|  |  |
| --- | --- |
|  | Министерство образования и науки Российской Федерации  Федеральное государственное автономное образовательное учреждение высшего образования  «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» |

Отчет

По лабораторной работе

# Основы работы с Docker и PostgreSQL

Выполнил Ермаков Д. А.

Группа  РИМ-150950

Преподаватель Кузьмин Д. И.

г. Екатеринбург

2025 г

**Цель работы:** освоить фундаментальные концепции и базовые операции Docker: создание образов, запуск контейнеров, управление ими, работа с сетями и томами. На практике закрепить навыки, запустив изолированную базу данных PostgreSQL и подключившись к ней извне.

**Задачи:**

1. Установить и проверить работу Docker.
2. Изучить базовые команды Docker.
3. Запустить контейнер с PostgreSQL в изолированном режиме.
4. Запустить контейнер с pgAdmin и подключить его к контейнеру с БД через сеть Docker.
5. Подключиться к БД из pgAdmin, создать схему и выполнить запросы.
6. Обеспечить сохранность данных БД с помощью томов Docker.

**Ход работы**

**Часть 0: Установка и проверка Docker**

Была проведена установка Docker Desktop для выбранной операционной системы по официальной инструкции. Для корректной работы docker compose на Linux выполнена отдельная установка согласно документации. После завершения установки в терминале были запущены команды для проверки: docker --version, docker compose --version, а также docker run hello-world. Команды корректно отработали: версии Docker и Docker Compose были успешно отображены, а команда запуска тестового контейнера вывела приветственное сообщение, подтверждающее работоспособность системы.

**Часть 1: Базовые команды Docker. Работа с образами и контейнерами**

В ходе работы были изучены основные команды Docker для управления образами и контейнерами. Сначала были просмотрены доступные образы и контейнеры, затем запущен простой контейнер на основе образа nginx:alpine с пробросом порта 8080 — результаты приведены на рисунке 1.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 - просмотр информации

Эти действия позволили открыть стартовую страницу по адресу [http://localhost:8080](http://localhost:8080/) и проверить работу сервера (рисунок 2).

Изображение выглядит как текст, Шрифт, снимок экрана, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 - проверка работы контейнера

После проверки контейнер был остановлен и удалён (рисунок 3).

Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 - остановка и удаление

**Часть 2: Запуск PostgreSQL в контейнере**

Для запуска PostgreSQL в контейнере был использован образ postgres:15. Сначала выполнена команда docker run -d --name my\_postgres\_db -e POSTGRES\_USER=postgres -e POSTGRES\_PASSWORD=postgres -e POSTGRES\_DB=test\_db -p 5432:5432 postgres:15. Эта команда создаёт и запускает контейнер с задаными параметрами пользователя, пароля и имени базы данных, а также пробрасывает порт 5432 для доступа к СУБД извне.

Чтобы убедиться, что контейнер успешно запущен и слушает порт, была выполнена команда docker ps (Рисунок 4).

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 – Запуск и проверка

Для подключения к базе данных напрямую из контейнера (через psql) введена команда:

docker exec -it my\_postgres\_db psql -U postgres -d test\_db

После этого в интерактивной консоли psql были выполнены тестовые запросы для создания таблицы, записи и чтения данных (рисунок 5).

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5 – Тестовые запросы

**Часть 3: Подключение к БД через pgAdmin из второго контейнера**

Для подключения к базе данных через pgAdmin сначала была создана общая сеть Docker командой docker network create my\_network, после чего к этой сети был подключен контейнер с базой PostgreSQL при помощи docker network connect my\_network my\_postgres\_db. Далее запущен контейнер pgAdmin, указав ту же сеть параметром --network my\_network и пробросив порт 8080 для доступа к веб-интерфейсу (рисунок 6).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 – Запуск pgAdmin

В веб-интерфейсе pgAdmin, доступном по адресу http://localhost:8080, выполнен вход под учётной записью admin@example.com. Создан новый сервер с именем по выбору, в качестве адреса указан my\_postgres\_db, имя пользователя и пароль — postgres. После успешного соединения через инструмент Query Tool выполнен SQL-запрос к таблице users. Отображается таблица, которая была создана ранее, и содержащиеся в ней данные (рисунок 7).

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 7 – Таблица users

**Часть 4: Сохранение данных с помощью Томов (Volumes)**

Сначала был остановлен и удалён текущий контейнер с базой данных, чтобы подготовить систему к использованию постоянного хранилища. После этого создан том для хранения данных PostgreSQL с помощью команды docker volume create postgres\_data и проведена проверка его наличия командой docker volume ls (рисунок 8).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8 – Остановка и удаление текущего контейнера и создание тома

Далее был запущен новый контейнер PostgreSQL, к которому подключён созданный том с помощью опции -v postgres\_data:/var/lib/postgresql/data, что обеспечивает сохранность всех данных при перезапуске контейнера. Контейнер также добавлен в ту же docker-сеть, что и pgAdmin, для дальнейшего взаимодействия.

Проверка сохранности данных выполнена через pgAdmin: после создания и заполнения таблицы users контейнеры были остановлены, затем запущены вновь, и после повторной проверки содержимого таблицы users данные остались на месте (рисунок 9).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9 – Таблица users

**Часть 4: Перенос конфигурации контейнеров в docker-compose.yaml**

Конфигурация для запуска контейнеров PostgreSQL и pgAdmin была перенесена в файл docker-compose.yaml, что позволило упростить процесс развертывания сервисов и их последующих запусков. В разделе services описаны два сервиса: db (используется официальный образ postgres:15, дополнительно подключается именованный том postgres\_data для постоянного хранения данных) и pgadmin (образ dpage/pgadmin4; заданы email и пароль для доступа, проброшен порт 5050).

Сервис pgAdmin зависит от контейнера с базой данных, а оба сервиса объединены общей пользовательской сетью test\_db, что позволяет им взаимодействовать друг с другом только внутри этой изолированной среды. Таким образом, весь процесс настройки необходимой среды для работы с PostgreSQL и визуального управления через pgAdmin сведен к запуску одной команды docker-compose up -d.

Такая организация позволяет запускать, останавливать и настраивать всю нужную инфраструктуру централизованно через docker-compose, обеспечивая удобство и воспроизводимость среды.Для переноса конфигурации контейнеров в единый файл была подготовлена рабочая версия docker-compose.yaml. В этом файле описаны два сервиса: база данных PostgreSQL, где в переменных окружения задаются пользователь, пароль и имя базы, а также указывается том для хранения данных, и pgAdmin с настройкой доступа и пробросом порта для удобного веб-интерфейса. Оба сервиса подключены к общей пользовательской сети, что гарантирует их доступность друг для друга по имени контейнера. Файл позволяет быстро и воспроизводимо поднимать всю систему одной командой, запуск сервиса pgAdmin автоматически ожидает готовности контейнера базы данных.

**Вопросы**

1. Docker — это платформа с открытым исходным кодом, предназначенная для автоматизации разработки, доставки и запуска приложений в изолированных контейнерах. Эти контейнеры содержат саму программу вместе со всеми необходимыми зависимостями и работают независимо от операционной системы хоста. Docker позволяет создавать стандартизированное, портативное и легко масштабируемое окружение для приложений, что упрощает их перенос, развертывание и управление.

2. Тома (Volumes) в Docker используются для постоянного хранения данных вне контейнера, чтобы данные сохранялись при удалении или перезапуске контейнера. Сети (Networks) позволяют изолировать и организовывать взаимодействие между контейнерами, обеспечивая их связь и обмен данными по именам контейнеров внутри общей виртуальной сети.

3. Чтобы подключиться к контейнеру и выполнять в нём команды, используется команда `docker exec`. Например, команда `docker exec -it <container\_name> bash` или `docker exec -it <container\_name> psql` запускает интерактивную сессию в контейнере, в которой можно вводить необходимые команды в терминале или консоли приложения внутри контейнера.

4. pgAdmin — это веб-интерфейс для удобного управления базами данных PostgreSQL, работающий через браузер. Он облегчает выполнение различных операций с базами данных, включая просмотр и редактирование данных, выполнение SQL-запросов и администрирование серверов PostgreSQL без необходимости напрямую использовать командную строку.

**Вывод**

В результате выполненных этапов работы были успешно установлены и проверены компоненты Docker, PostgreSQL и pgAdmin. Было продемонстрировано, как с помощью базовых команд Docker осуществлять запуск, остановку и удаление контейнеров, а также управлять образами и подключаться к работающим контейнерам для выполнения команд. Особое внимание уделено организации сохранения данных через тома (Volumes), что обеспечивает сохранность информации при перезапуске и обновлении контейнеров.

Также была налажена связка между контейнерами PostgreSQL и pgAdmin через созданную Docker-сеть, что позволило осуществлять визуальное и упрощённое управление базами данных через веб-интерфейс pgAdmin. Конфигурация всей системы была централизованно перенесена в docker-compose.yaml, что значительно упростило запуск и поддержку всех сервисов. Итоговая работа подтверждает возможность эффективного и удобного развертывания и эксплуатации базы данных и вспомогательных инструментов с использованием контейнеризации Docker.

Ссылка на github: <https://github.com/ErmakovDenis/lab2>