# ЛАБОРАТОРНАЯ РАБОТА № 8 ТЕМА: 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

### Условие

1. Установка Docker

o Установите Docker на вашу систему.

o Проверьте корректность установки командой docker --version.

2. Работа с образами

o Загрузите образ nginx из Docker Hub.

o Запустите контейнер на основе этого образа и откройте веб-страницу Nginx в браузере.

3. Создание пользовательского образа

o Напишите Dockerfile для создания образа с простым веб-приложением на Python (например, Flask).

o Соберите образ и запустите контейнер.

4. Работа с томами (Volumes)

o Создайте том и подключите его к контейнеру для сохранения данных.

o Убедитесь, что данные сохраняются после перезапуска контейнера.

5. Сеть в Docker

o Создайте два контейнера (например, веб-сервер и базу данных).

o Настройте сеть для взаимодействия между ними.

6. Docker Compose

o Напишите файл docker-compose.yml для развёртывания многоконтейнерного приложения (например, веб-сервер + БД).

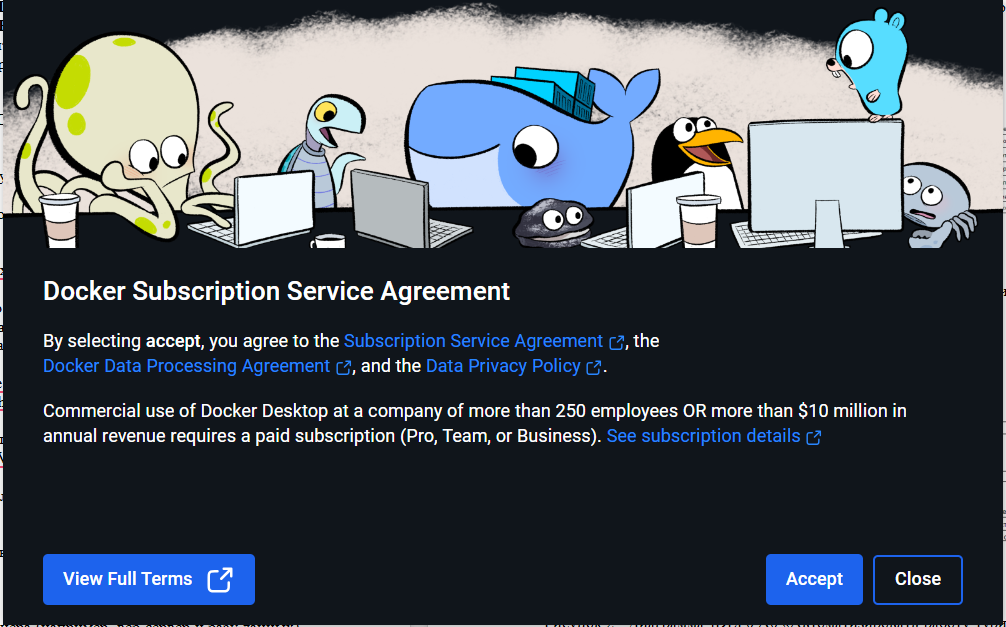
o Запустите приложение с помощью Docker Compose.

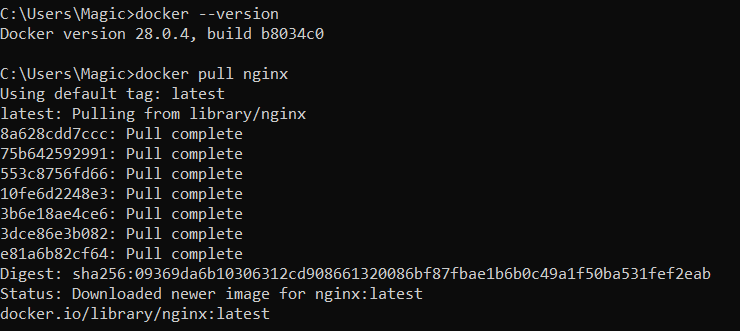
7. Очистка ресурсов

o Удалите все созданные контейнеры, образы и тома. Все задания должны быть выполнены и представлены в виде отчёта с описанием шагов и скриншотами результатов.

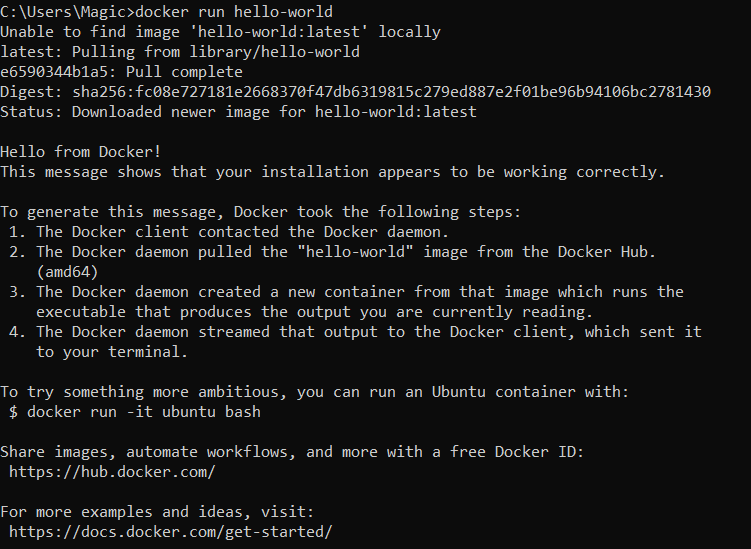
### Результат

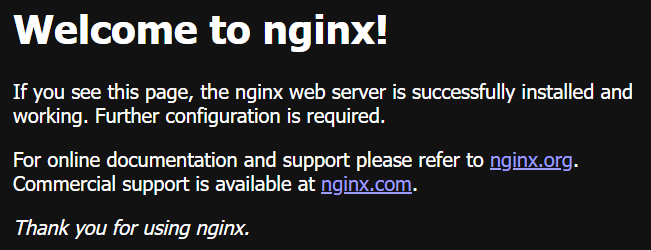
1. Этот отчёт представляет собой пошаговый анализ выполнения заданий по работе с Docker. Вначале была проведена установка Docker на систему и проверена его работоспособность командой docker --version.



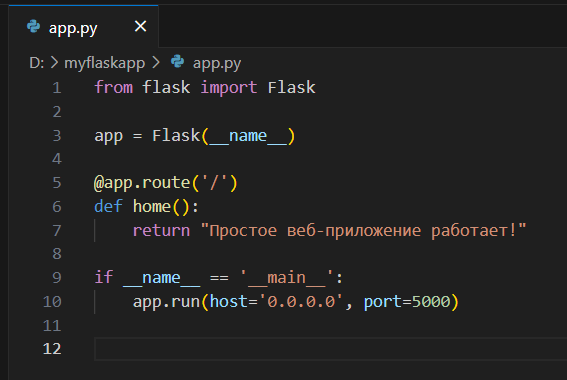
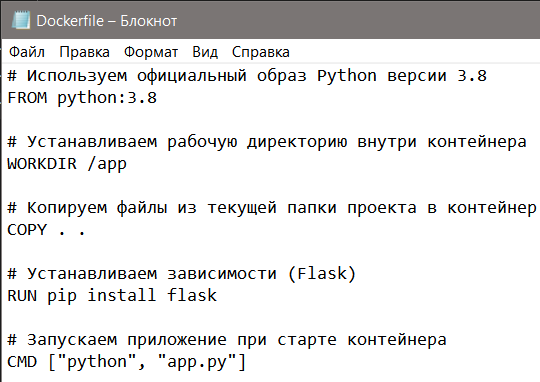


1. После этого был загружен образ nginx из официального репозитория Docker Hub с помощью команды docker pull nginx, а затем контейнер на его основе был запущен командой docker run -d -p 8080:80 --name mynginx nginx. Проверка веб-страницы http://localhost:8080 показала стандартное приветствие Nginx, что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

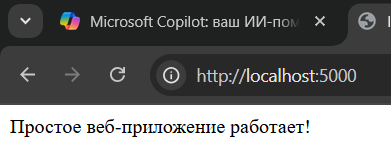
{69EED769-50B5-4748-90F8-B7E5ED5ACEDF}



1. 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.







1. 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.

D:\myflaskapp>docker volume create mydata

mydata

D:\myflaskapp>docker run -d -p 8080:80 -v mydata:/usr/share/nginx/html --name mynginx nginx

35d9dea6a2049f2efb35893a03e9aa0a2d84be744269f92b7cc5ba6344e3abee

D:\myflaskapp>docker restart mynginx

mynginx

D:\myflaskapp>docker exec -it mynginx ls /usr/share/nginx/html

50x.html index.html

D:\myflaskapp>docker exec -it mynginx cat /usr/share/nginx/html/index.html

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

html { color-scheme: light dark; }

body { width: 35em; margin: 0 auto;

font-family: Tahoma, Verdana, Arial, sans-serif; }

</style>

</head>

<body>

<h1>Welcome to nginx!</h1>

<p>If you see this page, the nginx web server is successfully installed and

working. Further configuration is required.</p>

<p>For online documentation and support please refer to

<a href="http://nginx.org/">nginx.org</a>.<br/>

Commercial support is available at

<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>

</body>

</html>

D:\myflaskapp>docker exec -it mynginx bash

echo "Тестовая страница Nginx" > /usr/share/nginx/html/test.html

exit

root@35d9dea6a204:/# echo "\320\242\320\265\321\201\321\202\320\276\320\

262\320\260\321\217 \321\201\321\202\321\200\320\260\320\275\320

\270\321\206\320\260 Nginx" > /usr/share/nginx/html/test.html

root@35d9dea6a204:/# exit

exit

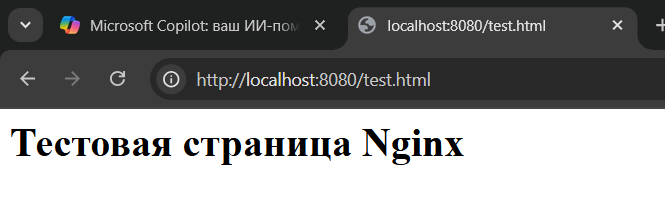
D:\myflaskapp>docker exec -it mynginx file -i /usr/share/nginx/html/test.html

D:\myflaskapp>docker restart mynginx

mynginx

D:\myflaskapp>docker exec -it mynginx ls /usr/share/nginx/html

50x.html index.html test.html



1. На следующем этапе была протестирована работа сети между контейнерами. Создана виртуальная сеть `mynetwork`, затем запущены два контейнера: веб-сервер `nginx` и база данных `mysql`, подключённые к одной сети. Проверка связи между контейнерами проводилась с помощью `curl http://webserver` внутри контейнера MySQL. Полученный ответ веб-сервера подтвердил успешное взаимодействие между контейнерами.

D:\myflaskapp>docker network create mynetwork

a325f6675de24e6026d81639950533d86abe3455395dac09f1af8cb97787866

D:\myflaskapp>docker network ls

NETWORK ID NAME DRIVER SCOPE

acb5fdbfb847 bridge bridge local

d58bfb67d7eb host host local

a325f6675de2 mynetwork bridge local

a800c281bfe1 none null local

D:\myflaskapp>docker run -d --network=mynetwork --name webserver nginx

06bf8fc1b9d1c1538cc7b796313e401359437c2b526a25a4dd6070fb86b8ccad

D:\myflaskapp>docker run -d --network=mynetwork --name database -e MYSQL\_ROOT\_PASSWORD=example mysql

D:\myflaskapp>docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

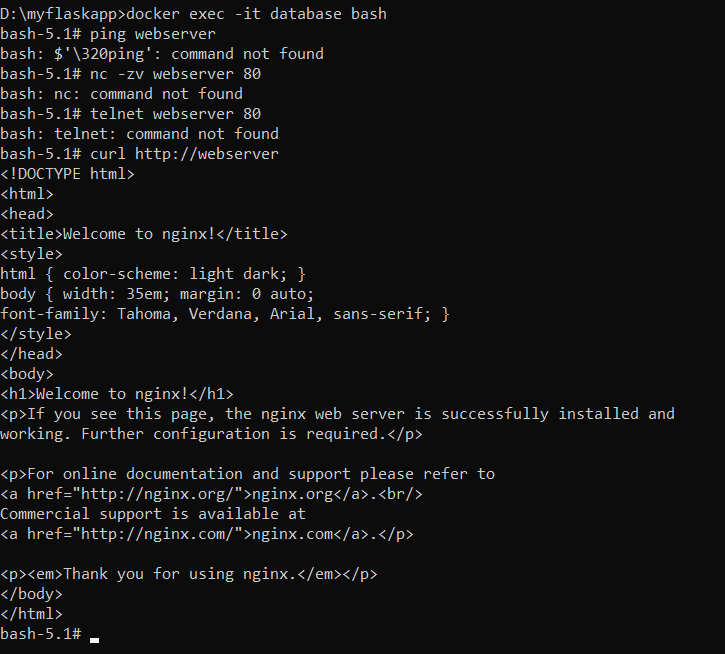
12f18fd9ceff mysql "docker-entrypoint.s…" 28 seconds ago Up 25 seconds 3306/tcp, 33060/tcp database

06bf8fc1b9d1 nginx "/docker-entrypoint.…" About a minute ago Up About a minute 80/tcp webserver

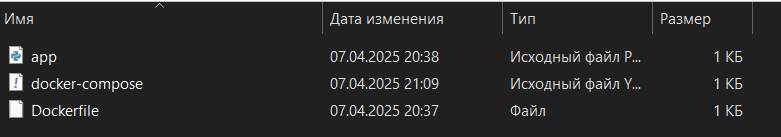
35d9dea6a204 nginx "/docker-entrypoint.…" 9 minutes ago Up 3 minutes 0.0.0.0:8080->80/tcp mynginx

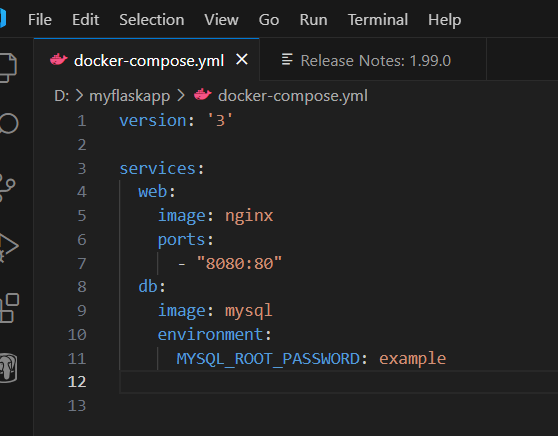
0ee79c78eaf7 myflaskapp "python app.py" 17 minutes ago Up 17 minutes 0.0.0.0:5000->5000/tcp amazing\_lamport

D:\myflaskapp>docker exec -it database bash



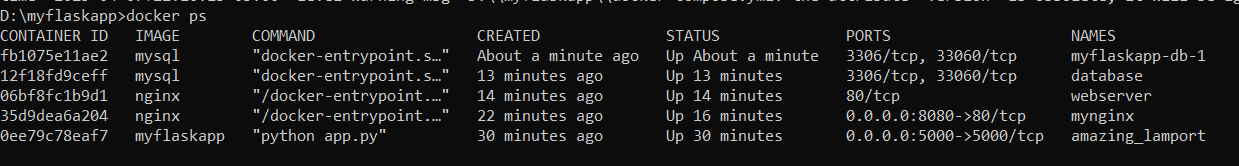
1. Далее для удобного управления контейнерами был написан `docker-compose.yml`, который включал описание сервисов `web` и `db`. Запуск приложения выполнялся с помощью `docker-compose up -d`, после чего проверка через `docker ps` показала, что оба контейнера успешно работают.





docker-compose up -d

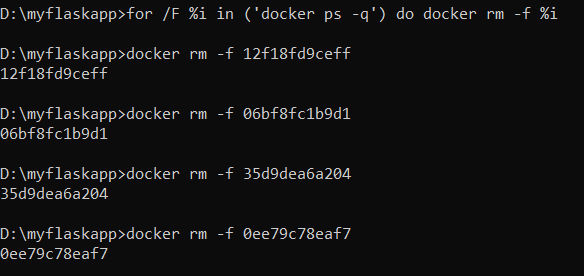
docker ps

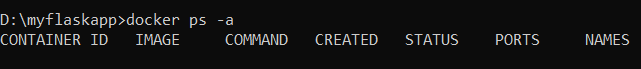




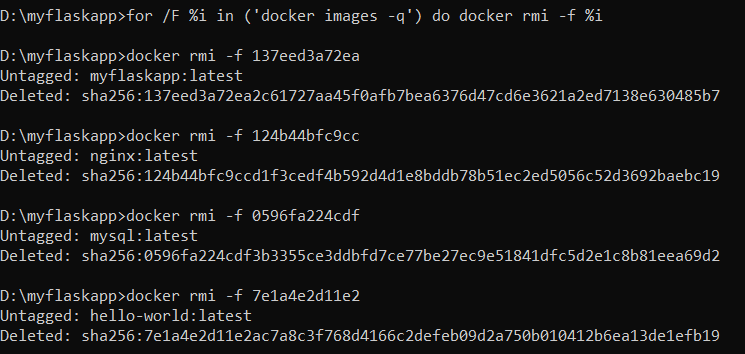
1. В заключении была проведена очистка ресурсов, чтобы удалить созданные контейнеры, образы и тома.

Удаление контейнеров:



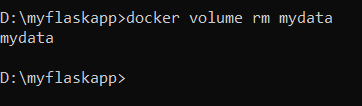


Удаление образов:





Удаление томов:





Таким образом, все этапы работы с Docker были выполнены в полном объёме. Был протестирован запуск контейнеров, работа с пользовательскими образами, создание и хранение данных в томах, настройка сетевого взаимодействия, управление контейнерами через Docker Compose и очистка ресурсов. После выполнения всех команд система полностью очищена от временных файлов, а процессы работы с Docker освоены и продемонстрированы на практике.