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 мИИВТ-241: Бокарев С.А.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

Цель лабораторной 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

В начале было проведено ознакомление с ресурсом https://www.docker.com и была изучена документация по Docker, доступная по адресу https://docs.docker.com. Далее, был зарегистрирован аккаунт на Docker Hub и установлен Docker Desktop на операционной системе Windows 10 (рис. 1).

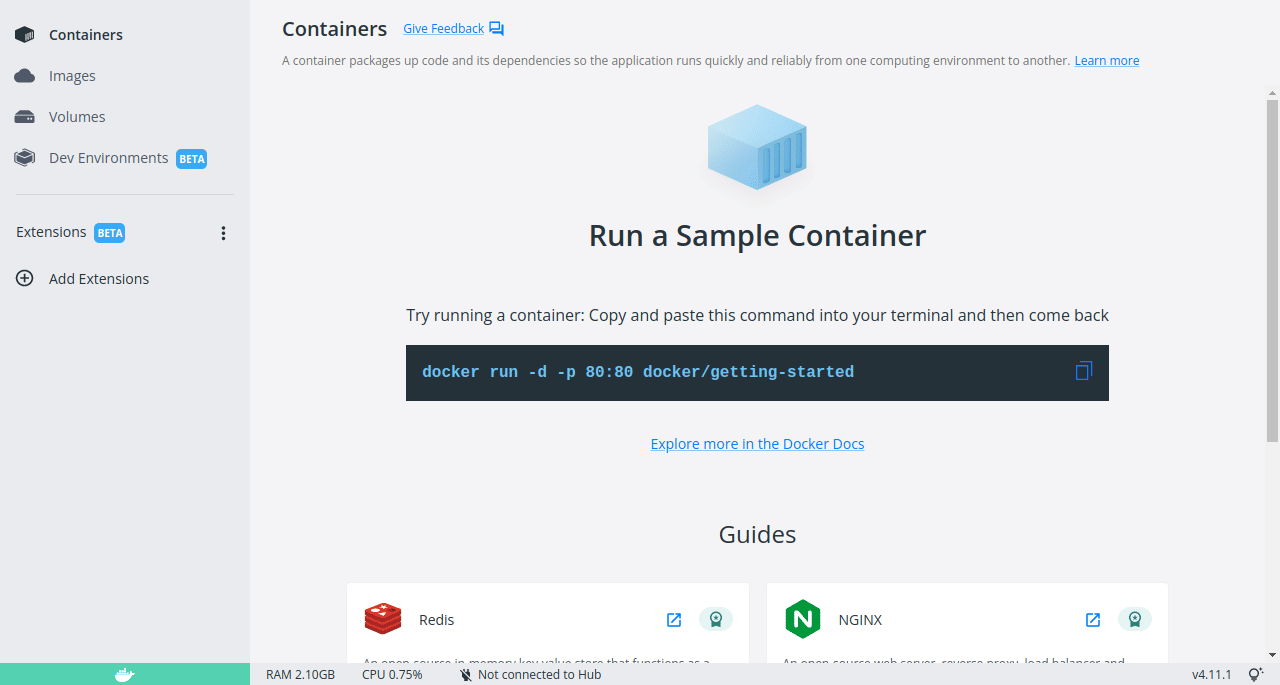


Рисунок 1 – Docker Desktop

Далее нам необходимо было установить DBeaver - бесплатную программу, предназначенную для работы с системами управления базами данных. С помощью DBeaver можно создавать новые базы данных, вносить изменения в существующие данные и выполнять SQL-запросы. Установленное приложение представлено на рисунке 2.

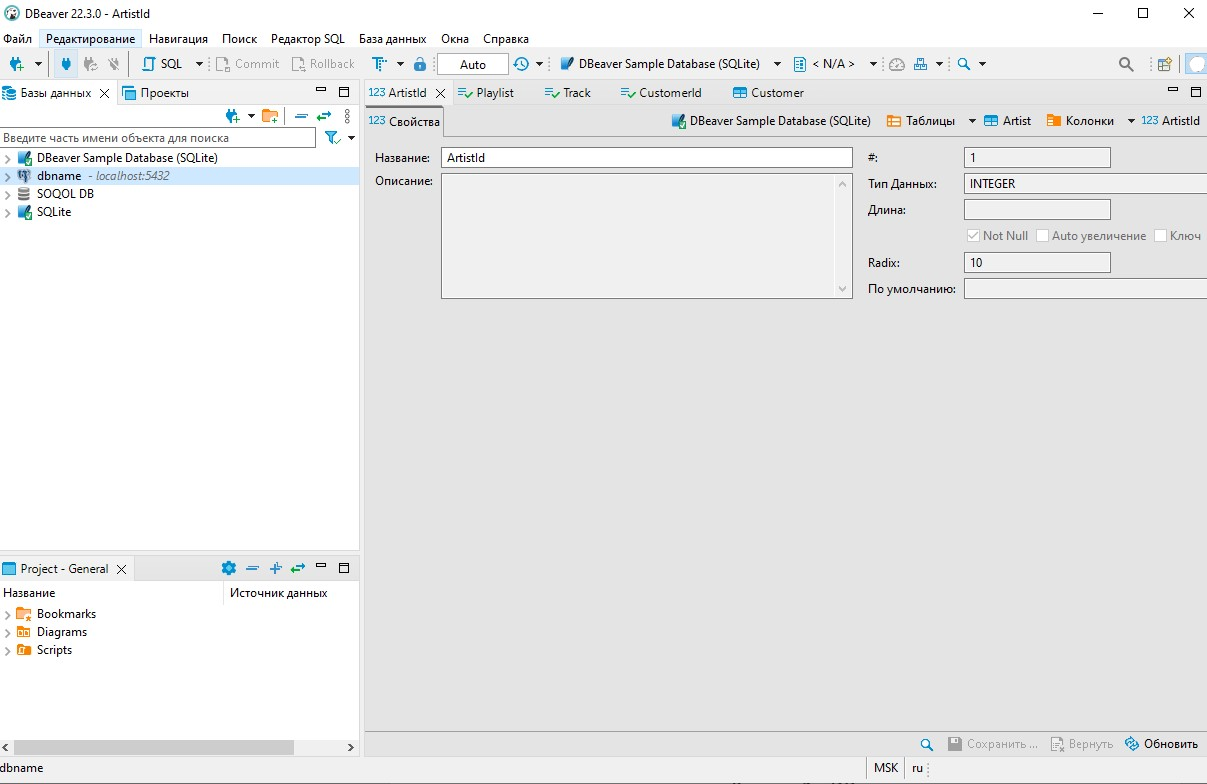


Рисунок 2 – DBeaver

Далее был скачен docker-образ с СУБД postgres с помощью команды для консоли (рис. 3).

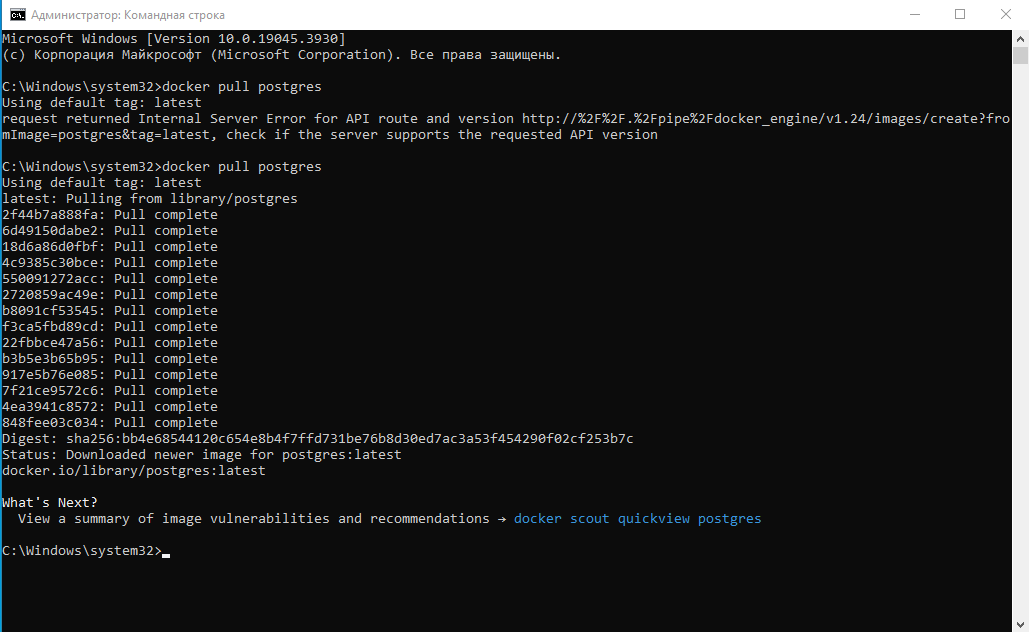


Рисунок 3 – Загрузка docker-образа

После загрузки был создан Dockerfile с необходимым содержанием (рис. 4). В данном файле определены следующие инструкции:

FROM postgres:latest:

Эта инструкция указывает базовый образ, который будет использован для создания нового образа. В данном случае, используется официальный образ PostgreSQL с тегом "latest". Это гарантирует использование последней версии образа PostgreSQL.

ENV POSTGRES\_PASSWORD=dbpass:

Данная инструкция устанавливает переменную окружения POSTGRES\_PASSWORD в значение "dbpass". Это значение будет использовано в качестве пароля для пользователя базы данных PostgreSQL.

ENV POSTGRES\_USER=dbuser:

Эта инструкция устанавливает переменную окружения POSTGRES\_USER в значение "dbuser". Это имя пользователя будет использовано для доступа к базе данных PostgreSQL.

ENV POSTGRES\_DB=dbname:

Здесь устанавливается переменная окружения POSTGRES\_DB в значение "dbname". Это имя базы данных, которая будет создана при инициализации контейнера.

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

Данная инструкция копирует SQL-скрипт init.sql из локальной директории init\_scripts/ внутрь контейнера по пути /docker-entrypoint-initdb.d/init.sql. Так как этот путь является стандартным для инициализации базы данных в контейнере PostgreSQL, скрипт будет выполнен при первом запуске контейнера, и его содержимое будет использовано для настройки базы данных.

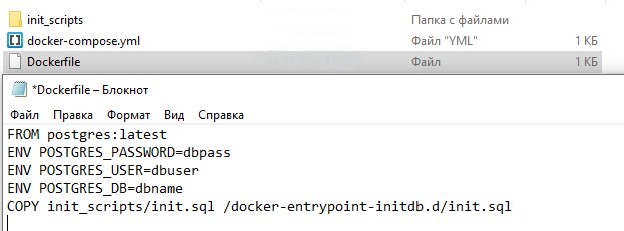


Рисунок 4 – Dockerfile

Содержимое файла init.sql изображено на рисунке 5 и содержит следующие SQL-запросы. CREATE TABLE IF NOT EXISTS public. index\_mass: Создает таблицу с именем index\_mass в схеме public. Оператор IF NOT EXISTS предотвращает ошибку, если таблица уже существует. 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.

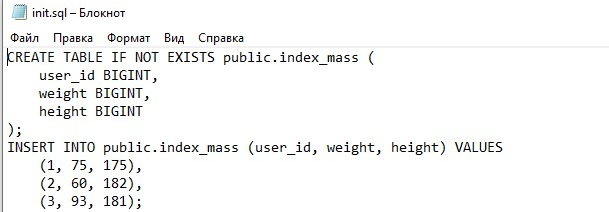


Рисунок 5 – init.sql

После этого был создан Docker-образ с именем "d\_img" с использованием команды "docker build -t d\_img:latest ." запущенной из каталога, содержащего Dockerfile (рис. 6).

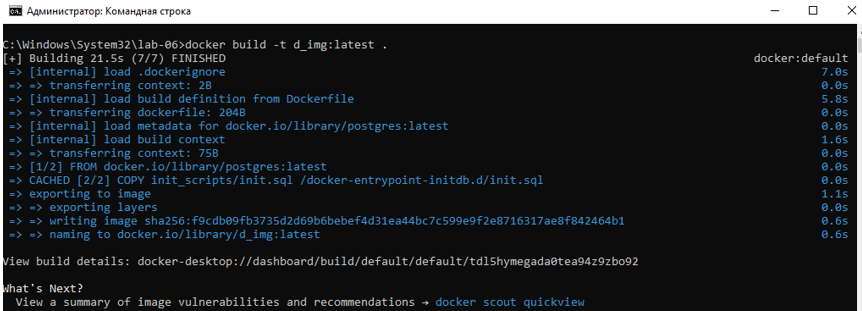


Рисунок 6 – Создание docker-образа

Для использования системы управления базами данных был запущен контейнер с именем «d\_cont» на основе образа «d\_img:latest» с использованием следующей команды: «docker run -d -p 5432:5432 --name d\_cont d\_img:latest» (рис. 7). При выполнении команды docker run, Docker осуществляет поиск образа d\_img:latest в локальном репозитории образов. Если образ не обнаруживается локально, он будет загружен из репозитория Docker Hub. После успешной загрузки образа Docker создает и запускает контейнер на основе этого образа. Контейнер, сформированный этой командой, работает в фоновом режиме, а порт 5432 контейнера привязан к порту 5432 хоста.

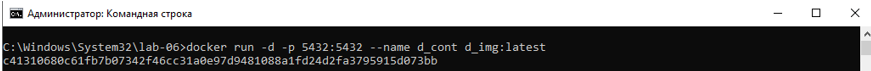


Рисунок 7 – Запуск контейнера

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

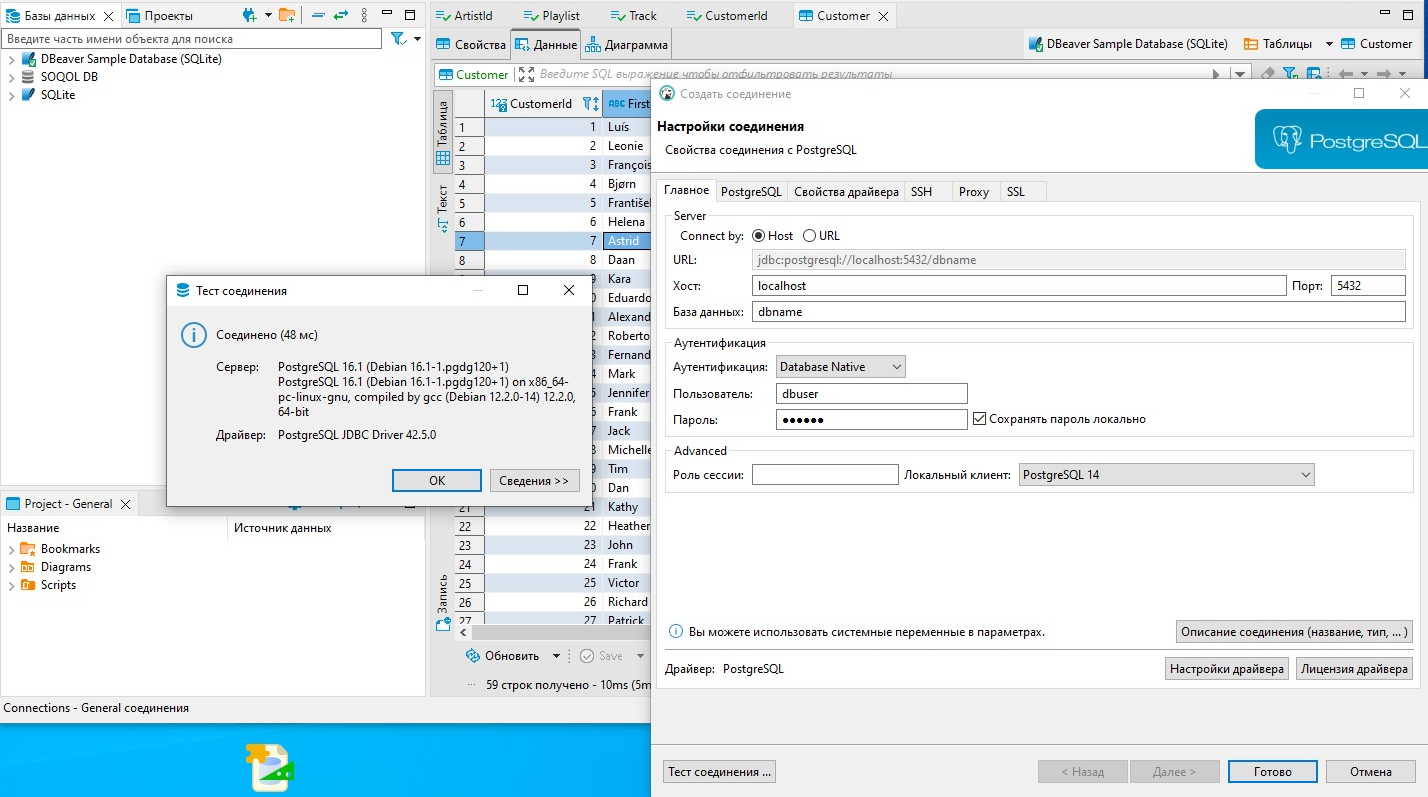


Рисунок 8 – Тестовое соединения

Для возможности взаимодействия с базой данных из консоли была разработана специальная команда: «docker exec -it d\_cont psql -U dbuser -d dbname». Эта команда обеспечивает подключение к активному контейнеру, запуск интерфейса psql и возможность внесения новых данных (рис. 9). При выполнении данной команды создается интерактивная сессия внутри контейнера с именем или идентификатором "d\_cont". Затем, внутри контейнера, запускается утилита командной строки psql для подключения к базе данных PostgreSQL с использованием указанного имени пользователя и наименования базы данных (-d dbname), что позволяет выполнять SQL-запросы и взаимодействовать с базой данных PostgreSQL изнутри Docker-контейнера.

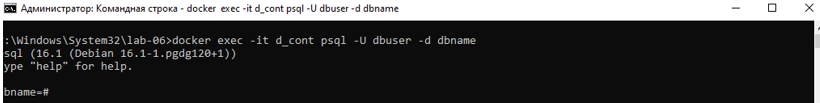


Рисунок 9 – Команда для подключения

На рисунке 10 продемонстрирован список имеющихся баз данных.

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рисунок 10 – Базы данных

Для того чтобы изменения данных, внесенные в процессе работы контейнера, оставались доступными после его удаления, требовалось создать том (volume). Для этого была внесена строка: «VOLUME /d\_data:/var/lib/postgresql/data». Таким образом, локально будет создан том (volume) в директории «/d\_data» (рис. 11).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 11 – Создание тома

Далее, после удаления созданного контейнера, был создан новый контейнер с использованием следующей команды: «docker run -d -p 5432:5432 --name d\_cont d\_img:latest -v /d\_data:/var/lib/postgresql/data» (рис. 12). Эта команда, аналогично предыдущей, 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. Флаг -v используется для привязки локальной директории /d\_data к директории контейнера /var/lib/postgresql/data. Таким образом, данные, сгенерированные контейнером PostgreSQL, сохраняются в локальной директории /d\_data.

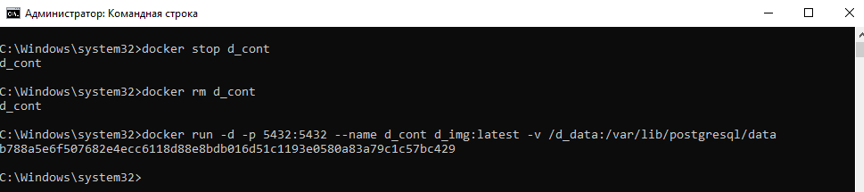


Рисунок 12 – Удаление и создание контейнера

В итоге был создан текстовый файл cmd.txt, в котором содержатся все команды, необходимые для развёртывания базы данных Postgres с использованием Dockerfile. Этот файл включает в себя шаги по созданию образа, запуску контейнера, созданию тома, команду для формирования Docker-контейнера с томом (volume) на основе Docker-образа, а также команду для запуска контейнера с интерфейсом psql для внесения новых данных в базу данных.

Затем был создан контейнер с применением файла docker-compose.yml, как показано на рисунке 13. В этом файле строка "version: "3.9"" указывает на используемую версию синтаксиса Docker Compose. Затем идет секция "services:", где определяются сервисы, создаваемые при запуске контейнеров. "dbpost:" представляет собой название выбранного нами сервиса базы данных PostgreSQL. "image: postgres:latest" указывает на образ контейнера для создания этого сервиса, и в данном случае используется официальный образ PostgreSQL с тегом "latest", обозначающим последнюю версию. "environment" определяет переменные окружения, передаваемые в контейнер, такие как 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, что позволяет инициализировать базу данных с использованием SQL-скрипта при запуске контейнера. "ports:" определяет маппинг портов между хост-системой и контейнером. Здесь порт 5432 на хосте маппится на порт 5432 внутри контейнера, обеспечивая внешний доступ к базе данных через этот порт.

Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 13 – 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, предназначенная для автоматизации процессов развёртывания, доставки и управления приложениями. Она обеспечивает возможность упаковки приложений и их зависимостей в контейнеры, которые могут быть запущены на различных операционных системах.

1. Зачем нужен Docker?

Docker является мощным инструментом, упрощающим процессы разработки, развёртывания и масштабирования приложений. Он также облегчает управление инфраструктурой и обеспечивает консистентность в различных средах.

1. Что такое docker-образ?

Docker-образ представляет собой исполняемую версию приложения, включающую все необходимые зависимости, файлы и конфигурации. Создается на основе Docker-файла, содержащего инструкции по установке и настройке приложения в контейнере.

1. Что такое docker-контейнер?

Docker-контейнер — это запускаемый экземпляр Docker-образа. Он создает изолированную среду, в которой работает приложение и его зависимости, используя ядро операционной системы хоста.

1. Что такое volume и зачем он нужен?

Volume в Docker - механизм управления данными, позволяющий обмениваться информацией между контейнерами и сохранять данные после удаления контейнера.

1. Что такое docker-compose?

Docker Compose - инструмент для определения и запуска нескольких контейнеров Docker как единого приложения, управления конфигурацией и зависимостями.

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

Dockerfile отвечает за создание Docker-образа, в то время как Docker Compose управляет несколькими контейнерами и их конфигурацией.

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

Команда docker exec используется для отправки различных задач в запущенный Docker-контейнер.

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

Для сохранения важных данных после аварийного отключения контейнера можно использовать Docker volumes или Docker data containers. Также можно применять инструменты, вроде Docker Compose, для создания многоконтейнерных приложений, где данные сохраняются в отдельных контейнерах или во внешних хранилищах, таких как базы данных или файловые системы.