Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г. В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

Специальность: 09.02.07 Информационные системы и программирование

Квалификация: Программист

Листов:

|  |  |
| --- | --- |
| Выполнил студент | Проверил преподаватель |
| Группы П50-9-21 | Образцова К.С. |
| Давыдов М.К. | «\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 года |

Оглавление

[Практическая работа № 1 3](#_Toc184989974)

[Пракичнеская работа № 2 9](#_Toc184989975)

[Практическая работа № 3 15](#_Toc184989976)

[1.1 Интерфейс docker desktop 18](#_Toc184989977)

[1.2 Еще больше образов 21](#_Toc184989978)

[Практическая №4 29](#_Toc184989979)

# Практическая работа № 1

Цель:

Работа с Git

1. Конспект двух лекций

2. Продемонстрировать конфигурацию Git

3. Создание локального репозитория

4. Продемонстрировать работу основных команд для работы с файлами в репозитории

5. Создание нескольких различных веток

6. Описать работу с указателями HEAD

1) Демонстрация конфигурации git

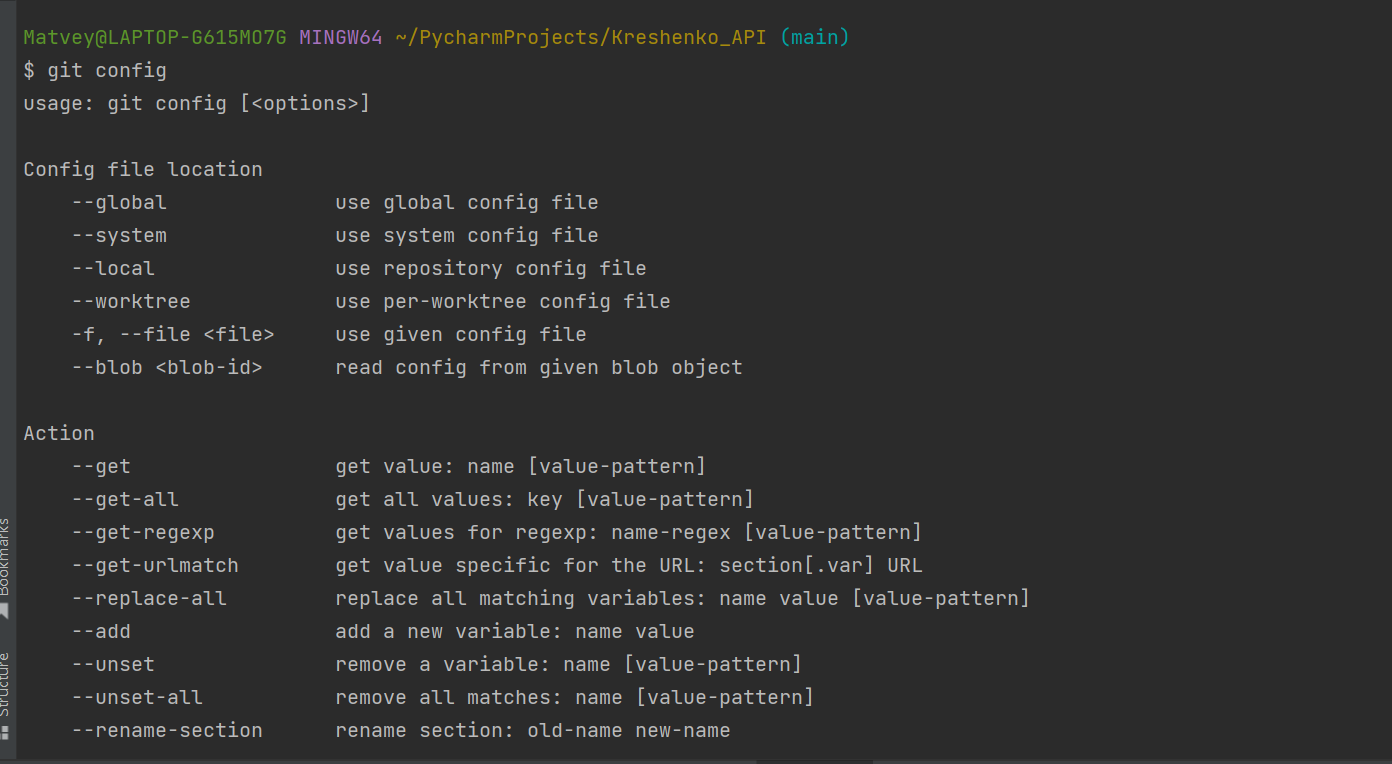


Рисунок 1 - Демонстрация конфигурации git

2) Создание репозитория

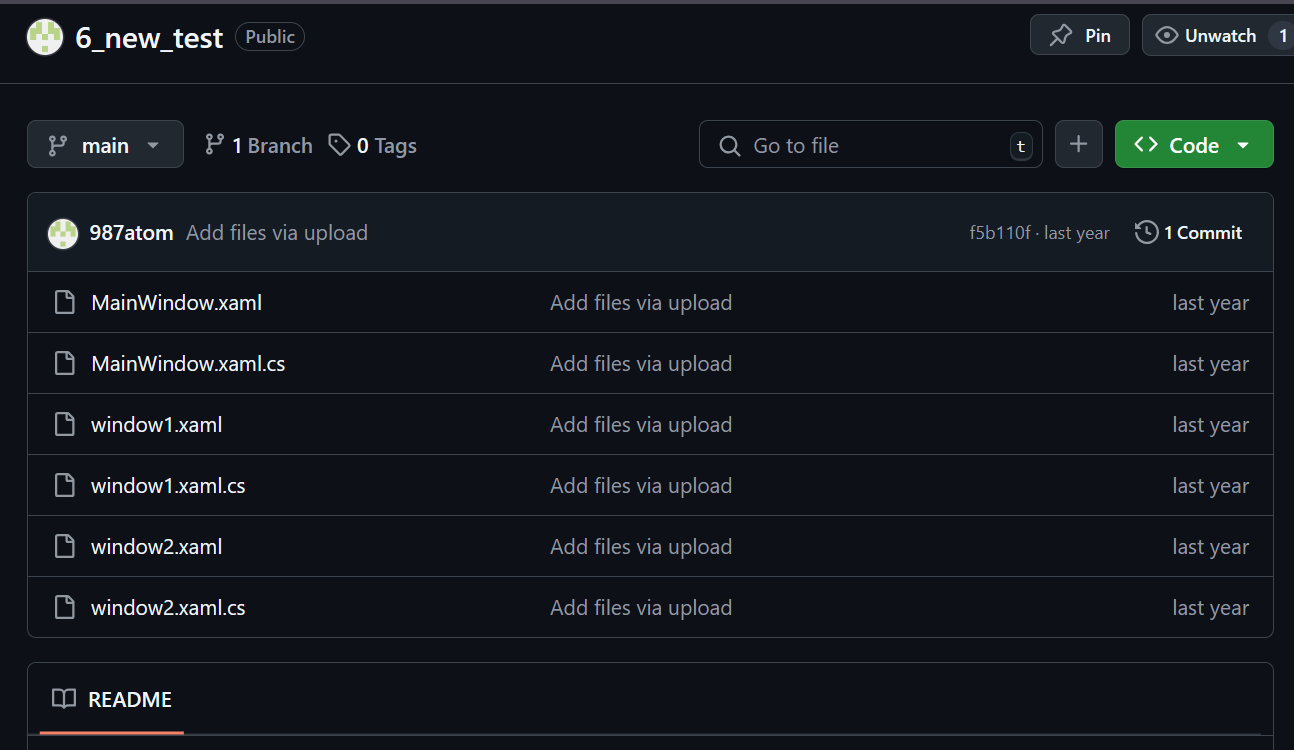


Рисунок 2 – репозиторий

3) Основные команды git

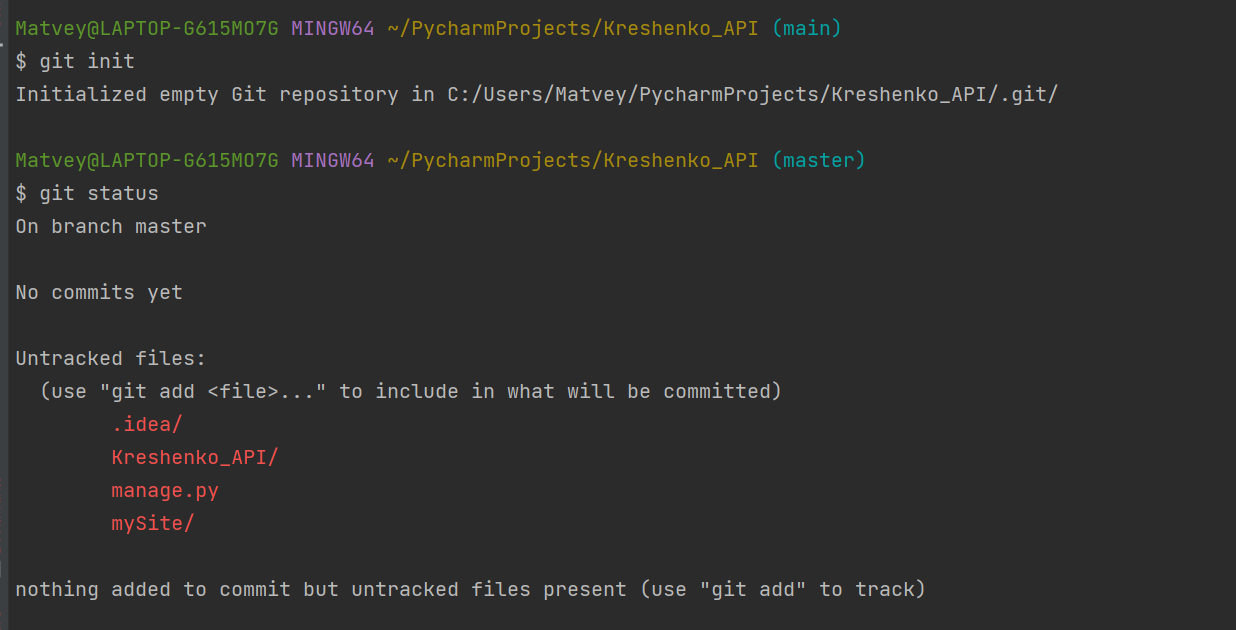


Рисунок 3 - Основные команды git

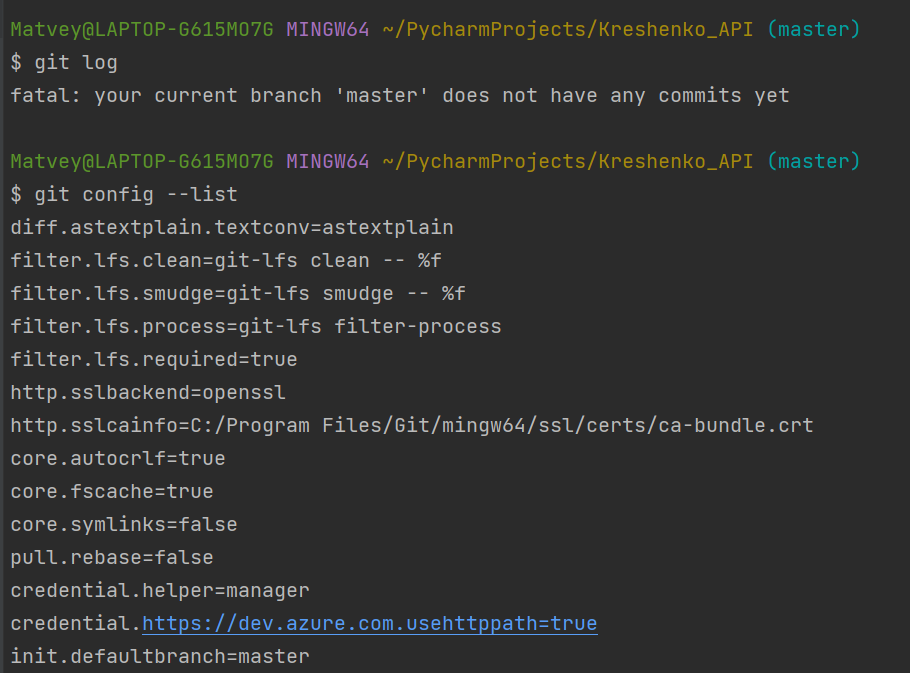


Рисунок 4 - Основные команды git

4) Несколько веток

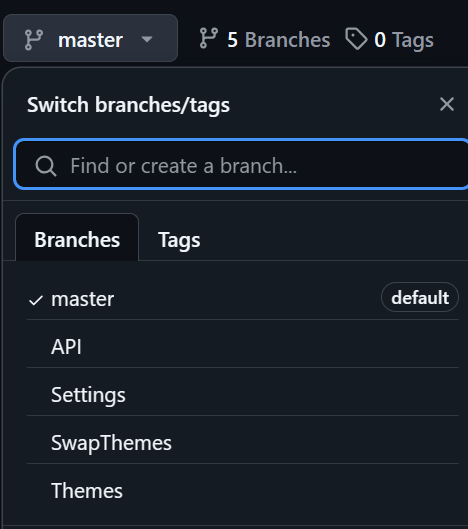


Рисунок 5 – ветки

5) описание работы с head

Git commit

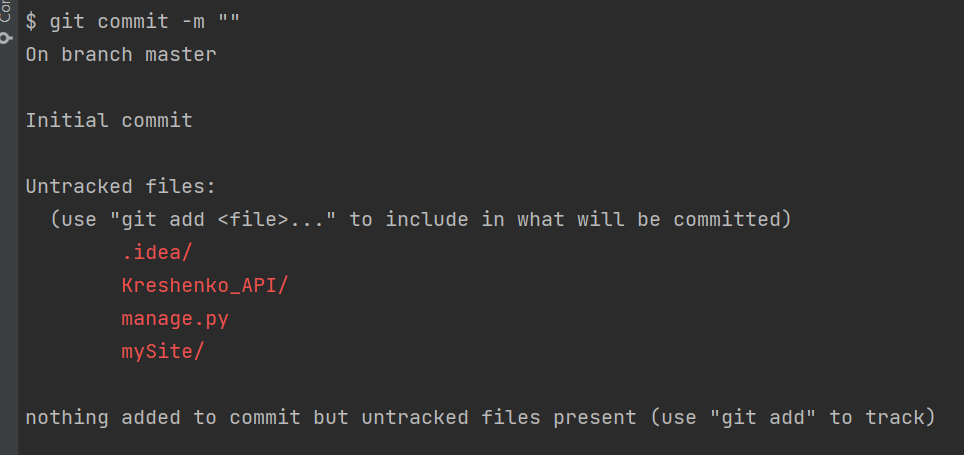


Рисунок 6 - Git commit

Создание ветки

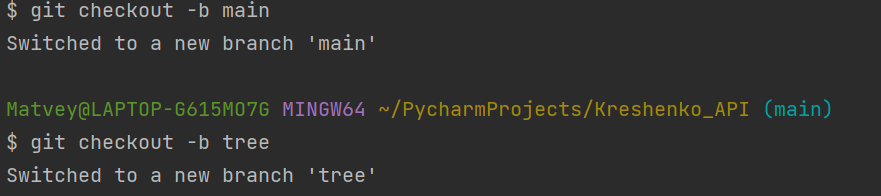


Рисунок 7 - ветки

Список комитов

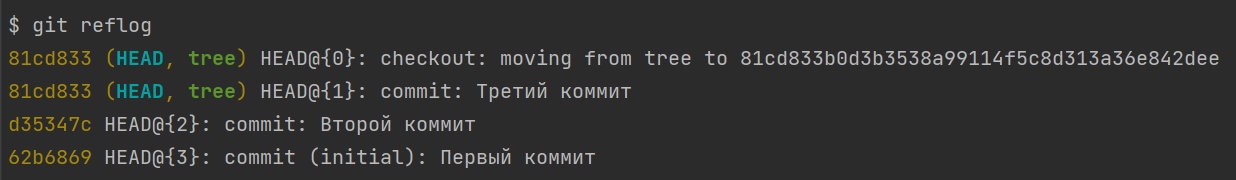


Рисунок 8 – комиты

Переключение между коммитами

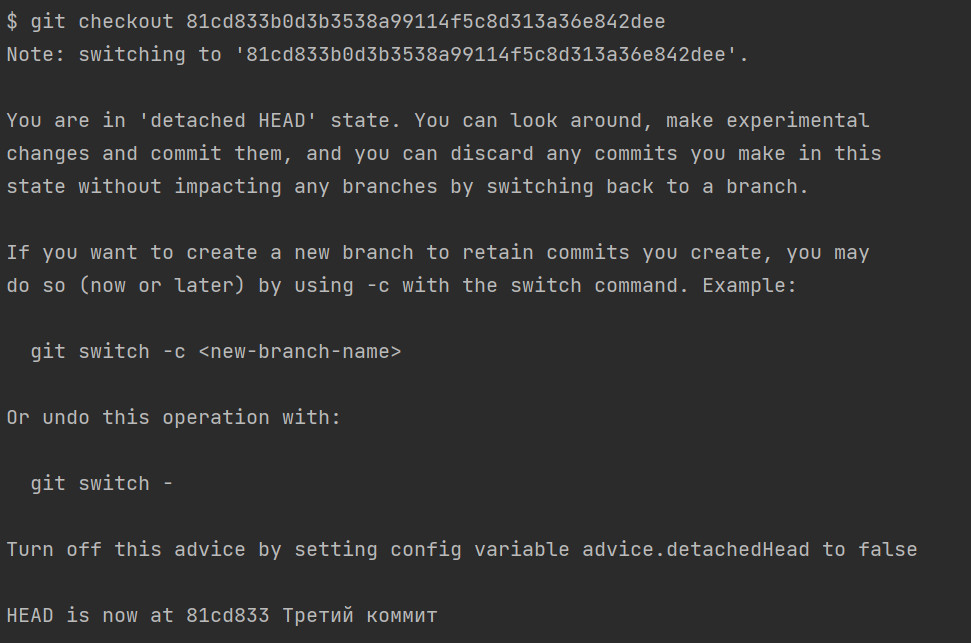


Рисунок 9 - переключение между коммитами

Вывод: Научился работать с основными командами git. Так же переключаться между ветками и коммитами.

# Пракичнеская работа № 2

Цель работы: Научиться работать с удаленным репозиторием git hub

1. Создание репозитория на сайта



Рисунок 12 – Создание репозитория

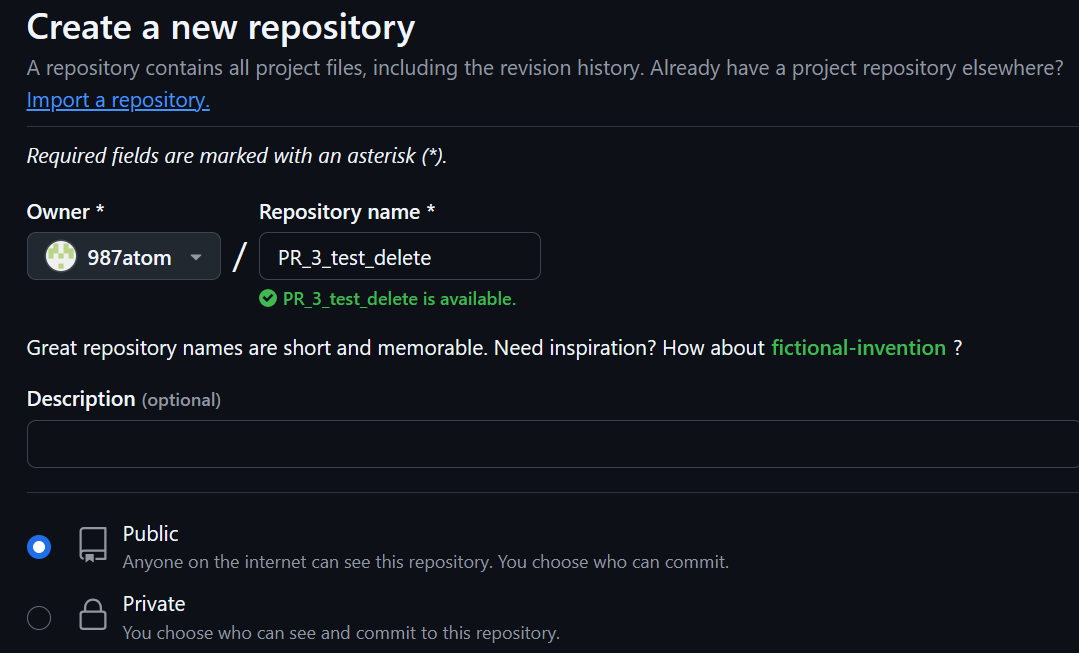


Рисунок 13 – Настройки репозитория

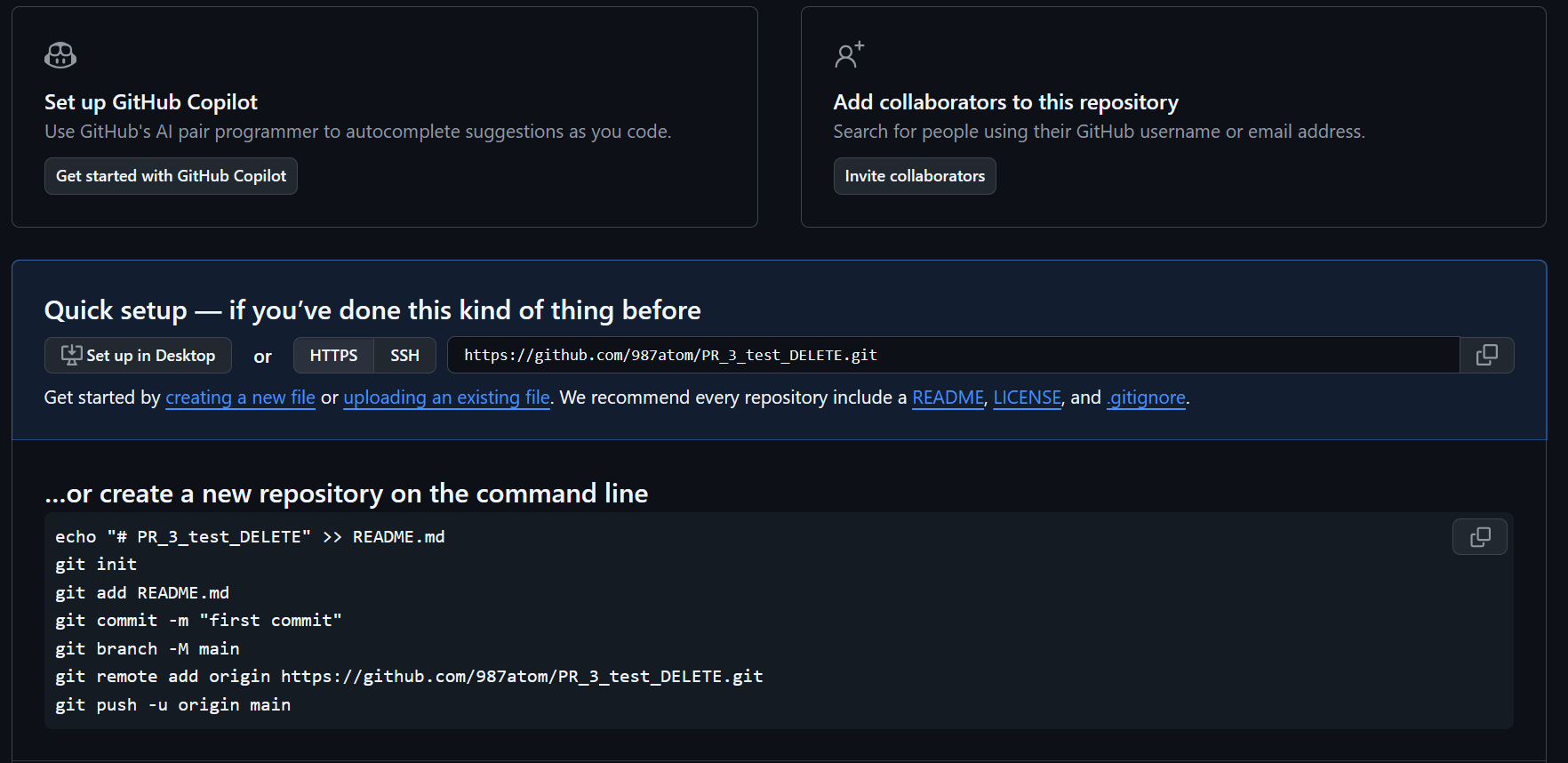


Рисунок 14 – Репозиторий создан

1. Основные команды
   1. Git remote add



Рисунок 15 – Добавление удаленного репозитория

* 1. Git remote –v

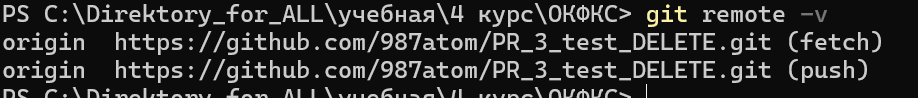


Рисунок 16 – Ссылки

* 1. git push

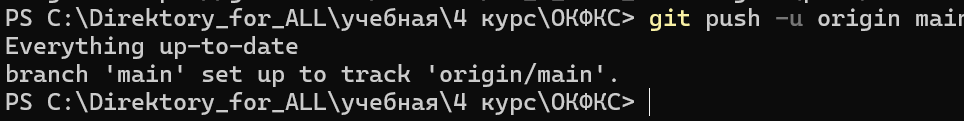


Рисунок 17 – Добавление изменений в репозиторий

* 1. git push –all

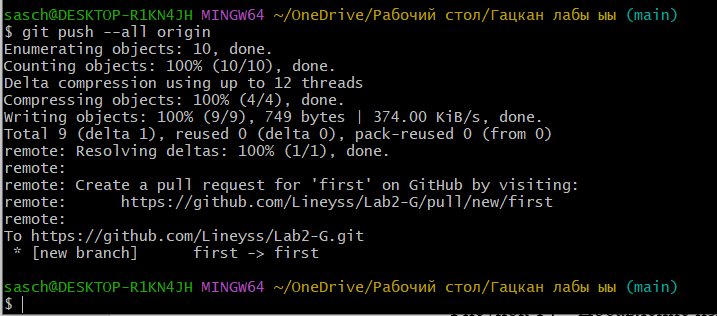


Рисунок 18 – Добавление всех веток в удаленный репозиторий

* 1. git push –f

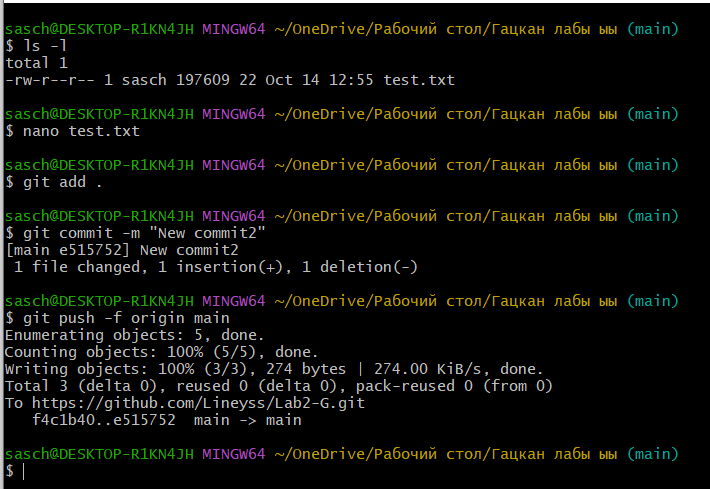


Рисунок 19 – Отправка коммитов на удаленый репозиторий

* 1. git clone

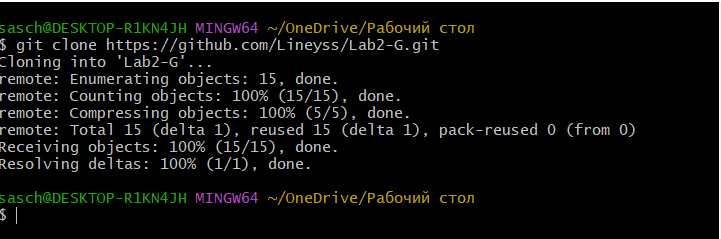


Рисунок 20 – Клонирование ветки

* 1. git clone –b

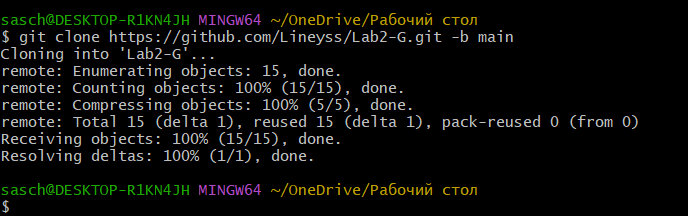


Рисунок 21 –Клонирование репозитория с переходом на конкретную ветку

* 1. git clone –single-branch –b

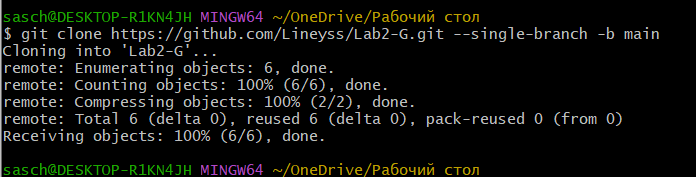


Рисунок 22 – Клонирование конкретной ветки репозитория

1. merge и pull реквесты

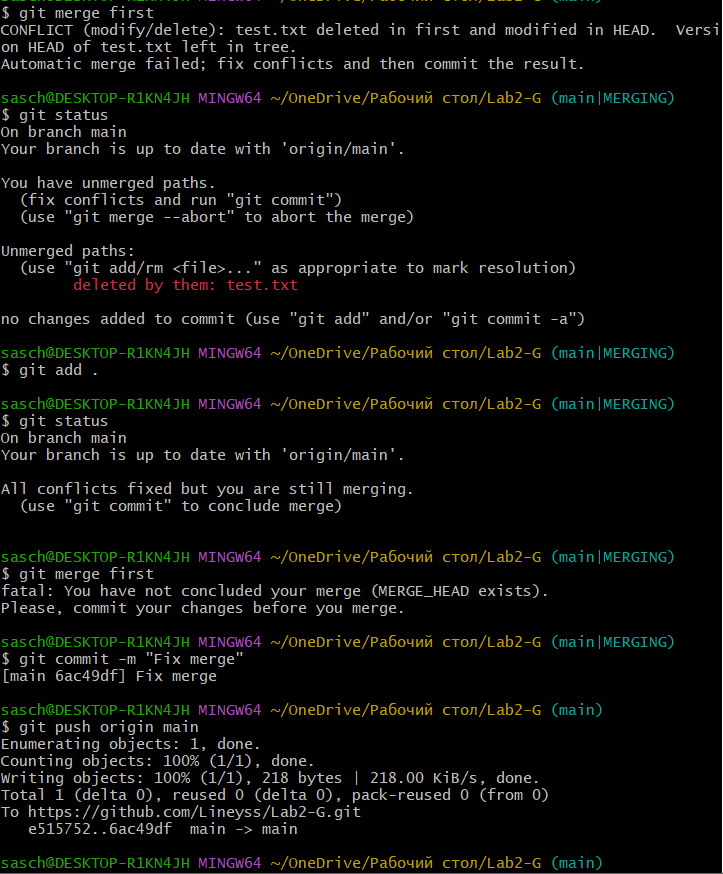


Рисунок 23 – merge веток

Вывод: Научился работать с удаленным репозиторием git hub

# Практическая работа № 3

Цель работы: Выполнить сборки и развертывание контейнеров из примера, и описать своими словами то, что именно вы сделали и как. Описать вкладки в Docker (включая вкладки внутри собранного образа и контейнера). Написать свои Doсkerfile и собрать контейнеры с программами из архива.

1. Контейнеры из примера
   1. Python

Создаем Dockerfile в директории.

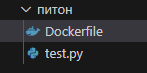


Рисунок 24 – Dockerfile

В нем пишем следующий код:

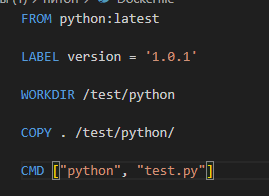


Рисунок 25 – Создание контейнера

From – образ на основе которого создаем свой образ

LABLE – метка образа

WORKDIR – создает директорию в образе и переходит в нее

COPY – копирует файл/папку в образ

CMD – команда, вызываемая при запуске контейнера

Собираем образ:

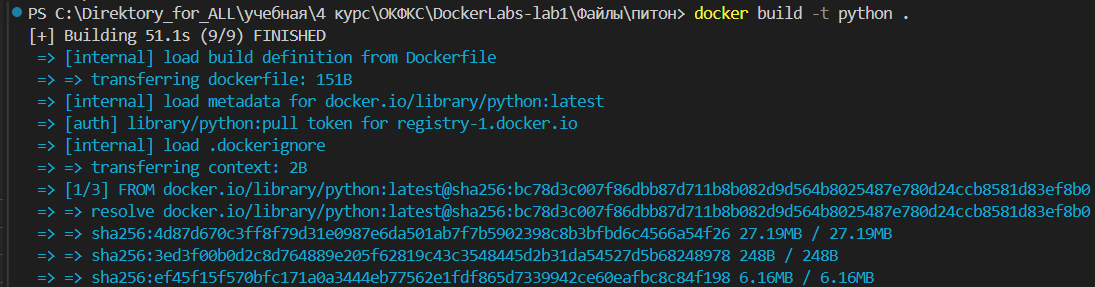


Рисунок 26 – Собираем образ

docker build – команда для сборки образа

-t python – ключ который задает имя собираемому образу



Рисунок 27 – Собранный образ

Запускаем контейнер:

Контейнер – оболочка работающая на основе image, в которой работают наши программы. Запуск программы:



Рисунок 28 – Запуск контейнера

--nane – название контейнера

python – имя образа на основе которого запускаем контейнер



Рисунок 29 – Созданный контейнер

* 1. Html

Создаем Dockerfile в директории

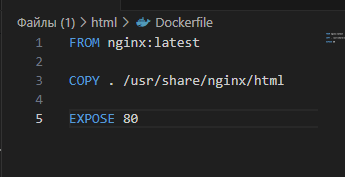


Рисунок 30 – Dockerfile

From – образ на основе которого создаем свой образ

COPY - копирование содержимое папки в образ

EXPOSE 80 – указываем что образ использует 80 порт и нужно запускать образ на этом же порту

Собираем образ:

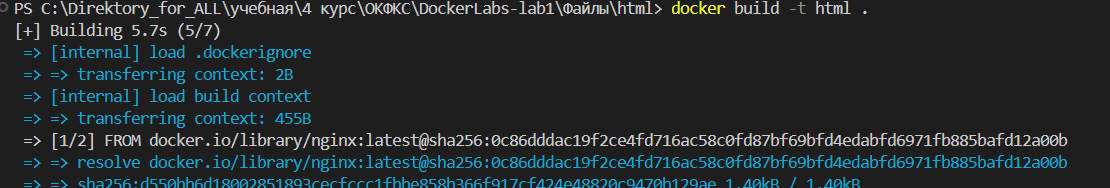
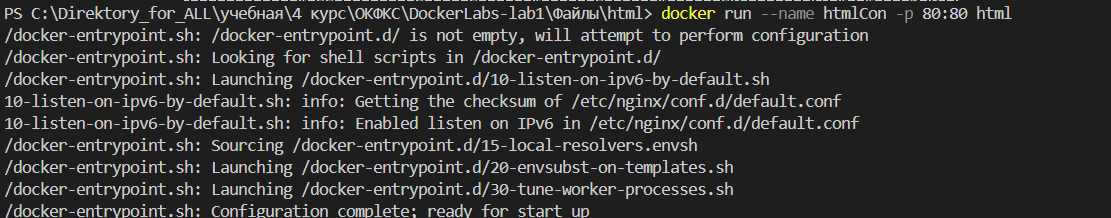


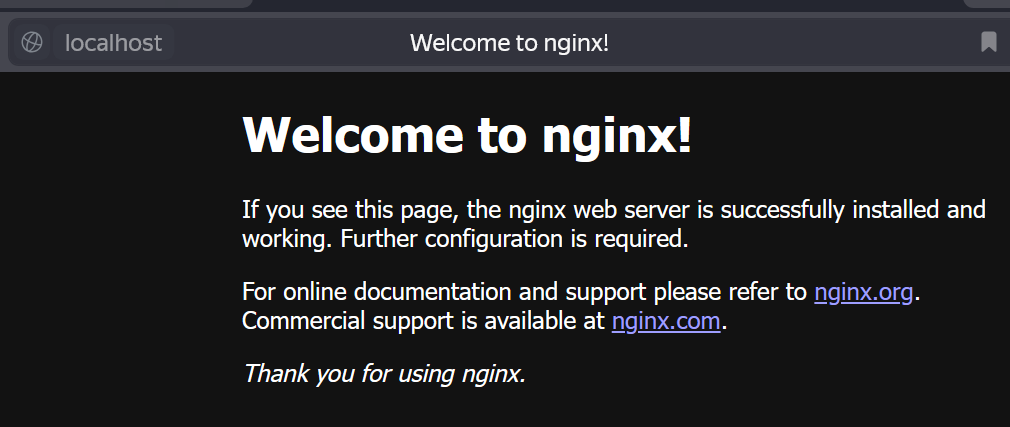
Рисунок 31 – Собираем образ

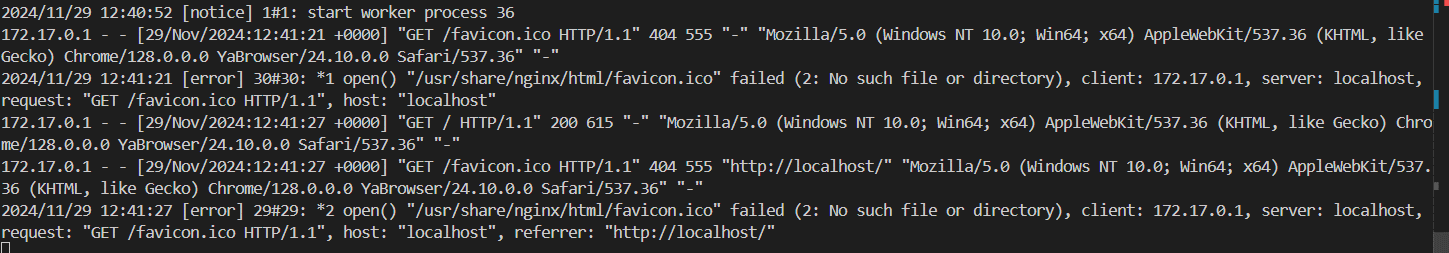


Рисунок 32 – Образ в docker image

Запуск контейнера:







## Интерфейс docker desktop

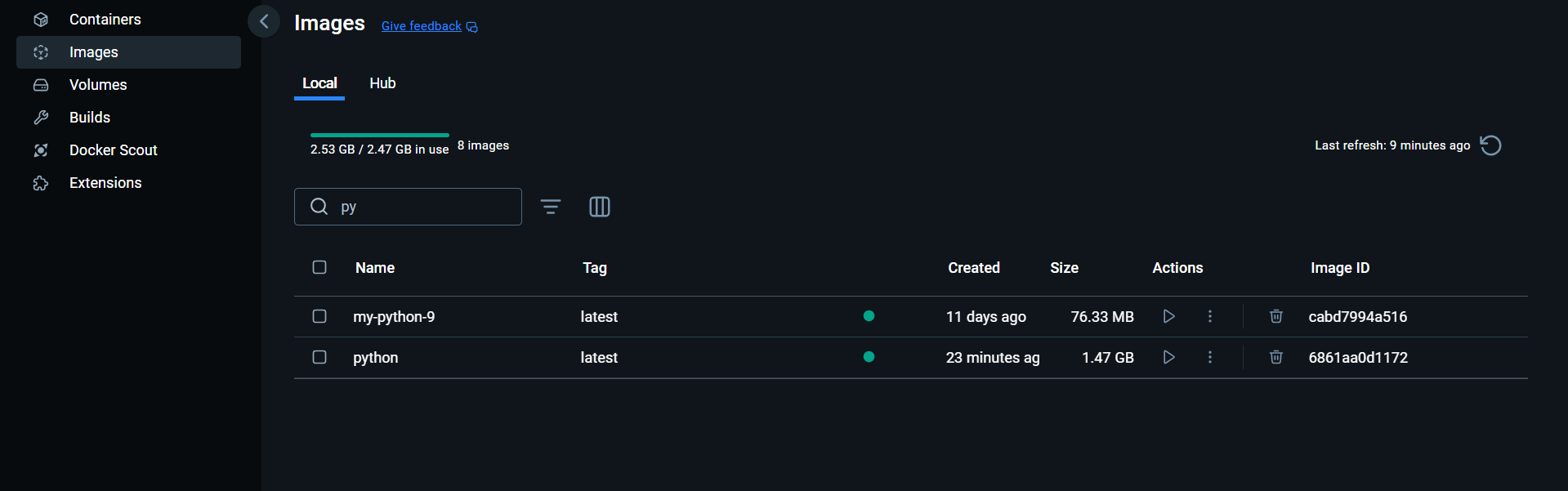


Рисунок 33 –Страница образов локальных

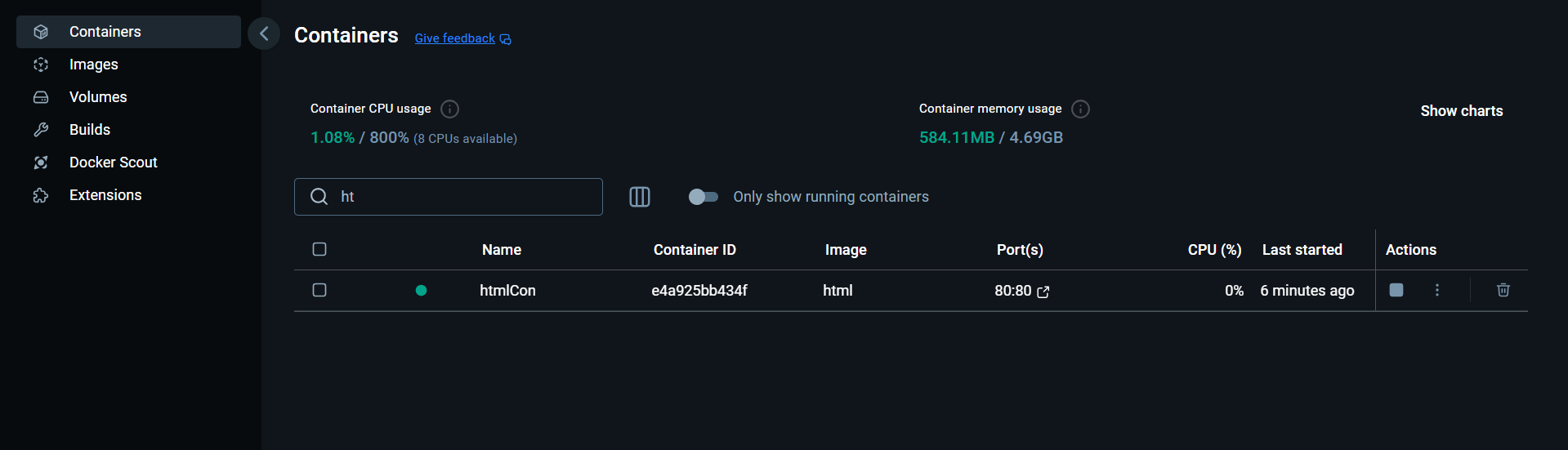


Рисунок 34 – Страница контейнеров

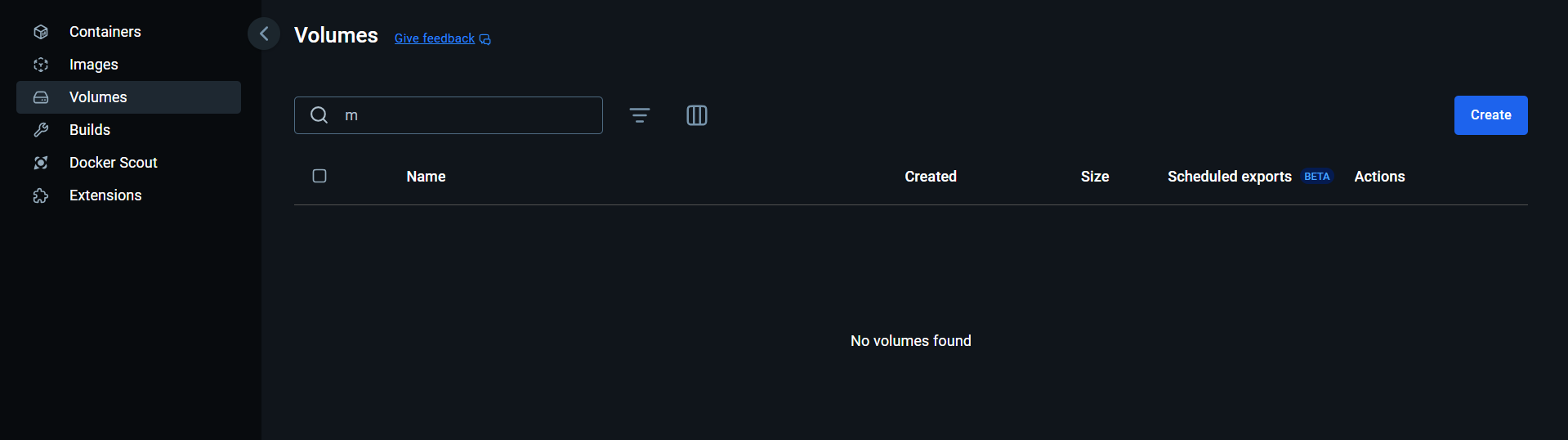


Рисунок 35 – Страница данных которые хранит контейнер локально

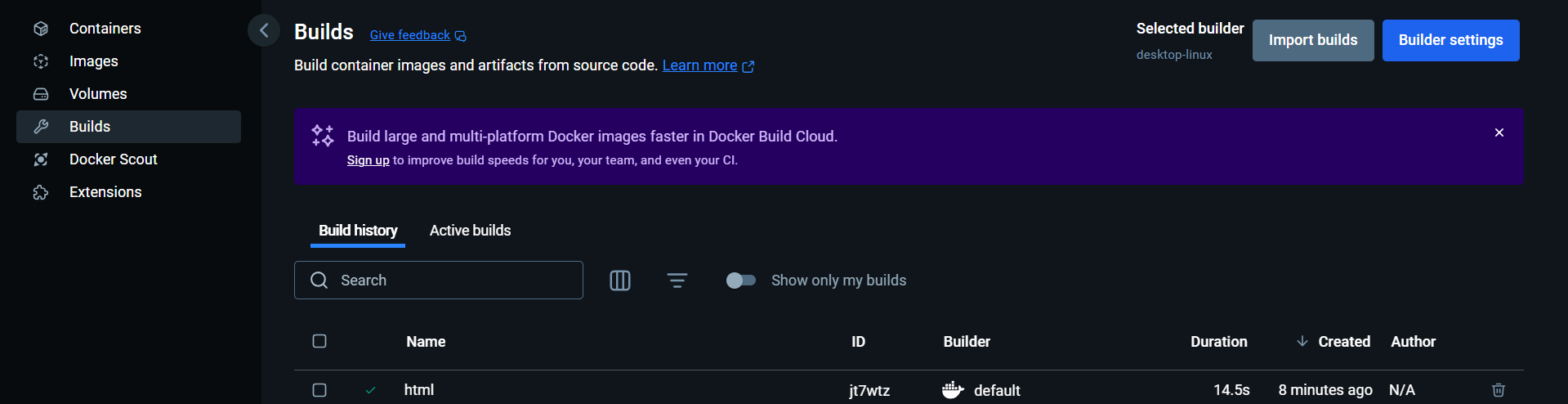


Рисунок 36 – Страница показывает все попытки собрать образ

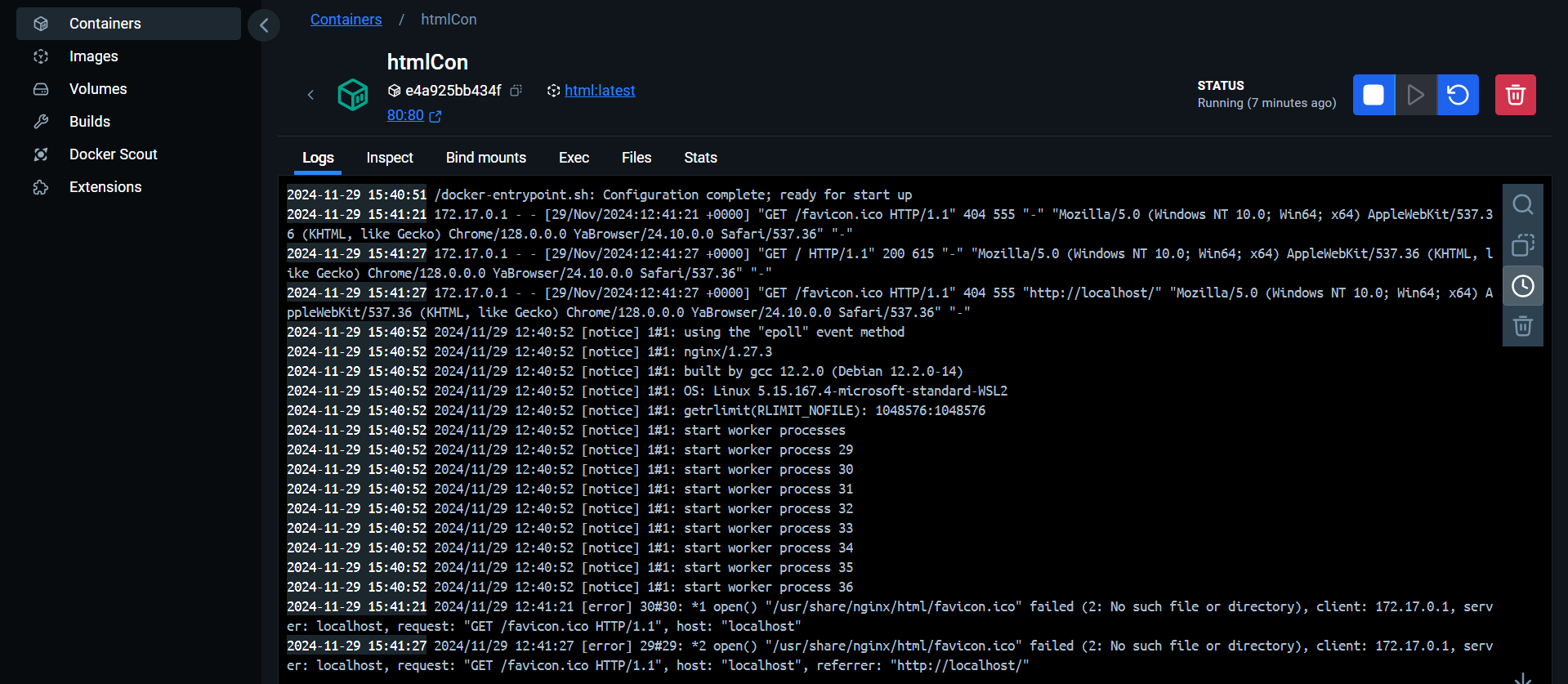


Рисунок 37 – Страница запущенного образа

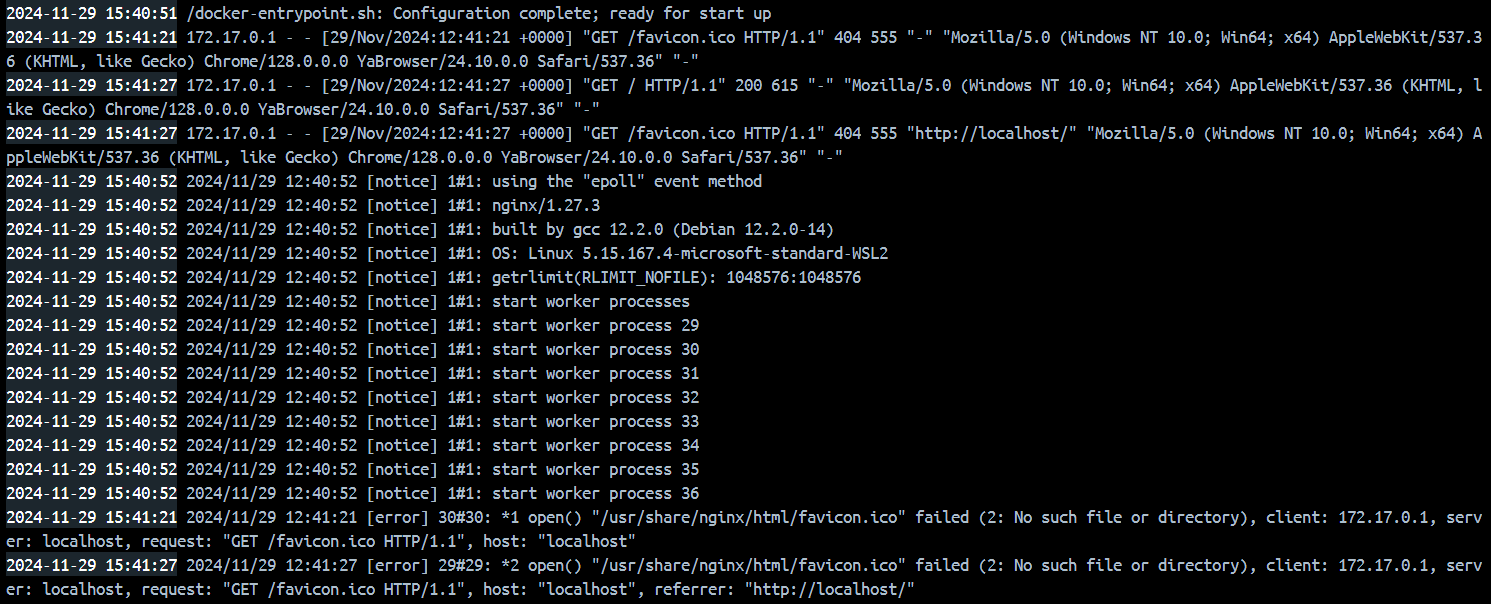


Рисунок 38 – Логи контейнера

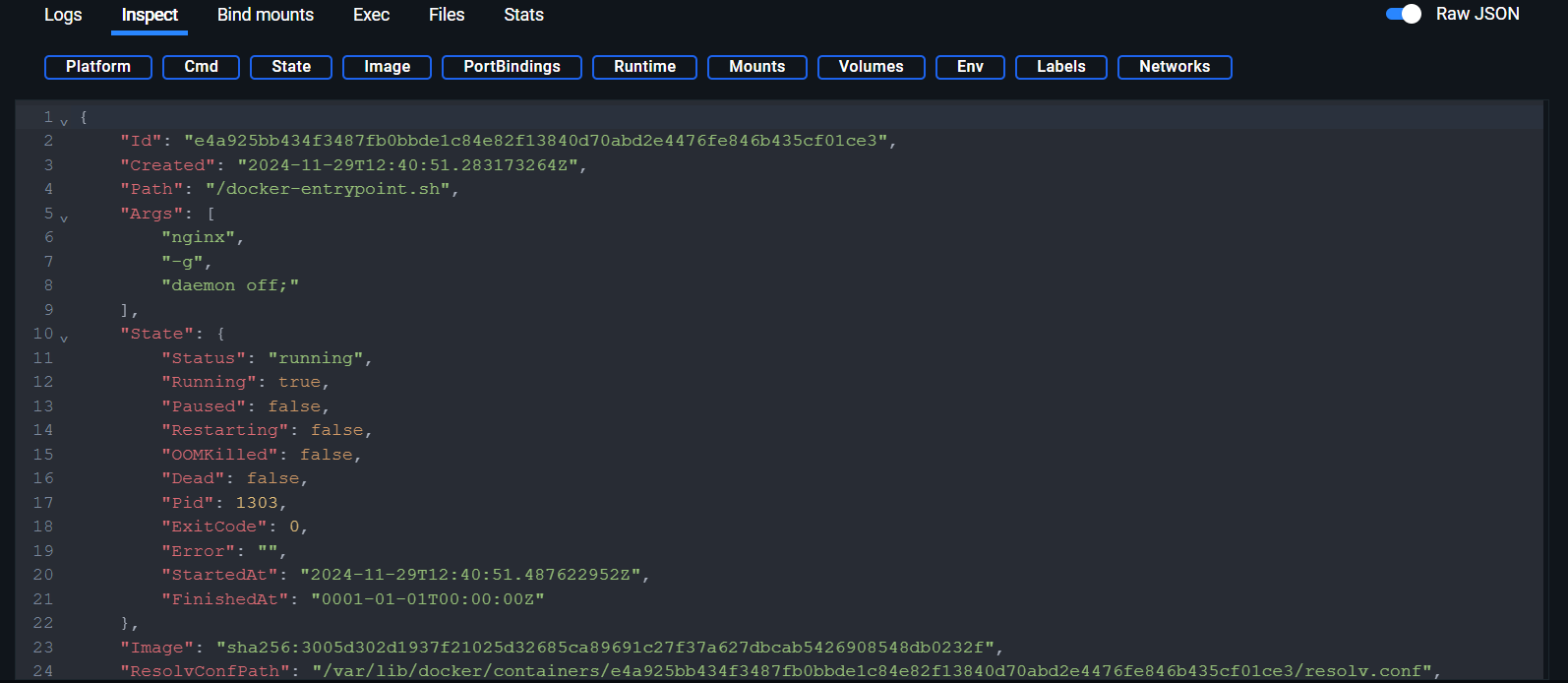


Рисунок 39 – Детальная информация о контейнере

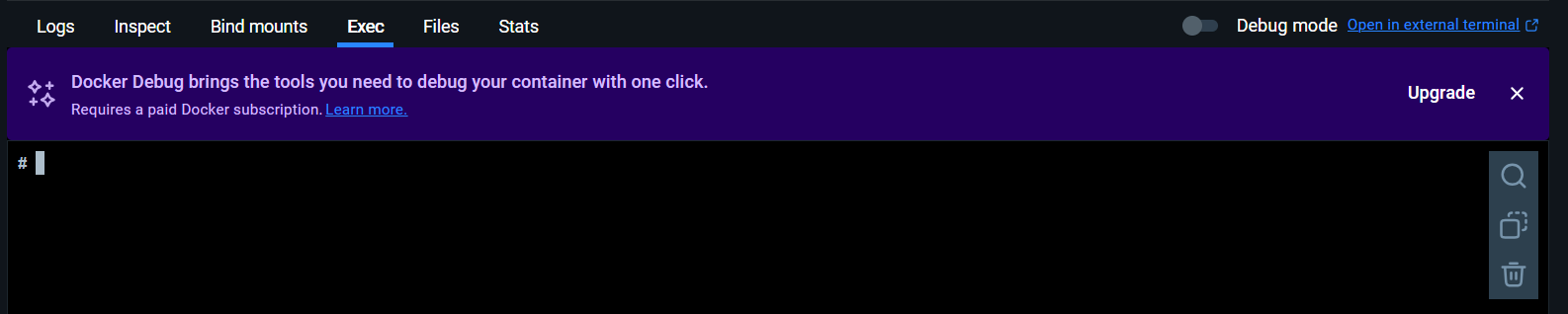


Рисунок 40 – Консоль контейнера

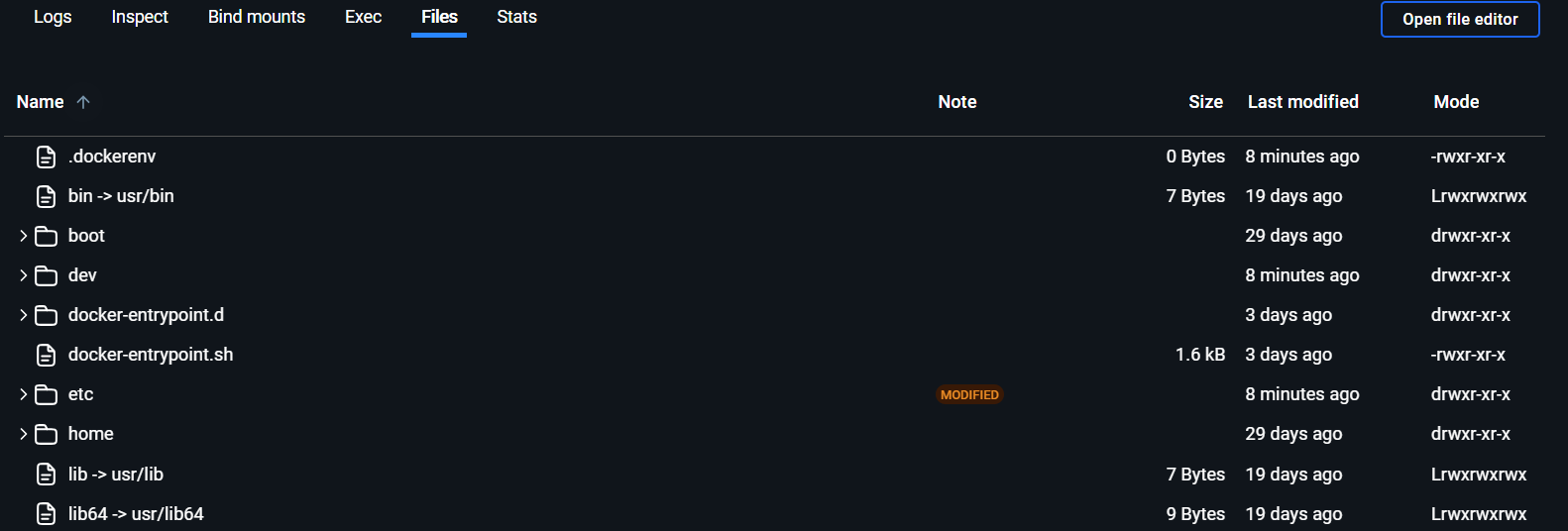


Рисунок 41 – Файловая система контейнера



Рисунок 42 – Статистика контейнера

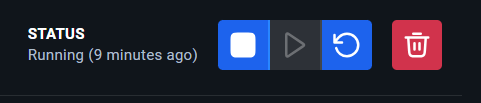


Рисунок 43 – Кнопки запуска, остановки, перезапуска, удаления контейнера

## Еще больше образов

* calc-java

Создаем образ

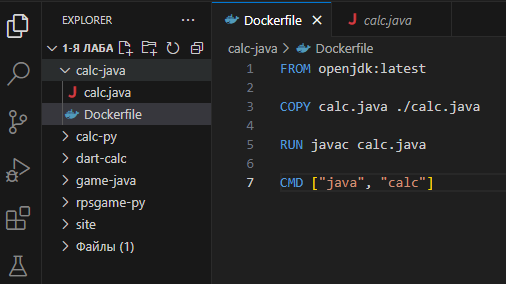


Рисунок 44 – java образ

Собираем образ

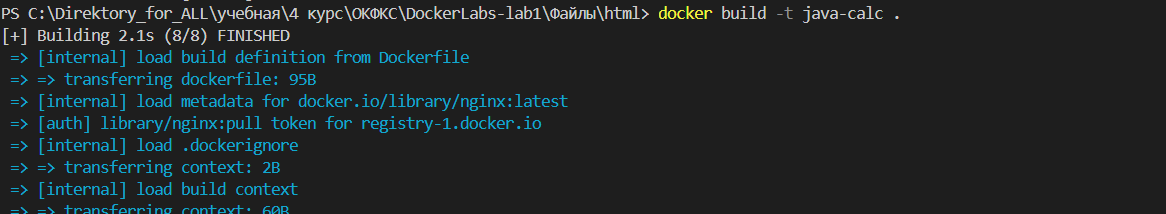


Рисунок 45 – Собираем образ

Запускаем контейнер

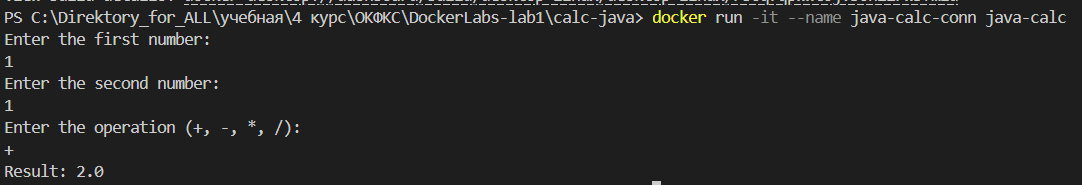


Рисунок 46 – Запускаем контейнер

* calc-py

Создаем образ

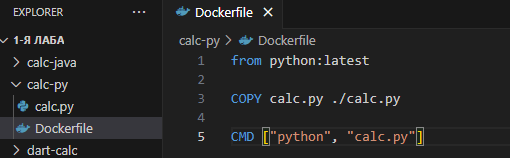


Рисунок 47 – Образ

Собираем образ

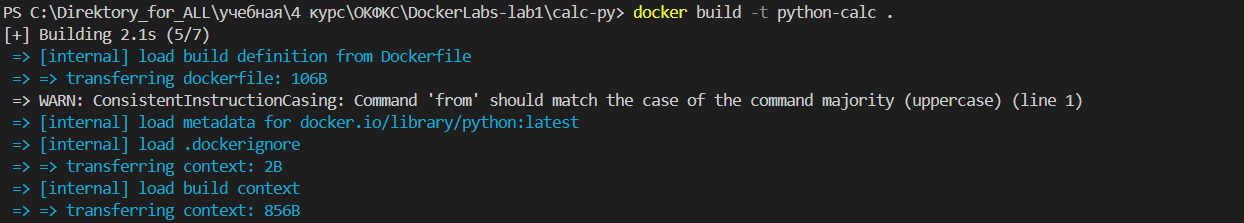


Рисунок 48 – Собираем образ

Запускам образ в контейнере

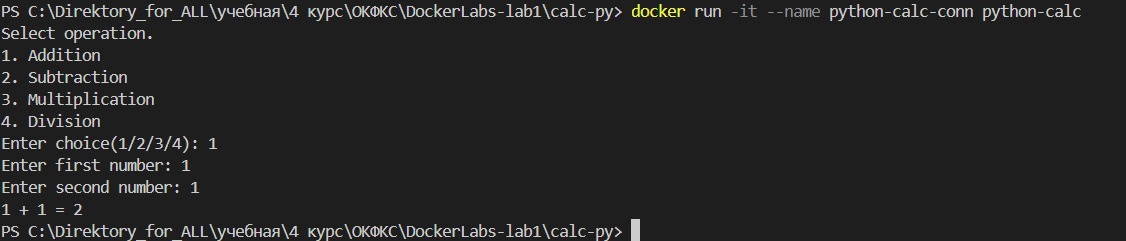


Рисунок 49 – Запуск образа

* dart-calc

Пишем образ

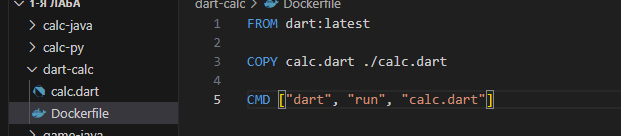


Рисунок 50 – Образ dart

Собираем образ

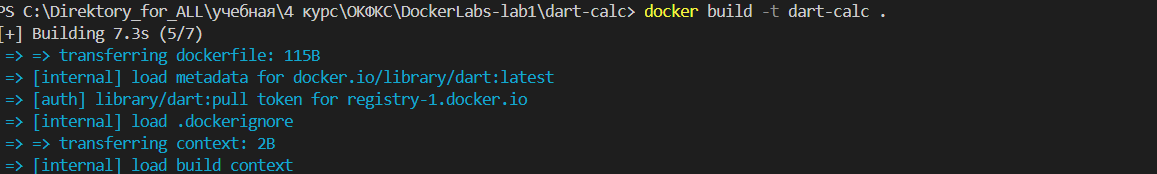


Рисунок 51 – Собираем образ

Запускаем контейнер

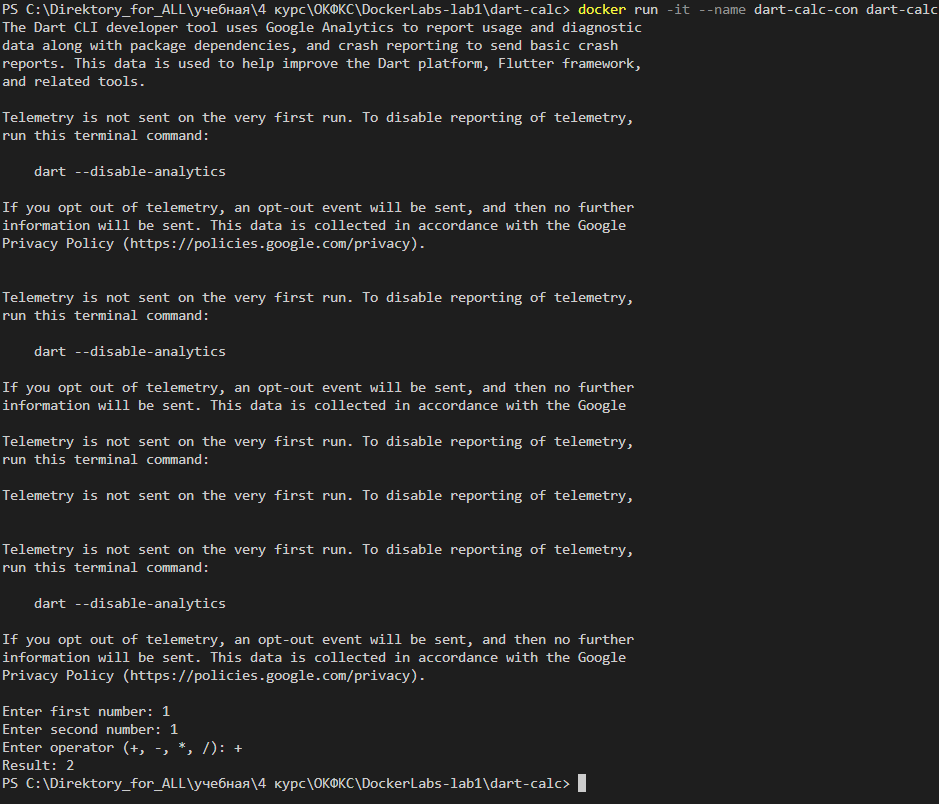


Рисунок 52 – Запуск контейнера

* game-java

Пишем образ

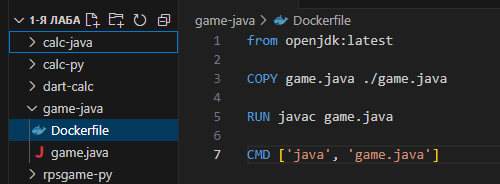


Рисунок 53 – Образ

Собираем образ

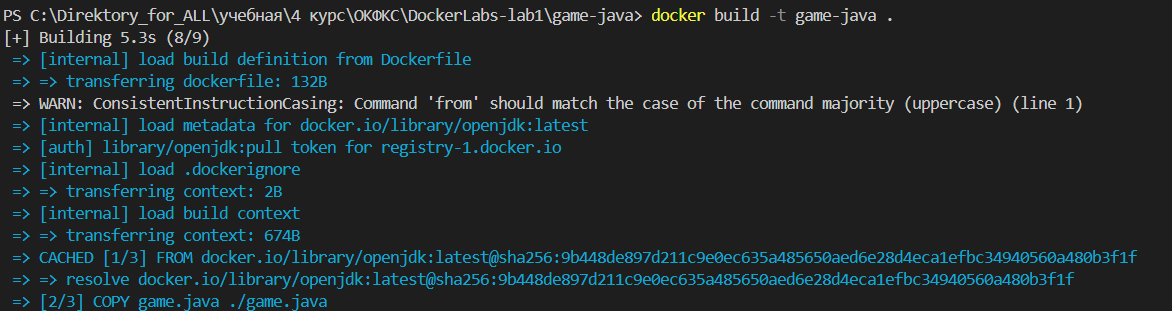


Рисунок 54 – Собираем образ

Запускаем контейнер

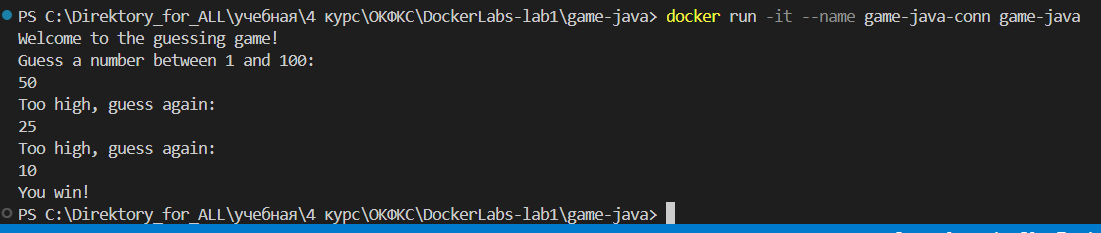


Рисунок 55 – Запуск контейненра

* rpsgame-py

Пишем образ

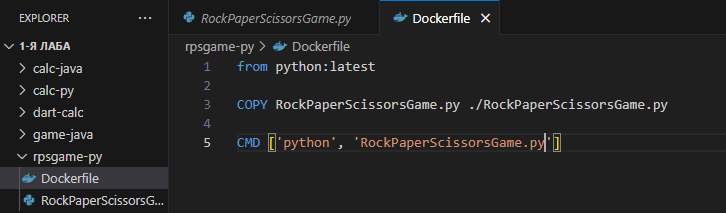


Рисунок 56 – Пишем образ

Собираем образ

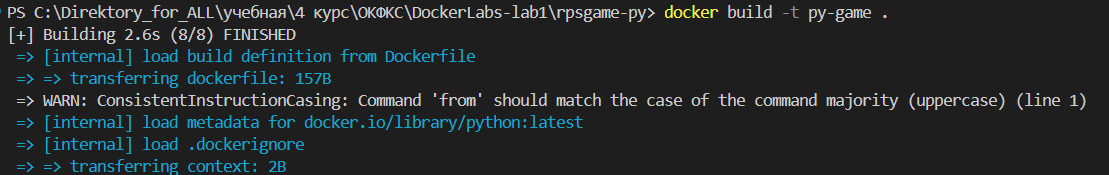


Рисунок 57 – Собранный образ

Запускаем контейнер

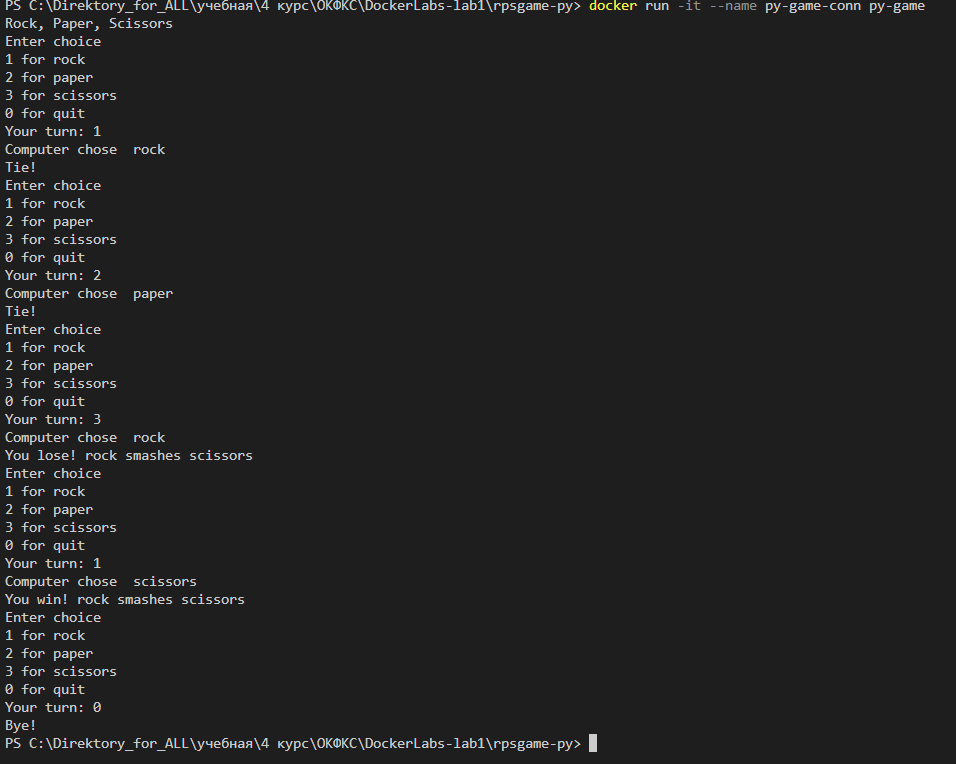


Рисунок 58 – Запуск контейнера

* site

Пишем образ

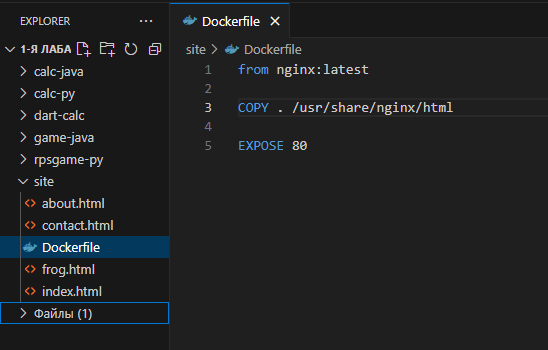


Рисунок 59 – Образ

Собираем образ

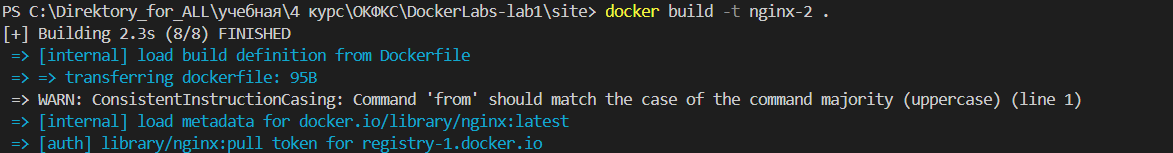


Рисунок 60 – Собранный образ

Запускаем контейнер

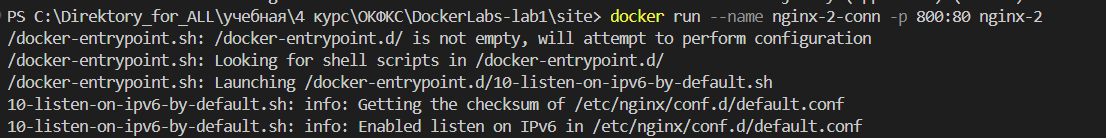


Рисунок 61 – Запуск контейнера

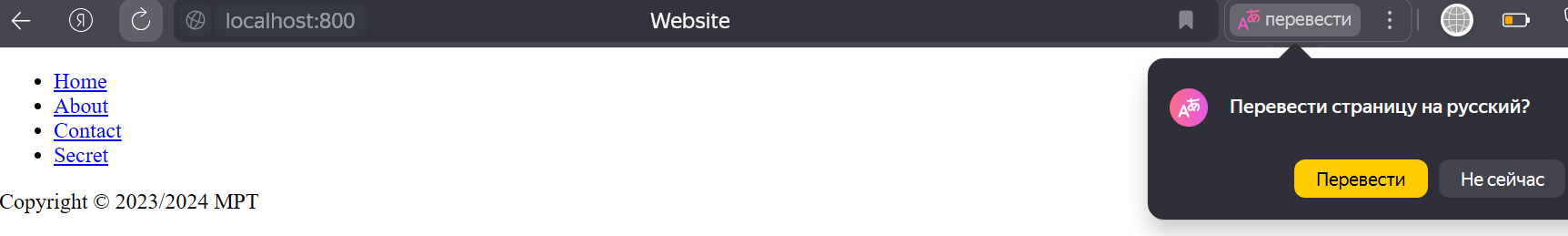


Рисунок 62 – index.html



Рисунок 63 - contact.html

Вывод: В результате выполненной работы были выполненны: сборка и развертывание контейнеров из примера, а так же описан процесс. Описаны вкладки в Docker (включая вкладки внутри собранного образа и контейнера). Написаны Doсkerfile и собраны контейнеры с программами из архива.

# Практическая №4

Цель: с использованием докер нужно изучить и повторить возможность развертывания связанного веб приложения с возможностью использования данного приложения вне контейнера

Посещение сайта

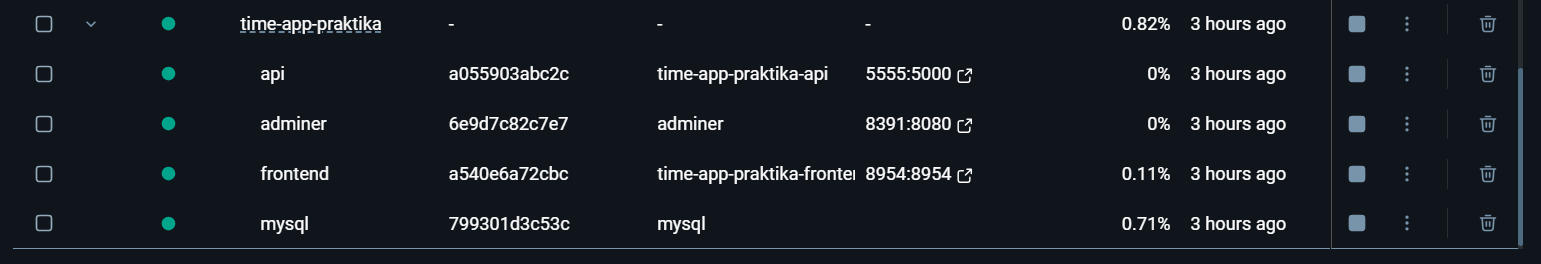


Рисунок - Посещение сайта

Докер файл для апи

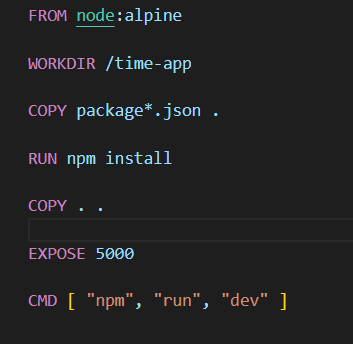


Рисунок - Докер файл для апи

Пример файла докер компосе

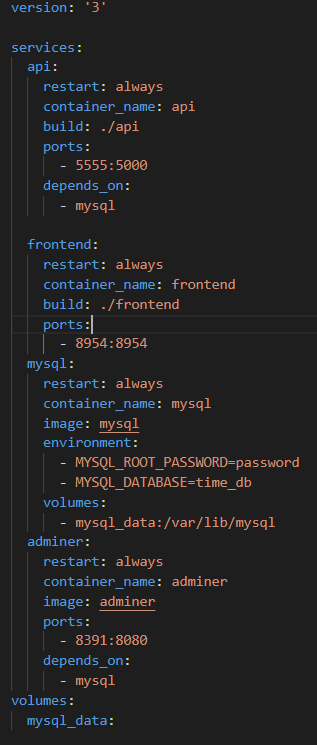


Рисунок - Пример файла докер компосе

Пример еще одного докер файла

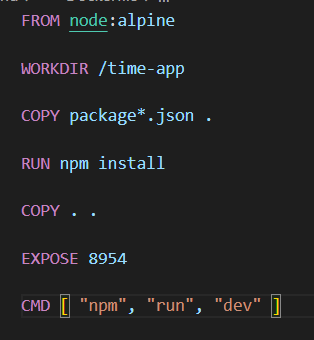


Рисунок - Пример еще одного докер файла

Пример работающего сайта

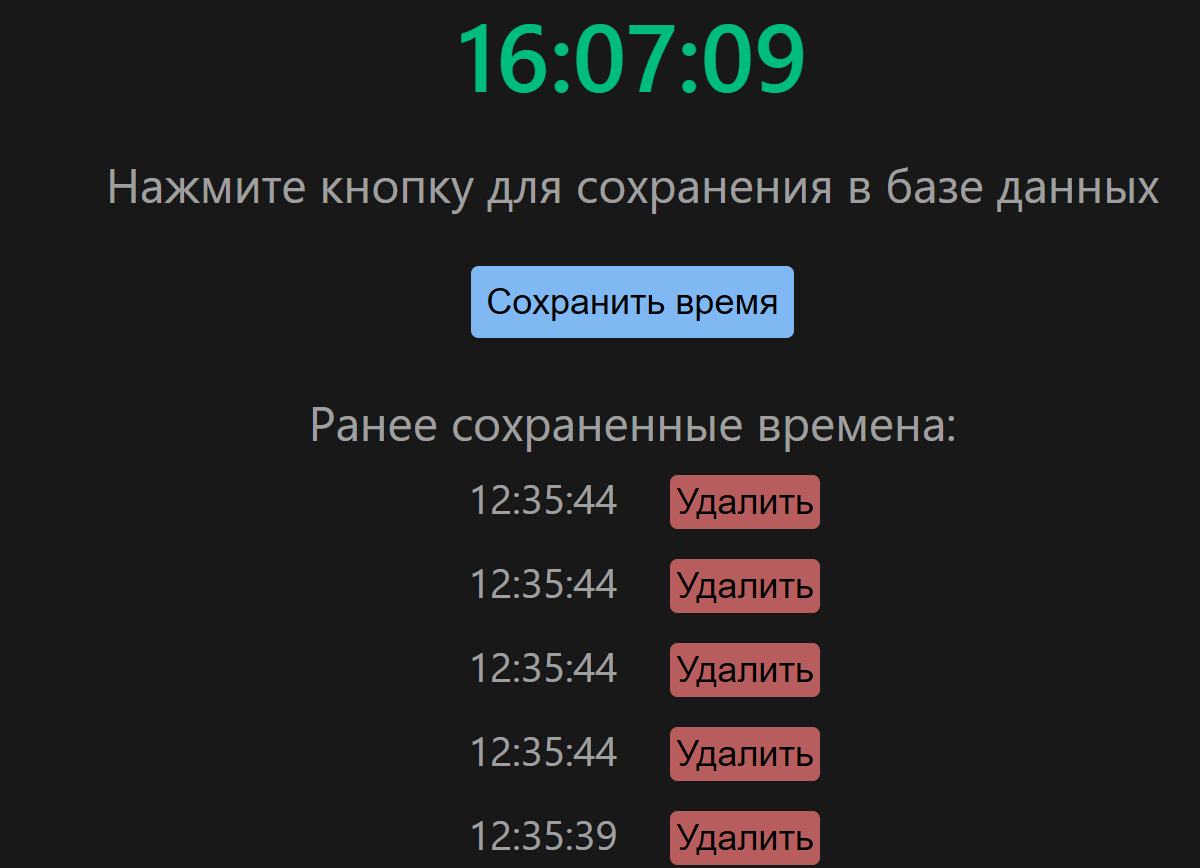


Рисунок - Пример работающего сайта

Вывод: В результате выполненной работы были, с использованием докер развернуты и повторена возможность развертывания связанного веб приложения с возможностью использования данного приложения вне контейнера