|  |
| --- |
|  |
| 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 (DOCKER)** |
| **по дисциплине** |
| **«Технологии разработки программных приложений»**  **Тема: «Системы контроля версий»** |

|  |  |
| --- | --- |
| Выполнил студент группы ИКБО-08-21 | Юнаков И.П. |
| Принял | Туманова М.Б. |

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

Москва 2023

Оглавление

[Часть 1 – Образы 3](#_Toc133933062)

[1. Посмотрите на имеющиеся образы, загрузить новый и проверить образы ещё раз: 3](#_Toc133933063)

[Часть 2 – Изоляция 3](#_Toc133933064)

[1. Просмотр информации о хостовой системе: 3](#_Toc133933065)

[2. Выполнение hostname через docker: 4](#_Toc133933066)

[Часть 3 – Работа с портами 5](#_Toc133933067)

[Часть 4 – Именованные контейнеры, остановка и удаление 5](#_Toc133933068)

[Часть 5 – Постоянное хранение данных 6](#_Toc133933069)

[Часть 5.1 – Тома 7](#_Toc133933070)

[Часть 5.2 – Монтирование директорий и файлов 7](#_Toc133933071)

[Часть 6 – Переменные окружения 8](#_Toc133933072)

[Часть 7 – Dockerfile 9](#_Toc133933073)

[Часть 8 – Индивидуальное задание 10](#_Toc133933074)

[Часть 9 – Вывод 10](#_Toc133933075)

# Часть 1 – Образы

1. Посмотрите на имеющиеся образы, загрузить новый и проверить образы ещё раз:
   * 1. Посмотреть на имеющиеся образы можно с помощью команды docker images.
     2. Загрузим образ со стандартной версией ubuntu, используя команду docker pull ubuntu
     3. Ещё раз проверим имеющиеся образы командой docker images

root@yunikeil:~# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

root@yunikeil:~# 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

root@yunikeil:~# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu latest 08d22c0ceb15 6 weeks ago 77.8MB

root@yunikeil:~#

# Часть 2 – Изоляция

1. Просмотр информации о хостовой системе:
   * 1. Просмотреть информацию о системе можно, используя команду hostname.
     2. Одинаковый ли результат будет при разных запусках?

Команда hostname предназначена для вызова одноименной утилиты, осуществляющей вывод и изменение имени узла, ассоциированного с системой. Имя узла является частью доменного имени системы, предназначенного для ее уникальной идентификации в рамках локальной сети без необходимости использования IP-адресов Результат будет одинаковым при различных запусках, если никто не изменял «компьютера».

root@yunikeil:/mnt/d/.repos/Python/TRPP/pract3(docker)# hostname

yunikeil

root@yunikeil:/mnt/d/.repos/Python/TRPP/pract3(docker)# who

root pts/1 2023-04-27 13:58

root@yunikeil:/mnt/d/.repos/Python/TRPP/pract3(docker)# hostname

yunikeil

1. Выполнение hostname через docker:
   * 1. Попробуем выполнить то же самое в контейнерах. Выполните два раза команду docker run ubuntu hostname.
     2. Одинаковый ли результат получился при разных запусках?

Вывод будет разным, т.к. команда docker run будет каждый раз запускать новый образ ubuntu с новым «именем системы».

root@yunikeil:/mnt/d/.repos/Python/TRPP/pract3(docker)# docker run ubuntu hostname

66613179dab3

root@yunikeil:/mnt/d/.repos/Python/TRPP/pract3(docker)# docker run ubuntu hostname

4b4f695b067d

root@yunikeil:/mnt/d/.repos/Python/TRPP/pract3(docker)# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

4b4f695b067d ubuntu "hostname" 3 minutes ago Exited (0) 3 minutes ago lucid\_yonath

66613179dab3 ubuntu "hostname" 3 minutes ago Exited (0) 3 minutes ago goofy\_ganguly

root@yunikeil:/mnt/d/.repos/Python/TRPP/pract3(docker)#

* + 1. Для того, чтобы подключиться к контейнеру можно использовать команду docker run -it ubuntu bash:

Флаги -it в Docker используются для запуска контейнера в интерактивном режиме с подключением к его терминалу.

Конкретно, -i (или --interactive) указывает Docker, что нужно открыть стандартный ввод (stdin) для контейнера. Это позволяет пользователю взаимодействовать с контейнером через терминал.

Флаг -t (или --tty) указывает Docker на выделение псевдо-TTY, то есть терминального интерфейса, для контейнера. Это позволяет пользователю взаимодействовать с контейнером через терминал и использовать команды оболочки внутри контейнера.

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker run -it ubuntu bash

root@d92b9b4a54db:/# hostname

d92b9b4a54db

root@d92b9b4a54db:/# exit

exit

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker ps -a

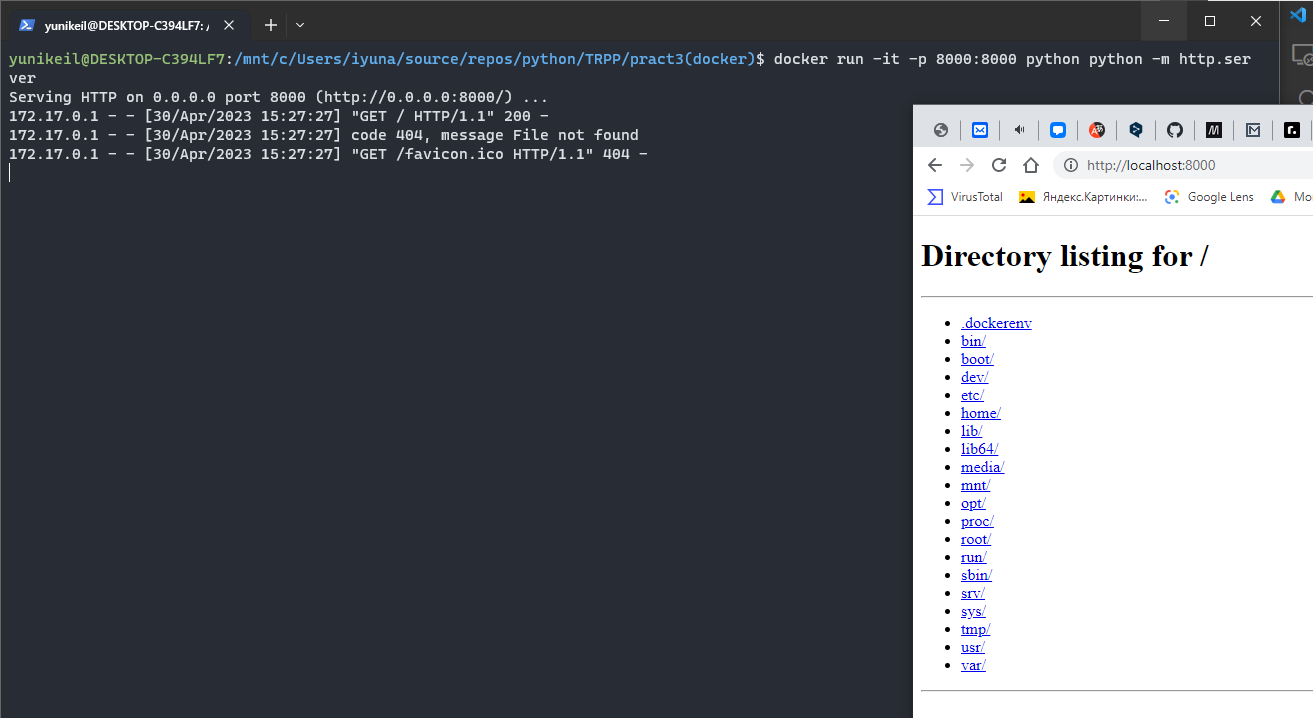
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d92b9b4a54db ubuntu "bash" 16 seconds ago Exited (0) 6 seconds ago flamboyant\_sinoussi

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$

# Часть 3 – Работа с портами

Для выполнения задания работы с портами потребуется установить любой веб сервер. Я буду использовать простой http встроенный в python. Командой docker run -it python python -m http.server запустится сервер, однако у нас не получится получить доступ к нему через стандартный localhost, т. к. отсутствует проброс портов. Исправить это можно, исправив оригинальную команду на docker run -it -p 8000:8000 python python -m http.server. В данном случае мы сделали проброс с 8000 порта контейнера на 8000 порт wsl2 windows.

Рисунок 1 – Создание простого http сервера использую docker и python

# Часть 4 – Именованные контейнеры, остановка и удаление

Не всегда понятно, что именно работает в контейнере по его id, для этого можно задать имя с помощью параметра --name. А для того, чтобы он не мешал работе или выполнению какого-либо скрипта контейнер можно запустить в фоне. Для этого понадобится команда docker run -p 8000:8000 --name pyserver -d python python -m http.server. Для того чтобы убедиться в работе контейнера можно отсортировать контейнеры по имени используя команду docker ps | grep pyserver. Просмотреть логи можно используя команду docker logs pyserver. Для остановки контейнера можно использовать docker stop pyserver, затем следует его удалять командой docker rm pyserver. Данные команды можно заменить одной, используя docker rm -f pyserver. Также можно снова подключиться к контейнеру, используя docker start и docker attach/-it: docker start -a pyserver. А для автоматического удаления после выполнения стоит добавить флаг –rm: docker run --rm -p8000:8000 --name pyserver -d python python -m http.server .

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker run -p8000:8000 --name pyserver -d python python -m http.server

7ce5551c9e005351b7f1ca7dff9ed4292082a8c786227b7bfc5f6b8e849a2657

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

7ce5551c9e00 python "python -m http.serv…" 6 seconds ago Up 5 seconds 0.0.0.0:8000->8000/tcp, :::8000->8000/tcp pyserver

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker logs pyserver

172.17.0.1 - - [30/Apr/2023 15:49:10] "GET / HTTP/1.1" 200 -

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker stop pyserver

pyserver

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker start -a pyserver

172.17.0.1 - - [30/Apr/2023 15:49:44] "GET / HTTP/1.1" 200 -

# Часть 5 – Постоянное хранение данных

Для запуска контейнера, в котором веб-сервер будет отдавать содержимое директории /mnt, можно использовать команду docker run с флагами -p 8000:8000, --name pyserver, -rm и -d, а также указать команду, которая должна была выполниться внутри контейнера: python -m http.server -d /mnt. Флаг -d /mnt указал модулю http.server, какая директория будет корневой для отображения.

Чтобы попасть в уже запущенный контейнер, существует команда docker exec -it pyserver bash. После ее выполнения, я попал в оболочку bash в контейнере. Внутри контейнера я перешел в директорию /mnt и создал файл hi.txt, используя команду echo "hello world" > hi.txt. Затем нужно выйти из контейнера, введя команду exit или нажав комбинацию клавиш Ctrl+D.

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

Остановить контейнер можно с помощью команды docker stop pyserver. После этого, чтобы запустить контейнер снова нужно использовать ту же команду docker run, но файл hi.txt уже не был доступен. Это произошло потому, что я использовал флаг --rm при запуске контейнера, который автоматически удаляет контейнер после его остановки.

Чтобы не терять какие-то данные, существует механизм монтирования.

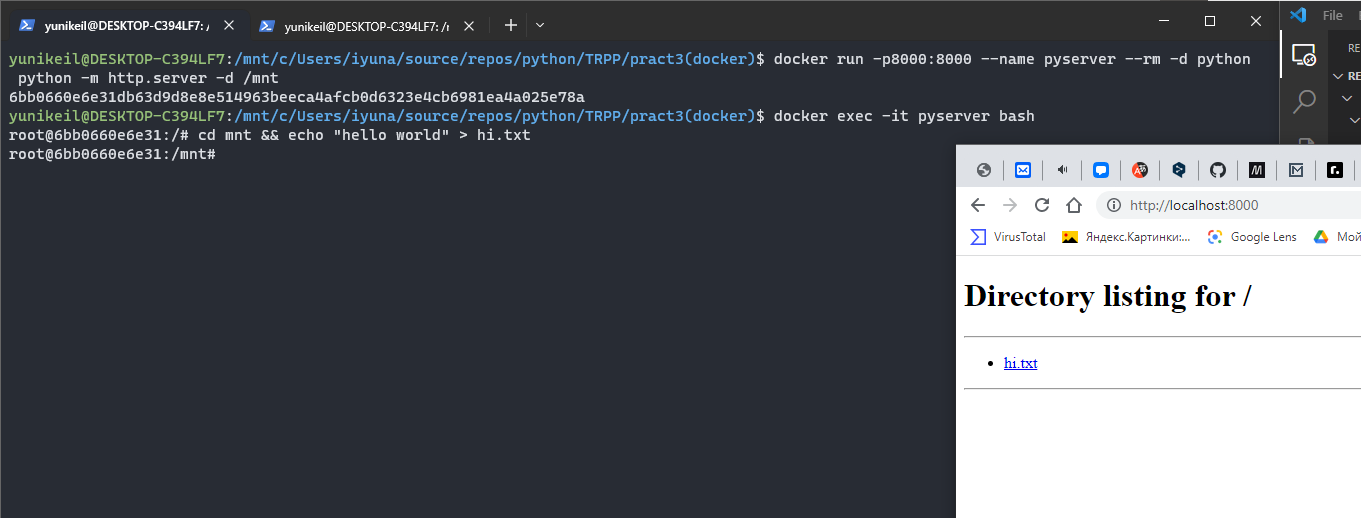


Рисунок 2 – Пятая часть практики

# Часть 5.1 – Тома

Контейнеры можно создавать с примонтированными томами: docker run -p8000 :8000 --rm --name pyserver -d -v $(pwd)/ myfiles :/mnt python python -m http.server -d /mnt. Затем если создать файл выполнив две команды, то даже после удаления контейнеры данные будут сохранены.

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker) docker exec -it pyserver bash

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker) cd mnt && echo "hello world" > hi.txt

Для того чтобы узнать, где именно хранятся данные нужно выполнить данную команду: docker inspect -f "{{json .Mounts }}" pyserver. В поле Source файлы json будет храниться путь до тома на хостовой машине. Для управления томами в docker существует команда volume.

# Часть 5.2 – Монтирование директорий и файлов

Для продолжения нужно остановить контейнер pyserver созданный на прошлом шаге используя команду docker stop pyserver. Чтобы пробросить в контейнер файл или директорию стоит использовать монтирование. Создадим директорию myfiles и файл host.txt…

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker) mkdir myfiles

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker) cd myfilest

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)/myfiles touch host.txt

… запустим контейнер командой docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python python -m http.server -d /mnt. Затем используя команду docker exec -it pyserver bash зайдём в контейнер, перейдём в директорию /mnt. При выводе списка файлов обнаружим что в директории появился файл host.txt, примонтированный вместе с директорией myfiles.

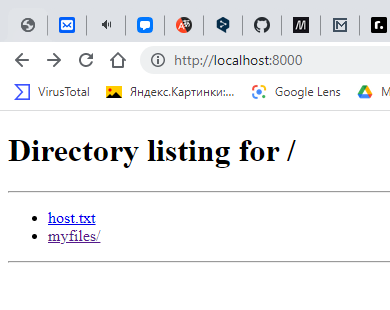


Рисунок 3 – Часть 5.2

# Часть 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".

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$ docker run -it --rm -e MIREA="ONE LOVE" ubuntu env

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:

HOSTNAME=40271d99920c

TERM=xterm

MIREA=ONE LOVE

HOME=/root

yunikeil@DESKTOP-C394LF7:/mnt/c/Users/iyuna/source/repos/python/TRPP/pract3(docker)$

# Часть 7 – Dockerfile

Для сборки образа, в который будут установлены дополнительные пакеты, смонтированы директории и т.д. создаётся файл Dockerfile.

FROM ubuntu:20.04

RUN apt-get update \

&& apt-get install -y python3 fortune \

&& cd /usr/bin \

&& ln -s python3 python

RUN /usr/games/fortune > /mnt/greeting-while-building.txt

ADD ./data /mnt/data

EXPOSE 80

CMD ["python", "-m", "http.server", "-d", "/mnt/", "80"]

Для сборки образа с тегом "mycoolimage" нужно выполнить команду: docker build -t mycoolimage . , для запуска контейнера нужно выполнить команду: docker run --rm -it -p 8099:80 mycoolimage. Эта команда запускает контейнер с образом "mycoolimage" и пробрасывает порт 8099 на хостовой машине на порт 80 в контейнере. Ключи --rm и -it указывают на удаление контейнера после его завершения и подключение к терминалу контейнера соответственно.

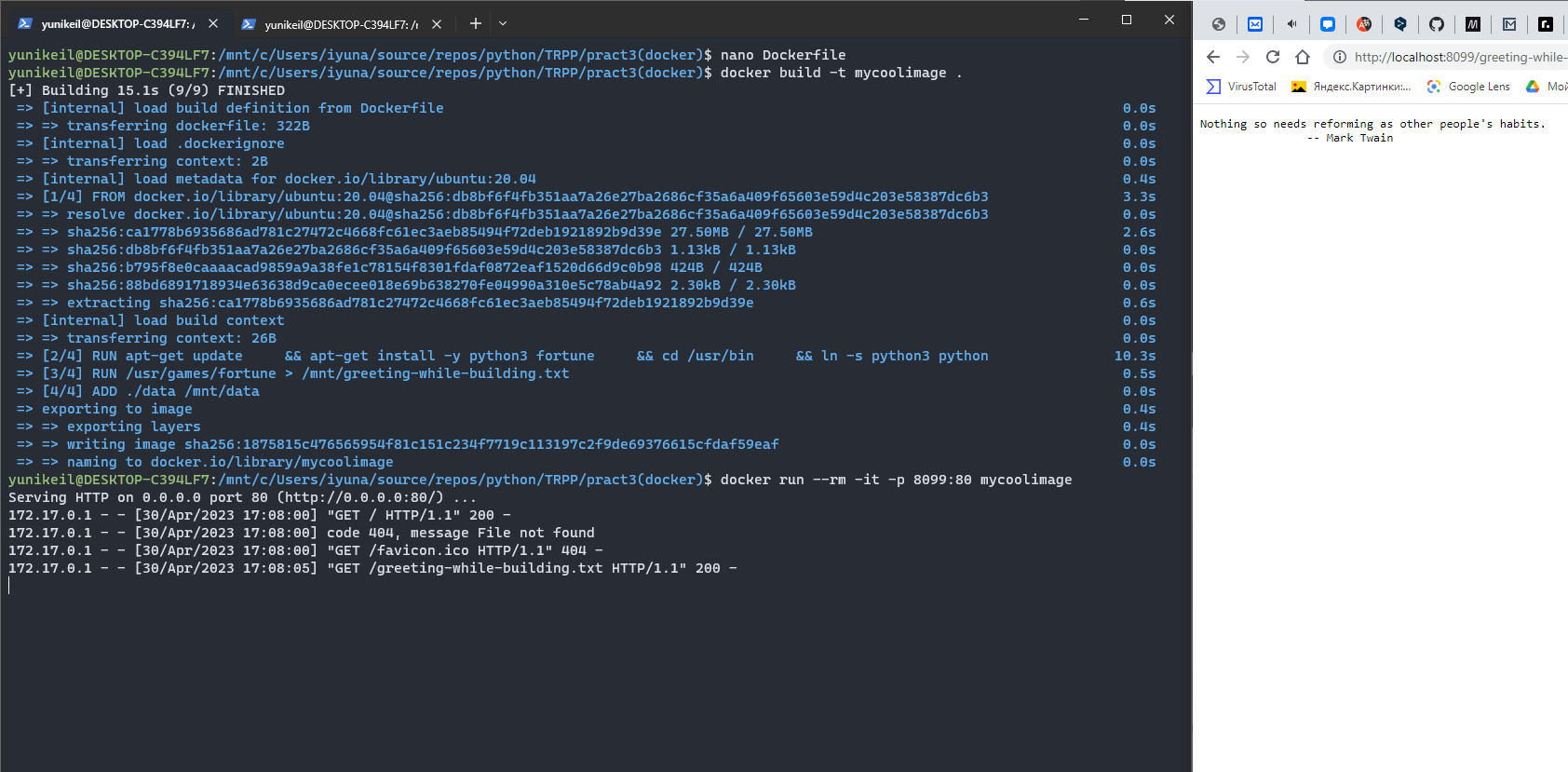
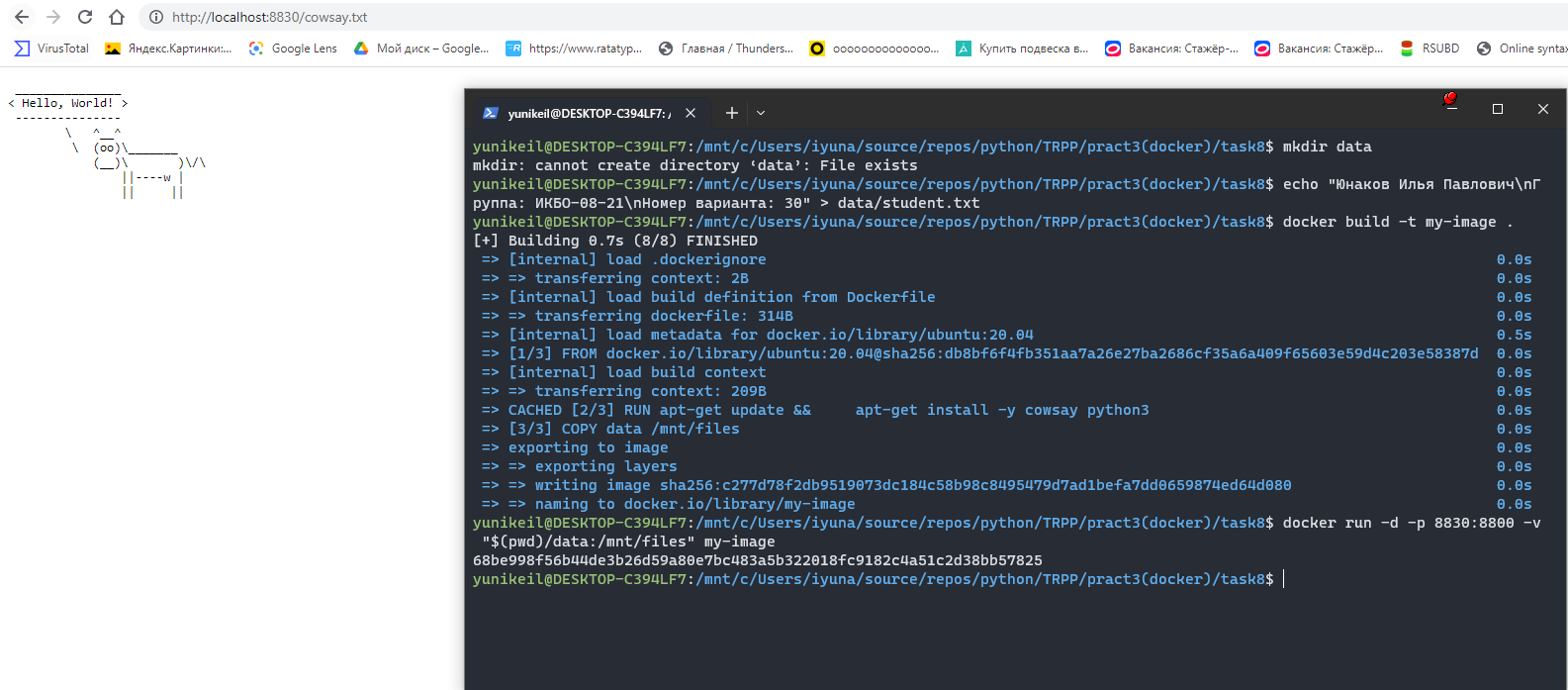


Рисунок 4 – Создание контейнера по Dockerfile

# Часть 8 – Индивидуальное задание

Листинг 1 – Dockerfile для выполнения индивидуального задания

1. FROM ubuntu:20.04
2. RUN apt-get update && \
3. apt-get install -y cowsay python3
4. COPY data /mnt/files
5. ENV PATH="/usr/games:${PATH}"
6. EXPOSE 8800
7. CMD ["sh", "-c", "cowsay 'Hello, World!' > /mnt/files/cowsay.txt && cd /mnt/files && python3 -m http.server 8800"]

Рисунок 5 – Индивидуальное задание

# Часть 9 – Вывод

В ходе данной работы я научился создавать удалять изменять образы и запущенные контейнеры Docker.