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

Институт радиоэлектроники и информационных технологий – РТФ

Лабораторная работа №1

Основы работы с Docker и PostgreSQL

Студент группы РИМ – 150950: _____ Вальнева А.Д.

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

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Anastasia> docker --version
Docker version 28.4.0, build d8eb465
PS C:\Users\Anastasia> docker-compose --version
Docker Compose version v2.39.4-desktop.1
PS C:\Users\Anastasia> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

PS C:\Users\Anastasia>
```

Ход выполнения:

1. **Установка Docker Desktop с официального сайта для операционной системы Windows.**

2. **Проверка установки Docker и Docker Compose:**

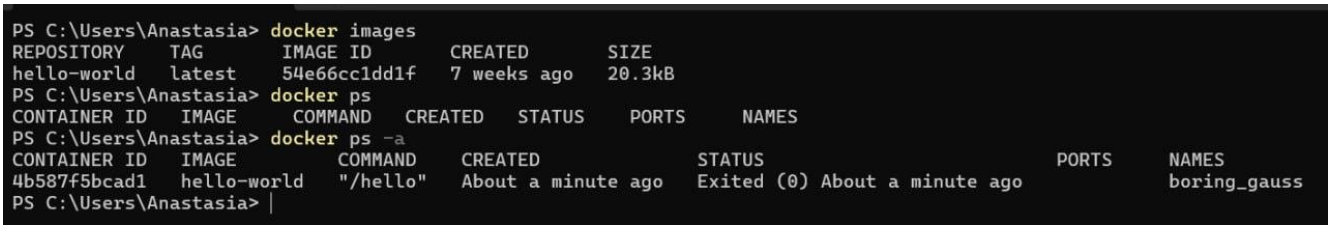
   - docker –version (Показывает корректную установку Docker в системе, версия - 28.4.0)
   - docker-compose –version (Docker Compose установлен и интегрирован в Docker Desktop. Установлена версия v2.39.4.)

3. Для окончательной проверки работоспособности всей системы Docker была выполнена команда запуска тестового контейнера docker run hello-world, который продемонстрировал весь цикл работы Docker: поиск образа, его загрузку из реестра, создание и запуск контейнера.

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

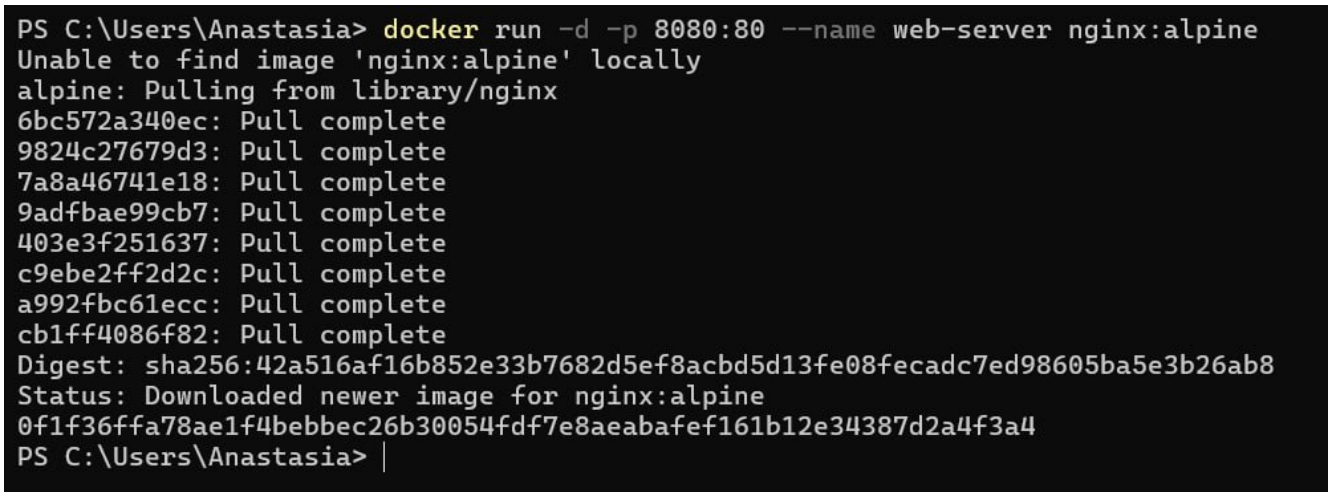
Ход выполнения:

1. **Просмотр информации о Docker**: перед запуском нового контейнера были выполнены команды для просмотра текущего состояния:



- docker images (Отображает все Docker-образы, сохраненные в локальной системе. Показал один образ - hello-world:latest с идентификатором 54e66cclddlf)
- docker ps (Показывает список активных контейнеров: пустой)
- docker ps -a (Показывает все контейнеры, включая остановленные)

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



- **docker run -d -p 8080:80 --name web-server nginx:alpine**

    - **-d (detach)** (контейнер был запущен в фоновом режиме)
    - **-p 8080:80** (был настроен проброс портов — порт 8080 на хосте связан с портом 80 внутри контейнера (порт, который прослушивает Nginx))
    - **--name web-server** (контейнеру было задано понятное имя web-server для удобства управления)
    - **nginx:alpine** (был указан образ для запуска alpine)

3. **Проверка, что контейнер работает (**Для проверки работы контейнера был открыт веб-браузер и перейдя по адресу **http://localhost:8080** в браузере отобразилась стандартная приветственная страница Nginx.)

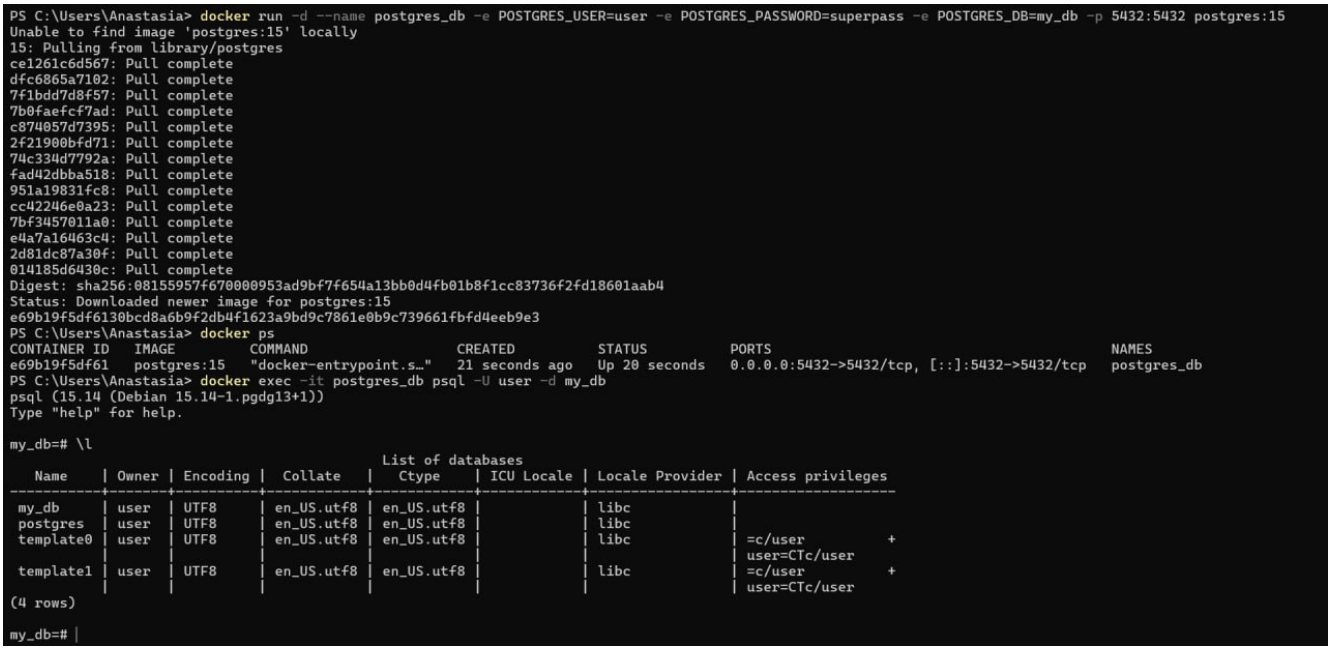
4. **Остановка и удаление контейнера**

```
PS C:\Users\Anastasia> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                       NAMES
0f1f36ffa78a   nginx:alpine   "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   web-server
PS C:\Users\Anastasia> docker stop web-server
web-server
PS C:\Users\Anastasia> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS                     PORTS     NAMES
0f1f36ffa78a   nginx:alpine   "/docker-entrypoint.…"   2 minutes ago   Exited (0) 5 seconds ago             web-server
4b587f5bcad1   hello-world    "/hello"                 5 minutes ago   Exited (0) 5 minutes ago             boring_gauss
PS C:\Users\Anastasia> docker rm web-server
web-server
PS C:\Users\Anastasia> docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS                     PORTS     NAMES
4b587f5bcad1   hello-world   "/hello"   6 minutes ago   Exited (0) 6 minutes ago             boring_gauss
PS C:\Users\Anastasia>
```

- docker ps (Проверяем, что контейнер находится в статусе "Up" (работает) в течение 2 минут)
- docker stop web-server (Команда остановки контейнера - успешно выполнилась)
- docker ps -a (Остановленный контейнер имеет статус Exited (0) 5 seconds ago, что подтверждает его остановку и продолжает отображаться в списке всех контейнеров до момента его удаления)
- docker rm web-server (Контейнер был полностью удален из системы)
- **Финальная проверка:** docker ps -a (Контейнер web-server был успешно удален из списка)

Часть 2: Запуск PostgreSQL в контейнере

Ход выполнения:



```
my_db=# \dt
Did not find any relations.
my_db=# CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(50));
CREATE TABLE
my_db=# INSERT INTO users (name) VALUES ('Nastya'), ('Alice'), ('Ivan');
INSERT 0 3
my_db=# SELECT * FROM users;
 id |  name
----+--------
  1 | Nastya
  2 | Alice
  3 | Ivan
(3 rows)

my_db=# \q
PS C:\Users\Anastasia>
```

1. Запуск контейнера с PostgreSQL

- docker run -d --name postgres_db -e POSTGRES_USER=user -e POSTGRES_PASSWORD=superpass -e POSTGRES_DB=my_db -p 5432:5432 postgres:15
    - -d — запуск в фоновом режиме
    - --name postgres_db — имя контейнера для удобства управления
    - -e POSTGRES_USER=user — установка переменной окружения с именем пользователя
    - -e POSTGRES_PASSWORD=superpass — установка пароля пользователя
    - -e POSTGRES_DB=my_db — создание базы данных при инициализации
    - -p 5432:5432 — проброс порта (порт хоста:порт контейнера)
    - postgres:15 — образ PostgreSQL версии 15

2. Проверка, что контейнер запущен и слушает порт:

- docker ps

3. Подключение к базе данных внутри контейнера:

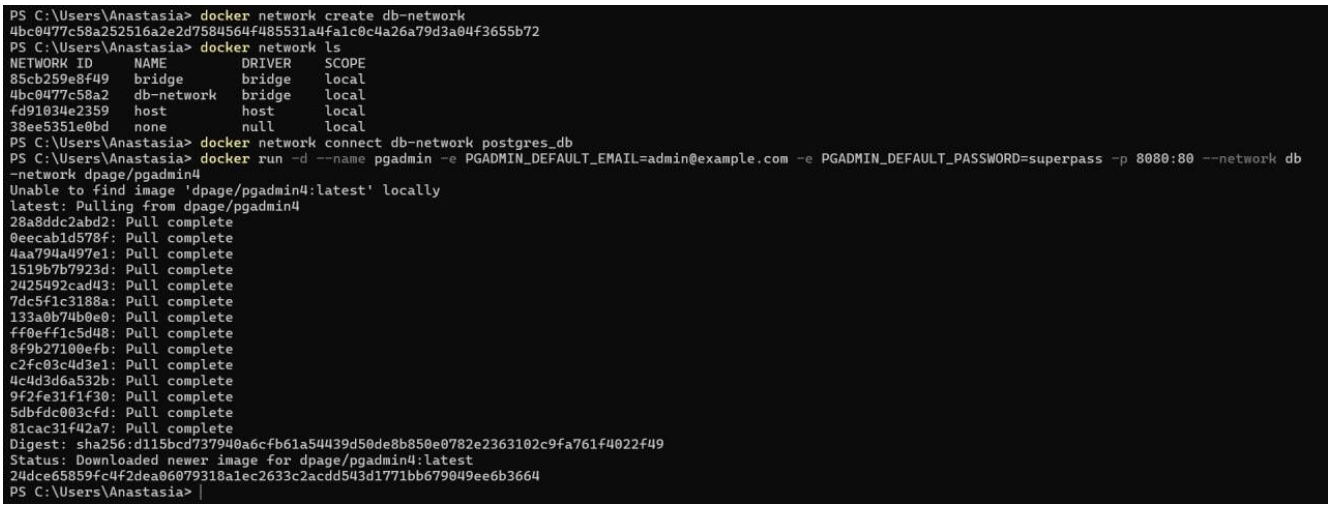
- docker exec -it postgres_db psql -U user -d my_db
    - exec — выполнение команды внутри running-контейнера
    - -it — интерактивный режим с терминалом
    - psql -U user -d my_db — запуск клиента PostgreSQL с указанием пользователя и базы данных

Выполненные операции в psql:

- Просмотр списка баз данных: \l (Система баз данных инициализирована корректно)
- Просмотр таблиц: \dt (В базе данных отсутствуют пользовательские таблицы)
- Создание таблицы: CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(50))
- Добавление данных: INSERT INTO users (name) VALUES ('Nastya'), ('Alice'), ('Ivan');
- Выборка данных: SELECT * FROM users;

Часть 3: Подключение к БД через pgAdmin из второго контейнера

Ход выполнения:



1. **Создание сети Docker: для обеспечения связи между контейнерами**
    - docker network create db-network (Создана пользовательская сеть db-network с драйвером bridge)
    - docker network ls (Команда отображает все доступные сети, включая только что созданную db-network)

2. **Подключение контейнера PostgreSQL к созданной сети**

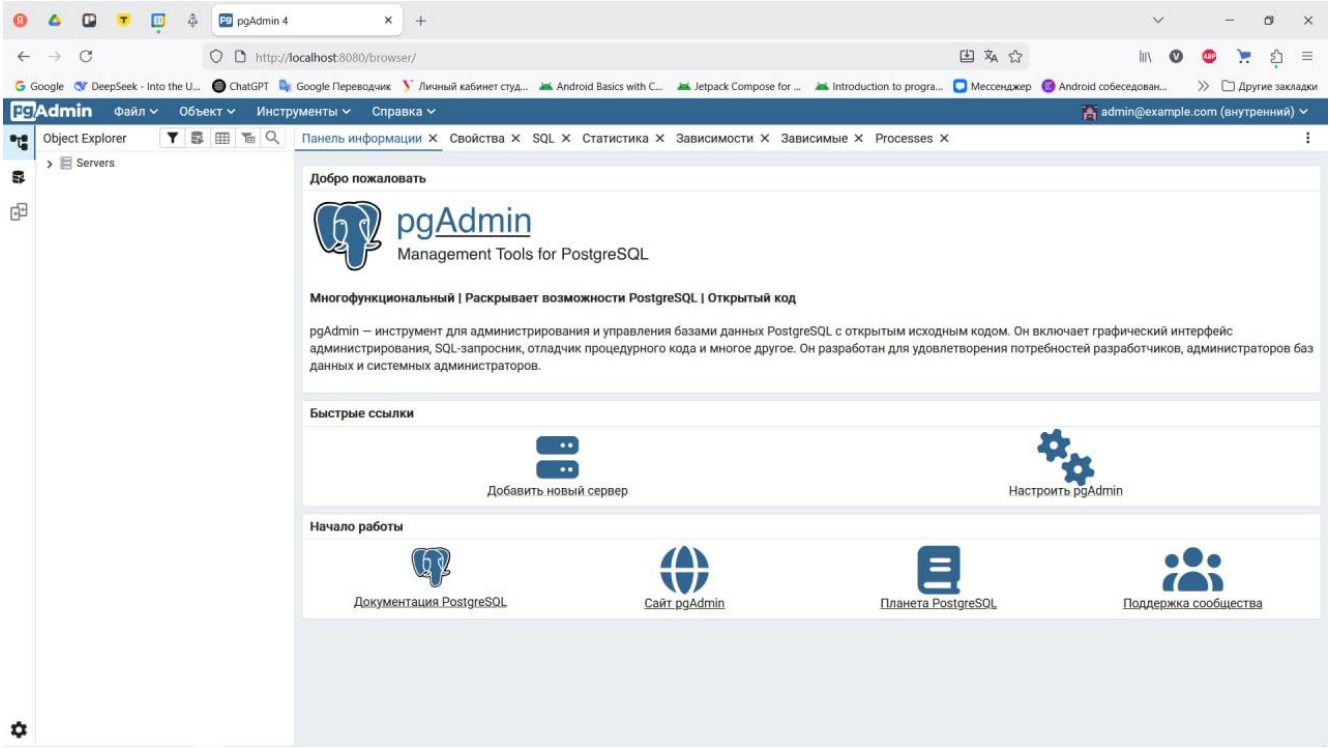
    - docker network connect db-network postgres_db (Контейнер postgres_db теперь находится в сети db-network и может быть доступен по имени из других контейнеров в этой сети)

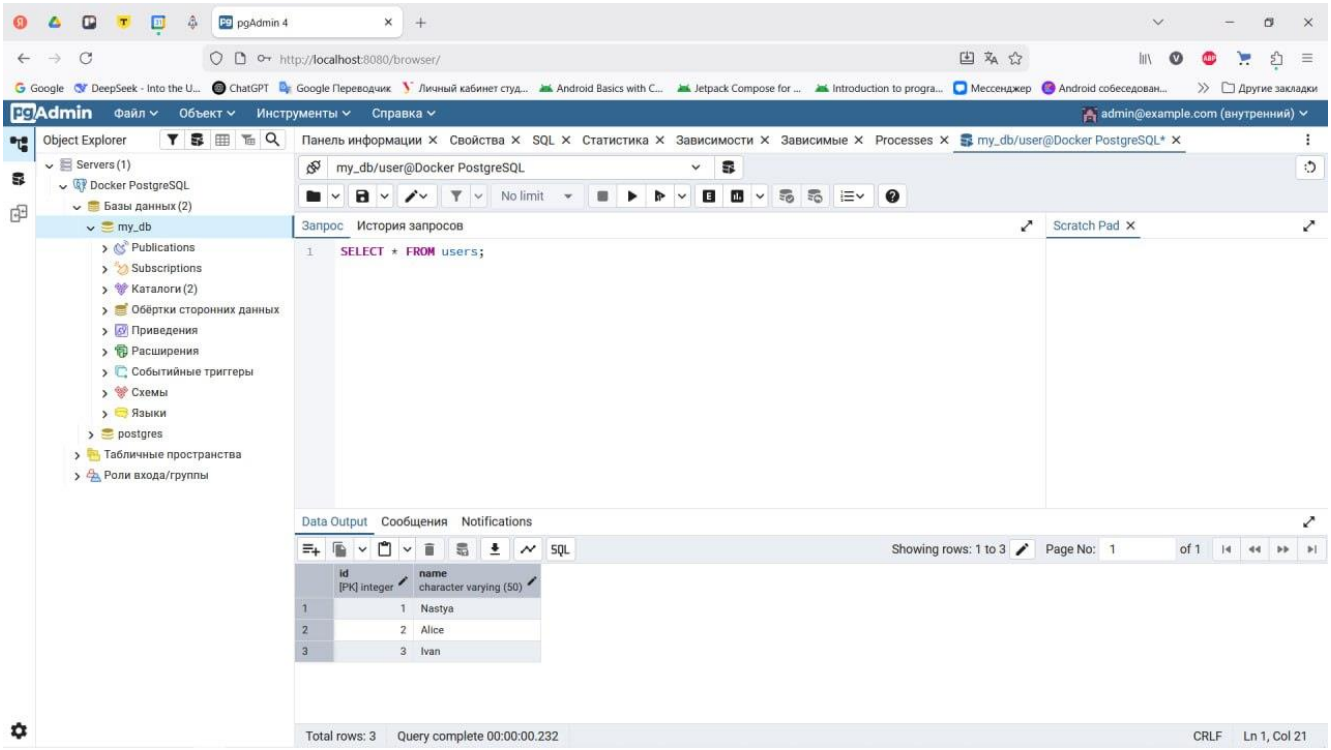
3. Запуск контейнера с pgAdmin, настроенный на использование созданной сети

- docker run -d --name pgadmin -e PGADMIN_DEFAULT_EMAIL=admin@example.com -e PGADMIN_DEFAULT_PASSWORD=superpass -p 8080:80 --network db-network dpage/pgadmin4

  - -d — запуск в фоновом режиме.
  - --name pgadmin — имя контейнера.
  - -e PGADMIN_DEFAULT_EMAIL=admin@example.com — email для входа в pgAdmin.
  - -e PGADMIN_DEFAULT_PASSWORD=superpass — пароль для входа.
  - -p 8080:80 — проброс порта (веб-интерфейс pgAdmin доступен на localhost:8080).
  - --network db-network — подключение контейнера к сети db-network.
  - dpage/pgadmin4 — образ pgAdmin.

4. Настройка подключения в pgAdmin и соединения с базой данных.





- Открытие в браузере http://localhost:8080. Отобразилась стартовая страница pgAdmin с заголовком "Добро пожаловать".
- Вход в систему с учетными данными:
    - Email: admin@example.com.
    - Пароль: superpass
- Добавление нового сервера в pgAdmin:
    - **General -> Name:** Docker PostgreSQL (произвольное имя для подключения)
    - **Connection -> Host name/address:** postgres.db
    - **Connection -> Username:** user (пользователь, указанный при запуске контейнера с PostgreSQL).
    - **Connection -> Password** (пароль, указанный при запуске).
- Выполнение SQL-запроса в Query Tool: SELECT * FROM users (в окне вывода данных отобразилась таблица с тремя записями)

Часть 4: Сохранение данных с помощью Томов (Volumes)

Ход выполнения:

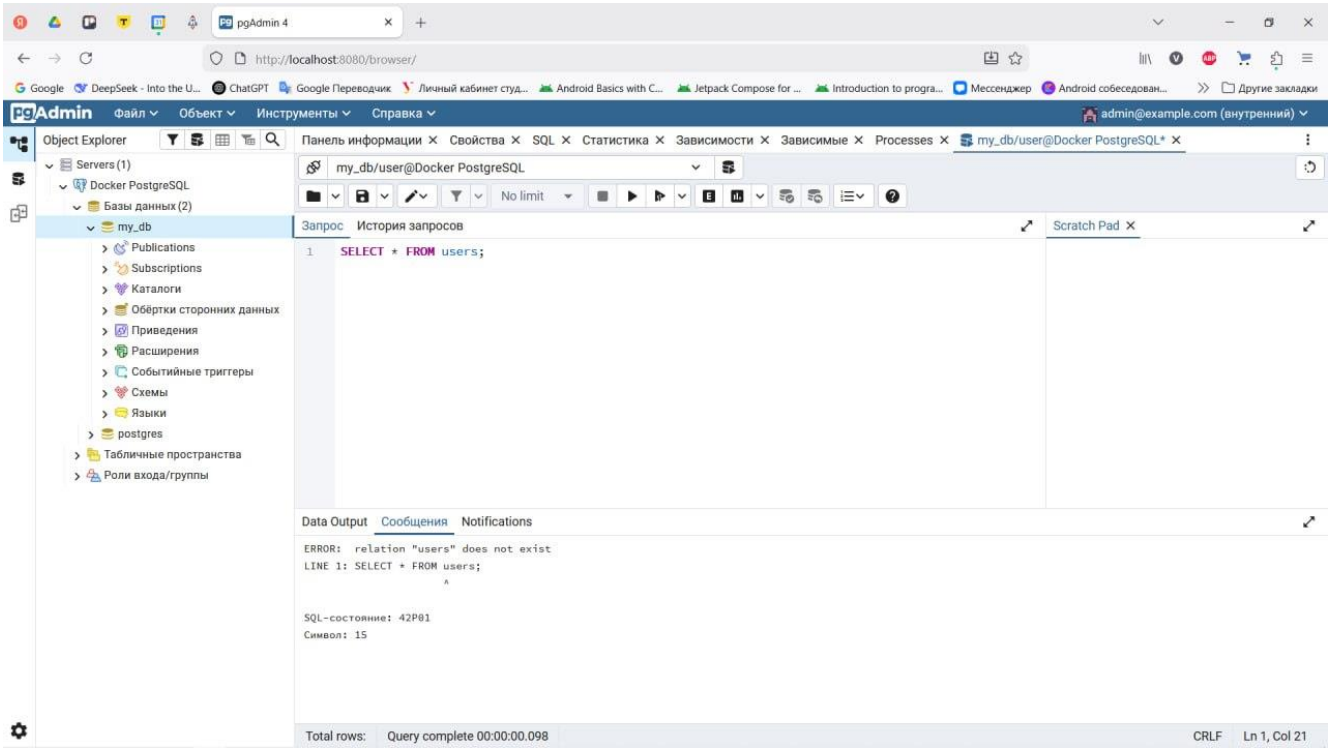
```
PS C:\Users\Anastasia> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
24dce65859fc   dpage/pgadmin4  "/entrypoint.sh"         7 minutes ago    Up 7 minutes    0.0.0.0:8080->80/tcp, [::]:8080->80/tcp                pgadmin
e69b19f5df61   postgres:15     "docker-entrypoint.s…"   14 minutes ago   Up 14 minutes   0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp            postgres_db
PS C:\Users\Anastasia> docker stop postgres_db
postgres_db
PS C:\Users\Anastasia> docker ps
CONTAINER ID   IMAGE           COMMAND            CREATED         STATUS         PORTS                                                  NAMES
24dce65859fc   dpage/pgadmin4  "/entrypoint.sh"   8 minutes ago   Up 8 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp                pgadmin
PS C:\Users\Anastasia> docker volume create postgres_data
postgres_data
PS C:\Users\Anastasia> docker volume ls
DRIVER    VOLUME NAME
local     606eb63debdedfaafa7e3f1344d7e8733472bcf978799f4fbc5d94b98b172d3c
local     a2c66363dfb5f21ac11716496f1b266fbcac417963e829c1b66617df6b8def59
local     postgres_data
```

1. **Остановка контейнера** postgres_db (docker stop) **и затем удаление** (docker rm), что имитирует потерю данных (так как данные по умолчанию хранятся внутри контейнера)

2. **Работа с томами (Volumes):**
   - docker volume create (Создан том с именем postgres_data с помощью команды)
   - docker volume ls (Выводит список всех томов (volumes), существующих в Docker. После создания тома postgres_data теперь существует три тома)

```
PS C:\Users\Anastasia> docker ps -a
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS                      PORTS                                                  NAMES
24dce65859fc   dpage/pgadmin4  "/entrypoint.sh"         10 minutes ago   Up 10 minutes               0.0.0.0:8080->80/tcp, [::]:8080->80/tcp                pgadmin
e69b19f5df61   postgres:15     "docker-entrypoint.s…"   17 minutes ago   Exited (0) 2 minutes ago                                                           postgres_db
4b587f5bcad1   hello-world     "/hello"                 33 minutes ago   Exited (0) 33 minutes ago                                                          boring_gauss
PS C:\Users\Anastasia> docker rm postgres_db
postgres_db
PS C:\Users\Anastasia> docker run -d --name postgres_db -e POSTGRES_USER=user -e POSTGRES_PASSWORD=superpass -e POSTGRES_DB=my_db -p 5432:5432 -v postgres_d
ata:/var/lib/postgresql/data --network db-network postgres:15
74c581c0aed14402b7f7742203b6f30ce8ffa38ed6c683244e0841d7ec6f1ea3
PS C:\Users\Anastasia>
```

3. Был запущен новый контейнер postgres_db с подключением тома postgres_data (команда docker run ... -v postgres_data:/var/lib/postgresql/data .. - монтирует том **postgres_data** в директорию внутри контейнера, где PostgreSQL хранит все свои данные)



- В pgAdmin был выполнен запрос SELECT * FROM users, но таблицы users не существует (ошибка "relation 'users' does not exist").

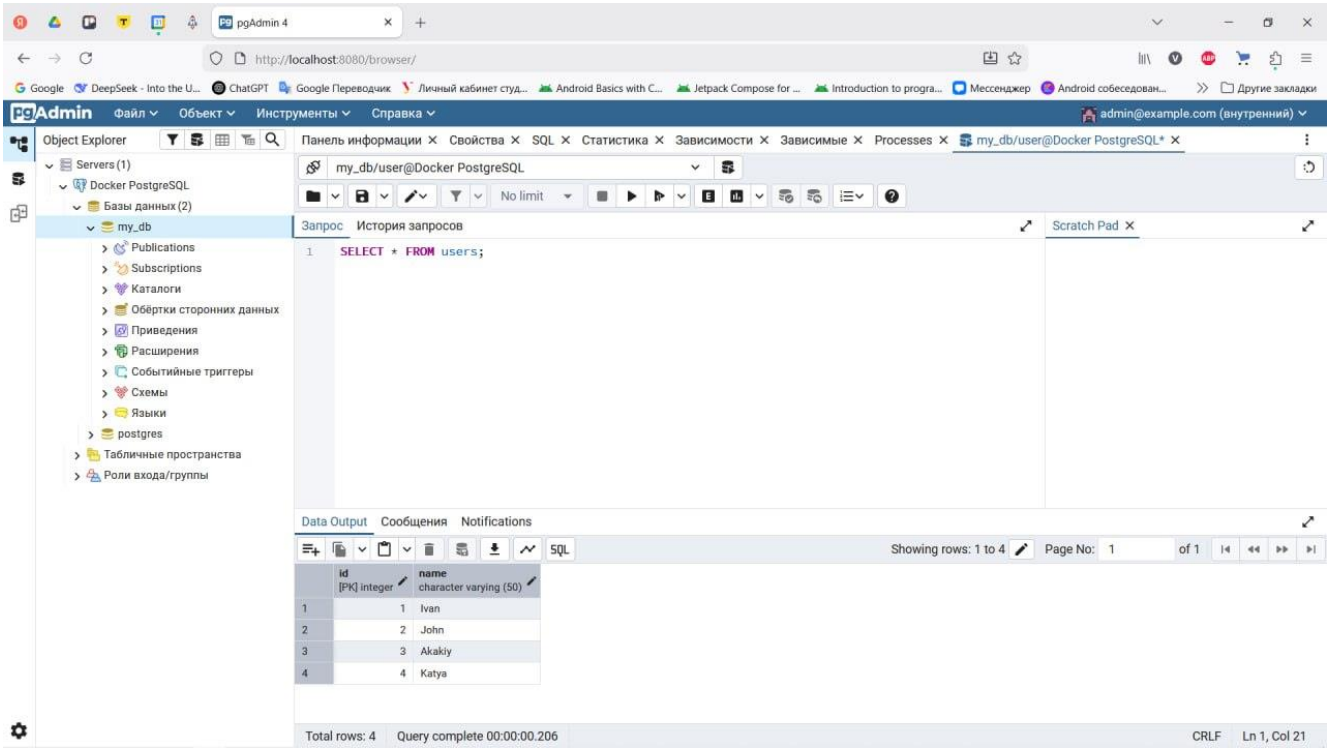
Почему данные не сохранились и как исправить?

- Ранее (часть 3) мы создавали контейнер postgres_db без тома
- Данный контейнер мы удалили, все данные внутри него были утеряны, т.к. по умолчанию контейнеры не сохраняют данные после удаления (без тома данные хранятся внутри контейнера).
- Поэтому, когда запустили новый контейнер без тома, база данных была чистой, и таблицы users в ней не существовало. При выполнении запроса SELECT * FROM users; получаем ошибку "relation 'users' does not exist".

```
PS C:\Users\Anastasia> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS                                                NAMES
74c581c0aed1   postgres:15     "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes    0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp          postgres_db
24dce65859fc   dpage/pgadmin4  "/entrypoint.sh"         13 minutes ago   Up 13 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp              pgadmin
PS C:\Users\Anastasia> docker exec -it postgres_db /bin/bash
root@74c581c0aed1:/# psql -U user -d my_db
psql (15.14 (Debian 15.14-1.pgdg13+1))
Type "help" for help.

my_db=# \l
                                              List of databases
   Name    | Owner | Encoding |  Collate   |   Ctype    | ICU Locale | Locale Provider | Access privileges
-----------+-------+----------+------------+------------+------------+-----------------+-------------------
 my_db     | user  | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            |
 postgres  | user  | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            |
 template0 | user  | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            | =c/user          +
           |       |          |            |            |            |                 | user=CTc/user
 template1 | user  | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            | =c/user          +
           |       |          |            |            |            |                 | user=CTc/user
(4 rows)

my_db=# \dt
Did not find any relations.
my_db=# CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(50));
CREATE TABLE
my_db=# INSERT INTO users (name) VALUES ('Ivan'), ('John'), ('Akakiy'), ('Katya');
INSERT 0 4
my_db=# SELECT * FROM users;
 id |  name
----+--------
  1 | Ivan
  2 | John
  3 | Akakiy
  4 | Katya
(4 rows)

my_db=# \q
root@74c581c0aed1:/# exit
exit
PS C:\Users\Anastasia>
```

```
PS C:\Users\Anastasia> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS                                                NAMES
74c581c0aed1   postgres:15     "docker-entrypoint.s…"   4 minutes ago    Up 4 minutes    0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp          postgres_db
24dce65859fc   dpage/pgadmin4  "/entrypoint.sh"         15 minutes ago   Up 15 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp              pgadmin
PS C:\Users\Anastasia> docker restart postgres_db pgadmin
postgres_db
pgadmin
PS C:\Users\Anastasia>
```

- docker ps (просмотр запущенных контейнеров - postgres_db и pgadmin)
- docker exec -it postgres_db /bin/bash (вход в контейнер postgres_db)
- psql -U user -d my_db (подключение)
- \dt (проверка существующих таблиц (их не было))
- INSERT INTO users (name) VALUES ('Ivan'), ('John'), ('Akakiy'), ('Katya') (добавление данных)
- SELECT * FROM users (проверка данных)
- docker restart postgres_db pgadmin (перезапустили оба контейнера)

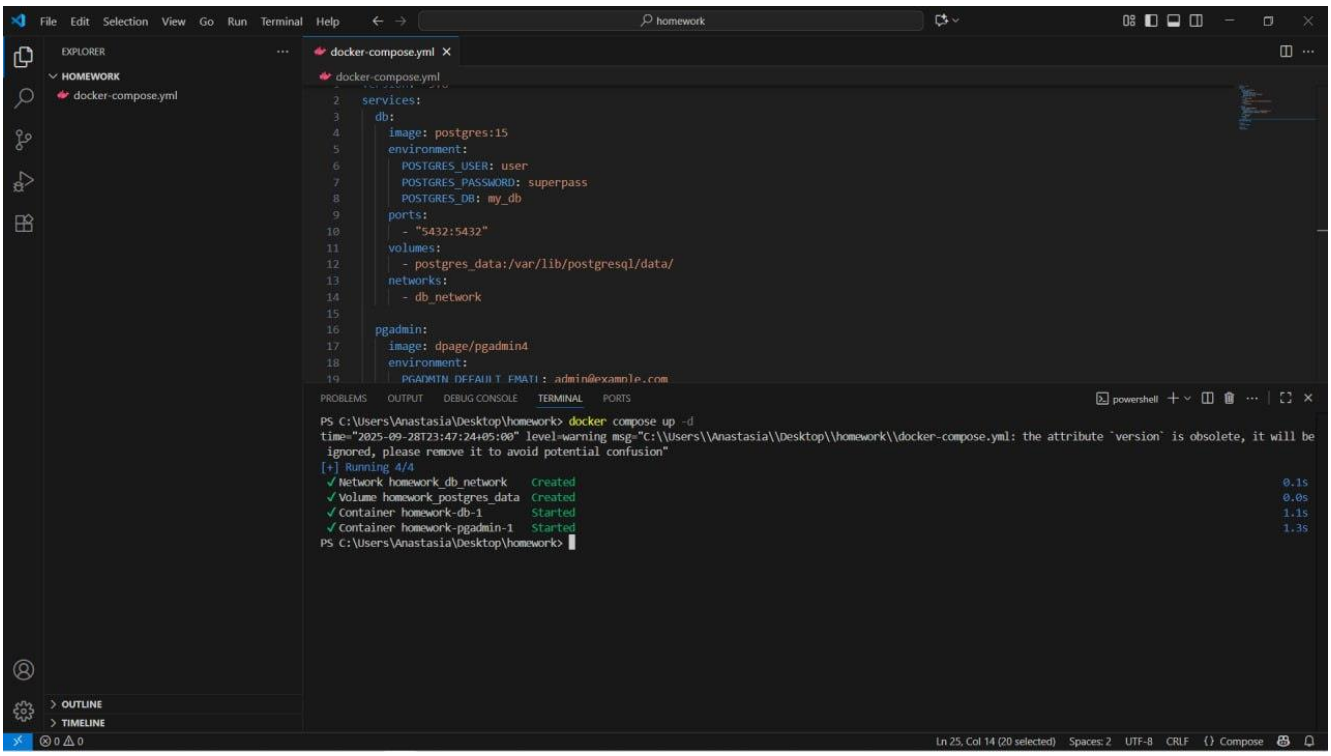


Успешная проверка – перезапуск тома показывает, что том действительно работает и данные сохраняются независимо от жизненного цикла контейнеров.

Таким образом, остановка контейнера не приводит к удалению данных, если они сохранены в томе.

Часть 5: Перенос конфигурации контейнеров в docker-compose.yaml

Был создан файл docker-compose.yml со следующей структурой



Описание конфигурации:

- Сервис db: PostgreSQL 15 с теми же переменными окружения, что и ранее
- Сервис pgadmin: Веб-интерфейс для управления PostgreSQL
- Volume postgres_data: Постоянное хранилище для данных БД
- Network db_network: Изолированная сеть для связи между контейнерами

Вопросы

1. Что такое docker?

Docker — программное обеспечение/ платформа для контейнеризации приложений. Позволяет «упаковать» приложение со всем его окружением и зависимостями (библиотеки, фреймворки, системные инструменты, настройки и т.д.) в изолированный контейнер.

Docker решает распространенную проблему –быстрое развертывание готовых программных продуктов, а также их масштабирования и перенос в другие среды с гарантированным сохранением стабильной работы.

2. Для чего нужны тома и сети docker?

- **Тома (Volumes)** используются для сохранения данных независимо от жизненного цикла контейнеров. Они позволяют сохранять данные (например, базы данных) и обмениваться данными между контейнерами.
- **Сети (Networks)** обеспечивают изолированное сетевое взаимодействие между контейнерами. Они позволяют контейнерам общаться друг с другом по именам, изолировать группы контейнеров и управлять доступом к внешнему миру.

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

Для подключения к запущенному контейнеру используется команда: **docker exec -it <container_name> /bin/bash**

4. Для чего нужен pgAdmin?

pgAdmin — веб-интерфейс для администрирования PostgreSQL. Позволяет управлять базами данных через графический интерфейс вместо командной строки: выполнять запросы, просматривать таблицы, настраивать пользователей.

Ссылка на git - https://github.com/AsunaIU/application-development-homework.git

Вывод:

В ходе комплексной лабораторной работы были успешно освоены ключевые аспекты работы с Docker:

- Установка и базовая настройка — Docker Desktop установлен и проверен на корректность работы.
- Работа с отдельными контейнерами — освоены команды управления жизненным циклом контейнеров, проброс портов, работа с переменными окружения.
- Работа с базами данных — запущен контейнер с PostgreSQL, созданы базы данных, таблицы и выполнены SQL-запросы.
- Сетевые взаимодействия — создана пользовательская сеть Docker, обеспечена связь между контейнерами.
- Сохранение данных с помощью томов – убедились на практике, что без томов данные в контейнерах являются временными и теряются при удалении контейнера. С

томами данные сохраняются независимо от жизненного цикла контейнеров. После перезапуска контейнеров все данные (таблицы users с записями) остаются нетронутыми

- Графические инструменты администрирования — развернут pgAdmin, настроено подключение к БД через веб-интерфейс.
- Оркестрация с Docker Compose — создан декларативный файл конфигурации для управления многоконтейнерным приложением.