МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра компьютерных интеллектуальных технологий проектирования

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

По дисциплине: «Системы хранения и обработки данных»

Тема: «Развёртывание СУБД Postgres с использованием средств автоматизации развёртывания и управления приложениями»

Выполнил работу студент группы змИИВТ-231: Агеев А.А.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

СОДЕРЖАНИЕ

[Задачи лабораторной работы 3](#_Toc157520505)

[1 Ход работы 3](#_Toc157520506)

[1.1 Создание Dockerfile 3](#_Toc157520507)

[1.2 Создание образа на основе dockerfile 5](#_Toc157520508)

[1.3 Запуск контейнера 6](#_Toc157520509)

[1.4 Подключение к работающему контейнеру 7](#_Toc157520510)

[1.5 Создание контейнера с томом 9](#_Toc157520511)

[1.6 Создание контейнера с помощью docker-compose.yml 10](#_Toc157520512)

[2 Контрольные вопросы 13](#_Toc157520513)

# Задачи лабораторной работы

* установка приложения для работы с Docker-контейнерами;
* установка программного средства для работы с СУБД DBeaver;
* скачивание docker-образа с СУБД postgres;
* создание Dockerfile;
* создание образа на основе Dockerfile;
* запуск контейнера для развёртывания СУБД Postgres;
* подключение к работающему контейнеру и запуск интерфейса psql;
* создание контейнера с томом (volume);
* создание контейнера с использованием файла docker-compose.yml

# 1 Ход работы

## 1.1 Создание Dockerfile

Для запуска postges в изолированной среде напишем докерфайл, изображен на рисунке 1.

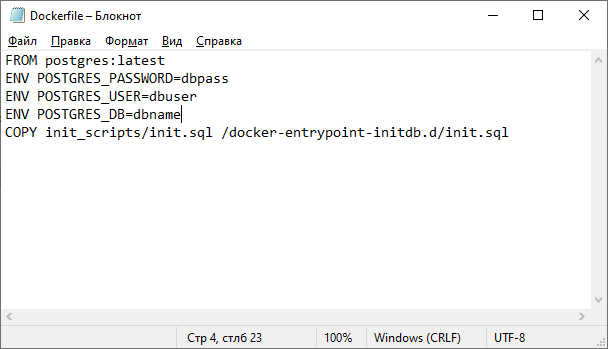


Рисунок 1 – Dockerfile

FROM postgres:latest - Эта строка указывает, что Docker должен использовать образ Postgres:latest в качестве базового образа для создания нового образа. Образ Postgres:latest содержит предустановленную базу данных PostgreSQL.

ENV POSTGRES\_PASSWORD=dbpass - Эта строка устанавливает переменную среды POSTGRES\_PASSWORD со значением dbpass. Переменная POSTGRES\_PASSWORD используется для установки пароля для пользователя PostgreSQL postgres.

ENV POSTGRES\_USER=dbuser - Эта строка устанавливает переменную среды POSTGRES\_USER со значением dbuser. Переменная POSTGRES\_USER используется для установки имени пользователя для пользователя PostgreSQL postgres.

ENV POSTGRES\_DB=dbname - Эта строка устанавливает переменную среды POSTGRES\_DB со значением dbname. Переменная POSTGRES\_DB используется для установки имени базы данных для базы данных PostgreSQL.

COPY init\_scripts/init.sql /docker-entrypoint-initdb.d/init.sql - Эта строка копирует файл init.sql из каталога init\_scripts в каталог /docker-entrypoint-initdb.d. Каталог /docker-entrypoint-initdb.d содержит файлы, которые выполняются при запуске контейнера PostgreSQL.

Далее создадим файл в директории init\_scripts с названием init.sql содержание изображено на рисунке 2.

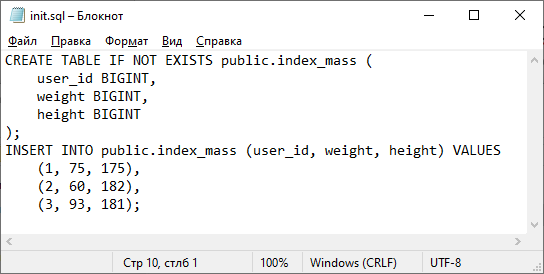


Рисунок 2 – init.sql

Запрос №1 - создает таблицу index\_mass в публичной схеме. Таблица имеет три столбца:

user\_id - целое число, которое представляет идентификатор пользователя.

weight - целое число, которое представляет вес пользователя в килограммах.

height - целое число, которое представляет рост пользователя в сантиметрах.

Ключевое слово IF NOT EXISTS означает, что таблица будет создана только в том случае, если она еще не существует.

Запрос №2 - вставляет три строки в таблицу index\_mass. Значения в каждой строке, следующие:

user\_id - 1, 2, 3

weight - 75, 60, 93

height - 175, 182, 181

Каждая строка представляет данные о пользователе, включая его идентификатор, вес и рост.

## 1.2 Создание образа на основе dockerfile

Запустим сборку с помощью команды «docker build -t d\_img:latest . »

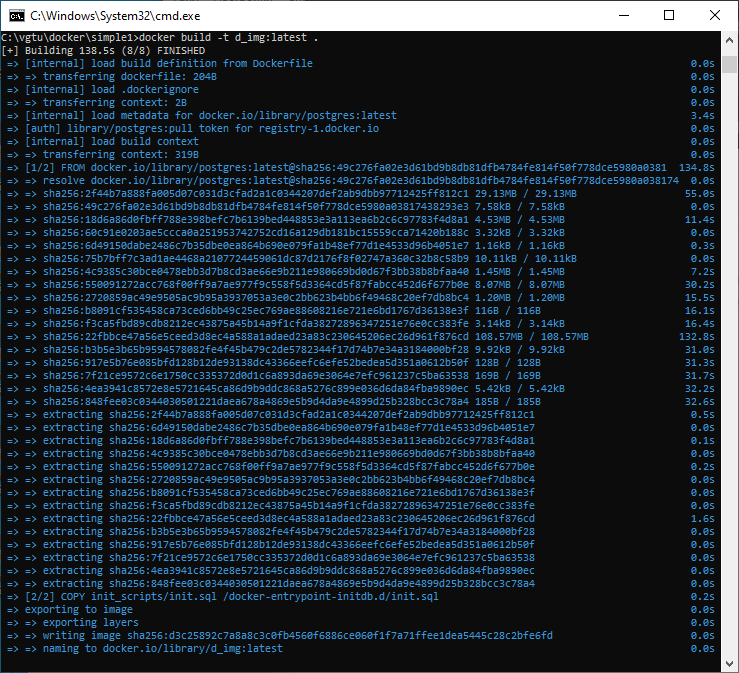


Рисунок 3 – Процесс сборки образа

## 1.3 Запуск контейнера

Для запуска напишем команду «docker run -d -p 5432:5432 --name d\_cont d\_img:latest», результат выполнения на рисунке 4.

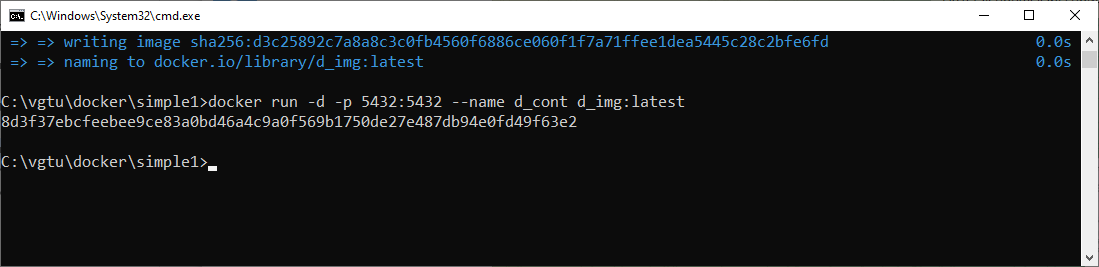


Рисунок 4 – Запуск контейнера

## 1.4 Подключение к работающему контейнеру

Подключимся и проверим работоспособность базы данных через DBeaver, изображено на рисунке 5.

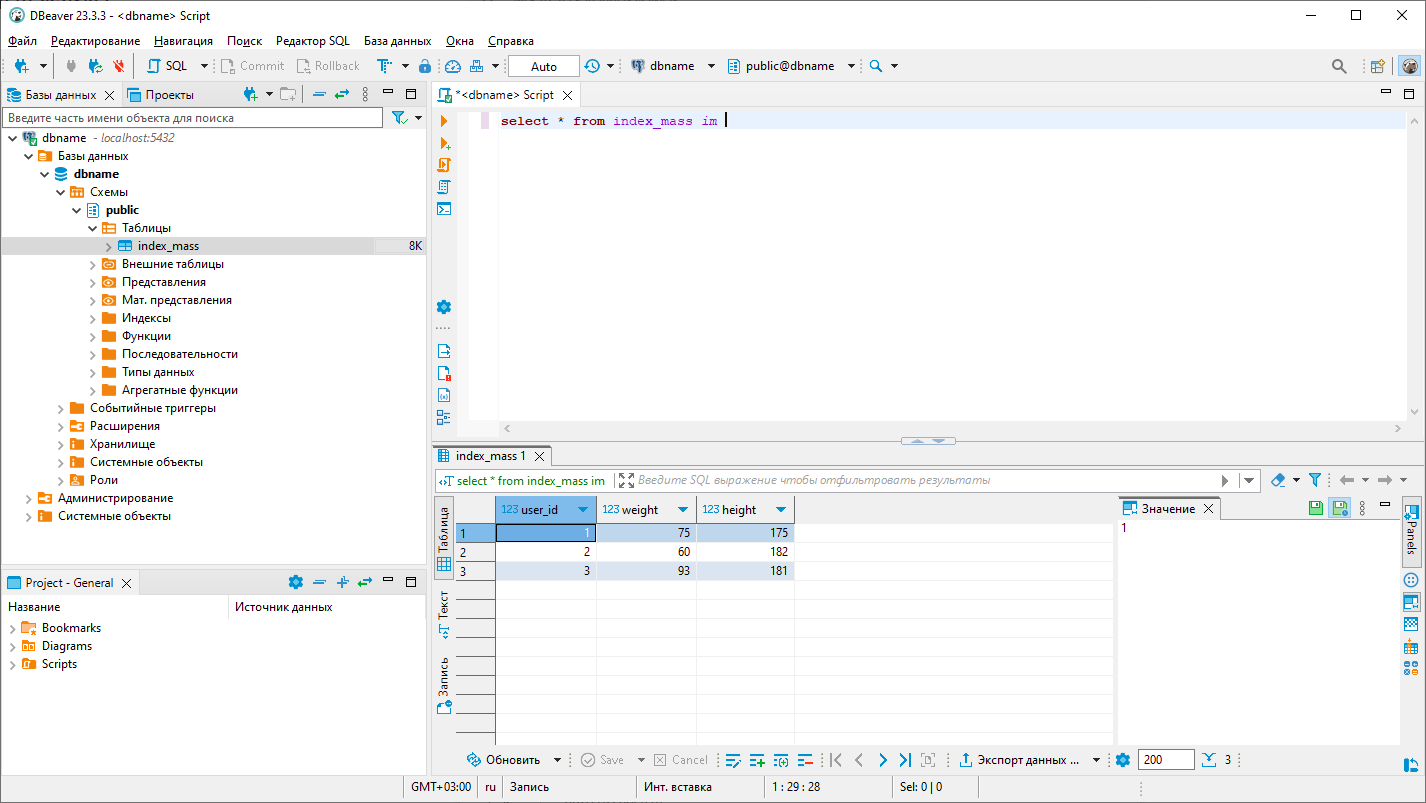


Рисунок 5- Подключение к базе

Напишем команду, которая позволит подключаться к работающему контейнеру, запускать интерфейс psql и вносить новые данные «на лету»: «docker exec -it d\_cont psql -d dbname -U dbuser», результат на рисунке 6.

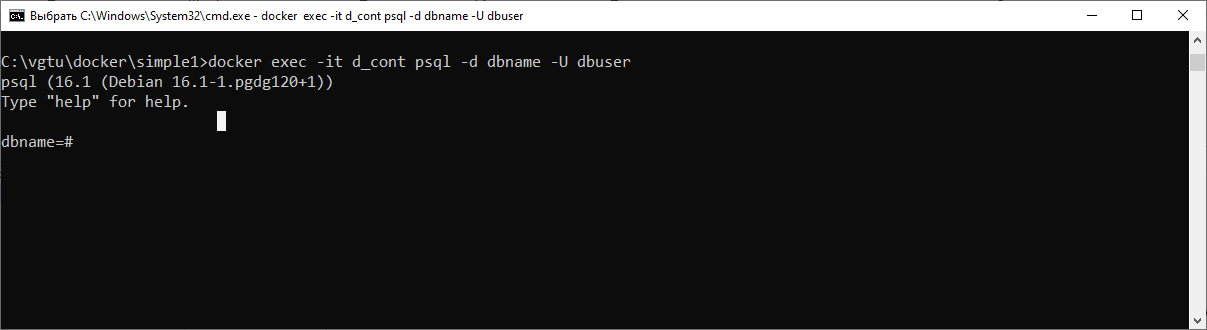


Рисунок 6 – Подключение к базе из консоли

Эта команда откроет новый сеанс терминала в контейнере, а затем выполнит команду psql. Флаги -it сообщают Docker о выделении контейнеру псевдо-TTY и интерактивного режима, чтобы вы могли взаимодействовать с сеансом psql. Флаг -d dbname указывает psql на необходимость подключения к указанной базе данных dbname, а -U dbuser флаг указывает на необходимость аутентификации пользователя dbuser.

Получим список всех баз данных с помощью запроса SELECT datname FROM pg\_database;, изображено на рисунке 7.

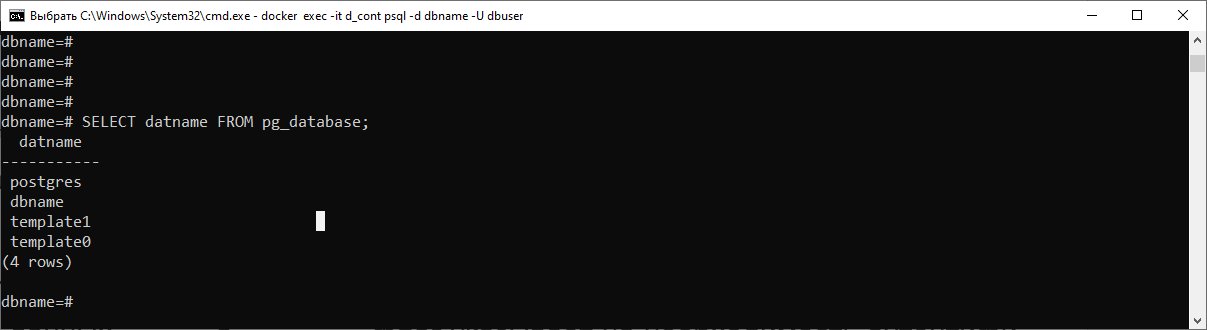


Рисунок 7 – Список баз данных

Чтобы данные, измененные в процессе работы контейнера, были доступны после удаления контейнера (во вновь созданном контейнере) необходимо создать том (volume).

## 1.5 Создание контейнера с томом

Добавим в Dockerfile добавить строку: «VOLUME /d\_data:/var/lib/postgresql/data»

Запустим контейнер с помощью команды docker run -d -p 5432:5432 --name d\_cont d\_img:latest -v /d\_data:/var/lib/postgresql/data, рисунок 8.

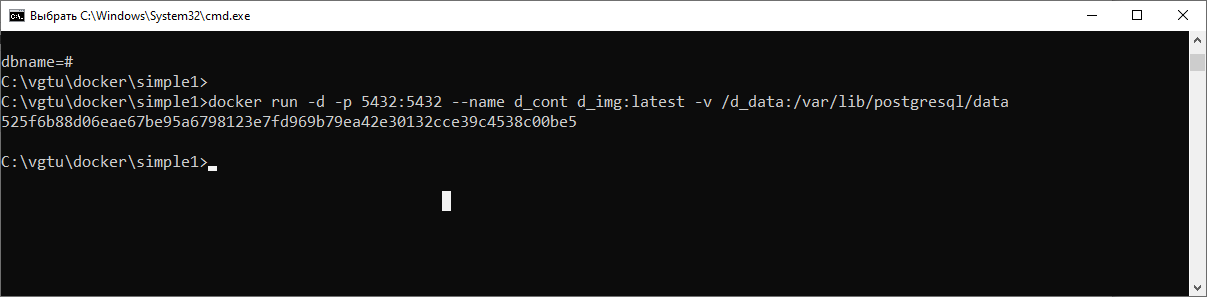


Рисунок 8 – Выполнение команды docker run

Команда docker run запускает контейнер Docker. В данном случае она выполняет следующие действия:

* -d - запускает контейнер в фоновом режиме.
* -p 5432:5432 - сопоставляет порт 5432 на хосте с портом 5432 в контейнере. Это позволяет подключиться к базе данных PostgreSQL извне контейнера.
* --name d\_cont - задаёт имя контейнеру.
* d\_img:latest - указывает на образ Docker, который необходимо запустить. В данном случае используется образ d\_img, последняя версия которого.
* -v /d\_data:/var/lib/postgresql/data - связывает директорию /d\_data на хосте с директорией /var/lib/postgresql/data в контейнере. Это позволяет использовать данные базы данных, расположенные на хосте, в контейнере.

При запуске команды Docker создаёт контейнер с именем d\_cont. В контейнере устанавливается образ d\_img, последняя версия которого. Порт 5432 на хосте сопоставляется с портом 5432 в контейнере. Директория /d\_data на хосте привязывается к директории /var/lib/postgresql/data в контейнере.

## 1.6 Создание контейнера с помощью docker-compose.yml

Создим контейнер с использованием файла docker-compose.yml

Содержание файла docker-compose.yml, показано на рисунке 9.

version: "3.9"

services:

 dbpost:

  image: postgres:latest

  environment:

   POSTGRES\_DB: dbcompose

   POSTGRES\_USER: usrcompose

   POSTGRES\_PASSWORD: passcompose

  volumes:

   - ./init\_scripts/init.sql:/docker-entrypoint-initdb.d/init.sql

  ports:

   - 5432:5432

Строка 1 - version: "3.9"

Указывает версию формата файла docker-compose.yml. В данном случае используется версия 3.9.

Строка 2 - services:

Начинает раздел, в котором определяются сервисы.

Строка 3 - dbpost:

Определяет сервис dbpost.

Строка 4 - image: postgres:latest

Указывает, что сервис dbpost должен быть построен на основе образа postgres:latest.

Строка 5 - environment:

Начинает раздел, в котором определяются переменные среды для сервиса dbpost.

Строка 6 - POSTGRES\_DB: dbcompose

Указывает имя базы данных для сервиса dbpost.

Строка 7 - POSTGRES\_USER: usrcompose

Указывает имя пользователя для сервиса dbpost.

Строка 8 - POSTGRES\_PASSWORD: passcompose

Указывает пароль для пользователя usrcompose.

Строка 9 - volumes:

Начинает раздел, в котором определяются тома для сервиса dbpost.

Строка 10 - ./init\_scripts/init.sql:/docker-entrypoint-initdb.d/init.sql

Связывает том ./init\_scripts/init.sql с каталогом /docker-entrypoint-initdb.d внутри контейнера dbpost. Это позволяет при запуске контейнера выполнить SQL-скрипт init.sql, который создаст базу данных dbcompose с пользователем usrcompose и паролем passcompose.

Строка 11 - ports:

Начинает раздел, в котором определяются порты для сервиса dbpost.

Строка 12 - 5432:5432

Перенаправляет порт 5432 из контейнера dbpost на порт 5432 хоста. Это позволяет подключаться к базе данных dbcompose извне контейнера.

Процесс работы файла:

1. Docker Compose читает файл docker-compose.yml и создает сервис dbpost.
2. Сервис dbpost строится на основе образа postgres:latest.
3. Переменные среды POSTGRES\_DB, POSTGRES\_USER и POSTGRES\_PASSWORD устанавливаются в контейнере dbpost.
4. Том ./init\_scripts/init.sql монтируется в каталог /docker-entrypoint-initdb.d внутри контейнера dbpost.
5. Скрипт init.sql выполняется внутри контейнера dbpost.
6. Порт 5432 из контейнера dbpost перенаправляется на порт 5432 хоста.

Результат:

После выполнения файла docker-compose.yml будет создан контейнер dbpost с запущенной базой данных dbcompose. К базе данных можно подключаться извне контейнера, используя порт 5432.

Выполним команду docker-compose up, подключимся к базе с помощью DBeaver и посмотрим на результат, рисунок 9.

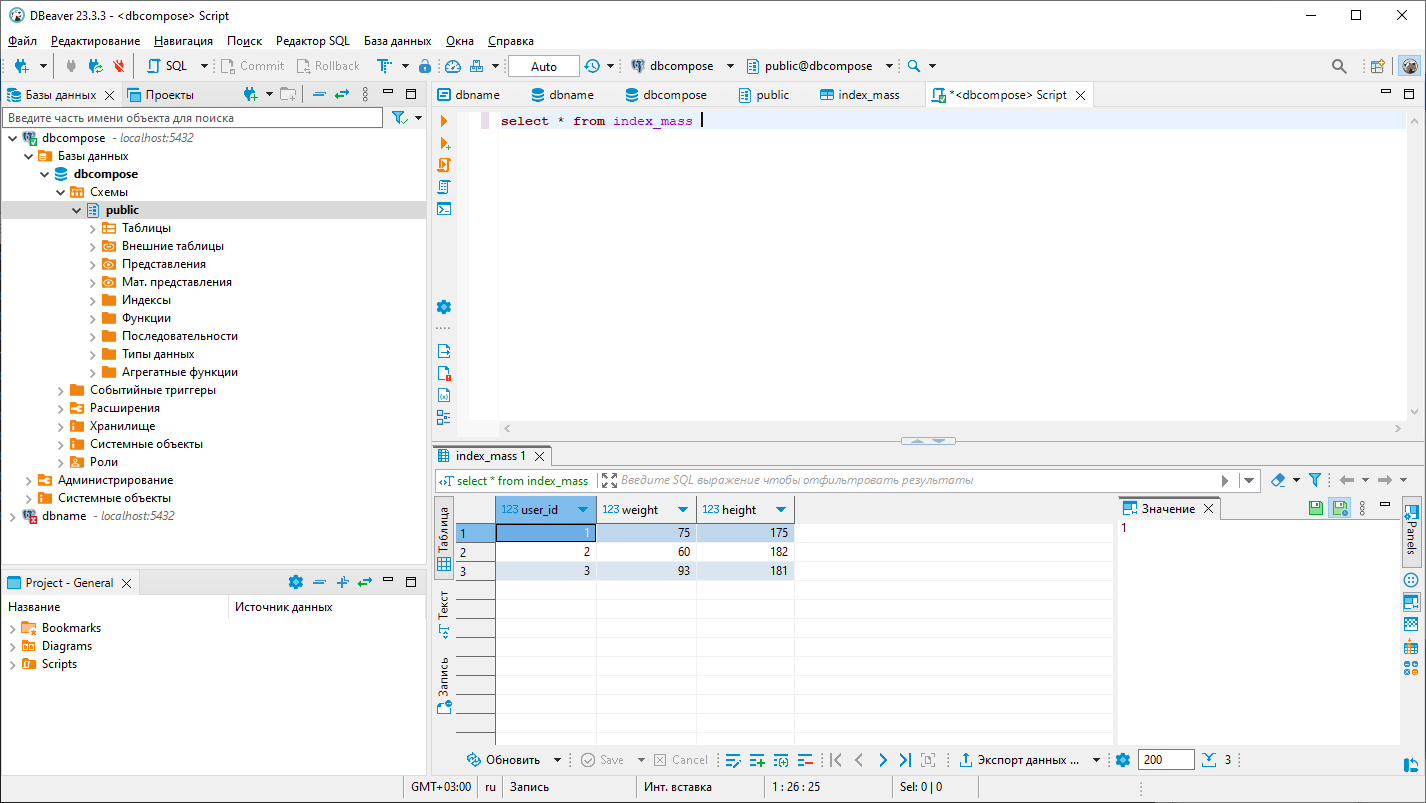


Рисунок 9 – Подключение к контейнеру, запущенному с помощью docker-compose up

# 2 Контрольные вопросы

1. Что такое Docker.

Docker - это платформа для разработки, доставки и запуска контейнерных приложений. Контейнеры - это изолированные среды выполнения, которые содержат все необходимое для работы приложения, включая код, библиотеки, системные инструменты и среду исполнения.

Docker позволяет разработчикам создавать, упаковывать, развертывать и запускать приложения в виде контейнеров. Это упрощает и ускоряет процесс разработки и развертывания приложений, поскольку контейнеры могут быть запущены на любой машине, которая поддерживает Docker.

2. Зачем нужен Docker.

Docker имеет ряд преимуществ по сравнению с традиционными способами развертывания приложений, такими как виртуализация или установка приложений на bare metal. Вот некоторые из основных преимуществ Docker:

* Ускорение разработки и развертывания приложений: Docker позволяет разработчикам быстро создавать контейнеры для своих приложений и запускать их на любой машине, которая поддерживает Docker. Это может значительно сократить время, необходимое для доставки приложений в производство.
* Улучшенная масштабируемость: Контейнеры легко масштабировать, добавляя или удаляя контейнеры по мере необходимости. Это может помочь организациям удовлетворить растущие требования к пропускной способности и производительности.
* Улучшенная безопасность: Контейнеры изолированы друг от друга, что помогает защитить приложения от внешних угроз.
* Экономия затрат: Docker может помочь организациям снизить затраты на инфраструктуру, поскольку контейнеры требуют меньше ресурсов, чем виртуальные машины или bare metal.

3. Что такое docker-образ.

Docker-образ - это набор файлов, который описывает, как создать контейнер. Образы могут быть созданы вручную или с помощью Dockerfile.

Dockerfile - это текстовый файл, который содержит инструкции по созданию образа. Инструкции Dockerfile могут включать в себя установку программного обеспечения, копирование файлов и настройку параметров.

4. Что такое docker-контейнер.

Docker-контейнер - это запущенный образ. Контейнеры представляют собой изолированные среды выполнения, которые содержат все необходимое для работы приложения.

Контейнеры можно запускать на любой машине, которая поддерживает Docker. Это упрощает и ускоряет процесс доставки приложений в производство.

5. Что такое volume и зачем он нужен.

Volume - это раздел диска, который используется для хранения данных, которые должны быть доступны для контейнеров. Volumes можно использовать для хранения данных, которые должны быть сохранены после остановки контейнера.

Volumes можно создавать вручную или с помощью Dockerfile.

6. Что такое docker-compose.

Docker-compose - это инструмент, который позволяет разработчикам создавать и управлять сложными приложениями, состоящими из нескольких контейнеров.

Docker-compose использует файлы YAML для описания структуры приложения и зависимостей между контейнерами.

7. В чем разница между dockerfile и docker-compose.

Dockerfile используется для создания образов, а docker-compose - для создания и управления контейнерами.

Dockerfile - это более простой инструмент, который лучше подходит для создания простых приложений. Docker-compose - это более мощный инструмент, который лучше подходит для создания сложных приложений.

8. Какая команда позволяет отправлять различные задания в запущенный докер-контейнер.

Команда docker exec позволяет отправлять различные задания в запущенный докер-контейнер.

Например, чтобы выполнить команду ls в запущенном контейнере с именем my-container, вы можете использовать следующую команду:

docker exec my-container ls

9. С помощью какого инструмента можно сохранить важные данные после аварийного отключения контейнера.

Для сохранения важных данных после аварийного отключения контейнера можно использовать volumes. Volumes - это разделы диска, которые используются для хранения данных, которые должны быть доступны для контейнеров.

Чтобы сохранить данные в volume после аварийного отключения контейнера, необходимо создать volume вручную или с помощью Dockerfile.