



МИНОБРАЗОВАНИЯ 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 № 4
по дисциплине
«Технология разработки программных приложений»

Тема: «Docker»

Выполнил студент группы ИКБО-13-22


Тринеев П.С

Принял доцент

Петренко А.А.


Практическая работа выполнена

✓ «27» 04 2024 г.


(подпись студента)

«Зачтено»

✓ «27» 04 2024 г.


(подпись руководителя)

Москва 2024

Содержание

Практической работы №4: Docker	2
Раздел 1. Образы.....	2
Раздел 2. Изоляция.	3
Раздел 3. Работа с портами.....	5
Раздел 4. Именованные контейнеры, остановка и удаление.	7
Раздел 5. Постоянное хранение данных.....	8
Раздел 5.1. Тома.....	10
Раздел. 5.2. Монтирование директорий и файлов.....	11
Раздел 6. Переменные окружения.	12
Раздел 7. Dockerfile.....	13
Раздел 8. Индивидуальные задания.....	15
Вывод.....	17

Практической работы №4: Docker

Цель работы: научиться работать с программным обеспечением *Docker*.
Выполнить все шаги из разделов 1-7, а также индивидуальное задание из раздела 8. Сделать вывод о проделанной работе, результаты выразить в отчете.

Раздел 1. Образы.

Посмотрите на имеющиеся образы: `docker images`.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	ca2b0f26964c	3 weeks ago	77.9MB
hello-world	latest	d2c94e258dc	10 months ago	13.3kB

Рис. 1. Имеющиеся образы

Загрузите образ: `docker pull ubuntu` — будет загружен образ `ubuntu:latest` — последняя доступная

```
root@MSI:~# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
bccc10f490ab: Pull complete
Digest: sha256:77906da86b60585ce12215807090eb327e7386c8fab5402369e421f44eff17e
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview ubuntu
```

Рис. 2. Загрузка образа `ubuntu:latest`

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

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	ca2b0f26964c	3 weeks ago	77.9MB
hello-world	latest	d2c94e258dc	10 months ago	13.3kB

Рис. 3. Имеющиеся образы

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

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
83c815a6c293	hello-world	"/hello"	31 minutes ago	Exited (0) 31 minutes ago		blissful_bardeen

Рис. 4. Список контейнеров

Команда `docker ps -a` выводит только запущенные контейнеры.

Раздел 2. Изоляция.

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

```
root@MSI:~# hostname
MSI
root@MSI:~# hostname
MSI
```

Рис. 5. Выполнение 2-х команд `hostname`

Вопрос: одинаковый ли результат получился при разных запусках?

Ответ: да, имя хоста не изменяется.

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

```
root@MSI:~# docker run ubuntu hostname
363d4b13cd54
root@MSI:~# docker run ubuntu hostname
e15740595ab2
```

Рис. 6. Выполнение 2-х команд `docker run ubuntu hostname`

Вопрос: Одинаковый ли результат получился при разных запусках?

Ответ: нет, потому что было запущено 2 изолированных контейнера.

Заново выполните `docker ps -a` — там должны появиться запущенные ранее контейнеры.

```
root@MSI:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS              PORTS          NAMES
e15740595ab2   ubuntu        "hostname"              17 hours ago   Exited (0) 17 hours ago           relaxed_bouman
363d4b13cd54   ubuntu        "hostname"              17 hours ago   Exited (0) 17 hours ago           fervent_matsumoto
83c815a6c293   hello-world   "/hello"                18 hours ago   Exited (0) 18 hours ago           blissful_bardeen
root@MSI:~#
```

Рис. 6. Выполнение команды `docker ps -a`

Запустите `bash` в контейнере: `docker run ubuntu bash`.

```
root@MSI:~# docker run ubuntu bash
root@MSI:~# |
```

Рис. 7. Вывод команды `docker run ubuntu bash`

```
root@MSI:~# docker run -it ubuntu bash
root@ab2cd94b6f70:/# |
```

Рис. 8. Вывод команды `docker run -it ubuntu bash`

Раздел 3. Работа с портами.

Для начала, загрузите образ python командой `docker pull python`.

```
root@MSI:~# docker pull python
Using default tag: latest
latest: Pulling from library/python
71215d55680c: Pull complete
3cb8f9c23302: Pull complete
5f899db30843: Pull complete
567db630df8d: Pull complete
d68cd2123173: Pull complete
63941d09e532: Pull complete
097431623722: Pull complete
09527fa4de8d: Pull complete
Digest: sha256:336461f63f4eb1100e178d5acbfea3d1a5b2a53dea88aa0f9b8482d4d02e981c
Status: Downloaded newer image for python:latest
docker.io/library/python:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview python
```

Рис. 9. Загрузка образа 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`.

```
root@MSI:~# 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.
```

Рис. 10. Запуск сервера

Сервер не будет работать, т.к. не были проброшены порты. Сделаем это.

```
root@MSI:~# 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 - - [22/Mar/2024 08:33:38] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [22/Mar/2024 08:33:38] code 404, message File not found
172.17.0.1 - - [22/Mar/2024 08:33:38] "GET /favicon.ico HTTP/1.1" 404 -
```

Рис. 11. Проброс портов и запуск сервера.

Directory listing for /

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

Рис. 12. Сайт с содержимым корневой директории в контейнере

Раздел 4. Именованные контейнеры, остановка и удаление.

Запустите контейнер в фоне при помощи команды `docker run -p8000:8000 --name pyserver -d python python -m http.server`. Убеждаемся, что контейнер все еще запущен при помощи команды: `docker ps | grep pyserver`. Для просмотра логов контейнера, воспользуйтесь командой `docker logs pyserver`.

```
root@MSI:~# docker run -p8000:8000 --name pyserver -d python python -m http.server
321e2030f6f4c573d91545efe4ea25088e6e38fda94bdfd9e8f34a364f1a94e6
root@MSI:~# docker ps | grep pyserver
321e2030f6f4    python    "python -m http.serv..."   2 minutes ago    Up 2 minutes    0.0.0.0:8000->8000/tcp    pyserver
root@MSI:~# docker logs pyserver
172.17.0.1 - - [22/Mar/2024 08:52:01] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [22/Mar/2024 08:52:02] "GET / HTTP/1.1" 200 -
```

Рис. 13. Запуск и проверка работы контейнера

Останавливаем контейнер при помощи команды: `docker stop pyserver`.
Удаляем контейнер при помощи команды: `docker rm pyserver`.

```
root@MSI:~# docker stop pyserver

pyserver
root@MSI:~#
root@MSI:~# docker rm pyserver

pyserver
```

Рис. 14. Остановка и удаление контейнера

Раздел 5. Постоянное хранение данных.

Запустите контейнер, в котором веб-сервер будет отдавать содержимое директории /mnt: `docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt`, где `-d /mnt` указывает модулю `http.server` какая директория будет корневой для отображения.

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

Ответ: `-p8000:8000`: Этот флаг определяет проброс портов. `--name pyserver`: Этот флаг задает имя для контейнера. `--rm`: Этот флаг указывает Docker на удаление контейнера после его остановки. `-d`: Этот флаг запускает контейнер в фоновом режиме.

Для того, чтобы попасть в уже запущенный контейнер, существует команда `docker exec -it pyserver bash` — вы попадёте в оболочку `bash` в контейнере. Попад в контейнер, выполните команду `cd mnt && echo "hello world" > hi.txt`, а затем выйдите из контейнера, введя команду `exit` или нажав комбинацию клавиш `Ctrl+D`.

```
root@MSI:~# docker exec -it pyserver bash
root@37ff0169705c:/# cd mnt && echo "hello world" > hi.txt
root@37ff0169705c:/mnt# exit
exit
```

Рис. 15. Добавлении файла `hi.txt` с содержимого “hello world”

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

Directory listing for /

- [hi.txt](#)

Рис. 16. Отображение файла `hi.txt` на веб-сайте

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

```
root@MSI:~# docker stop pyserver
pyserver
root@MSI:~# docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt
02dc32f59007f4dd144446aaa98e503b02c791707f3a8aaea4bd5af6590078ac
```

Рис. 17. Остановка и повторный запуск контейнера

Directory listing for /

Рис. 18. Отображение веб-страницы после повторного запуска сервера

Как мы видим, файла `hi.txt` нет. Мы не видим этот файл из-за того, что все изменения происходили в границах первого контейнера.

Раздел 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:

```
root@MSI:~# docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python python -m http.server -d /mnt
4413157b4fa611e89c663c89b72d43114d7e160d7783e2fddea50576b4e5f277
root@MSI:~# |
```

Рис. 19. Создание нового контейнера

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.

Чтобы узнать где хранятся данные, выполните команду `docker inspect -f "{{json .Mounts }}" pyserver`, в поле `Source` будет храниться путь до тома на хостовой машине.

```
root@MSI:~# docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python python -m http.server -d /mnt
4413157b4fa611e89c663c89b72d43114d7e160d7783e2fddea50576b4e5f277
root@MSI:~# docker exec -it pyserver bash
root@4413157b4fa6:/# cd mnt && echo "hello world" > hi.txt
root@MSI:~# docker inspect -f "{{json .Mounts }}" pyserver
[{"Type":"bind","Source":"/root/myfiles","Destination":"/mnt","Mode":"","RW":true,"Propagation":"rprivate"}]
```

Рис. 20. Отображение проделанных команд

Раздел. 5.2. Монтирование директорий и файлов

Создайте директорию: `mkdir myfiles`, в ней создайте файл `host.txt`: `touch myfiles/host.txt`.

Запустите контейнер: `docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python \ python -m http.server -d /mnt`.

```
root@MSI:~# mkdir myfiles
root@MSI:~# touch myfiles/host.txt
root@MSI:~# docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python \
python -m http.server -d /mnt
afeb8766b70c1240d949f636d80e7ff878b3ec9eff0217d6cd86db5bab36c29b
```

Рис. 21. Отображение проделанных команд

Затем, зайдите в контейнер: `docker exec -it pyserver bash`, перейдите в директорию `/mnt` командой `cd /mnt`. Если вывести список файлов командой `ls`, то там будет файл `host.txt`, примонтированный вместе с директорией `myfiles`

```
root@MSI:~# docker exec -it pyserver bash
root@afeb8766b70c:/# cd /mnt
root@afeb8766b70c:/mnt# ls
hi.txt  host.txt
```

Рис. 22. Отображение проделанных команд

Создайте файл `echo "hello world" > hi.txt`, а затем выйдите из контейнера: `exit`. Теперь на хостовой машине в директории `myfiles/` появится файл `hi.txt`. Проверить можно командой `ls myfiles`.

```
root@afeb8766b70c:/mnt# echo "hello world" > hi.txt
root@afeb8766b70c:/mnt# exit
exit
root@MSI:~# ls myfiles
hi.txt  host.txt
```

Рис. 23. Отображение проделанных команд

Раздел 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`

```
root@MSI:~# docker run -it --rm -e MIREA="ONE LOVE" ubuntu env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=a34f37f53bd8
TERM=xterm
MIREA=ONE LOVE
HOME=/root
```

Рис. 24. Отображение проделанных команд

Раздел 7. Dockerfile.

Соберите образ, в который будут установлены дополнительные пакеты, примонтируйте директорию и установите команду запуска. Для этого создаётся файл Dockerfile (без расширения).

```
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"]
```

Рис. 25. Отображение Dockerfile

Соберите образ с тегом mycoolimage с помощью команды `docker build -t mycoolimage .` Точка в конце указывает на текущую директорию, где лежит Dockerfile.

```
root@MSI:~# docker build -t mycoolimage .
[+] Building 123.6s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                 0.0s
=> => transferring dockerfile: 297B                                0.0s
=> [internal] load metadata for docker.io/library/ubuntu:20.04     2.6s
=> [auth] library/ubuntu:pull token for registry-1.docker.io       0.0s
=> [internal] load .dockerignore                                    0.0s
=> => transferring context: 2B                                       0.0s
=> [1/4] FROM docker.io/library/ubuntu:20.04@sha256:80ef4a44043dec4490506e6cc4289eeda2d106a70148b74b5ae91ee670e 25.2s
=> => resolve docker.io/library/ubuntu:20.04@sha256:80ef4a44043dec4490506e6cc4289eeda2d106a70148b74b5ae91ee670e9 0.0s
=> => sha256:80ef4a44043dec4490506e6cc4289eeda2d106a70148b74b5ae91ee670e9c35d 1.13kB / 1.13kB          0.0s
=> => sha256:48c35f3de33487442af224ed4aabc19fd9bfb91ee90e9471d412706b20ba73 424B / 424B            0.0s
=> => sha256:3cfff1c6ff37e0101a08119d0743ba95c7f505d00298f5a169129e4e9086cde9e 2.29kB / 2.29kB          0.0s
=> => sha256:17d0386c2fff30a5b92652bbef2b84639dba9b9f17bdbb819c8d10badd827fdb 27.51MB / 27.51MB       24.3s
=> => extracting sha256:17d0386c2fff30a5b92652bbef2b84639dba9b9f17bdbb819c8d10badd827fdb 0.7s
=> [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 95.1s
=> [3/4] RUN /usr/games/fortune > /mnt/greeting-while-building.txt 0.2s
=> [4/4] ADD ./data /mnt/data                                       0.1s
=> exporting to image                                              0.3s
=> => exporting layers                                              0.3s
=> => writing image sha256:ff683c4965fbafdc71548f8406410162b873bac6221e405ccc15f64196753757 0.0s
=> => naming to docker.io/library/mycoolimage                       0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Рис. 26. Отображение проделанной команды

Запуск производится командой `docker run --rm -it -p8099:80 mycoolimage`, где порт 8099 — порт на хостовой машине.

```
root@MSI:~# 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 - - [22/Mar/2024 15:37:54] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [22/Mar/2024 15:37:54] code 404, message File not found
172.17.0.1 - - [22/Mar/2024 15:37:54] "GET /favicon.ico HTTP/1.1" 404 -
```

Рис. 27. Запуск контейнера

Directory listing for /

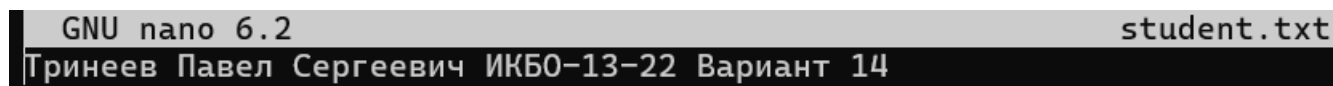
- [hi.txt](#)
 - [host.txt](#)
-

Рис. 28. Отображение веб-сайта

Раздел 8. Индивидуальные задания.

Вариант № 14. Номер по списку – 29.

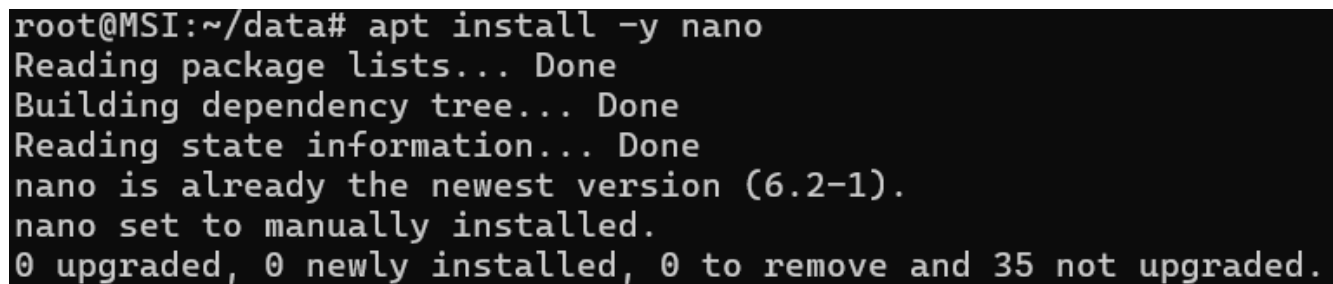
Для монтирования создайте директорию data и в ней файл student.txt, содержащий ФИО, название группы и номер варианта.



```
GNU nano 6.2 student.txt
Тринеев Павел Сергеевич ИКБ0-13-22 Вариант 14
```

Рис. 29. Файл student.txt

Для установки пакетов использовать команду `apt install -y название-пакета`.



```
root@MSI:~/data# apt install -y nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (6.2-1).
nano set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
```

Рис. 30. Установка пакета nano согласно варианту

Необходимо использовать базовый образ `ubuntu:20.10`

Примонтировать файл `data/student.txt` как `/mnt/files/student.txt` в контейнере

Запустить веб-сервер, отображающий содержимое `/mnt/files`, в хостовой системе должен открываться на порту $(8800 + \text{номер варианта})$. Например, для 22-го варианта это порт 8822.


```
# Используем базовый образ Ubuntu 20.10
FROM ubuntu:20.10

# Обновляем список пакетов и устанавливаем пакет nano
RUN apt-get update && \
    apt-get install -y nano && \
    apt-get clean

# Создаем директорию для монтирования и копируем файл student.txt
RUN mkdir -p /mnt/files
COPY data/student.txt /mnt/files/student.txt

# Открываем порт 8829
EXPOSE 8829

# Запускаем веб-сервер, отображающий содержимое /mnt/files
CMD ["python", "-m", "http.server", "--directory", "/mnt/files", "8829"]
```

Рис. 31. Содержимое файла Dockerfile

```
root@MSI:~# docker run -d -p 8829:8829 -v $(pwd)/data/student.txt:/mnt/files/student.txt myimage
```

Рис. 32. Команда запуска контейнера.

Вывод.

При выполнении работы были достигнуты цели научиться работать с программным обеспечением *Docker*. Выполнить все шаги из разделов 1-7, а также индивидуальное задание из раздела 8. Сделать вывод о проделанной работе, результаты выразить в отчете.