

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:
Ламская Ксения Вячеславовна
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 _____

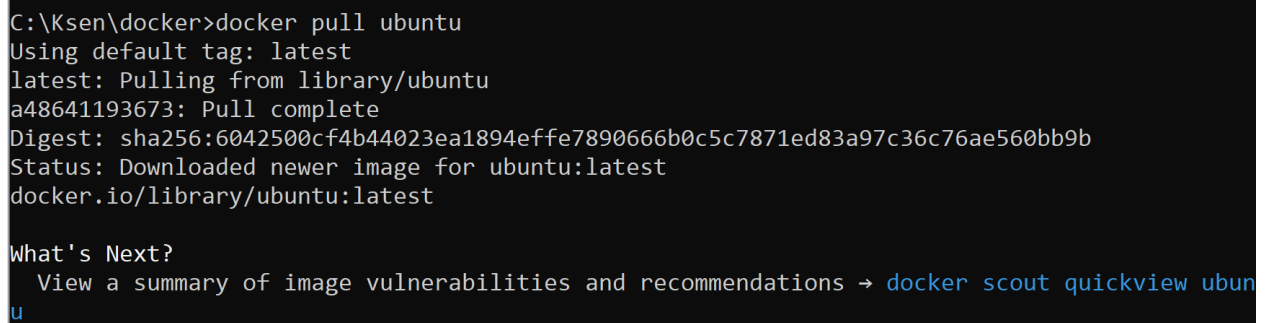
Ставрополь, 2023 г.

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.

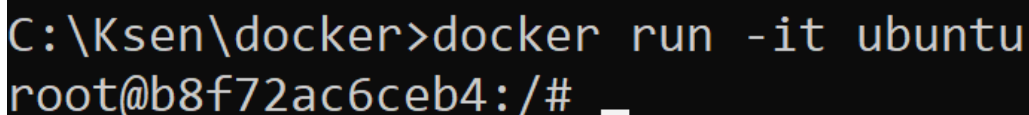
1. Загрузите образ Ubuntu с Docker Hub. Создайте и запустите контейнер на основе этого образа. Выйдите в созданный контейнер и выполните команду `ls`, чтобы просмотреть файлы внутри контейнера.



```
C:\Ksen\docker>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
a48641193673: Pull complete
Digest: sha256:604250cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

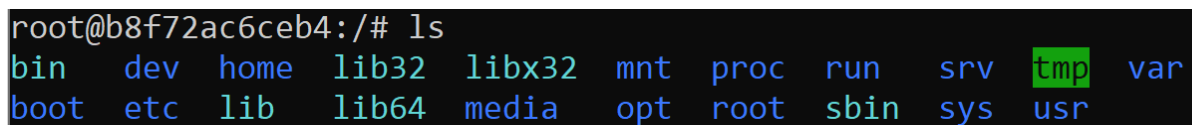
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview ubuntu
```

Рисунок 1.1 – Загрузка образа ubuntu



```
C:\Ksen\docker>docker run -it ubuntu
root@b8f72ac6ceb4:/#
```

Рисунок 1.2 – Запуск контейнера на основе образа ubuntu



```
root@b8f72ac6ceb4:/# ls
bin    dev    home  lib32  libx32  mnt    proc   run    srv    tmp    var
boot  etc    lib   lib64  media   opt    root   sbin   sys    usr
```

Рисунок 1.3 – 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.

```
C:\Ksen\docker>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
af107e978371: Pull complete
336ba1f05c3e: Pull complete
8c37d2ff6efa: Pull complete
51d6357098de: Pull complete
782f1ecce57d: Pull complete
5e99d351b073: Pull complete
7b73345df136: Pull complete
Digest: sha256:2bdc49f2f8ae8d8dc50ed00f2ee56d00385c6f8bc8a8b320d0a294d9e3b49026
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

Рисунок 2.1 – Загрузка nginx

```
C:\Ksen\docker>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
2024/01/07 12:06:29 [notice] 1#1: using the "epoll" event method
2024/01/07 12:06:29 [notice] 1#1: nginx/1.25.3
2024/01/07 12:06:29 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/01/07 12:06:29 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2024/01/07 12:06:29 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/01/07 12:06:29 [notice] 1#1: start worker processes
2024/01/07 12:06:29 [notice] 1#1: start worker process 30
2024/01/07 12:06:29 [notice] 1#1: start worker process 31
2024/01/07 12:06:29 [notice] 1#1: start worker process 32
2024/01/07 12:06:29 [notice] 1#1: start worker process 33
```

Рисунок 2.2 – Запуск контейнера










<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	 awesom b8f72ac6	ubuntu	Exited	0%		12 minutes ago	  
<input type="checkbox"/>	 hardcore 5a61c9f9	nginx	Running	0%	8080:80 	2 minutes ago	  

Рисунок 2.3 – Работающий контейнер

```
C:\Ksen\docker>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                   NAMES
5a61c9f9b705   nginx    "/docker-entrypoint...." 13 minutes ago Up 26 seconds   0.0.0.0:8080->80/tcp    hardcore_austin

C:\Ksen\docker>docker stop 5a61c9f9b705
5a61c9f9b705

C:\Ksen\docker>docker rm 5a61c9f9b705
5a61c9f9b705

C:\Ksen\docker>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    174c8c134b2a   3 weeks ago    77.9MB
nginx         latest    d453dd892d93   2 months ago   187MB

C:\Ksen\docker>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                   NAMES
b8f72ac6ceb4   ubuntu    "/bin/bash"          25 minutes ago Exited (0) 23 minutes ago           awesome_payne
```

Рисунок 2.4 – 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.

```
C:\Ksen\docker>docker run --name my_container -d nginx
fc7f09d07eb35671439490c252a997ecccf04a4efb2506aeaa169267a51cdbf7
```

Рисунок 3.1 – Запуск контейнера my_container

```
C:\Ksen\docker>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                   NAMES
fc7f09d07eb3   nginx    "/docker-entrypoint...." 35 seconds ago Up 25 seconds   80/tcp                my_container
```

Рисунок 3.2 – Запущенный контейнер

```
C:\Ksen\docker>docker stop my_container
my_container

C:\Ksen\docker>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                   NAMES
```

Рисунок 3.3 – Остановка контейнера и проверка его работы

```
C:\Ksen\docker>docker rm my_container
my_container
```

Рисунок 3.4 – 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.

```
C:\Ksen\docker>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    174c8c134b2a   3 weeks ago    77.9MB
alpine        latest    f8c20f8bbcb6   4 weeks ago    7.38MB
nginx         latest    d453dd892d93   2 months ago   187MB
```

Рисунок 4.1 – Образы ubuntu и alpine в системе

```
C:\Ksen\docker>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
ce6cf654a278   ubuntu    "/bin/bash"             4 minutes ago Up 8 seconds             container2
868d01e2185e   alpine    "/bin/sh"               7 minutes ago Up 7 minutes             container1
```

Рисунок 4.2 – Работающие контейнеры

```
C:\Ksen\docker>docker rmi -f 174c8c134b2a
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Deleted: sha256:174c8c134b2a94b5bb0b37d9a2b6ba0663d82d23ebf62bd51f74a2fd457333da
```

Рисунок 4.3 – Удаление образа Ubuntu

```
C:\Ksen\docker>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
alpine        latest    f8c20f8bbcb6   4 weeks ago    7.38MB
nginx         latest    d453dd892d93   2 months ago   187MB
```

Рисунок 4.4 – 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.

```
C:\Ksen\docker>docker run --name my_container -itd ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
a48641193673: Already exists
Digest: sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Status: Downloaded newer image for ubuntu:latest
04b7b9f9a064ca5b5bf8c9f4e866b2f7d450d97a0dba9a4a712a00094ed384c1
```

Рисунок 5.1 – Создание контейнера на основе образа ubuntu

```
C:\Ksen\docker>docker exec my_container ls -l /app
ls: cannot access '/app': No such file or directory
```

Рисунок 5.2 – Выполнение команды ls -l /app

```
C:\Ksen\docker>docker exec my_container ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.9	0.1	4624	3492	pts/0	Ss+	13:06	0:01	/bin/bash
root	15	0.0	0.0	7060	1636	?	Rs	13:09	0:00	ps aux

Рисунок 5.3 – Выполнение команды ps aux

```
C:\Ksen\docker>docker stop my_container
my_container

C:\Ksen\docker>docker rm my_container
my_container
```

Рисунок 5.4 – Остановка и удаление контейнера

Вопросы для самопроверки:

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.