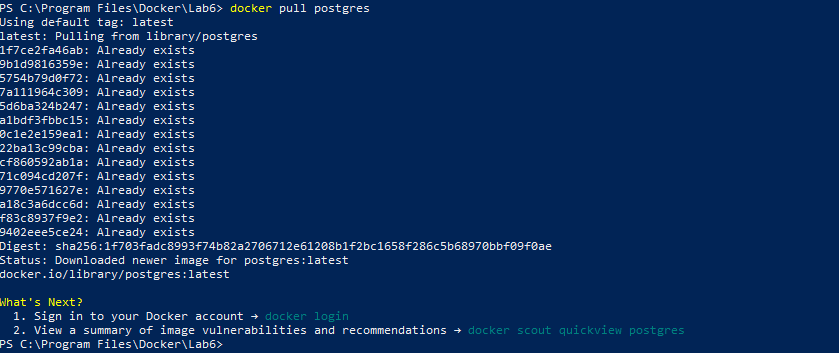
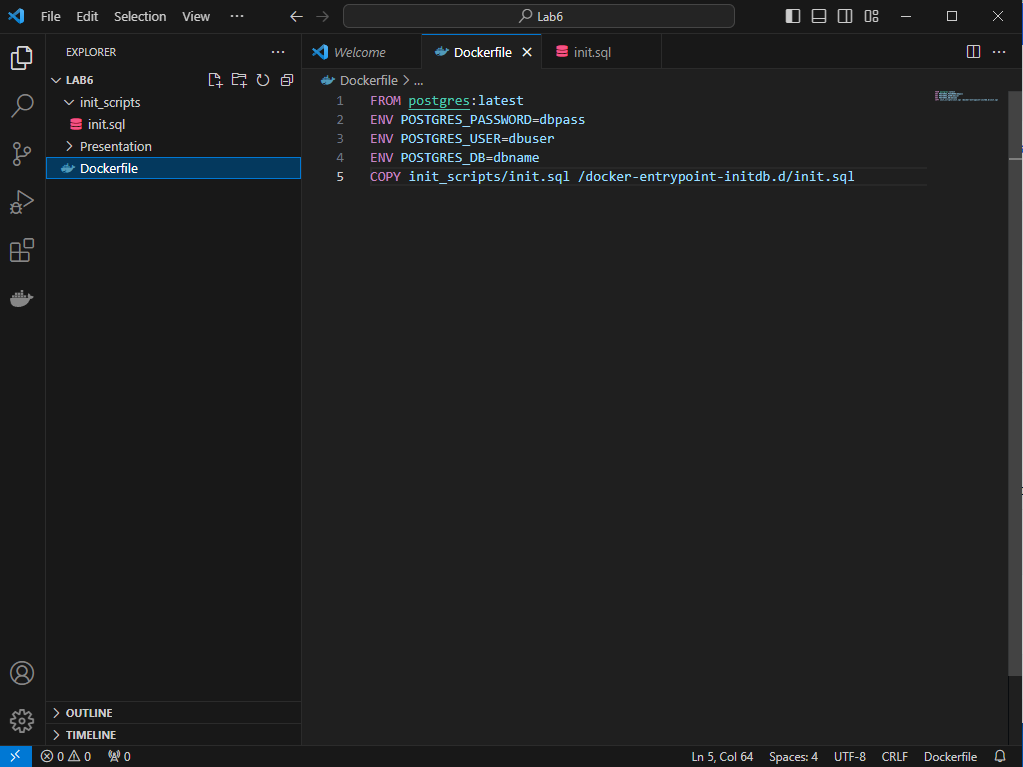
Киселев Андрей Артурович мИИВТ-231

Лабораторная работа №6 “Развёртывание СУБД Postgres с использованием средств автоматизации развёртывания и управления приложениями”

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

Скачиваем образ, в powershell.

1. Создать Dockerfile и init.sql



Этот код представляет собой часть Dockerfile, который создает пользовательский образ Docker на основе официального образа PostgreSQL. Каждая строка в этом Dockerfile выполняет определенную инструкцию:

FROM postgres:latest

Эта строка указывает базовый образ для создания вашего пользовательского образа. postgres:latest означает, что будет использоваться последняя версия официального образа PostgreSQL из Docker Hub.

ENV POSTGRES\_PASSWORD=dbpass

Устанавливает переменную среды POSTGRES\_PASSWORD внутри создаваемого образа, которая задает пароль для пользователя базы данных. В вашем случае пароль устанавливается как dbpass.

ENV POSTGRES\_USER=dbuser

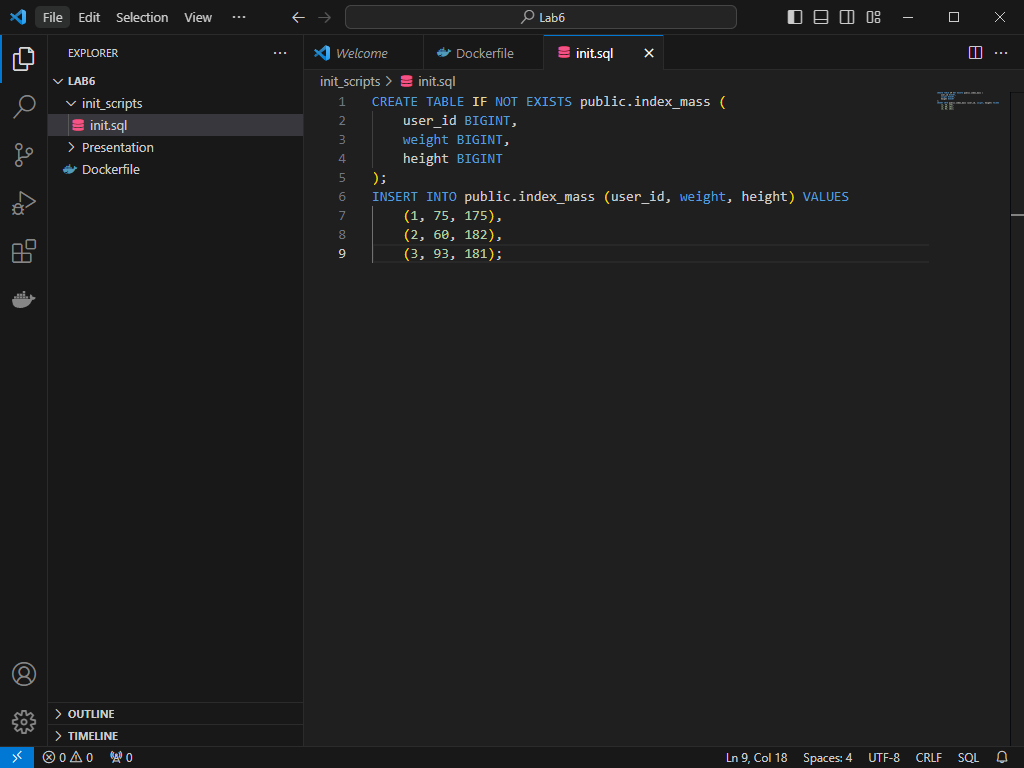
Устанавливает переменную среды POSTGRES\_USER внутри образа, которая задает имя пользователя, который будет использоваться для входа в базу данных PostgreSQL. Имя пользователя в данном случае — dbuser.

ENV POSTGRES\_DB=dbname

Устанавливает переменную среды POSTGRES\_DB для создания начальной базы данных при запуске контейнера. Название этой базы данных будет dbname.

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

Копирует файл init.sql из локальной папки init\_scripts внутрь образа, размещая его в папке /docker-entrypoint-initdb.d. PostgreSQL автоматически выполнит все SQL или bash скрипты в этой папке при первом запуске нового контейнера. Это используется для первоначальной инициализации базы данных, например, для создания таблиц или вставки начальных данных.



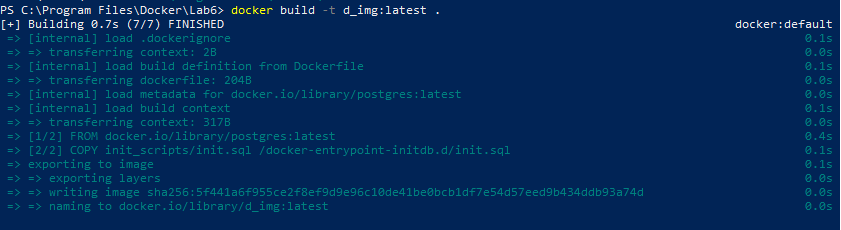
Этот SQL-скрипт содержит две основные команды:

CREATE TABLE IF NOT EXISTS public.index\_mass (user\_id BIGINT, weight BIGINT, height BIGINT);

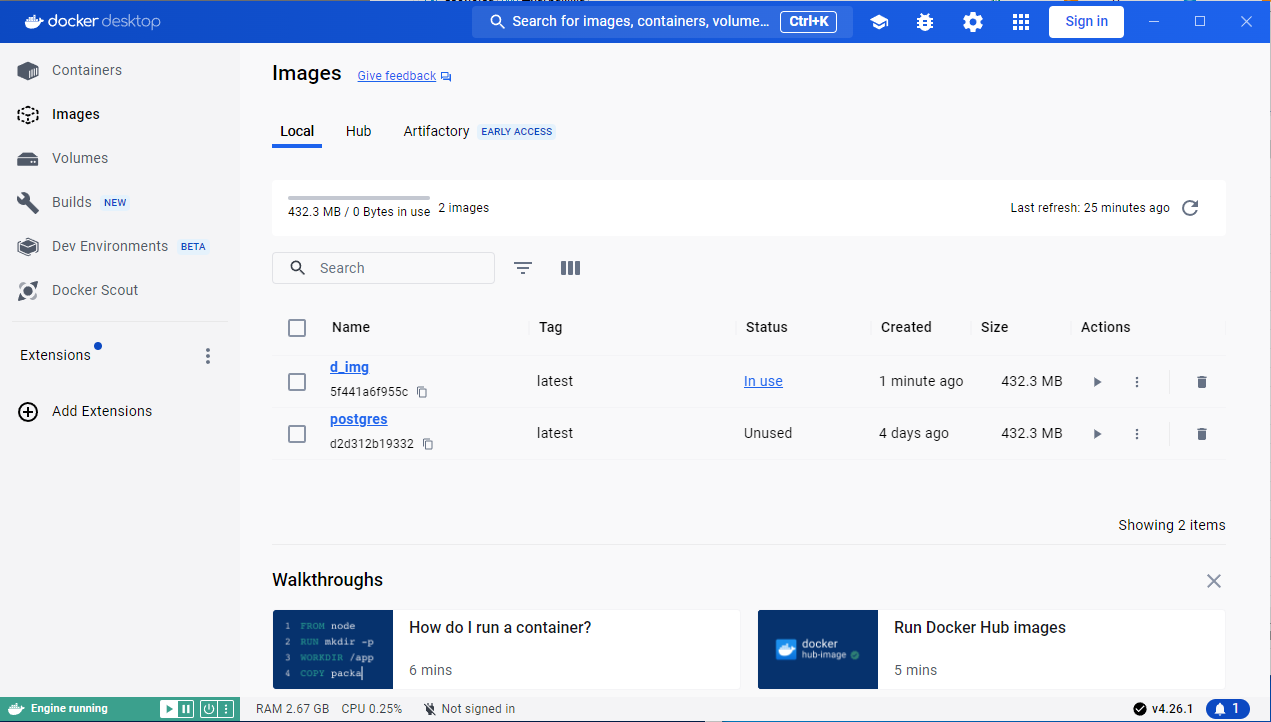
Создает в схеме public новую таблицу с именем index\_mass, если она еще не существует, с тремя столбцами: user\_id, weight, height. Тип данных для каждого столбца — BIGINT, который предназначен для больших целых чисел.

INSERT INTO public.index\_mass (user\_id, weight, height) VALUES (1, 75, 175), (2, 60, 182), (3, 93, 181);

Добавляет в таблицу index\_mass три новые строки с данными для каждого из столбцов

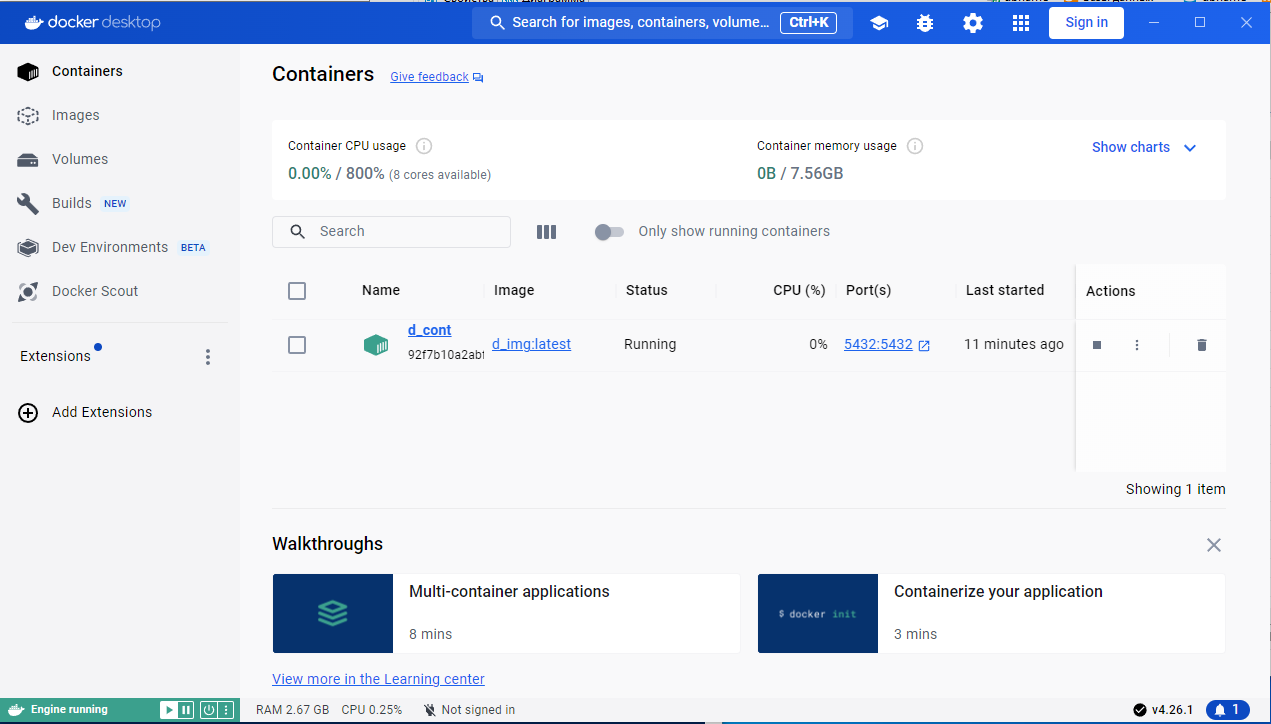
1. Создать образ с помощью команды «docker build -t d\_img:latest .»

Выполняем команду и получаем образы в Docker:



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»



Выполняем команду и получаем работающий контейнер:

Команда docker run -d -p 5432:5432 --name d\_cont d\_img:latest использует Docker для запуска нового контейнера с определенными параметрами:

docker run: Основная команда для создания и запуска контейнера Docker на основе образа.

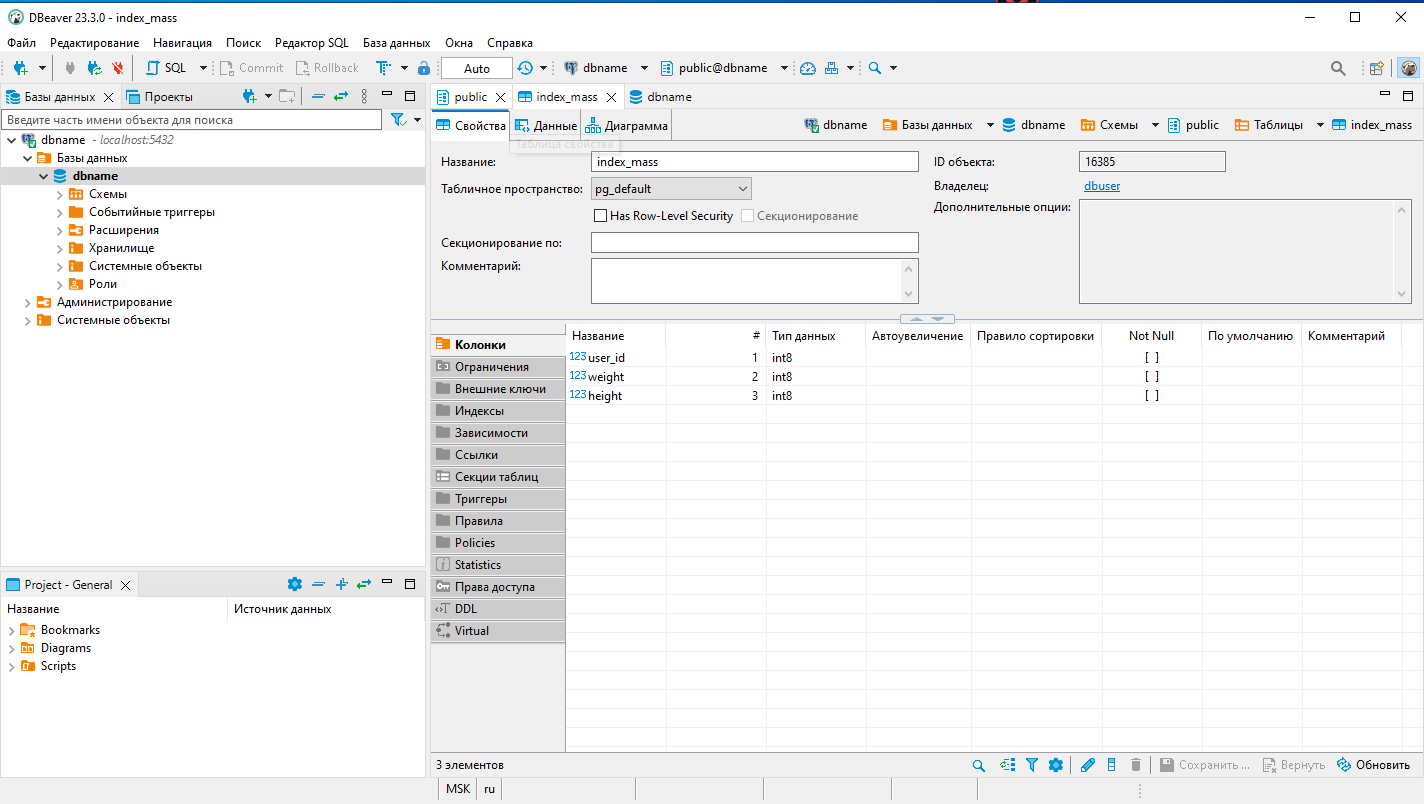
-d: Запускает контейнер в режиме detached (отсоединенном или фоновом), что означает, что контейнер будет работать в фоновом режиме и не будет блокировать терминал командной строки.

-p 5432:5432: Опция для проброса портов. Это говорит Docker создать маппинг порта между хост-машины и контейнером. В данном случае порт 5432 хоста будет направлен на порт 5432 внутри контейнера, что является стандартным портом для серверов PostgreSQL.

--name d\_cont: Задает имя запускаемого контейнера как d\_cont, что позволяет легко ссылаться на него при выполнении других команд Docker.

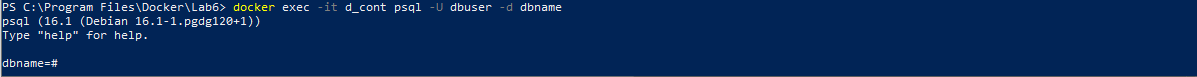
d\_img:latest: Определяет, какой образ Docker использовать для создания контейнера. В данном случае это образ d\_img с тегом latest, указывающим на последнюю версию этого образа.

1. Проверяем работу в DBeaver



1. 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»

Немного исправляем команду и выполняем:



Видим успешное подключение.

Команда docker exec -it d\_cont psql -U dbuser -d dbname используется для выполнения команды внутри работающего контейнера Docker:

docker exec: Основная команда для выполнения новой команды внутри запущенного контейнера.

-it: Этот параметр комбинирует две опции:

-i означает "interactive", что гарантирует, что стандартный ввод (STDIN) открыт в контейнере и может использоваться для взаимодействия с командой.

-t аллоцирует псевдо-tty (терминал), что делает возможным, например, нормальное отображение текстового интерфейса команды psql.

d\_cont: Имя контейнера, в котором будет выполнена команда. В данном случае, команда будет выполнена в контейнере с именем d\_cont.

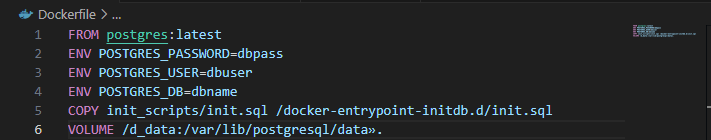
psql: Собственно команда, которую нужно запустить внутри контейнера. psql — это командно-строчный клиент базы данных PostgreSQL.

-U dbuser: Параметр, указывающий psql использовать имя пользователя dbuser для подключения к базе данных.

-d dbname: Задает имя базы данных dbname, к которой будет осуществлено подключение с помощью psql.

Итоговое действие этой команды — выполнение клиента psql в интерактивном режиме внутри контейнера d\_cont для подключения к базе данных dbname, используя учетные данные пользователя dbuser.

1. Создаем том, редактируем Dockerfile

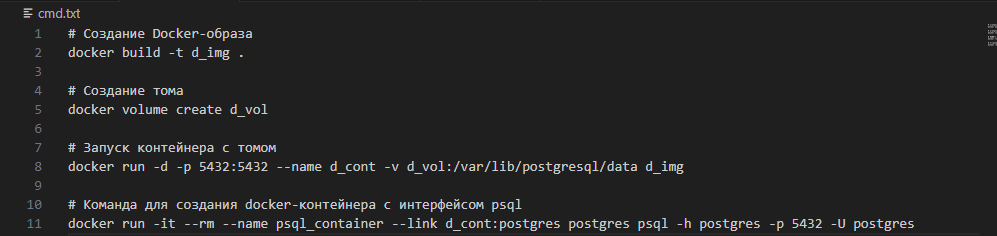


1. Создаем новый контейнер

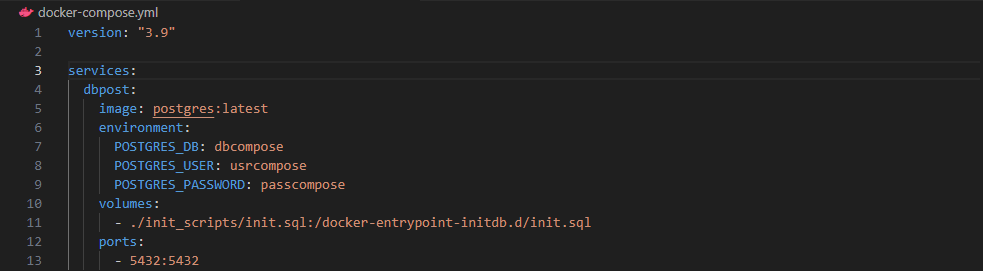


В Docker, параметр -v или --volume используется для монтирования тома (volume) или каталога файловой системы хоста внутри контейнера. Он создает точку монтирования с указанным путем и эффективно связывает каталог на хосте с каталогом внутри контейнера. Это используется для сохранения данных вне контейнеров или для предоставления данных в контейнера.

1. Создаем текстовый файл для задания



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



Приведенный код представляет файл docker-compose.yml в формате YAML, который определяет Docker-контейнер для базы данных PostgreSQL.

Краткое описание этого кода:

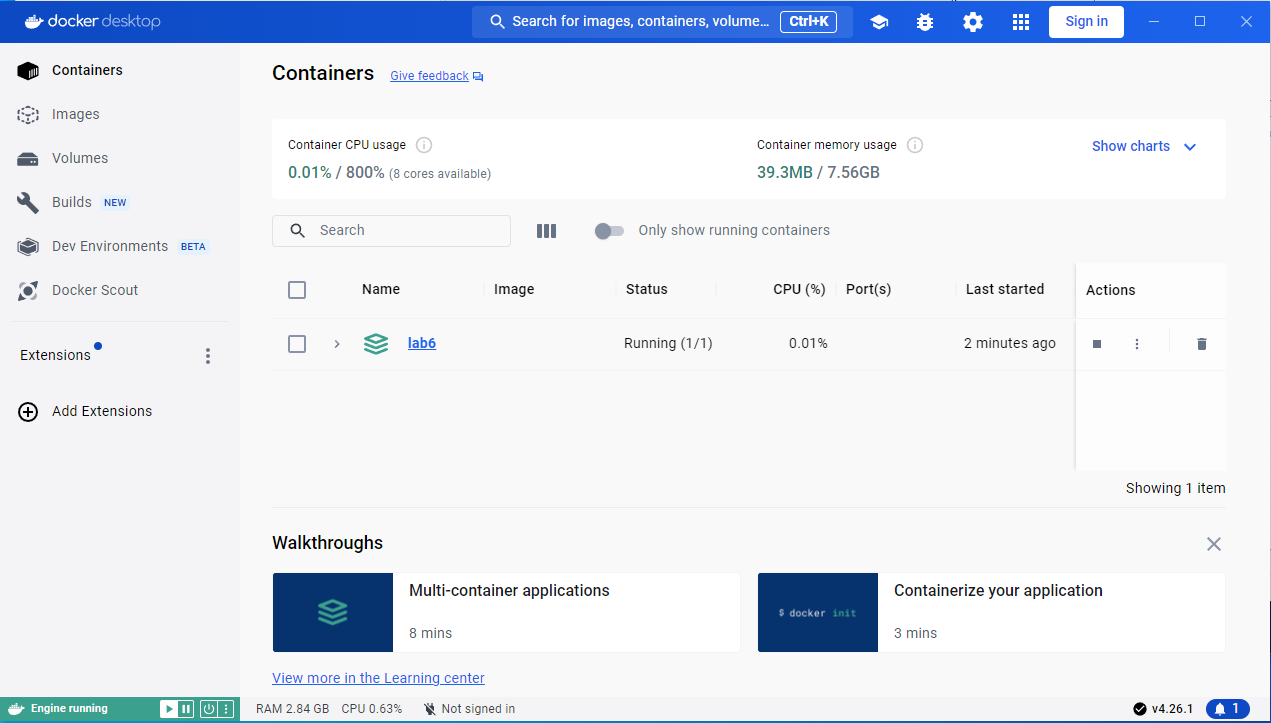
Версия Docker Compose: 3.9.

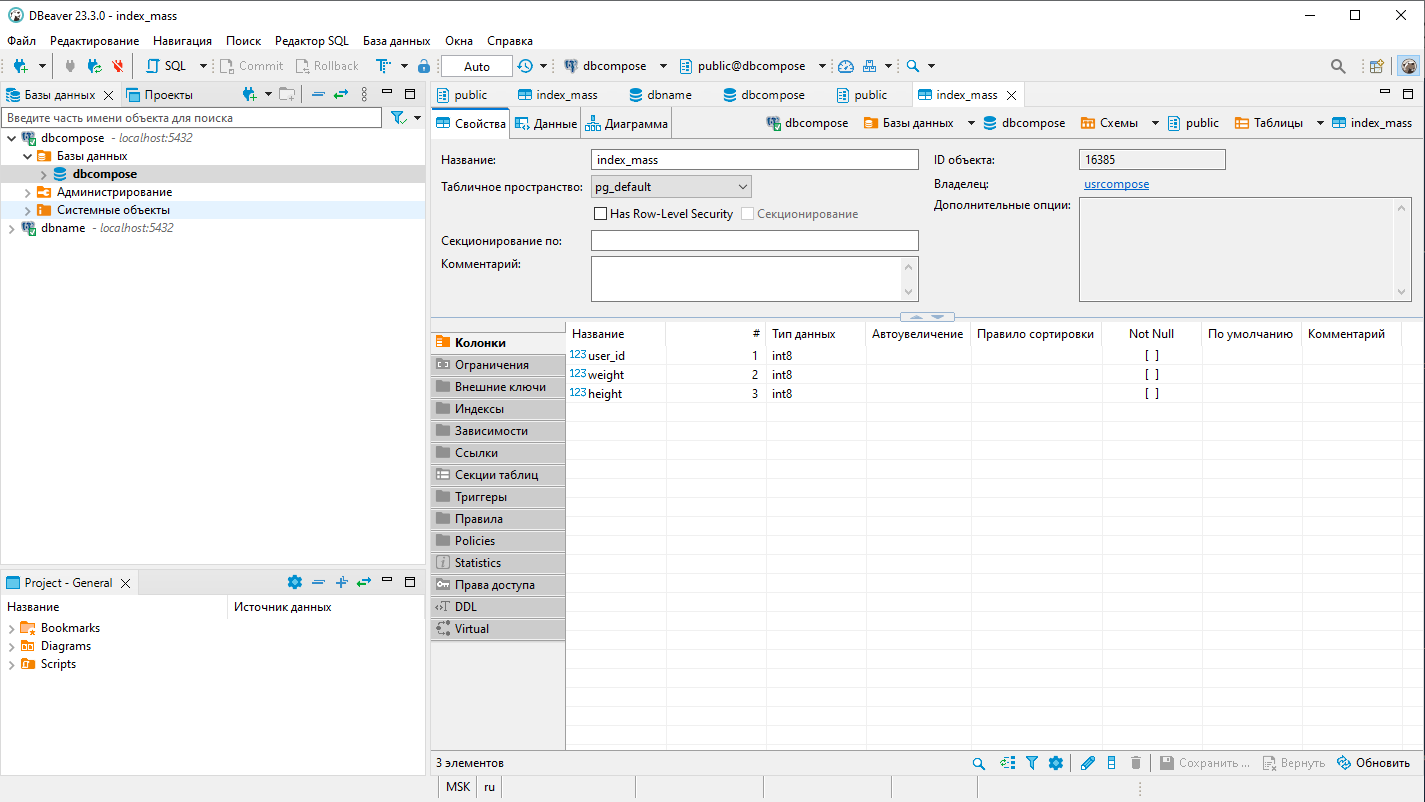
Определение сервиса dbpost, который будет использовать образ postgres:latest.

Определение переменных окружения для контейнера базы данных, включая имя базы данных (POSTGRES\_DB), имя пользователя (POSTGRES\_USER) и пароль (POSTGRES\_PASSWORD).

Монтирование файла init.sql из хост-системы в контейнер, чтобы инициализировать базу данных. Файл init.sql будет выполнен при запуске контейнера благодаря использованию специального каталога внутри образа.

Проброс портов, где порт 5432 на хост-системе будет связан с портом 5432 внутри контейнера.

Запускаем контейнер с помощью команды docker-compose up:



Проверяем и видим что все успешно.  
Контрольные вопросы:

1) Docker - это открытая платформа, которая позволяет автоматизировать развертывание, масштабирование и управление приложениями с использованием контейнеризации. Она создает изолированные контейнеры, которые включают все необходимое для запуска приложений, включая код, системные инструменты, библиотеки и настройки.

2) Docker предоставляет множество преимуществ, таких как:

- Повышение портативности: Docker контейнеры обеспечивают изолированную среду, которая позволяет запускать приложения на различных платформах и операционных системах без необходимости настройки окружения каждый раз.

- Упрощение развертывания: Docker обеспечивает единое и консистентное развертывание приложений, упрощая процесс установки и конфигурирования приложений.

- Увеличение масштабируемости: Docker позволяет горизонтальное масштабирование приложений путем запуска нескольких контейнеров, распределенных на нескольких узлах.

- Улучшение изоляции: Контейнеры Docker обеспечивают изоляцию приложений, что позволяет им работать независимо друг от друга и предотвращает конфликты между зависимостями и конфигурациями.

3) Docker-образ - это легковесная и автономная выполняемая единица, содержащая все необходимое для запуска приложений, включая код, среду выполнения, библиотеки, зависимости и настройки. Docker-образы создаются из Docker-файлов путем выполнения инструкций, описывающих, как собрать образ.

4) Docker-контейнер - это экземпляр Docker-образа. Он представляет собой сверхлегкий и изолированный исполняемый контейнер, который содержит все необходимое для запуска приложения. Каждый контейнер работает в изолированной среде, имеет собственное файловое пространство и ресурсы, и может быть запущен, остановлен и удален независимо от других контейнеров.

5) Volume (том) в Docker - это механизм для сохранения и управления данными Docker-контейнера. Он представляет собой точку монтирования, которая позволяет контейнеру сохранять данные вне его файловой системы и делиться данными между контейнерами. Volumes обеспечивают сохранение данных даже после удаления или перезапуска контейнера, что делает их полезными при работе с постоянными данными, такими как базы данных или файлы журналов.

6) Docker Compose - это инструмент, который позволяет определить и управлять множеством Docker-контейнеров, объединяя их в единую службу. Файл `docker-compose.yml` используется для определения параметров всех контейнеров, их зависимостей, портов и других настроек. Docker Compose позволяет легко развертывать и масштабировать сложные приложения, используя определенные конфигурации.

7) Основная разница между Dockerfile и docker-compose заключается в их целях и областях применения:

- Dockerfile - это текстовый файл, который содержит набор инструкций для сборки Docker-образа. Он используется для описания, как собрать и настроить образ. Dockerfile предоставляет гибкость и контроль над процессом сборки образа, позволяя определить все необходимые зависимости и настройки.

- Docker Compose - это инструмент для определения и управления множеством Docker-контейнеров. Он использует файл `docker-compose.yml`, чтобы определить параметры и зависимости всех контейнеров, объединяя их в составную службу. Docker Compose предоставляет более высокий уровень абстракции и позволяет определить композицию и оркестрацию служб, что упрощает развертывание и управление сложными приложениями.

8) Для отправки заданий в запущенный Docker-контейнер можно использовать команду `docker exec`. Синтаксис команды: `docker exec [OPTIONS] CONTAINER COMMAND [ARG...]`. Например, чтобы выполнить команду `ls -l` в контейнере `my\_container`, можно использовать следующую команду: `docker exec my\_container ls -l`.

9) Для сохранения важных данных после аварийного отключения контейнера можно использовать Docker Volume. Docker Volume позволяет создавать и использовать тома, которые представляют собой отдельное хранение данных, независимо от жизненного цикла контейнера. Тома могут быть подключены к контейнеру, чтобы хранить и обрабатывать данные, и все изменения в томах будут сохранены даже после перезапуска или удаления контейнера. Это обеспечивает сохранение и восстановление важных данных при работе с контейнерами Docker.