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

_____
(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: основы работы с Docker.

Цель работы: научиться использовать основные команды Docker для управления контейнерами и понимать их назначение.

Порядок выполнения работы

1. Загрузил образ Ubuntu с Docker Hub. Создал и запустил контейнер на основе этого образа. Вошел в созданный контейнер и выполнил команду ls, чтобы посмотреть файлы внутри контейнера.

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>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

C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    174c8c134b2a   2 weeks ago   77.9MB
```

Рисунок 1.1 – Загрузка образа Ubuntu

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker run -it ubuntu
root@439a8233dbeb:/#
```

Рисунок 1.2 – Создание и вход в контейнер

```
root@439a8233dbeb:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

Рисунок 1.3 – Результат выполнения команды ls

2. Загрузил образ Nginx с Docker Hub. Создал контейнер на основе этого образа и пробросил порт 8080 контейнера на порт 80 хоста. Посмотрел список активных контейнеров. Остановил и удалил контейнер.

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\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

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

Рисунок 2.1 – Загрузка образа Nginx

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\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
2023/12/28 19:08:44 [notice] 1#1: using the "epoll" event method
2023/12/28 19:08:44 [notice] 1#1: nginx/1.25.3
2023/12/28 19:08:44 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/12/28 19:08:44 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2023/12/28 19:08:44 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/12/28 19:08:44 [notice] 1#1: start worker processes
2023/12/28 19:08:44 [notice] 1#1: start worker process 29
2023/12/28 19:08:44 [notice] 1#1: start worker process 30
2023/12/28 19:08:44 [notice] 1#1: start worker process 31
2023/12/28 19:08:44 [notice] 1#1: start worker process 32
2023/12/28 19:08:44 [notice] 1#1: start worker process 33
2023/12/28 19:08:44 [notice] 1#1: start worker process 34
2023/12/28 19:08:44 [notice] 1#1: start worker process 35
2023/12/28 19:08:44 [notice] 1#1: start worker process 36
2023/12/28 19:08:44 [notice] 1#1: start worker process 37
2023/12/28 19:08:44 [notice] 1#1: start worker process 38
2023/12/28 19:08:44 [notice] 1#1: start worker process 39
2023/12/28 19:08:44 [notice] 1#1: start worker process 40
```

Рисунок 2.2 – Создание контейнера на основе образа Nginx

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS          PORTS                   NAMES
40dbf79f2d9a   nginx     "/docker-entrypoint.…"   5 minutes ago   Up 15 seconds   0.0.0.0:8080->80/tcp   cool_heisenberg
```

Рисунок 2.3 – Список активных контейнеров

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS          PORTS                   NAMES
40dbf79f2d9a   nginx     "/docker-entrypoint.…"   5 minutes ago   Up 15 seconds   0.0.0.0:8080->80/tcp   cool_heisenberg

C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker stop 40dbf79f2d9a
40dbf79f2d9a
```

Рисунок 2.4 – Остановка контейнера

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker rm 40dbf79f2d9a
40dbf79f2d9a
```

Рисунок 2.5 – Удаление контейнера

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS                     PORTS     NAMES
439a8233dbeb   ubuntu    "/bin/bash"   18 minutes ago   Exited (0) 15 minutes ago             tender_elbakyan
```

Рисунок 2.6 – Удаление контейнера

3. Запустил контейнер с именем «my_container». Используя команду docker ps, убедился, что контейнер запущен. Остановил контейнер. Проверил его статус. Удалил контейнер.

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker run --name my_container -d nginx
d0f753a677726f693d880dbc76bed9aeade5bff622aabc5f222df7ce8ef3c5d1
```

Рисунок 3.1 – Запуск контейнера с именем «my_container» в фоновом режиме

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
d0f753a67772   nginx     "/docker-entrypoint.…"   14 seconds ago   Up 13 seconds   80/tcp    my_container
```

Рисунок 3.2 – Список активных контейнеров

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker stop my_container
my_container
```

Рисунок 3.3 – Остановка контейнера

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Рисунок 3.4 – Проверка статуса контейнера

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker rm my_container
my_container
```

Рисунок 3.5 – Удаление контейнера

4. Загрузил образы Ubuntu и Alpine с Docker Hub. Создал контейнеры на основе обоих образов. Убедился, что контейнеры запущены и работают. Удалил образ Ubuntu. Убедился, что образ Ubuntu больше не существует.

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest

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

C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
661ff4d9561e: Pull complete
Digest: sha256:51b67269f354137895d43f3b3d810bfacd3945438e94dc5ac55fdac340352f48
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest

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

C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>
```

Рисунок 4.1 – Загрузка образов Ubuntu и Alpine

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    174c8c134b2a   2 weeks ago    77.9MB
alpine       latest    f8c20f8bbcb6   2 weeks ago    7.38MB
nginx        latest    d453dd892d93   2 months ago   187MB
```

Рисунок 4.2 – Список образов

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
0a0168c97c8c   alpine    "/bin/sh"     32 seconds ago   Up 21 seconds             clever_chaum
296416fd80cd   ubuntu    "/bin/bash"   41 seconds ago   Up 22 seconds             infallible_moser
```

Рисунок 4.3 – Список активных контейнеров

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    174c8c134b2a   2 weeks ago    77.9MB
alpine       latest    f8c20f8bbcb6   2 weeks ago    7.38MB
nginx        latest    d453dd892d93   2 months ago   187MB

C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker rmi -f 174c8c134b2a
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:6042500cf4b44023ea1894effe7890666b0c5c7871ed83a97c36c76ae560bb9b
Deleted: sha256:174c8c134b2a94b5bb0b37d9a2b6ba0663d82d23ebf62bd51f74a2fd457333da
```

Рисунок 4.4 – Удаление образа Ubuntu

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
alpine       latest    f8c20f8bbcb6   2 weeks ago    7.38MB
nginx        latest    d453dd892d93   2 months ago   187MB
```

Рисунок 4.5 – Список образов после удаления

5. Запустил контейнер с именем «my_container» в фоновом режиме. Используя команду docker exec, выполнил команду ls -l /app внутри контейнера. Выполнил команду ps aux внутри контейнера. Остановил и удалил контейнер.

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker run --name my_container -itd ubuntu
bd86a188b866e18c1313f0b9784efae5209e564796e2109d467a621beeea9a99
```

Рисунок 5.1 – Запустил контейнер с именем «my_container» в фоновом режиме

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\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 внутри контейнера

```
C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\Docker>docker exec my_container ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   4624  3728 pts/0    Ss+  19:53   0:00 /bin/bash
root          15  0.0  0.0   7060  1556 ?        Rs   19:57   0:00 ps aux
```

Рисунок 5.3 – Выполнение команды ps aux внутри контейнера

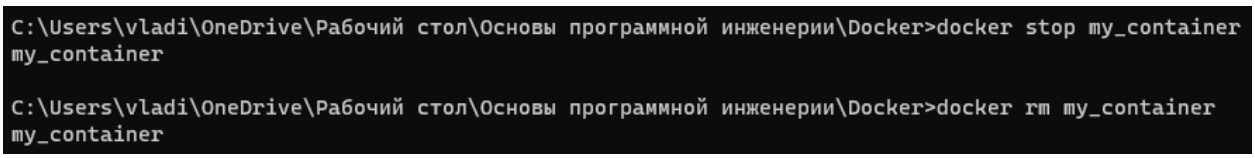

Рисунок 5.4 – Остановка и удаление контейнера

Ответы на контрольные вопросы

1. Команда docker pull используется для загрузки образа из Docker Hub или другого репозитория в локальное хранилище образов.

2. Синтаксис для загрузки образа с Docker Hub с помощью docker pull выглядит следующим образом: docker pull <имя_пользователя>/<имя_образа>:<тег>. Например, чтобы загрузить образ Ubuntu 20.04, необходимо выполнить команду docker pull ubuntu:20.04.

3. Для просмотра списка всех доступных образов на системе с помощью docker images нужно выполнить команду docker images.

4. Ключ -a используется для просмотра образов в формате таблицы с docker images. Чтобы просмотреть все образы, включая промежуточные, необходимо выполнить команду docker images -a.

5. Чтобы создать и запустить контейнер с использованием docker run, нужно выполнить команду docker run <имя_образа>. Например, чтобы создать и запустить контейнер на основе образа Ubuntu 20.04, необходимо выполнить команду docker run ubuntu:20.04.

6. Чтобы пробросить порт при запуске контейнера с docker run, нужно использовать опцию -p. Например, чтобы пробросить порт 80 в контейнере на порт 8080 на хосте, нужно выполнить команду docker run -p 8080:80 <имя_образа>.

7. Чтобы изменить имя контейнера при его создании с помощью docker run, нужно использовать опцию --name. Например, чтобы создать контейнер с именем my_container на основе образа Ubuntu 20.04, нужно выполнить команду docker run --name my_container ubuntu:20.04.

8. Чтобы создать контейнер в фоновом режиме с docker run, нужно использовать опцию -d. Например, чтобы создать и запустить контейнер на основе образа Ubuntu 20.04 в фоновом режиме, нужно выполнить команду docker run -d ubuntu:20.04.

9. Для просмотра активных контейнеров на системе используется команда docker ps.

10. Опция -a используется с docker ps для отображения остановленных контейнеров. Чтобы отобразить все контейнеры (включая остановленные), нужно выполнить команду docker ps -a.

11. Чтобы просмотреть список всех контейнеров, включая остановленные, с docker ps, нужно выполнить команду docker ps -a.

12. Команда docker start используется для запуска остановленного контейнера.

13. Синтаксис для запуска остановленного контейнера с помощью docker start выглядит следующим образом: docker start <идентификатор_контейнера>. Например, чтобы запустить контейнер с идентификатором 123abc, нужно выполнить команду docker start 123abc.

14. Чтобы запустить контейнер в фоновом режиме с docker start, нужно использовать опцию -d. Например, чтобы запустить контейнер с идентификатором 123abc в фоновом режиме, нужно выполнить команду docker start -d 123abc.

15. Команда docker stop используется для остановки работающего контейнера.

16. Чтобы остановить контейнер по его имени с помощью docker stop, нужно выполнить команду docker stop <имя_контейнера>. Например, чтобы остановить контейнер с именем my_container, нужно выполнить команду docker stop my_container.

17. Чтобы принудительно остановить контейнер с помощью docker stop, нужно использовать опцию -f. Например, чтобы принудительно

остановить контейнер с идентификатором 123abc, нужно выполнить команду docker stop -f 123abc.

18. Команда docker rm используется для удаления одного или нескольких контейнеров. При удалении контейнера с помощью docker rm, также удаляются все связанные с ним ресурсы, такие как сетевые интерфейсы и тома данных.

19. Чтобы удалить контейнер по его ID с использованием docker rm, нужно выполнить команду docker rm <идентификатор_контейнера>. Например, чтобы удалить контейнер с идентификатором 123abc, нужно выполнить команду docker rm 123abc.

20. Чтобы удалить несколько контейнеров сразу с помощью docker rm, нужно перечислить их идентификаторы через пробел. Например, чтобы удалить контейнеры с идентификаторами 123abc и 456def, нужно выполнить команду docker rm 123abc 456def.

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

22. Чтобы удалить Docker-образ по его имени и тегу с помощью docker rmi, нужно выполнить команду docker rmi <имя_образа>:<тег>. Например, чтобы удалить образ Ubuntu 20.04, необходимо выполнить команду docker rmi ubuntu:20.04.

23. Чтобы удалить несколько Docker-образов сразу с помощью docker rmi, нужно перечислить их имена и теги через пробел. Например, чтобы удалить образы Ubuntu 20.04 и Alpine 3.13, нужно выполнить команду docker rmi ubuntu:20.04 alpine:3.13.

24. Чтобы выполнить команду внутри работающего контейнера с помощью docker exec, нужно выполнить команду docker exec <идентификатор_контейнера> <команда>. Например, чтобы выполнить команду ls в контейнере с идентификатором 123abc, нужно выполнить команду docker exec 123abc ls.

25. Чтобы выполнить команду внутри контейнера в интерактивном режиме с помощью docker exec, нужно использовать опции -i и -t. Например, чтобы выполнить команду bash в контейнере с идентификатором 123abc, нужно выполнить команду docker exec -it 123abc bash.

26. Чтобы выполнить команду с использованием определенного пользователя внутри контейнера с помощью docker exec, нужно использовать опцию -u. Например, чтобы выполнить команду whoami от имени пользователя user1 в контейнере с идентификатором 123abc, нужно выполнить команду docker exec -u user1 123abc whoami.

27. Ключ -d используется для запуска команды в фоновом режиме с помощью docker exec. Например, чтобы выполнить команду `tail -f /var/log

28. Чтобы выполнить команду внутри контейнера с именем вместо ID с помощью docker exec, нужно использовать опцию --name. Например, чтобы выполнить команду ls в контейнере с именем my_container, нужно выполнить команду docker exec --name my_container ls.

29. Чтобы передать аргументы при выполнении команды с помощью docker exec, нужно добавить их после имени контейнера и команды. Например, чтобы выполнить команду echo с аргументом hello world в контейнере с идентификатором 123abc, нужно выполнить команду docker exec 123abc echo hello world.

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

31. Чтобы передать переменную окружения в контейнер при его запуске, нужно использовать опцию -e. Например, чтобы передать переменную окружения POSTGRES_PASSWORD со значением mysecretpassword при запуске контейнера с базой данных PostgreSQL, нужно выполнить команду docker run -e POSTGRES_PASSWORD=mysecretpassword postgres.

32. Ключ -d используется для запуска контейнера в фоновом режиме с командой docker run. Например, чтобы запустить контейнер на основе образа

Ubuntu 20.04 в фоновом режиме, нужно выполнить команду docker run -d ubuntu:20.04.

33. Чтобы проверить статус выполнения контейнеров на системе с помощью docker ps, нужно выполнить команду docker ps.

34. Чтобы завершить выполнение контейнера без его удаления, нужно выполнить команду docker stop <идентификатор_контейнера>. Например, чтобы остановить контейнер с идентификатором 123abc, нужно выполнить команду docker stop 123abc.

35. Чтобы удалить все остановленные контейнеры с системы, нужно выполнить команду docker container prune.

36. Опция -a при использовании docker ps показывает все контейнеры на системе, включая остановленные.

37. Опция -q при выполнении docker ps показывает только идентификаторы контейнеров.

38. Чтобы принудительно удалить контейнер с флагом -f, нужно выполнить команду docker rm -f <идентификатор_контейнера>. Например, чтобы принудительно удалить контейнер с идентификатором 123abc, нужно выполнить команду docker rm -f 123abc.

39. Для создания контейнера с базой данных PostgreSQL можно использовать Docker-образ postgres и команду docker run. Например, чтобы создать контейнер с именем my_postgres на основе образа postgres и передать переменную окружения POSTGRES_PASSWORD со значением mysecretpassword, нужно выполнить команду docker run --name my_postgres -e POSTGRES_PASSWORD=mysecretpassword -d postgres.

40. Ключ -i используется для выполнения команды внутри контейнера в интерактивном режиме с помощью docker exec. Например, чтобы выполнить команду bash в контейнере с идентификатором 123abc в интерактивном режиме, нужно выполнить команду docker exec -i 123abc bash.

41. Ключ -u можно использовать для передачи ID пользователя при выполнении команды внутри контейнера с помощью docker exec. Например, чтобы