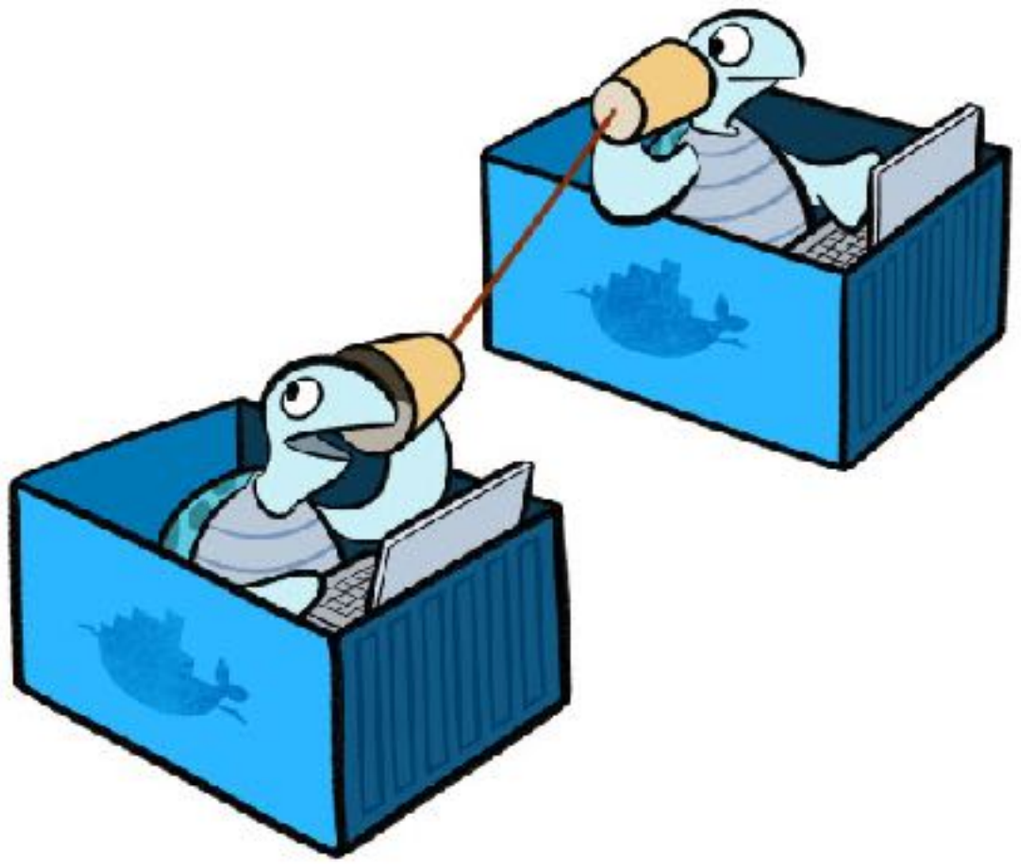
Docker: сети, docker-compose и тестирование образов

План

- Docker Networks
- Docker-compose
- Тестирование с помощью Docker

Работа с сетью в Docker

Как заставить контейнеры общаться друг с другом?



Как заставить контейнеры общаться друг с другом?

- Docker Network Drivers - подключаемые модули для управления сетью контейнеров
- **Native (встроенные в Docker)**
- Remote (сторонние)

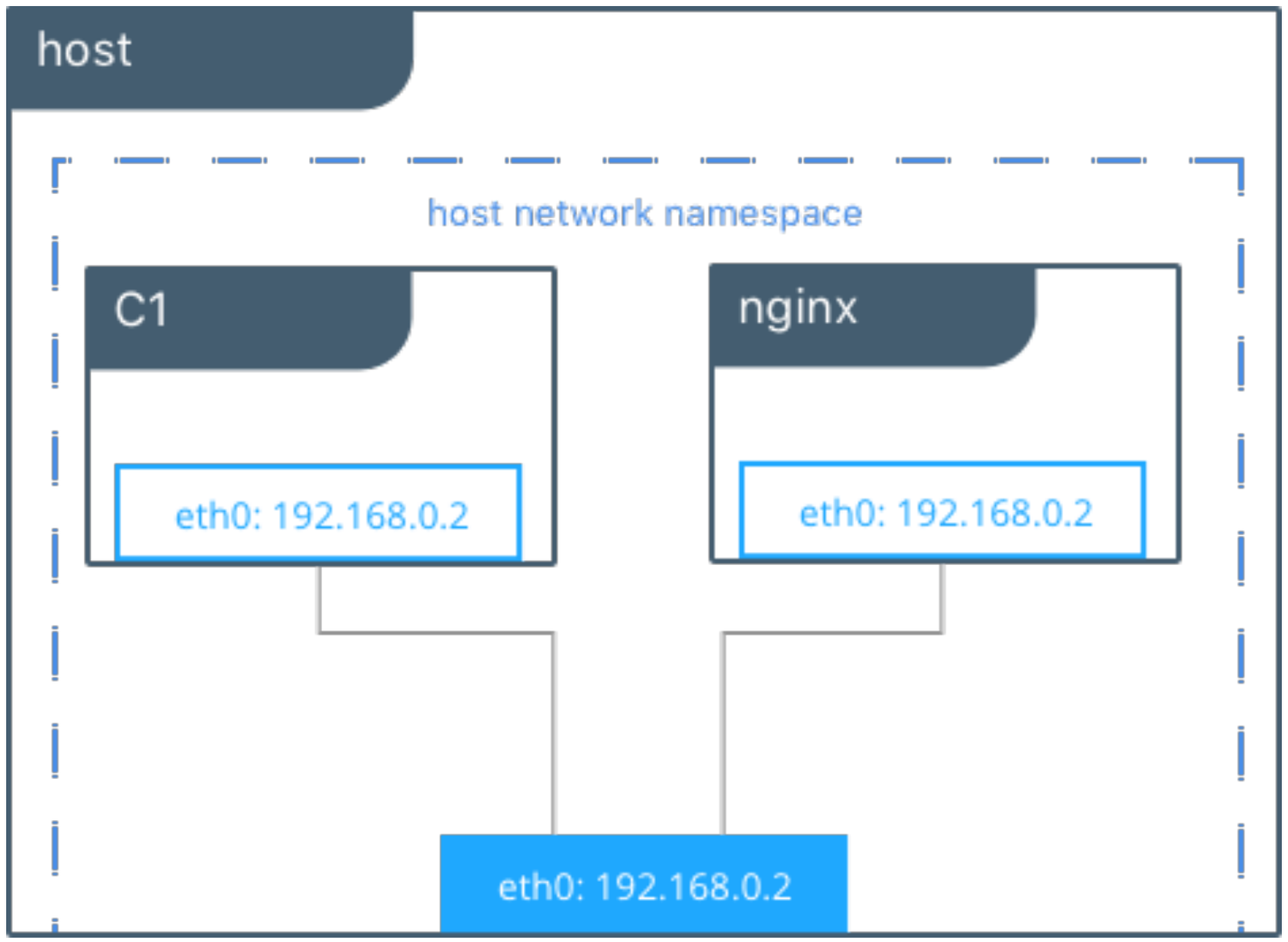
Native Docker network drivers

- None
- Host
- Bridge
- Overlay
- MACVLAN

Что в комплекте?

```
>docker network ls

NETWORK ID          NAME                DRIVER              SCOPE
6d10b6cf938f        bridge              bridge              local
a0f911148a5c        host                host                local
a89f8bfd263a        none                null                local
```

Host driver


host
host network namespace
C1
eth0: 192.168.0.2
nginx
eth0: 192.168.0.2
eth0: 192.168.0.2

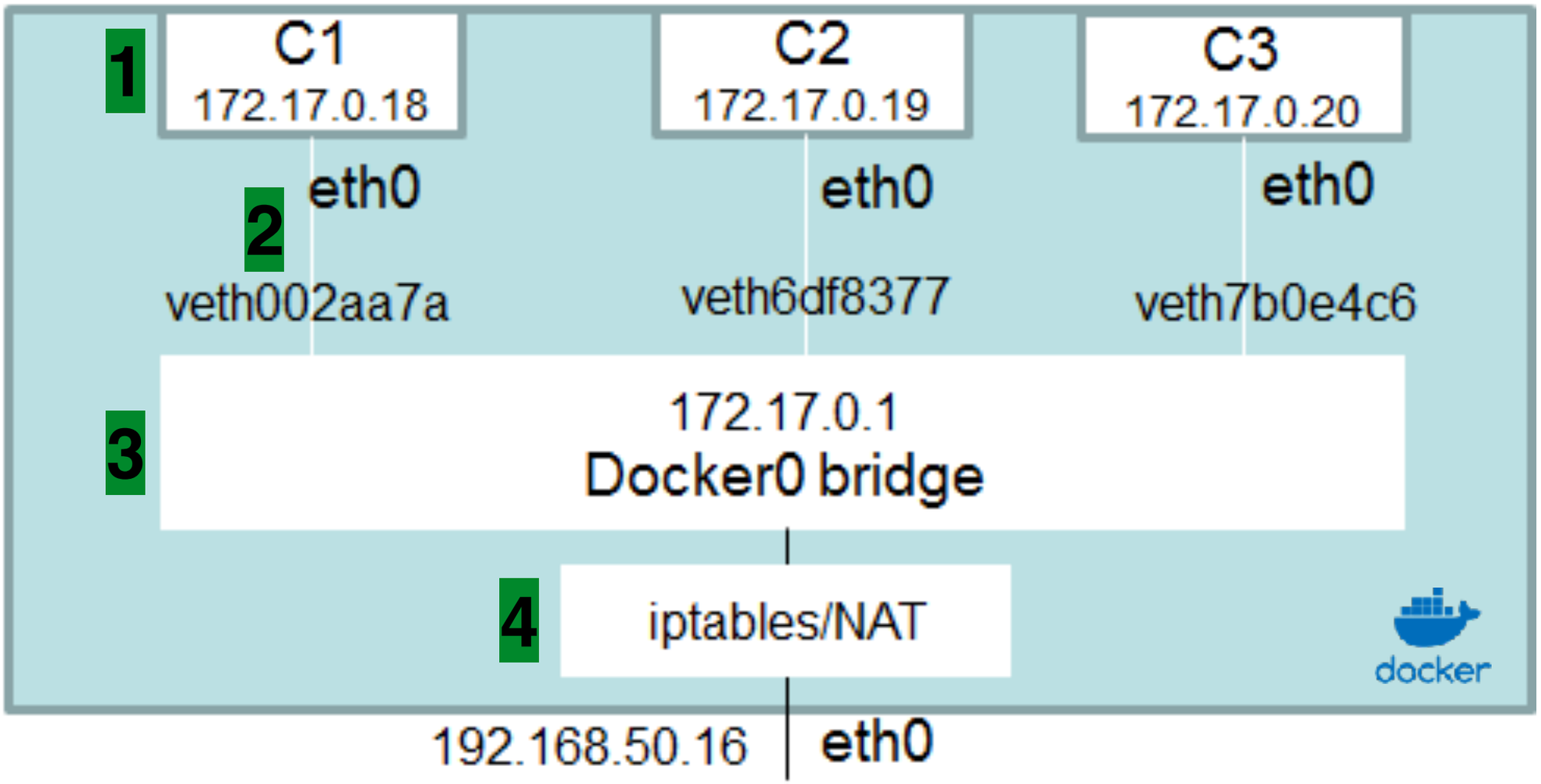
Host driver

- Контейнер использует Net-namespace хоста
- Сеть не управляется самим Docker
- Два сервиса в разных контейнерах **НЕ** могут запускаться на одном порту
- Производительность сети контейнера равна производительности сети хоста

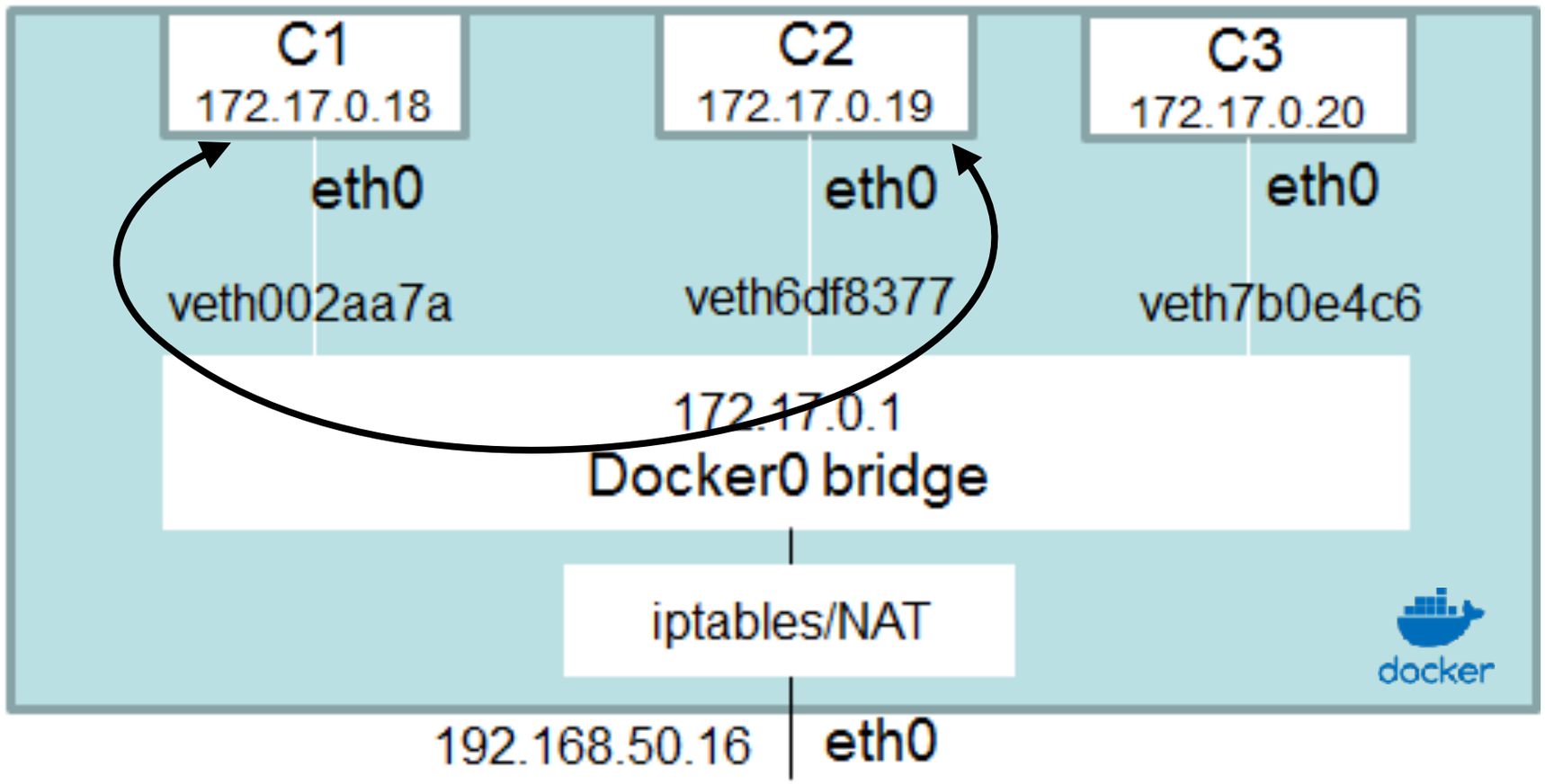
None driver

- Нет ничего, кроме loopback
- Для контейнера создается свой Net-namespace
- Сеть контейнера полностью изолирована

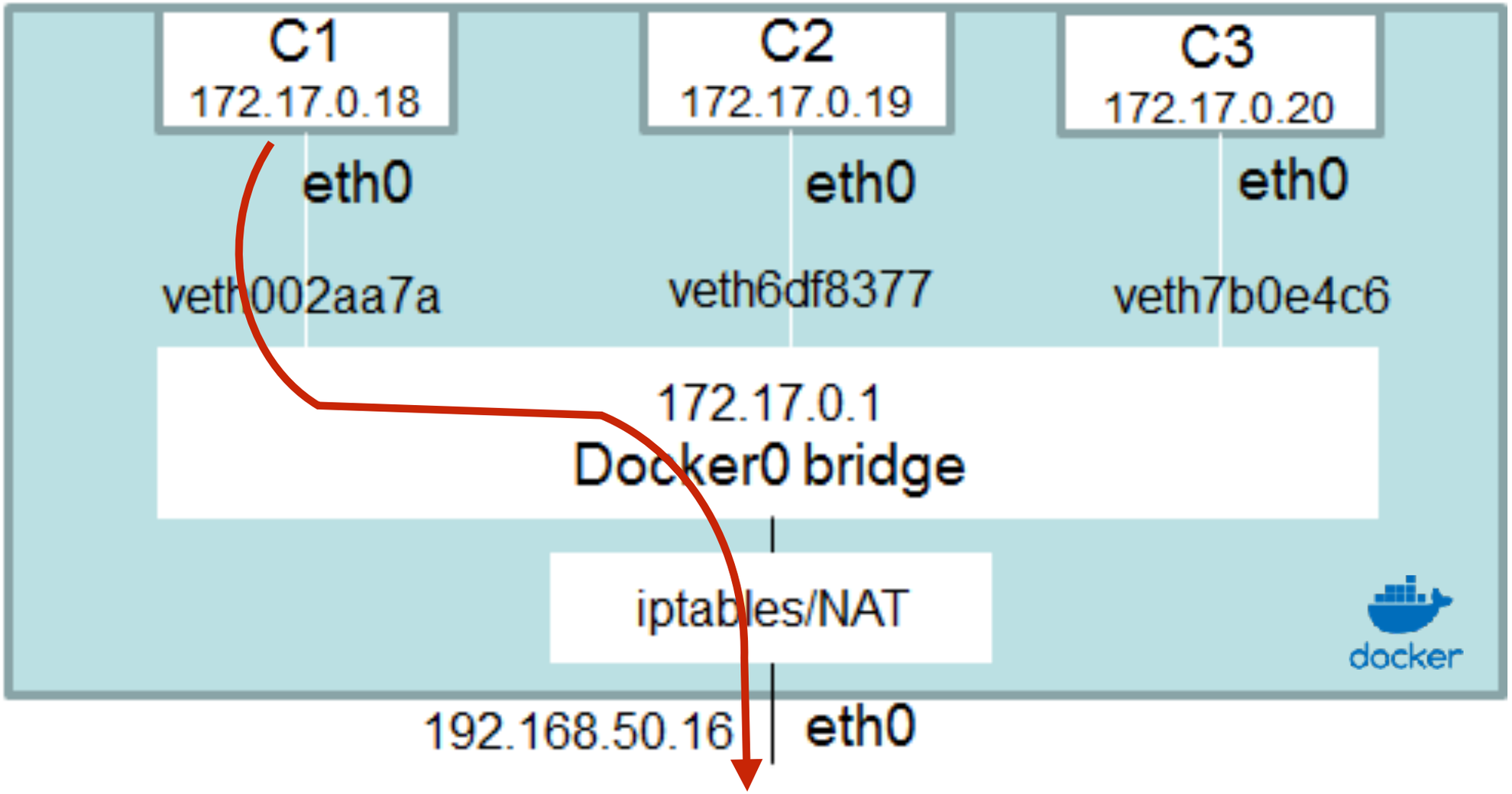
Bridge driver



Bridge driver



Bridge driver



Доступ к контейнерам извне

> docker run **-P** или **--publish-all** .

- распознает строки с ключевым словом **EXPOSE** в Dockerfile и флаг **--expose** при запуске контейнера
- привязывает доступный порт на хосте из диапазона 32768 - 61000
- Для избежания неявно открытых портов **НЕ РЕКОМЕНДУЕТСЯ** использовать

Доступ к контейнерам извне

docker run -p PORT --publish = PORT.

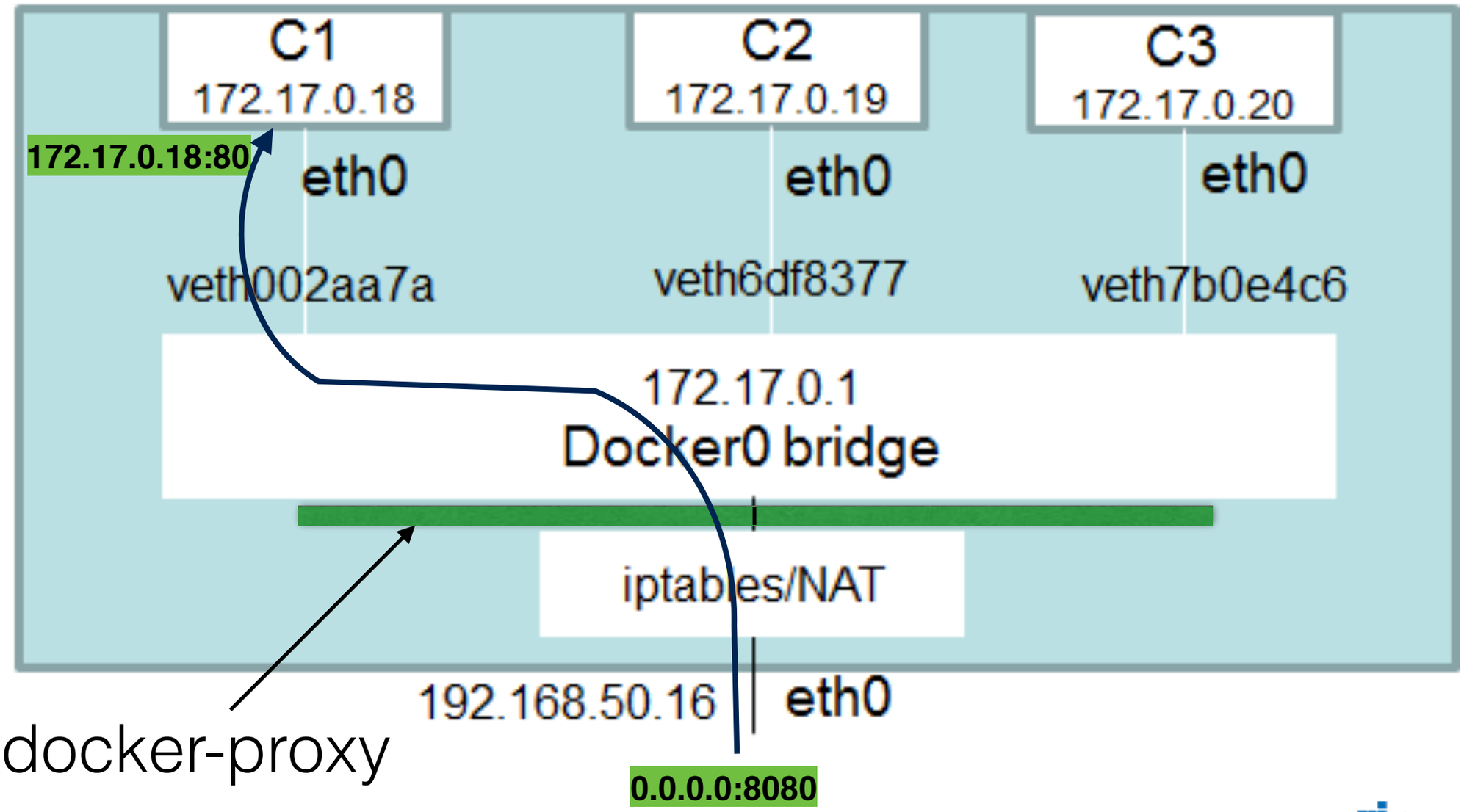
явно задает какой порт докер хоста мы хотим привязать к порту докер контейнера.

PORT может быть:

- ip_addr:hostPort:containerPort
- ip_addr::containerPort
- hostPort:containerPort
- containerPort

Обязательным для указания является порт контейнера.

```
› docker run  --name C1 -d -p 8080:80 nginx
```



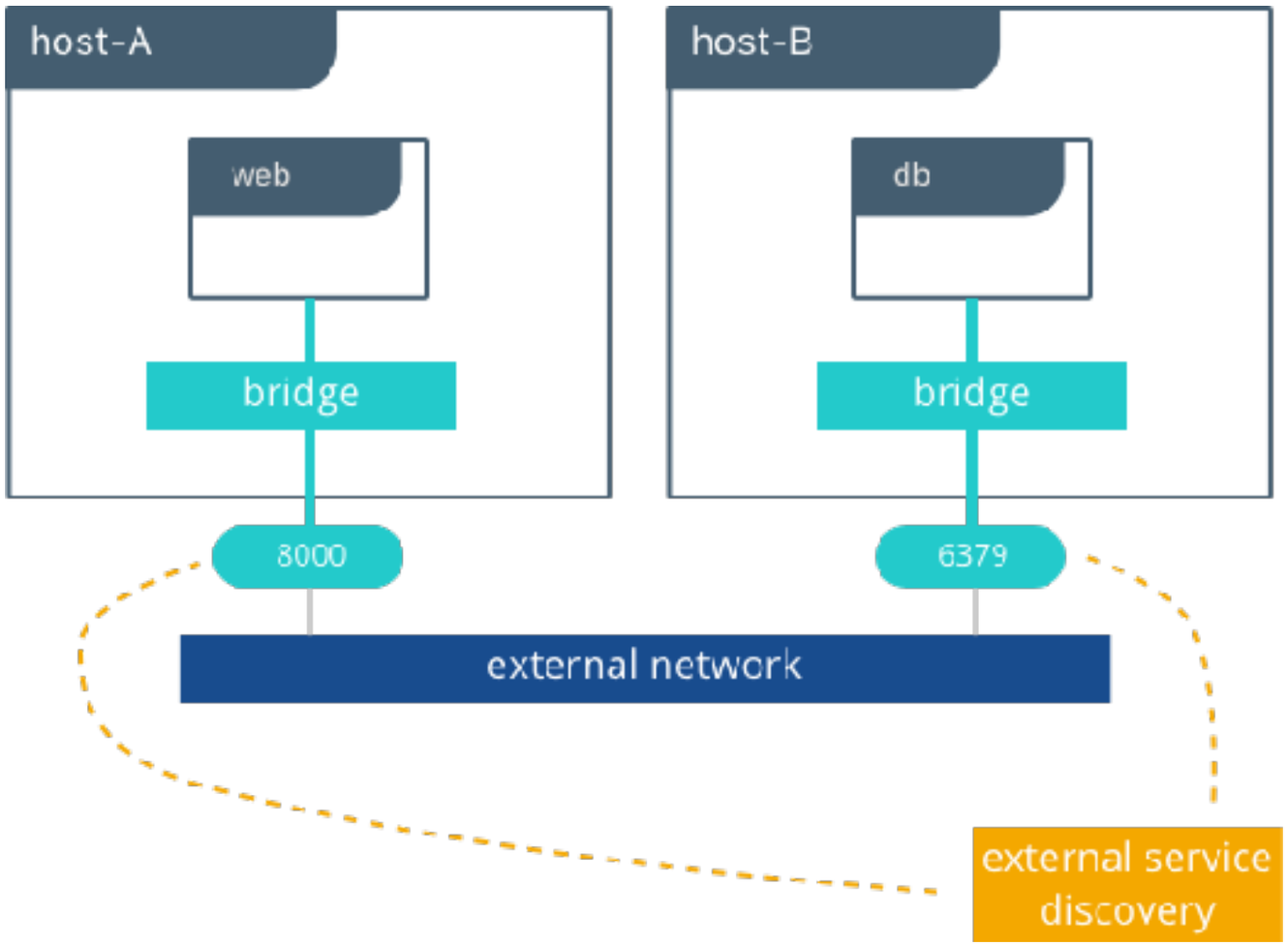
Особенности default bridge network

- Назначается по-умолчанию для контейнеров
- Нельзя вручную назначать IP-адреса
- Нет Service Discovery

Как контейнерам найти друг друга

- Внешний Service Discovery/DNS сервер
- Встроенный в Docker DNS
  (не работает для default bridge сети)
- docker links (**deprecated**, только для default bridge сети)

Внешний Service Discovery/DNS сервер



Встроенный в Docker Service-Discovery

- Не работает для default bridge-сети
- По адресу **127.0.0.11:53** для каждого контейнера

```
> docker run -d --network=reddit  -p 9292:9292 --name=ui_reddit chromko/ui:1.0
> docker run --network reddit tutum/dnsutils nslookup ui_red 127.0.0.11

   Server:      127.0.0.11
   Address: 127.0.0.11#53

   Non-authoritative answer:
   Name:    ui_red
   Address: 172.18.0.4
```

Docker links

- Работает **только** для default bridge-сети
- Deprecated
- Вносит запись в /etc/hosts файл запускаемого контейнера
- Заполняет переменные окружения
- Задается только при старте контейнера и не меняется runtime

User Defined Networks

- Bridge
- MacVlan
- Overlay

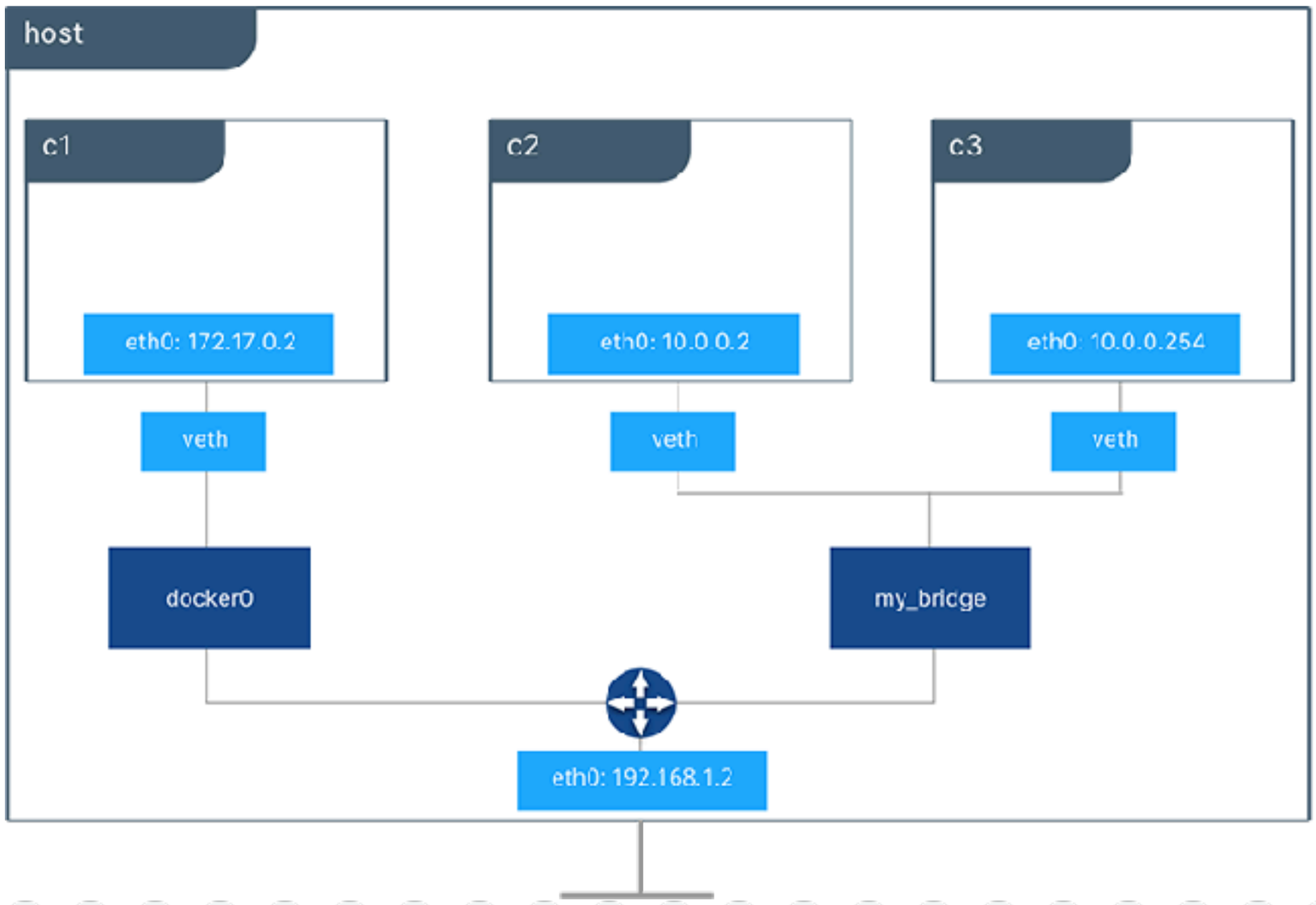
User Defined Bridge

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

User Defined Bridge

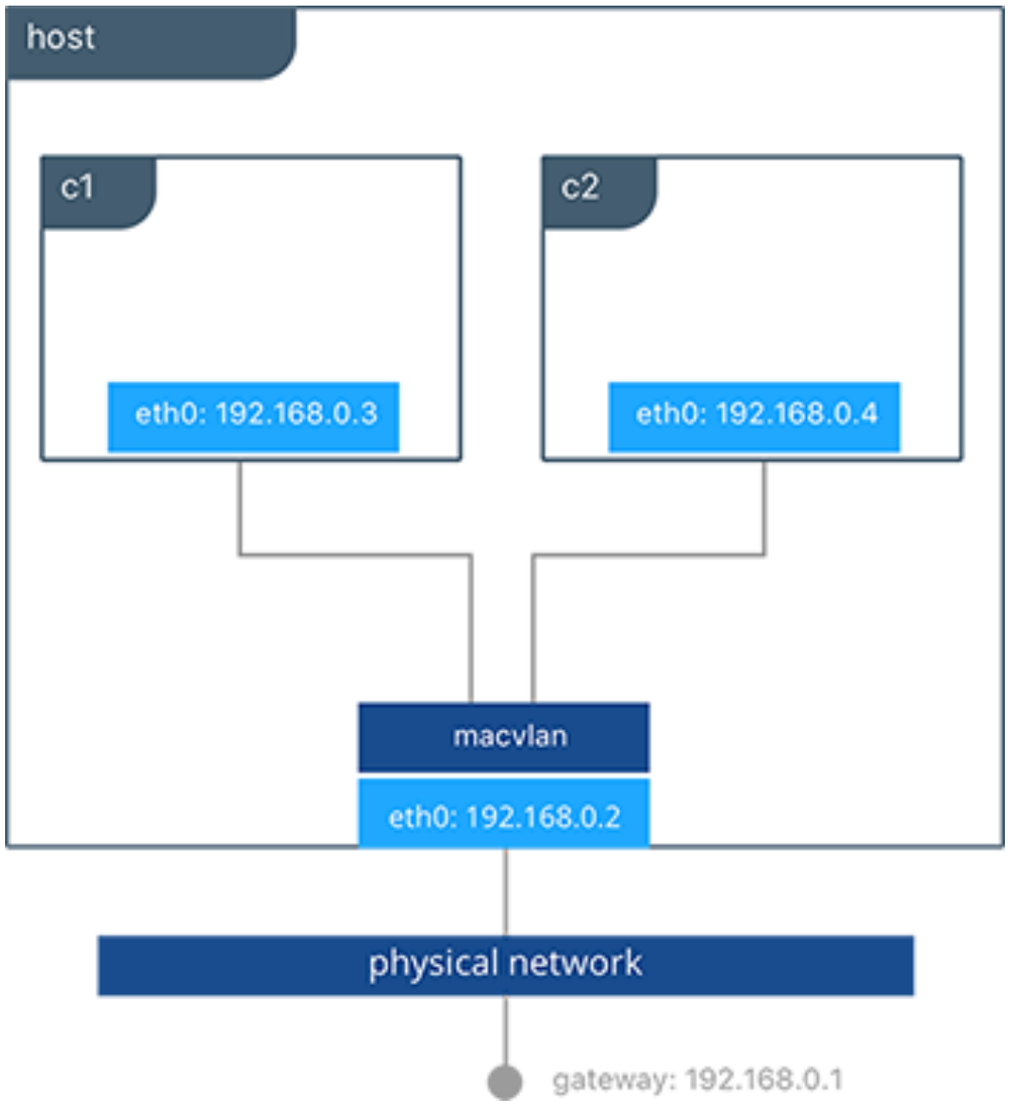
```
> docker network create -d bridge --subnet 10.0.0.0/24 my_bridge

> docker run --name c1 ubuntu
> docker run --network my_bridge --name c2 ubuntu
> docker run --network my_bridge --name c3 ubuntu
```

User Defined Bridge



Macvlan driver

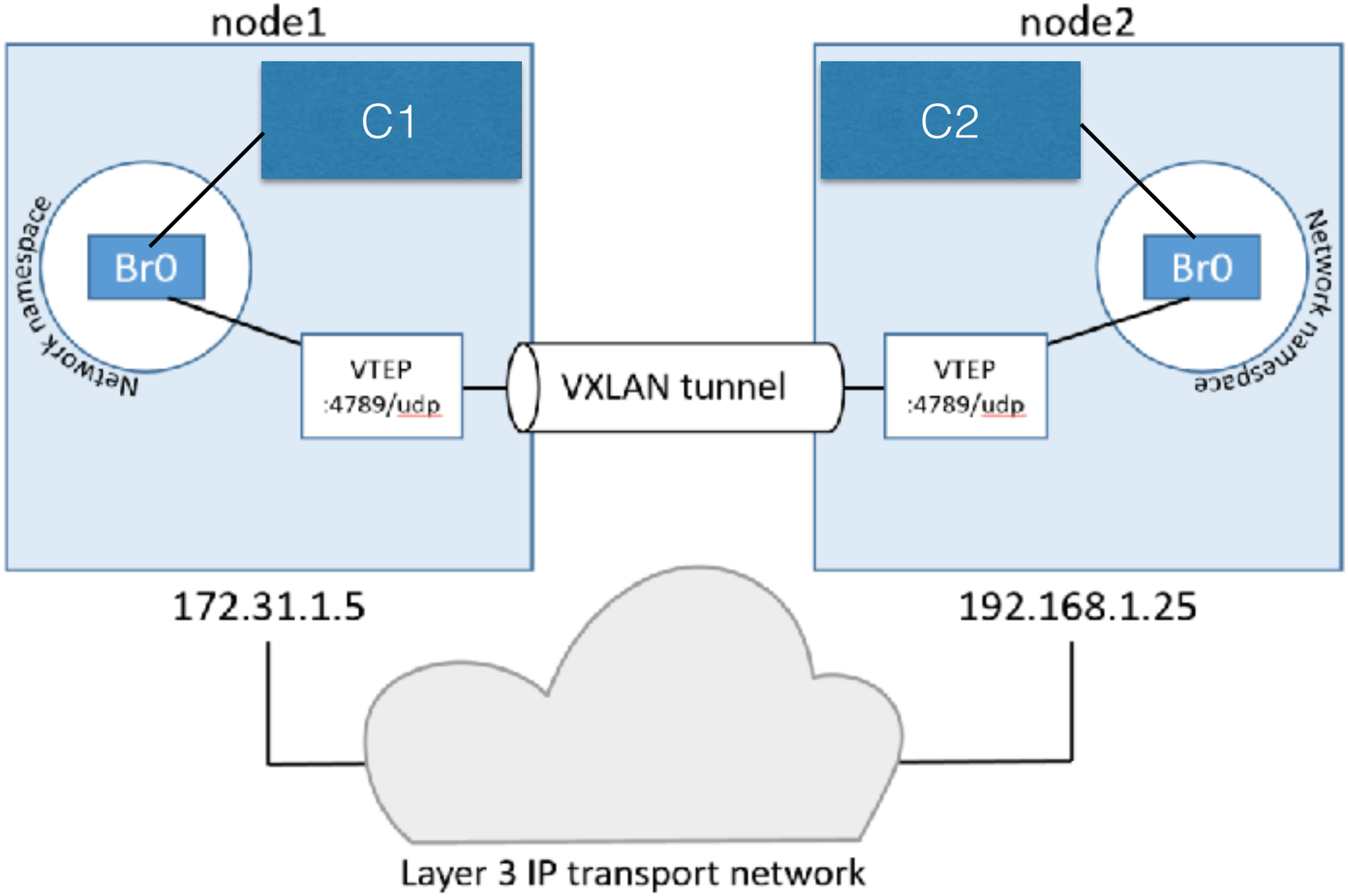
- Работает на основе sub-interfaces Linux
- Более производительный, чем Linux-bridge
- Если нужно подключить контейнер к локальной сети хоста
- Поддерживается тегирование VLAN (802.1Q)



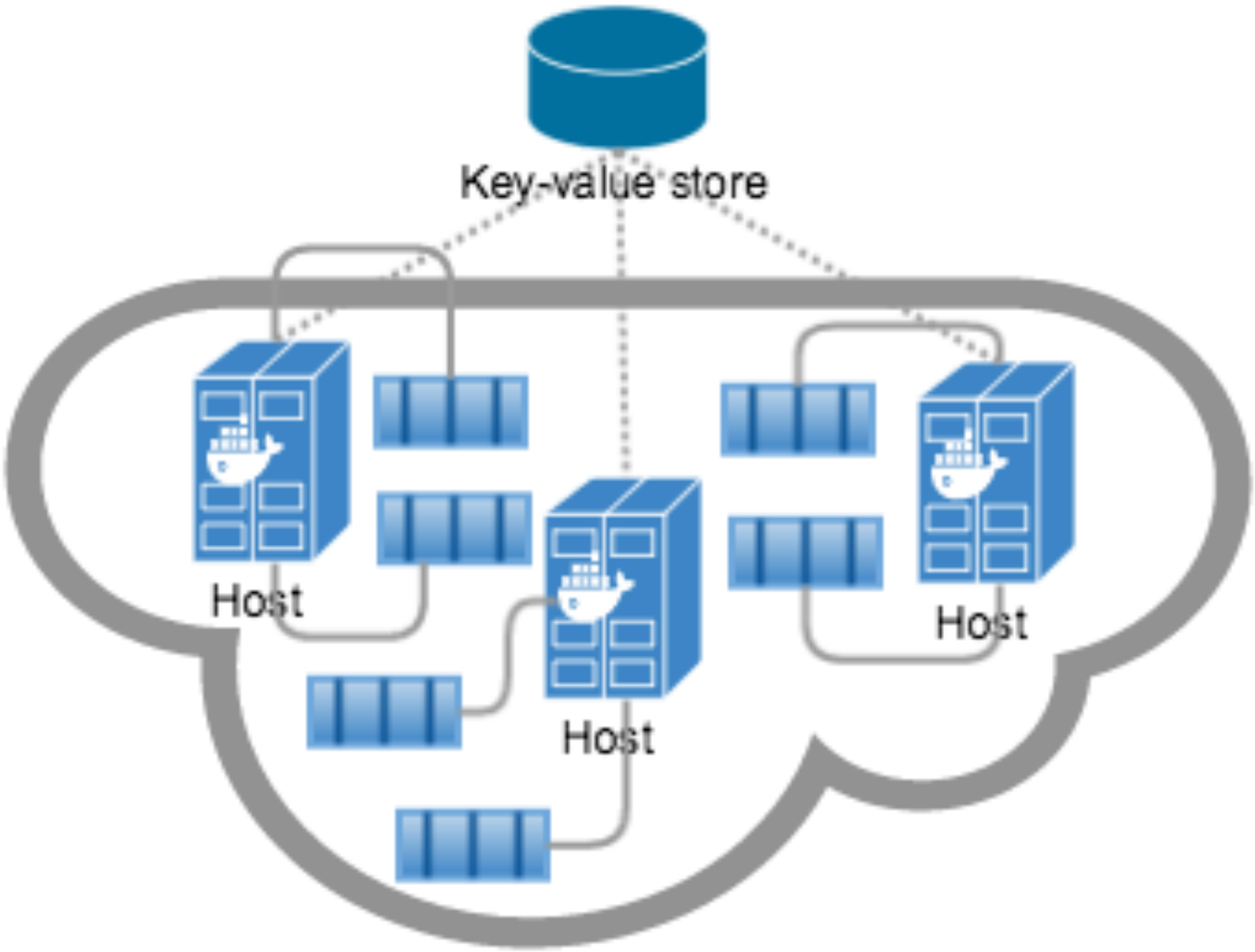
Overlay network

- Overlay network – позволяет объединить в одну сеть контейнеры нескольких докер хостов.
- работает поверх vxlan
- Нужно хранить состояние распределенной сети

Overlay network



Overlay network



Docker-compose

Проблемы

- Одно приложение состоит из множества контейнеров/сервисов
- Один контейнер зависит от другого
- Порядок запуска имеет значение
- docker build/run/create … (долго и много)

Docker-compose

- Отдельная утилита
- Декларативное описание docker-инфраструктуры в YAML-формате
- Управление многоконтейнерными приложениями

docker-compose.yml

```
version: ‘2'
services:
  post_db:
    image: mongo:3.2
    volumes:
      - post_db:/data/db
    networks:
      - reddit
  ui:
    build: ./ui
    image: ${USERNAME}/ui:1.0
    ports:
      - 9292:9292/tcp
    networks:
      - reddit
  post:
    build: ./post-py
    image: ${USERNAME}/post:1.0
    networks:
      - reddit
…
```

compose-file

Основные Секции:

```
version: '2'
services:
  …
volumes:
  …
networks:
  …
```

(required)

version - определяет поддерживаемый функционал

Services

Описываем конфигурацию запуска контейнера/группы контейнеров

```
services:
  post_db:
    image: mongo:3.2
    volumes:
      - post_db:/data/db
    networks:
      - reddit
  ui:
    build: ./ui
    image: ${USERNAME}/ui:1.0
    ports:
      - 9292:9292/tcp
    networks:
      - reddit
  post:
    build: ./post-py
    image: ${USERNAME}/post:1.0
    networks:
      - reddit
...
```

Volumes

```
services:
  post_db:
    volumes:
      - type: volume
        source: sample_vol
        target: /data/sample
      - post_db:/data/db

volumes:
  post_db:
  sample_vol:
```

} Long notation

} Short notation

Networks

```
version: '2'
services:
  mongo_db:
    image: mongo:3.2
    volumes:
      - post_db:/data/db
    networks:
      reddit:
        aliases:
          - post_db
          - comment_db

networks:
  reddit:
```

```
networks:
  reddit:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.16.238.0/24
```

docker-compose

> docker-compose up -d

```
Starting hw16_comment_1 ...
Starting hw16_comment_1
Starting hw16_post_1 ...
Starting hw16_ui_1 ...

Creating hw16_mongo_db_1 ...
Starting hw16_post_1
Starting hw16_ui_1
Starting hw16_ui_1 ... done
```

> docker-compose ps

Name	Command	State	Ports
hw16_comment_1	puma	Up	
hw16_mongo_db_1	docker-entrypoint.sh mongod	Up	27017/tcp
hw16_post_1	python3 post_app.py	Up	
hw16_ui_1	puma	Up	0.0.0.0:9292->9292/tcp

Тестирование с Docker

Что тестировать?

- Содержимое образа
- Интеграционное тестирование
- Healthchecks

Тестируем образ

А зачем?

- Dockerfile всё еще пишут люди
- Декларативность Dockerfile неоднозначна
- Кто-то проверяет, что внутри docker image?

Тестируем образ

- goss - утилита для тестирования инфраструктуры
- dgoss - обертка на bash, запускающая контейнер из данного образа и выполняющая тесты

Тестируем образ

- goss.yml

```
command:
  mongod --version | head -n1:
    exit-status: 0
    stdout:
      - db version v3.2.17
process:
  mongod:
    running: true
addr:
  tcp://localhost:27017:
    reachable: true
    timeout: 500
```

Тестируем образ

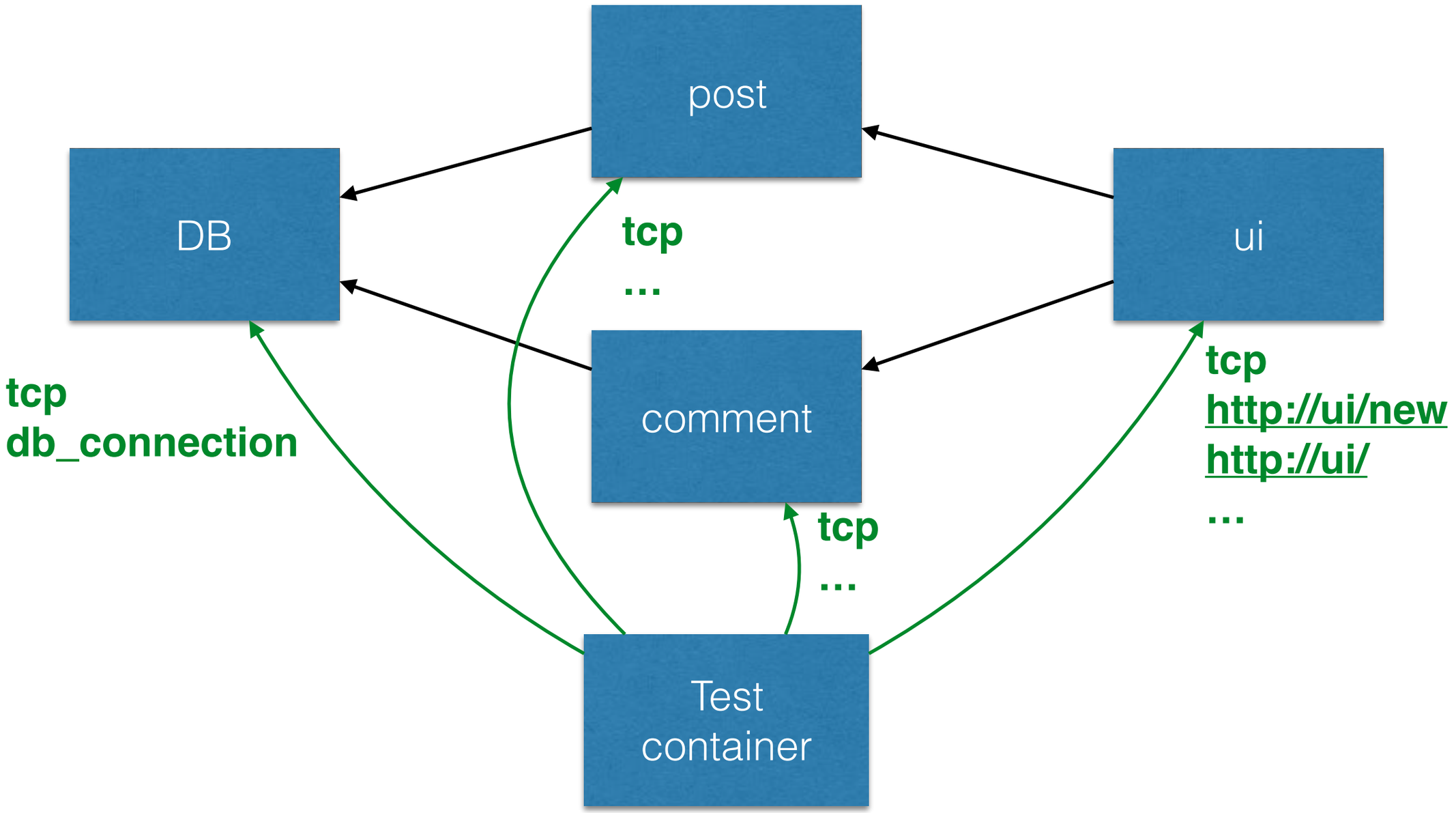
```
> dgoss run mongo:3.2

 INFO: Starting docker container
 INFO: Container ID: c23f92f0
 INFO: Sleeping for 0.2
 INFO: Running Tests
 Process: mongod: running: matches expectation: [true]
 Addr: tcp://localhost:27017: reachable: matches expectation: [true]
 Command: mongod --version | head -n1: exit-status: matches expectation: [0]
 Command: mongod --version | head -n1: stdout: matches expectation: [db version v3.2.17]

 Total Duration: 0.015s
 Count: 4, Failed: 0, Skipped: 0
 INFO: Deleting container
```

Интеграционные тесты

An **integration test** verifies the communication paths and interactions between components to detect interface defects

(Martin Fowler)

Интеграционные тесты



Healthchecks

Назначение: обеспечить контроль над работоспособностью сервиса средствами Docker

1) Оператор Dockerfile

```
HEALTHCHECK CMD curl --fail http://localhost:5000/healthcheck || exit 1
```

2) Инструкция docker-compose

```
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost"]
  interval: 1m30s
  timeout: 10s
  retries: 3
```

Healthchecks

> docker ps

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                    PORTS                    NAMES
9d392e622de0        chromko/ui:1.0      "puma"              42 seconds ago      Up 41 seconds (healthy)   0.0.0.0:9292->9292/tcp   hw16_ui_1
```