



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных
технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3
по дисциплине

«Технология разработки программных приложений»

Тема: «Docker»

Выполнил студент группы ИКБО-20-21

Алибеков А.Т.

Принял

Петренко А.А.

Практическая работа выполнена

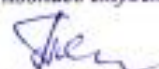
✓ «29» 04 2023 г.

 (подпись студента)

«Зачтено»



✓ «29» 04 2023 г.


(подпись руководителя)

Москва 2023

Оглавление

1. Постановка задачи	3
1.1. Цель работы.....	3
1.2. Задание (Вариант 1).....	3
2. Ход работы	3
2.1. Образы	3
2.2. Изоляция.....	4
2.3. Работа с портами.....	5
2.4. Именованные контейнеры, остановка и удаление	7
2.5. Постоянное хранение данных	8
2.5.1. Тома	9
2.5.2. Монтирование директорий и файлов	10
2.6. Переменные окружения	11
2.7. Dockerfile	12
2.8. Индивидуальные задания	12
3. Вывод	13

1. Постановка задачи

1.1. Цель работы

Знакомство с контейнеризатором приложении Docker. Возможности Docker.

1.2. Задание (Вариант 1)

В практической работе необходимо выполнить все шаги из разделов 1–7. В отчёт должны быть включены ответы на вопросы, выделенные курсивом, результаты выполнения команд из разделов 1–7, а также выполненное индивидуальное задание (раздел 8): листинг Dockerfile, а также команды сборки и запуска контейнера.

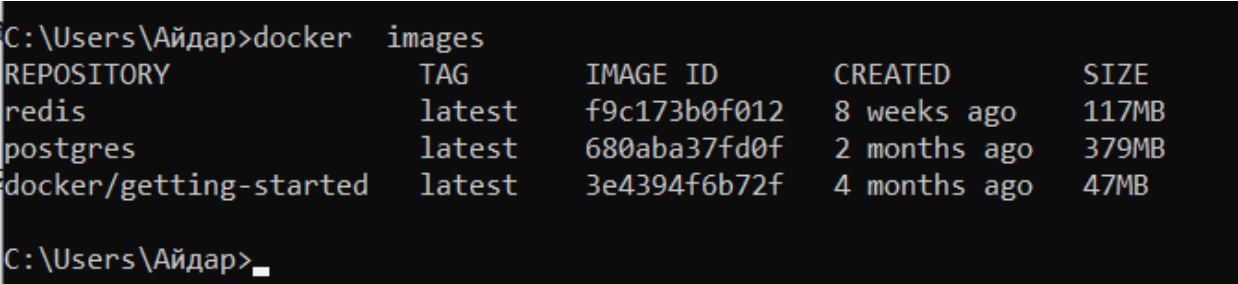
ВАРИАНТ 1:

Установить пакет, согласно варианту: 1. cowsay

2. Ход работы

2.1. Образы

Посмотрите на имеющиеся образы: `docker images`.



```
C:\Users\Айдар>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis                latest              f9c173b0f012       8 weeks ago        117MB
postgres             latest              680aba37fd0f       2 months ago       379MB
docker/getting-started latest              3e4394f6b72f       4 months ago       47MB
C:\Users\Айдар>
```

Рисунок 1 – Образы

Загрузите образ: `docker pull ubuntu` — будет загружен образ `ubuntu:latest` — последняя доступная версия. Для загрузки конкретной версии, нужно указать тег, например, 12.04: `docker pull ubuntu:12.04`.

Посмотрите на имеющиеся образы ещё раз: `docker images` — должны появиться новые загруженные образы.

Посмотрите список контейнеров, выполнив команду: `docker ps -a`.

```
C:\Users\Айдар>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
2ab09b027e7f: Pull complete
Digest: sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

C:\Users\Айдар>docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
ubuntu              latest          08d22c0ceb15   7 weeks ago    77.8MB
redis               latest          f9c173b0f012   8 weeks ago    117MB
postgres            latest          680aba37fd0f   2 months ago   379MB
docker/getting-started latest          3e4394f6b72f   4 months ago   47MB

C:\Users\Айдар>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS      PORTS          NAMES
18c9fa1ee6e2   redis    "docker-entrypoint.s..." 7 weeks ago    Exited (0) 7 weeks ago           MyCodtainer

C:\Users\Айдар>
```

Рисунок 2 – Загрузка образа

2.2. Изоляция

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

Вопрос: одинаковый ли результат получился при разных запусках?

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

Вопрос: Одинаковый ли результат получился при разных запусках?

```
C:\Users\Айдар>hostname
DESKTOP-FQ7KIVD

C:\Users\Айдар>hostname
DESKTOP-FQ7KIVD

C:\Users\Айдар>docker run ubuntu hostname
be805e4ed687

C:\Users\Айдар>docker run ubuntu hostname
ea0f867fedd0

C:\Users\Айдар>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS      PORTS          NAMES
ea0f867fedd0   ubuntu    "hostname"              13 seconds ago Exited (0) 11 seconds ago           trusting_bratta
in
be805e4ed687   ubuntu    "hostname"              18 seconds ago Exited (0) 17 seconds ago           priceless_frank
lin
18c9fa1ee6e2   redis    "docker-entrypoint.s..." 7 weeks ago    Exited (0) 7 weeks ago           MyCodtainer

C:\Users\Айдар>
```

Рисунок 3 – Информация о хостовой системе

Ответ на вопросы: для моего ПК – одинаковый, для контейнеров в докере – каждый раз разный.

Запустите `bash` в контейнере: `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`.

```
C:\Users\Айдар>docker run ubuntu bash
C:\Users\Айдар>docker run -it ubuntu bash
root@dc05de558cd0:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@dc05de558cd0:/#
```

Рисунок 4 – Запуск контейнера с `-it` и без

2.3. 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`.

```
C:\Users\Айдар>docker pull python
Using default tag: latest
latest: Pulling from library/python
b0248cf3e63c: Pull complete
127e97b4daf7: Pull complete
0336c50c9f69: Pull complete
1b89f3c7f7da: Pull complete
2d6277217976: Pull complete
273fcda609d8: Pull complete
58568d3a3a00: Pull complete
56fc9fb54f6e: Pull complete
8a22f29afe36: Pull complete
Digest: sha256:f7382f4f9dbc51183c72d621b9c196c1565f713a1fe40c119d215c961fa22815
Status: Downloaded newer image for python:latest
docker.io/library/python:latest

C:\Users\Айдар>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.

C:\Users\Айдар>
```

Рисунок 5 – Работа с образом `python`

Для проброса портов используется флаг -p hostPort:containerPort

Добавьте его, чтобы пробросить порт 8000:

`docker run -it -p8000:8000 python python -m http.server` — теперь по адресу <http://0.0.0.0:8000/> 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.

```
C:\Users\Айдар>docker run -it -p8000:8000 python python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.17.0.1 - - [28/Apr/2023 21:58:12] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [28/Apr/2023 21:58:12] code 404, message File not found
172.17.0.1 - - [28/Apr/2023 21:58:12] "GET /favicon.ico HTTP/1.1" 404 -
^C
Keyboard interrupt received, exiting.

C:\Users\Айдар>docker run -it -p8888:8000 python python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.17.0.1 - - [28/Apr/2023 21:59:01] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [28/Apr/2023 21:59:02] code 404, message File not found
172.17.0.1 - - [28/Apr/2023 21:59:02] "GET /favicon.ico HTTP/1.1" 404 -
^C
Keyboard interrupt received, exiting.

C:\Users\Айдар>
```

Рисунок 6 – Проброс порта

Directory listing for /

-
- [./dockerenv](#)
 - [bin/](#)
 - [boot/](#)
 - [dev/](#)
 - [etc/](#)
 - [home/](#)
 - [lib/](#)
 - [lib64/](#)
 - [media/](#)
 - [mnt/](#)
 - [opt/](#)
 - [proc/](#)
 - [root/](#)
 - [run/](#)
 - [sbin/](#)
 - [srv/](#)
 - [sys/](#)
 - [tmp/](#)
 - [usr/](#)
 - [var/](#)
-

Рисунок 7 – Содержимое по адресу

2.4.Именованные контейнеры, остановка и удаление

Запустите контейнер: `docker run -it -p8000:8000 python python -m http.server`. Для того, чтобы запустить контейнер в фоне, нужно добавить флаг `-d/--detach`. Также определим имя контейнеру, добавив флаг `--name`.

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

Убедитесь, что контейнер всё ещё запущен: `docker ps`. Для просмотра логов контейнера, воспользуйтесь командой `docker logs pyserver`. Для того, чтобы остановить выполнение контейнера, существует команда `docker stop pyserver`.

```
C:\Users\Айдар>docker run -it -p8000:8000 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.

C:\Users\Айдар>docker run -p 8000:8000 --name pyserver -d python python -m http.server
46f9e930fb12861e51d2df517613bbbed8561df791f8480cd4311818ea6d9a29

C:\Users\Айдар>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMES
46f9e930fb12   python   "python -m http.serv..." 28 seconds ago Up 27 seconds  0.0.0.0:8000->8000/
p   pyserver

C:\Users\Айдар>docker logs pyserver

C:\Users\Айдар>docker stop pyserver
pyserver

C:\Users\Айдар>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
C:\Users\Айдар>
```

Рисунок 8 – Запуск контейнера в фоне и с именем

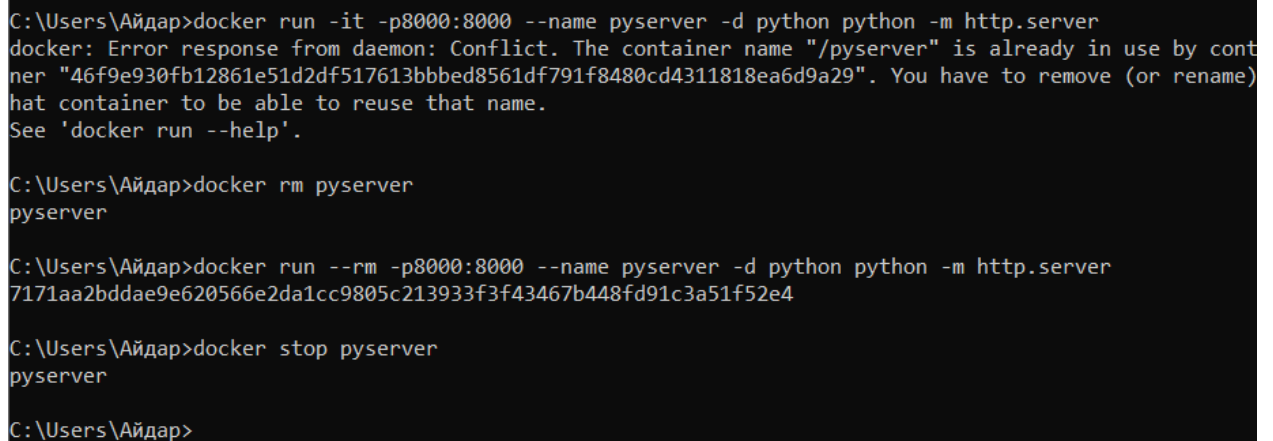
Однако, если снова попробовать запустить командой

`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
```



```
C:\Users\Айдар>docker run -it -p8000:8000 --name pyserver -d python python -m http.server
docker: Error response from daemon: Conflict. The container name "/pyserver" is already in use by container "46f9e930fb12861e51d2df517613bbbed8561df791f8480cd4311818ea6d9a29". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.

C:\Users\Айдар>docker rm pyserver
pyserver

C:\Users\Айдар>docker run --rm -p8000:8000 --name pyserver -d python python -m http.server
7171aa2bddae9e620566e2da1cc9805c213933f3f43467b448fd91c3a51f52e4

C:\Users\Айдар>docker stop pyserver
pyserver

C:\Users\Айдар>
```

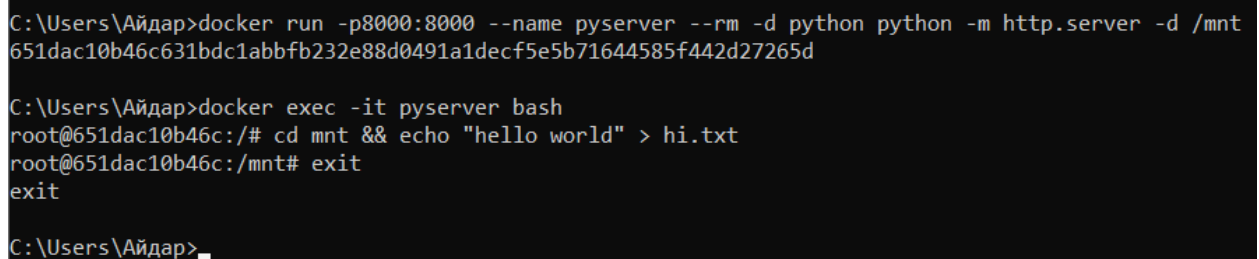
Рисунок 9 – Запуск контейнера с флагом `--rm`

2.5. Постоянное хранение данных

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

```
docker exec -it pyserver bash
cd /mnt && echo "hello world" > hi.txt
```

Выполните команды выше, а затем выйдите из контейнера, введя команду `exit`.



```
C:\Users\Айдар>docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt
651dac10b46c631bdc1abbfb232e88d0491a1decf5e5b71644585f442d27265d

C:\Users\Айдар>docker exec -it pyserver bash
root@651dac10b46c:/# cd /mnt && echo "hello world" > hi.txt
root@651dac10b46c:/mnt# exit
exit

C:\Users\Айдар>
```

Рисунок 10 – Запуск контейнера и переход в оболочку `bash`

Остановим контейнер: `docker stop pyserver`, а затем снова запустим. Видно, что файл пропал, ведь прошлый контейнер удалился (флаг `--rm`).


```
C:\Users\Айдар>docker stop pyserver
pyserver

C:\Users\Айдар>docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt
3ffc3ba5a93c800edcd6770e9b368f8d8d12ab334ffd65cdf7ebec8194ac135c

C:\Users\Айдар>docker exec -it pyserver bash
root@3ffc3ba5a93c:/# cd /mnt
root@3ffc3ba5a93c:/mnt# ls
root@3ffc3ba5a93c:/mnt#
```

Рисунок 11 – Повторный запуск контейнера и переход в оболочку bash

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

Ответ на вопрос:

- `-p8000:8000` – пробрасывает порт 8000 из контейнера в порт 8000 на хосте, так веб-сервер, запущенный в контейнере, может быть доступным по адресу <http://localhost:8000>;
- `--name pyserver` – задает имя контейнеру, конкретно – `pyserver`;
- `--rm` – флаг для автоматического удаления контейнера после его остановки;
- `-d` – запускает контейнер в фоновом режиме;
- `python python -m http.server -d /mnt` – команда, которая выполнится в контейнере сразу после его запуска: запустит веб-сервер на порту 8000 с корневой директорией `/mnt`.

2.5.1. 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.

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.

```

C:\Users\Айдар>docker run -p8000:8000 --rm --name pyserver -d -v mydirectory:/mnt python python -m http.server -d /mnt
14290f04fc7e17d24ac3a11b5d2bfb9ccd5ed2b1f77313d507d81511ea8ff9fc

C:\Users\Айдар>docker exec -it pyserver bash
root@14290f04fc7e:/# cd mnt && echo "hello world" > hi.txt
root@14290f04fc7e:/mnt# ls
hi.txt
root@14290f04fc7e:/mnt# exit
exit

C:\Users\Айдар>docker stop pyserver
pyserver

C:\Users\Айдар>docker run -p8000:8000 --rm --name pyserver -d -v mydirectory:/mnt python python -m http.server -d /mnt
74f412b94c62ebcd138f3694ee9653e7b31d8cc408815ab5a6d5f684fab0321a

C:\Users\Айдар>docker exec -it pyserver bash
root@74f412b94c62:/# cd mnt
root@74f412b94c62:/mnt# ls
hi.txt
root@74f412b94c62:/mnt#

```

Рисунок 12 – Данные сохранились

Чтобы узнать, где хранятся данные, выполните команду

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

```

C:\Users\Айдар>docker inspect -f "{{json .Mounts }}" pyserver
[{"Type":"volume","Name":"mydirectory","Source":"/var/lib/docker/volumes/mydirectory/_data","Destination":"/mnt","Driver":"local","Mode":"z","RW":true,"Propagation":""}]

C:\Users\Айдар>

```

Рисунок 13 – Местоположение данных

2.5.2. Монтирование директорий и файлов

Иногда требуется пробросить в контейнер конфигурационный файл или отдельную директорию. Для этого используется монтирование директорий и файлов. Создадим директорию и файлы, которые будем монтировать. Затем запустим контейнер и смонтируем созданный файл. Можно также создать файл в контейнере, даже после остановки и удаления контейнера – он будет находиться на хосте.

```

C:\Users\Айдар>docker run -p8000:8000 --rm --name pyserver -d -v C:/Users/Айдар/myfilesTRPP:/mnt python python -m http.server -d /mnt
e2c7c0f906f2aa8893d57bad15bce825caff51f275032194d26ae37e0152f2d3

C:\Users\Айдар>docker exec -it pyserver bash
root@e2c7c0f906f2:/# cd /mnt
root@e2c7c0f906f2:/mnt# ls
root@e2c7c0f906f2:/mnt# echo "hello world" > hi.txt
root@e2c7c0f906f2:/mnt# ls
hi.txt
root@e2c7c0f906f2:/mnt# exit
exit

C:\Users\Айдар>docker stop pyserver
pyserver

C:\Users\Айдар>cd myfilesTRPP

C:\Users\Айдар\myfilesTRPP>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 9056-8B84

Содержимое папки C:\Users\Айдар\myfilesTRPP

29.04.2023  09:27    <DIR>        .
29.04.2023  09:27    <DIR>        ..
29.04.2023  09:27                12 hi.txt
               1 файл          12 байт
               2 папок      5 843 076 байт свободно

C:\Users\Айдар\myfilesTRPP>

```

Рисунок 14 – Монтирование файла

2.6. 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`.

```

C:\Users\Айдар\myfilesTRPP>docker run -it --rm -e MIREA="ONE LOVE" ubuntu env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=84b4886e79c2
TERM=xterm
MIREA=ONE LOVE
HOME=/root

```

Рисунок 15 – Переменные окружения

2.7. Dockerfile

Соберите образ, в который будут установлены дополнительные пакеты, примонтируйте директорию и установите команду запуска. Для этого создаётся файл Dockerfile (без расширения). Соберите образ с тегом mycoolimage с помощью команды `docker build -t mycoolimage .` Точка в конце указывает на текущую директорию, где лежит Dockerfile.

```
C:\Users\Айдар\myfilesTRPP>docker build -t mycoolimage .
[+] Building 44.6s (9/9) FINISHED
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 32B                                              0.0s
=> [internal] load .dockerignore                                                 0.1s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/ubuntu:22.04                 0.8s
=> CACHED [1/4] FROM docker.io/library/ubuntu:22.04@sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21 0.0s
=> [internal] load build context                                                0.1s
=> => transferring context: 59B                                                0.0s
=> [2/4] RUN apt update && apt install -y python3 fortune && cd /usr/bin && ln -s python3 python 40.9s
=> [3/4] RUN /usr/games/fortune > /mnt/greeting-while-building.txt             0.9s
=> [4/4] ADD ./data /mnt/data                                                  0.2s
=> exporting to image                                                         1.3s
=> => exporting layers                                                         1.3s
=> => writing image sha256:92b07a039850713a4a3b8b8492ada06602396102e4abb9a758bfc3fad35f704 0.0s
=> => naming to docker.io/library/mycoolimage                                0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\Айдар\myfilesTRPP>docker run --rm -it -p8099:80 mycoolimage
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Рисунок 16 – Сборка образа

2.8. Индивидуальные задания

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 **8801**.

Установить пакет, согласно **варианту 1: cowsay**

```
FROM ubuntu:22.04
RUN apt-get update && apt-get install -y \
    cowsay
RUN mkdir /mnt/files
WORKDIR /mnt/files
VOLUME /data
EXPOSE 8801
ENTRYPOINT ["python3"]
CMD ["-m", "http.server", "-d", "/mnt/", "8801"]
```

Рисунок 17 – Содержимое файла Dockerfile

```
C:\Users\Айдап\myfilesTRPP>docker build -t myimage .
[+] Building 1.6s (9/9) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 257B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/ubuntu:22.04 1.4s
=> [auth] library/ubuntu:pull token for registry-1.docker.io 0.0s
=> [1/4] FROM docker.io/library/ubuntu:22.04@sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a 0.0s
=> CACHED [2/4] RUN apt-get update && apt-get install -y cowsay 0.0s
=> CACHED [3/4] RUN mkdir /mnt/files 0.0s
=> CACHED [4/4] WORKDIR /mnt/files 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => writing image sha256:6bd7827658d0283e942d8e5d345b0a4c574b2bcb74dacd8237efd364e769273c 0.0s
=> => naming to docker.io/library/myimage 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\Айдап\myfilesTRPP>docker run -it -p 8801:8801 -v C:/Users/Айдап/myfilesTRPP/data:/data myimage
```

Рисунок 18 – Сборка образа по индивидуальному заданию

3. Вывод

В ходе выполнения данной практической работы были изучены основные команды Docker, а также были получены практические навыки по работе с контейнеризатором приложений Docker.