

ОГЛАВЛЕНИЕ

1 Постановка задачи (Вариант 21)	1
1 Задание	1
2 Проектирование и реализация	2
2.1 Образы	2
2.2 Изоляция	3
2.3 Работа с портами	4
2.4 Именованные контейнеры, остановка и удаление	5
2.5 Постоянное хранение данных	6
2.5.1 Тома	7
2.5.2 Монтирование директорий и файлов	8
2.6 Переменные окружения	9
2.7 Dockerfile	10
2.8 Индивидуальное задание	11
Выводы	12

1 Постановка задачи (Вариант 21)

1 Задание

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.

2 Проектирование и реализация

2.1 Образы

```
C:\Users\sidor>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
2ab09b027e7f: Pull complete
Digest: sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babbc295a0428a6d21
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

C:\Users\sidor>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu               latest             08d22c0ceb15       2 weeks ago        77.8MB
postgres            latest             680aba37fd0f       6 weeks ago        379MB
redis                latest             2f66aad5324a       6 weeks ago        117MB
docker/getting-started latest             3e4394f6b72f       3 months ago       47MB

C:\Users\sidor>docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
18d212ca85cb       redis              "docker-entrypoint.s..." 4 weeks ago        Exited (255) 4 weeks ago 6379/tcp           gallant_elgamal
2343c9bb5397       redis              "docker-entrypoint.s..." 4 weeks ago        Exited (0) 4 weeks ago 6379/tcp           dazzling_pike
```

Рисунок 1 - команды для работы с образами

2.2 Изоляция

```
C:\Users\sidor>hostname
StasLaptop

C:\Users\sidor>hostname
StasLaptop

C:\Users\sidor>docker run ubuntu hostname
3b8d074e83e6

C:\Users\sidor>docker run ubuntu hostname
4462c5857971
```

Рисунок 2 - Результат выполнения команды hostname

```
C:\Users\sidor>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4462c5857971	ubuntu	"hostname"	34 seconds ago	Exited (0) 33 seconds ago		admiring_elbakyan
3b8d074e83e6	ubuntu	"hostname"	40 seconds ago	Exited (0) 39 seconds ago		recurring_kowalevski

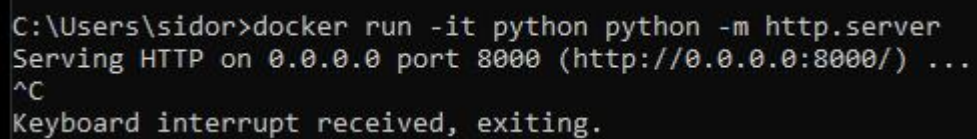
Рисунок 3 - Результат выполнения команды ps -a

```
C:\Users\sidor>docker run ubuntu bash

C:\Users\sidor>docker run -it ubuntu bash
root@88dc707e7f82:/# exit
exit
```

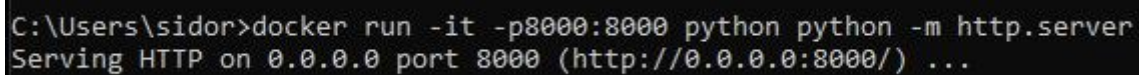
Рисунок 4 - Результат выполнения команды docker run с -it и без

2.3 Работа с портами



```
C:\Users\sidor>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 5 - Результат запуска веб сервера



```
C:\Users\sidor>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/) ...
```

Рисунок 6 - Результат запуска веб сервера с указанием порта

2.4 Именованные контейнеры, остановка и удаление

```
C:\Users\sidor>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS               NAMES
18e0c58a930b   python   "python -m http.serv..." 27 seconds ago Up 26 seconds    0.0.0.0:8000->8000/tcp pyserver

C:\Users\sidor>docker logs pyserver

C:\Users\sidor>docker stop pyserver
pyserver

C:\Users\sidor>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 container "18e0c58a930b". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
```

```
C:\Users\sidor>docker rm pyserver
pyserver
```

Рисунок 7 - Результат запуска контейнера в фоне

2.5 Постоянное хранение данных

```
C:\Users\sidor>docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt
d31055bce3ca92c71e2d51bdf508adf2651b121647705507df8b032d8c959606

C:\Users\sidor>docker exec -it pyserver bash
root@d31055bce3ca:/# cd /mnt && echo "hello world" > hi.txt
root@d31055bce3ca:/mnt#
exit

C:\Users\sidor>docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt
docker: Error response from daemon: Conflict. The container name "/pyserver" is already in use by container "d31055bce3ca92c71e2d51bdf508adf2651b121647705507df8b032d8c959606". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
```

Рисунок 8 - Результат ошибочного запуска контейнера

```
C:\Users\sidor>docker exec -it pyserver bash
root@3fddbbae83c4:/# cd /mnt/
root@3fddbbae83c4:/mnt# ls
```

Рисунок 9 - Данные после перезапуска контейнера

Ответ на вопрос:

В данной команде запуска контейнера с помощью Docker используются следующие флаги:

- `-p8000:8000`: пробрасывает порт 8000 из контейнера в порт 8000 на хосте, таким образом веб-сервер, запущенный в контейнере, будет доступен по адресу `http://localhost:8000`
- `--name pyserver`: задает имя контейнеру, в данном случае - `pyserver`;
- `--rm`: автоматически удаляет контейнер после его остановки;
- `-d`: запускает контейнер в фоновом режиме;
- `python python -m http.server -d /mnt`: команда, которая выполнится внутри контейнера и запускает веб-сервер на порту 8000 и с корневой директорией `"/mnt"`.

2.5.1 Тома

```
C:\Users\sidor>docker run -p8000:8000 --rm --name pyserver -d -v tmpDirect:/mnt python python -m http.server -d /mnt
102b4d9387f3ae5e270b82c8b6a176a5bcf607962033df08c8c2c834009d1aac

C:\Users\sidor>docker exec -it pyserver bash
root@102b4d9387f3:/# cd mnt && echo "hellow world" > hi.txt
root@102b4d9387f3:/mnt# exit
exit

C:\Users\sidor>docker stop pyserver
pyserver

C:\Users\sidor>docker run -p8000:8000 --rm --name pyserver -d -v tmpDirect:/mnt python python -m http.server -d /mnt
8e15c689ead91950bfc7af0836f653a7d84a02d5831e2ea9e032fbf8fddf0e94

C:\Users\sidor>docker exec -it pyserver bash
root@8e15c689ead9:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@8e15c689ead9:/# cd mnt
root@8e15c689ead9:/mnt# ls
hi.txt
root@8e15c689ead9:/mnt#
```

Рисунок 10 - Данные сохраненные монтировкой

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

Рисунок 11 - Местоположение данных

2.5.2 Монтирование директорий и файлов

```
C:\Users\sidor>docker run -p8000:8000 --rm --name pyserver -d -v tmpVolume5:/mnt python python -m http.server -d /mnt  
b3887d9a1992258141c9ba6307daacbd0c11f469f8a800e2cd13df6ed1fdc9e4
```

Рисунок 12 - Запуск с монтированием файла

2.6 Переменные окружения

```
C:\Users\sidor>docker run -it --rm -e MIREA="ONE_LOVE" ubuntu env  
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
HOSTNAME=57435edeef30  
TERM=xterm  
MIREA=ONE_LOVE  
HOME=/root
```

Рисунок 13 - Использование переменных окружения

2.7 Dockerfile

```
C:\Users\sidor\tmpDockerForTRPP>docker build -t mycoolimage .
[+] Building 485.9s (9/9) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 322B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/ubuntu:20.04 2.2s
=> [internal] load build context 0.0s
=> => transferring context: 26B 0.0s
=> CACHED [1/4] FROM docker.io/library/ubuntu:20.04@sha256:24a0df437301598d1a4b62ddf59fa0ed2960150d70d748c84225e6501e9c36b9 0.0s
=> [2/4] RUN apt update && apt install -y python3 fortune && cd /usr/bin && ln -s python3 python 481.8s
=> [3/4] RUN /usr/games/fortune > /mnt/greeting-while-building.txt 0.8s
=> [4/4] ADD ./data /mnt/data 0.1s
=> exporting to image 0.8s
=> => exporting layers 0.8s
=> => writing image sha256:90079b54a19359d2212ced07e71b2cdb31f30b1f212538b9eb31607541e1d645 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\sidor\tmpDockerForTRPP>docker run --rm -it -p8099:80 mycoolimage
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Рисунок 14 - Сборка образа описанного в задании

2.8 Индивидуальное задание

Вариант: 21

Пакет: patch

Ubuntu 20.04

Примонтировать директорию data в директорию /mnt/files/ в контейнере

```
FROM ubuntu:20.04|

RUN apt-get update && \
    apt-get install -y python3 && \
    apt-get install -y patch

RUN mkdir /mnt/files

COPY data/student.txt /mnt/files/student.txt

EXPOSE 8821

WORKDIR /mnt/files

CMD ["python3", "-m", "http.server", "8821"]
```

Рисунок 15 - Содержимое Dockerfile

```
C:\Users\sidor\tmpDirForTRPPDocker>docker build -t myimage .
[+] Building 186.8s (11/11) FINISHED
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 316B                                              0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/ubuntu:20.04                 1.1s
=> [internal] load build context                                                0.1s
=> => transferring context: 61B                                                0.0s
=> CACHED [1/6] FROM docker.io/library/ubuntu:20.04@sha256:24a0df437301598d1a4b62ddf59fa0ed2969150d70d748c84225  0.0s
=> [2/6] RUN                                                                    0.7s
=> [3/6] RUN apt-get update &&          apt-get install -y python3 &&          apt-get install -y patch      182.9s
=> [4/6] RUN mkdir /mnt/files                                                  0.8s
=> [5/6] COPY data/student.txt /mnt/files/student.txt                        0.1s
=> [6/6] WORKDIR /mnt/files                                                    0.1s
=> exporting to image                                                         0.8s
=> => exporting layers                                                         0.8s
=> => writing image sha256:bcf408300b4e7da5d64d6f9d0b2a324ab266ecc6995397b712ed114dc5f9e1f5      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\sidor\tmpDirForTRPPDocker>docker run -v ./data:/mnt/files -p8821:8821 myimage
```

Рисунок 16 - Сборка образа созданного по заданию

Выводы

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