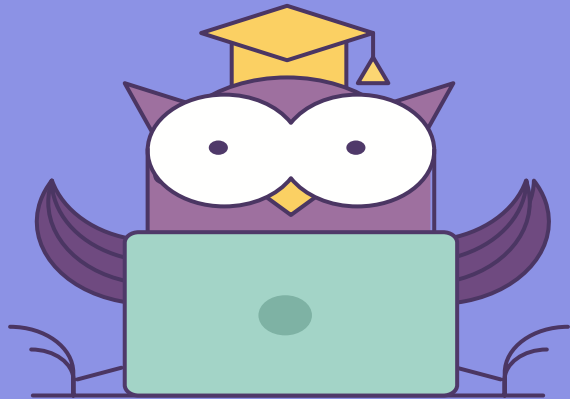





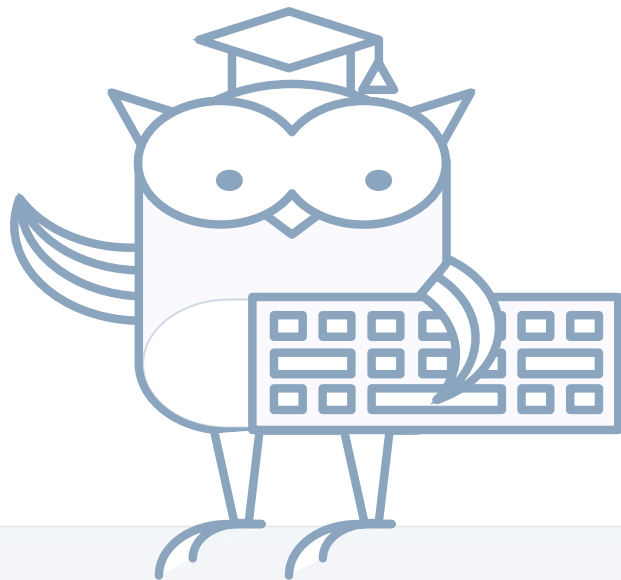
ОНЛАЙН-ОБРАЗОВАНИЕ

# Меня хорошо слышно && видно?



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

# Docker



- Network
- Volumes
- Docker-compose
- Немного разной магии

Подключаемые модули для управления сетью контейнеров

Native (встроенные в Docker)

Remote (сторонние)

По IP-адресам

Docker Links (deprecated, только для default bridge сети)

Встроенный в Docker DNS

Внешний Service Discovery/DNS сервер

## Встроенные модули

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

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

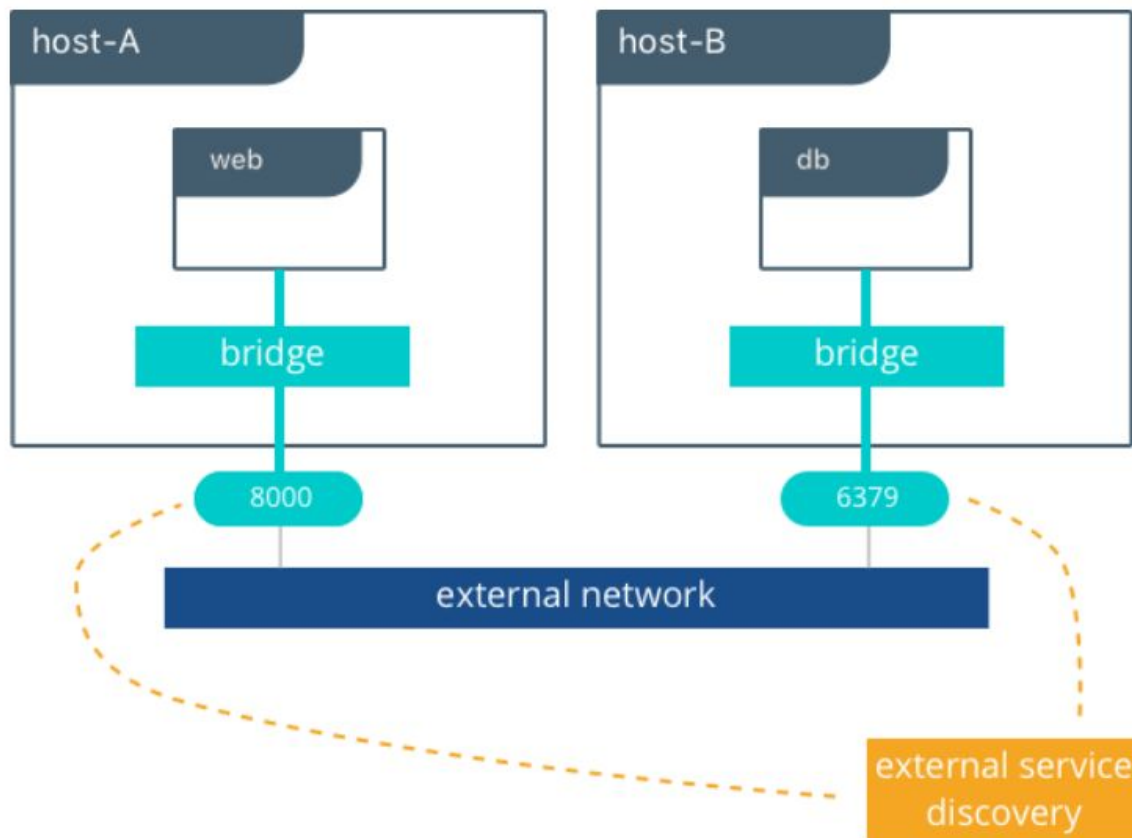
## Особенности default bridge network

Назначается по умолчанию для контейнеров

Нельзя вручную назначать IP-адреса

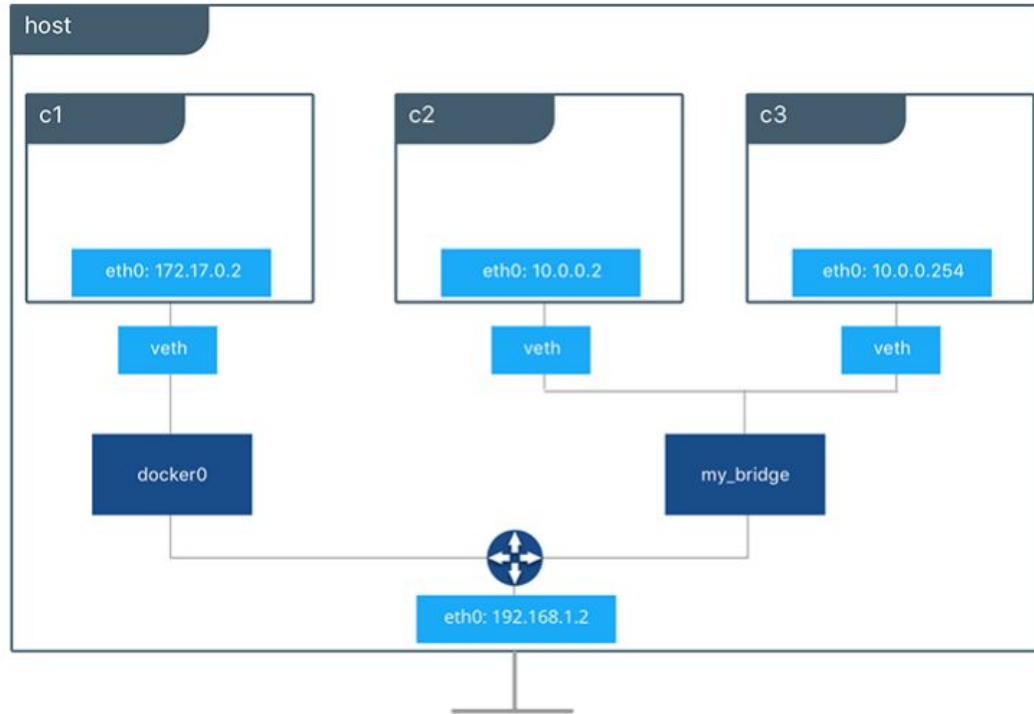
Нет Service Discovery

# Docker взаимодействие контейнеров

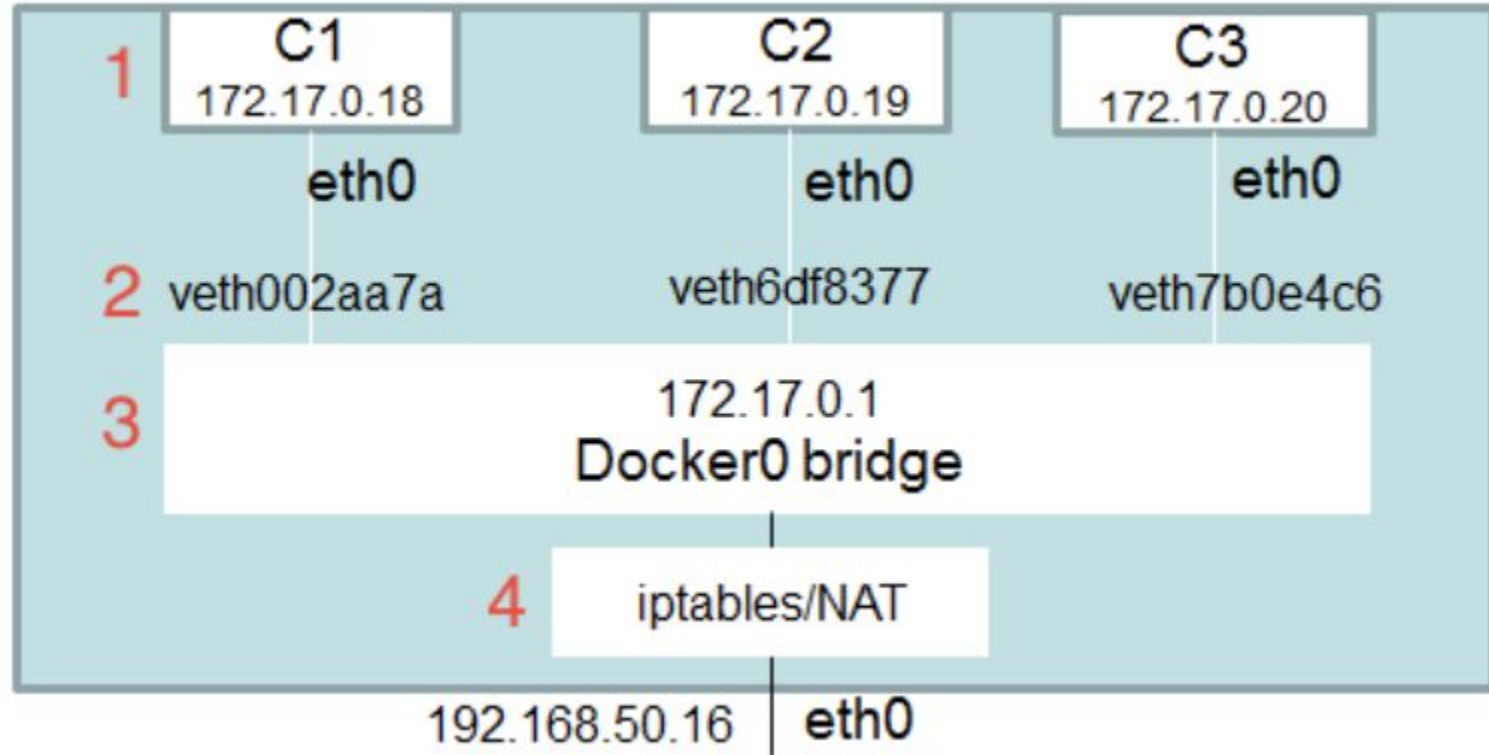


Если нужно отделить контейнер или группу контейнеров  
Контейнер может быть подключен к нескольким Bridge сетям  
(без рестарта)  
Работает Service Discovery  
Произвольные диапазоны IP-адресов

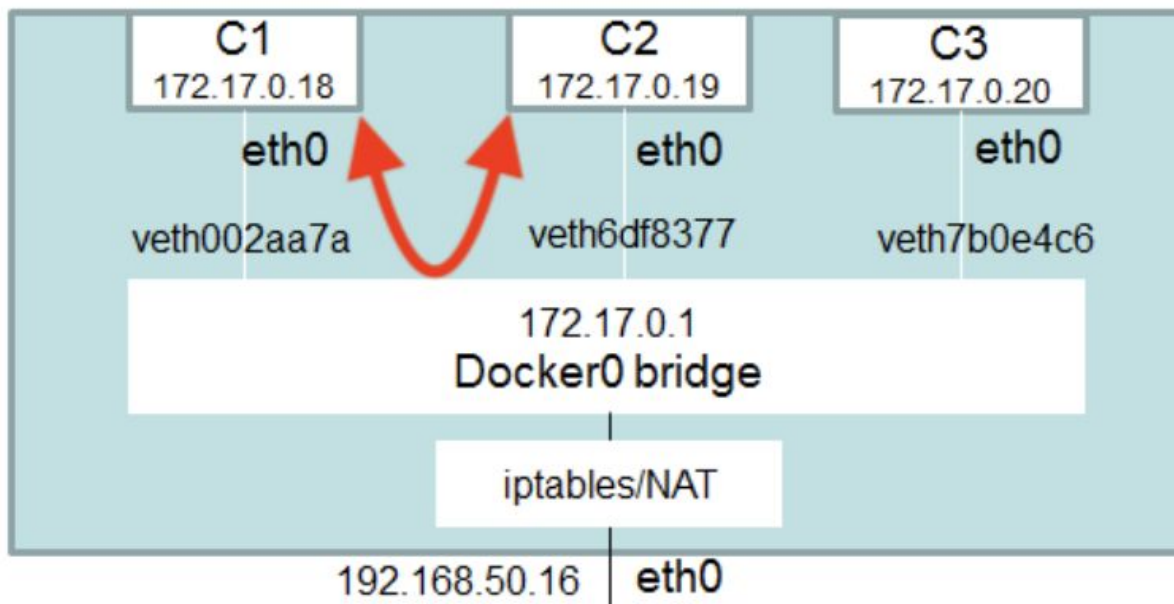
## Bridge



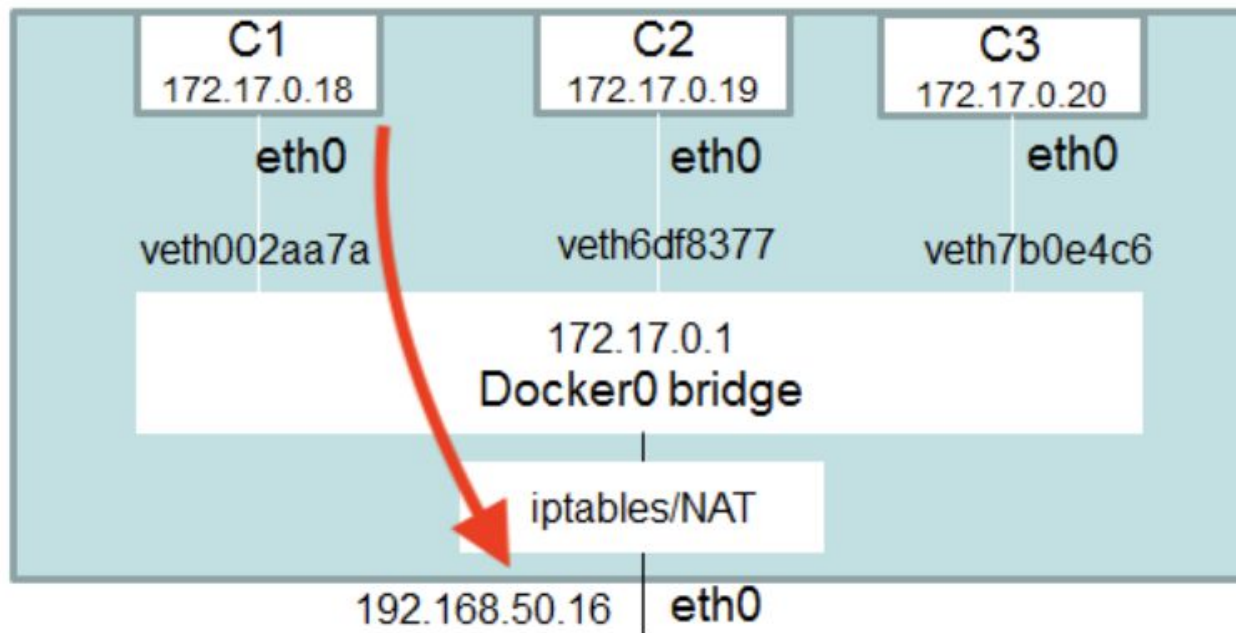
# Docker bridge



## Взаимодействие сервисов

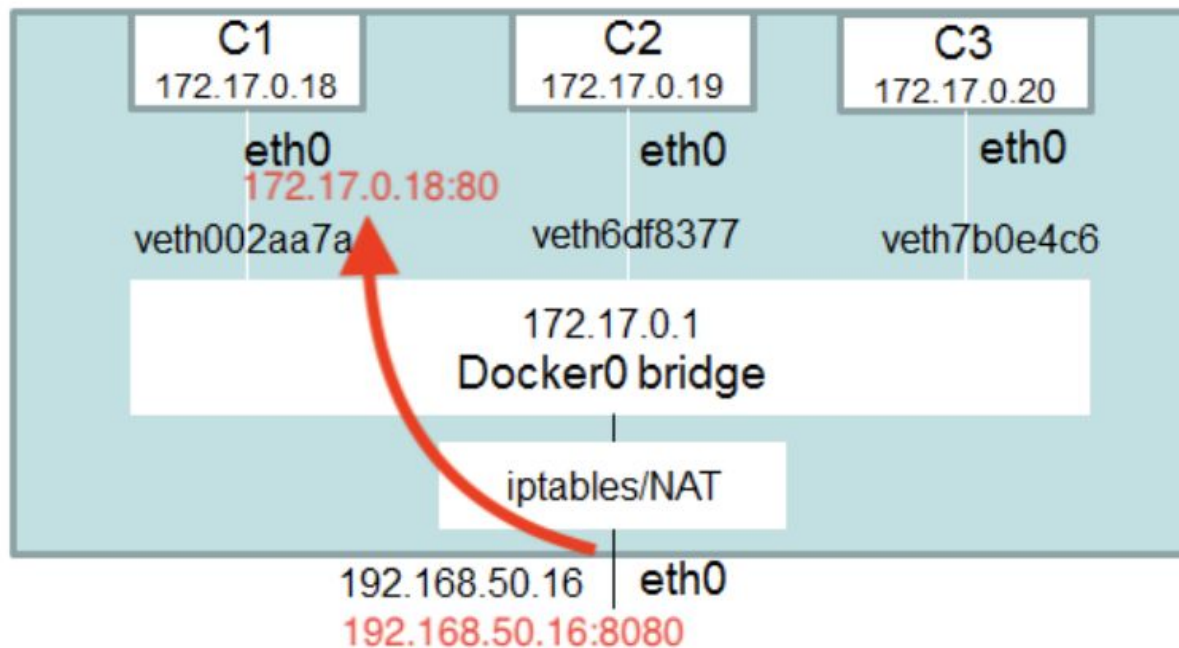


Доступ из контейнеров наружу





Доступ к контейнеру снаружи



Работает на основе sub-interfaces Linux

Более производительный, чем bridge

Если нужно подключить контейнер к локальной сети

Поддерживается тегирование VLAN (802.1Q)

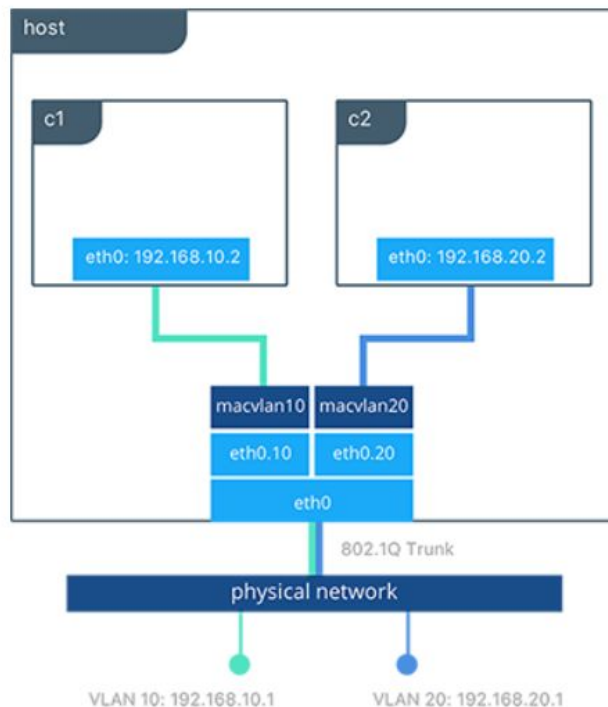
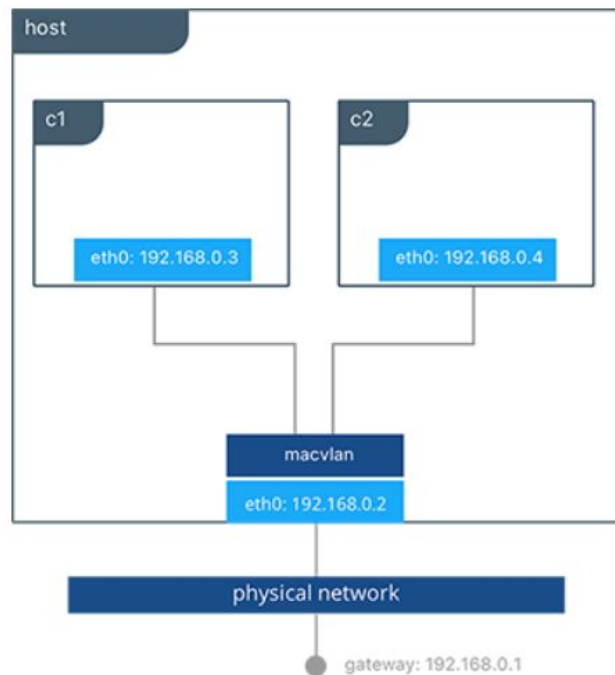
Особенности:

Легко исчерпать пул DHCP

Много MAC адресов в L2 сегменте

Сетевой интерфейс в promiscuous mode

## Macvlan

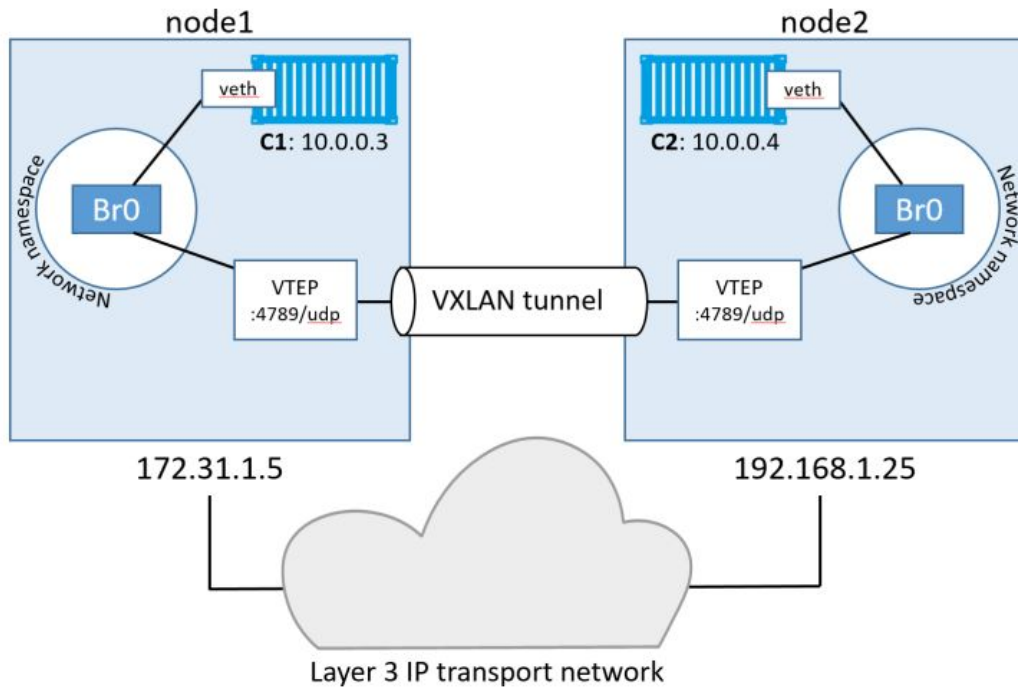


Позволяет объединить в одну сеть контейнеры нескольких Docker хостов

Работает поверх VXLAN

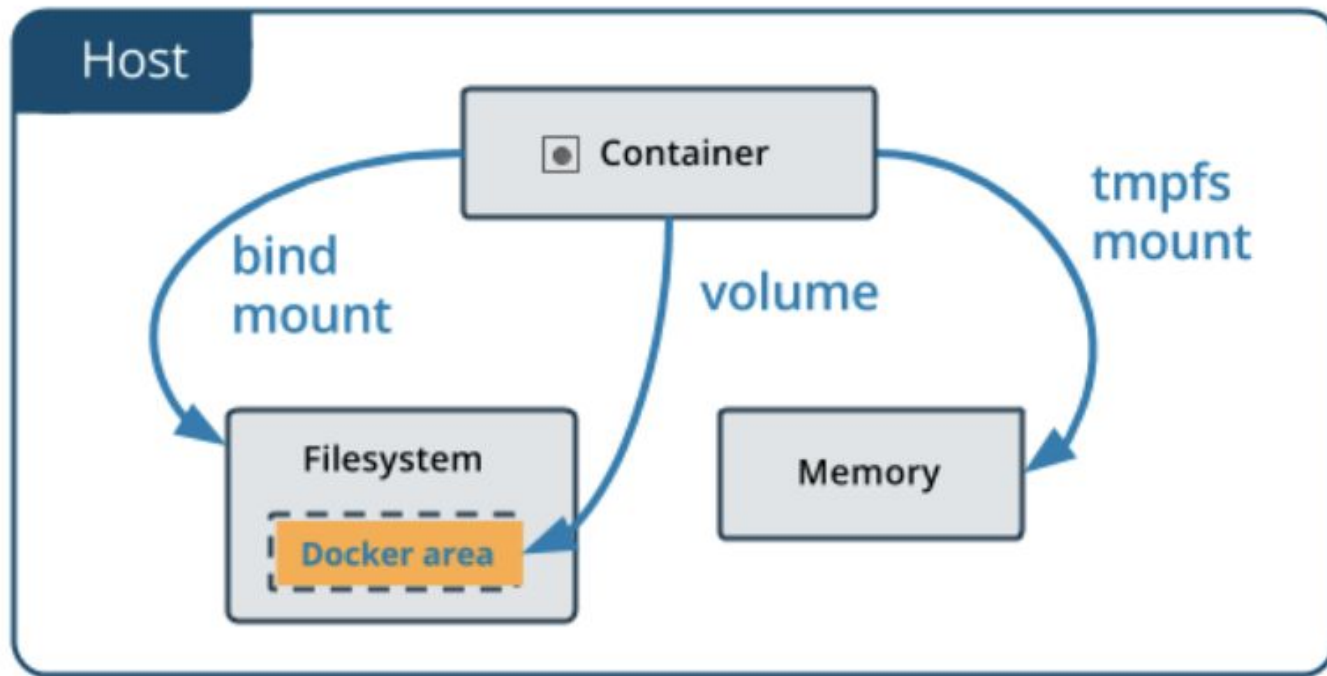
Необходимо хранить состояние распределенной сети

## Overlay



Позволяет отделить жизненный цикл данных, которые хранит том, от жизни самого контейнера, который эти данные создал.

По-простому, он позволяет избежать потери данных в случае его удаления контейнера.



**Volumes** - тома управляемые Docker'ом. Другие процессы не должны иметь к ним доступ

Именованные

Неименованные

**Bind mount** - директории на файловой системе хоста. Любой процесс может получить к ним доступ

**tmpfs** - тома расположенные в памяти хоста. Никогда не записываются на диск



## Когда использовать?

Доступ к данным из нескольких контейнеров

При миграции с хоста на хост (архивом, распределенной файловой системой, оркестратором)

Когда хотим спокойно переинициализировать контейнер и не хотим захламлять основную ФС

Пример из офф документации [тут](#)

Чет скучно, на практике посмотрим по веселее =)