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 Костюков К.А.

Принял: Короленко В.В.

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

**Основные задачи:**

* 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

**Ход работы:**

Сначала был изучен сайт https://www.docker.com, а также документация по docker (https://docs.docker.com), далее был зарегистрирован профиль на Docker Hub и был установлен на ОС Windows 10 Docker Desktop (Рисунок 1).

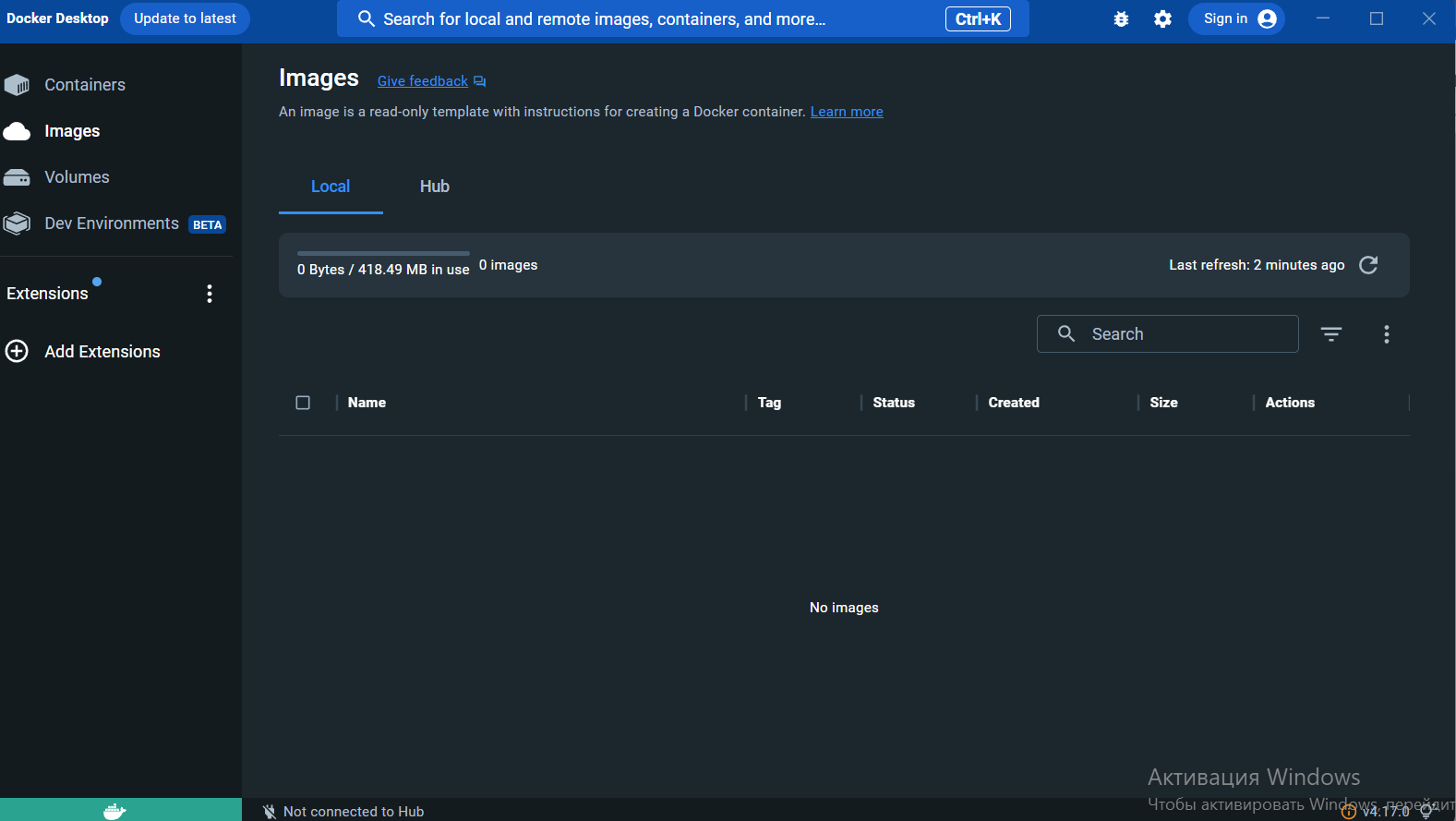
****

Рисунок 1 – Docker Desktop

Теперь необходимо установить на систему DBeaver - это бесплатная программа для работы с СУБД. С её помощью можно создавать новые базы, изменять и удалять данные в уже существующих, выполнять SQL-запросы. Установленная программа изображена на рисунке 2.

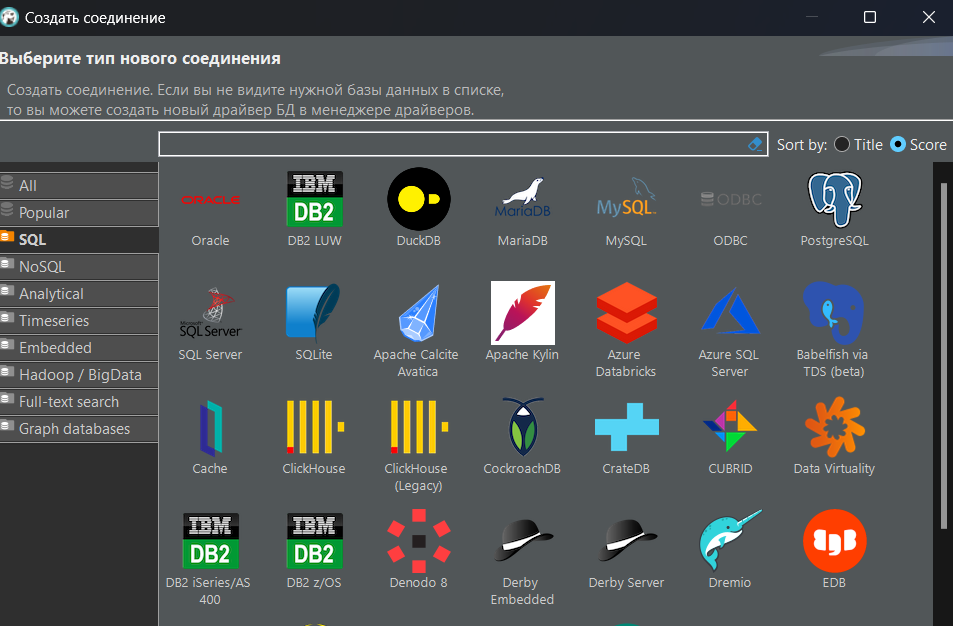
****

Рисунок 2 – DBeaver

Затем был скачан docker-образ с СУБД postgres с помощью команды для консоли (Рисунок 3).

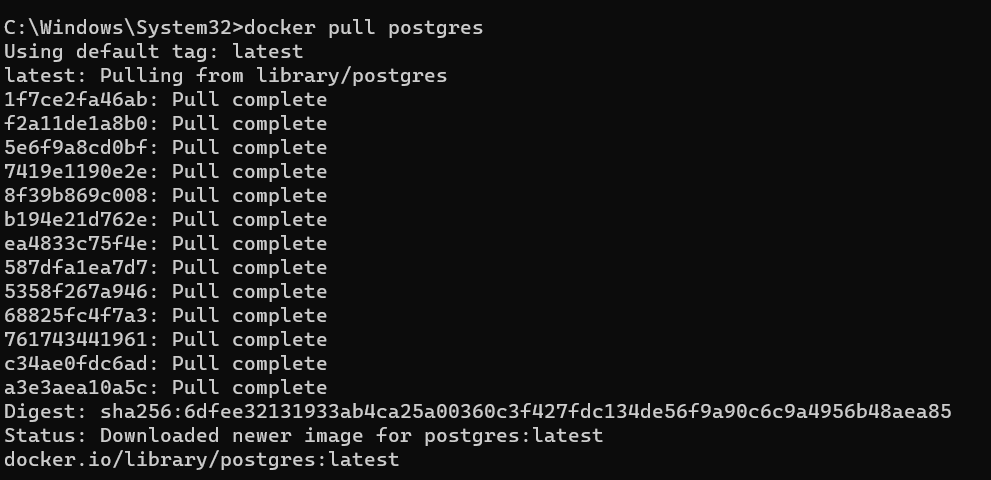
****

Рисунок 3 – Загрузка docker-образа

После загрузки был создан Dockerfile с необходимым содержанием (Рисунок 4), FROM postgres:latest: Указывает базовый образ, который будет использован для создания нового образа. В данном случае, используется официальный образ PostgreSQL с тегом "latest". Это означает, что будет использована последняя версия образа PostgreSQL. ENV POSTGRES\_PASSWORD=dbpass: Устанавливает переменную окружения POSTGRES\_PASSWORD в значение "dbpass". Это будет пароль для пользователя базы данных PostgreSQL. ENV POSTGRES\_USER=dbuser: Устанавливает переменную окружения POSTGRES\_USER в значение "dbuser". Это будет имя пользователя для доступа к базе данных PostgreSQL.   
ENV POSTGRES\_DB=dbname: Устанавливает переменную окружения POSTGRES\_DB в значение "dbname". Это будет имя базы данных, которая будет создана при инициализации контейнера. COPY init\_scripts/init.sql /docker-entrypoint-initdb.d/init.sql: Копирует SQL-скрипт init.sql из локальной директории init\_scripts/ внутрь контейнера по пути /docker-entrypoint-initdb.d/init.sql. Поскольку этот путь является стандартным местом для инициализации базы данных в контейнере PostgreSQL, скрипт будет выполнен при первом запуске контейнера, и его содержимое будет использовано для настройки базы данных.

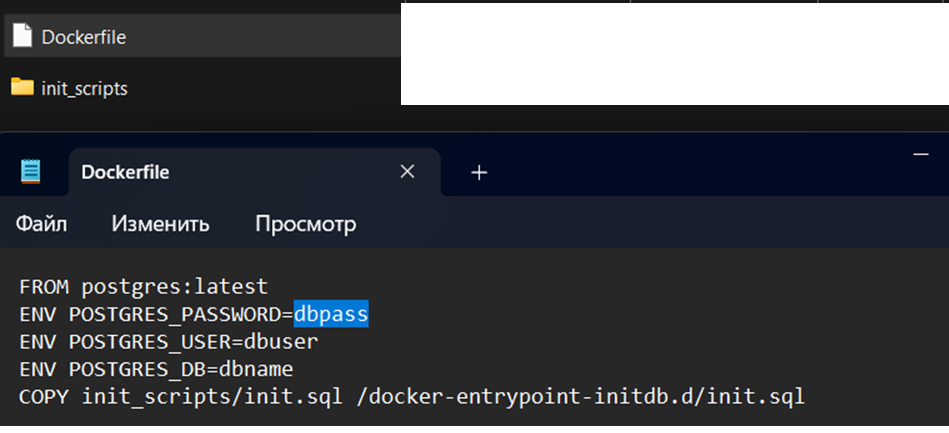
****

Рисунок 4 – Dockerfile

Содержимое файла init.sql изображено на рисунке 5. CREATE TABLE IF NOT EXISTS public.index\_mass: Создает таблицу с именем index\_mass в схеме public. Если такая таблица уже существует, то оператор IF NOT EXISTS позволяет избежать ошибки, если таблица уже была создана. Таблица будет содержать три столбца: user\_id (тип BIGINT), weight (тип BIGINT) и height (тип BIGINT). INSERT INTO public.index\_mass (user\_id, weight, height) VALUES (1, 75, 175), (2, 60, 182), (3, 93, 181) Добавляет три записи в таблицу index\_mass. Каждая запись представляет собой комбинацию значений для столбцов user\_id, weight и height.

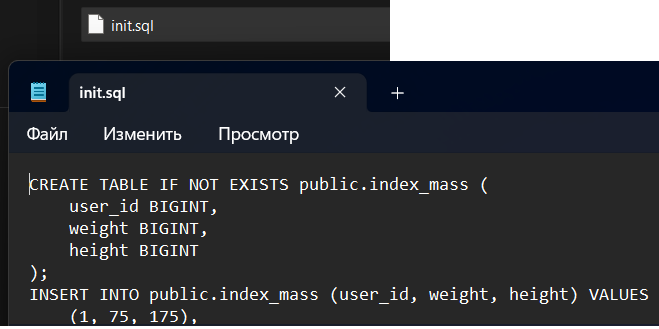
****

Рисунок 5 – init.sql

После этого, с помощью команды «docker build -t d\_img:latest .», запущенной из каталога с Dockerfile, был создан docker-образ «d\_img» (Рисунок 6).

****

Рисунок 6 – Создание docker-образа

Чтобы воспользоваться СУБД был запущен 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» (Рисунок 7). При запуске команды docker run, Docker будет искать образ dimg:latest` в локальном хранилище образов. Если образ не найден локально, он будет загружен из репозитория Docker Hub. После загрузки образа, Docker создаст и запустит контейнер на основе этого образа. Контейнер, созданный с помощью данной команды, будет работать в фоновом режиме, привязывая порт 5432 контейнера к порту 5432 хоста.

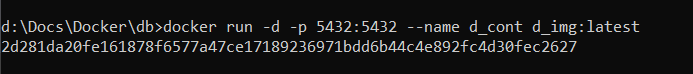
****

Рисунок 7 – Запуск контейнера

Далее в DBeaver была проверена развёрнутая СУБД и подключена база данных с параметрами, указанными в Dockerfile командами (Рисунок 8)**.**

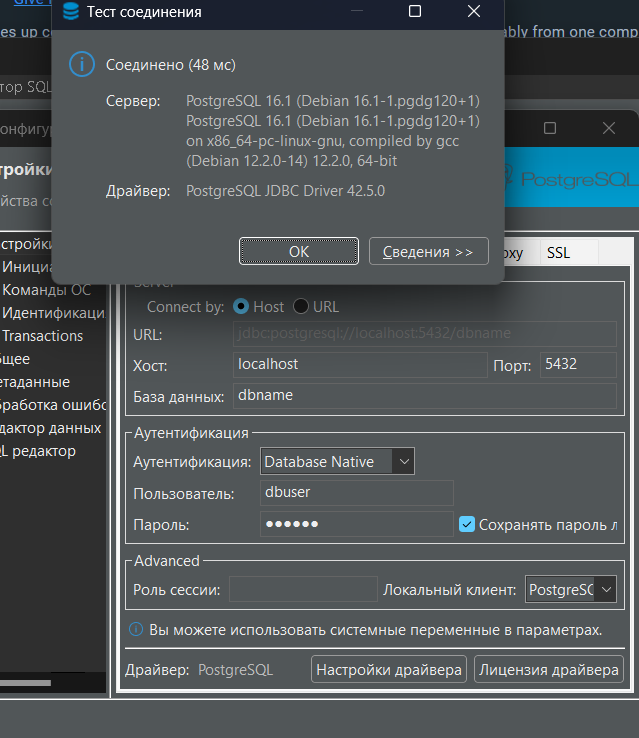
****

Рисунок 8 – Тест соединения

Чтобы была возможность работать с базой данных из консоли, была написана специальная команда «docker exec -it d\_cont psql -U dbuser -d dbname», которая позволит подключаться к работающему контейнеру, запускать интерфейс psql и вносить новые данные (Рисунок 9). Данная команда запускает интерактивную сессию внутри контейнера с именем или идентификатором d\_cont. Затем, внутри контейнера, она запускает утилиту командной строки psql для подключения к базе данных PostgreSQL с помощью указанного имени пользователя и наименования базы данных -d dbname, что позволяет выполнять SQL-запросы или взаимодействовать с базой данных PostgreSQL изнутри контейнера Docker.

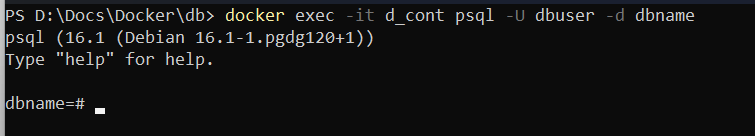


Рисунок 9 – Команда для подключения

Чтобы данные, измененные в процессе работы контейнера, были доступны после удаления контейнера необходимо создать том (volume). Была добавлена строка: «VOLUME /d\_data:/var/lib/postgresql/data», при этом локально будет создан том (volume) в папке «/d\_data» (Рисунок 10).

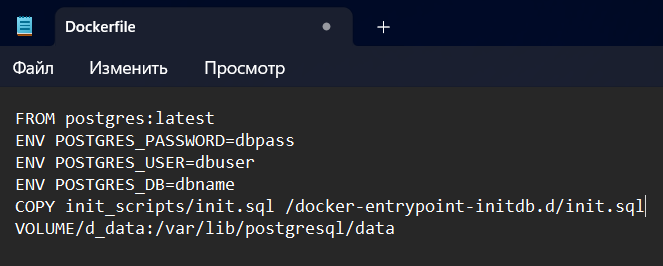


Рисунок 10 – Создание тома

Далее созданный контейнер был удален, и создан новый контейнер с помощью команды: «docker run -d -p 5432:5432 --name d\_cont d\_img:latest -v /d\_data:/var/lib/postgresql/data» (Рисунок 11). Команда, как и прошлая, запускает контейнер на основе образа d\_img:latest с именем d\_cont, привязывает порт 5432 хоста к порту 5432 контейнера, запускает контейнер в фоновом режиме и сохраняет данные в локальной директории /d\_data. При этом используется флаг -v, который привязывает локальную директорию /d\_data к директории контейнера /var/lib/postgresql/data. Таким образом, данные, генерируемые контейнером PostgreSQL, будут сохранены в локальной директории /d\_data.

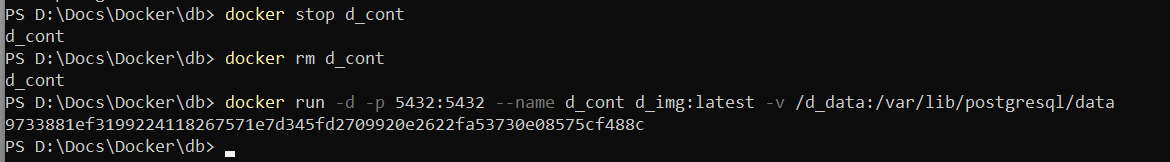


Рисунок 11 – Удаление и создание контейнера

После всего, был создан текстовый файл cmd.txt, в котором описаны все команды, которые необходимо использовать для развертывания базы данных Postgres с помощью Dockerfile (создание образа, запуск контейнера, создание тома, команда для создания docker-контейнера с томом (volume) на основе docker-образа и команда для запуска контейнера с интерфейсом psql для внесения новых данных в БД).

Затем, был создан контейнер с использованием файла docker-compose.yml, что показано на рисунке 12.

В данном файле строка version: "3.9" указывает на версию синтаксиса Docker Compose, которую необходимо использовать для интерпретации файла. В данном случае, используется версия "3.9". Cтрока services: начало секции, в которой определяются сервисы, которые будут созданы при запуске контейнеров. dbpost: название сервиса. В данном случае, это имя, которое мы сами выбрали для сервиса базы данных PostgreSQL. image: postgres:latest указывает образ контейнера, который будет использоваться для создания сервиса. Здесь используется официальный образ PostgreSQL с тегом "latest", что означает использование последней доступной версии. environment определяет переменные окружения, которые будут переданы в контейнер. В данном случае, устанавливаются три переменные окружения для базы данных: POSTGRES\_DB (название базы данных), POSTGRES\_USER (имя пользователя) и POSTGRES\_PASSWORD (пароль пользователя). volumes: указывает монтирование томов (папок или файлов) из хост-системы внутрь контейнера. В данном случае, указывается монтирование файла init.sql из локальной директории ./init\_scripts/ внутрь контейнера по пути /docker-entrypoint-initdb.d/init.sql. Это позволяет инициализировать базу данных при запуске контейнера с использованием SQL-скрипта. ports: указывает маппинг портов между хост-системой и контейнером. Здесь порт 5432 на хосте маппится на порт 5432 внутри контейнера, что позволяет внешним приложениям обращаться к базе данных через этот порт.

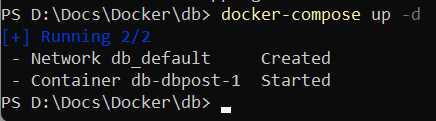


Рисунок 12 – Создание контейнера с файлом docker-compose.yml

**Вывод**: в данной работе были изучены основы работы программного средства для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker на примере решения задачи развёртывания СУБД Postgres.

**Контрольные вопросы**

1. Что такое Docker?

Docker - это открытая платформа, которая используется для автоматизации развёртывания, доставки и управления приложениями. Она позволяет упаковывать приложения и их зависимости в контейнеры, которые могут быть запущены на любой операционной системе.

1. Зачем нужен Docker?

Docker является мощным инструментом, который облегчает разработку, развёртывание и масштабирование приложений, а также упрощает управление инфраструктурой и обеспечивает консистентность в различных средах.

1. Что такое docker-образ?

Docker-образ представляет собой исполняемую версию приложения, которая включает в себя все необходимые зависимости, файлы и конфигурации. Он создается на основе Docker-файла, который содержит инструкции по установке и настройке приложения в контейнере. Docker-образ можно сравнить с шаблоном или основой, на основе которой создаются и запускаются контейнеры.

1. Что такое docker-контейнер?

Docker-контейнер - это запускаемый экземпляр образа Docker. Он представляет собой изолированную среду, в которой запускается приложение и все его зависимости. Контейнеры Docker используют ядро операционной системы хоста, но обеспечивают изолированное пространство для исполнения приложения, которое позволяет снизить влияние одного приложения на другие, обеспечивает консистентность и надежность запуска приложений на различных окружениях.

1. Что такое volume и зачем он нужен?

Volume представляет собой механизм, предоставляемый Docker для управления данными, которые могут быть использованы и обмениваться между контейнерами. Volume в Docker позволяет сохранить данные даже после удаления контейнера. Таким образом, контейнеры могут использовать и обмениваться общими данными, которые хранятся внутри volume.

1. Что такое docker-compose?

Docker Compose - это инструмент, который позволяет определить и запустить несколько контейнеров Docker как единое приложение. Он используется для создания и управления многоконтейнерными приложениями, состоящими из нескольких сервисов.

1. В чем разница между dockerfile и docker-compose?

Dockerfile отвечает за создание образа Docker, а Docker Compose - за управление несколькими контейнерами и их конфигурацией.

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

Для отправки различных заданий в запущенный Docker-контейнер используется команда docker exec.

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

Можно использовать Docker volumes или Docker data containers, чтобы разделить данные от контейнера и сохранить их независимо от жизненного цикла контейнера. Также можно использовать инструменты, подобные Docker Compose, для описания и создания многоконтейнерных приложений, где данные будут сохраняться в отдельных контейнерах или во внешних хранилищах, таких как базы данных или файловые системы.