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 «Технологии разработки программных приложений»

Тема практической работы: «Docker»

Выполнил:
Студент группы ИКБО-20-23 Комисарик М.А.

Проверил: Доцент кафедры МОСИТ,
кандидат технических наук, доцент
Чернов Е.А.

Москва 2025

СОДЕРЖАНИЕ

1 ВЫПОЛНЕНИЕ ЗАДАНИЯ

1.1 Образы

Посмотрите на имеющиеся образы: **docker images**.

Загрузите образ: **docker pull ubuntu** будет загружен образ ubuntu:latest последняя доступная версия.

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

```
PS C:\Users\krezo> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS C:\Users\krezo> docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
2726e237d1a3: Pull complete
Digest: sha256:1e622c5f073b4f6bfad6632f2616c7f59ef256e96fe78bf6a595d1dc4376ac02
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
PS C:\Users\krezo> docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    1e622c5f073b   5 days ago   117MB
PS C:\Users\krezo> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Рисунок 1 – Выполнение первого задания

1.2 Изоляция

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

```
PS C:\Users\krezo> hostname
Krezon
PS C:\Users\krezo> hostname
Krezon
```

Рисунок 2 – Выполнение команды hostname

Вопрос: одинаковый ли результат получился при разных запусках?

Да, так как пользователь не менялся между ними.

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

```
PS C:\Users\krezo> docker run ubuntu hostname
09db87bc610d
PS C:\Users\krezo> docker run ubuntu hostname
30532b909c27
```

Рисунок 3 – Выполнение команды hostname внутри контейнера

Вопрос: Одинаковый ли результат получился при разных запусках?

Нет, так как «пользователь» контейнера каждый раз уникален.

В случае запуска команды в контейнерах, ответ будет немного отличаться, будет разный hostname.

Так происходит, потому что из одного образа ubuntu были запущены два изолированных контейнера, поэтому у них и был разный hostname.

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

```
PS C:\Users\krezo> docker ps -a
CONTAINER ID   IMAGE     COMMAND      CREATED              STATUS                          PORTS     NAMES
30532b909c27   ubuntu    "hostname"   About a minute ago   Exited (0) About a minute ago             sleepy_chaum
09db87bc610d   ubuntu    "hostname"   About a minute ago   Exited (0) About a minute ago             beautiful_jones
6cba42ef7c30   ubuntu    "hostname"   2 minutes ago        Exited (0) 2 minutes ago                  clever_mcnulty
```

Рисунок 4 – Результат выполнения команды docker ps -a

Запуск контейнеров производится командой:

docker run --флаги-докера имя_контейнера команда-для-запуска --флаги-запуска-программы.

Запустите bash в контейнере: **docker run ubuntu bash**.

```
PS C:\Users\krezo> docker run ubuntu bash
PS C:\Users\krezo>
```

Рисунок 5 – Запуск команды docker run ubuntu bash

Ничего не произошло. Это не баг. Интерактивные оболочки выйдут после выполнения любых скриптовых команд, если только они не будут запущены в интерактивном терминале поэтому для того, чтобы этот пример не завершился,

вам нужно добавить флаги **-i -t** или сгруппированно **-it**: **docker run-it ubuntu bash**.

```
PS C:\Users\krezo> docker run -it ubuntu bash
root@5df830791b6c:/# ^C
root@5df830791b6c:/#
exit
```

Рисунок 6 – Запуск команды docker run -it ubuntu bash

Выполняя запуск контейнера, указывая образ ubuntu, неявно указывался образ ubuntu:latest. Следовательно, следующие команды равнозначны:

- docker run ubuntu hostname
- docker run ubuntu:latest hostname

Если бы мы хотели запустить ubuntu:12.04, то нужно было бы выполнить команду docker run ubuntu:12.04 hostname

1.3 Работа с портами

Для начала, загрузите образ python командой **docker pull python**.

В качестве примера, запустите встроенный в Python модуль веб-сервера из корня контейнера, чтобы отобразить содержание контейнера. **docker run -it python python -m http.server**

При запуске пишется, что сервер доступен по адресу http://0.0.0.0:8000/. Однако, если открыть этот адрес, то ничего не будет видно, потому что порты не проброшены. Завершите работу веб-сервера, нажав комбинацию клавиш Ctrl+C.

```
PS C:\Users\krezo> docker pull python
Using default tag: latest
latest: Pulling from library/python
1eb98adba0eb: Pull complete
23b7d26ef1d2: Pull complete
b617a119f8a2: Pull complete
07d1b5af933d: Pull complete
171e1bee1949: Pull complete
739b86d2a778: Pull complete
e25cca11fd29: Pull complete
Digest: sha256:34dc8eb488136014caf530ec03a3a2403473a92d67a01a26256c365b5b2fc0d4
Status: Downloaded newer image for python:latest
docker.io/library/python:latest
PS C:\Users\krezo> docker run -it python python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
^C
Keyboard interrupt received, exiting.
```

Рисунок 7 – Запуск веб-сервера без проброса портов

Для проброса портов используется флаг **-p hostPort:containerPort**

Добавьте его, чтобы пробросить порт 8000:

docker run -it -p 8000:8000 python python -m http.server — теперь по адресу http://0.0.0.0:8000/ (если не открывается на Windows, то вместо 0.0.0.0 нужно указать localhost) открывается содержимое корневой директории в контейнере.

Для того, чтобы доступный в контейнере на порту 8000 веб-сайт в хостовой системе открывался на порту 8888, необходимо указать флаг -p 8888:8000:

docker run -it -p8888:8000 python python -m http.server. Завершите работу веб-сервера, нажав комбинацию клавиш Ctrl+C.

```
PS C:\Users\krezo> docker run -it -p 8000:8000 python python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.17.0.1 - - [20/Apr/2025 11:10:28] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [20/Apr/2025 11:10:28] code 404, message File not found
172.17.0.1 - - [20/Apr/2025 11:10:28] "GET /favicon.ico HTTP/1.1" 404 -
^C
Keyboard interrupt received, exiting.
PS C:\Users\krezo> docker run -it -p 8888:8000 python python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.17.0.1 - - [20/Apr/2025 11:36:21] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [20/Apr/2025 11:36:21] code 404, message File not found
172.17.0.1 - - [20/Apr/2025 11:36:21] "GET /favicon.ico HTTP/1.1" 404 -
^C
Keyboard interrupt received, exiting.
```

Рисунок 8 – Запуск веб-сервера с проброшенными портами

1.4 Именованные контейнеры, остановка и удаление

Запустите контейнер: **docker run -it -p8000:8000 python python -m http.server**. Нажмите Ctrl+C

— выполнение завершится. Для того, чтобы запустить контейнер в фоне, нужно добавить флаг **-d/--detach**. Также определим имя контейнеру, добавив флаг **--name**.

docker run -p8000:8000 --name pyserver -d python python -m http.server

Убедитесь, что контейнер всё ещё запущен: **docker ps | grep pyserver** — вывод команды не должен быть пустым. Для просмотра логов контейнера, воспользуйтесь командой **docker logs pyserver**.

Рисунок 9 – Запуск сервера с названием

Для того, чтобы остановить выполнение контейнера, существует команда **docker stop pyserver**. Однако, если снова попробовать запустить командой

docker run -it -p8000:8000 --name pyserver -d python python -m http.server, то возникнет ошибка: контейнер с таким именем существует. Его нужно удалить **docker rm pyserver**.

Для остановки и удаления контейнера можно воспользоваться командой **docker rm -f pyserver** вместо выполнения двух отдельных команд **stop** и **rm**. После удаления контейнер с таким именем можно будет создать заново.

Для того, чтобы контейнер удалялся после завершения работы, нужно указать флаг **--rm** при его запуске — далее в работе мы будем использовать данный флаг:

docker run --rm -p8000:8000 --name pyserver -d python python -m http.server

1.5 Постоянное хранение данных

Запустите контейнер, в котором веб-сервер будет отдавать содержимое директории /mnt: **docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt**,

где **-d /mnt** указывает модулю http.server какая директория будет корневой для отображения.

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

-p 8000:8000 *– пробрасывает порт сервера 8000 на порт хоста 8000*

--name pyserver *– называет сервер именем "pyserver"*

--rm *– удаляет сервер при закрытии*

-d *– запускает сервер отдельно от данной сессии терминала*

python python -m http.server -d /mnt *– команда, выполняемая после запуска контейнера*

Для того, чтобы попасть в уже запущенный контейнер, существует команда **docker exec -it pyserver bash**

— вы попадёте в оболочку *bash* в контейнере. Попав в контейнер, выполните команду

cd mnt && echo "hello world" > hi.txt, а затем выйдите из контейнера, введя команду exit или нажав комбинацию клавиш Ctrl+D.

```
PS C:\Users\krezo> docker run -p 8000:8000 --name pyserver --rm -d python python -m http.server -d /m
b605d6c0214ebb5f512c7016f54741469c46ee3f7bfc20f807a9e4928ce64ee6
PS C:\Users\krezo> docker exec -it pyserver bash
root@b605d6c0214e:/# cd mnt && echo "hello world" > hi.txt
```

Рисунок 10 – Запуск команды на сервере

Если открыть http://0.0.0.0:8000/, там будет доступен файл hi.txt.

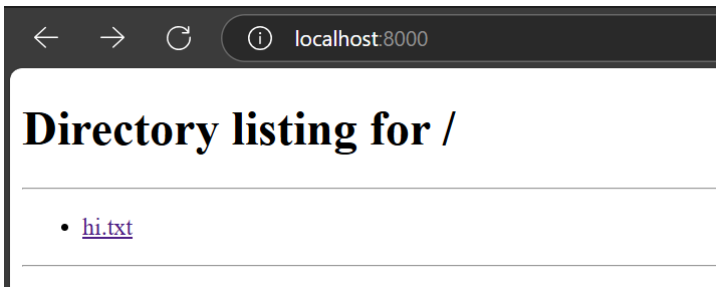

Рисунок 11 – Сайт по адресу localhost:8000

Остановим контейнер: **docker stop pyserver**, а затем снова запустим:

docker run -p 8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt.

Как мы видим, файл hi.txt пропал — это неудивительно, ведь мы запустили другой контейнер, а старый был удалён после завершения работы (флаг --rm). Остановим контейнер: **docker stop pyserver**.

Для того, чтобы не терялись какие-то данные (например, если запущен контейнер с СУБД, то чтобы не терялись данные из неё) существует механизм монтирования.

```
PS C:\Users\krezo> docker stop pyserver
pyserver
PS C:\Users\krezo> docker run -p 8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt
a0f939664b436088bf3a70d6b73099fed54f45532fc0bef0d8e15a439e2dac63
PS C:\Users\krezo> docker stop pyserver
pyserver
```

Рисунок 12 – Повторный запуск сервера

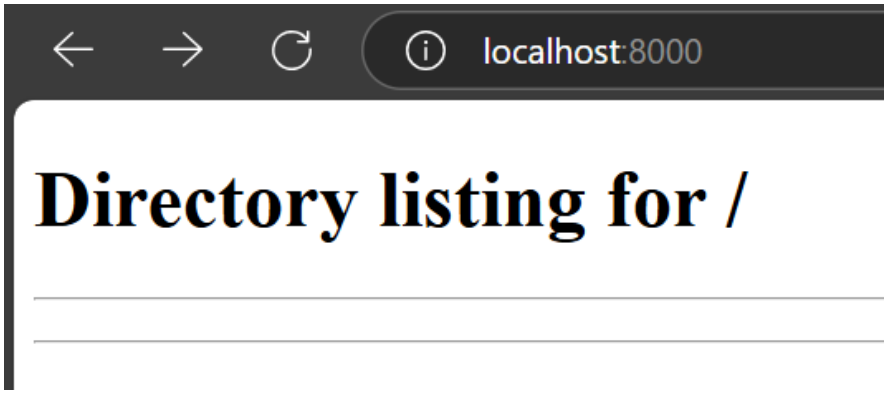

Рисунок 13 - Отсутствие файла hi.txt по адресу localhost:8000

1.5.1 Тома

Первый способ — это создать отдельный том с помощью ключа **-v myvolume:/mnt**, где myvolume — название тома, /mnt — директория в контейнере, где будут доступны данные.

Попробуйте снова создать контейнер, но уже с примонтированным томом:

docker run -p 8000:8000 --rm --name pyserver -d -v "$(pwd)/myfiles:/mnt" python python -m http.server -d/mnt

Затем, если создать файл (выполнить **docker exec -it pyserver bash** и внутри контейнера выполнить **cd mnt && echo "hello world" > hi.txt**), то даже после удаления контейнера данные в этом томе будут сохранены.

Чтобы узнать, где хранятся данные, выполните команду

docker inspect -f "{{json .Mounts }}" pyserver, в поле Source будет храниться путь до тома на хостовой машине.

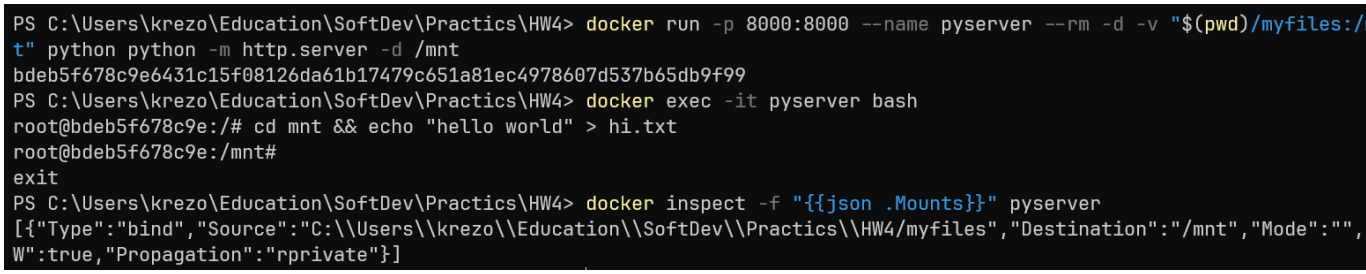

Рисунок 14 – Запуск сервера с томом

Для управления томами существует команда **docker volume**, ознакомиться с которой предлагается самостоятельно.

1.5.2 Монтирование директорий и файлов

Сперва, остановите контейнер, созданный на предыдущем шаге: **docker stop pyserver**.

Иногда требуется пробросить в контейнер конфигурационный файл или отдельную директорию. Для этого используется монтирование директорий и файлов.

Создадим директорию и файлы, которые будем монтировать. Часть из них нам понадобится дальше: создайте директорию: **mkdir myfiles**, в ней создайте файл host.txt: **touch myfiles/host.txt**

Запустите контейнер:

docker run -p 8000:8000 –name pyserver --rm -d -v "$(pwd)/myfiles:/mnt" python python -m http.server -d /mnt

Команда **pwd** — выведет текущую директорию, например: /home/user/dome-directory, в итоге получился абсолютный путь до файла: /home/user/dome-directory/myfiles.

Затем, зайдите в контейнер: **docker exec -it pyserver bash**, перейдите в директорию /mnt командой **cd /mnt**. Если вывести список файлов командой **ls**, то там будет файл host.txt, примонтированный вместе с директорией myfiles

Создайте файл **echo "hello world" > hi.txt**, а затем выйдите из контейнера: exit. Теперь на хостовой машине в директории myfiles/ появится файл hi.txt. Проверить можно командой **ls myfiles**.

Остановите контейнер: docker stop pyserver.

```
PS C:\Users\krezo\Education\SoftDev\Practics\HW4> docker run -p 8000:8000 --name pyserver --rm -d -v "$(pwd)/myfiles:/mnt" python python -m http.server -d /mnt
8aadea3d33dee050ec2a3fa8b211d5d458bad0739847985c76c58038ca1a3a1e
PS C:\Users\krezo\Education\SoftDev\Practics\HW4> docker exec -it pyserver bash
root@8aadea3d33de:/# cd mnt/
root@8aadea3d33de:/mnt# ls
host.txt
root@8aadea3d33de:/mnt# echo "hello world" > hi.txt
root@8aadea3d33de:/mnt#
exit
PS C:\Users\krezo\Education\SoftDev\Practics\HW4> ls .\myfiles\


    Directory: C:\Users\krezo\Education\SoftDev\Practics\HW4\myfiles

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          20.04.2025    20:16             12 hi.txt
-a---          20.04.2025    20:14              0 host.txt

PS C:\Users\krezo\Education\SoftDev\Practics\HW4> docker stop pyserver
pyserver
```

Рисунок 15 – Монтирование директории в сервер

Для того, чтобы примонтировать один файл, нужно указать ключ -v, например:

-v $(pwd)/myfiles/host.txt:/mnt/new-name-of-host.txt – файлу в контейнере присвоится другое имя: new-name-of-host.txt.

1.6 Переменные окружения

Для передачи переменных окружения внутрь контейнера используется ключ **-e**. Например, чтобы передать в контейнер переменную окружения *MIREA* со значением «*ONE LOVE*», нужно добавить ключ **-e MIREA="ONE LOVE"**.

Проверьте, выведя все переменные окружения, определённые в контейнере с помощью утилиты env: **docker run -it --rm -e MIREA="ONE LOVE" ubuntu env**. Среди списка переменных будет и MIREA

```
PS C:\Users\krezo\Education\SoftDev\Practics\HW4> docker run --rm -e MIREA="ONE LOVE" ubuntu env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=65a412cd0a78
MIREA=ONE LOVE
HOME=/root
```

Рисунок 16 – Добавление переменной окружения на сервер

1.7 Dockerfile

Соберите образ, в который будут установлены дополнительные пакеты, примонтируйте директорию и установите команду запуска. Для этого создаётся файл Dockerfile (без расширения).

```
1 FROM ubuntu:20.04
2 RUN apt update \
3     && apt install -y python3 fortune \
4     && cd /usr/bin \
5     && ln -s python3 python
6 RUN /usr/games/fortune > /mnt/greeting-while-building.txt
7 ADD ./data /mnt/data
8 EXPOSE 80
9 CMD ["python", "-m", " http.server", "-d", "/mnt/", "80"]
```

В строке (1) указывается базовый образ, на основе которого будет строиться новый образ.

В строках (2-5) указана команда, которая выполнится в процессе сборки. На самом деле, там выполняются несколько команд, соединённых && для того, чтобы создавать меньше слоёв в образе.

В строках (6) тоже указана команда, которая сгенерирует случайную цитату и перенаправит вывод в файл /mnt/greeting-while-building.txt. Файл будет сгенерирован во время сборки образа.

В строке (7) копируется всё содержимое директории ./data хостовой машины в директорию /mnt, которая будет доступна в контейнере.

В строке (8) указывается, какой порт у контейнера будет открыт.

В строке (9) указывается команда, которая будет выполнена при запуске, где 80 — порт, который будет слушать веб-сервер.

Соберите образ с тегом mycoolimage с помощью команды **docker build -t mycoolimage .** Точка в конце указывает на текущую директорию, где лежит Dockerfile.

Запуск производится командой **docker run --rm -it -p 8099:80 mycoolimage**, где порт 8099 — порт на хостовой машине.

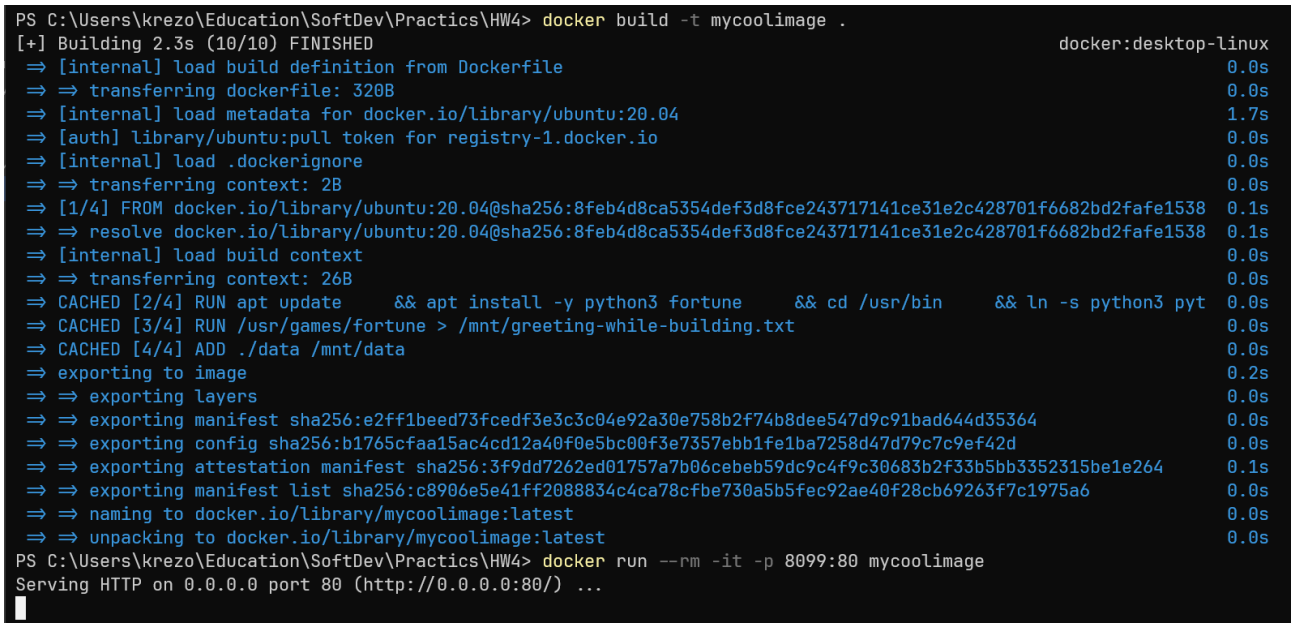

Рисунок 17 – Сборка образа с использованием Dockerfile

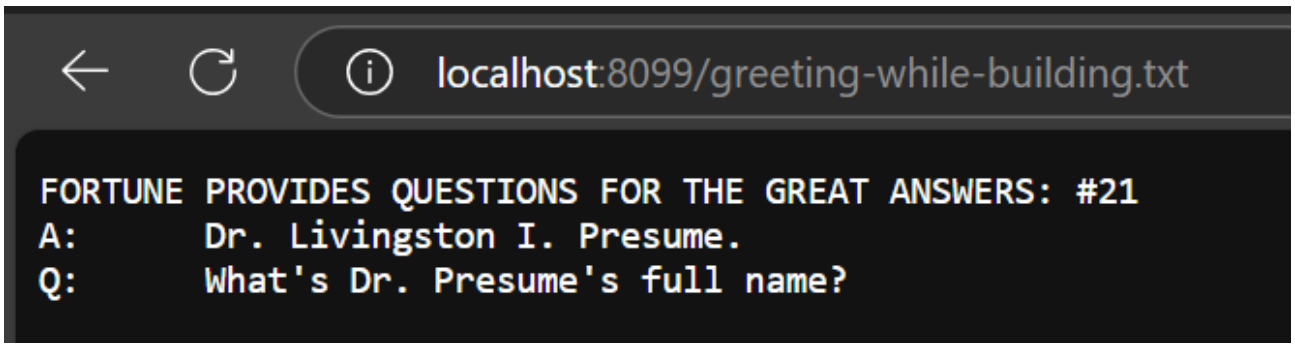

Рисунок 18 – Результат запуска сервера с образа

1.8 Индивидуальное задание

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

Для монтирования создайте директорию data и в ней файл student.txt, содержащий ФИО, название группы и номер варианта.

Для установки пакетов использовать команду **apt install -y** название-пакета. В качестве примера можно использовать Dockerfile из раздела 7.

Запустить веб-сервер, отображающий содержимое /mnt/files, в хостовой системе должен открываться на порту 880013.

Установить пакет **nginx**, при этом:

- необходимо использовать базовый образ ubuntu:20.04
- примонтировать директорию data в директорию /mnt/files/ в контейнере.

Сначала напишем подходящий Dockerfile.

```
Dockerfile
1 FROM ubuntu:20.04
2 RUN apt update \
3     && apt install -y python3 nginx \
4     && cd /usr/bin \
5     && ln -s python3 python
6 ADD ./data /mnt/files
7 EXPOSE 8000
8 CMD ["python", "-m", "http.server", "-d", "/mnt/", "8000"]
9
```

Рисунок 19 – Dockerfile для индивидуального задания

```
PS C:\Users\krezo\Education\SoftDev\Practics\HW4> docker build -t hw4 .
[+] Building 38.4s (9/9) FINISHED                                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                             0.0s
 => => transferring dockerfile: 264B                                                                                             0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                                                                  1.6s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                                                                    0.0s
 => [internal] load .dockerignore                                                                                                0.0s
 => => transferring context: 2B                                                                                                  0.0s
 => CACHED [1/3] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f6682bd2f                0.1s
 => => resolve docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f6682bd2fafe1538                0.1s
 => [internal] load build context                                                                                                0.0s
 => => transferring context: 127B                                                                                                0.0s
 => [2/3] RUN apt update     && apt install -y python3 nginx     && cd /usr/bin     && ln -s python3 python                     25.7s
 => [3/3] ADD ./data /mnt/files                                                                                                  0.2s
 => exporting to image                                                                                                          10.6s
 => => exporting layers                                                                                                          8.8s
 => => exporting manifest sha256:272777db82ca767aa6bd5d4a947522d2871bf07be0fddcac11ad5f4a4e112906                                0.0s
 => => exporting config sha256:a990e848a43d6baed5fca7ede2170a381d587e8deaf6f86426721d6beb08b00b                                  0.0s
 => => exporting attestation manifest sha256:7abbbab51315c09e7bdc5d50a637c2ec02589f5c01a5430de0dd6c1e9c16591b                    0.1s
 => => exporting manifest list sha256:7c6a285dbedf7ccc6862cb3ded6b132cdb218be443710c05cd31dbb35205e22b                           0.0s
 => => naming to docker.io/library/hw4:latest                                                                                    0.0s
 => => unpacking to docker.io/library/hw4:latest                                                                                 1.6s
PS C:\Users\krezo\Education\SoftDev\Practics\HW4> docker run --rm -p 8813:8000 hw4
```

Рисунок 20 – Запуск сервера индивидуального задания

Результаты запуска сервера представлены на рисунках

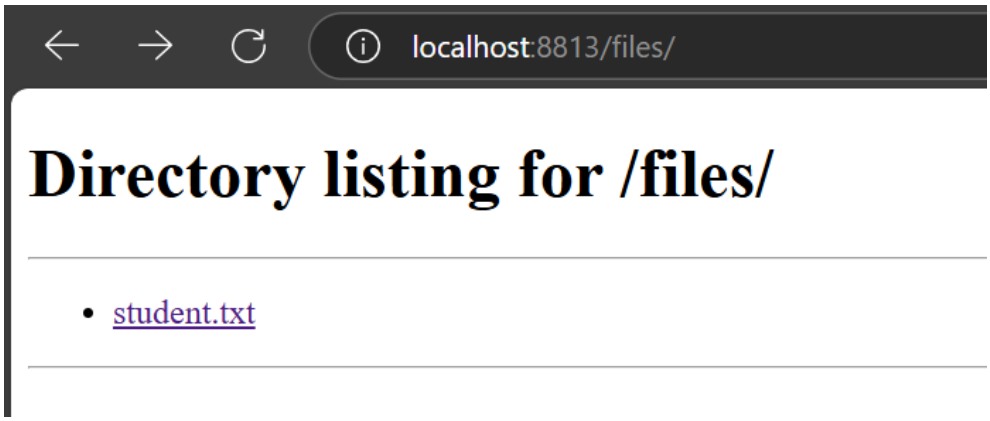

Рисунок 21

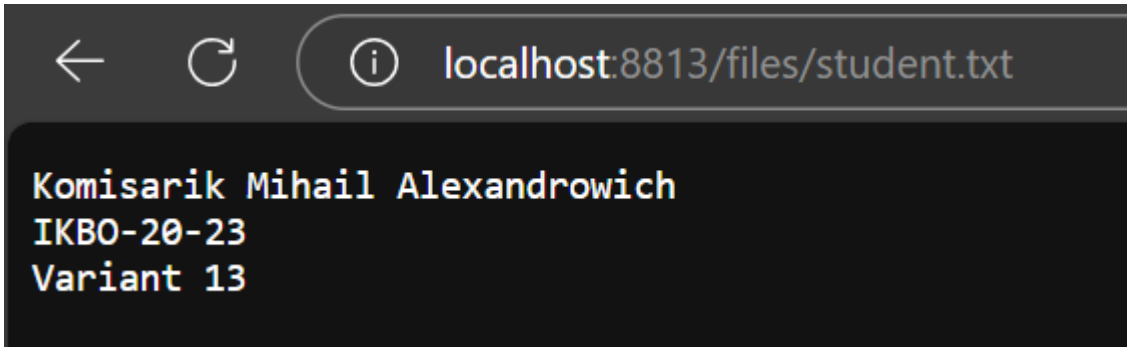

Рисунок 22

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы были приобретены и закреплены навыки работы с Docker, включая создание и настройку контейнеров, работу с HTTP-сервером в контейнеризованной среде, а также сборку проекта с использование Dockerfile.