Docker контейнеры

Не забудь включить запись!

План

- Под капотом Docker'a
- Работа с данными(Volumes)
- Moby, Docker EE, Datacenter, Docker machine
- Сборка образа

Под капотом

- Namespaces
- Cgroups
- UnionFS

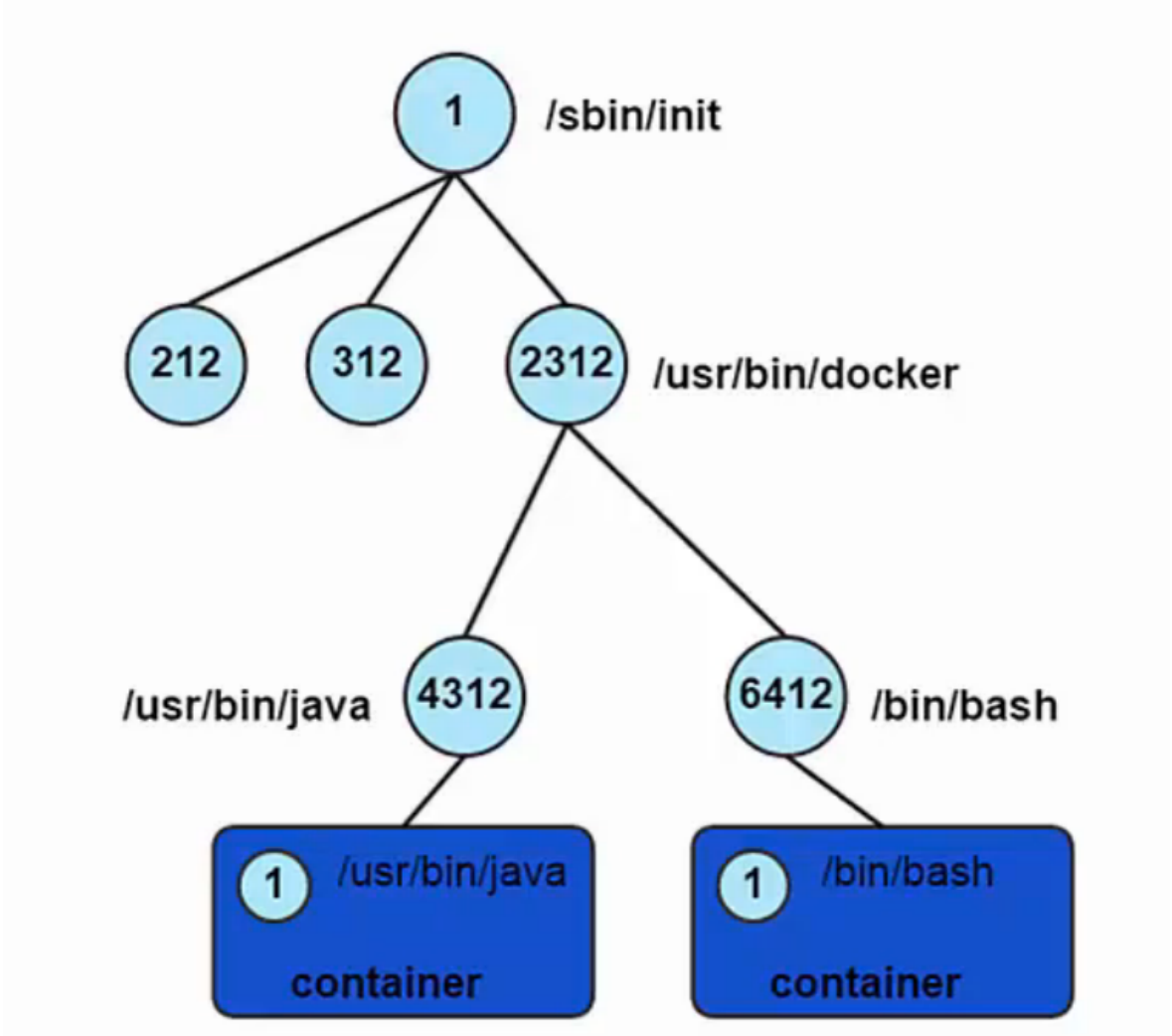
Namespaces

- PID
- mnt
- net
- uts
- IPC
- user

PID namespace

- Процессы внутри pid namespace'а видят только процессы из этого же nampespace'а
- Каждый pid namespace имеет свою нумерацию процессов (начиная с 1)
- **Когда процесс с pid 1 умирает, то умирает весь namespace**
- PID namespace'ы могут быть вложенными

PID namespace



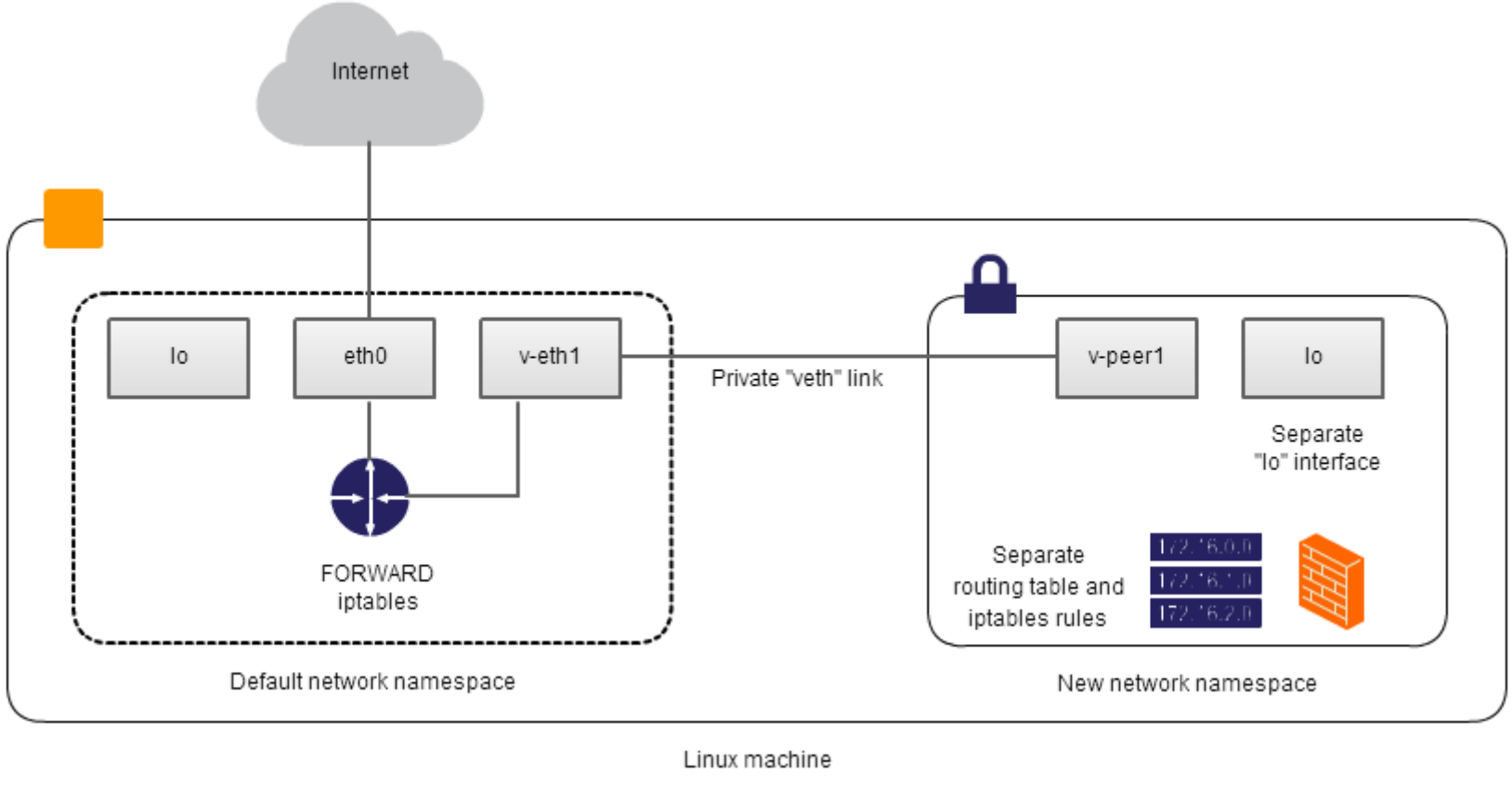
net namespace

- Процессы в net namespace'е имеют свой собственный сетевой стек, а именно:
  - Сетевые интерфейсы (включая lo)
  - Таблицу маршрутизации
  - Правила iptables
  - Socket'ы
- Можно перемещать сетевые интерфейсы между namespace'ами

net namespace

- Создаются два виртуальных сетевых интерфейса
- Eth0 внутри контейнера
- VethXXX на хост системе
- Все vethXXX соединены в один bridge-интерфейс (docker0)
- Флаг --net

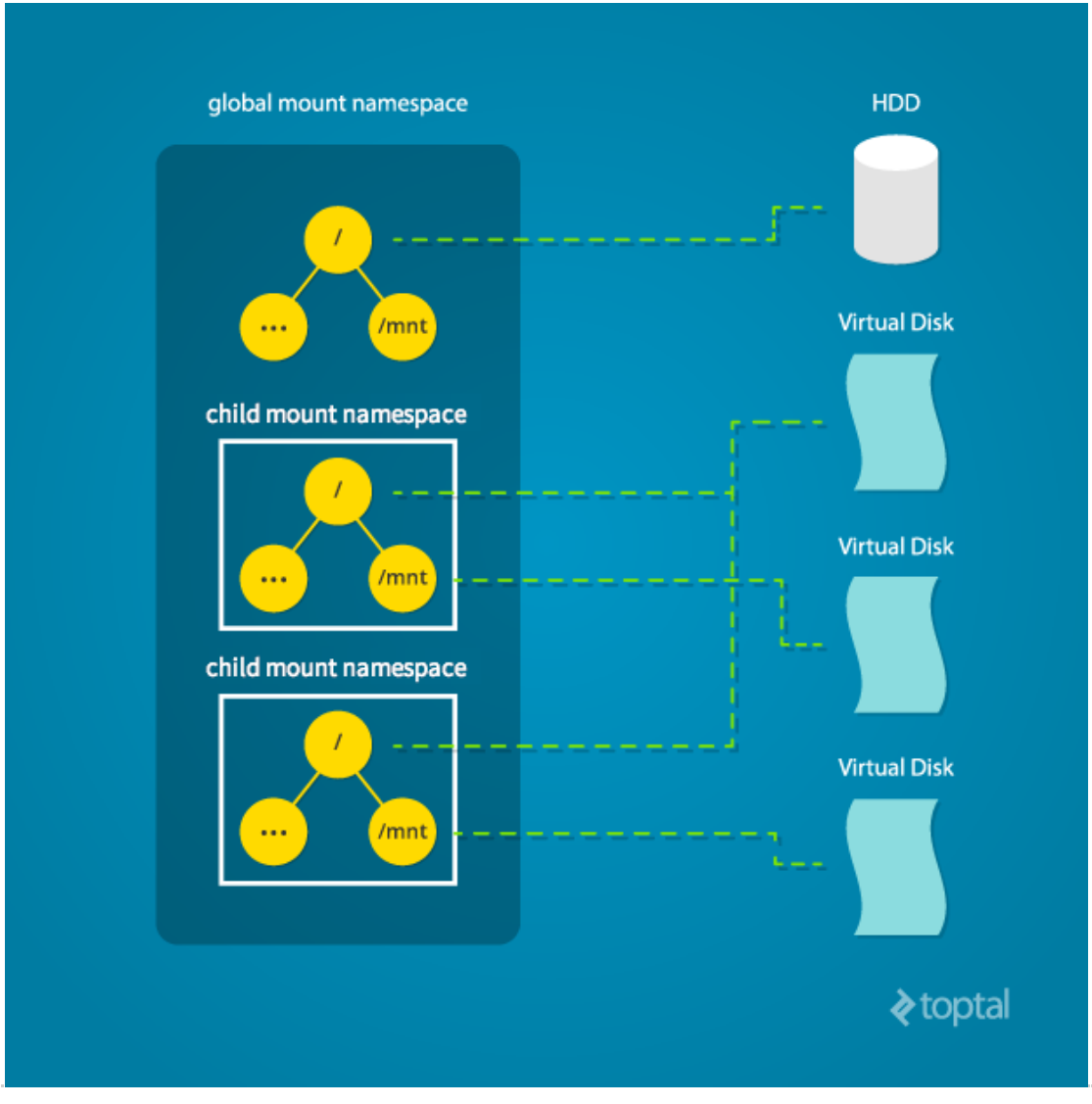
net namespace



mnt namespace

- Процессы могут иметь свой собственный root (аля chroot)
- У процессов могут быть свои приватные "Точки монтирования" (mounts)
  - /tmp
  - /proc, /sys
- "Точки монтирования" (mounts) могут быть приватными, а могут быть доступны в нескольких namespace'ах

mnt namespace



uts namespace

- Hostname
- Domain name

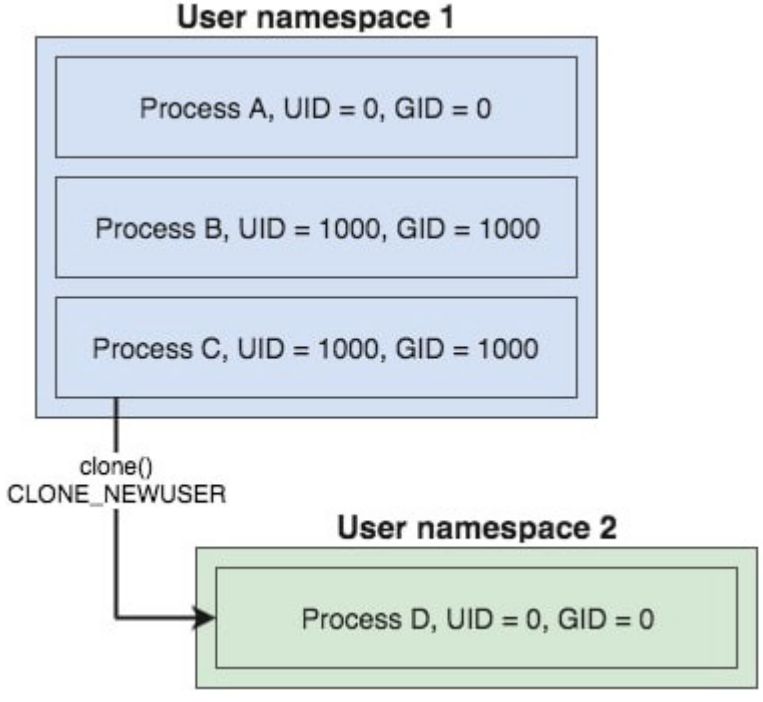
IPC namespace

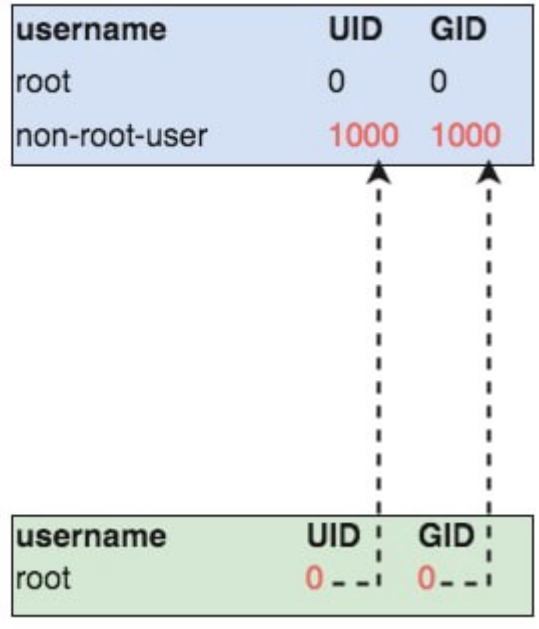
- Процессы или группы процессов могут иметь свои наборы:
  - IPC семафоров
  - IPC очередей сообщений
  - IPC совместно доступной памяти

user namespaces

- UID и GID внутри контейнеров отображаются в другом диапазоне UID и GID
- Мера безопасности (безопаснее чем флаг -u или USER директива)
- Влияет на все контейнеры и образа
- Для включения требует настройки docker-engine
- Можно выборочно отключать с помощью --userns=host

user namespaces





user namespaces

При включении user namespaces происходит

- Существующие образа и контейнеры не будут работать
- Доступ к ним закрывается (они пропадают из области видимости docker client)
- Новые контейнеры и образа помещаются в отдельную директорию
- Если хост был часть Swarm'a, то он выпадает из него
- Все появляется снова при отключении user namespaces

Namespaces

Нам может потребоваться выходить за рамки своих namespace'ов для:

- управления хостом
- управления другими контейнерами
- мониторинга

Namespaces

Сравните вывод:

- docker run --rm -ti tehbilly/htop
- docker run --rm --pid host -ti tehbilly/htop

cgroups

- Ограничивает доступ к ресурсам (в т.ч. устройствам)
- Ограничивает доступ к системных вызовам

Privileged

- docker run --privileged
- docker run --cap-add=NET_ADMIN
- Предоставляет доступ сравнимый с доступом обычного процесса

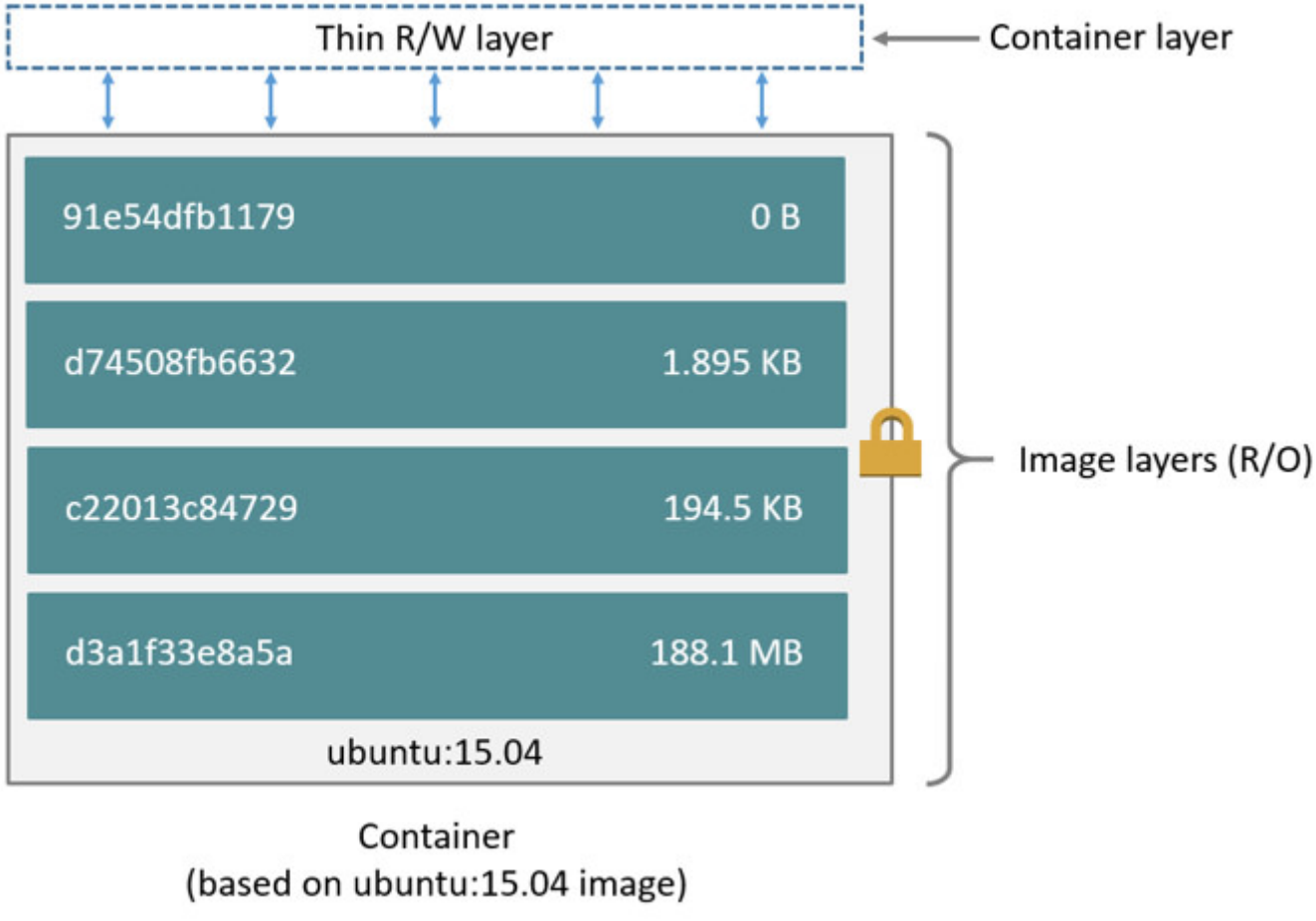
Хранение данных в Docker

- Storage (storage drivers)
- Data Volumes (volume drivers)

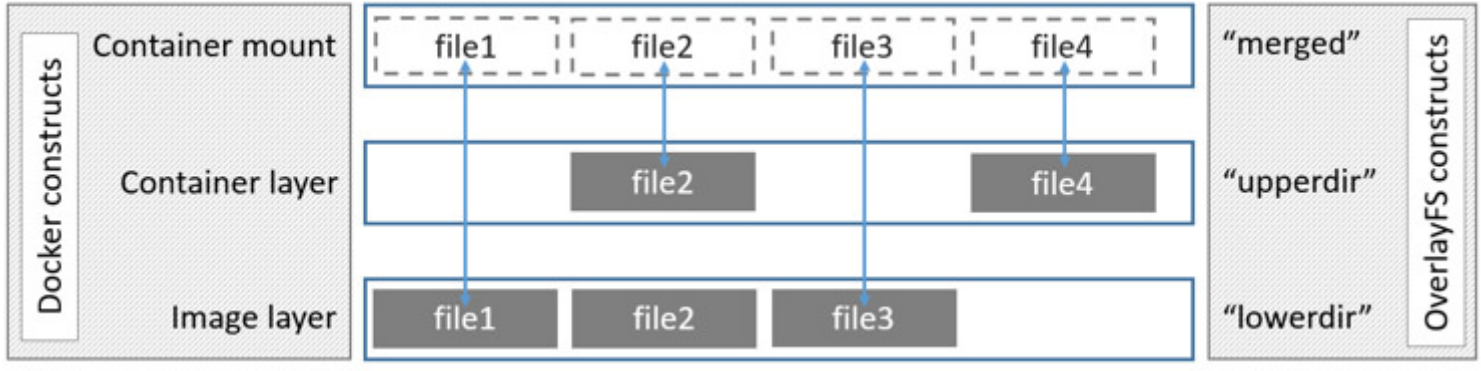
Storage

- Обеспечивает хранение слоев образов
- Обеспечивает слой для контейнера
- Драйвер выбирается в зависимости от потребностей (но чаще нет необходимости)

Storage


Thin R/W layer
Container layer
91e54dfb1179
0 B
d74508fb6632
1.895 KB
c22013c84729
194.5 KB
d3a1f33e8a5a
188.1 MB
ubuntu:15.04
Image layers (R/O)
Container
(based on ubuntu:15.04 image)

Storage


Docker constructs
Container mount
Container layer
Image layer
file1
file2
file3
file4
file2
file4
file1
file2
file3
"merged"
"upperdir"
"lowerdir"
OverlayFS constructs

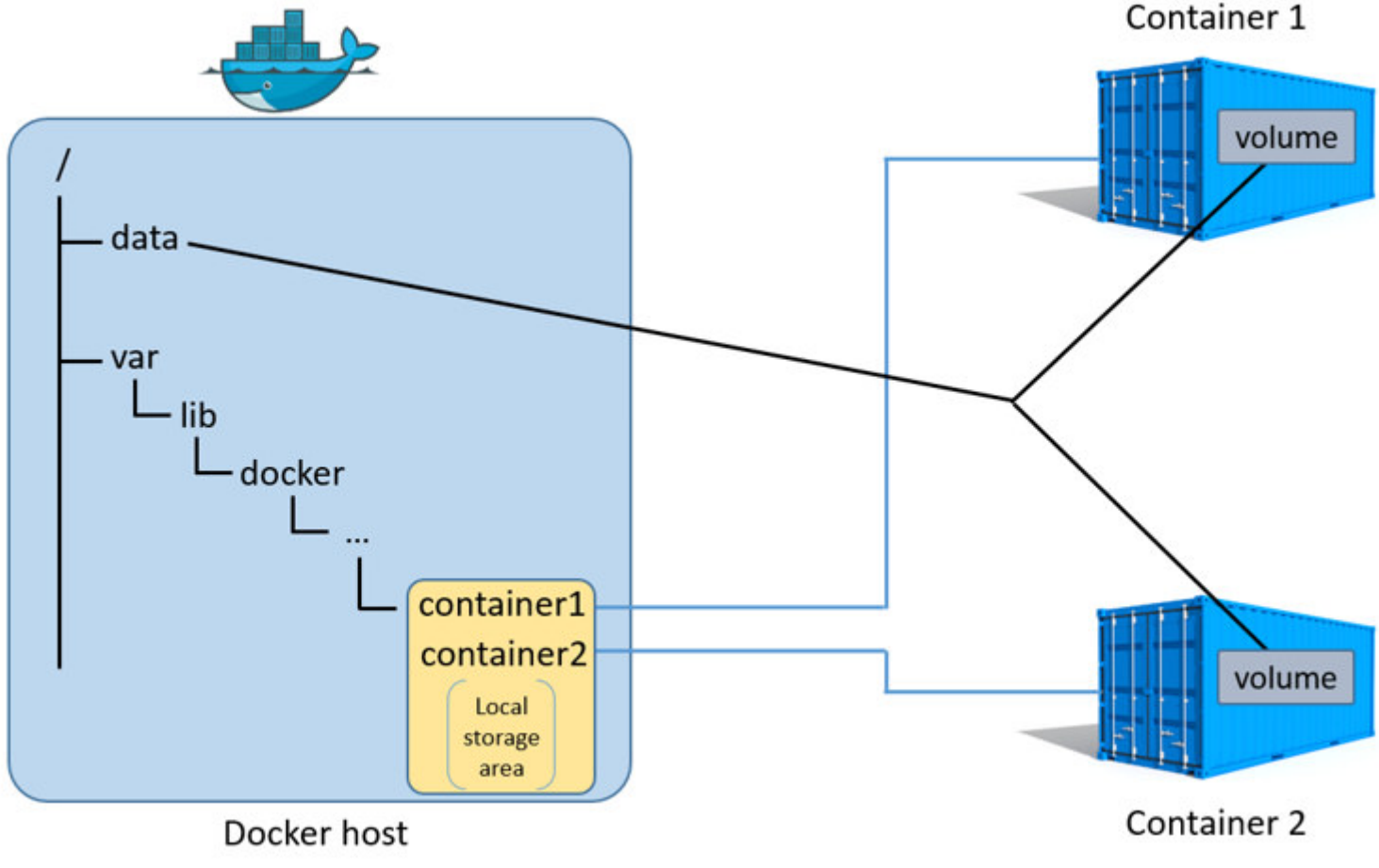
Storage

- Данные могут быть потеряны вместе с остановкой контейнера
- Верхний слой (RW-слой) тесно связан с контейнером
- Меньше производительность по отношению к data volumes

Data Volumes

Позволяют отделить жизненный цикл данных, которые он в себе хранит, от жизни самого контейнера, который эти данные создал

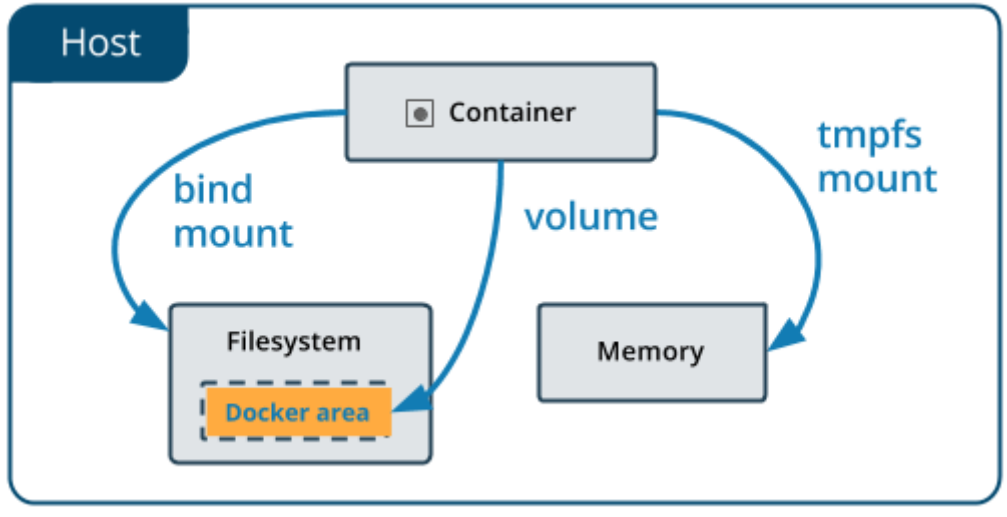
Data Volumes



Типы Volumes

- Volumes - тома управляемые Docker'ом. Другие процессы не должны иметь к ним доступ
- Bind mount - директории на файловой системе. Любой процесс может получить к ним доступ
- tmpfs - тома расположенные в памяти хоста. Никогда не записываются на диск

Где хранятся Volumes



Когда нужны Volumes

- Доступ к данным из нескольких контейнеров
- Когда неизвестна файловая структура на хосте
- Когда храним данные удаленно
- При миграции с хоста на хост(архивом, распределенной фс, оркестратором)

Bind mounts

- При совместном с хостом использовании конфигурации (/etc/resolv.conf)
- Совместный с хостом доступ к данным (исходному коду, артефактам)
- При известной структуре файловой системы хоста

tmpfs

- Когда данные не должны сохранятся на хосте или в контейнере

Local vs global volumes

- Global volumes можно подключить к контейнеру на любой ноде (учитывая ограничения драйверов)
- Global volumes требуют дополнительных плагинов
- Docker не включает в себя global volumes драйвера по умолчанию

Stateful сервисы

- Нужно ли вам поддерживать stateful сервисы?
- Умеет ли сервис падать?
- Все данные находятся на volume'ах
- Лучше позаботиться о резервных копиях и распределенных хранилищах.
- Свежезапущенный контейнер может получить доступ к volume'у
- Проверяйте!

Экосистема Docker

- Docker CE
- Docker registry / Hub / Cloud / Store
- Moby
- Docker EE
- Docker Datacenter
- Docker machine

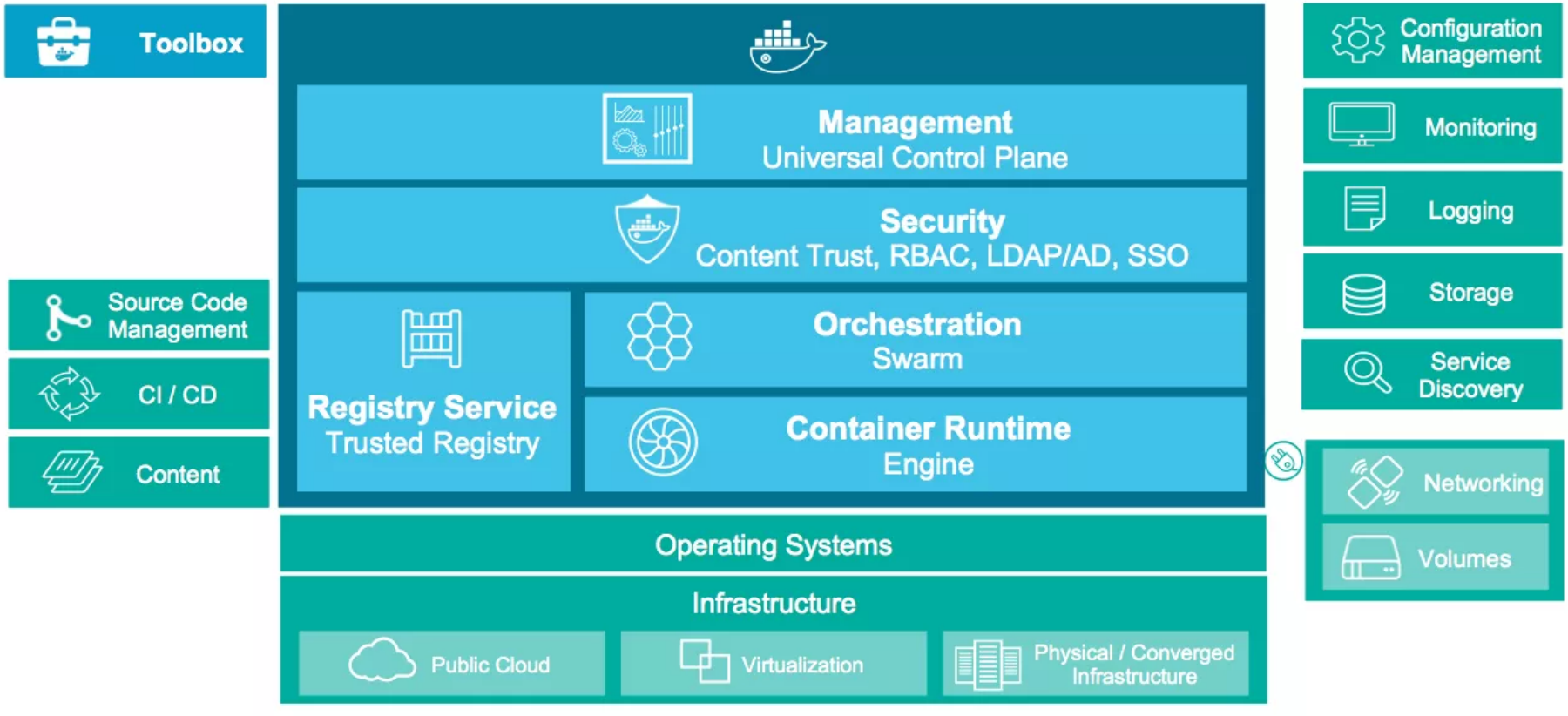
Что такое Moby?

- Фреймворк с
  - Компонентами для контейнеризации
  - Инструменты для сборки
- Теперь докер сделан на moby, а не монолитно
- Сам Moby мало кому нужен

Docker EE

- Стоит денег
- Сертифицированные дистрибутивы, образа и плагины
- Сканер безопасности для образов
- Продвинутое управление образами и контейнерами

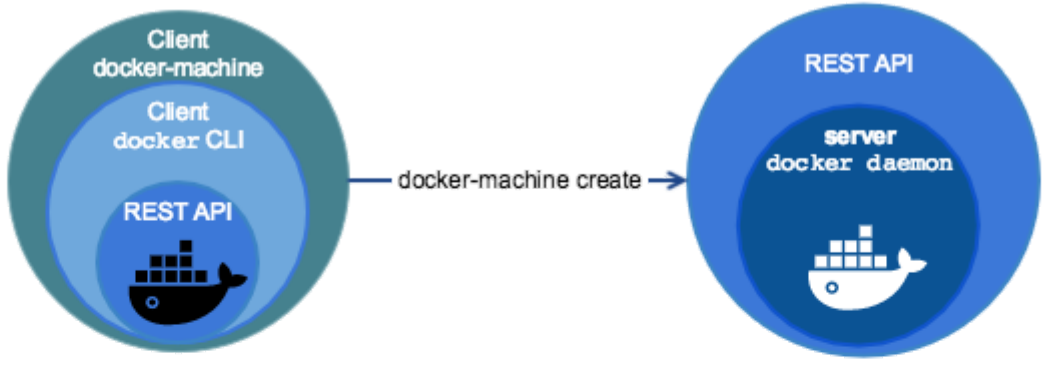
Docker Datacenter



Docker-machine

- Встроенный в докер инструмент для создания хостов и установки на них docker engine. Имеет поддержку облаков и систем виртуализации (Virtualbox, GCP и др.)
- Команда создания - это docker-machine create <имя>. Имен может быть много, переключение между ними через eval $(docker-machine env <имя>). Переключение на локальный докер - eval $(docker-machine env --unset)
- Все докер команды, которые запускаются в той же консоли после eval $(docker-machine env <имя>) работают с удаленным докер демоном в GCP.

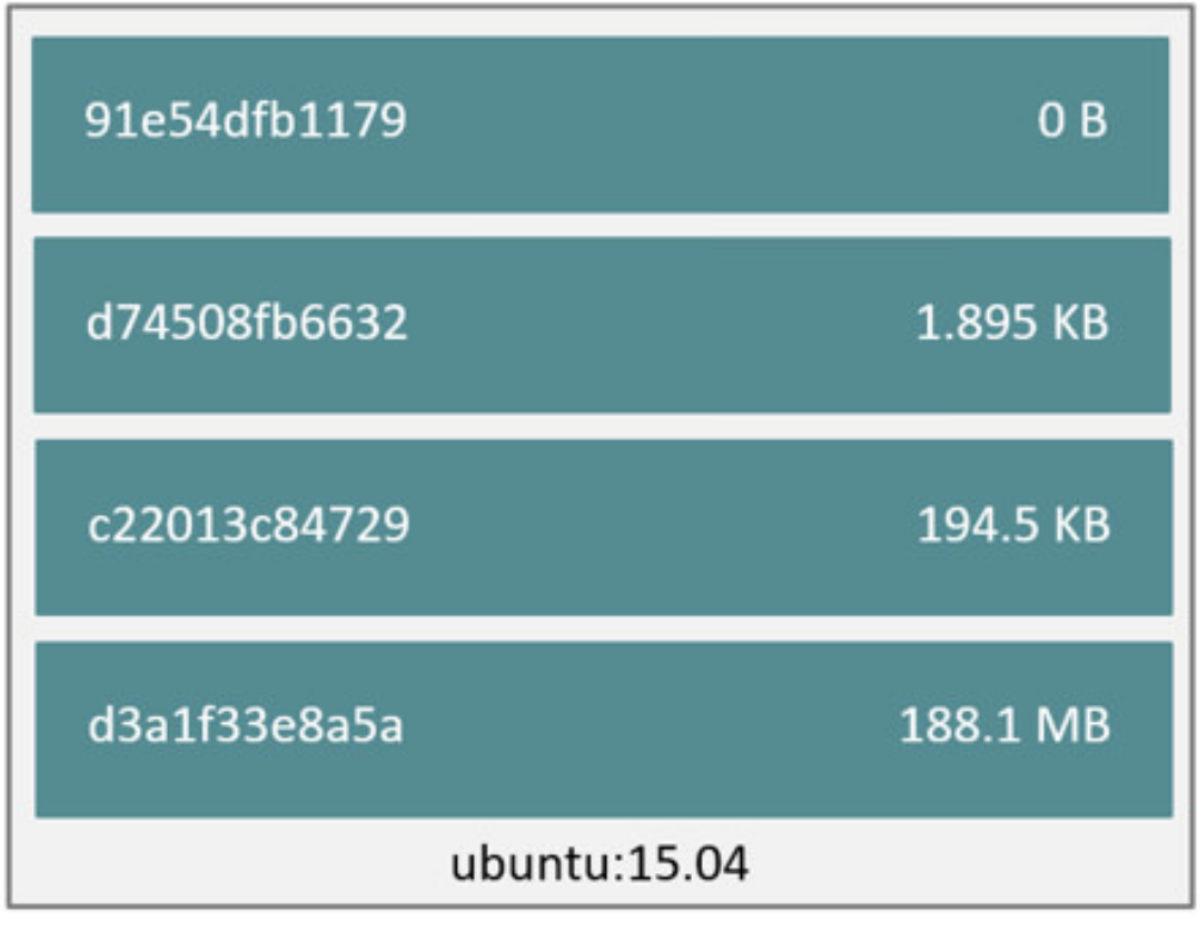
Docker-machine



Dockerfile

- Текстовый файл с build инструкциями
- Инструкции декларативно описывают image
- каждая инструкция – промежуточный image
- сам build делает docker daemon

Docker images



Image

RUN …

RUN …

ADD/COPY

Пишем Dockerfile

script.sh:

```
#!/bin/bash
echo "Hello declarative world!"
```

Dockerfile:

```
FROM ubuntu:16.04

LABEL maintainer="nikita@express42.com"
LABEL version="1.0"
LABEL description="Simple docker image \
with your script"

COPY ./script.sh /
```

Пишем Dockerfile

```
>docker build  .
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM ubuntu:16.04
 ---> e4415b714b62 <-- ubuntu 16.04 image
Step 2 : MAINTAINER Nikita Borzykh <nikita@express42.com>
 ---> Running in 1b7a0a847039 <-- Запуск в контейнере
 ---> ef8a151a8dda <-- промежуточные images
Removing intermediate container 1b7a0a847039 <-- контейнер удаляется после
выполнения
Step 3 : LABEL version "1.0"
 ---> Running in cb895217fe13
 ---> 9ac8deec0eec
Removing intermediate container cb895217fe13
Step 4 : LABEL description "Simple docker image with your script"
 ---> Running in a11744ec6d1d
 ---> 366de88a2967
Removing intermediate container a11744ec6d1d
Step 5 : COPY ./script.sh /
 ---> 550d8bcd5b57 <-- финальный image
Removing intermediate container 073f57b87e1e
Successfully built 550d8bcd5b57
```

Пишем Dockerfile

```
>docker images
REPOSITORY      TAG           IMAGE ID          CREATED            SIZE
<none>          <none>        50d8bcd5b57      48 minutes ago    128.1 MB
```

```
>docker images -a
REPOSITORY        TAG           IMAGE ID          CREATED            SIZE
<none>            <none>        550d8bcd5b57      48 minutes ago    128.1 MB
<none>            <none>        366de88a2967      52 minutes ago    128.1 MB
<none>            <none>        9ac8deec0eec      52 minutes ago    128.1 MB
<none>            <none>        ef8a151a8dda      52 minutes ago    128.1 MB
```

Пишем Dockerfile

```
>docker run -it 550d8bcd5b57
root@1f8b7931cbdf:/#/script.sh
bash: /script.sh: Permission denied
```

Пишем Dockerfile

Dockerfile:

```
FROM ubuntu:16.04

MAINTAINER Nikita Borzykh <nikita@express42.com>
LABEL version="1.0"
LABEL description="Simple docker image \
with your script"

COPY ./script.sh /

RUN chmod a+x /script.sh
```

Пишем Dockerfile

```
>docker build  .
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM ubuntu:16.04
 ---> e4415b714b62
Step 2 : MAINTAINER Nikita Borzykh <nikita@express42.com>
 ---> Using cache <-- команды не выполняются заново
 ---> ef8a151a8dda
Step 3 : LABEL version "1.0"
 ---> Using cache
 ---> 9ac8deec0eec
Step 4 : LABEL description "Simple docker image with your script"
 ---> Using cache
 ---> 366de88a2967
Step 5 : COPY ./script.sh /
 ---> Using cache
 ---> 550d8bcd5b57
Step 6 : RUN chmod a+x /script.sh
 ---> Running in dae3ee06743b <-- запустилась только новая команда
 ---> cd6c495df6c2 <-- создался новый image
Removing intermediate container dae3ee06743b
Successfully built cd6c495df6c2
```

Пишем Dockerfile

```
>docker run -it cd6c495df6c2
root@1f8b7931cbdf:/#/script.sh
Hello declarative world!
```

Пишем Dockerfile

Dockerfile:

```
FROM ubuntu:16.04

MAINTAINER Nikita Borzykh <nikita@express42.com>
LABEL version="1.0"
LABEL description="Simple docker image \
with your script"

COPY ./script.sh /
RUN chmod a+x /script.sh

CMD "/script.sh"
```

```
>docker build
```

CMD варианты

- CMD script param param <-- shell form
- CMD ["script", "param", "param"] <-- exec form

Dockerfile examples

```
# Install a more up to date mongodb than what is included in the default ubuntu
repositories.

FROM ubuntu:16.04

RUN apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
RUN echo "deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist
10gen" | tee -a /etc/apt/sources.list.d/10gen.list
RUN apt-get update
RUN apt-get -y install apt-utils
RUN apt-get -y install mongodb-10gen

#RUN echo "" >> /etc/mongodb.conf

CMD ["/usr/bin/mongod", "--config", "/etc/mongodb.conf"]
```

Полезные ссылки

Docker Documentation

Docker на bash

Книга "Э. Моуэт Использование Docker" (местами устарела)

Best practices написания Dockerfiles

Анонс Docker EE 2.0