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 №3

по дисциплине «Технологии разработки программных приложений»

Тема практической работы: «Docker»

Выполнил:

Студент группы ИКБО-04-20 Хан А.А.

Проверила: Овчинникова М.А.

Москва 2022 г.

СОДЕРЖАНИЕ

Цель работы: Получение навыков работы с Docker: установка Docker, скачивание и использование образов, работа с контейнерами, работа с переменными окружения, создание Dockerfile.

Индивидуальный вариант: 9.

Часть 1. Образы

Посмотрим на имеющиеся образы: docker images (см. Листинг 1.1).

Листинг 1.1 – Имеющиеся образы

```
khan@LAPTOP-UCPRRTEQ:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    feb5d9fea6a5   6 months ago   13.3kB
```

Загрузим образ: docker pull ubuntu – будет загружен образ ubuntu:latest – последняя доступная версия (см. Листинг 1.2).

Листинг 1.2 – Загрузка образа

```
khan@LAPTOP-UCPRRTEQ:~$ docker pull ubuntuUsing
default tag: latest
latest: Pulling from library/ubuntu
4d32b49e2995: Pull complete
Digest: sha256:bea6d19168bbfd6af8d77c2cc3c572114eb5d113e6f422573c93cb605a0e2ffb
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

Для загрузки конкретной версии, нужно указать тег, например, 12.04: docker pull ubuntu:12.04. Посмотрим на имеющиеся образы ещё раз: docker images – должны появиться новые загруженные образы (см. Листинг 1.3).

Листинг 1.3 – Имеющиеся образы

```
khan@LAPTOP-UCPRRTEQ:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    ff0fea8310f3   7 days ago     72.8MB
hello-world   latest    feb5d9fea6a5   6 months ago   13.3kB
```

Посмотрим список контейнеров, выполнив команду: docker ps –a (см. Листинг 1.4).

Листинг 1.4 – Список контейнеров

```
khan@LAPTOP-UCPRRTEQ:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS
PORTS     NAMES
71cc02f38a3e   hello-world   "/hello"   7 minutes ago   Exited (0) 7 minutes
```

```
ago             modest_dewdney
```

Часть 2. Изоляция

Посмотрим информацию о хостовой системе, выполнив команду hostname. Выполним её ещё один раз. Результат будет одинаковый (см. Листинг 2.1).

Листинг 2.1 – Информация о хостовой системе

```
khan@LAPTOP-UCPRRTEQ:~$ hostname
LAPTOP-UCPRRTEQ
khan@LAPTOP-UCPRRTEQ:~$ hostname
LAPTOP-UCPRRTEQ
```

Попробуем выполнить то же самое в контейнерах. Выполним два раза команду docker run ubuntu hostname. Результат будет разный. Это связано с тем, что из одного образа были запущены два изолированных контейнера (см. Листинг 2.2).

Листинг 2.2 – Информация о хостовой системе в контейнерах

```
khan@LAPTOP-UCPRRTEQ:~$ docker run ubuntu hostname
2b447873414c
khan@LAPTOP-UCPRRTEQ:~$ docker run ubuntu hostname
4040eb6756ec
```

Заново выполним docker ps –a, там появятся запущенные ранее контейнеры (см. Листинг 2.3).

Листинг 2.3 – Список контейнеров

```
khan@LAPTOP-UCPRRTEQ:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND      CREATED          STATUS
PORTS     NAMES
4040eb6756ec   ubuntu        "hostname"   3 minutes ago    Exited (0) 3
minutes ago              optimistic_hopper
2b447873414c   ubuntu        "hostname"   3 minutes ago    Exited (0) 3
minutes ago              sad_grothendieck
71cc02f38a3e   hello-world   "/hello"     17 minutes ago   Exited (0) 17
minutes ago            modest_dewdney
```

Запустим bash в контейнере: docker run ubuntu bash (см. Листинг 2.4).

Листинг 2.4 – Запуск bush

```
khan@LAPTOP-UCPRRTEQ:~$ docker run ubuntu bash
khan@LAPTOP-UCPRRTEQ:~$
```

Ничего не произошло. потому что нужно запустить контейнер в интерактивном режиме с помощью команды docker run -it ubuntu bash (см. Листинг 2.5).

Листинг 2.5 – Запуск bash в интерактивном контейнере

```
khan@LAPTOP-UCPRRTEQ:~$ docker run -it ubuntu bash
root@bbff642547ff:/# hostname
bbff642547ff
```

Из Листинга видно, что мы зашли в bash и получили имя хоста нашего контейнера.

Для запуска контейнера с определенной версией образа, необходимо указывать версию через двоеточие после названия образа. Например: docker run ubuntu:12.04 hostname.

Часть 3. Работа с портами

Загрузим образ python командой docker pull python. В качестве примера, запустим встроенный в Python модуль веб-сервера из корня контейнера, чтобы отобразить содержание контейнера: docker run -it python python -m http.server (см. Листинг 3.1).

Листинг 3.1 – Установка образа и запуск модуля

```
khan@LAPTOP-UCPRRTEQ:~$ docker pull pythonUsing
default tag: latest
latest: Pulling from library/python
5492f66d2700: Pull complete
540ff8c0841d: Pull complete
a0bf850a0df0: Pull complete
d751dc38ae51: Pull complete
9720a112e886: Pull complete
f97b81fbdbd9: Pull complete
a70c58953c25: Pull complete
6f7b858c1584: Pull complete
74b4b07d81e4: Pull complete
Digest: sha256:6441e2f0bd2e566de0df6445cb8e7e395ea1a376dd702de908d70401d3700961
Status: Downloaded newer image for python:latest
docker.io/library/python:latest
khan@LAPTOP-UCPRRTEQ:~$ docker run -it python python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

При запуске пишется, что сервер доступен по адресу http://localhost:8000/. Однако, если открыть этот адрес, то ничего не будет видно, потому что порты не проброшены. Для проброса портов используется флаг -phostPort:containerPort.

Добавим его, чтобы пробросить порт 8000: docker run -it –p 8000:8000 python python -m http.server (см. Листинг 3.2).

Листинг 3.2 – Подбрасывание портов

```
khan@LAPTOP-UCPRRTEQ:~$ 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 - - [25/Mar/2022 08:21:04] "GET / HTTP/1.1" 200 -
```

Теперь по адресу http://localhost:8000 открывается содержимое корневой директории в контейнере (см. Рисунок 3.1).

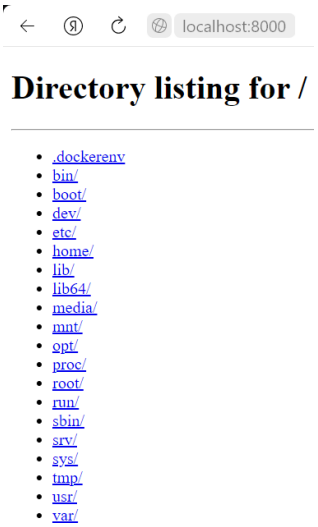

Рисунок 3.1 - Директория с файлами контейнера

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 (см. Листинг 3.3, Рисунок 3.2).

Листинг 3.3 - Запуск веб-сервер на порту 8888

```
khan@LAPTOP-UCPRRTEQ:~$ 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 - - [25/Mar/2022 08:30:43] "GET / HTTP/1.1" 200 -
```

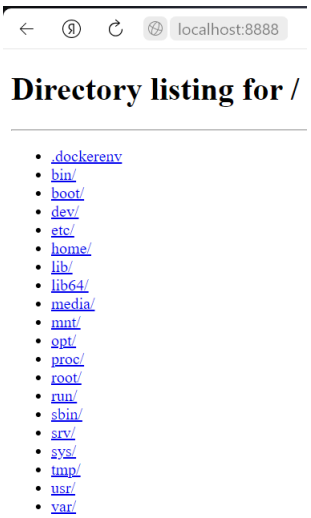

Рисунок 3.2 – Запуск веб-сервера на порту 8888

Часть 4. Именованные контейнеры, остановка и удаление

Для того, чтобы запустить контейнер в фоне необходимо добавить флаг -d. Запустим контейнер в фоновом режиме, а также присвоим ему имя при помощи флага –name (см. Листинг 4.1).

Листинг 4.1 – Запуск контейнера в фоновом режиме

```
khan@LAPTOP-UCPRRTEQ:~$ docker run -p8000:8000 --name pyserver -d pythonpython
-m http.server
d48dafec359e5f50d75bdcb92c1f51103c4d0794899d39bed6c503012d894f30
```

Убедимся, что контейнер работает с помощью команды docker ps | grep pyserver, а также просмотрим логи контейнера при помощи команды docker logs pyserver (см. Листинг 4.2).

Листинг 4.2 – Проверка работы конейнера и просмотр лога

```
khan@LAPTOP-UCPRRTEQ:~$ docker ps | grep pyserver
d48dafec359e   python    "python -m http.serv…"   3 minutes ago   Up 3
minutes   0.0.0.0:8000->8000/tcp   pyserver
khan@LAPTOP-UCPRRTEQ:~$ docker logs pyserver
172.17.0.1 - - [25/Mar/2022 08:43:42] "GET / HTTP/1.1" 200 -
```

Для остановки контейнера используется команда docker stop pyserver. Однако если после этого попробовать запустить его еще раз, возникнет ошибка, что такой контейнер уже существует. Поэтому его необходимо его удалить командой docker rm -f pyserver (см. Листинг 4.3).

Листинг 4.3 – Остановка и удаление контейнера

```
khan@LAPTOP-UCPRRTEQ:~$ docker stop pyserver
pyserver
khan@LAPTOP-UCPRRTEQ:~$ docker run -p8000:8000 --name pyserver -d pythonpython
-m http.server
docker: Error response from daemon: Conflict. The container name "/pyserver"is
already in use by container
"d48dafec359e5f50d75bdcb92c1f51103c4d0794899d39bed6c503012d894f30". You haveto
remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
khan@LAPTOP-UCPRRTEQ:~$ docker rm –f
pyserverpyserver
```

Для того, чтобы контейнер удалялся после завершения работы, нужно указать флаг –rm при его запуске. Команда: docker run --rm -p8000:8000 --name pyserver -d python python -m http.server.

Часть 5. Постоянное хранение данных

Запустим контейнер, в котором веб-сервер будет отдавать содержимое директории /mnt. Команда: docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt (см. Листинг 5.1).

Листинг 5.1 – Запуск контейнера, отслеживающего содержимое директории /mnt

```
khan@LAPTOP-UCPRRTEQ:~$ docker run -p8000:8000 --name pyserver --rm -dpython
python -m http.server -d /mnt
c0612390ad4452796742d79c07cd5f2fee57182e2c98e4a160ddb32aa32bd1d2
```

В данном случае мы используем следующие флаги:

- `--p` – проброс порта на хост;
- `--name` – название контейнера;
- `--rm` – удаление контейнера после завершения работы;
- `-d` – запуск контейнера в фоновом режиме;

Команда, которая выполнится в контейнере идет после указания модуля Python http-server: `-d`.

На данный момент в директории /mnt нет никаких файлов, поэтому зайдем в оболочку bash контейнера и создадим новый файл (см. Листинг 5.2).

Листинг 5.2 – Вход в оболочку контейнера и создание нового файла

```
khan@LAPTOP-UCPRRTEQ:~$ docker exec -it pyserver bash
```

```
root@c0612390ad44:/# cd mnt && echo "hello world" > hi.txt
root@c0612390ad44:/mnt# exit
exit
```

Теперь если открыть http://localhost:8000, там будет доступен файл hi.txt со строчкой «hello world» (см. Рисунки 5.1-5.2).

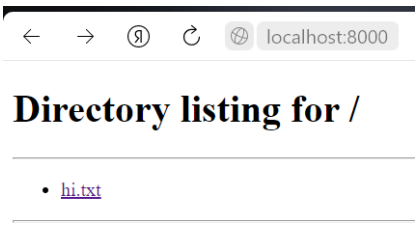

Рисунок 5.1 – Содержимое директория mnt на веб-сервере

Рисунок 5.2 – Содержимое файла hi.txt

Остановим контейнер: docker stop pyserver, а затем снова запустим: docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt (см. Листинг 5.3).

Листинг 5.3 – Остановка и запуск контейнера

```
khan@LAPTOP-UCPRRTEQ:~$ docker stop pyserver
pyserver
khan@LAPTOP-UCPRRTEQ:~$ docker run -p8000:8000 --name pyserver --rm -dpython
python -m http.server -d /mnt
be91112aa39a4dcf3f50206650f598a44440affc83b84bf6d0dbffc504086bb8
```

Снова откроем http://localhost:8000, файл hi.txt пропал (см. Рисунок 5.3). Это произошло потому что мы запустили другой контейнер, а старый был удалён после завершения работы (флаг--rm).

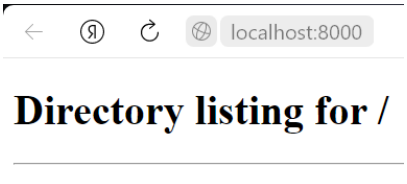

Рисунок 5.3 – Содержимое директория mnt на веб-сервере

Остановим контейнер:docker stop pyserver.

5.1 Тома

Для того, чтобы данные не терялись существует механизм монтирования. Делается это с помощью ключа -v myvolume:/mnt, где myvolume – название тома, /mnt – директория в контейнере, где будут доступны данные. Попробуем снова создать контейнер, но уже с примонтированным томом. Сразу же создадим файл как в прошлом пункте (см. Листинг 5.1.1).

Листинг 5.1.1 – Создание контейнера с промонтированным топом

```
khan@LAPTOP-UCPRRTEQ:~$ docker run -p8000:8000 --rm --name pyserver -d -v
$(pwd)/myfiles:/mnt python python -m http.server -d /mnt
94faacb531e1f62fc825b3d890a6cd0ad30983383907d5c4f60dd2e88421335b khan@LAPTOP-
UCPRRTEQ:~$ docker exec -it pyserver bash root@94faacb531e1:/# cd mnt && echo
"hello world" > hi.txt root@94faacb531e1:/mnt# exit
exit
```

Чтобы узнать где хранятся данные, выполним команду docker inspect -f "{{json .Mounts }}" pyserver, в поле Source будет храниться путь до тома на хостовой машине (см. Листинг 5.1.2).

Листинг 5.1.2 – Путь до тома

```
khan@LAPTOP-UCPRRTEQ:~$ docker inspect -f "{{json .Mounts }}" pyserver
[{"Type":"bind","Source":"/home/khan/myfiles","Destination":"/mnt","Mode
":"","RW":true,"Propagation":"rprivate"}]
```

Остановим контейнер:docker stop pyserver.

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. Часть из них нам понадобится дальше: создадим директорию: mkdir myfiles и файл host.txt в ней (см. Листинг 5.2.1).

Листинг 5.2.1 – Создание папки и файла. Запуск контейнера

```
khan@LAPTOP-UCPRRTEQ:~$ mkdir myfiles khan@LAPTOP-
UCPRRTEQ:~$ touch myfiles/host.txt
khan@LAPTOP-UCPRRTEQ:~$ docker run -p8000:8000 --rm --name pyserver -d -v
$(pwd)/myfiles:/mnt python python -m http.server -d /mnt
68e53b5b526a42dd46f160f56ede615537e8bdbb67c95ea62df519b6dd16a843
```

Затем зайдем в контейнер: docker exec -it pyserver bash, перейдем в

директорию /mnt командой cd /mnt. Если вывести список файлов командой ls, то там будет файл host.txt, примонтированный вместе с директорией myfiles. Создадим файл echo "hello world" > hi.txt, а затем выйдем из контейнера: exit. Теперь на хостовой машине в директории myfiles/ появится файл hi.txt. Проверить можно командой ls myfiles. Остановим контейнер: docker stop pyserver (см. Листинг 5.2.2).

Листинг 5.2.2 – Создание папки и файла. Запуск контейнера

```
khan@LAPTOP-UCPRRTEQ:~$ docker exec -it pyserver bash
root@68e53b5b526a:/# cd mnt
root@68e53b5b526a:/mnt# ls
host.txt
root@68e53b5b526a:/mnt# echo "hello world" > hi.txt
root@68e53b5b526a:/mnt# exit
exit
khan@LAPTOP-UCPRRTEQ:~$ ls myfileshi.txt
host.txt
```

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 (см. Листинг 5.2.3).

Листинг 5.2.3 – Монтирование отдельного файла в контейнер

```
khan@LAPTOP-UCPRRTEQ:~$ docker run -p8000:8000 --rm --name pyserver -d -v
$(pwd)/myfiles/host.txt:/mnt/new-name-of-host.txt python python -m http.server
-d /mnt 8100235b253c6afe9d72a9903e2ec2fffd664fe767a408ad66e35973f33b250c
khan@LAPTOP-UCPRRTEQ:~$ docker exec -it pyserver bash root@8100235b253c:/# cd
mnt
root@8100235b253c:/mnt# ls
new-name-of-host.txt
root@8100235b253c:/mnt# exit
exit
```

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". Проверим, 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 (см. Листинг 6.1).

Листинг 6.1 – Передача переменной окружения в контейнер

```
khan@LAPTOP-UCPRRTEQ:~$ docker run -it --rm -e MIREA="ONE LOVE" ubuntuenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=c2cd891deaf8
TERM=xterm
MIREA=ONE LOVE
HOME=/root
```

Часть 7. Dockerfile

Соберем образ, в который будут установлены дополнительные пакеты, примонтируем директорию и установим команду запуска. Для этого создадим Dockerfile. Содержимое файла представлено в Листинге 7.1.

Листинг 7.1 – Содержимое Dockerfile

```
FROM ubuntu:20.04
RUN apt update \
      && apt 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"]
```

Соберем 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 -p8099:80 mycoolimage ., где порт 8099 – порт на хостовой машине (см. Листинг 7.2, Рисунки 7.1-7.2).

Листинг 7.2 – Билд изображения и запуск

```
khan@LAPTOP-UCPRRTEQ:~$ docker build -t mycoolimage .[+]
Building 62.1s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                    0.0s
 => => transferring dockerfile: 38B                                     0.0s
 => [internal] load .dockerignore                                       0.0s
 => => transferring context: 2B                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04         8.0s
 => CACHED [1/4] FROM
      docker.io/library/ubuntu:20.04@sha256:bea6d19168bbfd6af8d77c2cc3c572114eb5d1
  13e6f422573c93cb605a0e2ff                                               0.0s
   => [internal] load build context                                      0.0s
   => => transferring context: 26B                                       0.0s
 => [2/4] RUN apt update && apt install -y python3 fortune && cd /usr/bin &&ln
  -s python3 python                                                     51.6s
   => [3/4] RUN /usr/games/fortune > /mnt/greeting-while-building.txt    1.1s
 => [4/4] ADD ./data /mnt/data                                          0.1s
 => exporting to image                                                  1.0s
 => => exporting layers                                                 1.0s
 => => writing image
sha256:6aecbdc926ad8c915e2c2d71adb4fb529d0f36545a6641462b695822e730b971    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
khan@LAPTOP-UCPRRTEQ:~$ docker run --rm -it -p8099:80 mycoolimage
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
172.17.0.1 - - [26/Mar/2022 08:11:50] "GET / HTTP/1.1" 200
```

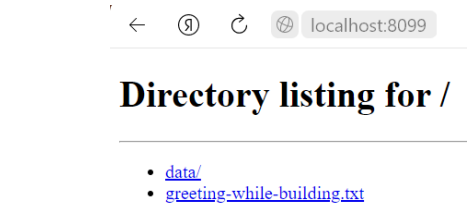

Рисунок 7.1 – Запуск веб-сервера

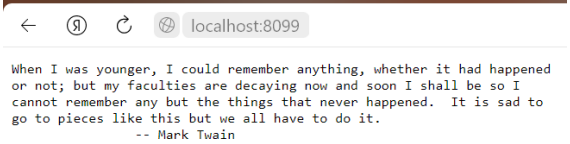

Рисунок 7.2 – Содержимое файла greeting-while-building.txt

Часть 8. Индивидуальное задание

1. Написать Dockfile (базовый образ ubuntu:20.04, для установки пакета использовать командуapt install -y mysql-client), собрать образ (см. Листинг 8.1 – 8.2).

Листинг 8.1 – Содержимое Dockerfile

```
FROM ubuntu:20.04
RUN  apt update && apt install -y python3 mysql-client && cd /usr/bin && ln
-s python3 python
EXPOSE 80
CMD ["python", "-m", "http.server", "-d", "/mnt/files/", "80"]
```

Листинг 8.2 – Сборка образа

```
khan@LAPTOP-UCPRRTEQ:~$ docker build -t individuall .[+]
Building 6.8s (6/6) FINISHED
 => [internal] load build definition from Dockerfile                        0.1s
 => => transferring dockerfile: 230B                                        0.0s
 => [internal] load .dockerignore                                           0.0s
 => => transferring context: 2B                                             0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04             6.5s
 => [1/2] FROM
docker.io/library/ubuntu:20.04@sha256:bea6d19168bbfd6af8d77c2cc3c572114eb5d1
13e6f422573c93cb605a0e  0.0s
 => CACHED [2/2] RUN apt update && apt install -y python3 mysql-client &&cd
/usr/bin && ln -s python3 python        0.0s
 => exporting to image                                                      0.0s
 => => exporting layers                                                     0.0s
 => => writing image
sha256:745db407c2eceb39da128b4f14e58e380a62fd7964fc836b0bab69e6f34c1162    0.0s
 => => naming to docker.io/library/individuall
```

2. Для монтирования создать директорию data и в ней файл student.txt, содержащий ФИО, название группы и номер варианта (см. Листинг 8.3).

Листинг 8.3 – Создание папки и файла

```
khan@LAPTOP-UCPRRTEQ:~$ mkdir data
khan@LAPTOP-UCPRRTEQ:~$ echo "/Khan A.A. IKBO-04-20 9 variant" >
data/student.txt
```

3. Примонтировать директорию data в директорию /mnt/files/ в контейнере (см. Листинг 8.4).

Листинг 8.1 – Монтирование директории

```
khan@LAPTOP-UCPRRTEQ:~$ docker run -it --rm -v $(pwd)/data:/mnt/files -
p8009:80 individuall
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
172.17.0.1 - - [26/Mar/2022 11:05:20] "GET /student.txt HTTP/1.1" 200 -
```

4. Запустить веб-сервер, отображающий содержимое /mnt/files, в хостовой системе должен открываться на порту (8800 + номер варианта: 9). Результат представлен на Рисунках 8.1 – 8.2

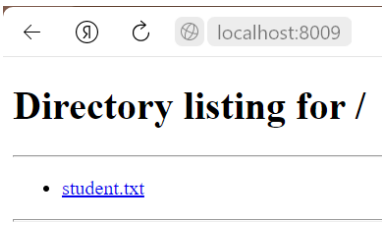

Рисунок 8.1 – Запуск веб-сервера

Рисунок 8.2 – Содержимое примонтированного файла

Вывод

В ходе работы были получены навыки работы с Docker: установка Docker, скачивание и использование образов, работа с контейнерами, работа с переменными окружения, создание Dockerfile.

Список информационных источников

1. Методические указания для выполнения практической работы №3 «Docker» М. А. Овчинникова, МИРЭА – Российский Технологический Университет, 2022.