



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-62-23**

Кокшарова А.А.

**Проверил преподаватель:**

Золотухин С.А.

Москва 2025

## Оглавление

Постановка задачи.....	3
Реализация разделов .....	4
Образы.....	4
Изоляция .....	4
Работа с портами .....	5
Именованные контейнеры, остановка и удаление.....	7
Постоянное хранение данных.....	7
Тома .....	8
Монтирование директорий и файлов .....	9
Переменные окружения .....	9
Dockerfile.....	9
Индивидуальный вариант .....	11
Вывод .....	13

## **Постановка задачи**

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.

1. Образы
2. Изоляция
3. Работа с портами
4. Именованные контейнеры, остановка и удаление
5. Постоянное хранение данных
6. Переменные окружения
7. Dockerfile
8. Индивидуальное задание

## Реализация разделов

### Образы

На рисунке 1 изображена загрузка образа, а также просмотр имеющихся образов и списка контейнеров

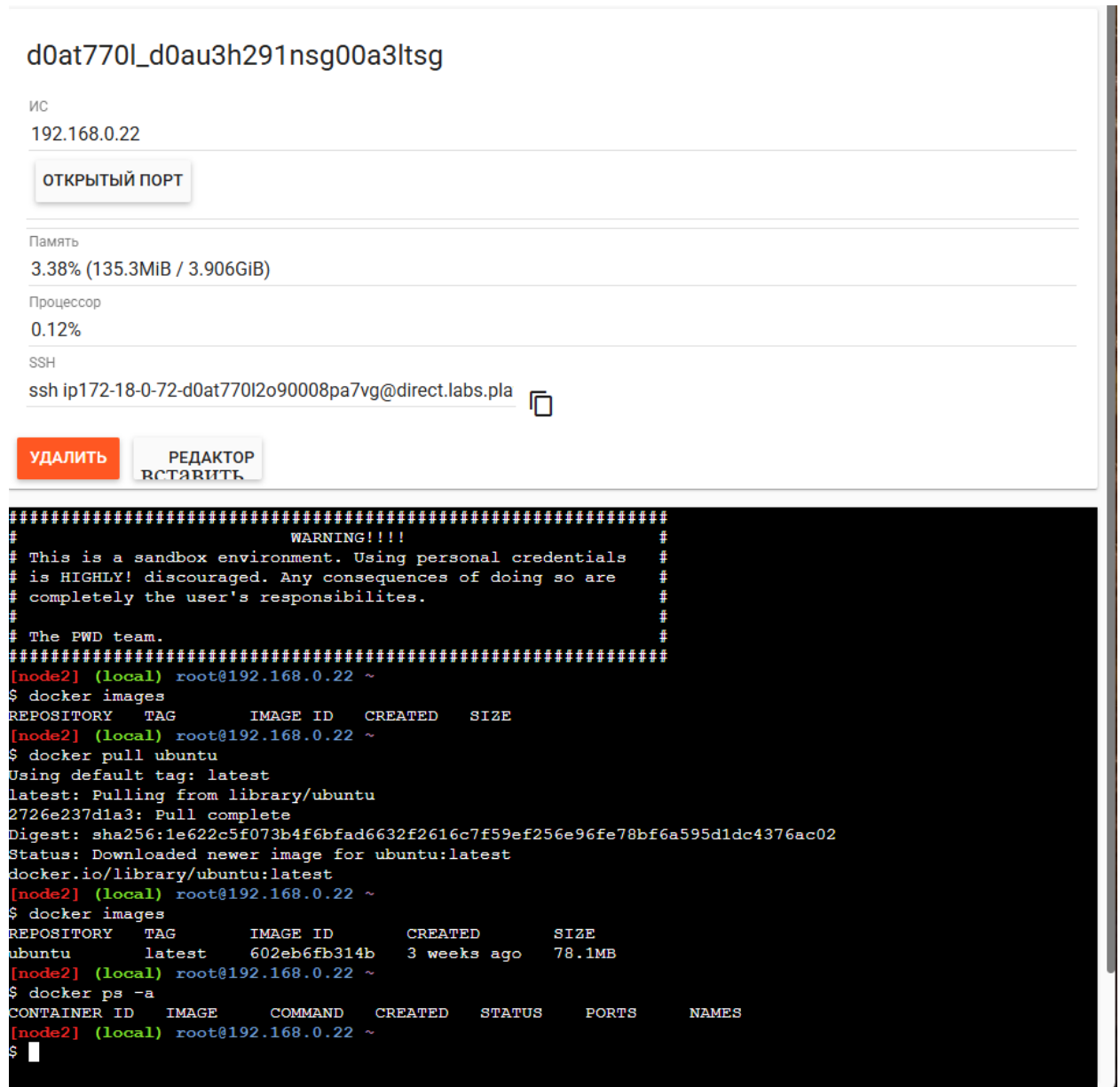


Рисунок 1 – Загрузка образа

### Изоляция

На рисунке 2 результат просмотра информации о хостовой системе и запуск контейнеров

```

[node2] (local) root@192.168.0.22 ~
$ hostname
node2
[node2] (local) root@192.168.0.22 ~
$ hostname
node2
[node2] (local) root@192.168.0.22 ~
$ docker run ubuntu hostname
63fb4315bdc9
[node2] (local) root@192.168.0.22 ~
$ docker run ubuntu hostname
2b371f8958d9
[node2] (local) root@192.168.0.22 ~
$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS          NAMES
2b371f8958d9   ubuntu   "hostname"              9 seconds ago    Exited (0) 8 seconds ago           intelligen
t hermann
63fb4315bdc9   ubuntu   "hostname"              16 seconds ago   Exited (0) 15 seconds ago           optimistic
archimedes
[node2] (local) root@192.168.0.22 ~
$ docker run -it ubuntu bash
root@24e46d6fede8:/# exit
exit
[node2] (local) root@192.168.0.22 ~
$

```

Рисунок 2 – Запуск bash в контейнере

## Работа с портами

На рисунках 3–6 изображена загрузка образа python, проброс портов, запуск сервера

```

[node2] (local) root@192.168.0.22 ~
$ docker pull python
Using default tag: latest
latest: Pulling from library/python
cf05a52c0235: Pull complete
63964a8518f5: Pull complete
ca513cad200b: Pull complete
c187b51b626e: Pull complete
776493ee5e4c: Pull complete
39ca2d92e129: Pull complete
ab89b3116421: Pull complete
Digest: sha256:884da97271696864c2eca77c6362b1c501196d6377115c81bb9dd8d538033ec3
Status: Downloaded newer image for python:latest
docker.io/library/python:latest
[node2] (local) root@192.168.0.22 ~
$ 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.
[node2] (local) root@192.168.0.22 ~
$ 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/) ...
172.18.0.1 - - [03/May/2025 09:36:13] code 400, message Bad request version ('\\x00\\x12\\x00\\x10\\x04
\\x03\\x08\\x04\\x04\\x01\\x05\\x03\\x08\\x05\\x05\\x01\\x08\\x06\\x06\\x01Dí\\x00\\x05\\x00\\x03\\x02h
2\\x00\\x1b\\x00\\x03\\x02\\x00\\x02\\x00\\x12\\x00\\x00\\x00\\x00\\x02\\x01\\x01\\x00#\\x00\\x00\\x00
+\\x00\\x07\\x06\\x8a\\x8a\\x03\\x04\\x03\\x03\\x00')
172.18.0.1 - - [03/May/2025 09:36:13] "\\x16\\x03\\x01\\x07Q\\x01\\x00\\x07M\\x03\\x03\\x98tOô÷\\x02â4bß\\x91ØiÈ>\\x
01^>\\x15a°*nB0ÄVßiu?\\x8a #QÄfÖ%\\x06\\x99°E\\x11f°\\x0bÇ:m\\x19M\\x96NK"\\x84Èñ\\x1d;4«p\\x81\\x00 \\x9a\\x9a\\x13\\x
01\\x13\\x02\\x13\\x03Ä+Ä/Ä,Ä0i@i~Ä\\x13Ä\\x14\\x00\\x9c\\x00\\x9d\\x00/\\x005\\x01\\x00\\x06aJJ\\x00\\x00\\x00\\x0d\\x00\\x
12\\x00\\x10\\x04\\x03\\x08\\x04\\x04\\x01\\x05\\x03\\x08\\x05\\x05\\x01\\x08\\x06\\x06\\x01Dí\\x00\\x05\\x00\\x03\\x02h2\\x00\\
x1b\\x00\\x03\\x02\\x00\\x02\\x00\\x12\\x00\\x00\\x00-\\x00\\x02\\x01\\x01\\x00#\\x00\\x00\\x00+\\x00\\x07\\x06\\x8a\\x8a\\x03\\
x04\\x03\\x03\\x00" 400 -
172.18.0.1 - - [03/May/2025 09:36:14] code 400, message Bad request version ('°c^EÜ{ðQ\\x8fEwG@t)R*\\x0e/í\\x10')
172.18.0.1 - - [03/May/2025 09:36:14] "\\x16\\x03\\x01\\x07\\x11\\x01\\x00\\x07\\x0d\\x03\\x03\\x81\\ä\\x042\\x15\\x01\\
x98\\x1ffr/hçæË6r\\x9aE-\\x9bwüÖ÷i`«çÊ\\ °c^EÜ{ðQ\\x8fEwG@t)R*\\x0e/í\\x10" 400 -

```

Рисунок 3 – загрузка образа python и запуск сервера с портом 8000

## Directory listing for /

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

Рисунок 4 - запуск сервера с портом 8000

```
[node2] (local) root@192.168.0.22 ~
$ 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/) ...
172.18.0.1 - - [03/May/2025 09:37:48] code 400, message Bad HTTP/0.9 request type ('\\x16\\x03\\x01\\x071\\x01\\x00\\x07-\\x03\\x03\\x84·ø\\x9bó\\x00\\x8e*>\\x98íVà\\x9aX*\\x14')
172.18.0.1 - - [03/May/2025 09:37:48] "\\x16\\x03\\x01\\x071\\x01\\x00\\x07-\\x03\\x03\\x84·ø\\x9bó\\x00\\x8e*>\\x98íVà\\x9aX*\\x14 )pp :\\x1bú\\x07\\x11\\x92n" 400 -
172.18.0.1 - - [03/May/2025 09:37:49] code 400, message Bad request version ('»')
172.18.0.1 - - [03/May/2025 09:37:49] "\\x16\\x03\\x01\\x06ñ\\x01\\x00\\x06í\\x03\\x03\\x930Ae#4ãQ38ò\\x10*6Ln\\x10\\x96íaJK\\x81\\x1a\\x02B\\x91hY\\x13\\x0b° ÉÕ$E=kPmý)\\x86\\x8fu`0\\x87"\\x88-ãÁ\\x00°çûÉNEzDL\\x15\\x00 \\x8a\\x8a\\x13\\x01\\x13\\x02\\x13\\x03À+À/À,À0İ@İ"À\\x13À\\x14\\x00\\x9c\\x00\\x9d\\x00/\\x005\\x01\\x00\\x06\\x84jj\\x00\\x00\\x00\\x0b\\x00\\x02\\x01\\x00\\x00\\x1b\\x00\\x03\\x02\\x00\\x02\\x00-\\x00\\x02\\x01\\x01DÍ\\x00\\x05\\x00\\x03\\x02h2\\x00\\x0d\\x00\\x12\\x00\\x10\\x04\\x03\\x08\\x04\\x04\\x01\\x05\\x03\\x08\\x05\\x05\\x01\\x08\\x06\\x06\\x01\\x003\\x04i\\x04i*^\\x00\\x01\\x00\\x11i\\x04ÀC\\x83\\x05!\\x09\\x192\\x01\\x0b\\x88Y\\x93«\\x09!p\\x04İ\\x96J! |0)İ)\\x9b\\x8a\\x16c`\\x8cç\\x05NØñÀK\\x88\\x14çgN*İ\\x99\\x06\\x0b\\x07·ð\\x9ep[STóZÀÀ,yL\\x98ó1\\x19\\x9f@>í²\\x9a!6É²Z\\x0b8\\x9aX\\x12\\x87Á(-\\x9aãÀèi\\x90wü"gwÅ5Rİ[\\x99ÀÈ·O_C~Se"ç#~Ëv0MD*\\x110ç\\x10\\x9b\\x13\\x10\\x87=\\x80\\x92\\x8bù \\x85-\\x82\\x94pã@wİR4,\\x82\\x8a"\\x09\\x02ôÀ\\x89R\\x1b á\\x16ã;²ÜÀ\\x07\\x0fU\\x87- »" 400 -
172.18.0.1 - - [03/May/2025 09:37:51] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [03/May/2025 09:37:52] code 404, message File not found
172.18.0.1 - - [03/May/2025 09:37:52] "GET /favicon.ico HTTP/1.1" 404 -
^C
Keyboard interrupt received, exiting.
[node2] (local) root@192.168.0.22 ~
$
```

Рисунок 5 - запуск сервера с портом 8888

## Directory listing for /

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

Рисунок 6 - запуск сервера с портом 8888

## Именованные контейнеры, остановка и удаление

На рисунке 7 изображен результат выполнения команд. Запущен контейнер в фоновом режиме, проверка на работоспособность, просмотр логов, остановка контейнера, повторный запуск, удаление контейнера

```
[node2] (local) root@192.168.0.22 ~
$ docker run -p8000:8000 --name pyserver -d python python -m http.server
8d6bd16ca7dcbbe5c21117fcc6421075d4fdeb5675e6b3dc81e7ac5fc8d2c63a
[node2] (local) root@192.168.0.22 ~
$ docker ps | grep pyserver
8d6bd16ca7dc  python  "python -m http.serv..." 21 seconds ago Up 21 seconds 0.0.0.0:8000->8000/t
cp pyserver
[node2] (local) root@192.168.0.22 ~
$ docker logs pyserver
[node2] (local) root@192.168.0.22 ~
$ docker stop pyserver
pyserver
[node2] (local) root@192.168.0.22 ~
$ docker run -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 conta
iner "8d6bd16ca7dcbbe5c21117fcc6421075d4fdeb5675e6b3dc81e7ac5fc8d2c63a". You have to remove (or rename)
that container to be able to reuse that name.
See 'docker run --help'.
[node2] (local) root@192.168.0.22 ~
$ docker rm pyserver
pyserver
[node2] (local) root@192.168.0.22 ~
$ docker run --rm -p8000:8000 --name pyserver -d python python -m http.server
8fe7299a5e0441e14233eb4227e18df38960421bc99bed26ddf30ddd8253619d
[node2] (local) root@192.168.0.22 ~
$
```

Рисунок 7 – Остановка и удаление контейнера

## Постоянное хранение данных

На рисунках 8–10 изображен запуск контейнера с корневой директорией mnt, вход в контейнер, запись информации и проверка в браузере

```
[node2] (local) root@192.168.0.22 ~
$ docker run -p8000:8000 --rm --name pyserver -d python python -m http.server -d /mnt
963feb3dcd54a9c3f3a0c9f6b88b4bc6949d75c92a1f8b1b5b40393c180c30eb
[node2] (local) root@192.168.0.22 ~
$ docker exec -it pyserver bash
root@963feb3dcd54:/# cd /mnt
root@963feb3dcd54:/mnt# echo "Hello World!">hi.txt
root@963feb3dcd54:/mnt# exit
exit
[node2] (local) root@192.168.0.22 ~
$ docker stop pyserver
pyserver
[node2] (local) root@192.168.0.22 ~
$
```

Рисунок 8 – Запуск контейнера и запись информации

## Directory listing for /

- [hi.txt](#)

Рисунок 9 – Файл, в который была записана информация

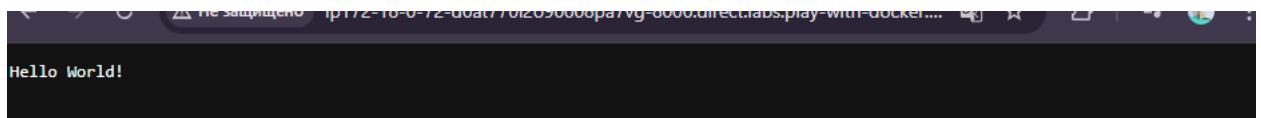


Рисунок 10 – Информация в файле hi.txt

## Тема

На рисунках 11–12 изображено создание тома, создание контейнера с примонтированным томом, запись информации в файл и проверка в браузере

```
[node2] (local) root@192.168.0.22 ~
$ mkdir myfiles
[node2] (local) root@192.168.0.22 ~
$ docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python python -m http.server -d /mnt
19a76e31ccf7808e0cb83c62fdb4046cba769c334d62a2b4373d9427d44354f8
[node2] (local) root@192.168.0.22 ~
$ docker exec -it pyserver bash
root@19a76e31ccf7:/# cd /mnt
root@19a76e31ccf7:/mnt# echo "Hello World2" > hi.txt
root@19a76e31ccf7:/mnt# exit
exit
[node2] (local) root@192.168.0.22 ~
$ docker inspect -f "{{.json .Mounts}}" pyserver
[{"Type":"bind","Source":"/root/myfiles","Destination":"/mnt","Mode":"","RW":true,"Propagation":"rprivate"}]
[node2] (local) root@192.168.0.22 ~
$ docker stop pyserver
pyserver
[node2] (local) root@192.168.0.22 ~
$
```

Рисунок 11 – Создание файла и запись информации

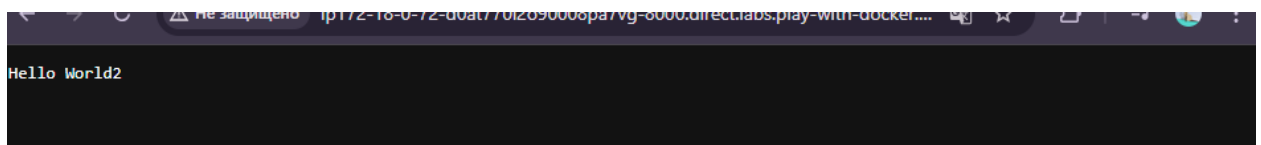


Рисунок 12 – проверка в браузере



## Монтирование директорий и файлов

На рисунке 13 изображено создание директории и файла, запуск контейнера, монтирование директории и файла

```
[node2] (local) root@192.168.0.22 ~
$ mkdir myfiles2
[node2] (local) root@192.168.0.22 ~
$ touch myfiles2/host.txt
[node2] (local) root@192.168.0.22 ~
$ docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles2:/mnt python python -m http.server -
d /mnt
d722fb8792f5efe88acdefc8dd4628e0df8de8bcd8b0dd21730668c3de2708be5
[node2] (local) root@192.168.0.22 ~
$ docker exec -it pyserver bash
root@d722fb8792f5:/# cd /mnt
root@d722fb8792f5:/mnt# ls
host.txt
root@d722fb8792f5:/mnt# echo "Hello World3" >hi.txt
root@d722fb8792f5:/mnt# exit
exit
[node2] (local) root@192.168.0.22 ~
$ ls myfiles2
hi.txt      host.txt
[node2] (local) root@192.168.0.22 ~
$ docker stop pyserver
pyserver
[node2] (local) root@192.168.0.22 ~
$
```

Рисунок 13 – Монтирование директории и файла

## Переменные окружения

На рисунке 14 изображена передача в контейнер переменной окружения MIREA со значением “ONE LOVE”

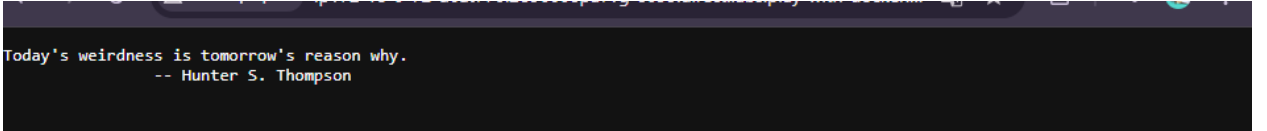
```
[node2] (local) root@192.168.0.22 ~
$ docker run -it --rm -e MIREA="ONE LOVE" ubuntu env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=e97c56ea374f
TERM=xterm
MIREA=ONE LOVE
HOME=/root
[node2] (local) root@192.168.0.22 ~
$
```

Рисунок 14 – Передача переменной окружения MIREA

## Dockerfile

На рисунках 15–20 изображена сборка образа и проверка в браузере





```
Today's weirdness is tomorrow's reason why.
-- Hunter S. Thompson
```

Рисунок 18 – Содержимое сгенерированного файла

## Directory listing for /data/

- [example.txt](#)

Рисунок 19 – Содержимое файла data



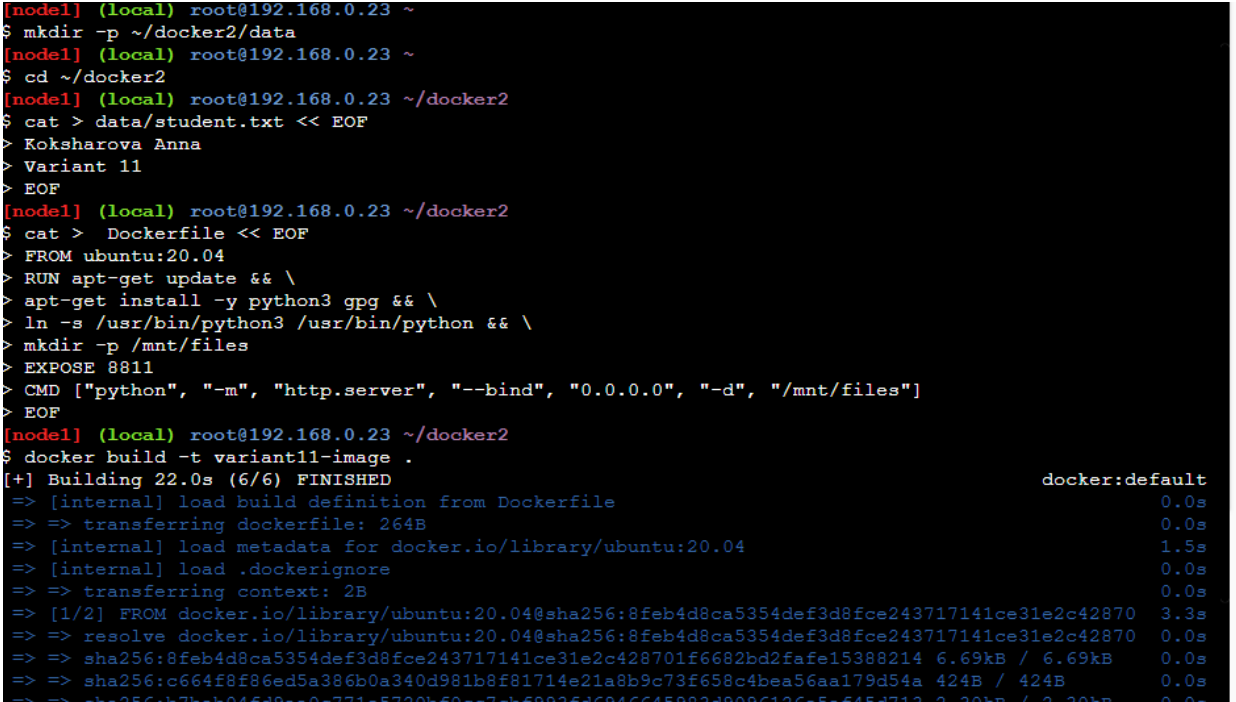
```
Test
```

Рисунок 20 – Содержимое файла example.txt

## Индивидуальный вариант

Необходимо использовать базовый образ `ubuntu:20.04`. Примонтировать директорию `data` в директорию `/mnt/files/` в контейнере. Запустить веб-сервер, отображающий содержимое `/mnt/files`, в хостовой системе должен открываться на порту (8800 + номер варианта). Например, для 11-го варианта это порт 8811. Установить пакет `gpg`

На рисунках 21-24 изображена распаковка образа, запуск контейнера



```
[node1] (local) root@192.168.0.23 ~
$ mkdir -p ~/docker2/data
[node1] (local) root@192.168.0.23 ~
$ cd ~/docker2
[node1] (local) root@192.168.0.23 ~/docker2
$ cat > data/student.txt << EOF
> Koksharova Anna
> Variant 11
> EOF
[node1] (local) root@192.168.0.23 ~/docker2
$ cat > Dockerfile << EOF
> FROM ubuntu:20.04
> RUN apt-get update && \
> apt-get install -y python3 gpg && \
> ln -s /usr/bin/python3 /usr/bin/python && \
> mkdir -p /mnt/files
> EXPOSE 8811
> CMD ["python", "-m", "http.server", "--bind", "0.0.0.0", "-d", "/mnt/files"]
> EOF
[node1] (local) root@192.168.0.23 ~/docker2
$ docker build -t variant11-image .
[+] Building 22.0s (6/6) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 264B                               0.0s
=> [internal] load metadata for docker.io/library/ubuntu:20.04    1.5s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [1/2] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8f3e243717141ce31e2c42870 3.3s
=> => resolve docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8f3e243717141ce31e2c42870 0.0s
=> => sha256:8feb4d8ca5354def3d8f3e243717141ce31e2c428701f6682bd2fafe15388214 6.69kB / 6.69kB 0.0s
=> => sha256:c664f8f86ed5a386b0a340d981b8f81714e21a8b9c73f658c4bea56aa179d54a 424B / 424B 0.0s
=> => sha256:b7b04f4d9a0c771e5720b60cc7cbf993fd6946645983d8096126e5a545d713 2.30kB / 2.30kB 0.0s
```

Рисунок 21 – Создание директории

8811

Память

9.71% (388.2MiB / 3.906GiB)

Процессор

0.41%

SSH

ssh ip172-18-0-28-d0at770i2o90008pa7vg@direct.labs.pla



УДАЛИТЬ

РЕДАКТОР  
ВСТАВИТЬ

```
[node1] (local) root@192.168.0.23 ~/docker2
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
e287aa1fc14f   variant11-image "python -m http.serv..." About a minute ago Up About a minute 0.0.0.0:8811->8811/tcp
variant11-container
[node1] (local) root@192.168.0.23 ~/docker2
$ docker logs variant11-container
[node1] (local) root@192.168.0.23 ~/docker2
$ docker exec variant11-container ls -la /mnt/files
total 4
drwxr-xr-x 2 root root 25 May  3 12:05 .
drwxr-xr-x 1 root root 19 May  3 12:11 ..
-rw-r--r-- 1 root root 27 May  3 12:05 student.txt
[node1] (local) root@192.168.0.23 ~/docker2
$ docker build -t variant11-image . --no-cache
[+] Building 18.4s (6/6) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 264B                             0.0s
=> [internal] load metadata for docker.io/library/ubuntu:20.04 0.5s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> CACHED [1/2] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e 0.0s
=> [2/2] RUN apt-get update && apt-get install -y python3 gpg && ln -s /usr/bin/python3 /usr/b 16.9s
=> exporting to image                                           1.0s
=> => exporting layers                                           1.0s
=> => writing image sha256:f5d328f793e4104773d505d6398bdfaf1c79d8cb32c7116ee84e2e4f10754a6b 0.0s
=> => naming to docker.io/library/variant11-image               0.0s
[node1] (local) root@192.168.0.23 ~/docker2
$ docker run -d --rm -p8811:8811 -v $(pwd)/data:/mnt/files --name variant11-container variant11-image
docker: Error response from daemon: Conflict. The container name "/variant11-container" is already in use by container "e287aa1fc14f1eb7e5ef0e2e32513a2315c07d75629425e26e9152df324cfab7". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
[node1] (local) root@192.168.0.23 ~/docker2
```

Рисунок 22 – Сборка образа

```
=> [internal] load metadata for docker.io/library/ubuntu:20.04 1.5s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [1/2] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c42870 3.3s
=> => resolve docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c42870 0.0s
=> => sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f6682bd2faf15388214 6.69kB / 6.69kB 0.0s
=> => sha256:c664f8f86ed5a386b0a340d981b8f81714e21a8b9c73f658c4bea56aa179d54a 424B / 424B 0.0s
=> => sha256:b7bab04fd9aa0c771e5720bf0cc7cbf993fd6946645983d9096126e5af45d713 2.30kB / 2.30kB 0.0s
=> => sha256:13b7e930469f6d3575a320709035c6acf6f5485a76abcf03d1b92a64c09c2476 27.51MB / 27.51MB 0.7s
=> => extracting sha256:13b7e930469f6d3575a320709035c6acf6f5485a76abcf03d1b92a64c09c2476 2.3s
=> [2/2] RUN apt-get update && apt-get install -y python3 gpg && ln -s /usr/bin/python3 /usr/b 16.1s
=> exporting to image                                           1.0s
=> => exporting layers                                           0.9s
=> => writing image sha256:06df0f1476f92df95f0b4c024338827b51412b7816183ddb31962350840cc45a 0.0s
=> => naming to docker.io/library/variant11-image               0.0s
[node1] (local) root@192.168.0.23 ~/docker2
$ docker run -d --rm -p8811:8811 -v $(pwd)/data:/mnt/files --name variant11-container variant11-image
e287aa1fc14f1eb7e5ef0e2e32513a2315c07d75629425e26e9152df324cfab7
[node1] (local) root@192.168.0.23 ~/docker2
$
```

Рисунок 23 – Запуск

## **Вывод**

В данной работе было установлены и проверены образы ubuntu и python, изучена изоляция контейнеров, освоена информация с портами, проверено монтирование томов и директорий