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

ОТЧЕТ 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 мИИВТ-231 Талдыкин И.С.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2023

Цель работы: изучить основы работы программного средства для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker на примере решения задачи развёртывания СУБД Postgres.

Задание:

1. Изучить сайт <https://www.docker.com>, документацию по docker (https://docs.docker.com), зарегистрироваться на Docker Hub. Изучить основные команды docker для консоли.
2. Скачать и установить Docker Desktop. При работе в команде необходимо выполнять работу на различных операционных системах. Приоритетный порядок выбора операционных систем: Astra Linux, Windows 10, CentOS, любая другая система, кроме указанных ранее, включая MacOS. Как минимум в одном варианте необходимо установить Docker и далее работать на виртуальной машине (операционная система – по вашему выбору). Количество вариантов (систем, на которые устанавливается Docker) может не превышать количество участников команды (не запрещается делать больше вариантов, чем участников команды).
3. Скачать и установить средство для работы с СУБД DBeaver. Разрешается использовать другое подобное средство, но необходимо обосновать свой выбор.
4. Скачать docker-образ с СУБД postgres с помощью команды для консоли.
5. Создать Dockerfile со следующим содержанием:

FROM postgres:latest

ENV POSTGRES\_PASSWORD=dbpass

ENV POSTGRES\_USER=dbuser

ENV POSTGRES\_DB=dbname

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

Описать каждую строку данного файла (что означает и какую задачу решает).

Содержание файла init.sql:

CREATE TABLE IF NOT EXISTS public.index\_mass (

user\_id BIGINT,

weight BIGINT,

height BIGINT

);

INSERT INTO public.index\_mass (user\_id, weight, height) VALUES

(1, 75, 175),

(2, 60, 182),

(3, 93, 181);

Описать каждую строку данного файла.

1. Создать образ с помощью команды «docker build -t d\_img:latest .».

* Команду надо запускать из каталога, в котором находится Dockerfile
* В результате запустится Dockerfile и будет создан docker-образ «d\_img»

1. Чтобы воспользоваться созданной СУБД необходимо запустить контейнер с именем «d\_cont» на основе образа «d\_img:latest» с помощью команды: «docker run -d -p 5432:5432 --name d\_cont d\_img:latest».

Описать данную команду (все составляющие и что происходит при запуске).

1. Проверить через DBeaver развёрнутую СУБД. Подключить базу данных с параметрами, указанными в Dockerfile командами ENV: (POSTGRES\_PASSWORD=dbpass (пароль), POSTGRES\_USER=dbuser (имя пользователя), POSTGRES\_DB=dbname [название базы]), порт – 5432.
2. Написать специальную команду docker, которая позволит подключаться к работающему контейнеру, запускать интерфейс psql и вносить новые данные «на лету»: «docker exec -it d\_cont psql -d -U usr dbn».

Теперь есть возможность работать с базой данных из консоли.

Необходимо описать данную команду (все составляющие и что происходит при запуске).

Вывести в консоль список имеющихся баз данных.

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

* Для добавления в Dockerfile инструкции для вызова создания контейнера с volume необходимо в Dockerfile добавить строку: «VOLUME /d\_data:/var/lib/postgresql/data».
* При этом локально будет создан том (volume) в папке «/d\_data», в котором, в свою очередь, будут сохраняться изменения, аналогичные тем, которые происходят в контейнере в папке «/var/lib/postgresql/data».

1. Удалить созданный контейнер. Создать контейнер с помощью команды: «docker run -d -p 5432:5432 --name d\_cont d\_img:latest -v /d\_data:/var/lib/postgresql/data».

Необходимо описать данную команду (все составляющие и что происходит при запуске).

1. Создать текстовый файл cmd.txt, в котором описать все команды, которые необходимо использовать для развертывания базы данных Postgres с помощью Dockerfile (создание образа, запуск контейнера, создание тома).

Также в этом файле необходимо указать команду для создания docker-контейнера с томом (volume) на основе docker-образа и команду для запуска контейнера с интерфейсом psql для внесения новых данных в БД.

1. Создать контейнер с использованием файла docker-compose.yml

Содержание файла docker-compose.yml:

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

Описать каждую строку файла (что означает, какую задачу решает), процесс работы данного файла и результат.

1. При работе в команде необходимо выполнять работу на различных операционных системах. Приоритетный порядок выбора операционных систем: Astra Linux, Windows 10, CentOS, любая другая система, кроме указанных ранее.
2. Подготовить отчёт о проделанной работе. Все операции по работе с docker необходимо контролировать через приложение Docker Desktop и делать соответствующие скрины.
3. Отчётные материалы загрузить в репозиторий Git и отправить ссылку на ваш репозиторий на платформе github на почту преподавателю. Репозиторий должен быть публичным.

Пакет отчётных документов в репозитории должен включать:

* отчёт;
* презентацию;
* файл «Dockerfile»;
* SQL-скрипт «init.sql» (если в инструкции Dockerfile указано, что этот скрипт находится в определенной папке, то вкладывать его в соответствующую папку, либо изменить соответствующим образом Dockerfile);
* файл с описанием команд для запуска контейнера «cmd.txt»;
* файл «docker-compose.yml».

Ход работы:

Docker и DBeaver были установлены без проблем на хост машину Windows 10.

Командой docker pull postgres был загружен образ postgres, рисунок 1.

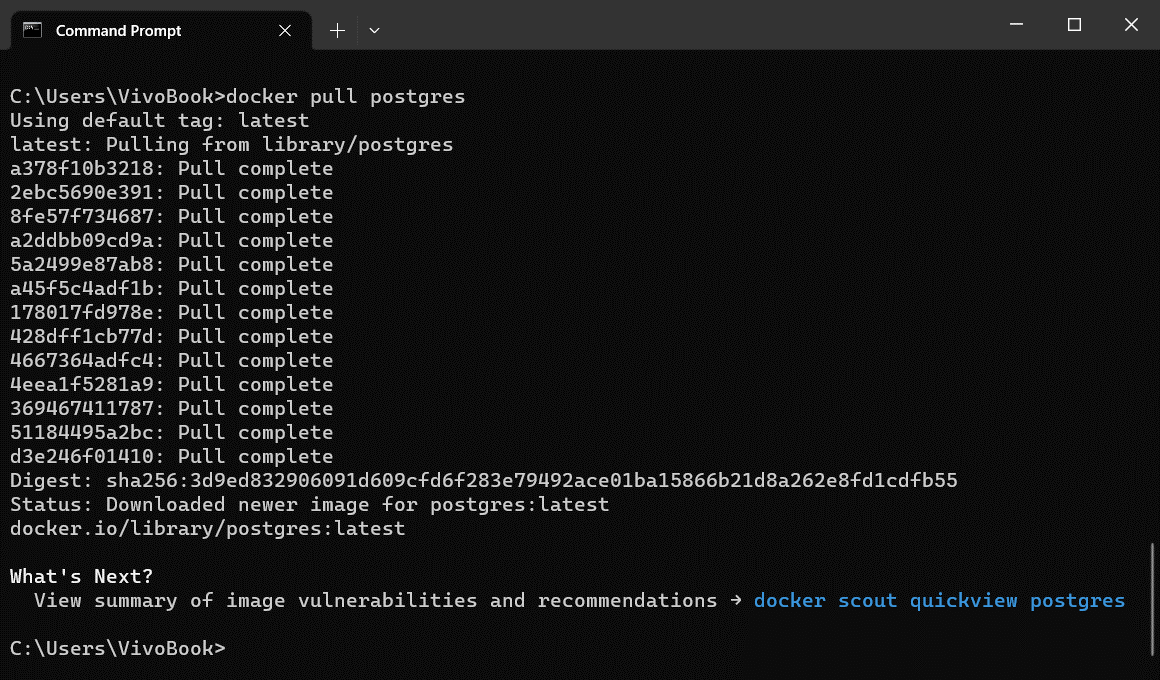


Рисунок 1 – Загрузка образа postgres

На рисунке 2 представлен образ postgres в Docker.

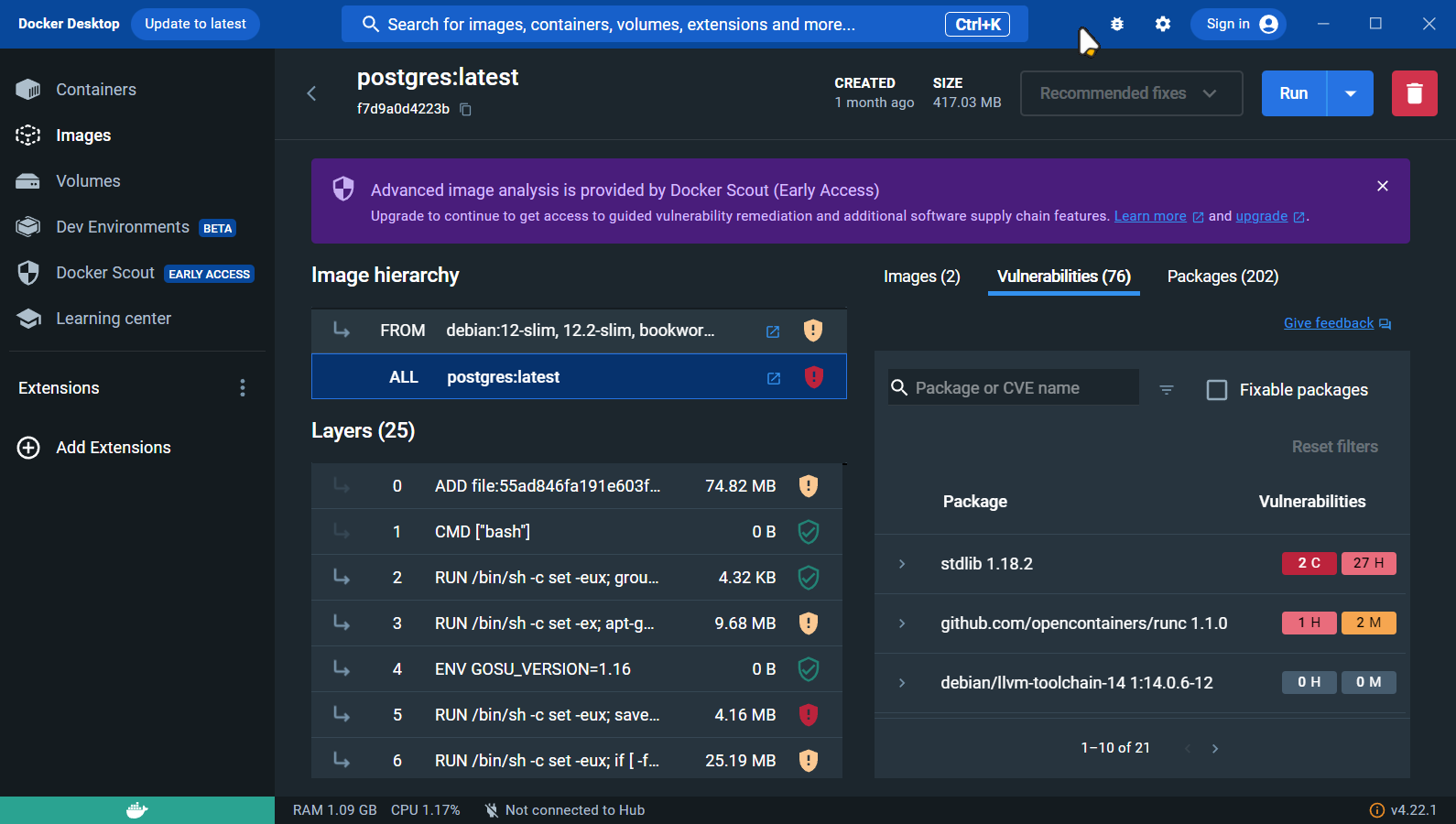


Рисунок 2 – Образ Postgres

На рисунках 3 и 4 представлены файлы Docker и init.sql, соответственно.

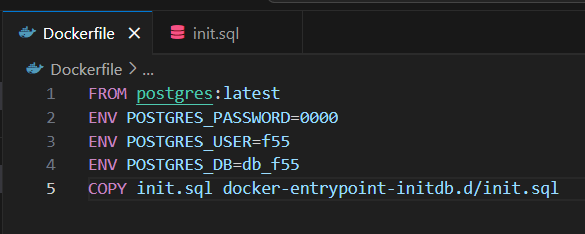


Рисунок 3 – Docker файл

«FROM postgres:latest»: Это инструкция, которая указывает базовый образ, на основе которого будет создан контейнер. В данном случае используется официальный образ PostgreSQL с тегом "latest", что означает последнюю доступную версию.

«ENV POSTGRES\_PASSWORD=0000»: Эта инструкция устанавливает переменную окружения POSTGRES\_PASSWORD с значением "0000". Это пароль, который будет использоваться для подключения к PostgreSQL базе данных.

«ENV POSTGRES\_USER=f55»: Эта инструкция устанавливает переменную окружения POSTGRES\_USER с значением "f55". Это имя пользователя, которое будет создано для доступа к базе данных.

«ENV POSTGRES\_DB=db\_f55»: Эта инструкция устанавливает переменную окружения POSTGRES\_DB с значением "db\_f55". Это имя базы данных, которая будет создана при инициализации PostgreSQL.

«COPY init.sql docker-entrypoint-initdb.d/init.sql»: Эта инструкция копирует локальный файл init.sql внутрь контейнера и размещает его в директории docker-entrypoint-initdb.d. В контейнерах PostgreSQL, файлы, размещенные в этой директории, будут автоматически выполнены при инициализации базы данных. В данном случае, init.sql содержит SQL-скрипт, который будет выполнен при первом запуске контейнера для настройки и инициализации базы данных.

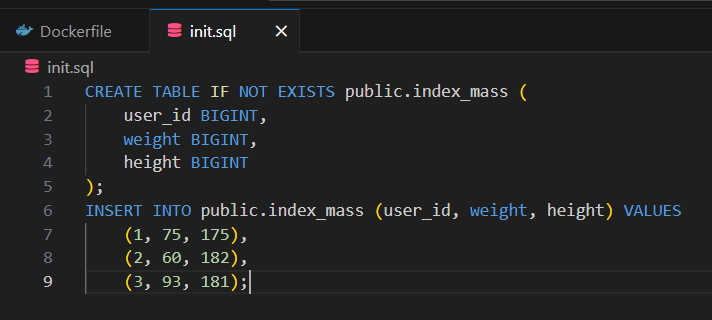


Рисунок 4 – SQL файл

SQL-файл выполняет две основные операции: создание таблицы и вставку данных в эту таблицу.

«CREATE TABLE IF NOT EXISTS public.index\_mass (user\_id BIGINT, weight BIGINT, height BIGINT);»: Эта строка создает новую таблицу с именем index\_mass в схеме public. Если таблица с таким именем уже существует, она не будет пересоздана благодаря условию IF NOT EXISTS. В таблице определены три столбца: user\_id, weight и height, каждый из которых имеет тип данных BIGINT.

«INSERT INTO public.index\_mass (user\_id, weight, height) VALUES (1, 75, 175), (2, 60, 182), (3, 93, 181);»: Эта строка вставляет данные в созданную таблицу index\_mass. Она указывает столбцы user\_id, weight и height, в которые будут вставлены значения. Затем в скобках чередуются наборы значений, каждый из которых представляет собой новую строку в таблице. В данном случае, вставляются три строки данных с соответствующими значениями:

(1, 75, 175), (2, 60, 182) и (3, 93, 181).

Далее был построен образ командой «docker build -t d\_img:latest .» и запущен контейнер (команда «docker run -d -p 5432:5432 --name d\_cont d\_img:latest») на основе вышеуказанного файла Docker, рисунок 5. На рисунке 6 представлен запущенный контейнер «d\_cont».

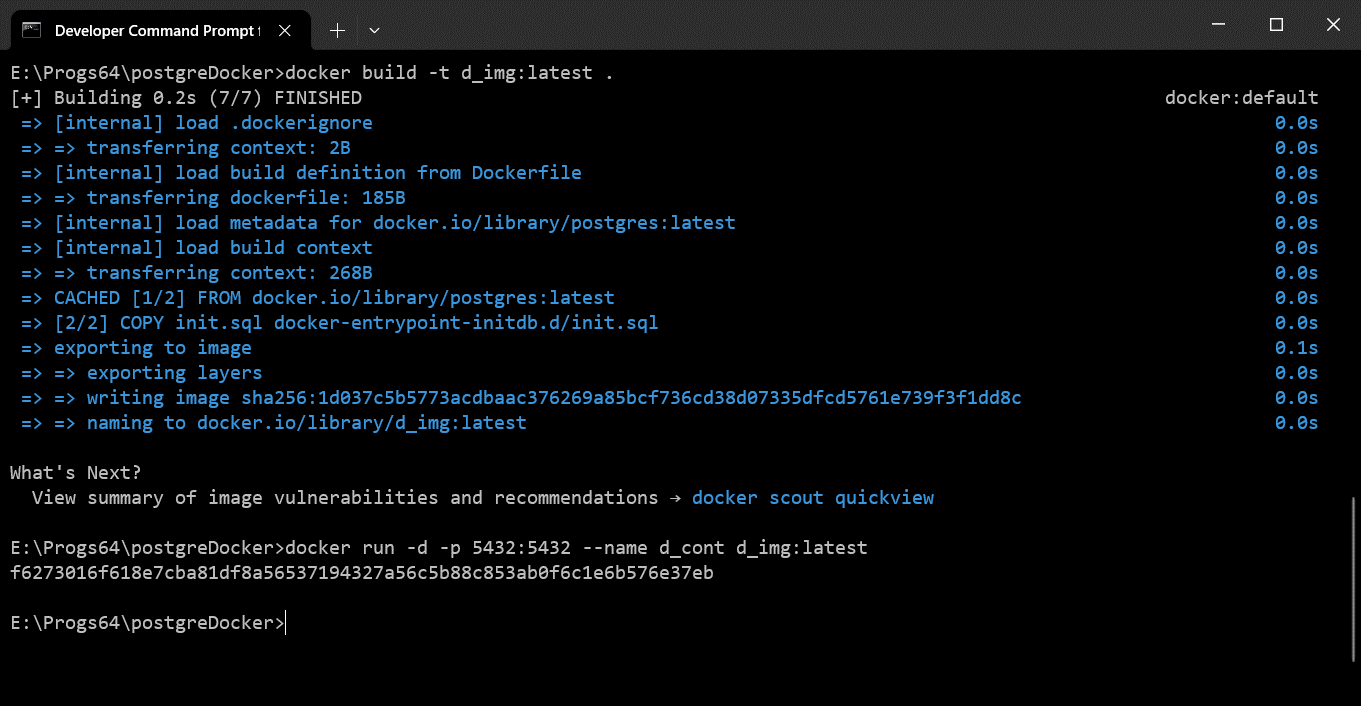


Рисунок 5 – Образ построен, контейнер запущен

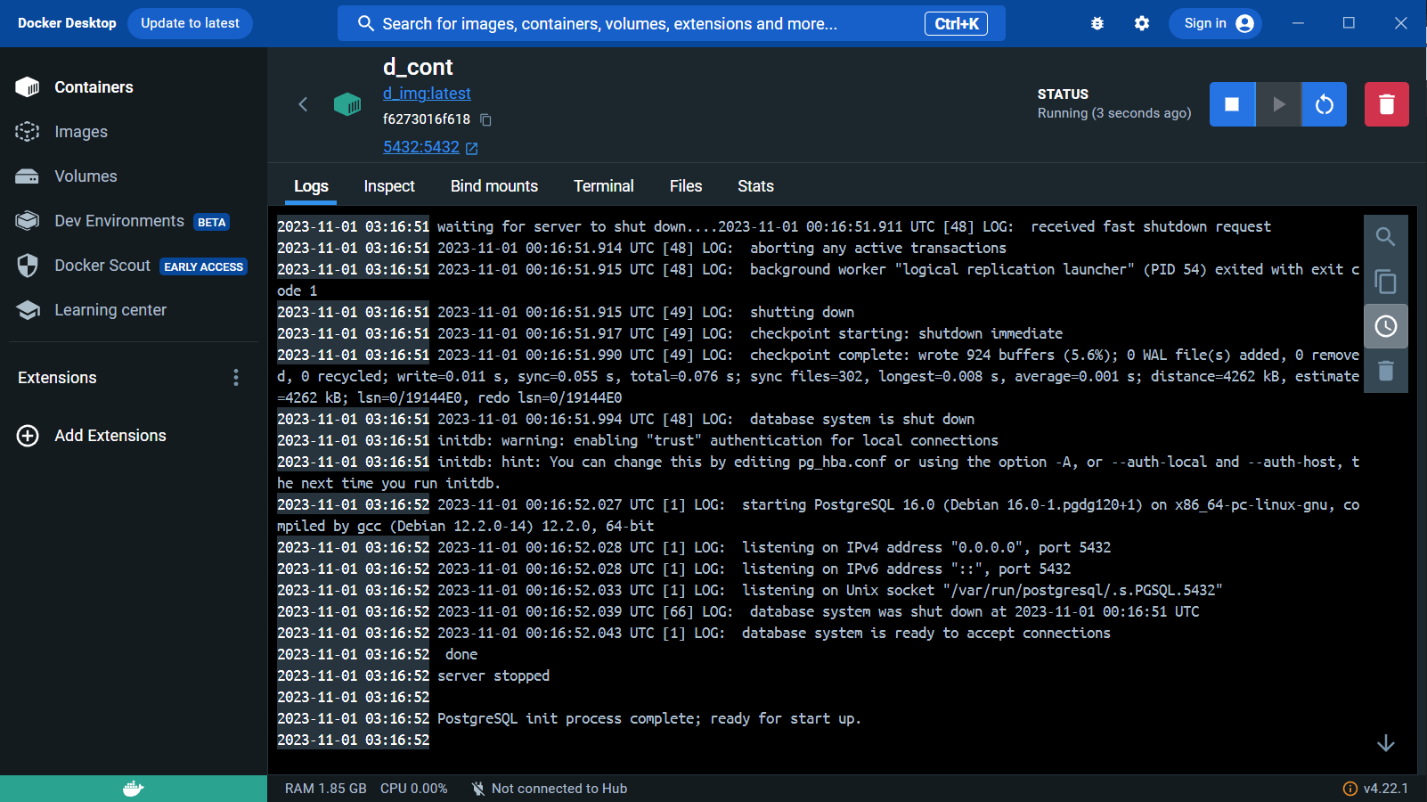


Рисунок 6 – Контейнер для Postgres

В установленной ранее программе DBeaver была просмотрена созданная с помощью sql-скрипта база данных в контейнере Docker, рисунок 7.

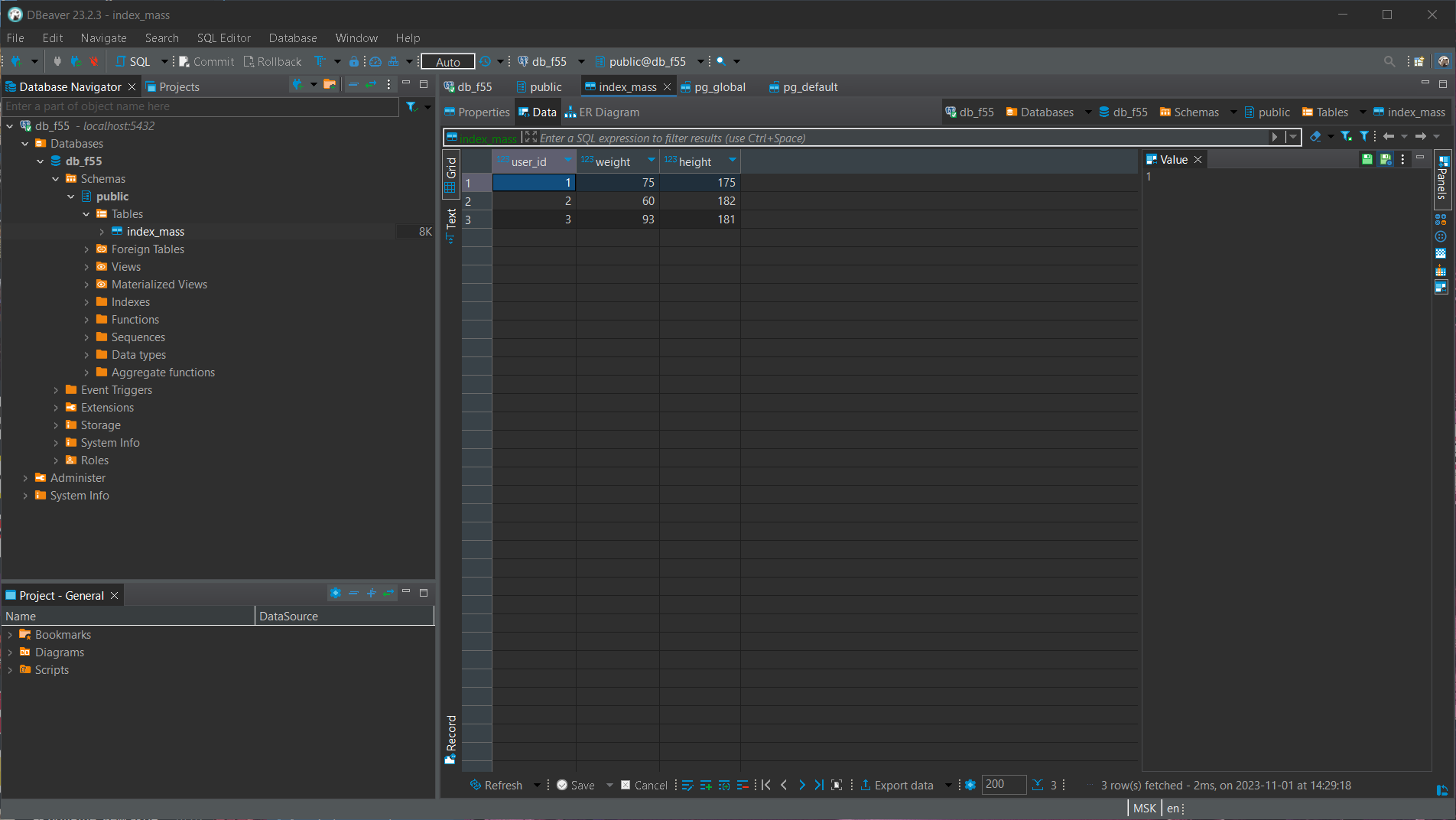


Рисунок 7 – Просмотр базы данных в DBeaver

Далее, с помощью команды «docker exec -it d\_cont psql -d db\_f55 -U f55», было выполнено подключение к интерфейсу «psql» контейнера «d\_cont». После этого, появится возможность взаимодействовать с СУБД напрямую, например, просмотреть список всех имеющихся баз данных, рисунок 8.

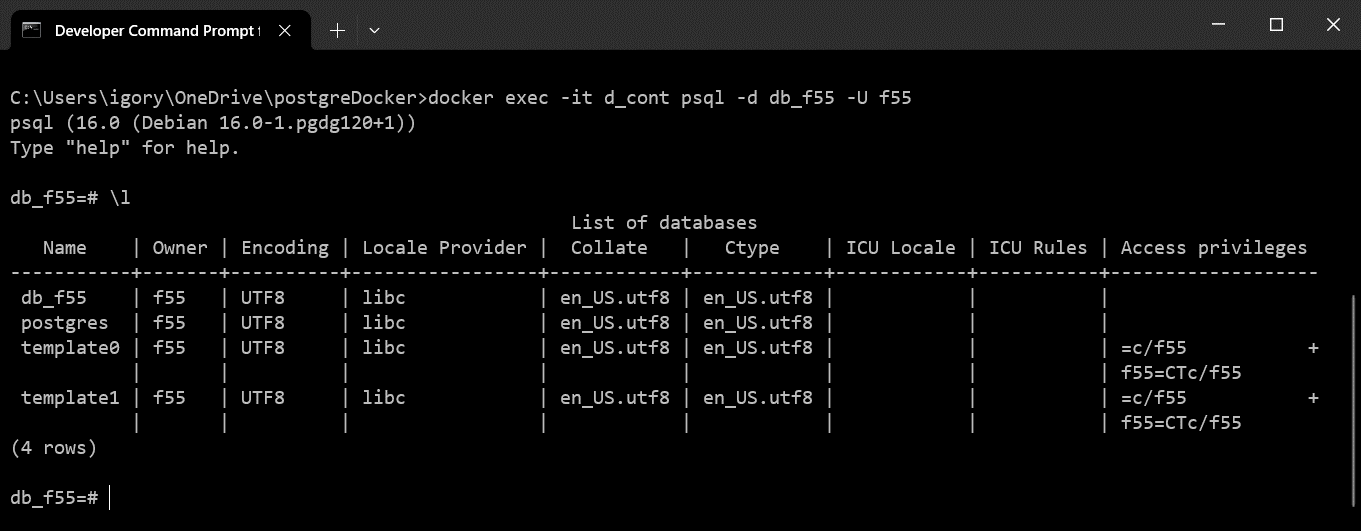


Рисунок 8 – Просмотр баз данных через интерфейс «psql»

Команда "docker exec -it d\_cont psql -d db\_f55 -U f55" выполняет следующие действия:

docker exec – это команда Docker для выполнения команды внутри контейнера;

-it – эти флаги означают "интерактивный терминал". Они позволяют взаимодействовать с командой внутри контейнера через терминал;

d\_cont – это имя или идентификатор контейнера, в котором вы хотите выполнить команду;

psql – это команда для взаимодействия с PostgreSQL базой данных через командную строку;

-d db\_f55 – этот флаг указывает имя базы данных, к которой будет происходить подключение;

-U f55: Этот флаг указывает имя пользователя, с которым вы хотите подключиться к базе данных.

Далее, в Dockerfile были добавлены изменения, которые позволят создать том для хранения информации за пределами контейнера, рисунок 9.

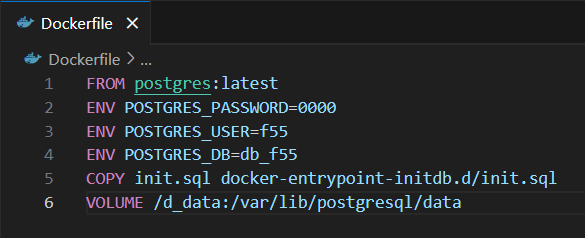


Рисунок 9 – Изменённый Dockerfile

Далее, было выполнено удаление имеющегося контейнера «d\_cont» и было выполнено создание нового контейнера «d\_cont» с монтированием, созданного ранее, тома для хранения данных, рисунок 10.

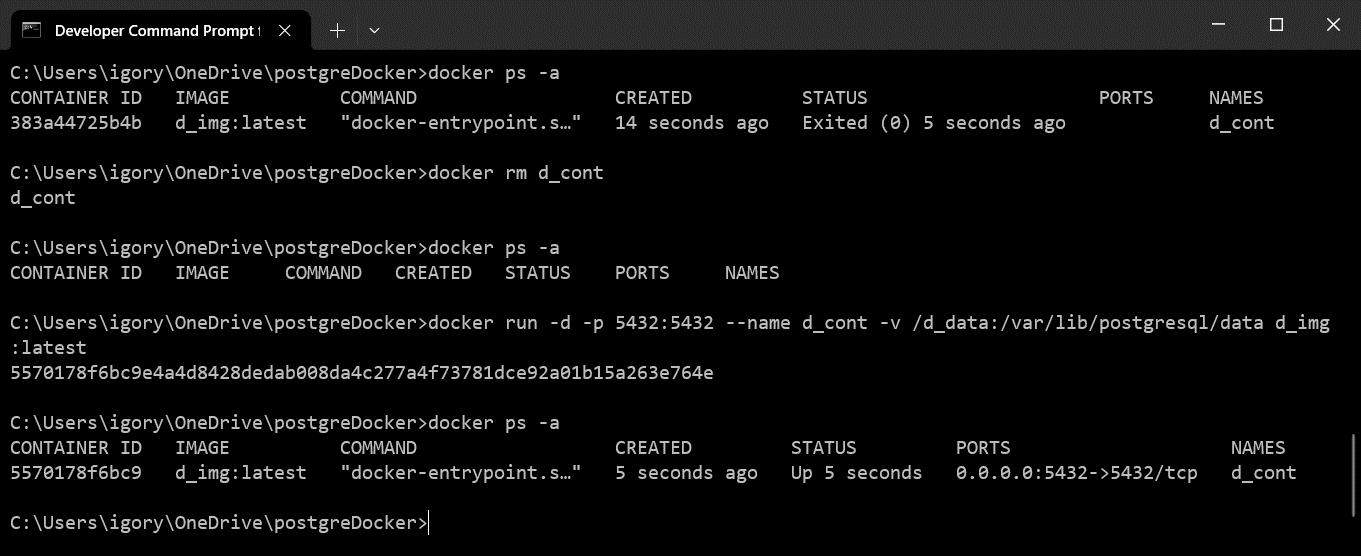


Рисунок 10 – Удаление и создание нового контейнера с монтированием тома

Команда «docker run -d -p 5432:5432 --name d\_cont -v /d\_data:/var/lib/postgresql/data d\_img:latest» уже была описана ранее. Она отличается лишь тем, что в данной команде опцией «-v» выполняется монтирование тома с хоста внутрь контейнера, для возможности сохранять данных за пределы контейнера. Это позволит, в случае удаления контейнера сохранить информацию.

Далее, был создан файл «cmd.txt», где были описаны все команды, которые необходимо использовать для развертывания базы данных PostgreSQL, с помощью Dockerfile, рисунок 11.

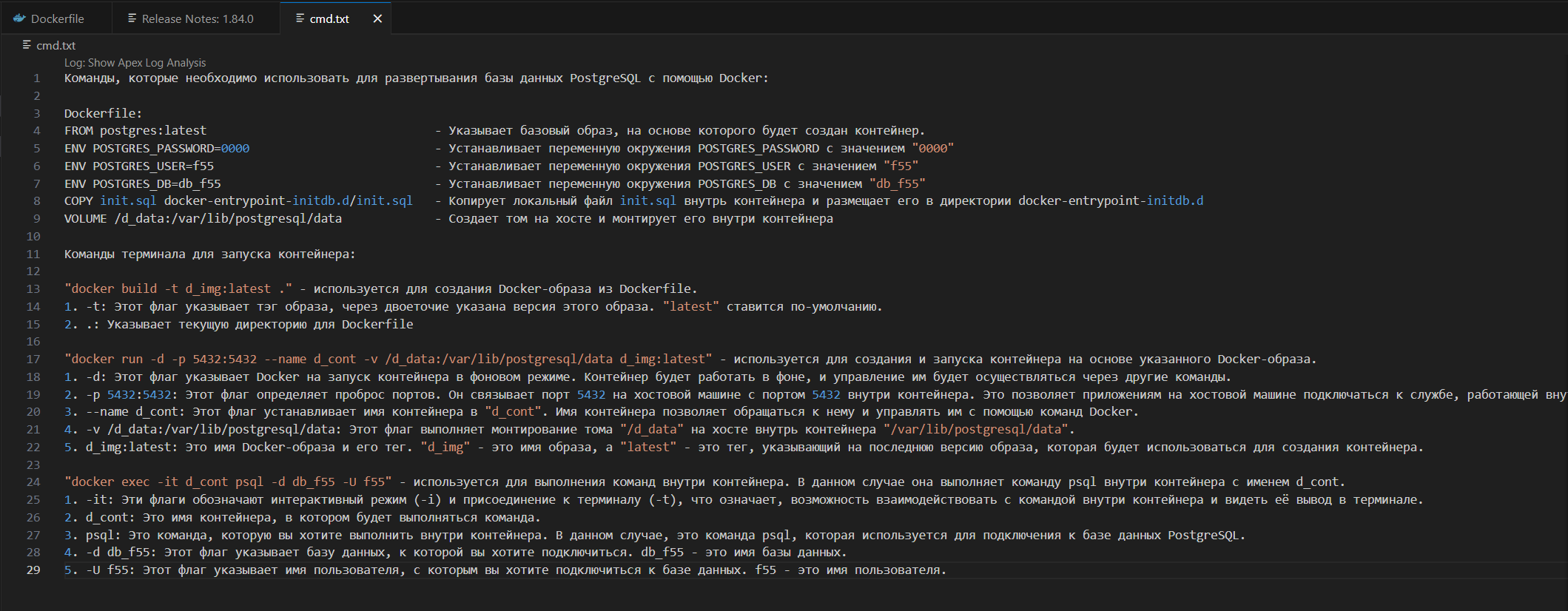


Рисунок 11 – Описания команд для развертывания баз данных PostgreSQL

Далее, было выполнено создание контейнера с использованием файла  
«docker-compose.yml» (рисунок 12, рисунок 13, рисунок 14, рисунок 15).

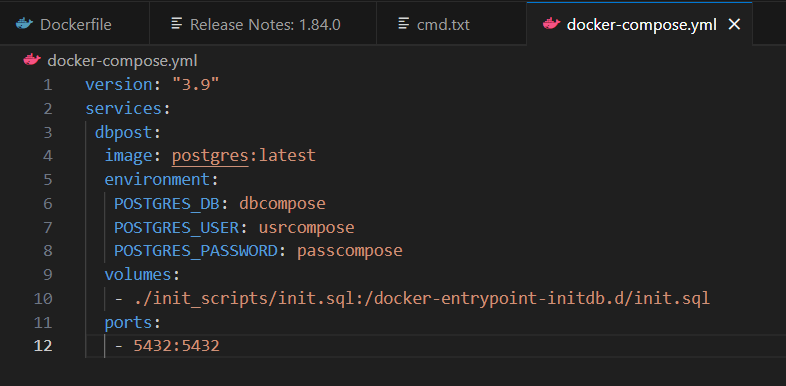


Рисунок 12 – Файл docker-compose.yml

Файл docker-compose.yml представляет собой описание структуры и параметров Docker-контейнеров и их конфигурацию. Подробнее:

- version: "3.9": это строка, которая указывает версию формата Docker Compose. В данном случае, используется версия "3.9" формата. Эта версия определяет структуру и параметры файла.

- services: тот ключевой раздел определяет сервисы, которые будут запущены с использованием Docker Compose. В вашем случае, определен только один сервис с именем dbpost.

- dbpost: это название вашего сервиса. Оно также используется как имя контейнера, когда он будет создан.

- image: postgres:latest: Эта строка определяет образ Docker, который будет использоваться для создания контейнера. В данном случае, используется образ PostgreSQL с тегом latest, что означает последнюю доступную версию образа.

- environment: этот раздел определяет переменные среды, которые будут доступны внутри контейнера. Ваши настройки для базы данных PostgreSQL определены здесь, включая имя базы данных, имя пользователя и пароль.

- POSTGRES\_DB: dbcompose: это определяет имя базы данных, которое будет создано при запуске контейнера PostgreSQL. В данном случае, база данных будет называться dbcompose.

- POSTGRES\_USER: usrcompose: это определяет имя пользователя, который будет создан при запуске контейнера PostgreSQL. В данном случае, пользователь будет называться usrcompose.

- POSTGRES\_PASSWORD: passcompose: это определяет пароль для пользователя PostgreSQL, созданного в контейнере. В данном случае, пароль установлен как passcompose.

- volumes: этот раздел определяет монтирование томов (файловой системы) в контейнер. Здесь определен один том, который монтируется из локальной директории ./init\_scripts/init.sql в контейнер в /docker-entrypoint-initdb.d/init.sql.: это позволяет выполнить инициализацию базы данных с помощью SQL-скрипта init.sql при запуске контейнера.

- ports: этот раздел определяет мэппинг портов между хостом и контейнером. В данном случае, порт 5432 в контейнере (который обычно используется PostgreSQL) мэппится на порт 5432 на хостовой машине.

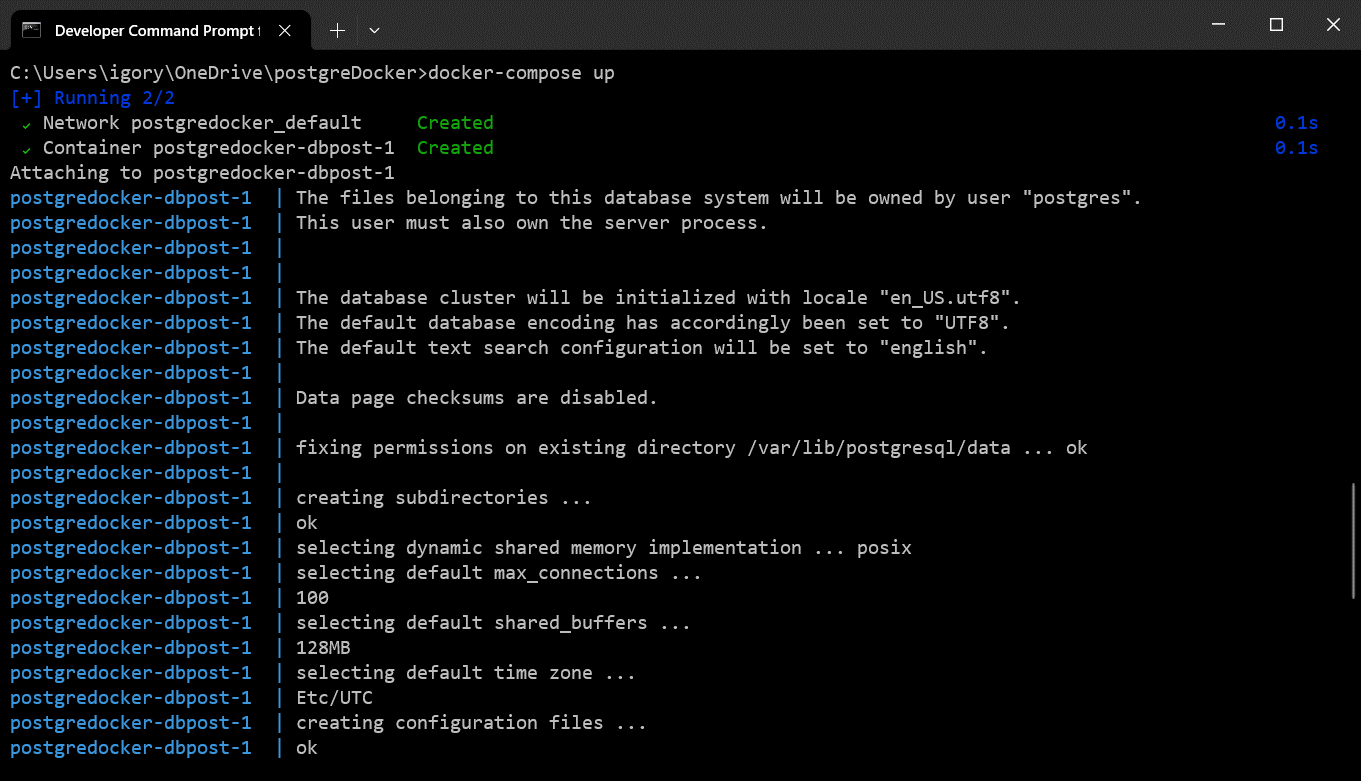


Рисунок 13 – Создание и запуск контейнера, с помощью docker-compose

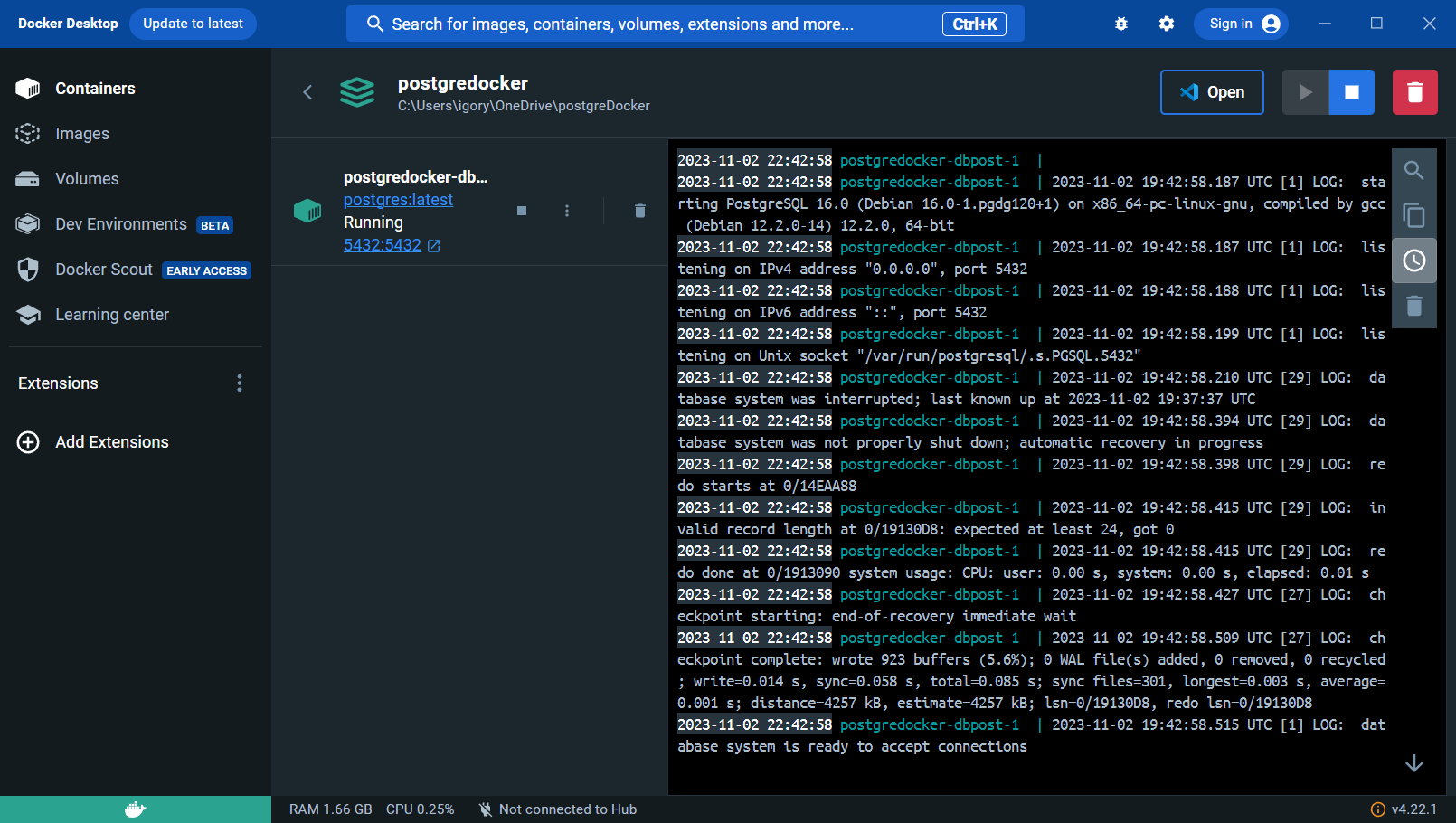


Рисунок 14 – Контейнер с наименованием директории создан

Контрольные вопросы:

1. Что такое Docker?
2. Зачем нужен Docker?
3. Что такое docker-образ?
4. Что такое docker-контейнер?
5. Что такое volume и зачем он нужен?
6. Что такое docker-compose?
7. В чем разница между dockerfile и docker-compose?
8. Какая команда позволяет отправлять различные задания в запущенный докер-контейнер?
9. С помощью какого инструмента можно сохранить важные данные после аварийного отключения контейнера?
10. Docker — это клиент-серверное приложение, которое состоит из двух частей: сервера и клиента. Сервер еще называют Docker-движком. Это фоновый процесс, который непосредственно управляет контейнерами.
11. Docker используется для создания, развертывания и запуска приложений с использованием контейнеров. Он позволяет разработчикам упаковывать приложение и его зависимости в контейнер, который может быть легко перенесен на любую систему, поддерживающую Docker. Это обеспечивает консистентность среды между разработкой и продакшеном, упрощает процесс развертывания и помогает изолировать приложения друг от друга.
12. Docker-образ — это неизменяемый файл, содержащий исходный код, библиотеки, зависимости, инструменты и другие файлы, необходимые для запуска приложения.
13. Docker-контейнер — это исполняемый экземпляр образа Docker. Контейнеры обычно изолированы друг от друга, хотя тома и сети могут позволять им взаимодействовать14.
14. Volumes или же тома – это технология, которая позволяет сохранить данные для контейнера даже после его перезапуска. Это часто бывает необходимым. За счет томов вы можете хранить как значения из базы данных, так и всевозможные файлы, документы и прочее.
15. Docker Compose - инструмент для определения и управления приложениями с множеством контейнеров на платформе Docker. Он позволяет объединять несколько контейнеров вместе и определять все необходимые настройки, зависимости и связи между ними в файле конфигурации.
16. Dockerfile используется для создания образа Docker. Он содержит набор инструкций, которые описывают, как создать образ. С другой стороны, docker-compose используется для определения и запуска многоконтейнерных приложений Docker. Он позволяет определить несколько контейнеров, которые должны быть запущены вместе, а также настройки сети и томов.
17. Команда docker exec позволяет отправлять различные задания в запущенный докер-контейнер. Эта команда требует дополнительного поиска информации.
18. Вопрос о сохранении важных данных после аварийного отключения контейнера требует дополнительного поиска информации.

Вывод: в ходе лабораторной работы были получены основные навыки работы с Docker, а также опыт развертывания базы данных PostgreSQL в контейнере Docker.