейнеры