

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №16
дисциплины «Основы программной инженерии»

Выполнил:
Матвеев Александр Иванович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Основы Docker.

Цель работы: познакомиться с основами Docker и командами для работы с контейнерами, освоить команды для управления контейнерами и образами Docker, изучить команды мониторинга и управления контейнерами, освоить команды для удаления образов и оптимизации использования дискового пространства, научиться взаимодействовать с работающим контейнером и выполнить ко

Ход работы.

1. Загрузите образ Ubuntu с Docker Hub. Создайте и запустите контейнер на основе этого образа. Выйдите в созданный контейнер и выполните команду ls, чтобы просмотреть файлы внутри контейнера.

```
d:\Docker_test\lab1>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
a48641193673: Pull complete
Digest: sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
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

d:\Docker_test\lab1>
```

Рисунок 1 – Загрузка образа ubuntu

```
d:\Docker_test\lab1>docker run -it ubuntu
root@635cc0c44414:/# |
```

Рисунок 2 – Запуск контейнера на основе образа ubuntu

```
root@635cc0c44414:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@635cc0c44414:/# |
```

Рисунок 3 – Выполнение команды ls внутри созданного контейнера

2. Загрузите образ Nginx с Docker Hub, создайте контейнер на основе этого образа и пробросьте порт 8080 контейнера на порт 80 хоста. Посмотрите

список активных контейнеров и убедитесь, что ваш контейнер работает. Остановите и удалите контейнер.

```
d:\Docker_test\lab1>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
1f7ce2fa46ab: Pull complete
9b16c94bb686: Pull complete
9a59d19f9c5b: Pull complete
9ea27b074f71: Pull complete
c6edf33e2524: Pull complete
84b1ff10387b: Pull complete
517357831967: Pull complete
Digest: sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbb4d62ee
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

Рисунок 4 – Загрузка nginx

```
d:\Docker_test\lab1>docker run -p 8080:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/12/18 10:03:41 [notice] 1#1: using the "epoll" event method
2023/12/18 10:03:41 [notice] 1#1: nginx/1.25.3
2023/12/18 10:03:41 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/12/18 10:03:41 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2023/12/18 10:03:41 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/12/18 10:03:41 [notice] 1#1: start worker processes
2023/12/18 10:03:41 [notice] 1#1: start worker process 30
2023/12/18 10:03:41 [notice] 1#1: start worker process 31
2023/12/18 10:03:41 [notice] 1#1: start worker process 32
2023/12/18 10:03:41 [notice] 1#1: start worker process 33
2023/12/18 10:03:41 [notice] 1#1: start worker process 34
2023/12/18 10:03:41 [notice] 1#1: start worker process 35
2023/12/18 10:03:41 [notice] 1#1: start worker process 36
2023/12/18 10:03:41 [notice] 1#1: start worker process 37
2023/12/18 10:03:41 [notice] 1#1: start worker process 38
2023/12/18 10:03:41 [notice] 1#1: start worker process 39
2023/12/18 10:03:41 [notice] 1#1: start worker process 40
2023/12/18 10:03:41 [notice] 1#1: start worker process 41
172.17.0.1 - - [18/Dec/2023:10:04:00 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.5993.731 YaBrowser/23.11.1.731 Yowser/2.5 Safari/537.36" "-"
172.17.0.1 - - [18/Dec/2023:10:04:00 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.5993.731 YaBrowser/23.11.1.731 Yowser/2.5 Safari/537.36" "-"
2023/12/18 10:04:00 [error] 30#30: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "localhost:8080", referrer: "http://localhost:8080/"
```

Рисунок 5 – Запуск контейнера











<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	 welcome-to-docker aded89866895	welcome-to-docker	Exited (137)	0%	8088:3000	8 minutes ago	   
<input type="checkbox"/>	 flamboyant_perلمان 23e49f4c90d3	nginx	Running	0%	8080:80	2 minutes ago	   

Рисунок 6 – Работающий контейнер

```
D:\Docker_test>docker rm -f 23e49f4c90d3c3b45d42106dde9028fdcf6c3f2bb2ad805f34c58f17d34432f
23e49f4c90d3c3b45d42106dde9028fdcf6c3f2bb2ad805f34c58f17d34432f

D:\Docker_test>|
```

Рисунок 7 – Удаление и остановка контейнера

3. Запустите контейнер с именем “my_container”. Используя команду `docker ps`, убедитесь, что контейнер запущен. Остановите контейнер. Проверьте его статус снова и убедитесь, что он остановлен. Удалите контейнер.

```
D:\Docker_test>docker run --name my_container -d nginx
d6b6eadc43e400b5423889c89033e22798eaedfd80d8f8a4bf3be798685d2a37
```

Рисунок 8 – Запуск контейнера my_container

```
D:\Docker_test>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d6b6eadc43e4	nginx	"/docker-entrypoint..."	28 seconds ago	Up 27 seconds	80/tcp	my_container

Рисунок 9 – Запущенный контейнер

```
D:\Docker_test>docker stop my_container
my_container

D:\Docker_test>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

Рисунок 10 – Остановка контейнера и проверка его работы

```
D:\Docker_test>docker rm my_container
my_container
```

Рисунок 11 – Удаление контейнера

4. Загрузите образы ubuntu и alpine с docker hub. Создайте контейнеры на основе обоих образов. Убедитесь, что контейнеры запущены и работают. Удалите образ ubuntu. Проверьте, что образ ubuntu больше не существует, но образ alpine остался на системе.

```
D:\Docker_test>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
welcome-to-docker	latest	45301e8cb099	24 minutes ago	225MB
ubuntu	latest	174c8c134b2a	5 days ago	77.9MB
alpine	latest	f8c20f8bbcb6	10 days ago	7.38MB

Рисунок 12 – Образы ubuntu и alpine в системе

```
D:\Docker_test>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
58853300fb97	alpine	"/bin/sh"	17 seconds ago	Up 3 seconds		reverent_lederberg
85a6c4270275	ubuntu	"/bin/bash"	30 seconds ago	Up 5 seconds		beautiful_blackburn

Рисунок 13 – Работающие контейнеры

```
D:\Docker_test>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
welcome-to-docker	latest	45301e8cb099	32 minutes ago	225MB
<none>	<none>	174c8c134b2a	5 days ago	77.9MB
alpine	latest	f8c20f8bbcb6	10 days ago	7.38MB

Рисунок 14 – Список образов после удаления

5. Запустите контейнер с именем "my_container" в фоновом режиме. Запустите контейнер с именем "my_container" в фоновом режиме. Запустите контейнер с именем "my_container" в фоновом режиме. Остановите и удалите контейнер.

```
→ docker git:(main) ✗ docker run --name my_container -d ubuntu tail -f /dev/null
```

```
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
005e2837585d: Pull complete
Digest: sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Status: Downloaded newer image for ubuntu:latest
10a3c384e680a32e16b03e9a0042a2c7306919d6d4847c510f049ef1fabbb0e3d
```

Рисунок 15 – Создание контейнера на основе образа ubuntu

```

10: cannot access /app: No such file or directory
→ docker git:(main) ✖ docker exec my_container mkdir app
→ docker git:(main) ✖ docker exec my_container ls -l /app
total 0
→ docker git:(main) ✖

```

Рисунок 16 – Выполнение команды ls -l /app

```

total 0
→ docker git:(main) ✖ docker exec my_container ps aux

```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	2236	1152	?	Ss	06:39	0:00	tail -f /dev/null
root	25	0.0	0.0	6412	2432	?	Rs	06:42	0:00	ps aux

```

→ docker git:(main) ✖

```

Рисунок 17 – Выполнение команды ps aux

```

[→ docker git:(main) ✖ docker stop my_container
my_container
→ docker git:(main) ✖
[→ docker git:(main) ✖ docker rm my_container
my_container

```

Рисунок 18 – Остановка и удаление контейнера

Контрольные вопросы:

1. Команда ``docker pull`` используется для загрузки образа или репозитория из реестра.
2. Синтаксис для загрузки образа с Docker Hub с помощью ``docker pull`` выглядит так: ``docker pull [OPTIONS]``
3. Чтобы просмотреть список всех доступных образов на системе, используйте команду ``docker images``.
4. Команда ``docker images`` по умолчанию отображает образы в формате таблицы. Если вы хотите увидеть больше деталей, вы можете использовать флаг ``-a``.
5. Чтобы создать и запустить контейнер, используйте команду ``docker run``.
6. Чтобы пробросить порт при запуске контейнера, используйте флаг ``-p`` или ``--publish`` с командой ``docker run``. Например, ``docker run -p 8080:80 image_name``.
7. Чтобы изменить имя контейнера при его создании, используйте флаг ``--name`` с командой ``docker run``. Например, ``docker run --name my_container image_name``.
8. Чтобы создать контейнер в фоновом режиме, используйте флаг ``-d`` или ``--detach`` с командой ``docker run``.
9. Команда ``docker ps`` используется для просмотра активных контейнеров на системе.
10. Чтобы отобразить остановленные контейнеры, используйте флаг ``-a`` или ``--all`` с командой ``docker ps``.
11. Чтобы просмотреть список всех контейнеров, включая остановленные, используйте команду ``docker ps -a``.
12. Команда ``docker start`` используется для запуска одного или нескольких остановленных контейнеров.
13. Синтаксис для запуска остановленного контейнера с ``docker start`` выглядит так: ``docker start [OPTIONS] CONTAINER [CONTAINER...]'``.

14. Команда ``docker start`` по умолчанию запускает контейнер в фоновом режиме.

15. Команда ``docker stop`` используется для остановки одного или нескольких работающих контейнеров.

16. Чтобы остановить контейнер по его имени, используйте команду ``docker stop container_name``.

17. Чтобы принудительно остановить контейнер, используйте флаг ``-f`` или ``--force`` с командой ``docker stop``.

18. Команда ``docker rm`` используется для удаления одного или нескольких контейнеров.

19. Чтобы удалить контейнер по его ID, используйте команду ``docker rm container_id``.

20. Чтобы удалить несколько контейнеров сразу, перечислите их ID через пробел после команды ``docker rm``. Например, ``docker rm container_id1 container_id2``.

21. Команда ``docker rmi`` используется для удаления одного или нескольких образов.

22. Чтобы удалить Docker-образ по его имени и тегу, используйте команду ``docker rmi image_name:tag``.

23. Чтобы удалить несколько Docker-образов сразу, перечислите их имена или ID через пробел после команды ``docker rmi``. Например, ``docker rmi image_name1 image_name2``.

24. Чтобы выполнить команду внутри работающего контейнера, используйте команду ``docker exec``. Синтаксис: ``docker exec [OPTIONS] CONTAINER COMMAND [ARG...]'``.

25. Чтобы выполнить команду внутри контейнера в интерактивном режиме, используйте флаг ``-it`` с командой ``docker exec``. Например, ``docker exec -it container_id /bin/bash``.

26. Чтобы выполнить команду с использованием определенного пользователя внутри контейнера, используйте флаг `-u` или `--user` с командой ``docker exec``. Например, ``docker exec -u root container_id command``.

27. Команда ``docker exec`` по умолчанию выполняет команду в фоновом режиме.

28. Чтобы выполнить команду внутри контейнера с именем вместо ID, просто замените ID контейнера на имя в команде ``docker exec``. Например, ``docker exec container_name command``.

29. Чтобы передать аргументы при выполнении команды, просто добавьте их после команды в ``docker exec``. Например, ``docker exec container_id command arg1 arg2``.

30. Чтобы проверить список доступных команд и опций для ``docker exec``, используйте команду ``docker exec --help``.

31. Чтобы передать переменную окружения в контейнер при его запуске, используйте флаг `-e` или `--env` с командой ``docker run``. Например, ``docker run -e VAR=value image_name``.

32. Чтобы запустить контейнер в фоновом режиме, используйте флаг `-d` или `--detach` с командой ``docker run``.

33. Чтобы проверить статус выполнения контейнеров на системе, используйте команду ``docker ps``.

34. Чтобы завершить выполнение контейнера без его удаления, используйте команду ``docker stop container_id``.

35. Чтобы удалить все остановленные контейнеры с системы, используйте команду ``docker container prune``.

36. Опция `-a` или `--all` при использовании ``docker ps`` показывает все контейнеры, включая остановленные.

37. Опция `-q` или `--quiet` при выполнении ``docker ps`` выводит только числовые ID контейнеров.

38. Чтобы принудительно удалить контейнер, используйте флаг `-f` или `--force` с командой ``docker rm``. Например, ``docker rm -f container_id``.

39. Чтобы создать контейнер с базой данных PostgreSQL, вы можете использовать образ `postgres`. Например, `docker run --name some-postgres -e POSTGRES_PASSWORD=mysecretpassword -d postgres`.

40. Ключ `-it` используется для выполнения команды внутри контейнера в интерактивном режиме.

41. Ключ `-u` или `--user` можно использовать для передачи ID пользователя при выполнении команды внутри контейнера.