**ЛАБОРАТОРНАЯ РАБОТА №7**

**DOCKER: ОСНОВЫ РАБОТЫ С КОНТЕЙНЕРАМИ**

**Цель работы**: освоить основные принципы работы с Docker, научиться создавать, запускать и управлять контейнерами, а также работать с Docker-образами.

**Задание:**

1. **Установка Docker**

* Установите Docker на вашу систему.
* Проверьте корректность установки командой docker --version.

1. **Работа с образами**

* Загрузите образ nginx из Docker Hub.
* Запустите контейнер на основе этого образа и откройте веб-страницу Nginx в браузере.

1. **Создание пользовательского образа**

* Напишите Dockerfile для создания образа с простым веб-приложением на Python (например, Flask).
* Соберите образ и запустите контейнер.

1. **Работа с томами (Volumes)**

* Создайте том и подключите его к контейнеру для сохранения данных.
* Убедитесь, что данные сохраняются после перезапуска контейнера.

1. **Сеть в Docker**

* Создайте два контейнера (например, веб-сервер и базу данных).
* Настройте сеть для взаимодействия между ними.

1. **Docker Compose**

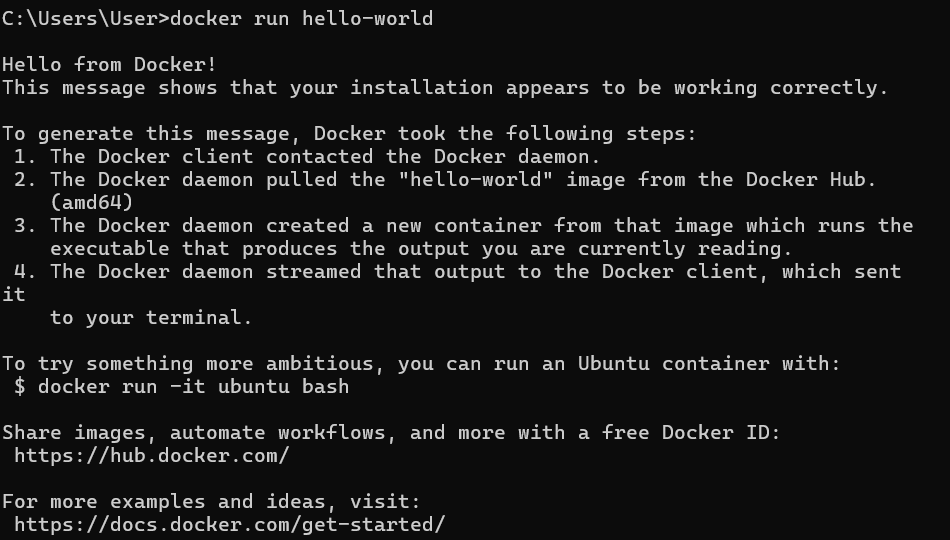
* Напишите файл docker-compose.yml для развёртывания многоконтейнерного приложения (например, веб-сервер + БД).
* Запустите приложение с помощью Docker Compose.

1. **Очистка ресурсов**

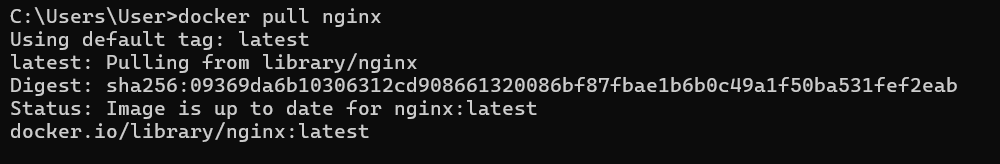
* Удалите все созданные контейнеры, образы и тома.

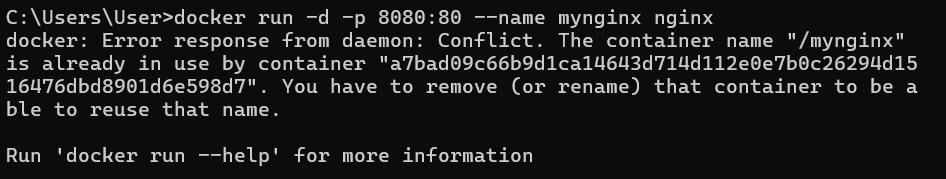
Этот отчёт представляет собой пошаговый анализ выполнения заданий по работе с Docker. Вначале была проведена установка Docker на систему и проверена его работоспособность командой docker --version.

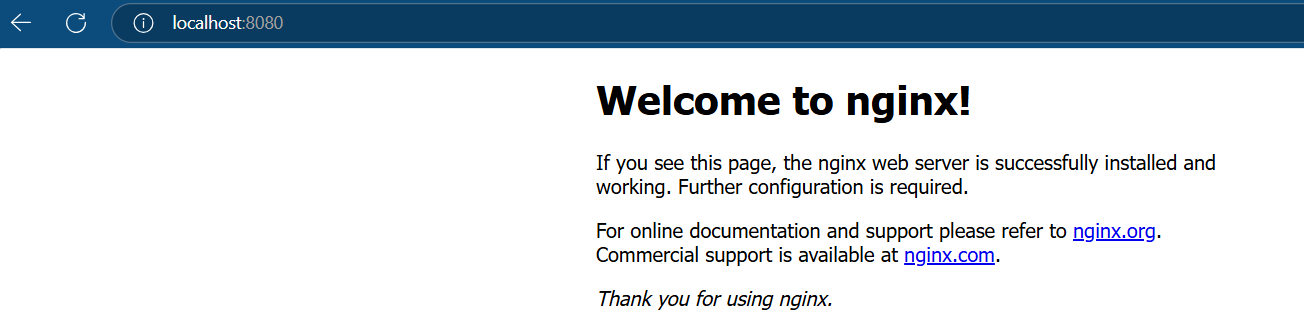




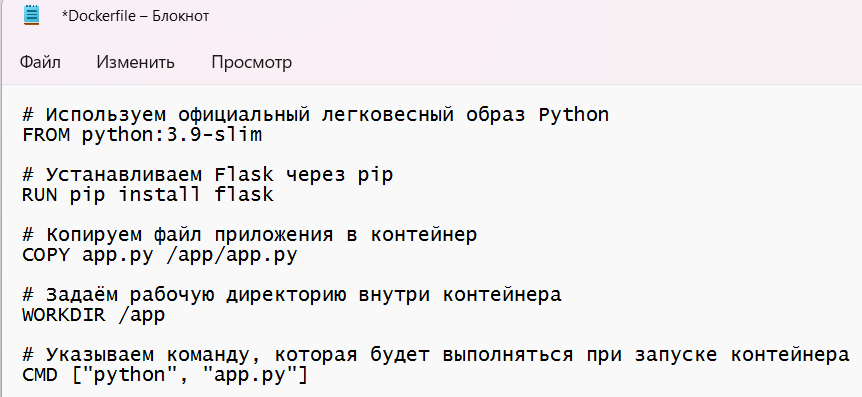
После этого был загружен образ nginx из официального репозитория Docker Hub с помощью команды docker pull nginx, а затем контейнер на его основе был запущен командой docker run -d -p 8080:80 --name mynginx nginx. Проверка веб-страницы http://localhost:8080 показала стандартное приветствие Nginx, что подтверждало успешный запуск контейнера.

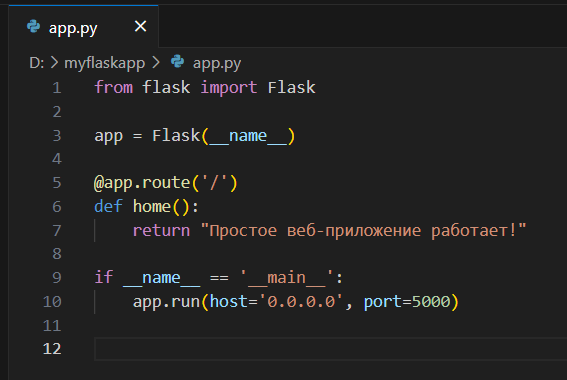


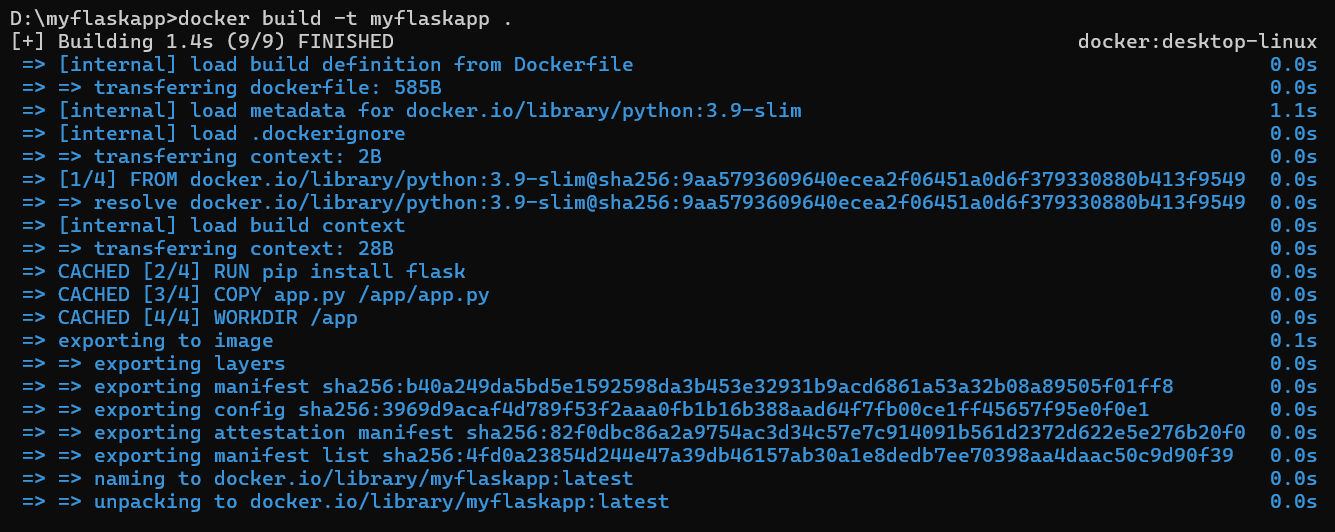


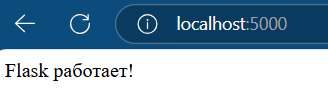


Следующим шагом стало создание пользовательского образа. Был написан Dockerfile, который задавал конфигурацию сборки веб-приложения на Flask, включал установку Python и необходимых библиотек. В коде app.py создавалось простое веб-приложение, которое при обращении к корневому маршруту отображало текст "Простое веб-приложение работает!". Образ был собран командой docker build -t myflaskapp ., а затем контейнер запущен с пробросом порта docker run -d -p 5000:5000 myflaskapp. Проверка веб-страницы http://localhost:5000 подтвердила успешную работу приложения.

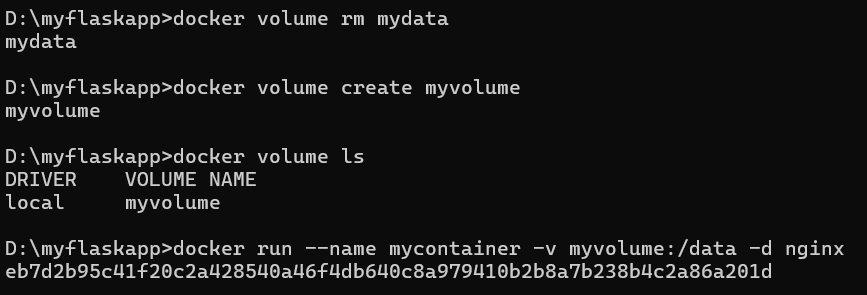


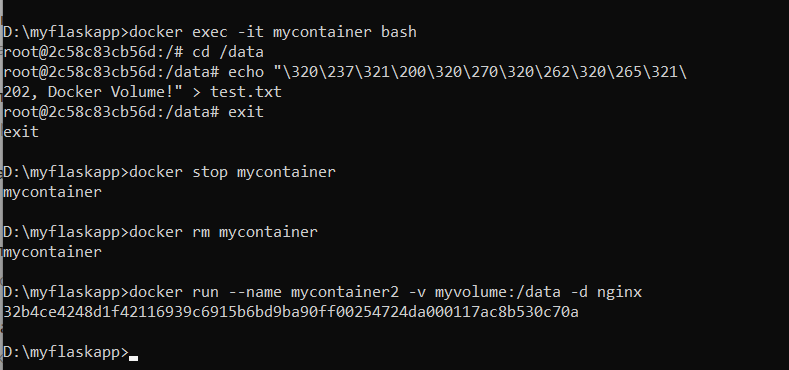


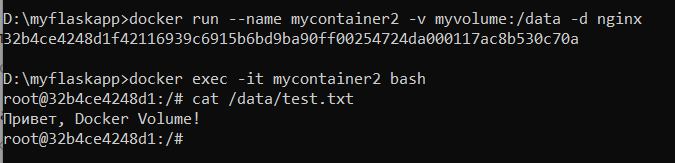


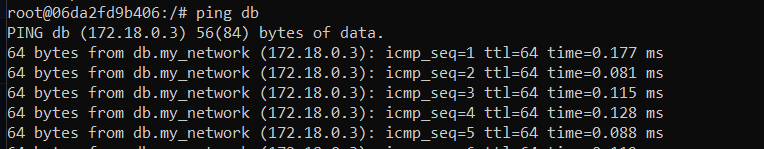


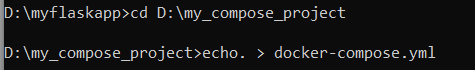
После этого была протестирована работа с томами. Создан том mydata (docker volume create mydata), который затем был подключён к контейнеру Nginx при его запуске. Проверка каталога /usr/share/nginx/html внутри контейнера показала наличие стандартных файлов index.html и 50x.html. Далее был создан файл test.html с тестовым содержимым, который оставался доступным после перезапуска контейнера, подтверждая успешность сохранения данных внутри тома.

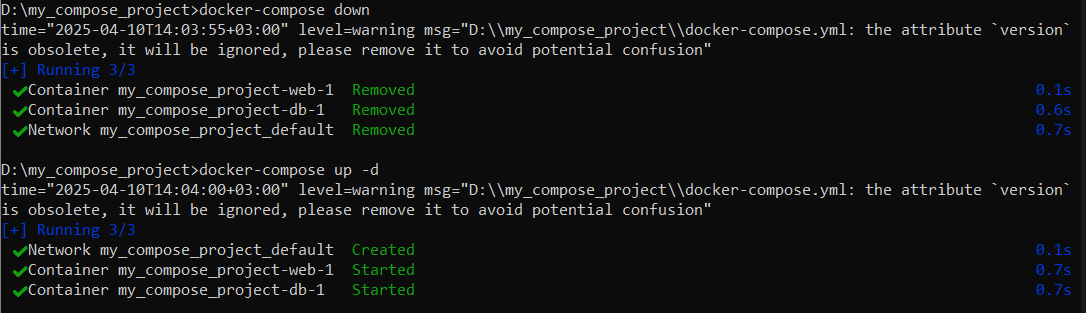


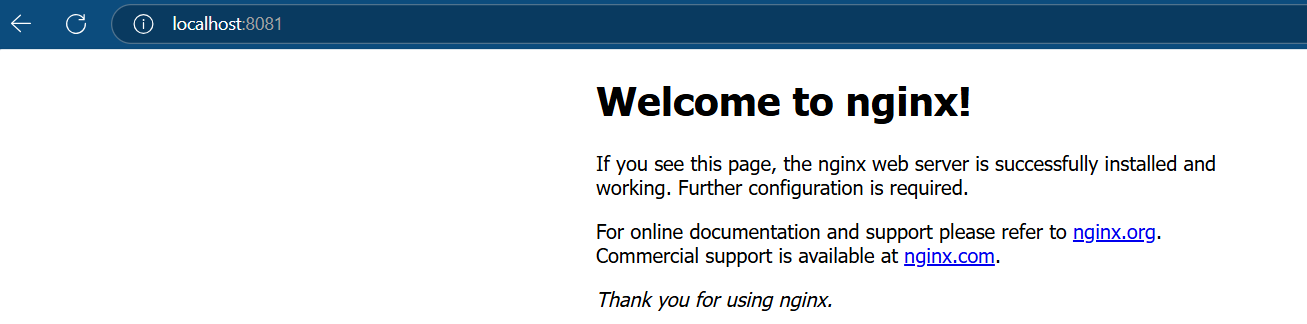






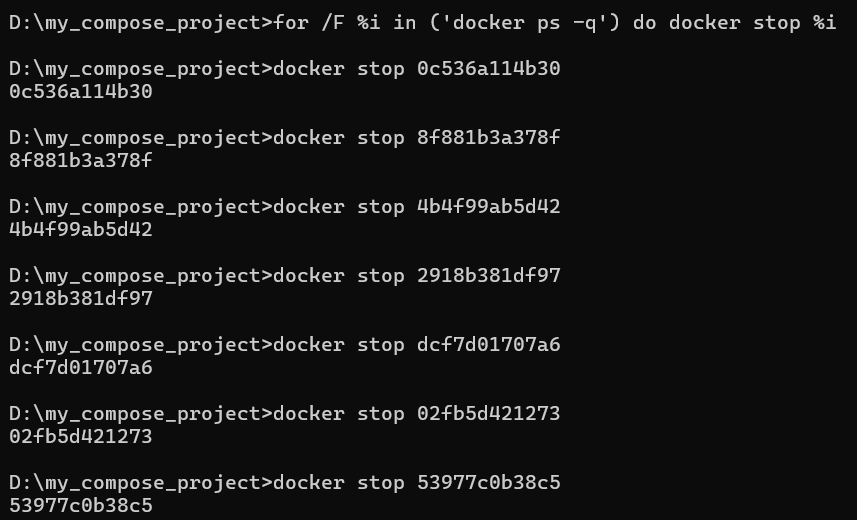




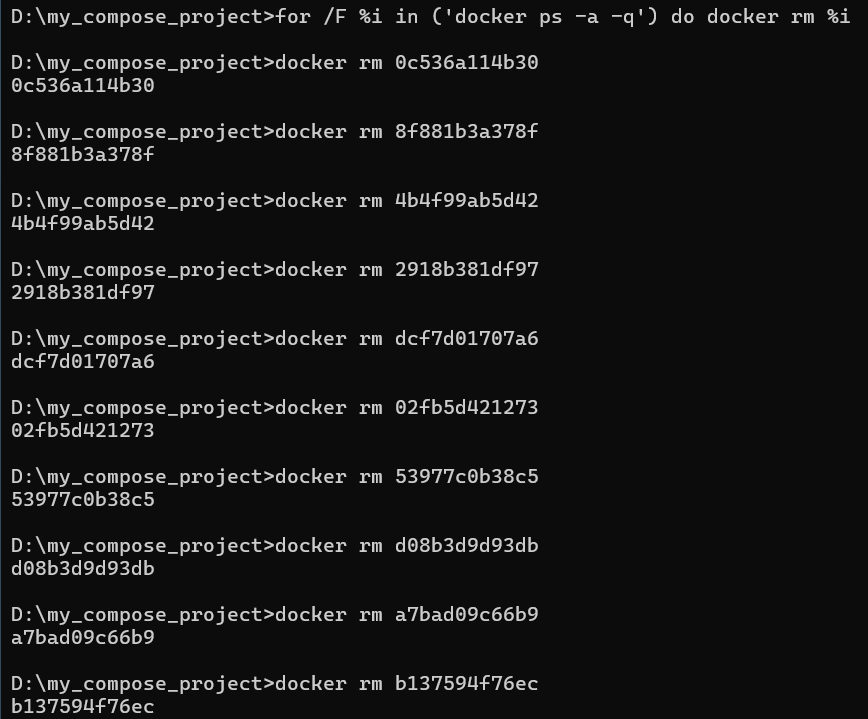


В заключении была проведена очистка ресурсов, чтобы удалить созданные контейнеры, образы и тома.

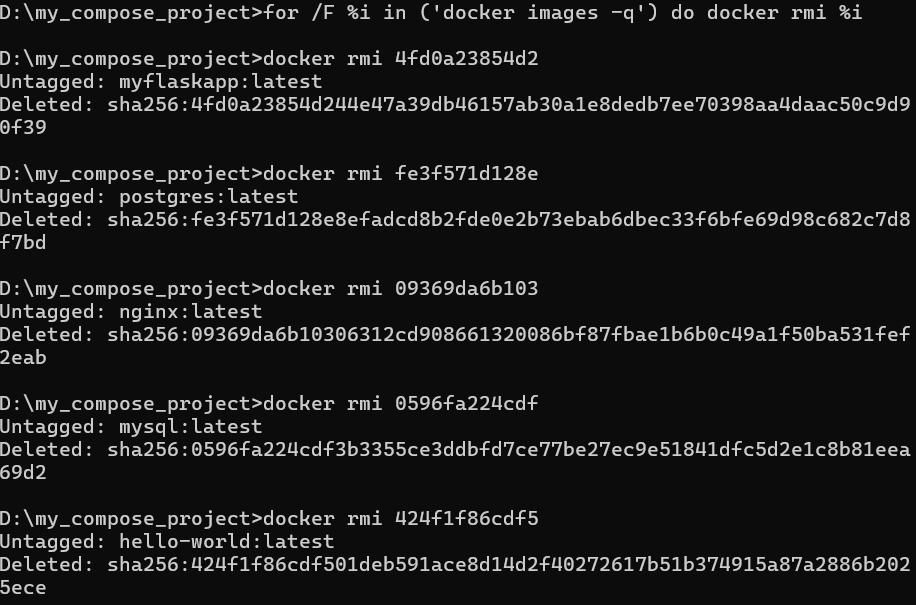
Остановка контейнеров



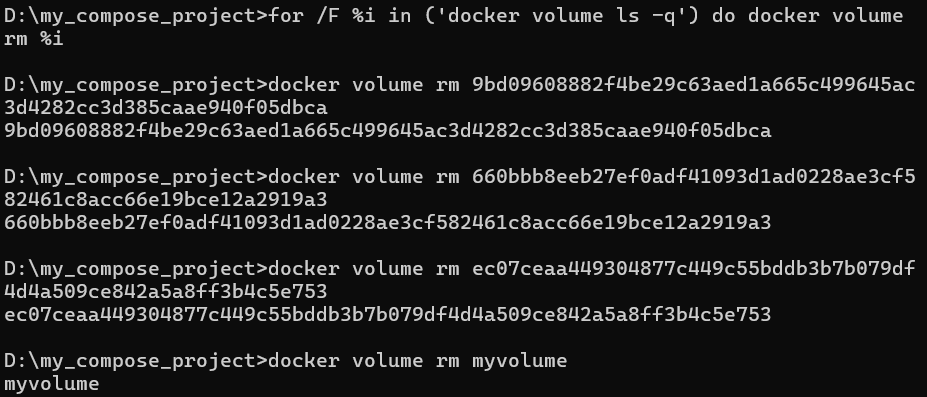
Удаление контейнеров



Удаление образов



Удаление томов



Удаление сети

{D737AAB9-3B97-4A98-AC3C-DEDC679920AA}

