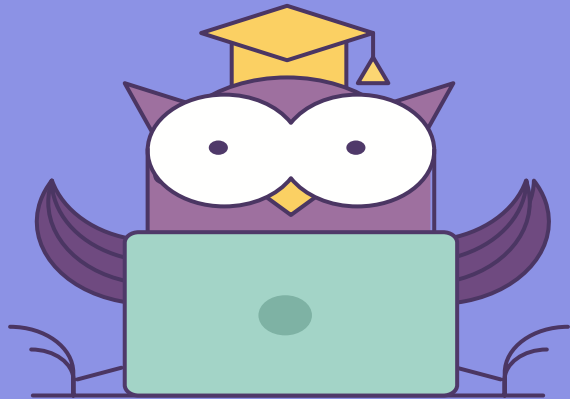





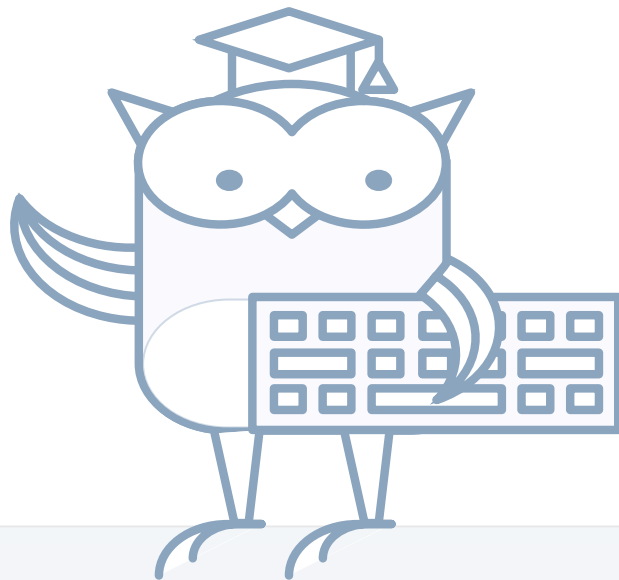
ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!
Ставьте  если все хорошо

Docker Dockerfile



- Контейнеризация, краткий обзор
- Компоненты docker
 - engine, cli, registry
- Сборка контейнеров, dockerfile
- Практика
 - build, run, up, down, pull, push

Существовало достаточно давно

Не было широкого распространения

В определенных случаях была заменена аппаратная виртуализация

Не столько про контейнеры (как технология)

Особенности:

- Абстракция от host-системы

- Легковесные изолированные окружения

- Общие слои файловой системы

- Компоновка и предсказуемость

- Простое управления зависимостями

- Дистрибуцию и тиражируемость, воспроизводимость

- Стандартизация описания окружения, сборки, деплоя

- 100% консистентная среда приложения

Docker - это не виртуальная машина!

Это приложение и его зависимости упакованные в окружение.

Принцип работы:

Namespaces

Cgroups

UnionFS

RunC

Изоляция окружения

Индивидуальный namespace для каждого контейнера

Pid, net, mount, etc.

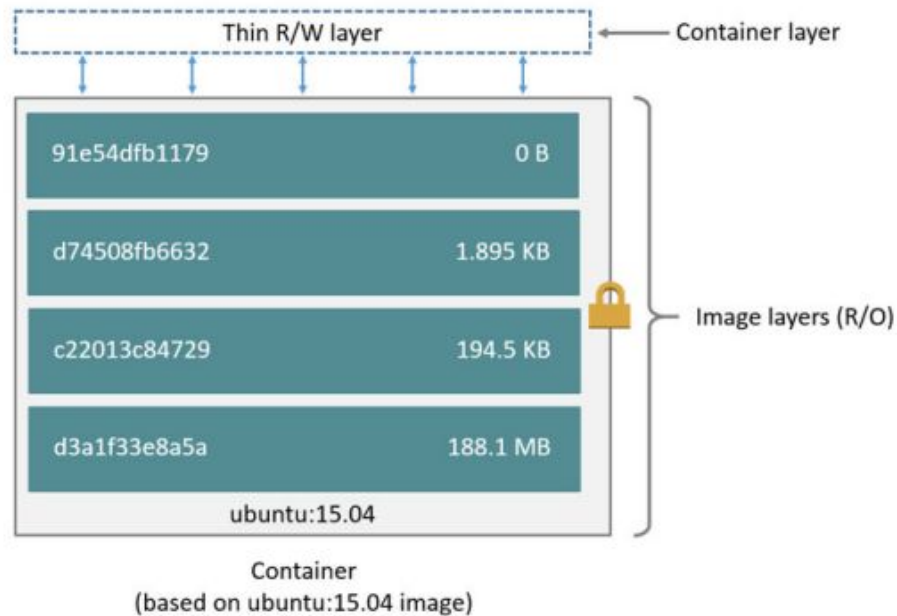
Namespace прекращает свое существование после окончания работы PID1

Использование контейнерами общих ресурсов

Ограничение ресурсов

CPU, memory, IO, etc.

Разделение по слоям
Переиспользование слоев
Overlays2

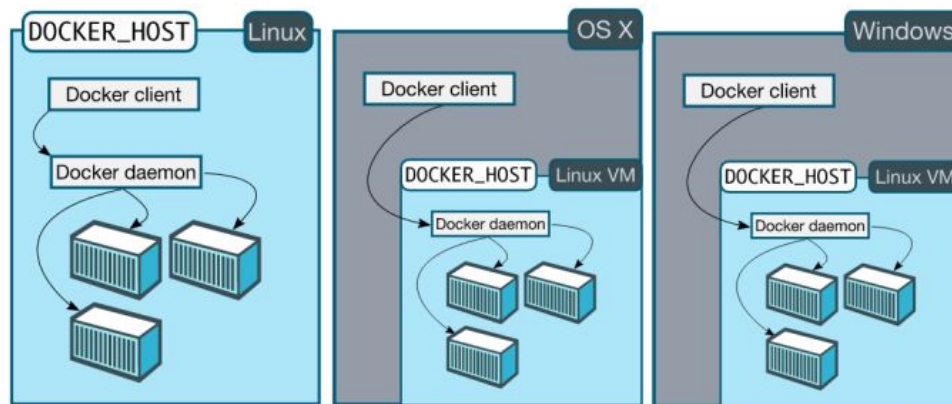


Daemon:

- предоставляет api
- управляет объектами
- взаимодействует с другими daemon`ми

cli:

- принимает команды от пользователя
- взаимодействует с api docker daemon



dockerd

containerd

cri-o

podman

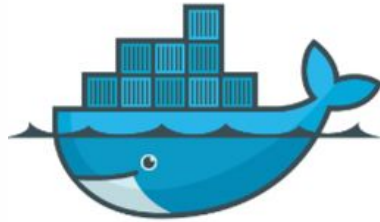
buildah (only for build)

Kata Containers (specific kernel)

rkt (RIP)

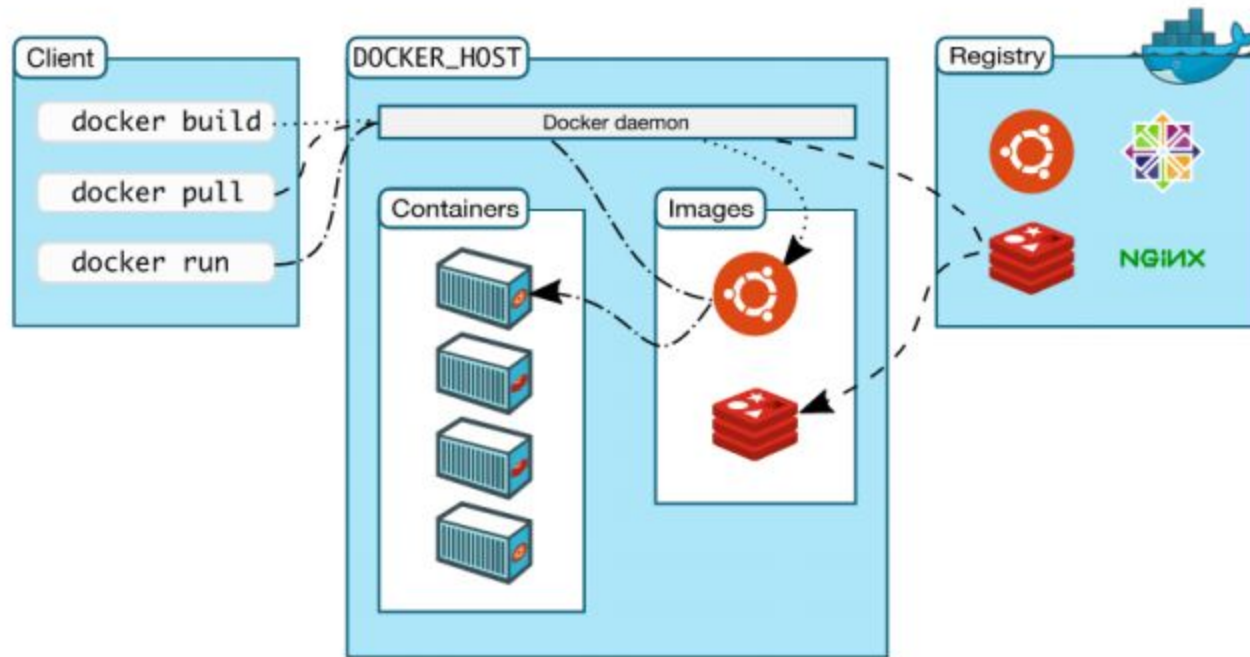
Так называемый приватный “репозиторий” для хранения image.

Docker hub, Private registry, Docker Store



GitLab

Docker engine



Вот [ссылка](#) на Dockerfile

Так хорошо?)

И вот тут [ссылка](#) на Dockerfile

Вот так хорошо :=)

ENV - переменные окружения

ARG - переменные во время сборки

COPY - скопировать файл или папку

ADD - скопировать файл или папку, скачать по ссылке, разархивировать архив

EXPOSE - документация

А как запустить то?0_o

CMD

ENTRYPOINT

Режимы работы:

shell

```
FROM alpine
```

```
ENTRYPOINT ping www.google.com
```

exec

```
FROM alpine
```

```
ENTRYPOINT ["ping", "www.google.com"]
```

Комбинированное использование

```
FROM alpine
ENTRYPOINT ["ls", "/usr"]
CMD ["/var"]
```

А если я хочу более сложную конструкцию для запуска?

Никто не запрещает использовать bash-скрипт

Подключаемые модули для управления сетью контейнеров

Native (встроенные в Docker)

Remote (сторонние)

Встроенные модули

1. None
2. Host
3. Bridge
4. Macvlan
5. Overlay

По IP-адресам

Docker Links (deprecated, только для default bridge сети)

Встроенный в Docker DNS

Внешний Service Discovery/DNS сервер

Для контейнера создается свой network namespace

У контейнера есть только loopback интерфейс

Сеть контейнера полностью изолирована

Контейнер использует network namespace хоста

Сеть не управляется самим Docker

Два сервиса в разных контейнерах не могут слушать один и тот же порт

Производительность сети контейнера равна
производительности сети хоста

Особенности default bridge network

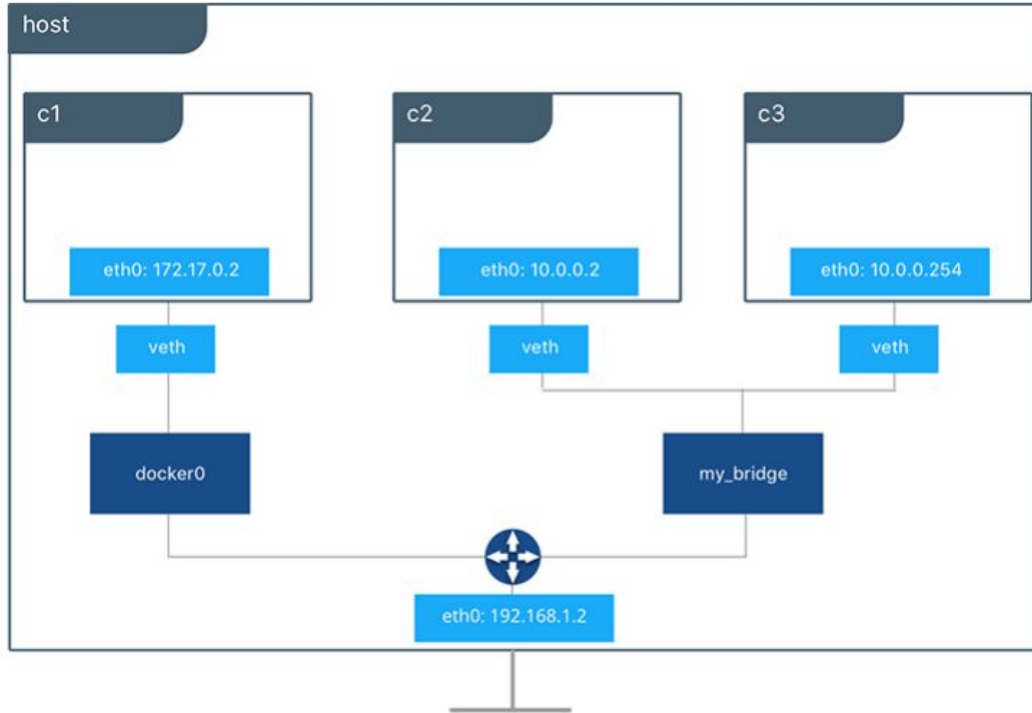
Назначается по умолчанию для контейнеров

Нельзя вручную назначать IP-адреса

Нет Service Discovery

Если нужно отделить контейнер или группу контейнеров
Контейнер может быть подключен к нескольким Bridge сетям
(без рестарта)
Работает Service Discovery
Произвольные диапазоны IP-адресов

Bridge



Привет из документации кстати [ТУТ](#) =)

Ну как бе вроде бы и норм 0_o

Более интересный вариант рассмотрим на практике ^_^

```
docker pull nginx
docker run -d -p 80:80 nginx
docker run -it -p 80:80 nginx
docker stop $(docker ps -aq)
docker stop cont_hash
docker build -t id/cont-name:ver .
docker push id/cont-name:ver
```