|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |  |
|  | |  |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |  |
|  |  | |
|  |  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4** | |
| **по дисциплине** | |
| **«**Технология разработки программных приложений**»**  **Тема: «Docker»** | |
|  | |
| Выполнил студент группы ИКБО-33-22 | Шило Ю.С. |
| Принял преподаватель | Петрова А.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

**ЦЕЛЬ РАБОТЫ**

Знакомство с Docker.

**ЧАСТЬ 1 – ОБРАЗЫ**

*Задание 1*

Посмотрите на имеющиеся образы: docker images

*Решение*

Для выполнения задания была использована команда docker images данная команда показывает установленные образцы. Результат выполнения команды представлен на рисунке 1.1.



Рисунок 1.1 – Результат выполнение задания

*Задание 2*

Загрузите образ: docker pull ubuntu

*Решение*

Для выполнения данного задания была использована команда docker pull ubuntu, данная команда загружает последний на данный момент образ Ubuntu. Результат выполнения команды представлен на рисунке 1.2.

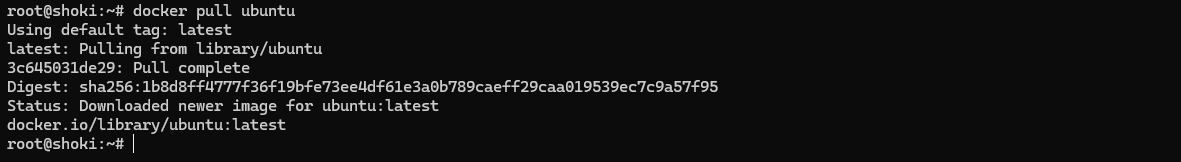


Рисунок 1.2 – Результат выполнения задания

*Задание 3*

Посмотрите на имеющиеся образы ещё раз: docker images — должны появиться новые загруженные образы. Посмотрите список контейнеров, выполнив команду: docker ps -a

*Решение*

Для просмотра текущих установленных образов была использована команда docker images. Для просмотра списка контейнеров была использована команда docker ps -a. Результат выполнения представлен на рисунке 1.3.



Рисунок 1.3 – Результат выполнения задания

**ЧАСТЬ 2 – ИЗОЛЯЦИЯ**

*Задание 4*

Посмотрим информацию о хостовой системе, выполнив команду hostname. Выполните её ещё один раз.

*Решение*

Для выполнения данного задания была дважды использована команда hostname. Она возвращает текущее имя хоста. Результат выполнения представлен на рисунке 2.1.



Рисунок 2.1 – Результат выполнения задания

*ВОПРОС*

1. Одинаковый ли результат получился при разных запусках?

Если между запусками в конфигурации системы ничего не изменилось, то команда hostname будет выводить одинаковое название хоста

*Задание 5*

Попробуем выполнить то же самое в контейнерах. Выполните два раза команду docker run ubuntu hostname

*Решение*

Для выполнения данного задания была дважды использована команда docker run ubuntu hostname. Результат выполнения команд представлен на рисунке 2.2.



Рисунок 2.2 – Результат выполнения задания

*ВОПРОС*

1. Одинаковый ли результат получился при разных запусках?

Результат будет разный, т. к. у нас запускается новый контейнер с уникальными состоянием, отличных от других. Хостнейм нового контейнера будет отличаться от предыдущего

*Задание 6*

Заново выполните docker ps -a - там должны появиться запущенные ранее контейнеры.

*Решение*

Для выполнения данного задания была выполнена команда docker ps -a. Команда показала два запущенных контейнера из предыдущего задания. Результат выполнения команды представлен на рисунке 2.3.

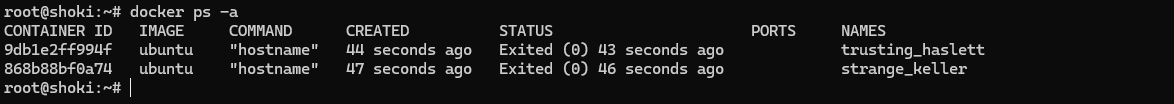


Рисунок 2.3 – Результат выполнения задания

**ЧАСТЬ 3 – РАБОТА С ПОРТАМИ**

*Задание 7*

Для начала, загрузите образ python командой docker pull python.

*Решение*

Для выполнения данного задания была выполнена команда docker pull python. Результат выполнения команды представлен на рисунке 3.1



Рисунок 3.1 – Результат выполнения задания

*Задание 8*

Запустите встроенный в Python модуль веб-сервера из корня контейнера, чтобы отобразить содержание контейнера. Пробросить порт 8000. Завершить работу веб-сервера.

*Решение*

Для выполнения данного задания была использована команда docker run -it -p8000:8000 python python -m http.server. Если просто запустить docker run -it python python -m http.server, то доступа к веб-серверу не будет, поскольку порт 8000 не был проброшен. Результат выполнения команды представлен на рисунке 3.2.

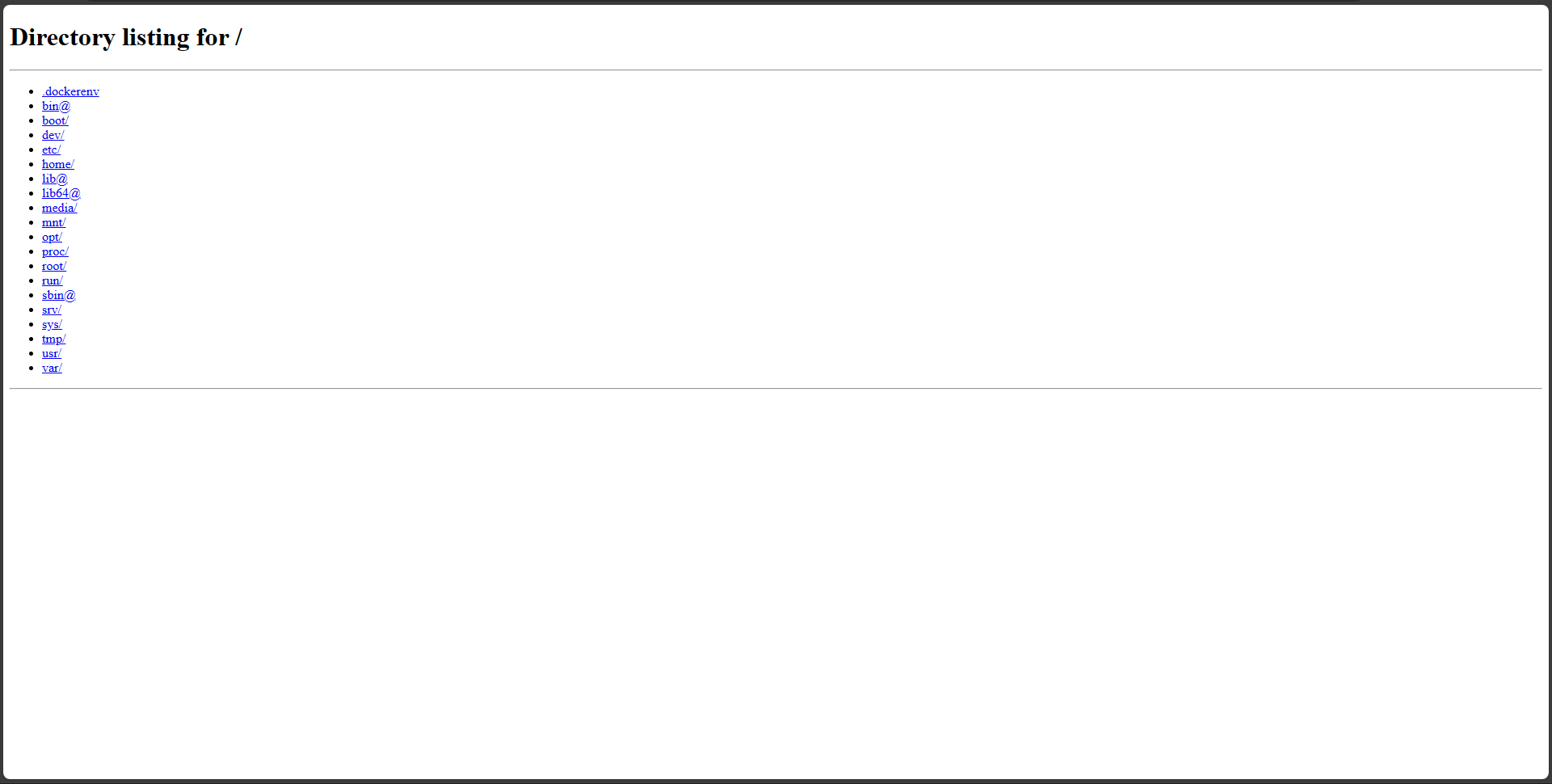


Рисунок 3.2 – Результат выполнения задания

Вывод в консоли после выполнения команды представлен на рисунке 3.3.

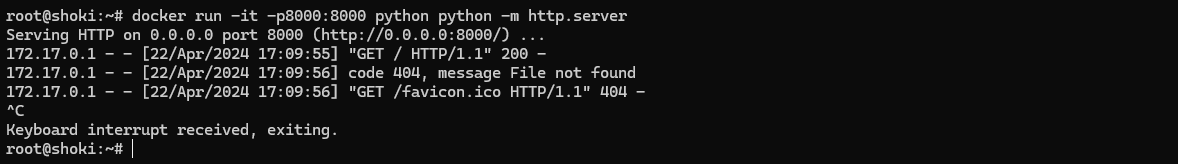


Рисунок 3.3 – Результат выполнения задания

**ЧАСТЬ 4 – ИМЕНОВАННЫЕ КОНТЕЙНЕРЫ, ОСТАНОВКА И УДАЛЕНИЕ**

*Задание 9*

Запустить контейнер с именем pyserver. Убедиться в том, что он запущен.

*Решение*

Для выполнения данного задания была использована команда docker run -p8000:8000 --name pyserver -d python python -m http.server. Флаг — name позволяет дать запущенному контейнеру имя. Чтобы проверить что контейнер запущен была выполнена команда docker ps | grep pyserver. Результат выполнения команд представлен на рисунке 4.1.

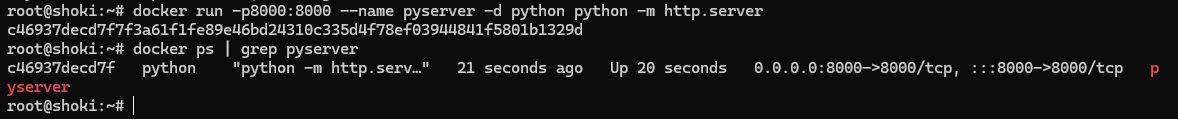


Рисунок4.1 — Результат выполнения задания

*Задание 10*

Остановить и удалить контейнер с именем pyserver.

*Решение*

Для выполнения данного задания была использована команда docker stop pyserver, а затем docker rm pyserver. Так же можно удалить все одной командой: docker rm -f pyserver. Для того, чтобы контейнер сам удалялся после остановки можно установить при запуске флаг —rm. Результат остановки и удаления контейнера представлен на рисунке 4.2.

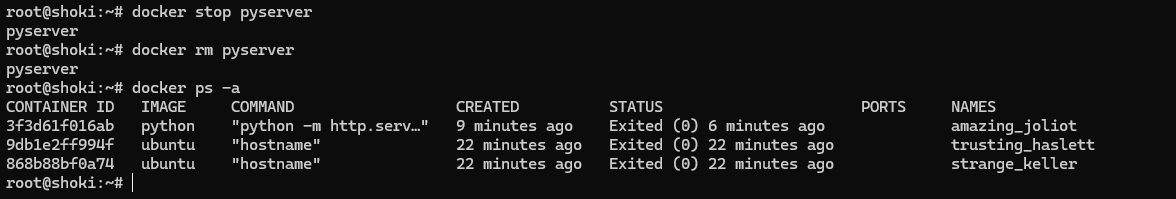


Рисунок4.2 — Результат выполнения задания

**ЧАСТЬ 5 – ПОСТОЯННОЕ ХРАНЕНИЕ ДАННЫХ**

*Задание 11*

Запустите контейнер, в котором веб-сервер будет отдавать содержимое директории /mnt.

*Решение*

Для выполнения данного задания была использована команда docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt. Результат выполнения предоставлен на рисунке 5.1

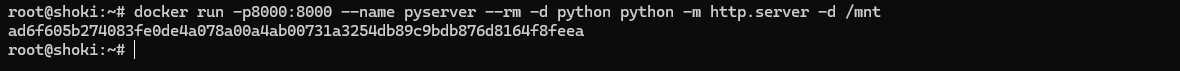


Рисунок 5.1 – Результат выполнения задания

*ВОПРОС*

3.

Что значат остальные флаги запуска? Где здесь команда, которая выполнится в контейнере?

-p 8000:8000 — пробрасывает порт 8000, чтобы к контейнеру можно было по нему обращаться. --name pyserver — указывает имя «pyserver» для контейнера, --rm — удаляет контейнер после завершения работы, -d — запускает контейнер в фоновом режиме, python python -m http.server -d /mnt — команда, которая будет выполняться в контейнере.

*Задание 12*

Попасть в контейнер pyserver и выполнить cd mnt && echo "hello world" > hi.txt. Затем проверить наличие файл hi.txt на <http://0.0.0.0:8000/>.

*Решение*

Для выполнения данного задания была использована команда docker exec -it pyserver bash. Далее в оболочке bash была выполнена команда cd mnt && echo "hello world" > hi.txt. Результат выполнения команд представлен на рисунке 5.2.

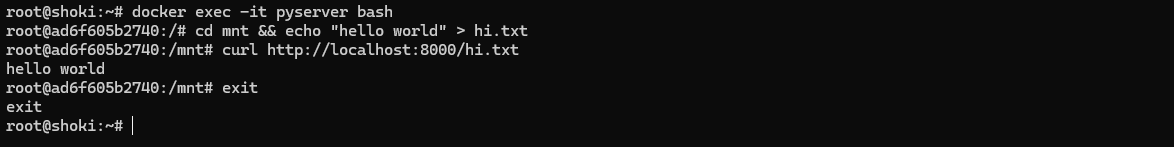


Рисунок 5.2 – Результат выполнения задания

*Задание 13*

Остановить контейнер и проверить наличие файла.

*Решение*

Для выполнения данного задания была использована команда docket stop pyserver. Заходим в оболочку docker exec -it pyserver bash. Далее проверяем наличие файла в директории /mnt. Результат выполнения команд представлен на рисунке 5.3.

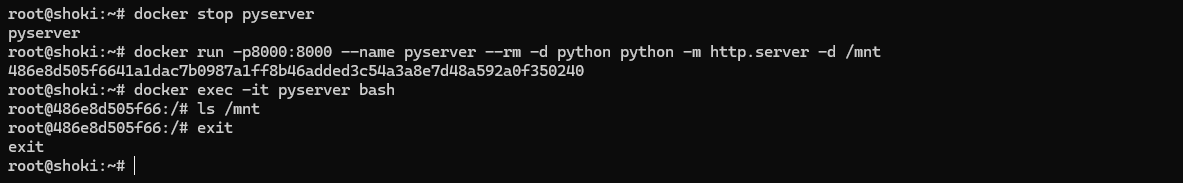


Рисунок 5.3 – Результат выполнения задания

**ЧАСТЬ 5.1 – ТОМА**

*Задание 14*

Создать контейнер, но уже с примонтированным томом. Создать внутри контейнера файл, затем удалить контейнер. Узнать, где хранятся данные после удаления контейнера.

*Решение*

Для создания контейнера с примонтированным томом была использована команда: «docker run -p 8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python python -m http.server -d /mnt». Для создания внутри контейнера файла использовались команды: «docker exec -it pyserver bash», «cd mnt», «echo "hello world" > hi.txt». Для того, чтобы узнать куда отправляются файлы после удаления контейнера была использована команда: «docker inspect -f "{{json .Mounts }}" pyserver». Результат выполнения команд представлен на рисунке 5.4.

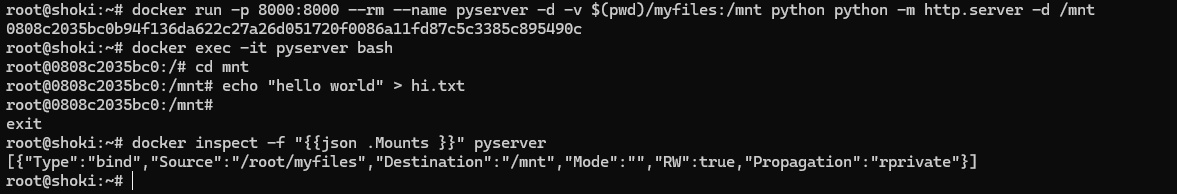


Рисунок 5.4 – Результат выполнения задания

**ЧАСТЬ 5.2 – МОНТИРОВАНИЕ ДИРЕКТОРИЙ И ФАЙЛОВ**

*Задание 15*

Создать директорию и примонтировать её к контейнеру. В контейнере создать файл и проверить его наличие на хостовой машине. Остановить контейнер.

*Решение*

Для создания контейнера с примонтированной директорий использовалась команда: «docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python \ python -m http.server -d /mnt». Результаты выполнения команды представлены на рисунке 5.5.

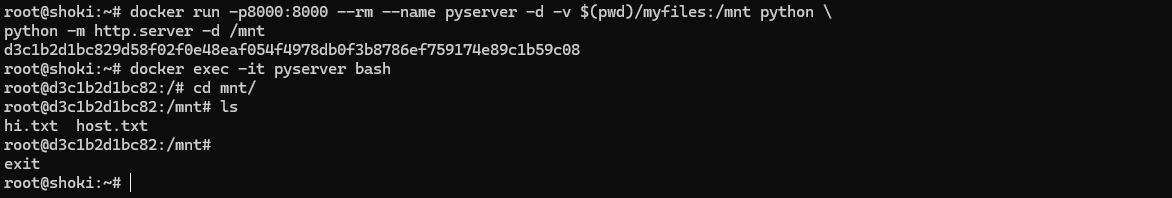


Рисунок 5.5 – Результат выполнения задания

**ЧАСТЬ 6 – ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ**

*Задание 16*

Создать контейнер с переменной окружения MIREA=«One love». Проверить переменные окружения.

*Решение*

Для создания контейнера с переменной окружения и проверкой переменных окружения использовалась команда: «docker run -it --rm -e MIREA="ONE LOVE" ubuntu env». Результат выполнения команды представлен на рисунке 6.1.

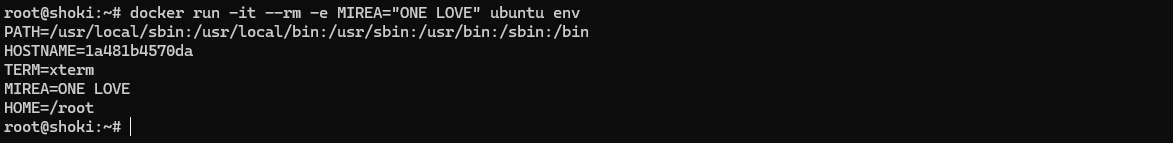


Рисунок 6.1 – Результат выполнения задания

**ЧАСТЬ 7 – DOCKERFILE**

*Задание 17*

Создать Dockerfile и внести в него необходимые из задания изменения. Собрать образ. Запустить и проверить корректность работы.

*Решение*

Был создан Dockerfile, который был изменен согласно заданию. Содержание Dockerfile представлено на рисунке 7.1.

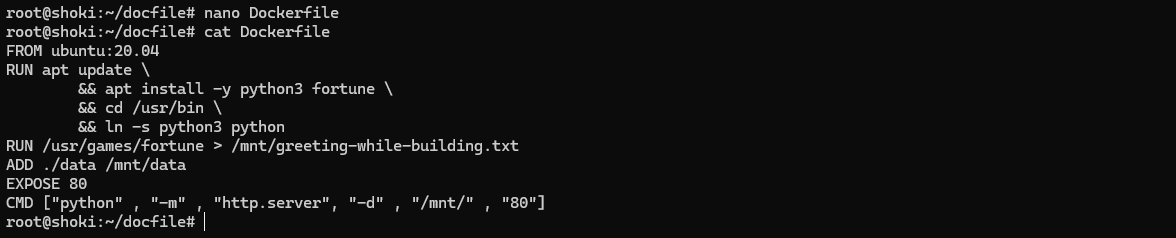


Рисунок 7.1 – Содержимое файла Dockerfile

Для создания образа была использована команда «docker build -t mycoolimage .». Результат выполнения представлен на рисунке 7.2.

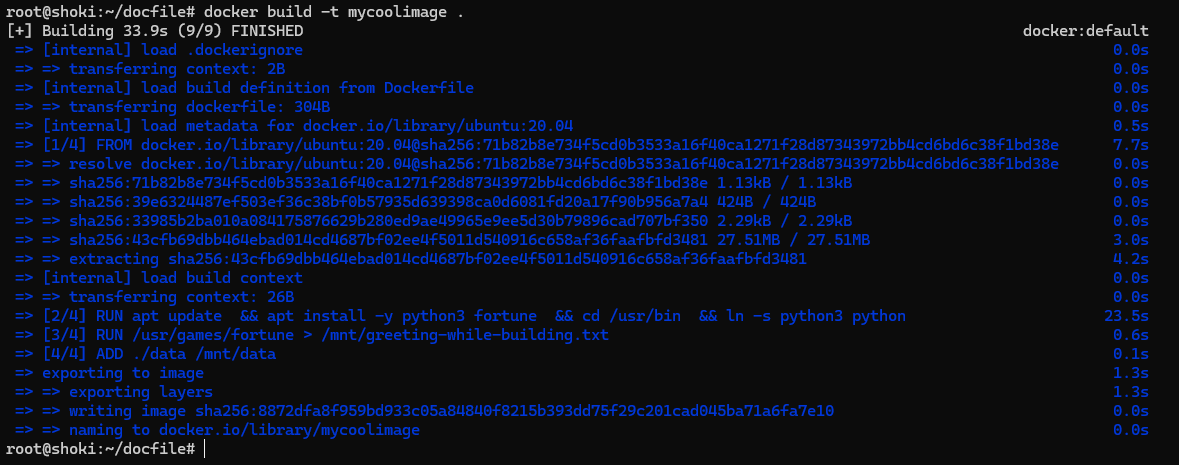


Рисунок 7.2 – Результат выполнения команды

Для запуска контейнера по собранному образу использовалась команда: «docker run --rm -it -p8099:80 mycoolimage». Результат проверки работоспособности представлен на рисунке 7.3.



Рисунок 7.3 – Результат проверки

**ЧАСТЬ 8 - ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ**

*Вариант 29*

*Задание*

Написать Dockerfile, собрать образ, запустить контейнер (и записать команду для его запуска).

*Решение*

Был создан Dockerfile, который был изменен согласно индивидуальному заданию (29 вариант). Содержание Dockerfile представлено на рисунке 8.1.

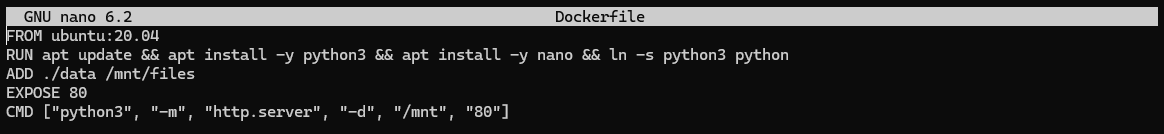


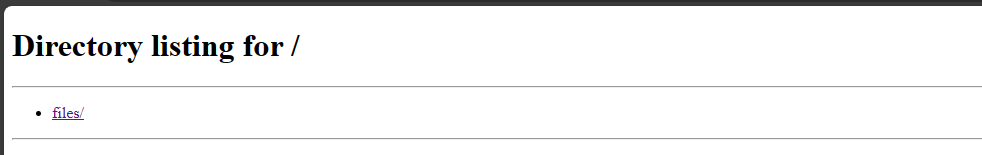
Рисунок 8.1 – Содержимое файла Dockerfile

Для создания образа была использована команда «docker build -t ind .». Результат выполнения представлен на рисунке 8.2.



Рисунок 8.2 – Результат выполнения команды

Для запуска контейнера использовалась команда «docker run --rm -v "$(pwd)/data:/mnt/files" -p 8829:80 ind». Результат выполнения предоставлен на рисунках 8.3.





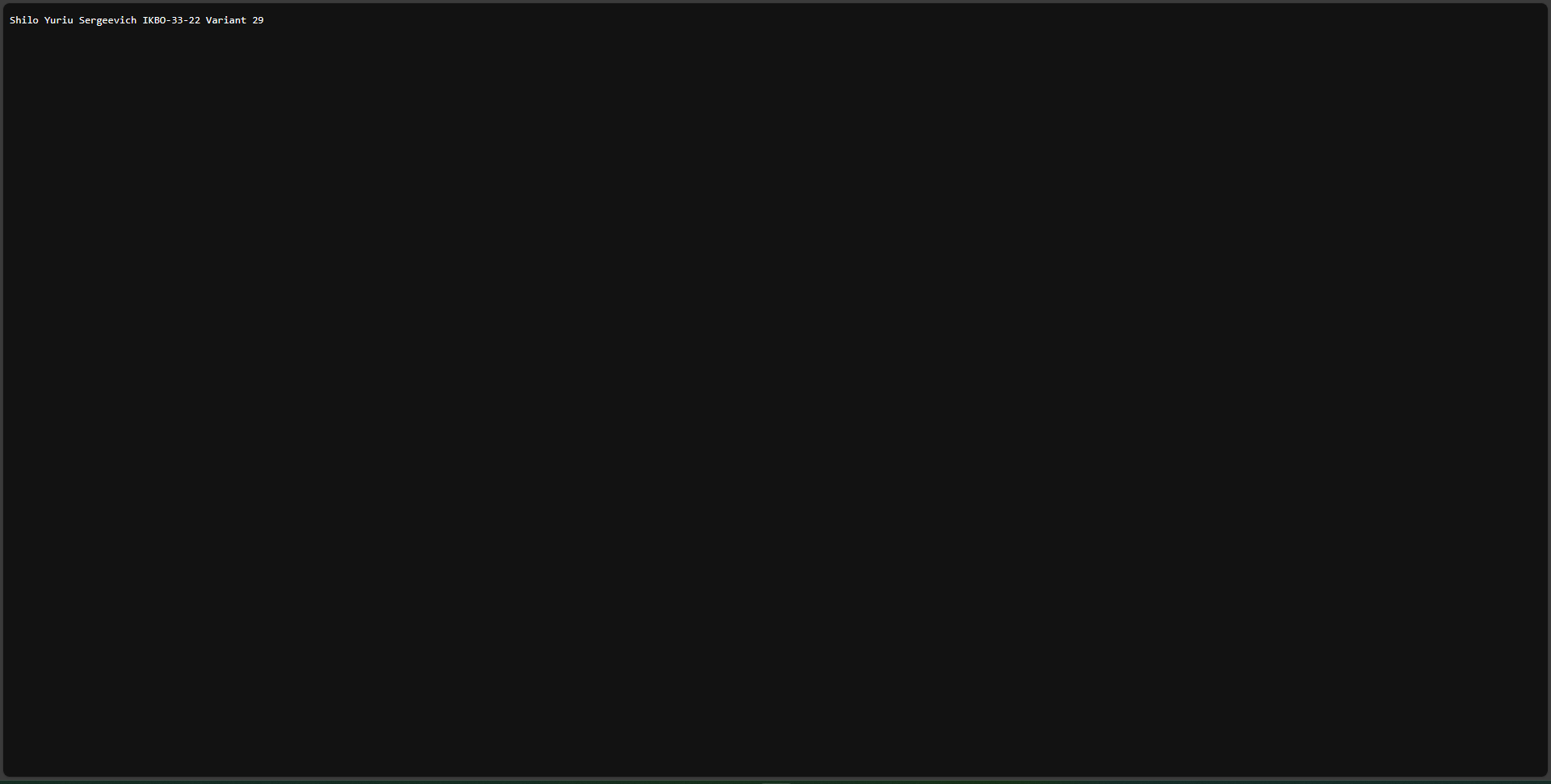


Рисунок 8.3 – Результат выполнения задания

**ВЫВОД**

В ходе данной практической работы были получены базовые навыки по работе с Docker.