

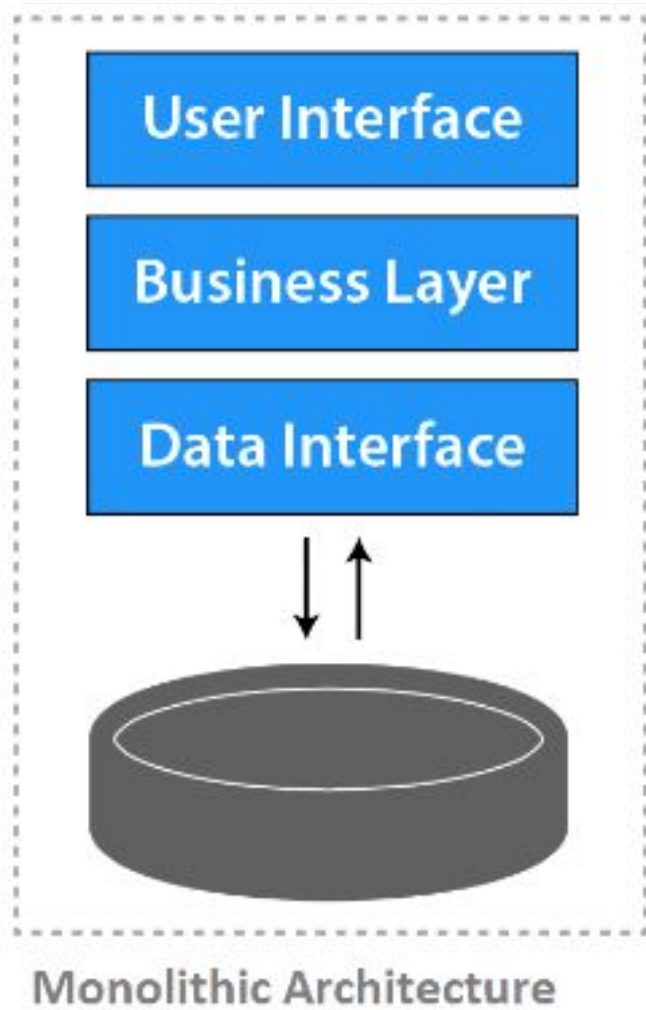
Избранные главы информатики

ЛЕКЦИЯ 2

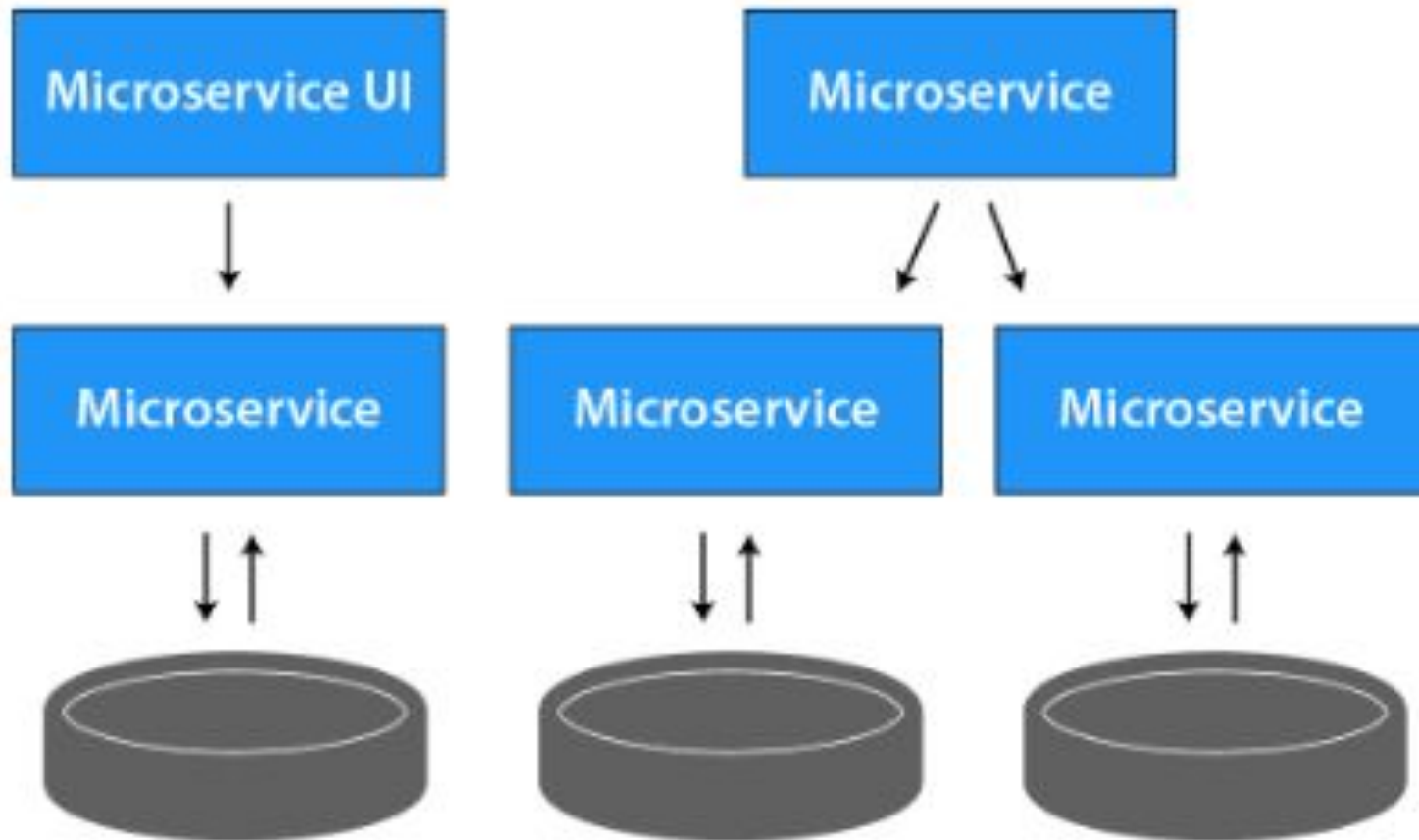
Docker

1. Docker
2. Основные команды
3. Dockerfile
4. Docker Compose
5. Docker Networking

Развертывание приложений - Монолитная архитектура

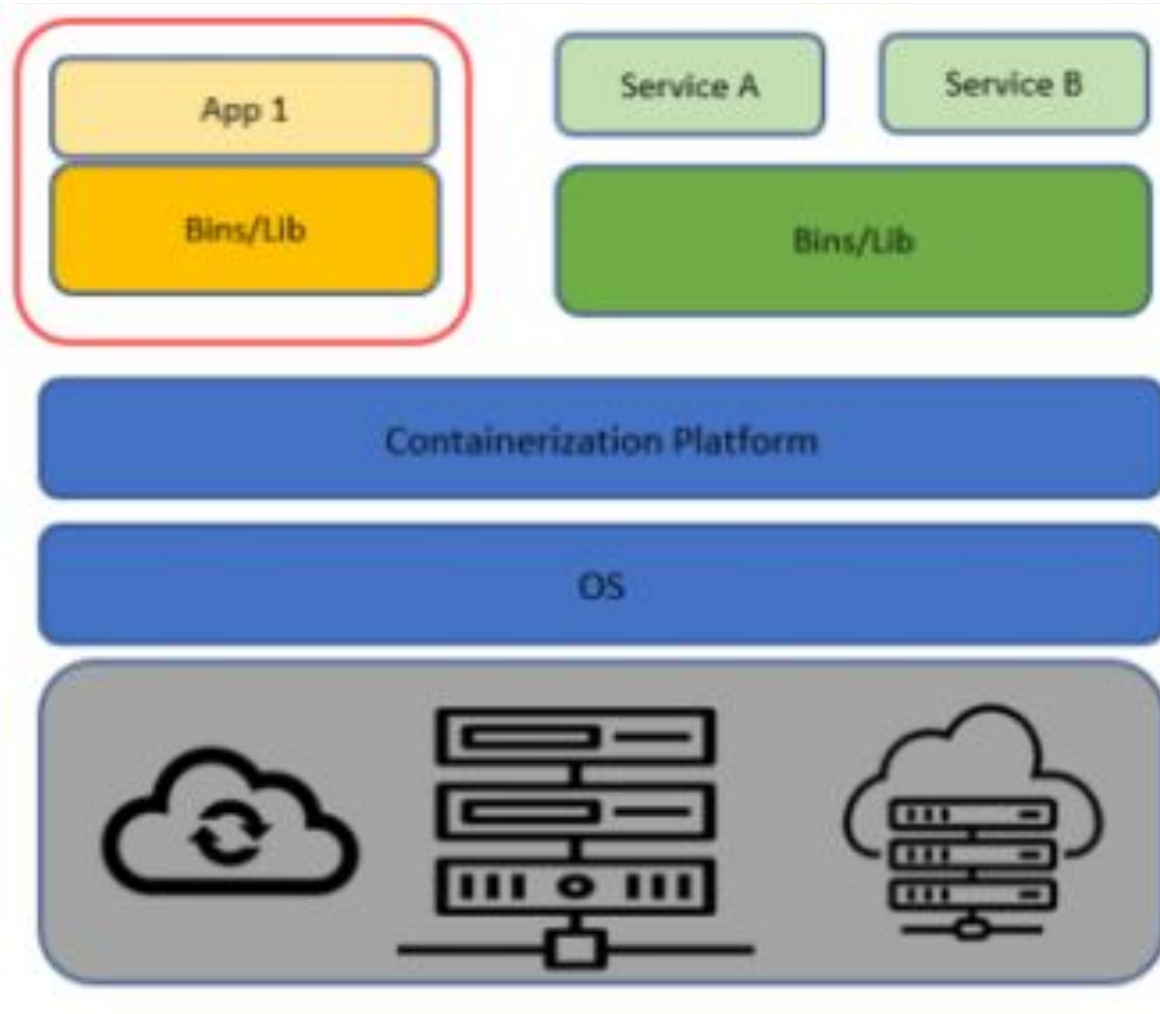


Развертывание приложений - Микросервисная архитектура

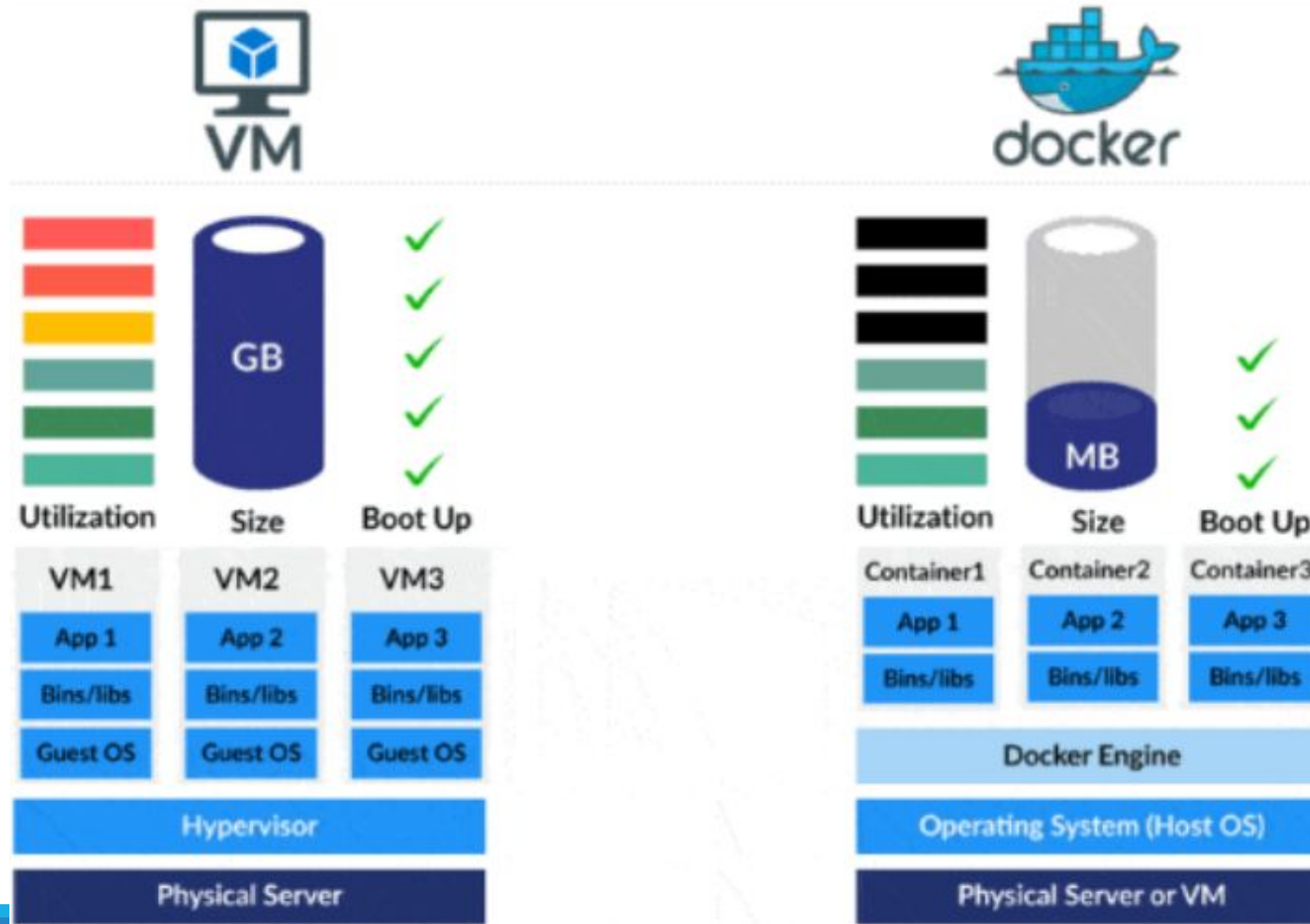


Micro-services Architecture

Развертывание приложений – Контейнерная архитектура



Развертывание приложения – Контейнеры vs Виртуальные машины



Docker

Docker - это платформа для создания, совместного использования и запуска приложений в контейнерах.



Docker - Полезные ссылки

1. <https://www.docker.com/> - официальный сайт (установка/документация)
2. <https://hub.docker.com/>
3. <https://k21academy.com/docker-kubernetes/docker-networking-different-types-of-networking-overview-for-beginners/>

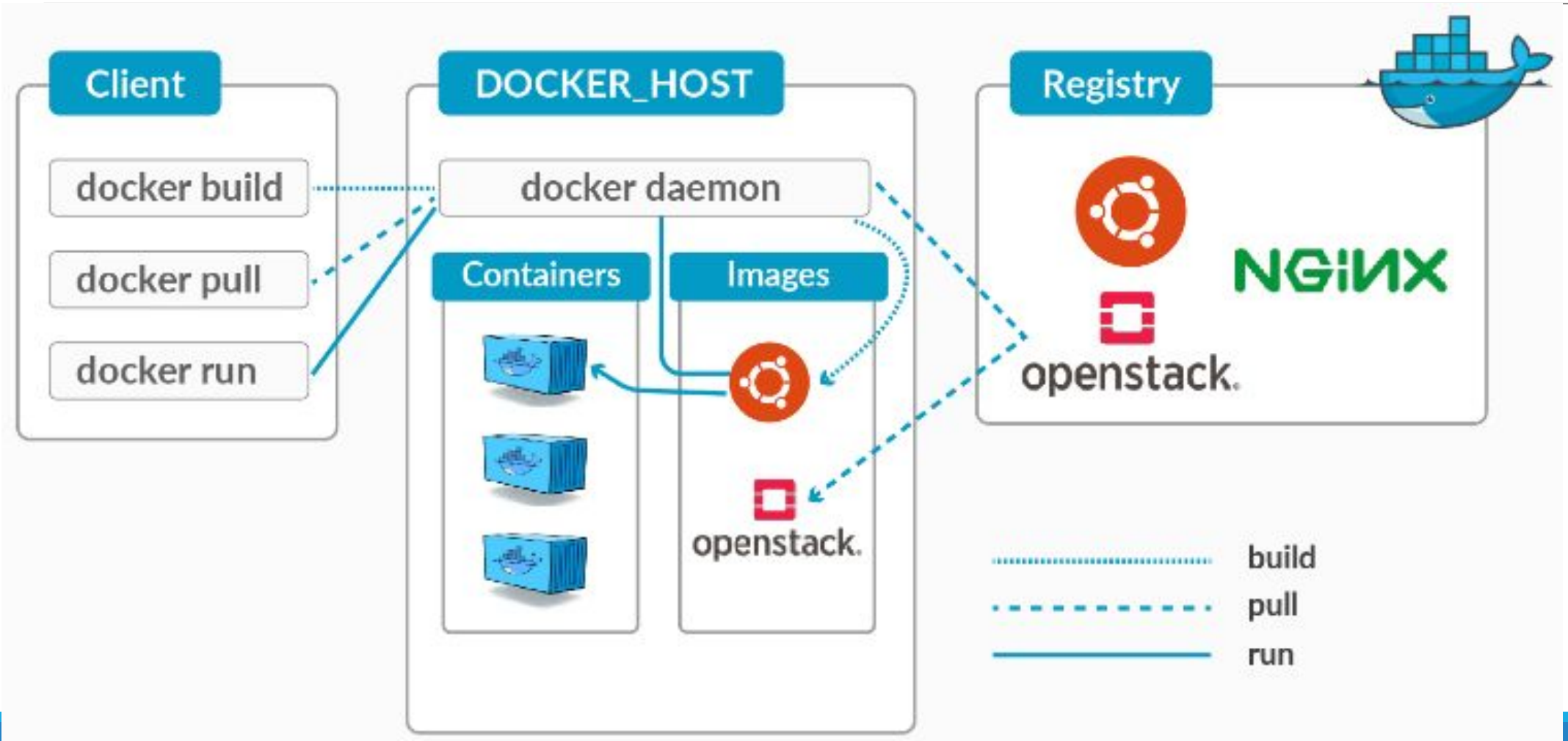
Docker

- Улучшение портируемости приложений - возможность запуска практически везде
- Защита системы
- Организованность зависимостей - мы всегда будем знать что у нас стоит внутри контейнера

Docker

- Безопасность
- Легковесность
- Портируемость
- Масштабируемость
- Слабая связность

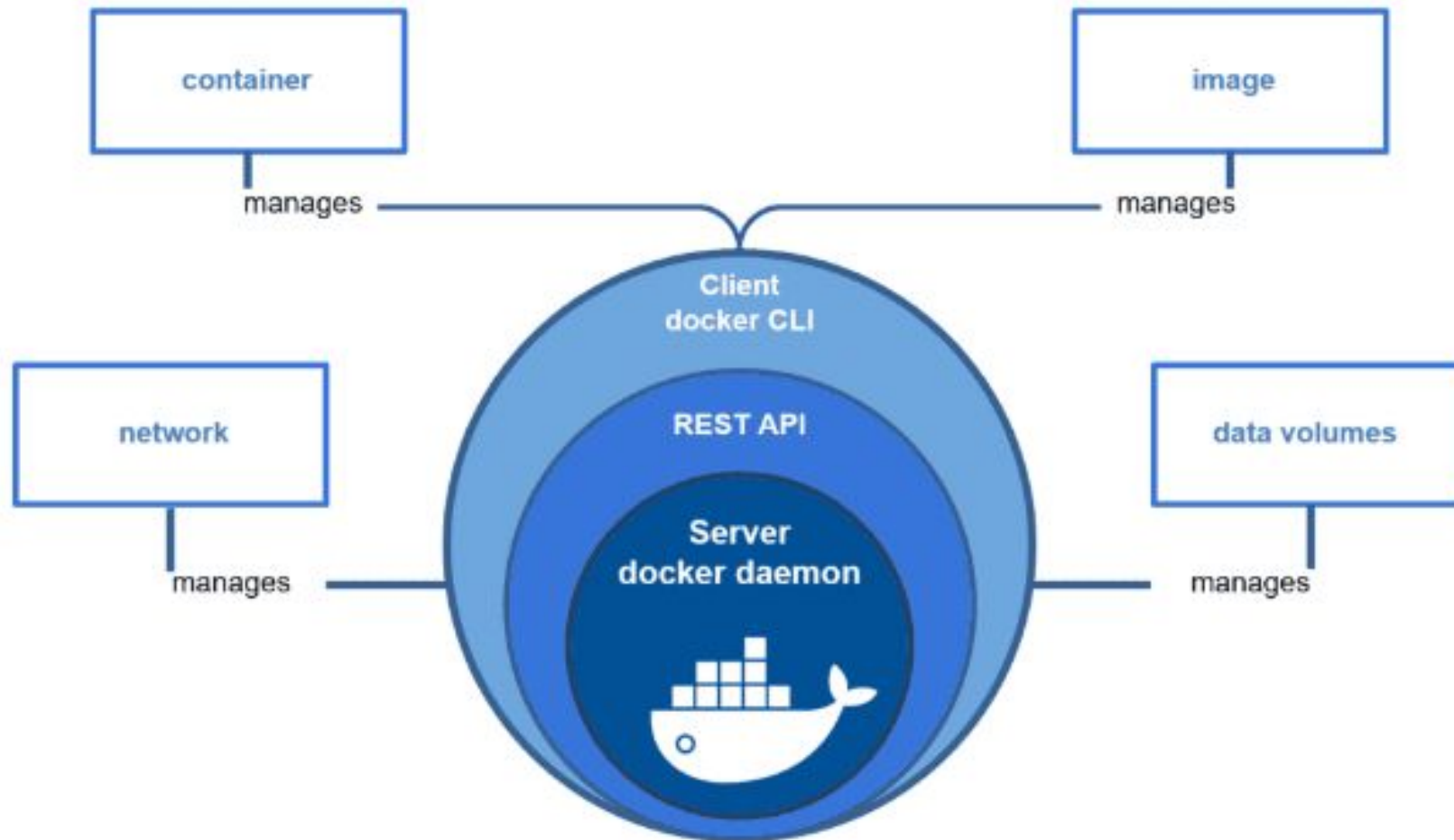
Docker - Архитектура



Docker - Изоляция ресурсов в контейнере

- PID namespace — идентификация процессов и из возможностей
- UTS namespace — хост и доменное имя
- MNT namespace — изоляция файловой системы
- IPC namespace — коммуникация процессов через разделяемую память
- NET namespace — изоляция контроллера сетевого интерфейса
- USR namespace — изоляция пользователей
- chroot() — контроль местоположения корня файловой системы
- cgroups — защита ресурсов

Docker - Docker Engine



Docker

Установка: <https://www.docker.com/>

Docker Desktop

CLI

Docker – основные команды

Чтобы увидеть
основные команды
Docker можно в
командной строке
набрать: docker

docker run --help –
справка по
команде run

```
C:\Users\ANNA>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers


Options:
  --config string      Location of client config files (default
                        "C:\\Users\\ANNA\\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\\Users\\ANNA\\.docker\\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\\Users\\ANNA\\.docker\\cert.pem")
  --tlskey string       Path to TLS key file (default
                        "C:\\Users\\ANNA\\.docker\\key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit
```

Docker – Docker Desktop

 Containers

 Images

 Volumes


 Dev Environments BETA

Extensions BETA


⋮

 Add Extensions

Containers

[Give feedback](#) 

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)



Run a Sample Container

Try running a container: Copy and paste this command into your terminal and then come back

```
docker run -d -p 80:80 docker/getting-started
```



Docker – ОСНОВНЫЕ КОМАНДЫ

```
docker run -d -p 80:80 docker/getting-started
```

Если возникает ошибка from daemon, то нужно остановить службу веб-публикации W3SVC в диспетчере задач










```
C:\Users\ANNA>docker run -d -p 80:49154 docker/getting-started
a04ca9715094aed89a5cbad8ae5ca3b7d879db518a3634eae8cb2ce7f1df1602
docker: Error response from daemon: Ports are not available: exposing port TCP
P 0.0.0.0:80 -> 0.0.0.0:0: listen tcp 0.0.0.0:80: bind: An attempt was made to
access a socket in a way forbidden by its access permissions.
```

```
C:\Users\ANNA>docker run -d -p 80:80 docker/getting-started
85296d0bc3f767fc13d0a728fce6199f54e791ca25581665eeb032b1dce2d4fd
```

Docker – Containers

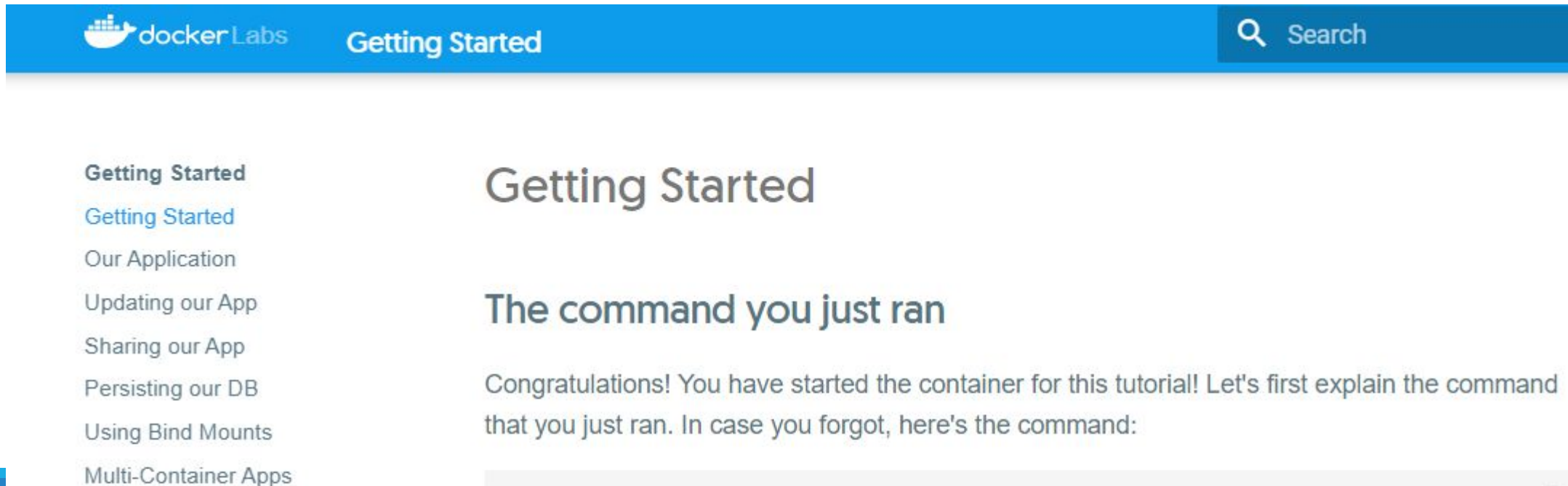
```
docker run -d -p 80:80 docker/getting-started
```

The screenshot shows the Docker Desktop application window. The top bar is blue with the 'Docker Desktop' logo, an 'Upgrade plan' button, a search bar, and icons for settings, sign in, and window controls. The left sidebar contains navigation links: 'Containers' (selected), 'Images', 'Volumes', 'Dev Environments' (marked BETA), 'Extensions' (marked BETA), and 'Add Extensions'. The main area is titled 'Containers' with a 'Give feedback' link. Below the title is a description: 'A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)'. A toggle switch 'Only show running containers' is turned on. A search bar is present. Below is a table of containers:

<input type="checkbox"/>	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	 hardcore_shirley 4e166bc2c170 	docker/getting-started:latest	Running	80:80 	1 minute ago	 View details  Copy docker run  Open in terminal  Pause  Restart  Open with browser

Docker – ОСНОВНЫЕ КОМАНДЫ

Открыть контейнер в браузере:
Open with browser – из Docker Desktop
Ввести в браузере – localhost:80





Docker – Images


Docker Desktop


Upgrade plan


Search **Ctrl+K**








Sign In 











 Containers

 Images


 Volumes

 Dev Environments **BETA**

Extensions **BETA** 

 Add Extensions

Images

[Give feedback](#) 


An image is a read-only template with instructions for creating a Docker container. [Learn more](#)

LOCAL


REMOTE REPOSITORIES


Refresh to see disk usage





1 images

Last refresh: Never 

Search





<input type="checkbox"/>	NAME	TAG	STATUS	CREATED	SIZE	ACTIONS
<input type="checkbox"/>	docker/getting-started 3e4394f6b72f 	latest	In use	7 days ago	46.96 MB	<div><div> View details</div><div> Pull</div><div> Push to Hub</div></div>

Docker – ОСНОВНЫЕ КОМАНДЫ

>docker info -
полная инфо о
докере

>docker images –
об образах

>docker ps

>docker ps –a

О контейнерах

```
C:\Users\ANNA>Docker info
Client:
Context:    default
Debug Mode: false
Plugins:
buildx: Docker Buildx (Docker Inc., v0.9.1)
compose: Docker Compose (Docker Inc., v2.13.0)
dev: Docker Dev Environments (Docker Inc., v0.0.5)
extension: Manages Docker extensions (Docker Inc., v0.2.16)
sbom: View the packaged-based Software Bill Of Materials (SBOM)
e (Anchore Inc., 0.6.0)
scan: Docker Scan (Docker Inc., v0.22.0)

Server:
Containers: 1
  Running: 1
  Paused: 0
  Stopped: 0
Images: 1
Server Version: 20.10.21
```

```
C:\Users\ANNA>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker/getting-started	latest	3e4394f6b72f	6 days ago	47MB

```
C:\Users\ANNA>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4e166bc2c170	docker/getting-started	"/docker-entrypoint..."	19 minutes ago	Up 19 minutes	0.0.0.0:80->80/tcp	hardcore_shirley

Docker – ОСНОВНЫЕ КОМАНДЫ

Просмотр логов контейнера (по имени или идентификатору)

```
>docker logs name | id
```

```
>docker logs --tail <number> <container_id>
```

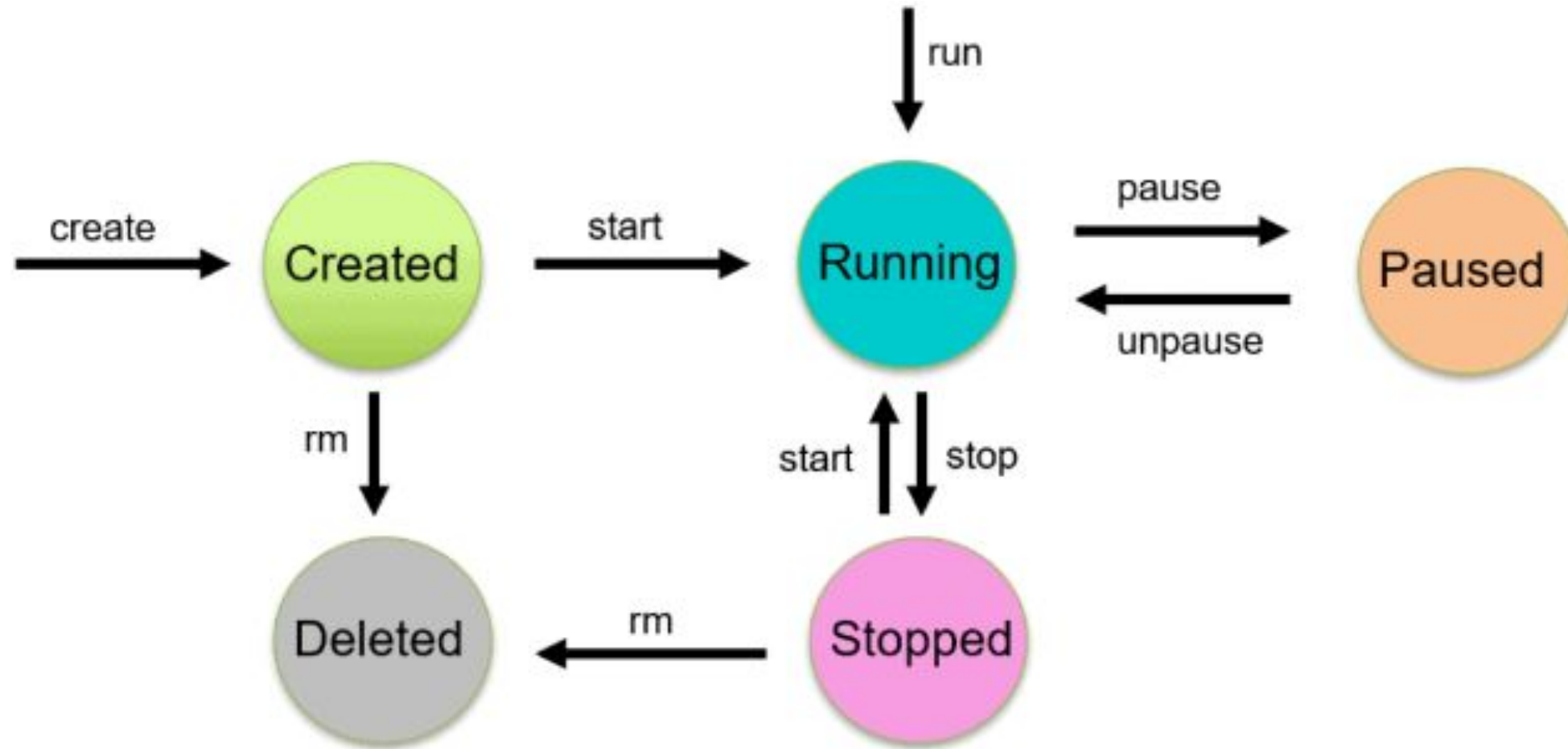
```
>docker logs --tail 4 fb54fd336bc4
```

1) --follow, -f - логи в реальном времени;

2) --timestamps, -t -показывать время (timestamp) перед каждой строчкой лога;

3) --tail - количество строк для вывода (по умолчанию - все).

Докер-жизненный цикл контейнера



Docker – ОСНОВНЫЕ КОМАНДЫ

- >docker stop <CONTAINER ID>
- >docker start <CONTAINER ID>
- >docker pause<CONTAINER ID>
- >docker unpause<CONTAINER ID>
- >docker restart<CONTAINER ID>
- >docker image rm 1d6d4f4c152

```
docker start e7cb6065b188
docker pause e7cb6065b188
docker unpause e7cb6065b188
docker restart e7cb6065b188
```


Docker — Pause vs STOP

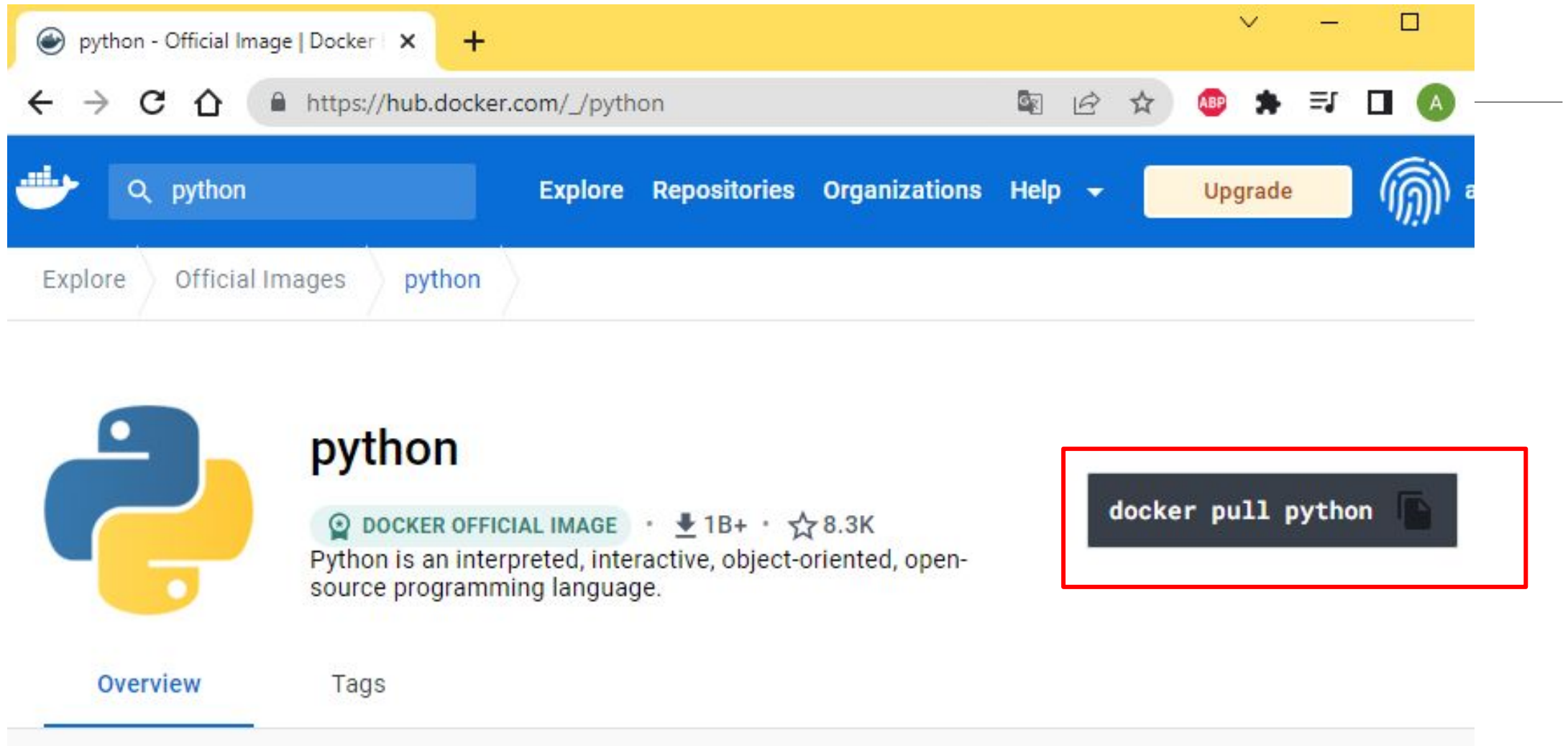
docker pause посылает SIGSTOP сигнал всем процессам в контейнере
docker stop же посылает SIGTERM сигнал главному процессу в контейнере (PID=1) и спустя какое-то время SIGKILL.

SIGTERM — сигнал завершения. По умолчанию используется чтобы завершить процесс, но они иногда могут быть проигнорированы. Его необходимо обрабатывать, если важно провести очистку используемых ресурсов.

SIGKILL — сигнал принудительного завершения. Используется для незамедлительного завершения процесса. Отсюда следует, что о никакой очистке ресурсов и речи быть не может.

SIGSTOP — сигнал паузы. Сигнал не может быть отловлен и проигнорирован приложением. Используется для контроля над приложениями.

Docker – dockerhub



The screenshot shows the Docker Hub interface for the 'python' official image. The browser address bar displays 'https://hub.docker.com/_/python'. The page header includes a search bar with 'python' entered, and navigation links for 'Explore', 'Repositories', 'Organizations', and 'Help'. A yellow 'Upgrade' button is visible on the right. Below the header, a breadcrumb trail shows 'Explore' > 'Official Images' > 'python'. The main content area features the Python logo, the name 'python', and a badge indicating it is a 'DOCKER OFFICIAL IMAGE'. It also shows download statistics ('1B+') and star counts ('8.3K'). A description states: 'Python is an interpreted, interactive, object-oriented, open-source programming language.' At the bottom, there are tabs for 'Overview' and 'Tags'. On the right side, a red rectangular box highlights a dark button with the text 'docker pull python' and a document icon.


python - Official Image | Docker

https://hub.docker.com/_/python

python

Explore Repositories Organizations Help Upgrade

Explore Official Images python

 **python**

DOCKER OFFICIAL IMAGE • 1B+ • 8.3K

Python is an interpreted, interactive, object-oriented, open-source programming language.

Overview Tags

docker pull python

Docker – dockerhub

```
C:\Users\ANNA>docker pull python
Using default tag: latest
latest: Pulling from library/python
32de3c850997: Pull complete
fa1d4c8d85a4: Pull complete
c796299bbbdd: Pull complete
81283a9569ad: Pull complete
60b38700e7fb: Pull complete
0f67f32c26d3: Pull complete
1922a20932d4: Pull complete
47dd72d73dba: Pull complete
25f882f6cd8b: Pull complete
Digest: sha256:250990a809a15bb6a3e307fec72dead200
Status: Downloaded newer image for python:latest
docker.io/library/python:latest
```

Docker – ОСНОВНЫЕ КОМАНДЫ

Images

[Give feedback](#)

An image is a read-only template with

```
C:\Users\ANNA>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker/getting-started	latest	3e4394f6b72f	12 days ago	47MB
python	latest	75cc8d87cc34	13 days ago	932MB

LOCAL

REMOTE REPOSITORIES

Refresh to see disk usage 2 images

Last refresh: Never

Search






<input type="checkbox"/>	NAME	TAG	STATUS	CREATED	SIZE	ACTIONS
<input type="checkbox"/>	docker/getting-started 3e4394f6b72f	latest	In use	13 days ago	46.96 MB	
<input type="checkbox"/>	python 75cc8d87cc34	latest	Unused	14 days ago	931.8 MB	

Docker – основные команды

Containers [Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

☐ Only show running containers

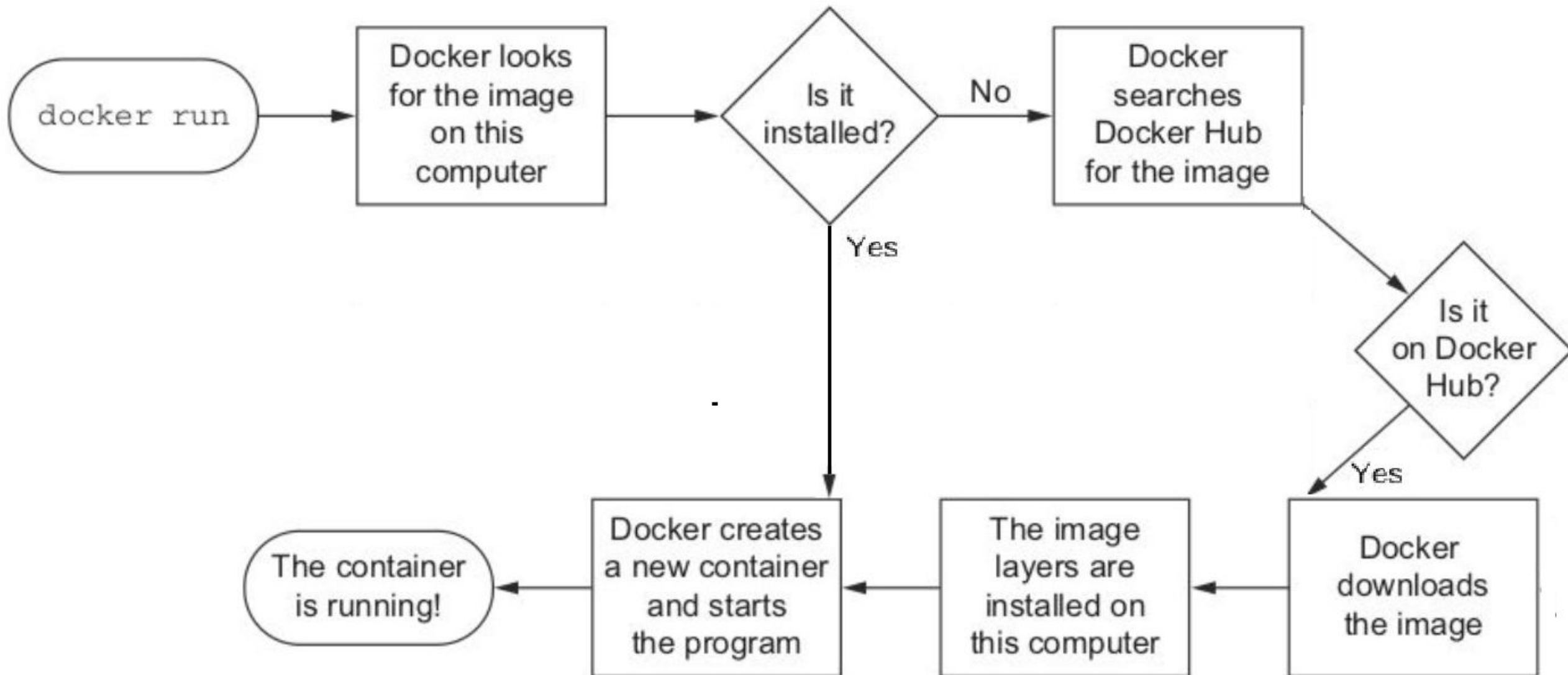
<input type="checkbox"/>		NAME	IMAGE	STATUS	PORT(S)
<input type="checkbox"/>		hardcore_shirley 4e166bc2c170 	docker/getting-started:latest	Exited	80:80 
<input type="checkbox"/>		great_nash c50cadb9451b 	python:latest	Exited	

```
C:\Users\ANNA>docker run python
```

```
C:\Users\ANNA>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
TS	NAMES			
a8d36159fd9c	docker/getting-started:latest	"/docker-entrypoint..."	19 minutes ago	Exited (0) 18 minutes ago
68032520ac3e	python:latest	"python3"	25 minutes ago	Exited (0) 24 minutes ago
c50cadb9451b	python	"python3"	31 minutes ago	Exited (0) 24 minutes ago
4e166bc2c170	docker/getting-started	"/docker-entrypoint..."	5 days ago	Exited (0) 19 minutes ago
	hardcore_shirley			

Docker – запуск контейнера



Docker – ОСНОВНЫЕ КОМАНДЫ

Запустить образ в интерактивном режиме (-it)

С указанием имени для контейнера (--name MyPython)

docker run -it --name MyPython python

```
C:\Users\ANNA>docker run -it --name MyPython python
Python 3.11.1 (main, Dec 21 2022, 18:32:57) [GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+2
3
```



MyPython

745dc9e4d03f



[python:latest](#)

Running

1 minute ago



Docker – ОСНОВНЫЕ КОМАНДЫ

Вернуться в Docker:

<Ctrl+d>

Запустить созданный контейнер:

docker start MyPython

```
C:\Users\ANNA>docker start MyPython
MyPython
```


Docker – ОСНОВНЫЕ КОМАНДЫ

Ubuntu

>docker pull ubuntu

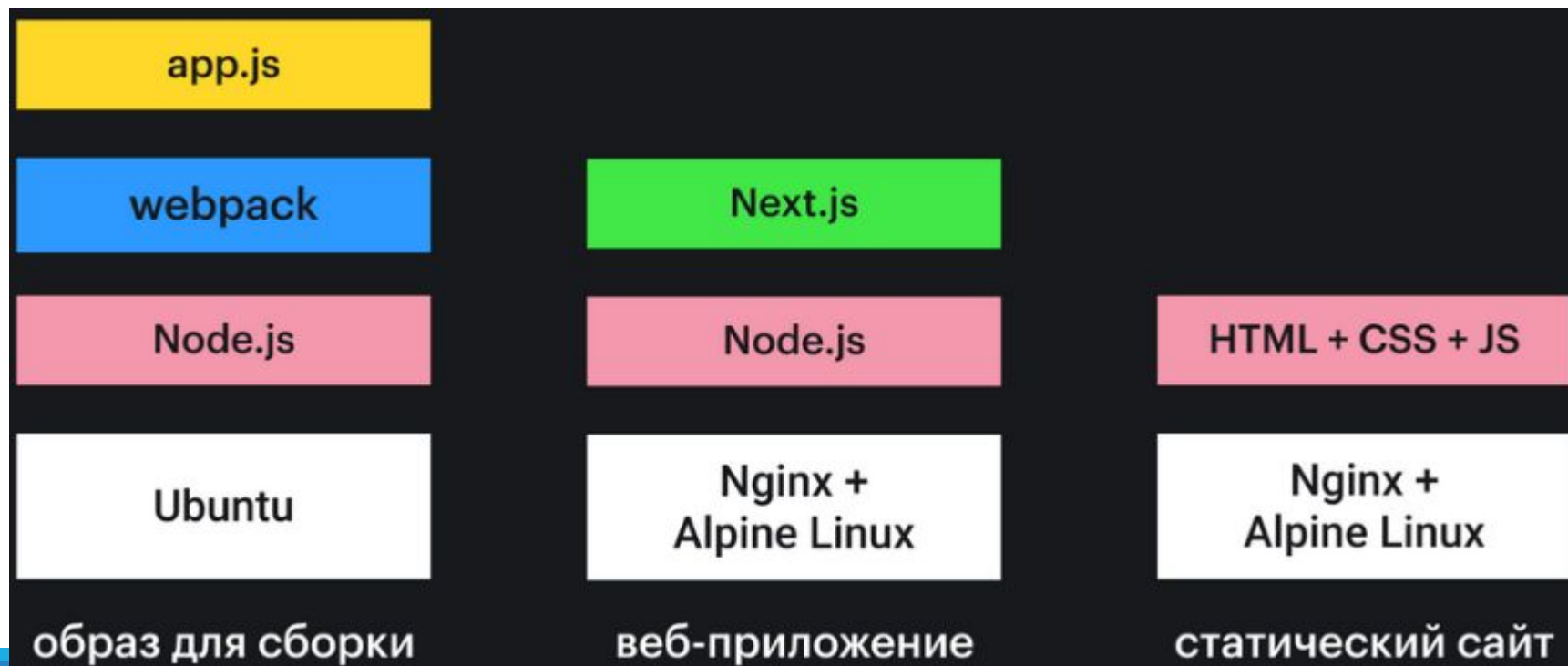
>docker run -it --name MyUbuntu ubuntu

```
C:\Users\ANNA>docker run -it --name MyUbuntu ubuntu
root@f625b704290e:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media
mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@f625b704290e:/# cd media
root@f625b704290e:/media# touch file.txt
root@f625b704290e:/media# ls
file.txt
root@f625b704290e:/media#
```

Docker – Dockerfile

Comment

INSTRUCTION arguments



Docker – Dockerfile

FROM	Скачиваемый образ (можно указать версию)
WORKDIR	Рабочая папка
COPY	Файлы из вашего проекта, которые копируются на хост машину (какие файлы будут выполнены за счет возможностей образа)
EXPOSE	Порт для проекта. Будет работать в том случае, если в образе есть локальный сервер
RUN	Команда, выполняемая один раз при сборке всего контейнера. Используется для установки в контейнер пакетов.

Docker – Dockerfile

LABEL	Метаданные. Например — сведения о том, кто создал и поддерживает образ
ENV	Переменные среды
ADD	копирует файлы и папки в контейнер, может распаковывать локальные .tar-файлы
ARG	задаёт переменные для передачи Docker во время сборки образа
ENTRYPOINT	Команда с аргументами для вызова во время выполнения контейнера. Аргументы не переопределяются.
EXPOSE	Указывает на необходимость открыть порт
VOLUME	Создаёт точку монтирования для работы с постоянным хранилищем

Docker – Dockerfile

https://hub.docker.com/_/python

```
FROM python:3

WORKDIR /usr/src/app

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD [ "python", "./your-daemon-or-script.py" ]
```

Docker – Dockerfile

```
FROM python  
COPY . /python  
WORKDIR /python
```

```
CMD [ "python", "./yourScript.py" ]
```

FROM python:3.7.2-alpine3.8

LABEL maintainer="user@gmail.com"

Устанавливаем зависимости

RUN apk add --update git

Задаём текущую рабочую директорию

WORKDIR /usr/src/my_app_directory

Копируем код из локального контекста в рабочую директорию образа

COPY . .

Задаём значение по умолчанию для переменной

ARG my_var=my_default_value

Настраиваем команду, которая должна быть запущена в контейнере во время его выполнения

ENTRYPOINT ["python", "./app/my_script.py", "my_var"]

Открываем порты

EXPOSE 8000

Создаём том для хранения данных

VOLUME /my_volume

Docker – Build

`docker build [options] path | url`

options - все опции доступны по

<https://docs.docker.com/engine/reference/commandline/build/>

path - путь к контексту сборки

url - url по которому находится контекст сборки

Docker – Build

> docker build .

> docker build d:/DockerTest

```
C:\Users\ANNA>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	55b90549b758	14 seconds ago	932MB

>docker run ID

>docker run 55b90549b758

```
C:\Users\ANNA>docker run 55b90549b758
Hello!!!!
```

Docker – Build

с указанием имени и тэга

```
> docker build d:/DockerTest -t web-app:1.0.0
```

```
C:\Users\ANNA>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
web-app	1.0.0	e362499a459e	39 minutes ago	932MB

Залить в удаленный репозиторий

```
> docker push web-app:1.0.0
```

Docker – Docker Compose

```
✓ APP
  > php
  🔥 docker-compose.yml

🔥 docker-compose.yml
1  # версия
2  version: '3.1'
3  # сервисы-различные образы, которые будут подключаться
4  # и каждому характеристики подключения
5  services:
6  #СУБД
7    db:
8      image: mariadb:10.6
9      restart: always
10     environment:
11       MYSQL_ROOT_PASSWORD: 12345
12  # графический интерфейс php myadmin
13  phpmyadmin:
14    image: phpmyadmin
15    restart: always
16    ports:
17      - 8080:80
18    environment:
19      - PMA_ARBITRARY=1
```

Собрать проект

>docker-compose build

Запустить контейнер

>docker-compose up


Остановить проект

>docker-compose down

A solid blue horizontal bar at the bottom of the slide.

Docker – Docker Compose

http://localhost:8080/index.php?route=/server/sql



Добро пожаловать в phpMyAdmin

Язык (Language)

Русский - Russian

Авторизация

Сервер:

Пользователь: root

Пароль:

Авторизация

phpMyAdmin

Недавнее Избранное

- Создать БД
- information_schema
- mysql
- performance_schema
- sys

Сервер: db

Основные настройки

- Изменить пароль
- Сопоставление кодировки соединения: utf8mb4_unicode_ci
- Дополнительные настройки

Настройки внешнего вида

- Язык (Language): Русский - Russian
- Тема: pmahomme

Docker – Docker Compose

🔥 docker-compose.yml

```
1  version: '3.1'
2  services:
3
4  php:
5    build: ./php
6    ports:
7      - 8081:80
8
9  db:
10    image: mysql
11    command: --default-authentication-plugin=mysql_native_password
12    restart: always
13    environment:
14      MYSQL_ROOT_PASSWORD: 12345
15
16  phpmyadmin:
17    image: phpmyadmin
18    restart: always
19    ports:
20      - 8080:80
21    environment:
22      - PMA_ARBITRARY=1
```


Docker – Docker Compose

```
PS D:\DockerTest\app> docker compose build
[+] Building 1.7s (8/8) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/php:7.2-apache 1.4s
=> [1/3] FROM docker.io/library/php:7.2-apache@sha256:4dc0f0115acf8c2f0df69295ae822e49f5ad5fe849725847f15aa0e 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 60B 0.0s
=> CACHED [2/3] WORKDIR /var/www/html 0.0s
=> CACHED [3/3] COPY . /var/www/html 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:4f31cfec004ec0442235e2693eb9ee38985db25faa8839abe92b877bb13ec303 0.0s
=> => naming to docker.io/library/app-php 0.0s
```

>docker compose up

Docker – Docker Compose




Images [Give feedback](#)

An image is a read-only template with instructions for creating a Docker container. [Learn more](#)

LOCAL REMOTE REPOSITORIES


 Refresh to see disk usage 3 images

Last refresh: N

<input type="checkbox"/>	NAME	TAG	STATUS	CREATED	SIZE	ACTI
<input type="checkbox"/>	app-php 4f31cfec004e 	latest	In use	23 minutes ago	410.07 MB	▶
<input type="checkbox"/>	phpmyadmin e0932b3f75d5 	latest	In use	15 days ago	510.56 MB	▶
<input type="checkbox"/>	mysql 7484689f290f 	latest	In use	29 days ago	538.01 MB	▶

Docker – Docker Compose

Containers










[Give feedback](#) 

A container packages up code and its dependencies so the application runs quickly and reliably in any environment. [Learn more](#)

☐ Only running




S

<input type="checkbox"/>		NAME	IMAGE	STATUS	PORT(S)
<input type="checkbox"/>		app	-	Exited	
<input type="checkbox"/>		php-1 2bb7e9b0deed 	app-php:lat	Exited	8081:80 
<input type="checkbox"/>		phpmyadmin-1 a264b37252e2 	phpmyadm	Exited	8080:80 
<input type="checkbox"/>		db-1 d6298f7cbc99 	mysql:lates	Exited	

Docker – Docker Compose

ⓘ http://localhost:8080/index.php?route=/server/sql 🔑 ↗ ☆



Добро пожаловать в phpMyAdmin

Язык (Language)

Русский - Russian ▾

Авторизация ⓘ

Сервер:

Пользователь:

Пароль:

Авторизация

← → ↻ 🏠 ⓘ http://localhost:8081

Hellooooo Anna!!

Сервер: db

Базы данных SQL Состояние Учетные записи пользователей Экспорт Импорт Настройки Ещё

Основные настройки

🔑 Изменить пароль

☰ Сопоставление кодировки соединения: ⓘ

utf8mb4_unicode_ci ▼

🔑 Дополнительные настройки

Настройки внешнего вида

🗣️ Язык (Language) ⓘ Русский - Russian ▼

🎨 Тема rmahomme ▼ View all

Сервер баз данных

- Сервер: db via TCP/IP
- Тип сервера: MySQL
- Соединение сервера: SSL не используется ⓘ
- Версия сервера: 8.0.31 - MySQL Community Server - GPL
- Версия протокола: 10
- Пользователь: root@172.19.0.4
- Кодировка сервера: UTF-8 Unicode (utf8mb4)

Веб-сервер

- Apache/2.4.54 (Debian)
- Версия клиента базы данных: libmysql - mysqlnd 8.0.26
- PHP расширение: mysqli ⓘ curl ⓘ mbstring ⓘ
- Версия PHP: 8.0.26

phpMyAdmin

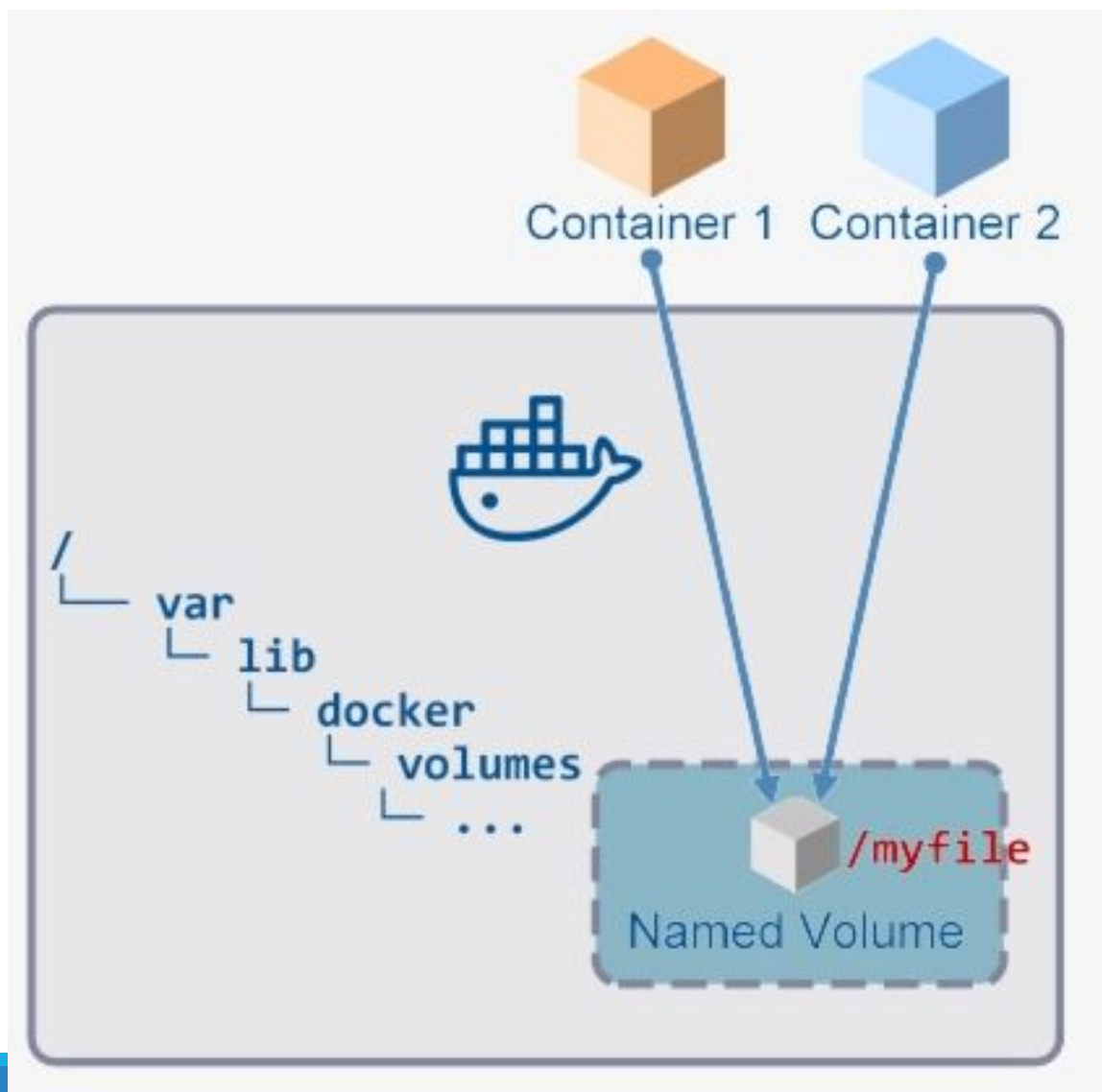
- Информация о версии: 5.2.0 (актуально)
- Документация

Docker – Хранилища данных

По умолчанию все файлы, которые создаются в контейнере, хранятся в специальном записывающем слое контейнера. Это значит:

- 1) Данные не будут существовать без контейнера, и данные будет очень сложно найти, если они понадобятся другому процессу;
- 2) Записывающий слой тесно связан с хост системой. Переместить эти данные куда-то будет непросто;
- 3) Для записи в этот слой необходимы специальные драйвера. Драйвер для хранилища предоставляет объединенную файловую систему, используя ядро линукс. Данный дополнительный слой абстракции замедляет производительность.

Docker – Хранилища данных



Docker – Volume

2 типа:

1) volumes;

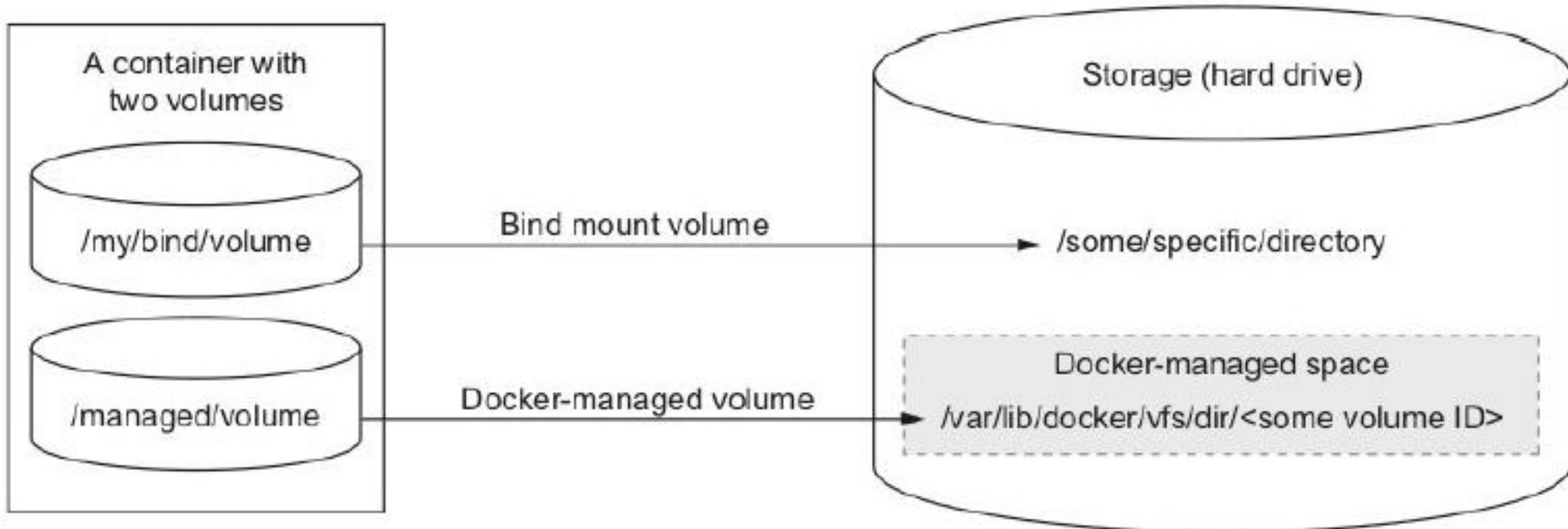
2) mount binds;

Также, если вы работаете на Linux, то можно использовать `tmpfs mount`. Если

на windows, то можно использовать именованные потоки.

Рекомендуется использовать 1-й тип, то есть **VOLUME**.

Docker – Volume



Docker – Volume

Преимущества Volumes над mount binds:

- 1) У volume проще создать резервную копию или переместить ее;
- 2) управлять docker volumes можно через docker CLI и docker API;
- 3) docker volumes работают как на linux, так и на windows;
- 4) более безопасный шаринг volumes между контейнерами;
- 5) для volumes можно использовать различные драйверы, которые позволяют хранить volumes на удаленных машинах или в облаке, шифровать их или предоставляют другую функциональность;
- 6) Новые volumes могут презаполняться контейнером при старте (удобно, что не надо выдавать кучу лишних прав).

Docker – Volume

Создать Volume

>docker volume create <volume-name>

Удалить Volume

>docker volume rm <volume-name>

Список Volume

>docker volume ls

```
PS D:\DockerTest\app> docker volume create volumeTest
volumeTest
PS D:\DockerTest\app> docker volume ls
DRIVER      VOLUME NAME
local       575882f5ff461f5d8aa9b3861dd56c897d0e16256661
local       volumeTest
PS D:\DockerTest\app> docker volume rm volumeTest
volumeTest
PS D:\DockerTest\app> docker volume ls
DRIVER      VOLUME NAME
local       575882f5ff461f5d8aa9b3861dd56c897d0e16256661
```

Docker – Volume


Создать docker volume и запустить контейнер для хранения логов приложения:

```
>docker volume create app-logs
```


```
>docker volume ls
```

```
>docker run -v app-logs:/usr/src/app/log
```

Подключаем сервис для работы с базой данных postgres, django, python и создаем проект

 Dockerfile > ...

```
1 FROM python:3
2 WORKDIR /usr/src/app
3 COPY requirements.txt ./
4 RUN pip install -r requirements.txt
```

 requirements.txt

```
1 Django>=3.0,<4.0
2 psycopg2-binary>=2.8
```

Подключаем сервис для работы с базой данных postgres, django, python и создаем проект

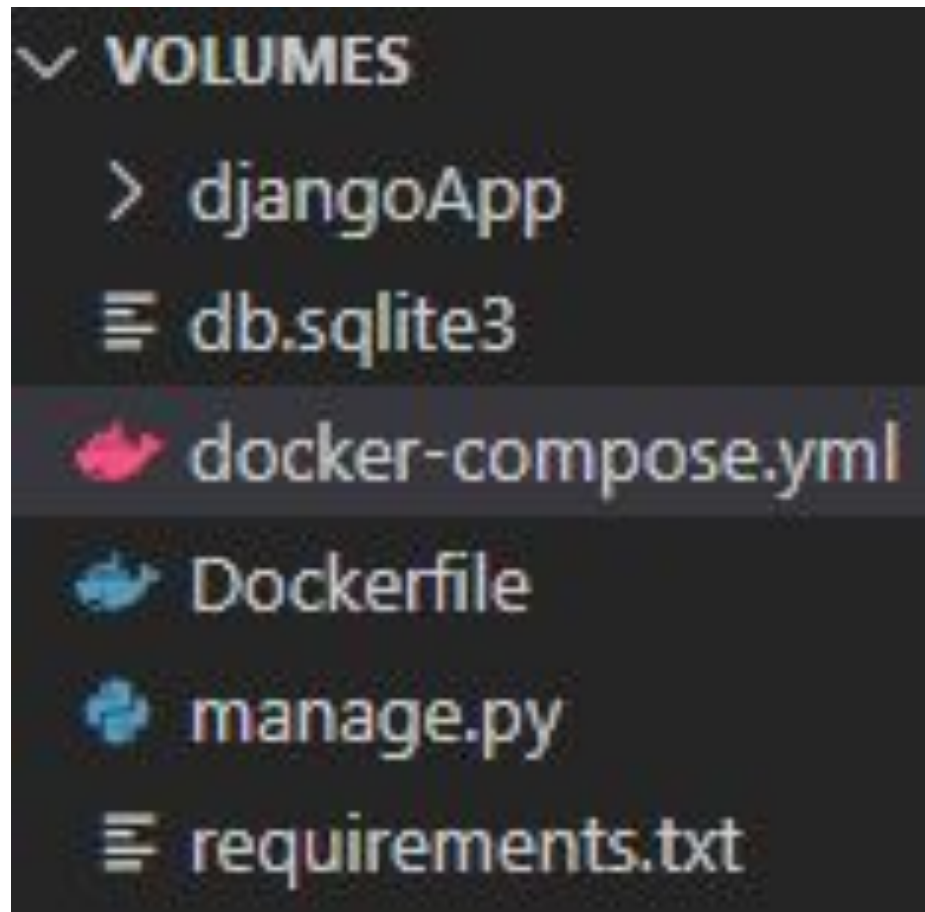
```
🔥 docker-compose.yml
1  version: '3'
2
3  services:
4    django:
5      # все настройки для этого сервиса находятся в Dockerfile
6      # путь не пишем, потому что находится здесь же
7      build: .
8      container_name: django
9      command: python manage.py runserver 0.0.0.0:8000
10     volumes:
11       - ../usr/srs/app
12     ports:
13       - "8000:8000"
14     depends_on:
15       - pgdb
```


Подключаем сервис для работы с базой данных postgres, django, python и создаем проект

```
16
17  ✓ pgdb:
18      image: postgres
19  ✓      environment:
20          - POSTGRES_DB=postgres
21          - POSTGRES_PASSWORD=postgres
22          - POSTGRES_USER=postgres
23      container_name: pgdb
24  ✓      volumes:
25          - pgdbdata:/var/lib/postgresql/data/
26
27  ✓ volumes:
28      | pgdbdata: null
```

Подключаем сервис для работы с базой данных postgres, django, python и создаем проект

> docker-compose run django django-admin startproject djangoApp .



Подключаем сервис для работы с базой данных postgres, django, python и создаем проект

- > docker-compose run django python manage.py migrate

- > docker-compose run django python manage.py createsuperuser

```
PS D:\DockerTest\volumes> docker-compose run django python manage.py createsuperuser
[+] Running 1/0
 - Container pgdb Running
Username (leave blank to use 'root'): admin
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

django

View [release notes](#) for Django 3.2



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



Django Documentation

Topics, references, & how-to's



Tutorial: A Polling App

Get started with Django

Django administration

Username:

Password:

LOG IN

Django administration

Username:

Password:

LOG IN



Django administration

WELCOME, ANNA. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

[+ Add](#) [✎ Change](#)

Users

[+ Add](#) [✎ Change](#)

Recent actions

My actions

None available

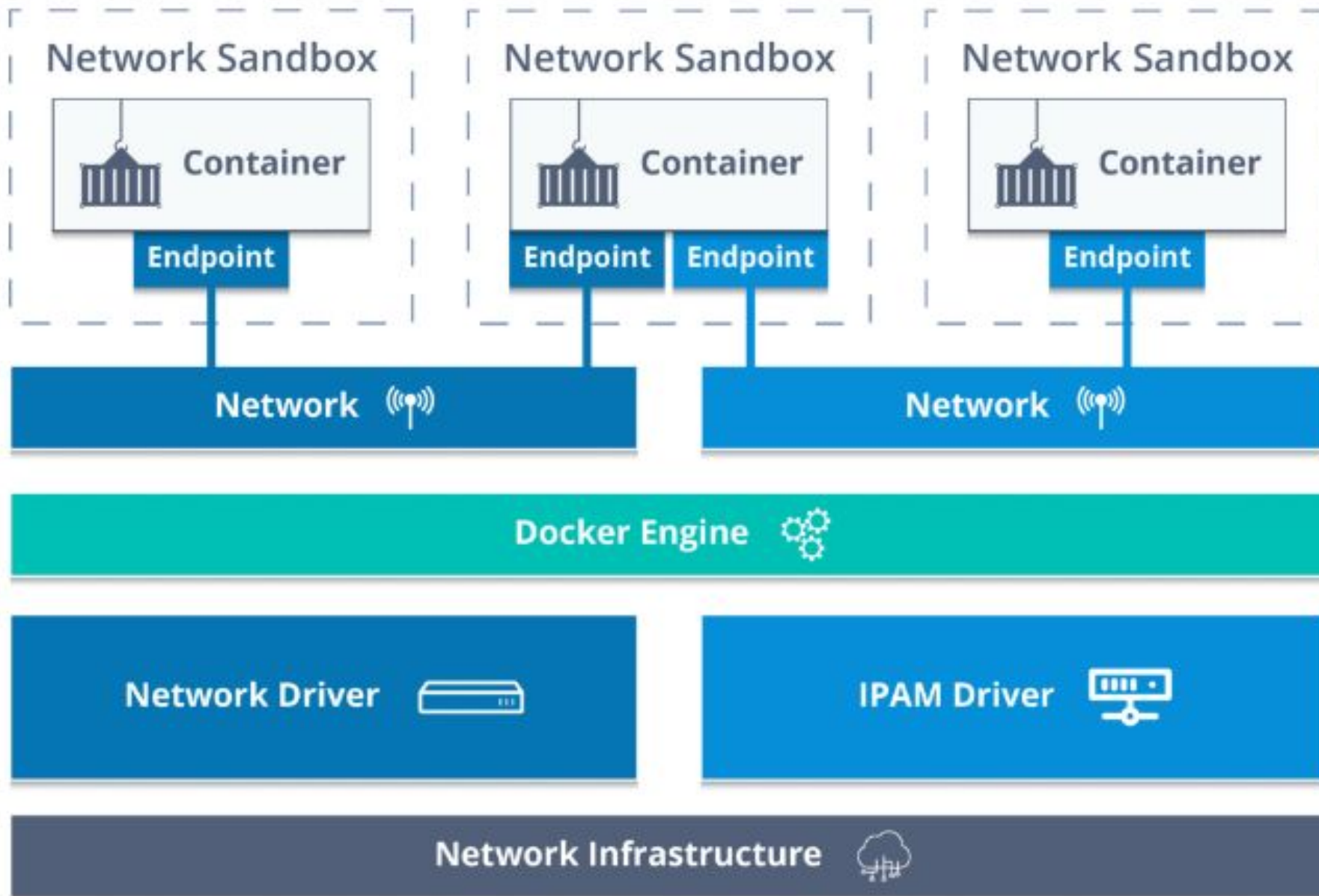
Подключаем сервис для работы с базой данных postgres, django, python и создаем проект

> docker compose down

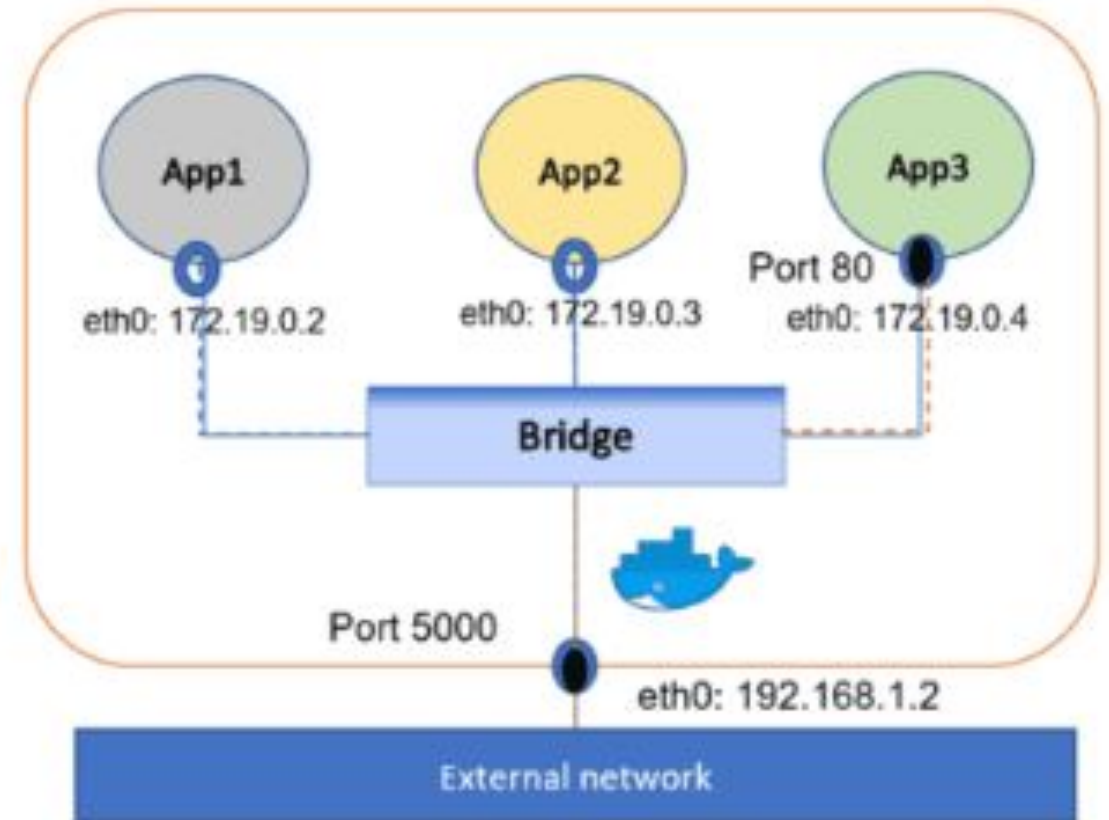
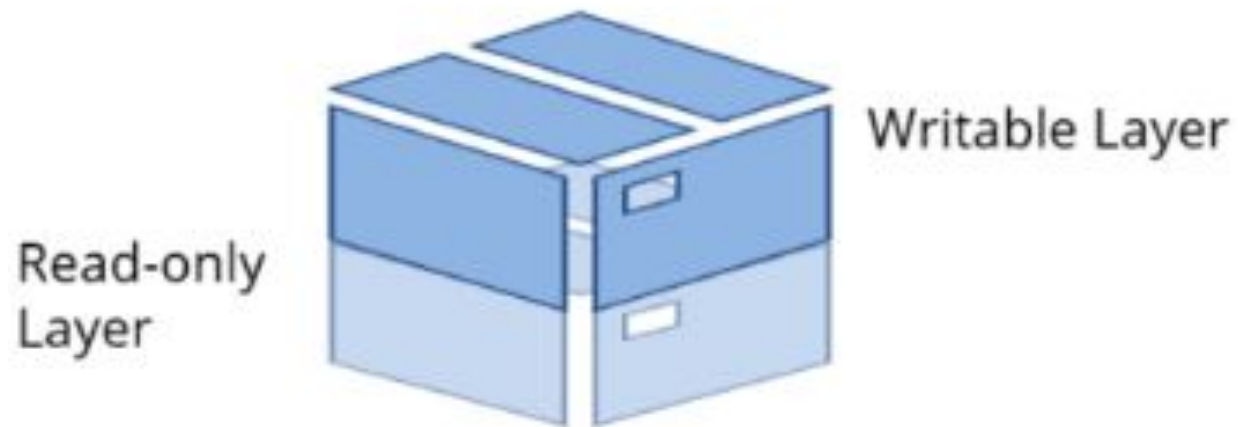
> docker compose up

Логин пароль сохраняются в volume

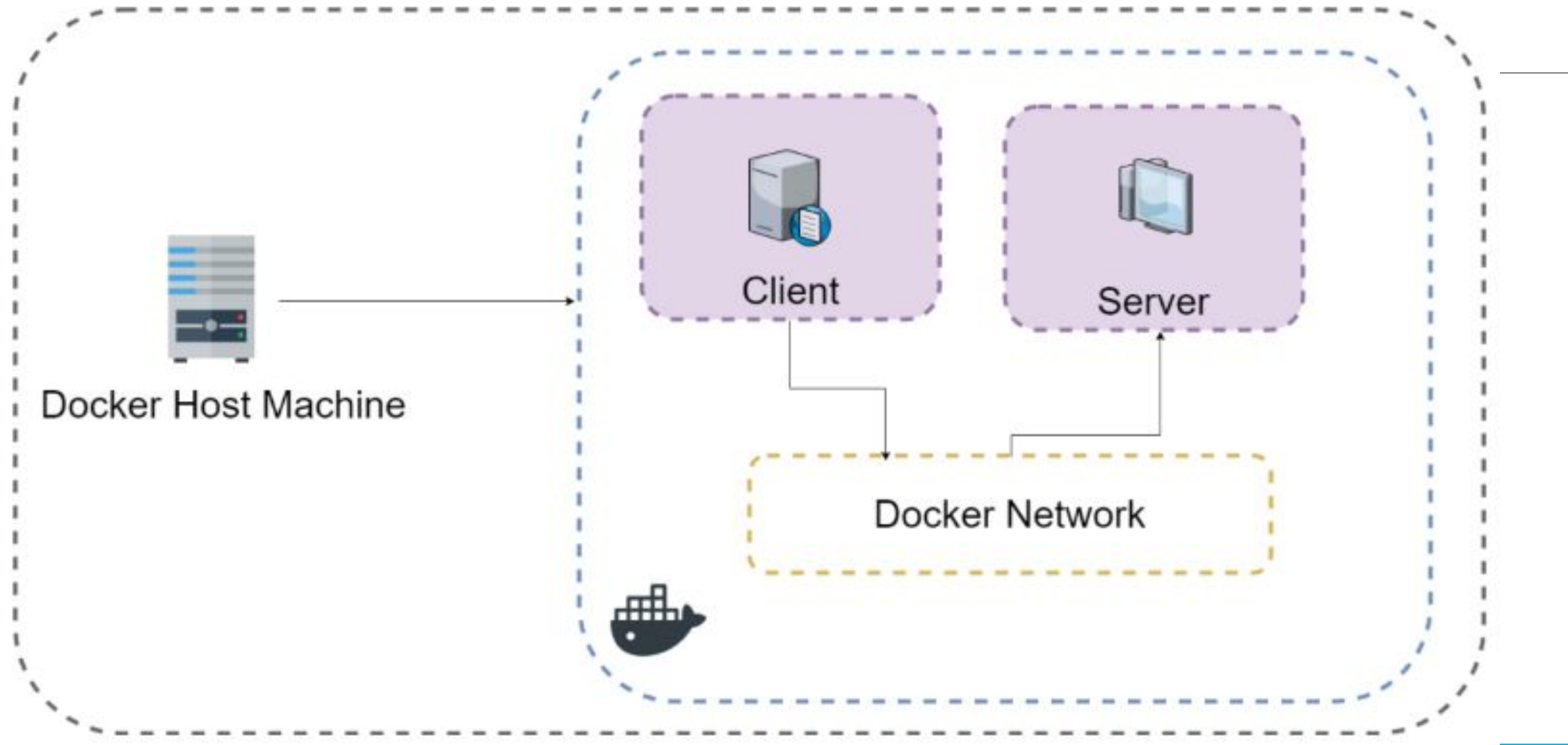
Docker networking



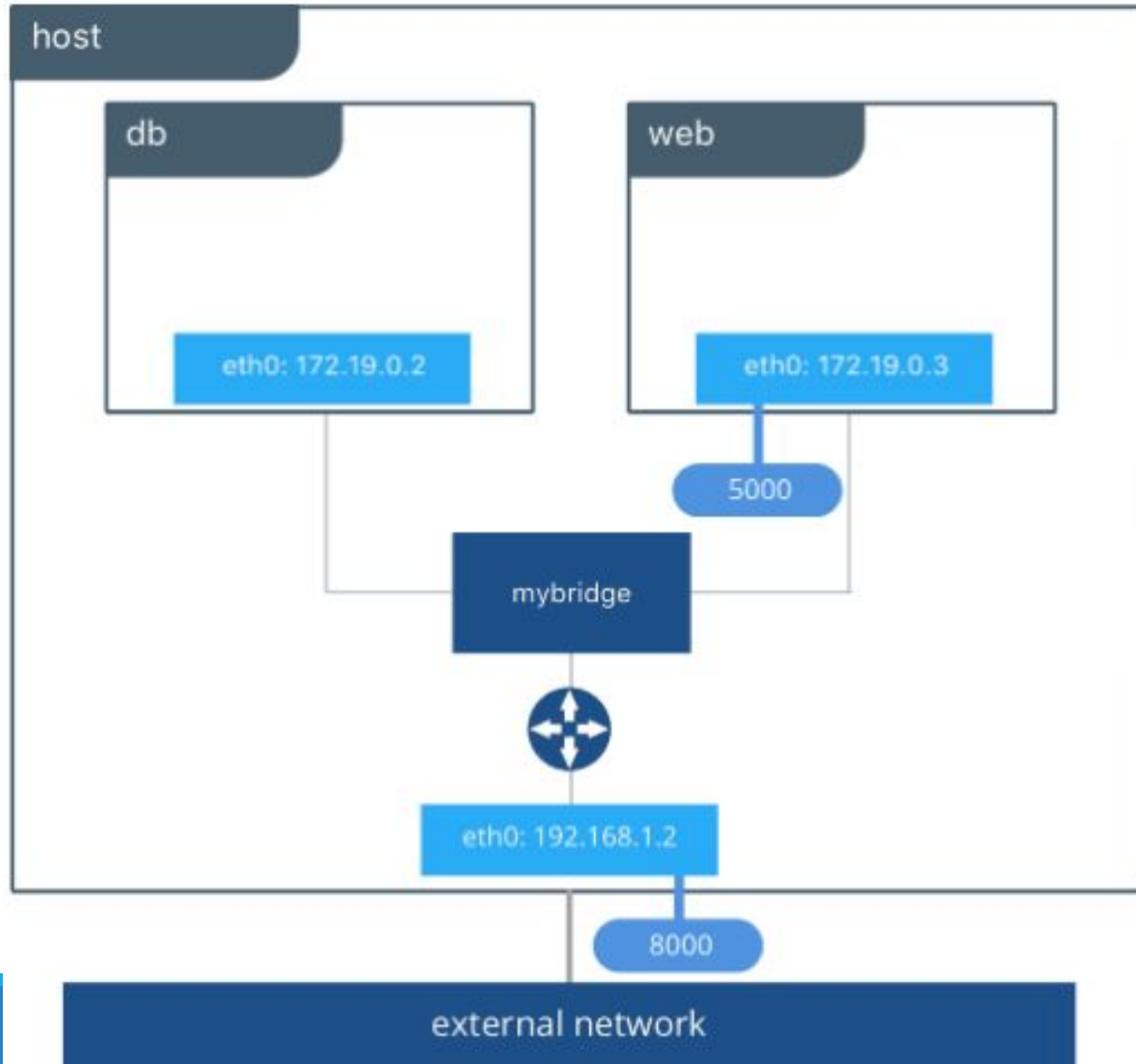
Docker networking



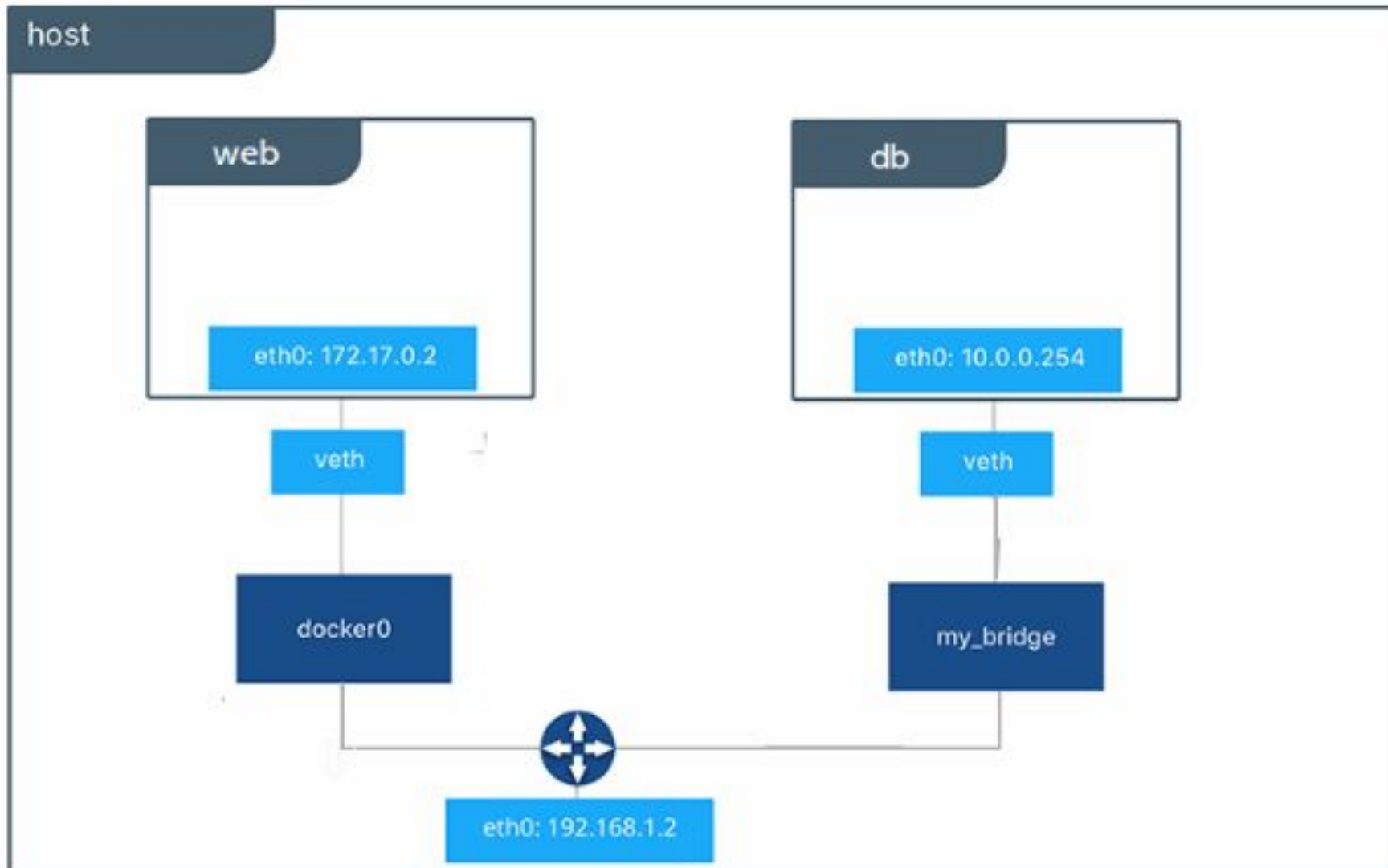
Docker networking



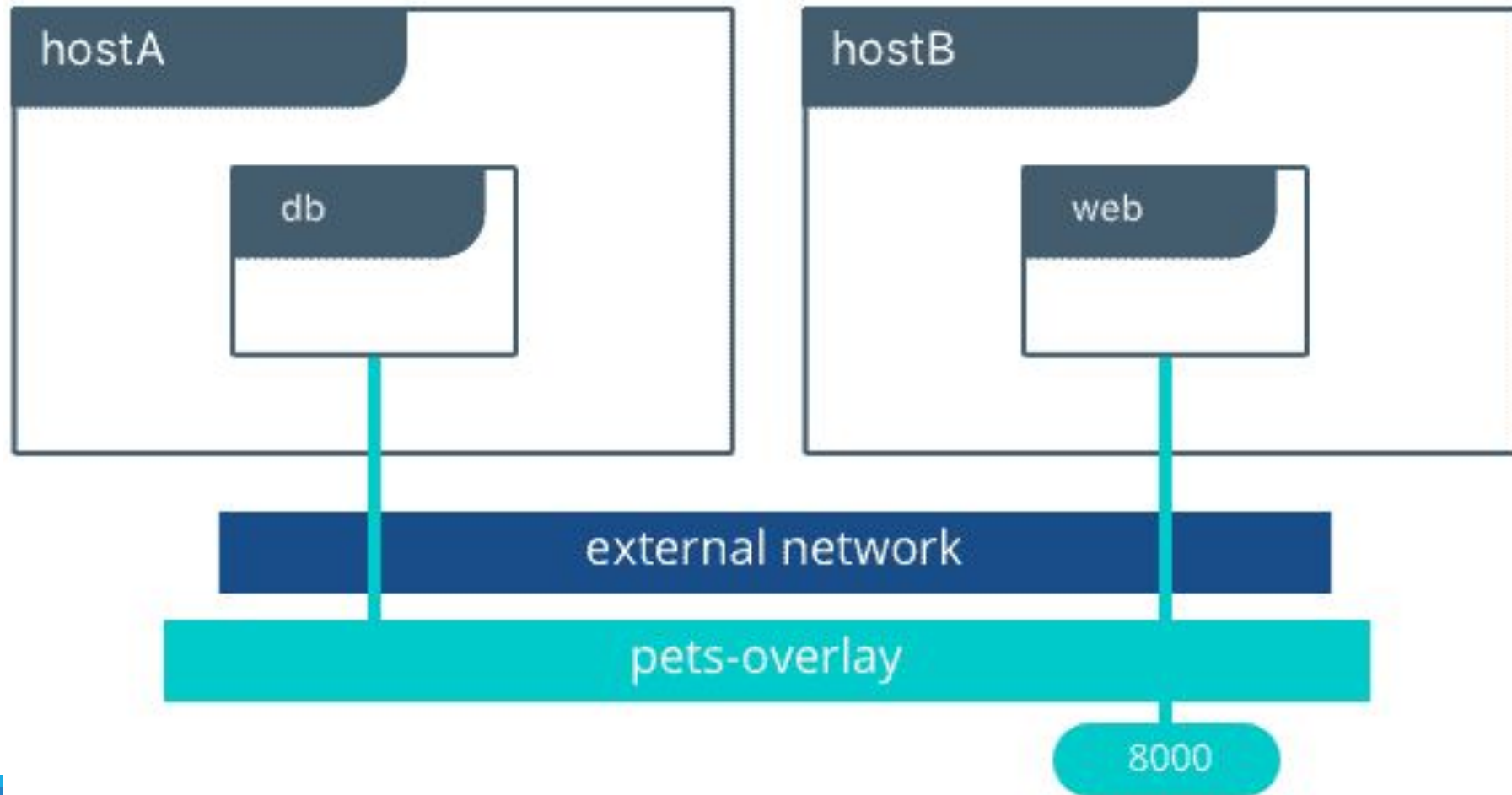
Docker networking - Bridge



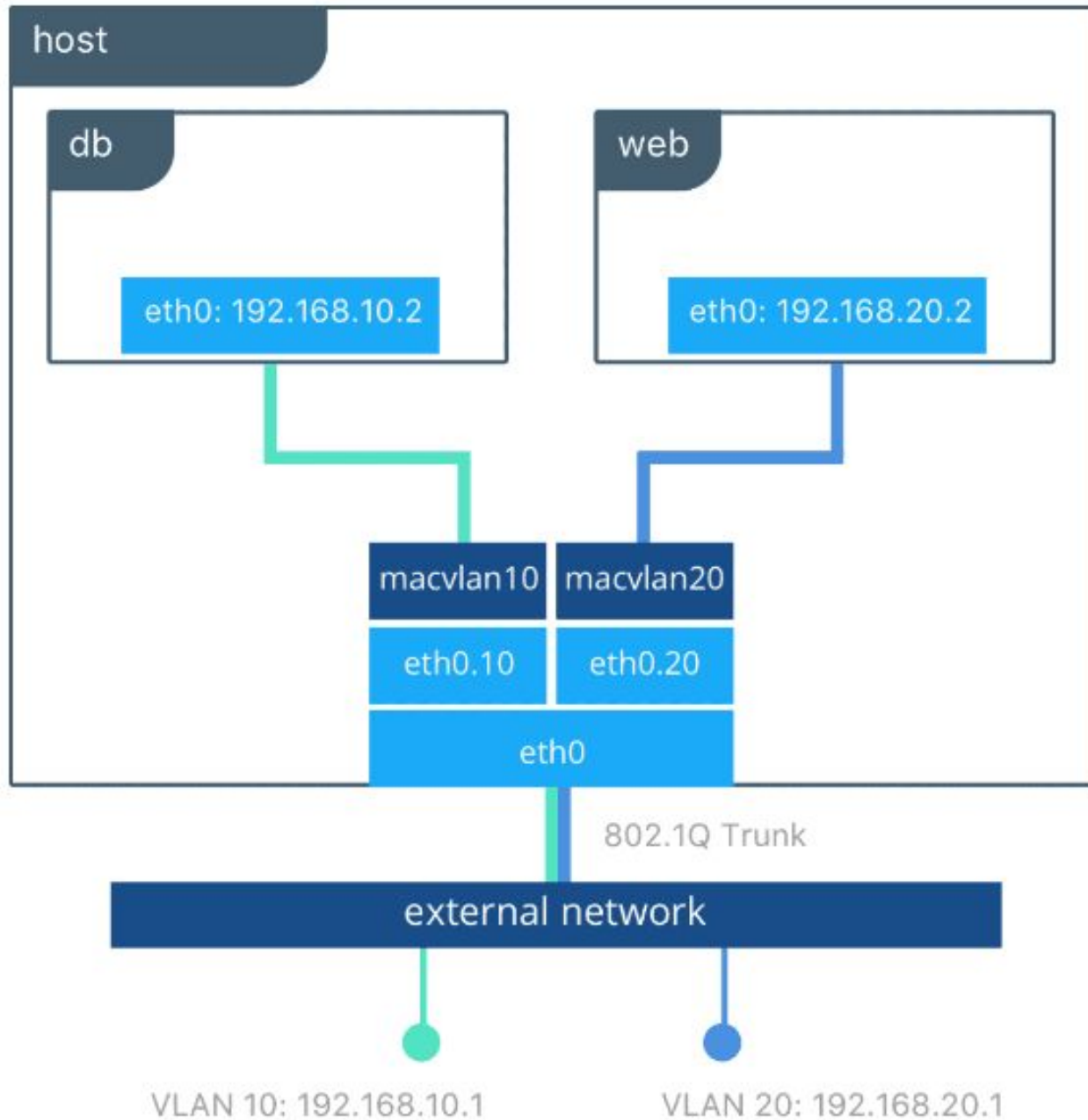
Docker networking - Host



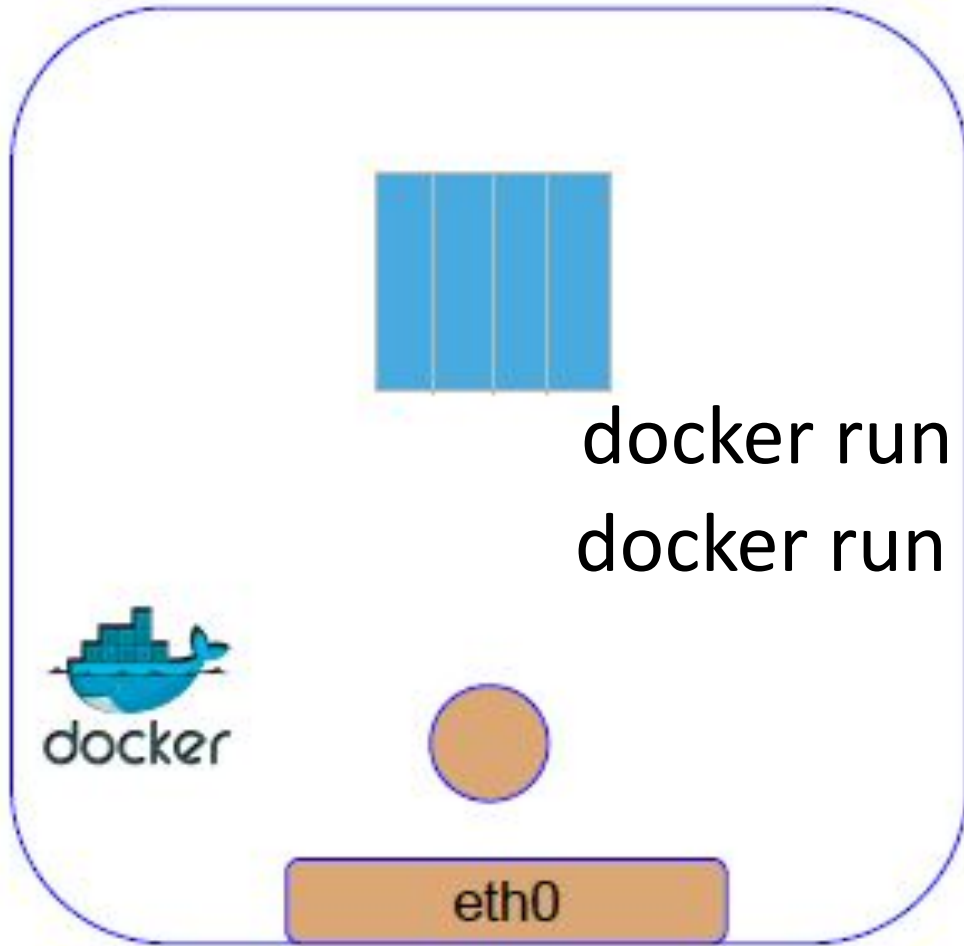
Docker networking - Overlay



Docker networking - Macvlan



Docker networking - None



`docker run --network none <containerName>`
`docker run --network none docker/getting-started`

192.168.1.2

Docker networking – ОСНОВНЫЕ КОМАНДЫ

docker network

Справка о командах

Manage networks

Commands:

connect	Connect a container to a network
create	Create a network
disconnect	Disconnect a container from a network
inspect	Display detailed information on one or more networks
ls	List networks
prune	Remove all unused networks
rm	Remove one or more networks

Docker networking – ОСНОВНЫЕ КОМАНДЫ

`docker network ls`

Листинг всех сетей на текущем хосте

```
PS D:\DockerTest\volumes> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4d4ac804aafe        bridge             bridge             local
09daad94c2ad        host               host               local
6908ce2ee335        none              null              local
```

Docker networking – ОСНОВНЫЕ КОМАНДЫ

`docker network inspect <networkType>`

получить все подробности о
типе сети

```
PS D:\DockerTest\volumes> docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "4d4ac804aafe38ee59ed3bcb004800f001979a28e044d2cfc90747092b9b6a8d",
    "Created": "2023-01-06T08:24:04.1442469Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    }
  },
]
```

Docker networking – ОСНОВНЫЕ КОМАНДЫ

`docker network inspect <network>`

получить все подробности
о типе сети

```
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
  "Network": ""
},
"ConfigOnly": false,
"Containers": {
  "ba3801eb01b75bcd1331e70aa5273e0b5460d2cc9504831abe15213e3f609f0": {
    "Name": "frosty_keller",
    "EndpointID": "4208d7f49d043a3e835e953126425e28213e649e564e0e35986dab5eaa2638ce",
    "MacAddress": "02:42:ac:11:00:02",
    "IPv4Address": "172.17.0.2/16",
    "IPv6Address": ""
  }
},
"Options": {
  "com.docker.network.bridge.default_bridge": "true",
  "com.docker.network.bridge.enable_icc": "true",
  "com.docker.network.bridge.enable_ip_masquerade": "true",
  "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
  "com.docker.network.bridge.name": "docker0",
  "com.docker.network.driver.mtu": "1500"
},
"Labels": {}
```


Docker networking – ОСНОВНЫЕ КОМАНДЫ

```
docker network create --driver <networkType>  
<networkName>
```

Создать сеть

```
docker network create <networkName>
```

По умолчанию bridge

networkType = bridge | overlay | host | none

```
PS D:\DockerTest\volumes> docker network create --driver bridge itisgood_network  
2858dee1f6bb8ef93bbfb54abb7f9155a7d69136030120b2fdce09226c72d145
```

```
PS D:\DockerTest\volumes> docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
4d4ac804aafe	bridge	bridge	local
09daad94c2ad	host	host	local
2858dee1f6bb	itisgood_network	bridge	local
6908ce2ee335	none	null	local

Docker networking – ОСНОВНЫЕ КОМАНДЫ

```
docker run -it -d  
--network=<networkName> httpd
```

Запустить Docker-контейнер
в созданной сети

```
PS D:\DockerTest\volumes> docker run -it -d --network=itisgood_network httpd  
d3032441647501ff62ad1729c61982459bd27a2088349ce9beea34fd092c9b79
```

```
PS D:\DockerTest\volumes> docker network inspect itisgood_network
```

```
[  
  {  
    "Name": "itisgood_network",  
    "Id": "2858dee1f6bb8ef93bbfb54abb7f9155a7d69136030120b2fdce09226c72d145"
```

```
    "Containers": {  
      "06d393cdf3d00afea45db5b115cde89e477": {  
        "Name": "clever_lovelace",  
        "EndpointID": "a427971cf0ba78124",  
        "MacAddress": "02:42:ac:12:00:02"  
      },  
      "d3032441647501ff62ad1729c61982459bd27a2088349ce9beea34fd092c9b79": {  
        "Name": "cranky_mestorf",  
        "EndpointID": "ed616271a83cdca16"
```


Docker networking – ОСНОВНЫЕ КОМАНДЫ

`docker network disconnect <networkName>
<containerName>`

ОТКЛЮЧИТЬ СЕТЬ ОТ
КОНТЕЙНЕРА

```
PS D:\DockerTest\volumes> docker network disconnect itisgood_network cranky_mestorf
PS D:\DockerTest\volumes> docker network inspect itisgood_network
[
  {
    "Name": "itisgood_network",
    "Id": "2858dee1f6bb8ef93bbfb54abb7f9155a7d69136030120b2fdce09226c72d145",
```

```
"Containers": {
  "06d393cdf3d00afea45db5b115cde89e477b54": {
    "Name": "clever_lovelace",
    "EndpointID": "a427971cf0ba78124369",
    "MacAddress": "02:42:ac:12:00:02",
    "IPv4Address": "172.18.0.2/16",
    "IPv6Address": ""
  }
},
```

Docker networking – ОСНОВНЫЕ КОМАНДЫ

<code>docker network prune</code>	Удалить все сети, к которым не подключен ни один контейнер
<code>docker network rm <networkName></code>	Удалить сеть с именем <code>networkName</code>

Docker networking – три контейнера в одной сети

- > docker network create myNet
- > docker run -it -d --name A1 --network myNet alpine ash
- > docker run -it -d --name A2 --network myNet alpine ash
- > docker run -it -d --name A3 --network myNet alpine ash

```
PS D:\DockerTest\volumes> docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9abd32b1a31a	alpine	"ash"	39 seconds ago	Up 38 seconds		A3
bcccbba1abe3	alpine	"ash"	50 seconds ago	Up 49 seconds		A2
0fe2b43611d8	alpine	"ash"	About a minute ago	Up About a minute		A1

сети

подключиться к любому из контейнеров и пропинговать два других, используя имя контейнера

```
PS D:\DockerTest\volumes> docker container attach A2
/ # ping -c 2 A1
PING A1 (172.19.0.2): 56 data bytes
64 bytes from 172.19.0.2: seq=0 ttl=64 time=45.574 ms
64 bytes from 172.19.0.2: seq=1 ttl=64 time=0.075 ms

--- A1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.075/22.824/45.574 ms
/ # ping -c 2 A3
PING A3 (172.19.0.4): 56 data bytes
64 bytes from 172.19.0.4: seq=0 ttl=64 time=21.773 ms
64 bytes from 172.19.0.4: seq=1 ttl=64 time=0.084 ms

--- A3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.084/10.928/21.773 ms
```