# Docker-контейнеры

# План

- Под капотом Docker'а
- Работа с данными
- Docker экосистема

# Под капотом

- Namespaces
- Cgroups
- UnionFS

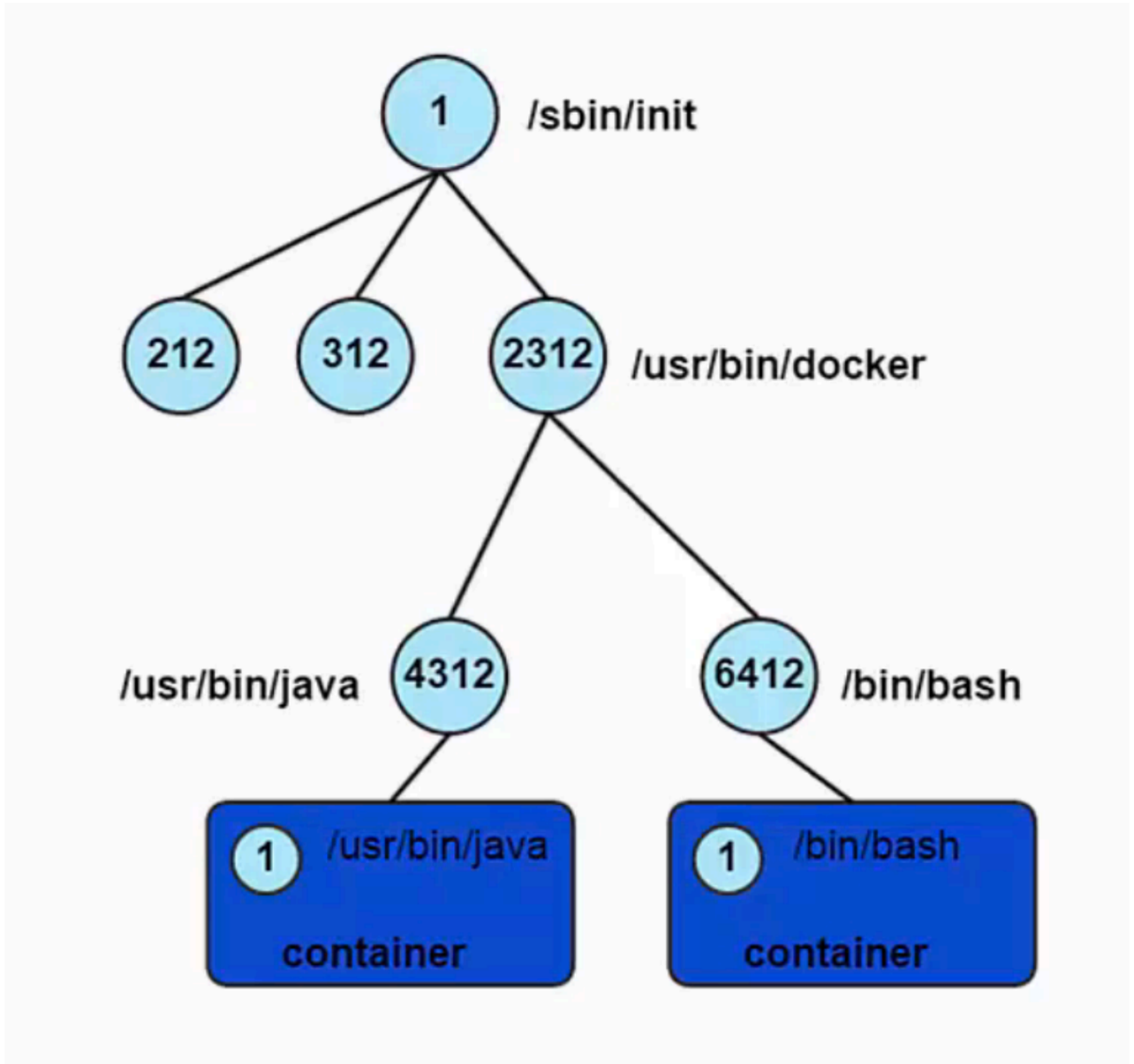
# Namespaces

- 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



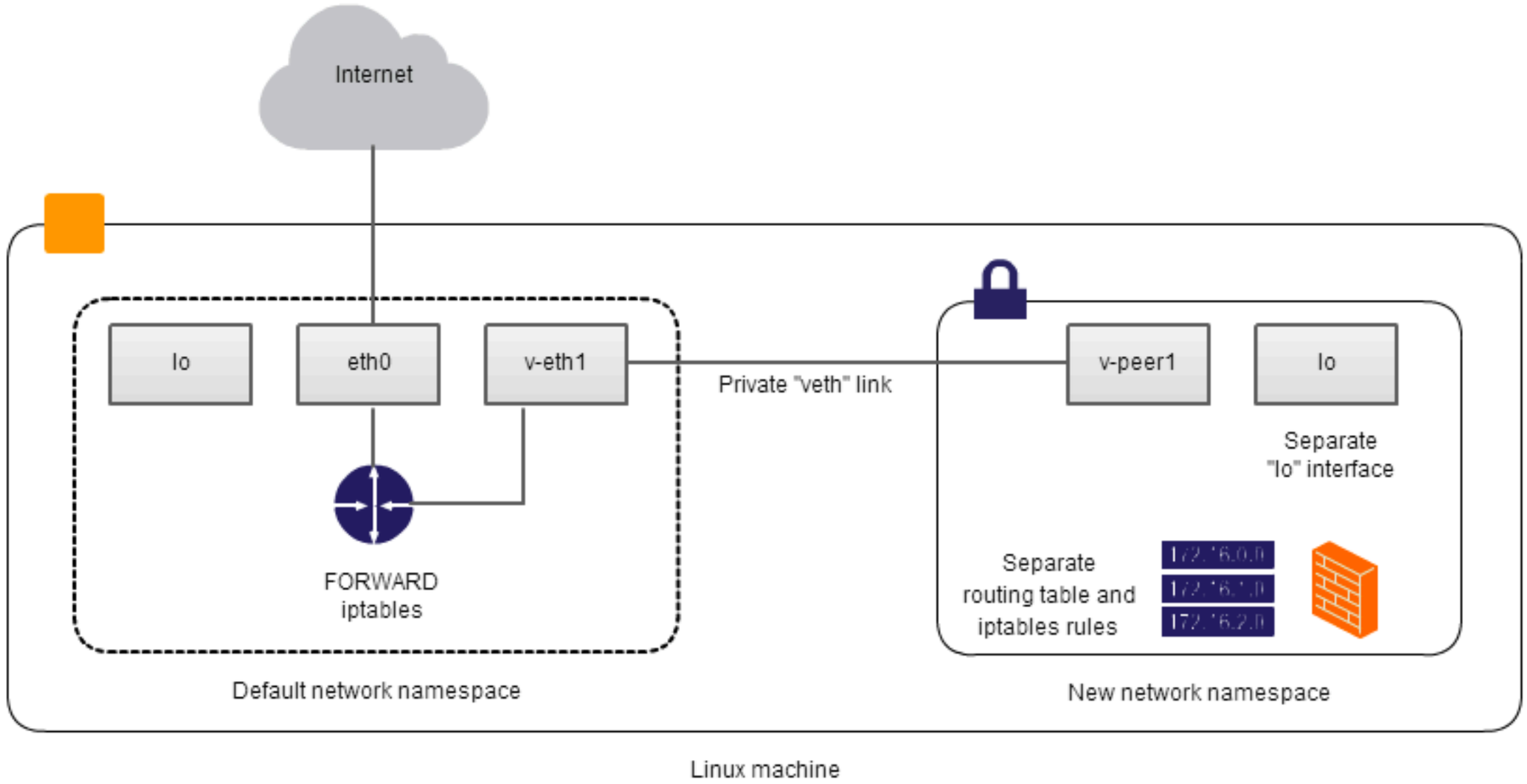
# Net Namespace

- 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

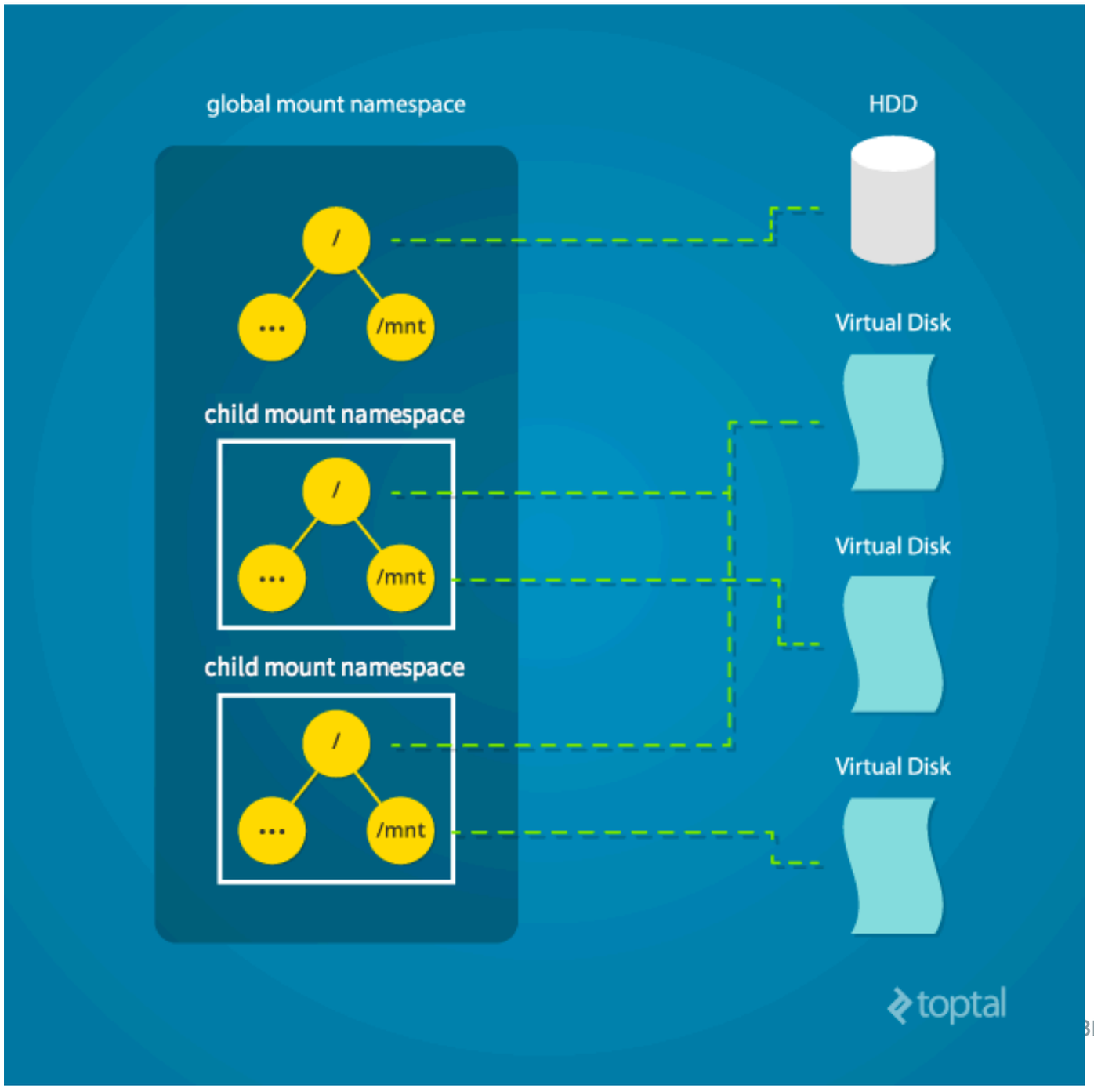
# Net Namespace



# 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'ax

# 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

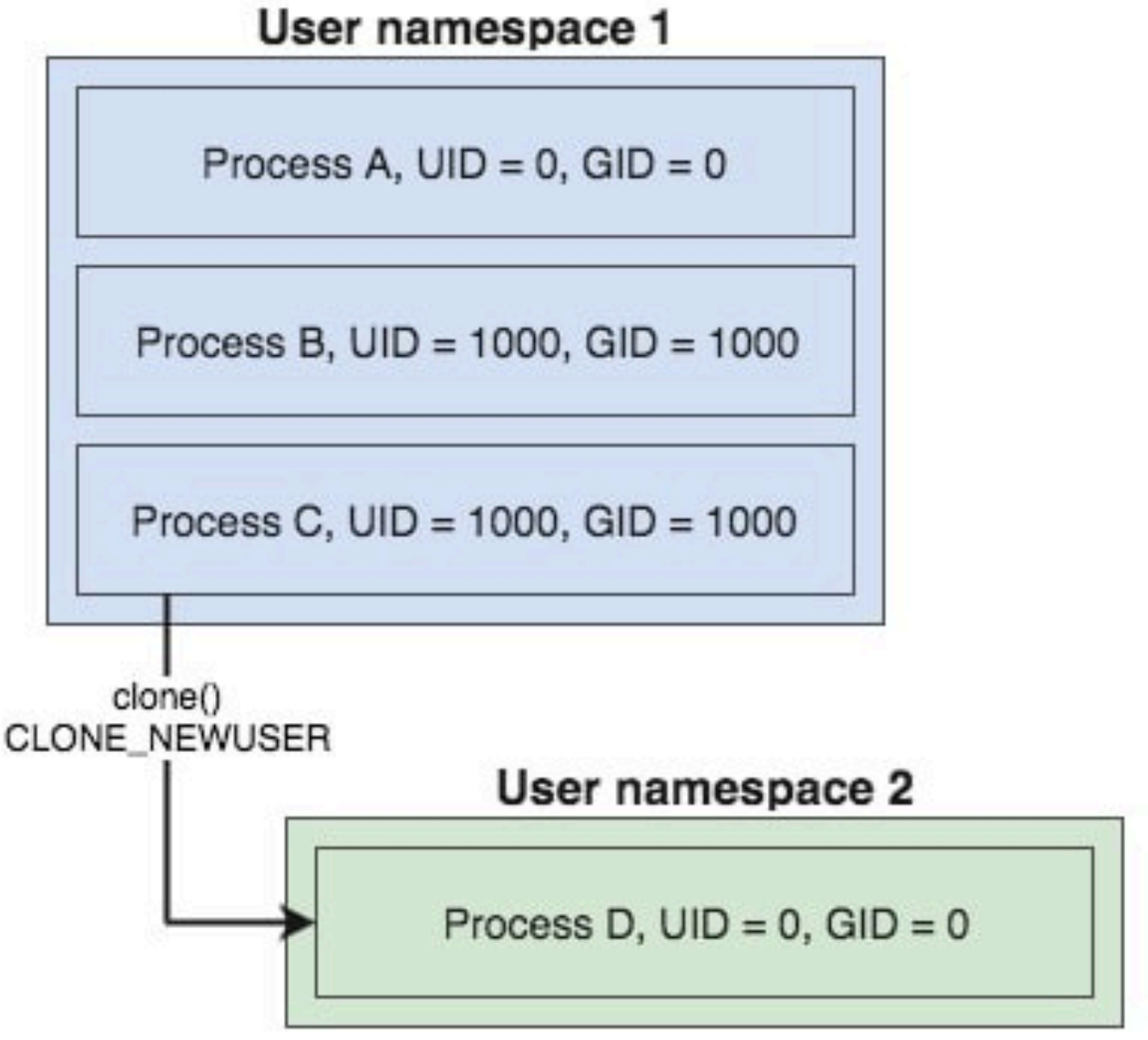
# User Namespaces

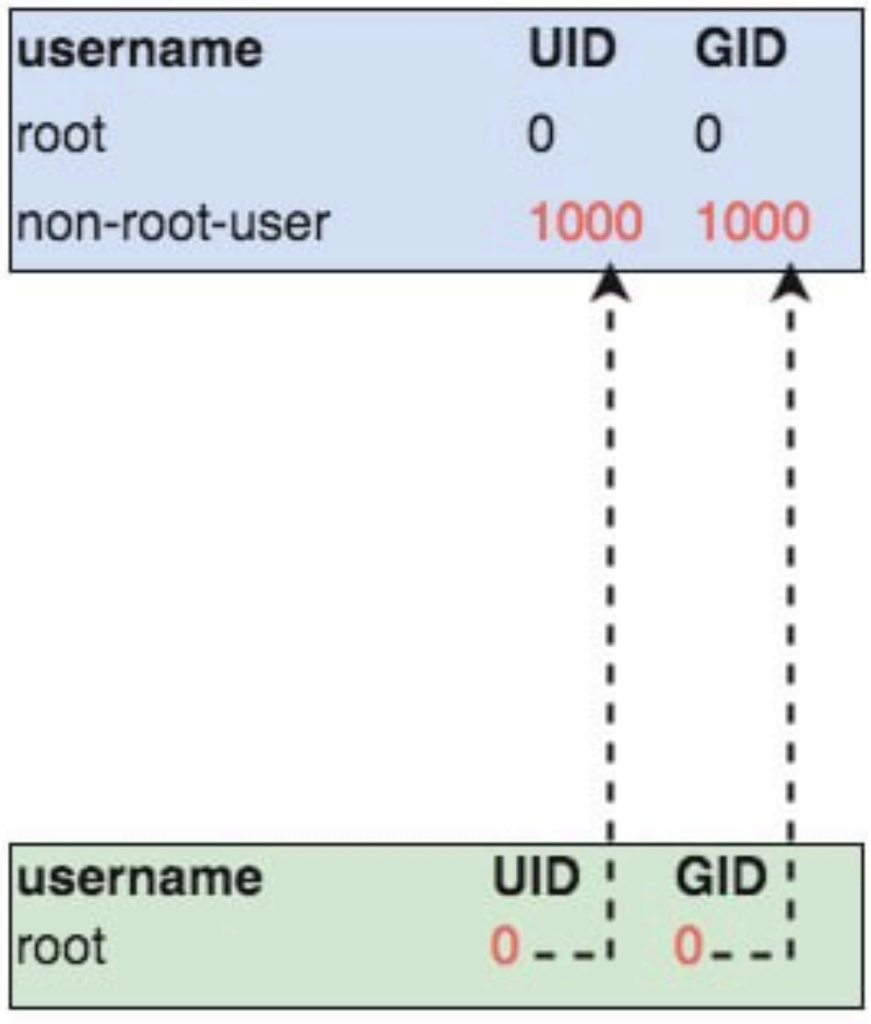
- UID и GID внутри контейнеров отображаются в другом диапазоне UID и GID
- Мера безопасности (безопаснее чем флаг -u или USER директива)
- Влияет на все контейнеры и образа
- Для включения требует настройки docker-engine
- Можно выборочно отключать с помощью --userns=host

# User Namespaces

- 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

# User Namespaces



# Namespaces

- Нам может потребоваться выходить за рамки своих namespace'ов для:
  - управления хостом
  - управления другими контейнерами

# 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

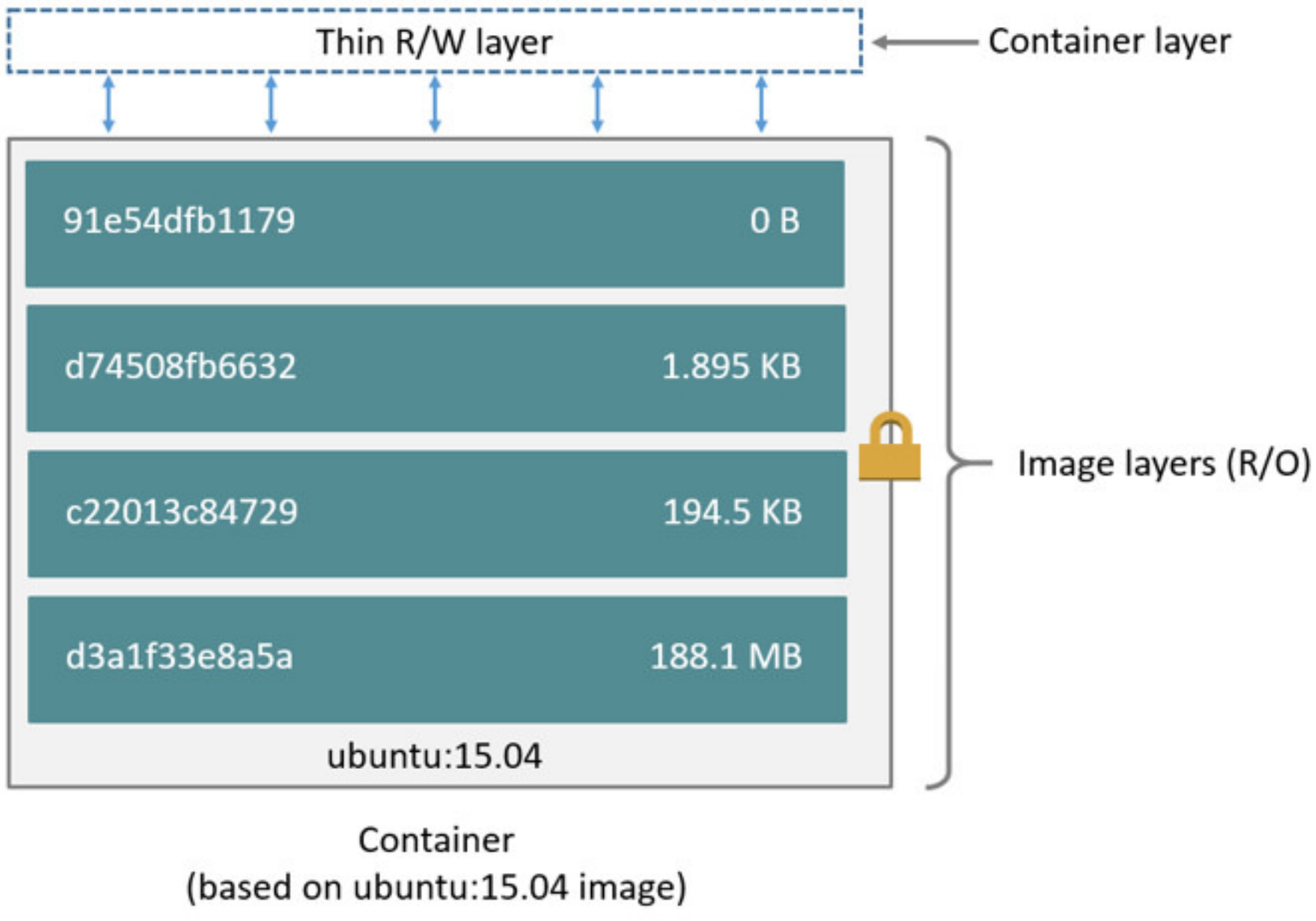
# Хранение данных в Docker

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

# Storage

- Обеспечивает хранение слоев образов
- Обеспечивает слой для контейнера
- Можно выбрать в зависимости от потребностей и возможностей (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)

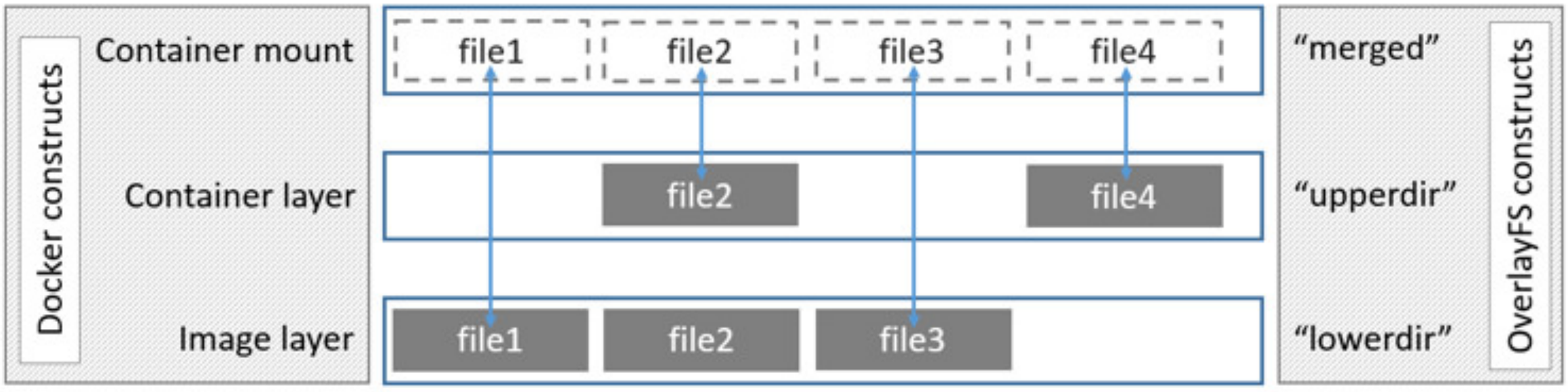
# Storage

- Данные могут быть потеряны вместе с остановкой контейнера

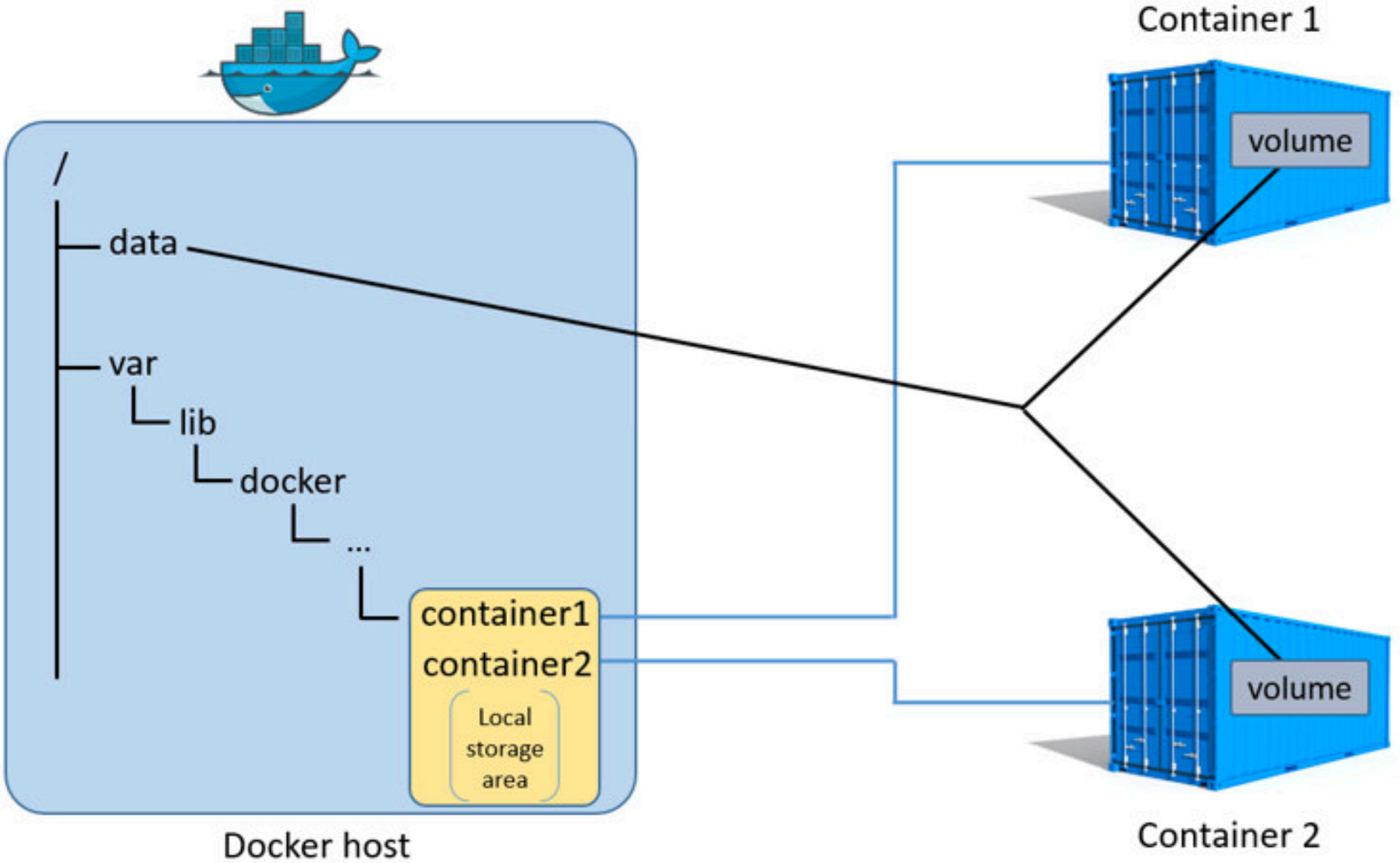
- Верхний слой (RW-слой) тесно связан с контейнером

- Меньше производительность по отношению к data volumes

# Storage


Container mount
Docker constructs
Container layer
Image layer
file1
file2
file3
file4
file2
file4
file1
file2
file3
“merged”
“upperdir”
“lowerdir”
OverlayFS construc

# Data Volumes



# Data Volumes

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

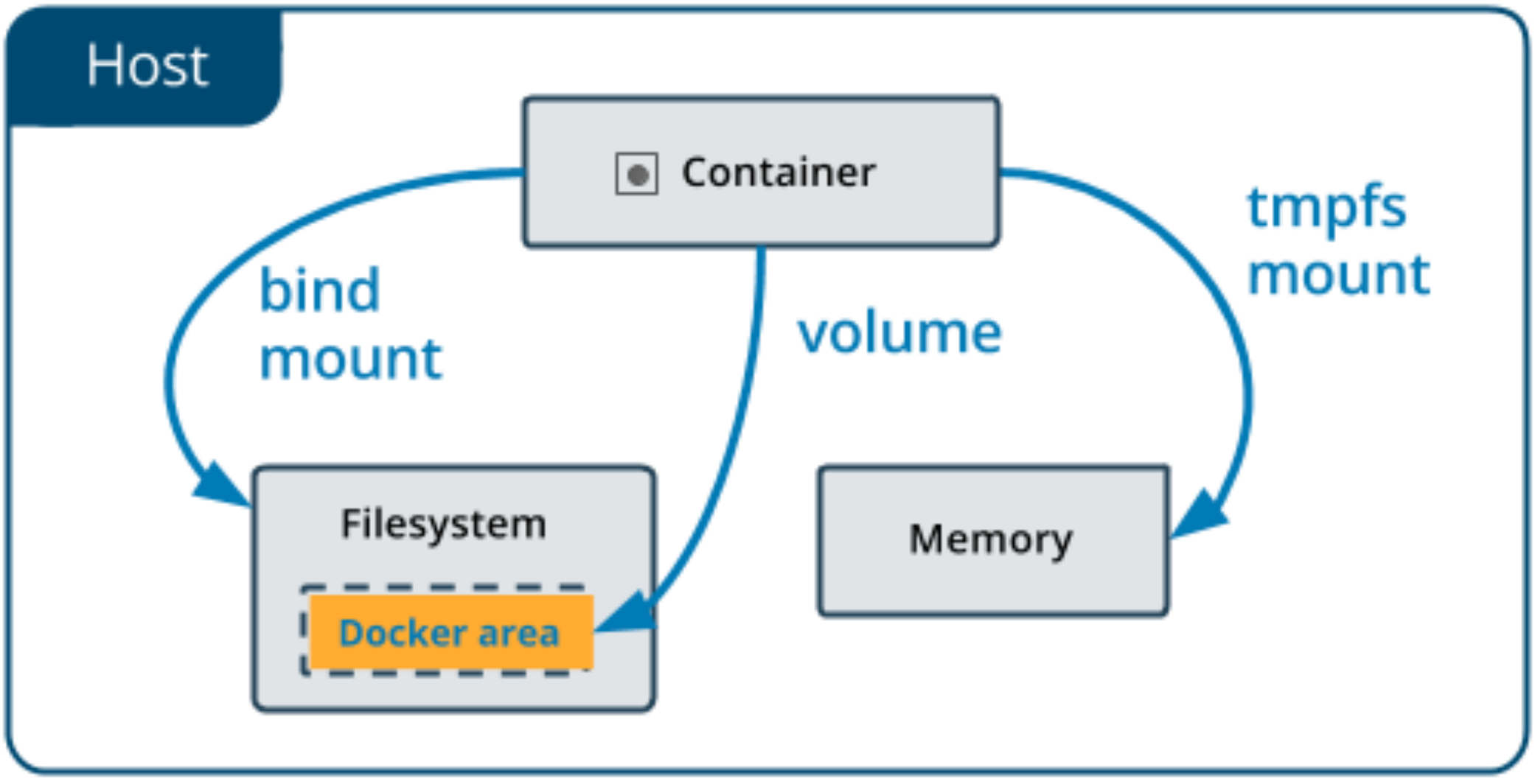
# Data Volumes

- 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

# Data Volumes



# Docker Volumes

- Доступ к данным из нескольких контейнеров
- Когда неизвестна файловая структура на хосте
- Когда храним данные удаленно
- Предпочтительнее при миграции с хоста на хост
- Бывают именнованые и неименованые

# 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

- ~~Хак вместо ramdisk'a~~

# 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'ax
- Лучше позаботиться о резервных копиях и распределенных хранилищах.
- Свежезапущенный контейнер может получить доступ к volume'y
- Проверяйте!

# Moby

- Это новый Docker?

# Docker CE

- Бесплатный
- Доступен для Mac, Windows и основных Linux

# Docker registry

- Контроль за местом хранения образов
- Полная ответственность за процесс распространения образов
- Тесная интеграция с внутренним процессом разработки

# Docker hub

- SAAS Docker registry
- Интегрирован с другими сервисами Docker
- Интегрируется с CI системами (webhook)
- Подходит для небольших организаций

# Docker Cloud

- Docker registry с автоматическим билдом и тестированием
- Инструменты для управления инфраструктурой
- Инструменты для управления жизненным циклом приложения

# Docker Store

- Дистрибутивы
- Проверенные образа
- Предоставляет образа за деньги
- В определенных случаях обращается к Docker Hub

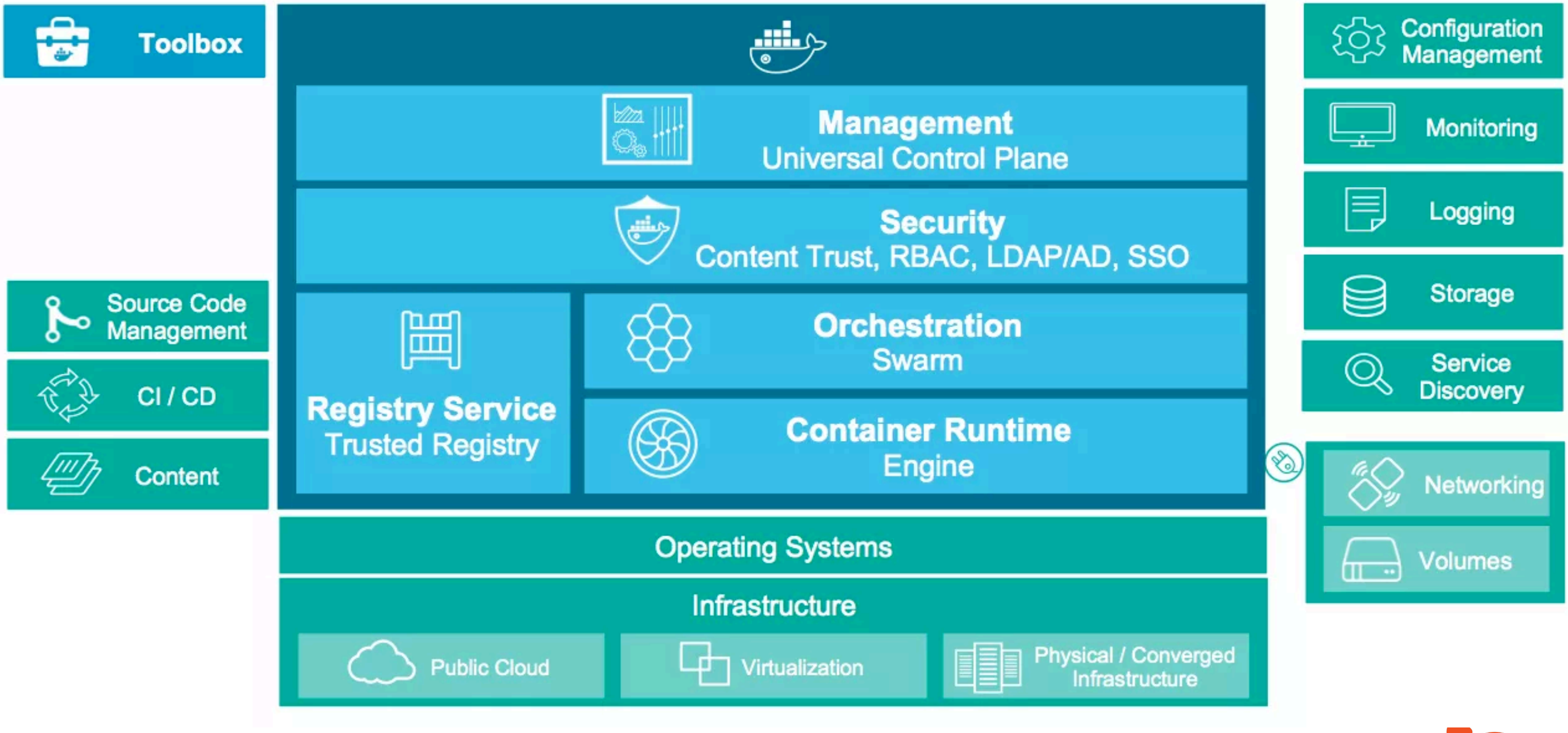
# Docker EE

- Стоит денег
- Сертифицированные дистрибутивы, образа и плагины

---

- Сканер безопасности для образов
- Продвинутое управление образами и контейнерами

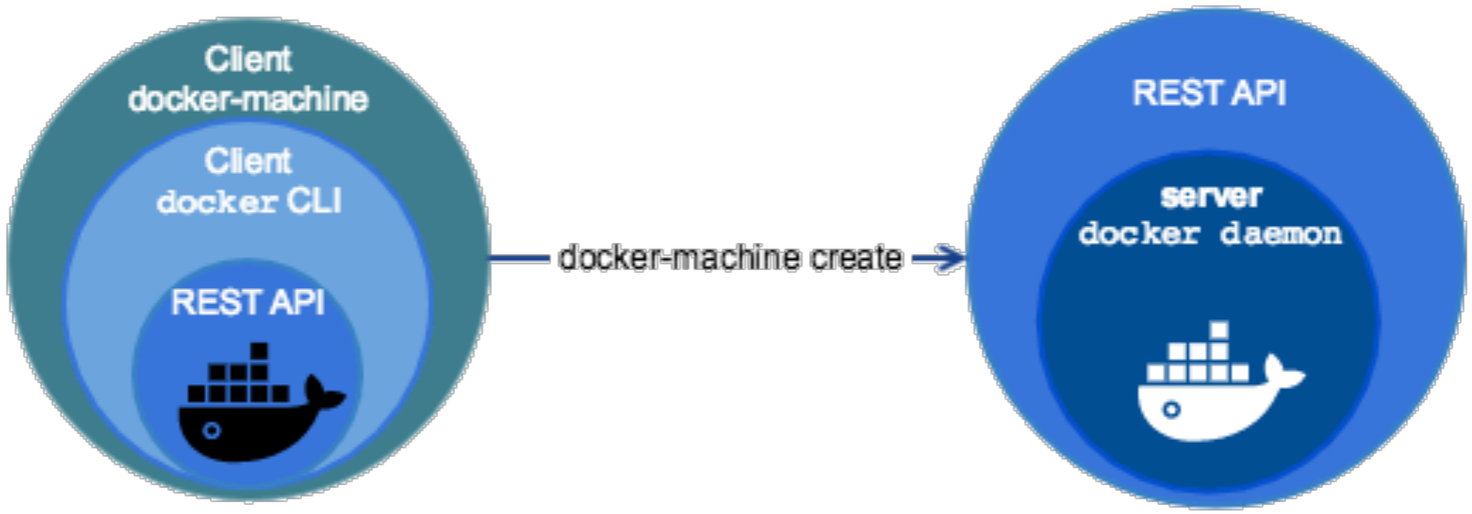
# Docker Datacenter



# Docker-machine

- Инструмент для установки docker engine на удаленные машины и управления ими
- Имеет поддержку облаков и систем виртуализации

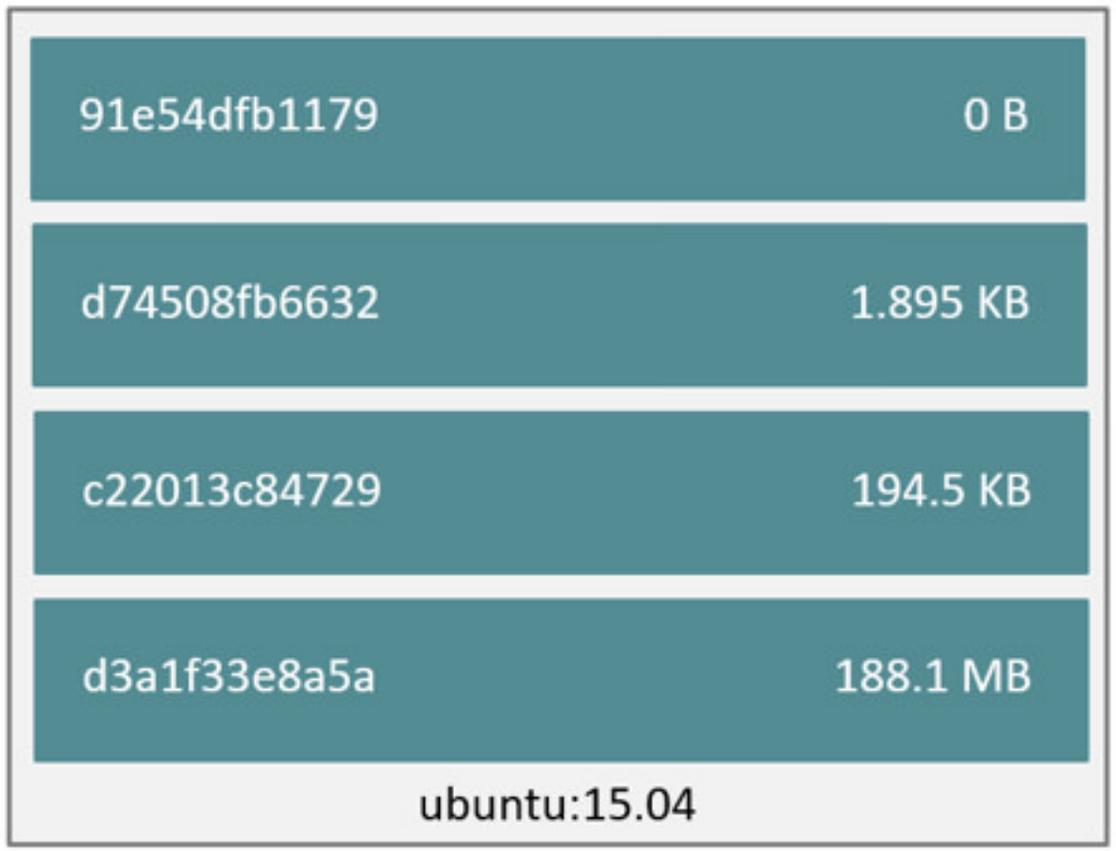
# Docker-machine




# Dockerfile

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

# Docker images




Image

CMD

RUN …

RUN …

ADD/COPY