

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение высшего
образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Интеграция и развертывание программного обеспечения с помощью
контейнеров

Лабораторная работа №1-1

Тема:

«Установка и настройка Docker. Работа с контейнерами в Docker»

Выполнил(а): Морозова Валерия АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

Цель работы: освоить процесс установки и настройки Docker, научиться работать с контейнерами и образами Docker.

Задачи:

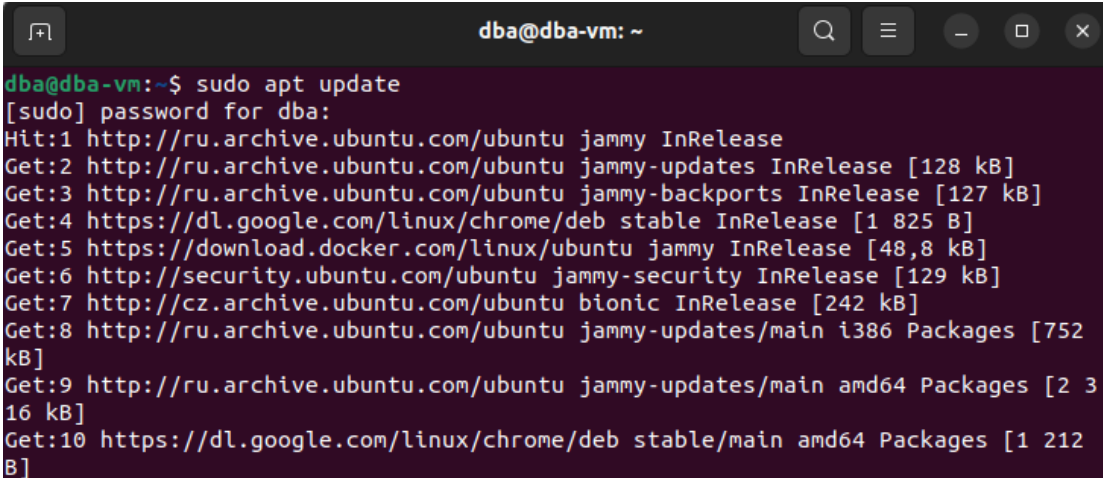
1. Установить Docker на локальный компьютер.
2. Проверить корректность установки Docker.
3. Ознакомиться с основными командами Docker CLI для работы с образами и контейнерами.
4. Выполнить индивидуальное задание.

Вариант 10. Загрузить образ `elasticsearch`, запустить контейнер, настроить маршрутизацию портов и проверить доступность `Elasticsearch` через REST API.

Ход работы

1. Установка Docker.

`sudo apt update`



```
dba@dba-vm:~$ sudo apt update
[sudo] password for dba:
Hit:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 https://dl.google.com/linux/chrome/deb stable InRelease [1 825 B]
Get:5 https://download.docker.com/linux/ubuntu jammy InRelease [48,8 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:7 http://cz.archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:8 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [752 kB]
Get:9 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2 316 kB]
Get:10 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1 212 B]
```

Рисунок 1. Обновление базы данных с доступными для скачивания и установки пакетами программного обеспечения

`sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release`

```
dba@dba-vm:~$ sudo apt install apt-transport-https ca-certificates curl gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
lsb-release set to manually installed.
ca-certificates is already the newest version (20240203~22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.20).
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
apt-transport-https is already the newest version (2.4.13).
0 upgraded, 0 newly installed, 0 to remove and 109 not upgraded.
```

Рисунок 2. Установка пакетов apt-transport-https, ca-certificates, curl, gnupg и lsb-release

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

```
dba@dba-vm:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Рисунок 3. Загрузка и добавление официального GPG-ключа Docker

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
dba@dba-vm:~$ echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

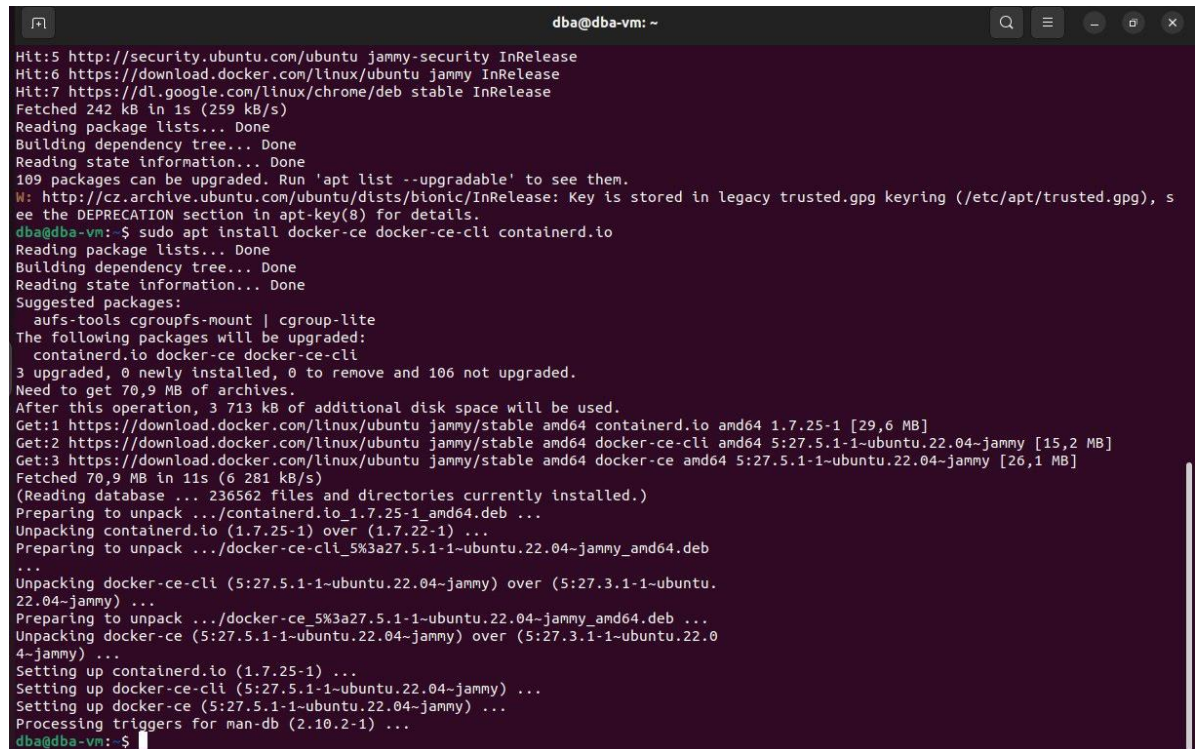
Рисунок 4. Добавление официального репозитория Docker

```
sudo apt update
```

```
dba@dba-vm:~$ sudo apt update
[sudo] password for dba:
Hit:1 http://ru.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ru.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ru.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://cz.archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:7 https://dl.google.com/linux/chrome/deb stable InRelease
Fetched 242 kB in 1s (259 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
109 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: http://cz.archive.ubuntu.com/ubuntu/dists/bionic/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
```

Рисунок 5. Повторное обновление базы данных с доступными для скачивания и установки пакетами программного обеспечения

```
sudo apt install docker-ce docker-ce-cli containerd.io
```



```
dba@dba-vm: ~  
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Hit:6 https://download.docker.com/linux/ubuntu jammy InRelease  
Hit:7 https://dl.google.com/linux/chrome/deb stable InRelease  
Fetched 242 kB in 1s (259 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
109 packages can be upgraded. Run 'apt list --upgradable' to see them.  
W: http://cz.archive.ubuntu.com/ubuntu/dists/bionic/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.  
dba@dba-vm: ~$ sudo apt install docker-ce docker-ce-cli containerd.io  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Suggested packages:  
  aufs-tools cgroupfs-mount | cgroup-lite  
The following packages will be upgraded:  
  containerd.io docker-ce docker-ce-cli  
3 upgraded, 0 newly installed, 0 to remove and 106 not upgraded.  
Need to get 70,9 MB of archives.  
After this operation, 3 713 kB of additional disk space will be used.  
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io amd64 1.7.25-1 [29,6 MB]  
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-cli amd64 5:27.5.1-1-ubuntu.22.04~jammy [15,2 MB]  
Get:3 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd64 5:27.5.1-1-ubuntu.22.04~jammy [26,1 MB]  
Fetched 70,9 MB in 11s (6 281 kB/s)  
(Reading database ... 236562 files and directories currently installed.)  
Preparing to unpack .../containerd.io_1.7.25-1_amd64.deb ...  
Unpacking containerd.io (1.7.25-1) over (1.7.22-1) ...  
Preparing to unpack .../docker-ce-cli_5%3a27.5.1-1-ubuntu.22.04~jammy_amd64.deb ...  
Unpacking docker-ce-cli (5:27.5.1-1-ubuntu.22.04~jammy) over (5:27.3.1-1-ubuntu.22.04~jammy) ...  
Preparing to unpack .../docker-ce_5%3a27.5.1-1-ubuntu.22.04~jammy_amd64.deb ...  
Unpacking docker-ce (5:27.5.1-1-ubuntu.22.04~jammy) over (5:27.3.1-1-ubuntu.22.04~jammy) ...  
Setting up containerd.io (1.7.25-1) ...  
Setting up docker-ce-cli (5:27.5.1-1-ubuntu.22.04~jammy) ...  
Setting up docker-ce (5:27.5.1-1-ubuntu.22.04~jammy) ...  
Processing triggers for man-db (2.10.2-1) ...  
dba@dba-vm: ~$
```

Рисунок 6. Установка последней версии Docker Community Edition (CE) в систему

Для установки требуется три пакета:

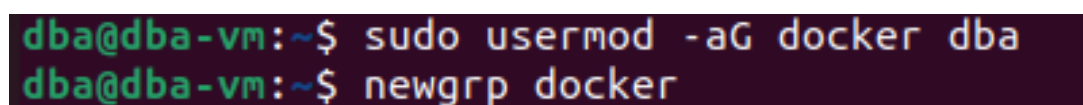
docker-ce — демон Docker Engine.

docker-ce-cli — интерфейс командной строки Docker, с которым будет взаимодействовать пользователь.

containerd.io — среда выполнения контейнера, которая запускает контейнеры.

После завершения установки добавим своего пользователя в группу docker, чтобы избежать необходимости использования sudo при выполнении команд Docker:

```
sudo usermod -aG docker $USER  
newgrp docker
```



```
dba@dba-vm: ~$ sudo usermod -aG docker dba  
dba@dba-vm: ~$ newgrp docker
```

Рисунок 7. Добавление своего пользователя

2. Проверка установки Docker.

Выполните команду `docker --version`, чтобы убедиться, что Docker установлен правильно. Вывод должен содержать информацию о версии Docker.

```
dba@dba-vm:~$ docker --version
Docker version 27.5.1, build 9f9e405
```

Рисунок 8. Версия Docker

Выполните команду `docker run hello-world`, чтобы проверить, что Docker запущен и может загружать образы и запускать контейнеры.

```
dba@dba-vm:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:e0b569a5163a5e6be84e210a2587e7d447e08f87a0e90798363fa44a0464a1e8
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Рисунок 9. Проверка докера на запуск и возможность загружать образы и запускать контейнеры прошла успешно

3. Знакомство с основными командами Docker CLI.

Выполните команду `docker images`, чтобы просмотреть список локальных образов Docker.

```
dba@dba-vm:~$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
hello-world         latest          74cc54e27dc4   3 weeks ago    10.1kB
myapp               latest          3e1b602bae81   3 months ago    649MB
postgres            latest          d57ed788c154   4 months ago    434MB
```

Рисунок 10. Список локальных образов Docker

Выполните команду `docker ps`, чтобы просмотреть список запущенных контейнеров.

```
dba@dba-vm:~$ docker ps
```

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

Рисунок 11. Список запущенных контейнеров

Выполните команду `docker ps -a`, чтобы просмотреть список всех контейнеров, включая остановленные.

```
dba@dba-vm:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
eac4c86be3c0	hello-world	"/hello"	4 minutes ago	Exited (0)
a457dbff5f72	myapp	"python -m uvicorn s..."	3 months ago	Exited (25
5)	0.0.0.0:8000->8000/tcp, :::8000->8000/tcp	myapp	4 weeks ago	
fc7c96c36fe7	postgres	"docker-entrypoint.s..."	3 months ago	Exited (25
5)	0.0.0.0:5432->5432/tcp, :::5432->5432/tcp	mydb	4 weeks ago	

Рисунок 12. Список всех контейнеров, включая остановленные

4. Выполнение индивидуального задания

Вариант 10. Загрузить образ `elasticsearch`, запустить контейнер, настроить маршрутизацию портов и проверить доступность `Elasticsearch` через REST API.

Elasticsearch — это высокомасштабируемая распределённая поисковая система полнотекстового поиска и анализа данных с веб-интерфейсом, REST API и неформализованными JSON-документами, которая работает в режиме реального времени.

Образ `Elasticsearch` может использоваться для централизованного хранения данных, что позволяет обнаруживать ожидаемые и неожиданные результаты.

```
dba@dba-vm:~$ docker pull elasticsearch
Using default tag: latest
Error response from daemon: manifest for elasticsearch:latest not found: manifest unknown: manifest unknown
```

Рисунок 13. Установка образа через команду `pull` (без указания версии)

Как видно на рисунке 13, установка не была выполнена, так как возникла ошибка по причине отсутствия указания последней версии.


```
dba@dba-vm:~$ docker pull elasticsearch:8.11.3
8.11.3: Pulling from library/elasticsearch
527f5363b98e: Pull complete
a88b906db9c1: Pull complete
02840f0dd2d9: Pull complete
4f4fb700ef54: Pull complete
d36bf0362a7b: Pull complete
7cfad007f0ac: Pull complete
4bcb210dd3ca: Pull complete
a1bae0d7353f: Pull complete
2aa937b89ef3: Pull complete
7132da5d7ea5: Pull complete
Digest: sha256:58a3a280935d830215802322e9a0373faaacdfd646477aa7e718939c2f29292a
Status: Downloaded newer image for elasticsearch:8.11.3
docker.io/library/elasticsearch:8.11.3
```

Рисунок 14. Загрузка версии 8.11.3 образа `elasticsearch`

```
dba@dba-vm:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	74cc54e27dc4	3 weeks ago	10.1kB
myapp	latest	3e1b602bae81	3 months ago	649MB
postgres	latest	d57ed788c154	4 months ago	434MB
elasticsearch	8.11.3	ac1eef415132	14 months ago	1.41GB

```
dba@dba-vm:~$
```

Рисунок 15. Список локальных образов Docker

Как видно, среди локальных образов теперь есть `elasticsearch`

Elasticsearch в основном использует два порта: 9200 и 9300. Порт 9200 — это порт HTTP по умолчанию для Elasticsearch, используемый для взаимодействия с клиентами и отправки REST-запросов. С другой стороны, порт 9300 — это транспортный порт по умолчанию, используемый для взаимодействия узлов внутри кластера.

```
dba@dba-vm:~$ docker run --rm --name elasticsearch_container -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" -e "xpack.security.enabled=false" elasticsearch:8.11.3
Feb 19, 2025 2:01:07 PM sun.util.locale.provider.LocaleProviderAdapter <clinit>
WARNING: COMPAT locale provider will be removed in a future release
{"@timestamp":"2025-02-19T14:01:09.417Z","log.level":"INFO","message":"Java vector incubator API enabled; uses preferredBitSize=256","ecs.version":"1.2.0","service.name":"ES_ECS","event.dataset":"elasticsearch.server","process.thread.name":"main","log.logger":"org.apache.lucene.internal.vectorization.PanamaVectorizationProvider","elasticsearch.node.name":"28f2feb845df","elasticsearch.cluster.name":"docker-cluster"}
```

Рисунок 16. Запуск контейнера с маршрутизацией портов 9200 и 9300

Команда `docker run --rm --name elasticsearch_container -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" -e "xpack.security.enabled=false" elasticsearch:8.11.3` запускает контейнер с именем `elasticsearch_container`, сопоставляет порты 9200 и 9300 Elasticsearch с хост-машиной и устанавливает тип обнаружения как одноузловой. Здесь порт 9200 используется для доступа к REST API Elasticsearch, а порт 9300 — для связи между узлами.

```
dba@dba-vm:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
f1edc5a5c70a	elasticsearch:8.11.3	"/bin/tini -- /usr/..."	7 minutes ago	Exited (78) 6 minutes ago	
eac4c86be3c0	hello-world	"/hello"	30 hours ago	Exited (0) 30 hours ago	
a457dbff5f72	myapp	"python -m uvicorn s..."	3 months ago	Exited (255) 4 weeks ago	0.0.0.0:8000->8000/tcp, ::8000->8000/tcp
fc7c96c36fe7	postgres	"docker-entrypoint.s..."	3 months ago	Exited (255) 4 weeks ago	0.0.0.0:5432->5432/tcp, ::5432->5432/tcp

Рисунок 17. Контейнер запущен

Чтобы проверить доступность через REST API и убедиться, что Elasticsearch работает, нужно открыть веб-браузер и перейти по адресу <http://localhost:9200>. В ответе должна быть информация о кластере.

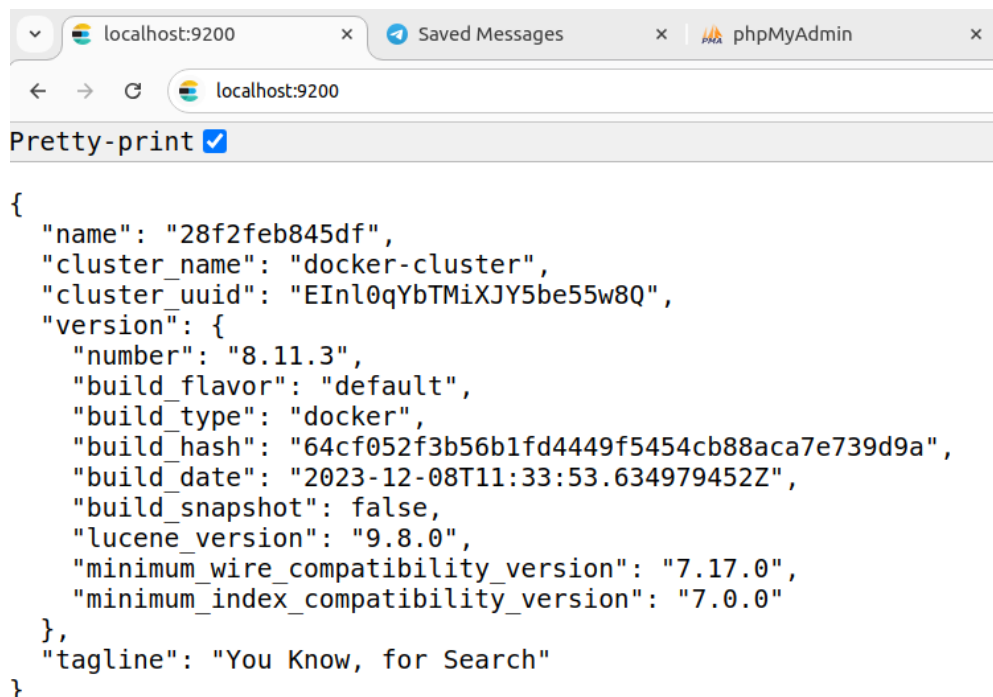


Рисунок 18. Веб-страница с информацией о кластере

```
dba@dba-vm:~$ docker stop elasticsearch
elasticsearch
dba@dba-vm:~$ docker rm elasticsearch
elasticsearch
```

Рисунок 19. Остановка и удаление контейнера

```
dba@dba-vm:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
eac4c86be3c0	hello-world	"/hello"	33 hours ago	Exited (0)
a457dbff5f72	myapp	"python -m uvicorn s..."	3 months ago	Exited (255)
fc7c96c36fe7	postgres	"docker-entrypoint.s..."	3 months ago	Exited (255)

Рисунок 20. Проверка удаления контейнера

Как видно, контейнер `elasticsearch` был успешно удален.

Выводы:

1. Освоила процесс установки и настройки Docker на локальный компьютер.
2. Проверила корректность установки Docker.
3. Ознакомилась и научилась работать с образами и контейнерами с помощью основных команд Docker CLI.
4. В рамках выполнения индивидуального задания:
 - 4.1. изучила и установила образ Elasticsearch последней версии 8.11.3, устранив таким образом ошибку
 - 4.2. запустила контейнер с маршрутизацией портов 9200 и 9300
 - 4.3. проверила доступность через REST API и убедилась, что Elasticsearch работает
 - 4.4. остановила и удалила контейнер

Контрольные вопросы:

1. Что такое Docker и для чего он используется?

Docker — это программная платформа для разработки, доставки и запуска контейнерных приложений. Она позволяет создавать контейнеры, автоматизировать их запуск и развёртывание, управляет жизненным циклом.

Основная идея — создание стандартного и предсказуемого окружения, где приложения могут работать независимо от операционной системы или инфраструктуры.

Docker используется для следующих целей:

- ❖ **Ускорение разработки.** Контейнеры запускаются за секунды, что позволяет разработчикам быстрее тестировать и внедрять изменения.
- ❖ **Упрощение переноса приложений между окружениями.**
- ❖ **Контроль ресурсов.** Docker предоставляет возможность ограничивать доступ контейнеров к ресурсам, таким как процессор, память и дисковое пространство.

❖ **Повышение безопасности.** Изоляция контейнеров и использование проверенных образов из реестров значительно снижают риски утечек данных и уязвимостей.

❖ **Управление микросервисной архитектурой.** Каждый сервис можно обернуть в контейнер. Это обеспечивает как контроль и безопасность для отдельных компонентов системы, так и удобство эксплуатации этих сервисов с технической точки зрения.

❖ **Управление инфраструктурой сложных систем.** Docker обеспечивает удобное управление инфраструктурой сложных систем, особенно в связке с инструментами оркестрации, например Kubernetes.

❖ **Масштабирование.** Контейнеры легко масштабируются, что позволяет эффективно распределять нагрузку и обеспечивать высокую доступность сервисов.

2. Какие преимущества дает использование контейнеров Docker по сравнению с виртуальными машинами?

Виртуальная машина (VM) — это виртуальный компьютер со всеми виртуальными устройствами и виртуальным жёстким диском, на который и устанавливается новая независимая ОС (гостевая ОС) вместе с виртуальными драйверами устройств, управлением памятью и другими компонентами.

При использовании VM появляются дополнительные расходы на эмуляцию виртуального оборудования и запуск гостевой ОС, поддержка и администрирование необходимого окружения для работы приложения. Также при разворачивании большого количества виртуальных машин на сервере объем занимаемого ими места на жёстком диске будет только расти, т.к. для каждой VM требуется место, как минимум, для гостевой ОС и драйверов для виртуальных устройств.

Docker — это ПО для создания приложений на основе контейнеров. Контейнеры и виртуальные машины решают одну задачу, но делают это по-разному. Контейнеры занимают меньше места, т.к. переиспользуют большее количество общих ресурсов хост-системы чем VM, т.к. в отличие от VM,

обеспечивает виртуализацию на уровне ОС, а не аппаратного обеспечения.
Такой подход обеспечивает меньший объем занимаемого места на жёстком диске, быстрое развертывание и более простое масштабирование.

Docker-контейнер даёт более эффективный механизм инкапсуляции приложений, обеспечивая необходимые интерфейсы хост-системы. Данная возможность позволяет контейнерам разделить ядро системы, где каждый из контейнеров работает как отдельный процесс основной ОС, у которого есть своё собственное виртуальное адресное пространство, таким образом данные, принадлежащие разным областям памяти, не могут быть изменены.

3. Что такое образ Docker и как он связан с контейнерами?

Образ — основной элемент, из которого создаются контейнеры. Образ создаётся из Dockerfile, добавленного в проект и представляет собой набор файловых систем (слоёв), наслоённых друг на друга и сгруппированных вместе, доступных только для чтения; максимальное число слоёв равно 127.

В основе каждого образа находится базовый образ, который указывается командой FROM — входная точка при формировании образа Dockerfile. Каждый слой является readonly-слоем и представлен одной командой, модифицирующей файловую систему, записанной в Dockerfile. Данный подход позволяют разным файлам и директориям из разных файловых слоёв прозрачно накладываться, создавая каскадно-объединённую файловую систему. Слои содержат метаданные, позволяющие сохранять сопутствующую информацию о каждом слое во время выполнения и сборки. Каждый слой содержит ссылку на следующий слой, если слой не имеет ссылки, значит это самый верхний слой в образе.

Контейнер — это абстракция на уровне приложения, объединяющая код и зависимости. Контейнеры всегда создаются из образов, добавляя доступный для записи верхний слой и инициализирует различные параметры. Т. к. контейнер имеет свой собственный слой для записи и все изменения сохраняются в этом слое, несколько контейнеров могут совместно использовать доступ к одному и тому же образу. Каждый контейнер можно

настроить через файл в проекте `docker-compose.yml`, задавая различные параметры, такие как имя контейнера, порты, идентификаторы, зависимости между другими контейнерами. Если в настройках не задавать имя контейнера, то Docker каждый раз будет создавать новый контейнер, присваивая ему имя случайным образом.

Когда контейнер запускается из образа, Docker монтирует файловую систему для чтения и записи поверх любых слоев ниже. Именно здесь будут выполняться все процессы.

4. Какие основные команды Docker CLI вы узнали в ходе выполнения лабораторной работы?

docker run — запуск нового контейнера;

docker ps — список запущенных контейнеров;

docker stop — остановка запущенного контейнера;

docker start — запуск остановленного контейнера;

docker rm — удаление остановленного контейнера;

docker pull — загрузка образа из Docker Hub;

docker images — просмотреть список локальных образов Docker;

docker ps — просмотреть список запущенных контейнеров;

docker ps -a — просмотреть список всех контейнеров, включая остановленные;

5. Как можно настроить маршрутизацию портов при запуске контейнера Docker?

Чтобы настроить маршрутизацию портов при запуске контейнера Docker, нужно использовать опцию `--publish` или `-p`. Она позволяет сопоставить порт хостовой машины с портом контейнера. Также можно настроить динамическое сопоставление портов. Вместо указания конкретного порта хоста можно позволить Docker выбрать доступный порт в системе хоста, если опустить порт хоста.