Министерство образования и науки Российской Федерации

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

**Лабораторная работа №1: Основы работы с Docker и PostgreSQL**

Студент А. Н. Шубина

группы РИМ-150950

Екатеринбург 2025

**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.

## Часть 0: Установка и проверка Docker

1. **Установила Docker Desktop** (<https://www.docker.com/products/docker-desktop/>).
2. Обновила WSL
3. Открыла терминал и проверила установку:

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

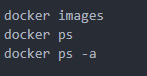
## Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение Содержимое, созданное искусственным интеллектом, может быть неверным.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

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

1. **Просмотр информации:**

****

1. **Запуск простого контейнера (на примере Nginx):**

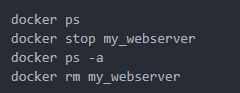


**-d** (detach) — запустить контейнер в фоне.

**-p 8080:80** — проброс порта (хост:контейнер).

**--name** — задать понятное имя контейнеру.

**nginx:alpine** — имя образа и его тег (версия).



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

Содержимое, созданное искусственным интеллектом, может быть неверным.

1. **Проверила, что контейнер работает:** Открыла в браузере **http://localhost:8080**. Увидела стартовую страницу Nginx.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

1. **Остановила и удалила контейнер:**

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Часть 2: Запуск PostgreSQL в контейнере

1. **Запустила контейнер с PostgreSQL:**

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

**-e** — установка переменных окружения внутри контейнера.

**-p** – проброс портов для работы с БД внешних систем

Переменные **POSTGRES\_\*** необходимы для первоначальной настройки СУБД.

1. **Проверила, что контейнер запущен и слушает порт:**



1. **Подключилась к БД прямо из контейнера (через psql):**



Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Оказалась в консоли PostgreSQL. Выполнила тестовые запросы:

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Часть 3: Подключение к БД через pgAdmin из второго контейнера

1. **Создала сеть Docker:** Контейнеры по умолчанию изолированы. Чтобы они "увидели" друг друга по именам, нужна общая сеть.



1. **Подключила контейнер с PostgreSQL к сети:**



1. **Запустила pgAdmin в той же сети:**

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Ключевой параметр --network my\_network помещает этот контейнер в ту же сеть, что и БД.

4. **Настроила подключение в pgAdmin:**

Открыла **http://localhost:8080** в браузере.

Вошла под **admin@example.com** / **admin**.

**Добавила новый сервер:**

**General -> Name:** **Docker PostgreSQL**

**Connection -> Host name/address:** **my\_postgres\_db**

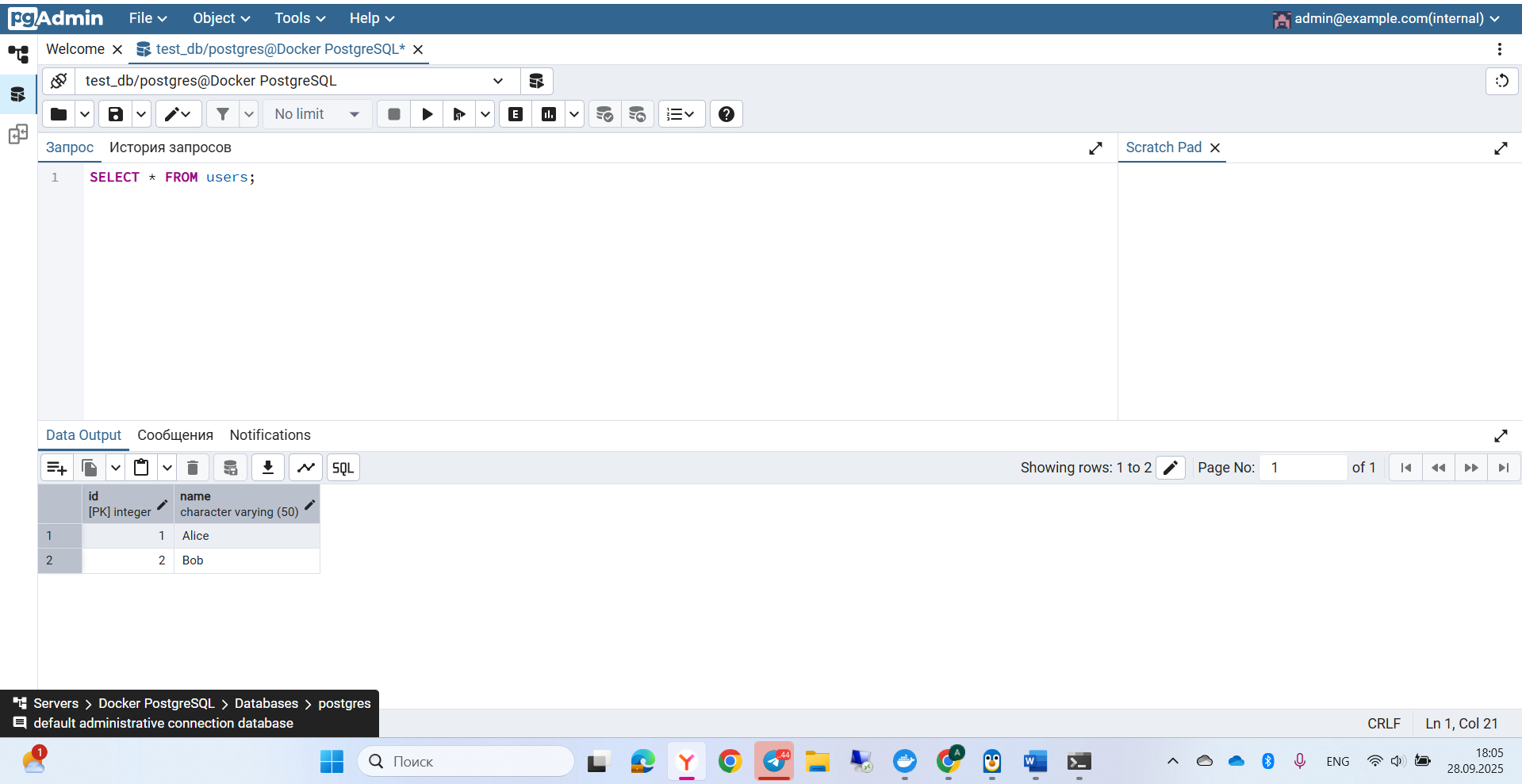
**Connection -> Username:** **postgres**

**Connection -> Password:** **postgres**

Сохранила. Подключение прошло успешно.

**Через Query Tool в pgAdmin выполнила запрос:**





Часть 4: Сохранение данных с помощью Томов (Volumes)

1. **Остановила и удалила текущий контейнер с БД:**

*Все данные, созданные внутри контейнера, будут безвозвратно утеряны.*

1. **Создала том для хранения данных БД:**



1. **Запустила новый контейнер с PostgreSQL, подключив том:**

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Проверила сохранность данных:**

1. Через pgAdmin (**http://localhost:8080**) снова выполнила запрос **SELECT \* FROM users;** добавила таблицу и данные
2. Остановила контейнеры с БД и pgAdmin
3. Запустила контейнеры 
4. Проверила таблицу users

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Вопросы

1. Что такое docker?  
Docker — это платформа контейнеризации, которая реализует операционную систем-уровневую виртуализацию. В отличие от традиционных виртуальных машин, контейнеры используют общее ядро хостовой ОС, что обеспечивает значительно лучшую производительность и плотность размещения.

С архитектурной точки зрения, Docker состоит из:

* Docker Daemon — управляет контейнерами, образами, томами
* Docker Client — CLI для взаимодействия с демоном
* Docker Registry — репозиторий образов (Docker Hub)
* Docker Images — неизменяемые шаблоны для создания контейнеров

В контексте разработки ПО, Docker решает проблему "works on my machine" через единообразие сред выполнения, что критически важно для CI/CD пайплайнов.

2. Для чего нужны тома и сети docker?

**Тома** — механизм persistence storage в Docker. Реализуются через managed volumes (управляемые Docker) или bind mounts (привязка к хостовой ФС). В нашем проекте том postgres\_data обеспечивает сохранность данных БД при lifecycle контейнеров, что соответствует принципу stateless-контейнеров.

**Сети** — обеспечиют сетевую изоляцию и service discovery. Docker предоставляет несколько драйверов:

* bridge — дефолтная сеть для single host
* overlay — для multi-host коммуникации (Docker Swarm)
* host — прямой доступ к сетевому стеку хоста

В нашей конфигурации сеть test\_network создает изолированный segment, где сервисы резолвятся по DNS именам (db, pgadmin).

3. Как подключится к контейнеру и выполнить в нём команды?

Для подключения к контейнеру используется команда docker exec. Например:

# Подключиться к контейнеру с PostgreSQL

docker exec -it my\_postgres\_db bash

# Или сразу запустить psql внутри контейнера

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

Флаги -it означают:

* -i — интерактивный режим (можно вводить команды)
* -t — эмуляция терминала

Внутри контейнера можно выполнять любые команды, как в обычной Linux-системе.

4. Для чего нужен pgAdmin?

— это веб-интерфейс для управления PostgreSQL. Вместо того чтобы работать с БД только через командную строку (psql), pgAdmin предоставляет удобный графический интерфейс где можно:

В нашей работе pgAdmin помог:

* Визуально увидеть структуру базы данных
* Выполнять SQL-запросы в удобном редакторе
* Убедиться, что наша таблица users создалась правильно
* Проверить, что данные сохраняются после перезапуска контейнеров

Это особенно полезно когда нужно быстро посмотреть, что находится в базе, или выполнить сложные запросы с подсказками синтаксиса.

Вывод  
В ходе выполнения работы был успешно освоен Docker и сопутствующие технологии для контейнеризации приложений. На практике реализована многоконтейнерная архитектура, включающая:

* Развертывание PostgreSQL в изолированном контейнере
* Настройка pgAdmin для визуального управления БД
* Организация сетевого взаимодействия между контейнерами
* Обеспечение сохранность данных через Docker volumes
* Автоматизация развертывания с помощью Docker Compose

Работа продемонстрировала ключевые преимущества контейнеризации: воспроизводимость сред, изоляцию сервисов, эффективное управление зависимостями и упрощение деплоя сложных приложений. Полученные навыки позволяют эффективно применять Docker в реальных проектах разработки ПО.