Сетевое взаимодействие Docker контейнеров. Docker Compose. Тестирование образов

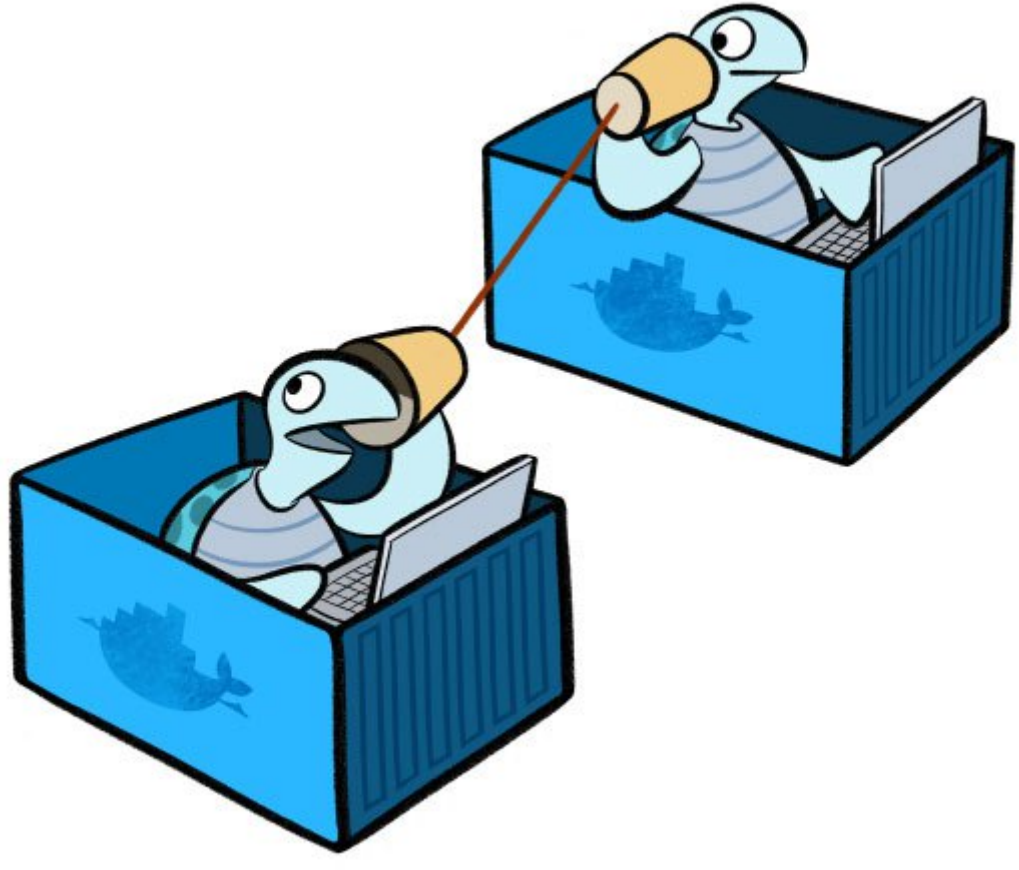
Не забудь включить запись!

План

- Работа с сетью в Docker
- Docker Compose
- Тестирование Docker образов

Работа с сетью в Docker

Как заставить контейнеры общаться друг с другом



Docker Network Drivers

Подключаемые модули для управления сетью контейнеров

- **Native** (встроенные в Docker)
- **Remote** (сторонние)

Docker Network Drivers

Встроенные модули

- None
- Host
- Bridge
- Macvlan
- Overlay

Docker Network Drivers

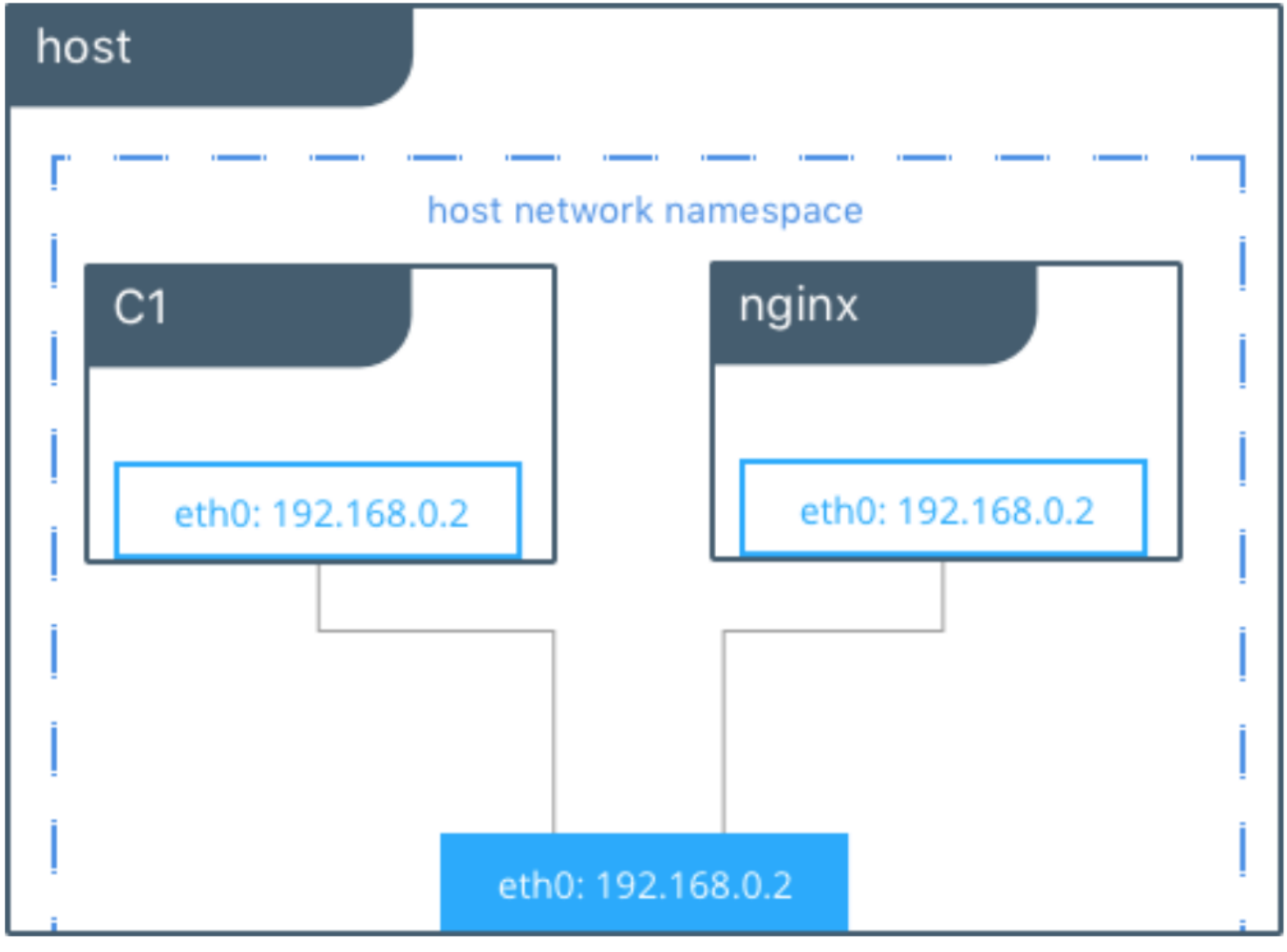
```
$ docker network ls

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

Null Driver

- Для контейнера создается свой network namespace
- У контейнера есть только loopback интерфейс
- Сеть контейнера полностью изолирована

Host Driver

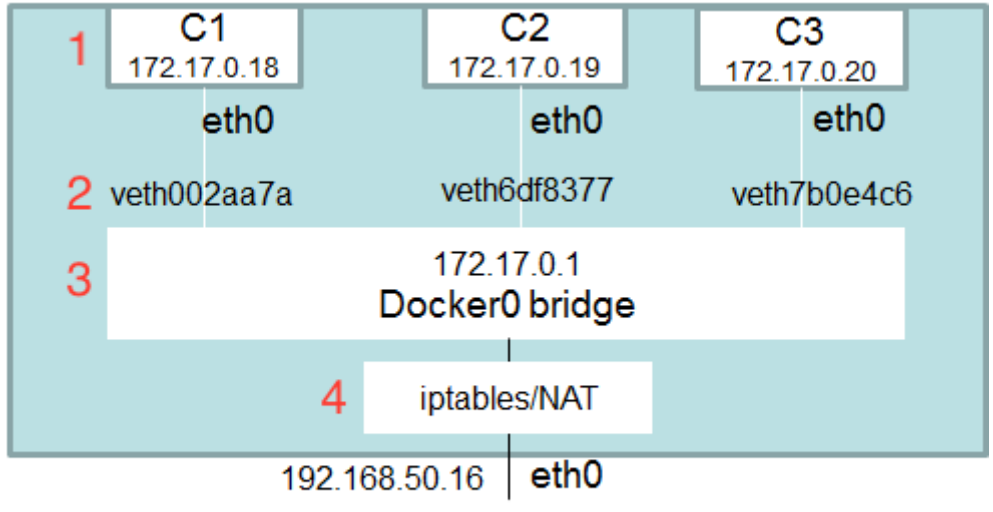

host
host network namespace
C1
eth0: 192.168.0.2
nginx
eth0: 192.168.0.2
eth0: 192.168.0.2

Host Driver

- Контейнер использует 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

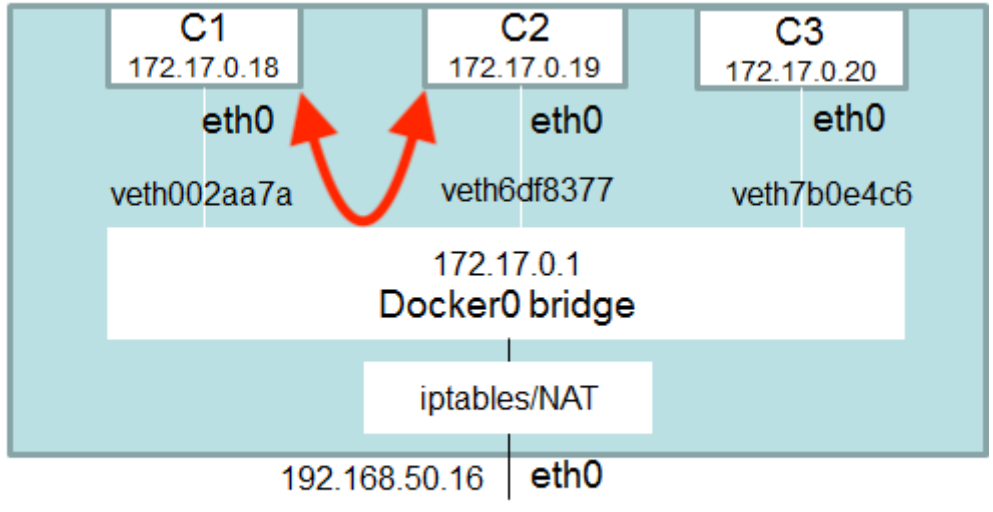
Bridge Driver

Компоненты



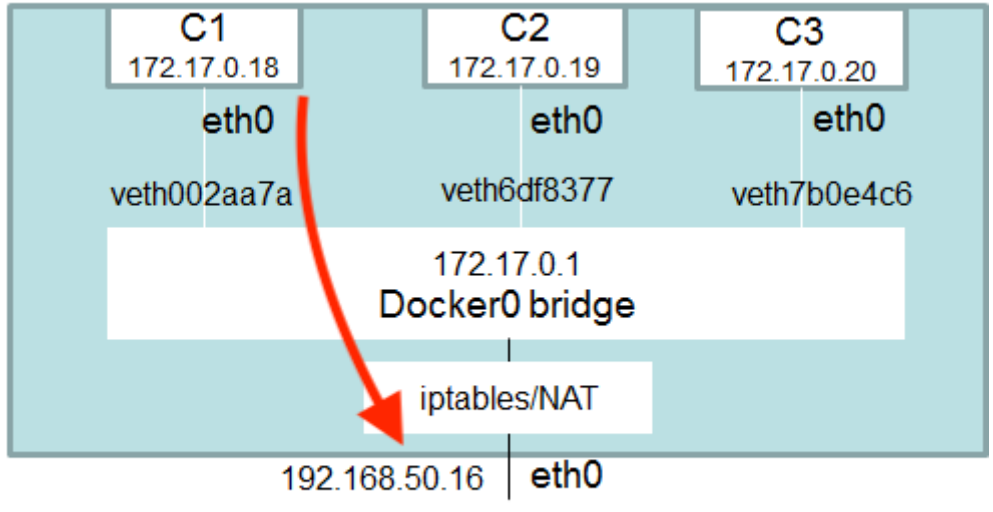
Bridge Driver

Взаимодействие между контейнерами



Bridge Driver

Доступ из контейнеров наружу



Bridge Driver

Доступ к сервисам извне

```
$ docker run -P (--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

Bridge Driver

Доступ к сервисам извне

```
$ docker run -p PORT (--publish=PORT)
```

PORT может быть в формате:

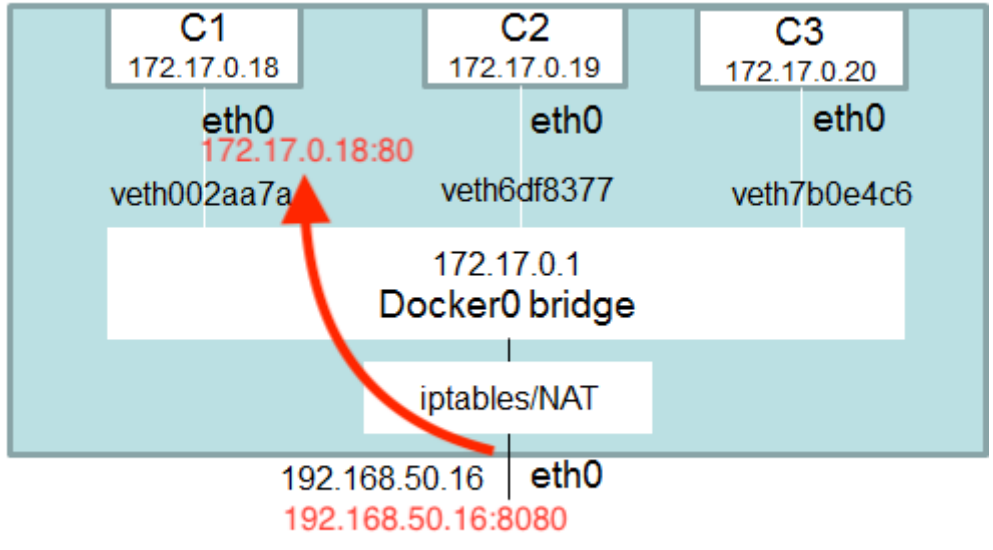
- ip_addr:hostPort:containerPort/proto
- ip_addr:hostPort:containerPort
- hostPort:containerPort
- containerPort (будет работать как -P)

Обязательным для указания является порт контейнера

Bridge Driver

Доступ к сервисам извне

```
$ docker run –p 192.168.50.16:8080:80 nginx
```



Bridge Driver

Особенности default bridge network

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

Как контейнерам найти друг друга

- По IP-адресам
- Docker Links (**deprecated**, только для default bridge сети)
- Встроенный в Docker DNS
- Внешний Service Discovery/DNS сервер

Docker Links

Deprecated

- Вносит запись в /etc/hosts файл запускаемого контейнера
- Пробрасывает переменные окружения
- Задается только при старте контейнера и после этого не меняется

Встроенный в Docker DNS

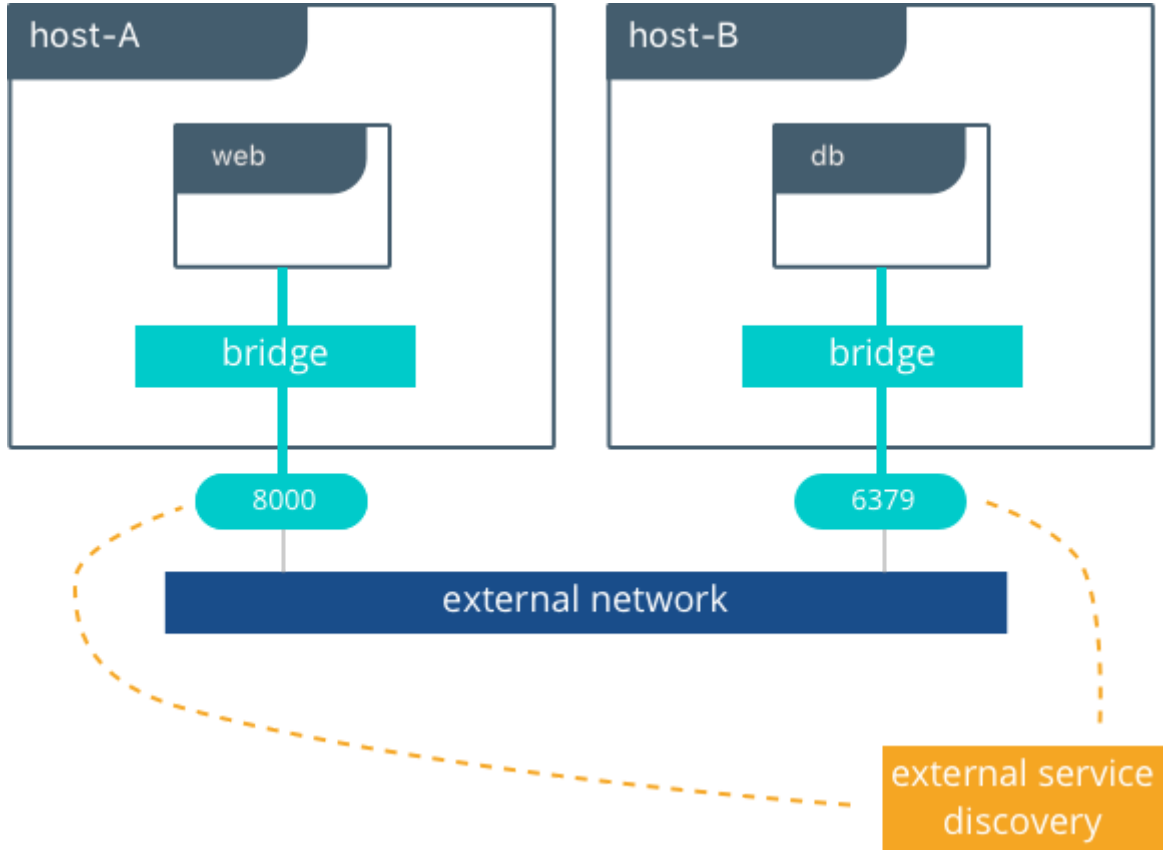
- Не работает для default bridge сети
- Доступен по адресу 127.0.0.11:53 для каждого контейнера

```
$ docker network create reddit
$ docker run -d --network=reddit  -p 9292:9292 –name=ui_reddit express42/ui:1.0
$ docker run --network reddit tutum/dnsutils nslookup ui_reddit 127.0.0.11

Server: 127.0.0.11
Address: 127.0.0.11#53

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

Внешний Service Discovery/DNS сервер



User Defined Networks

- ~~Null~~
- ~~Host~~
- Bridge
- Macvlan
- Overlay

User Defined Networks

Bridge

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

User Defined Networks

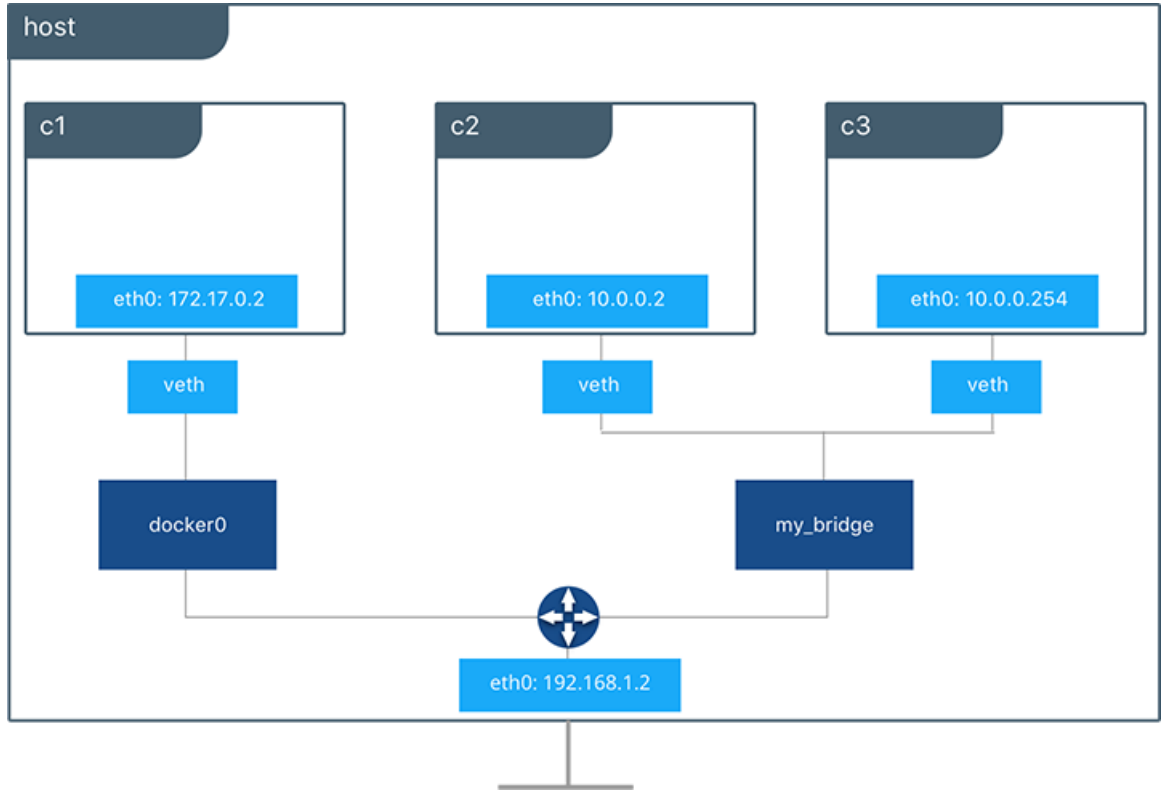
Bridge

```
$ 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 Networks

Bridge



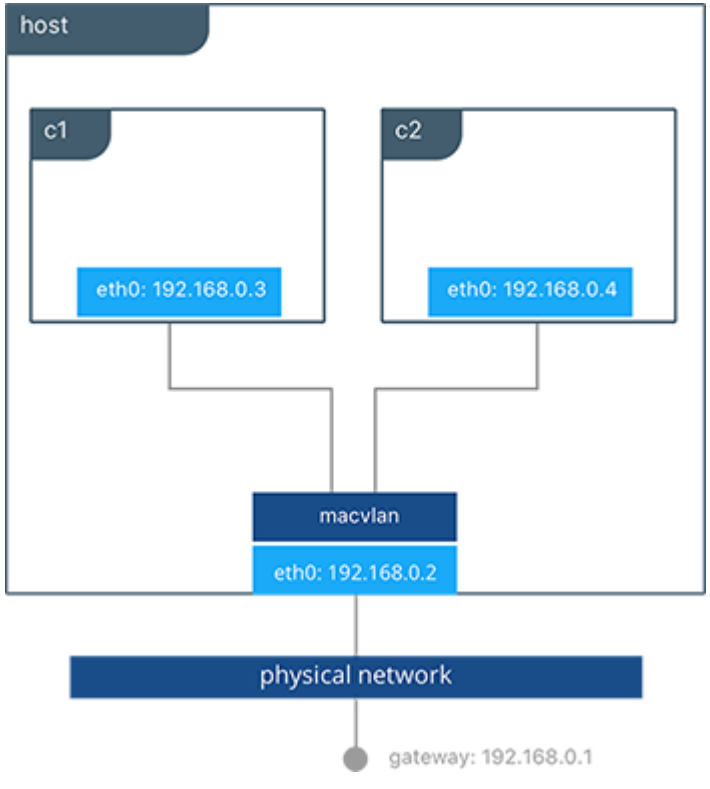
User Defined Networks

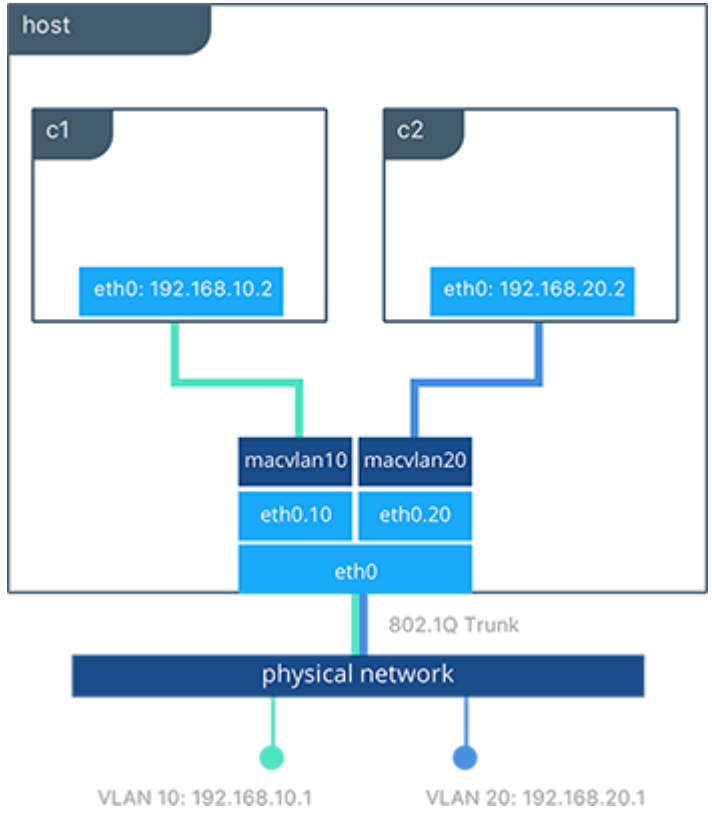
MacVlan

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

User Defined Networks

Macvlan





User Defined Networks

Macvlan

- Легко исчерпать пул DHCP
- Много MAC адресов в L2 сегменте
- Сетевой интерфейс в promiscuous mode

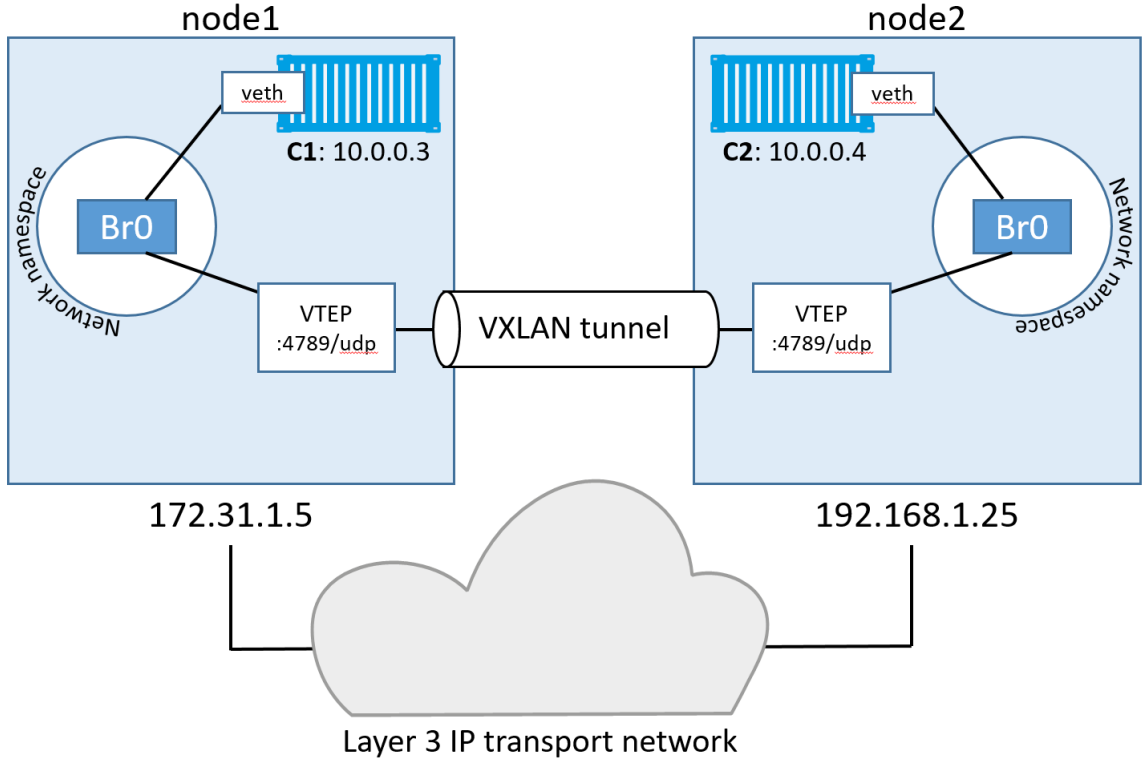
User Defined Networks

Overlay

- Позволяет объединить в одну сеть контейнеры нескольких Docker хостов
- Работает поверх VXLAN
- Необходимо хранить состояние распределенной сети

User Defined Networks

Overlay



Docker Compose

Проблемы

- Одно приложение состоит из нескольких контейнеров
- Один контейнер зависит от другого
- Порядок запуска имеет значение
- docker build/run/create ... (много и нигде не хранится)

Docker Compose

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

Docker и Docker Compose

- Dockerfile – конфигурация сборки окружения и приложений в нем
- docker-compose.yml - конфигурация проекта

Docker Compose

docker-compose.yml

```
version: "3"
services:
  mongo_db:
    image: mongo:3.2
    volumes:
      - db:/data/db
    networks:
      - reddit
  ui:
    build: ./ui
    image: ${USERNAME}/ui:1.0
    ports:
      - 9292:9292/tcp
    networks:
      - reddit
...
```

Docker Compose

docker-compose.yml

```
version: "3" – обязательная секция
...
services: – обязательная секция
...
volumes:
...
networks:
...
```

Docker Compose

Services

Описываем конфигурацию запуска контейнера или группы контейнеров

```
...
services:
  mongo_db:
    image: mongo:3.2
    volumes:
      - db:/data/db
    networks:
      - reddit
...
```

Docker Compose

Volumes

```
...
services:
  mongo_db:
    image: mongo:3.2
    volumes:
      - type: volume
        source: sample_volume
        target: /data/sample
      - db: /data/db
volumes:
  db:
  sample_volume:
```

Docker Compose

Networks

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

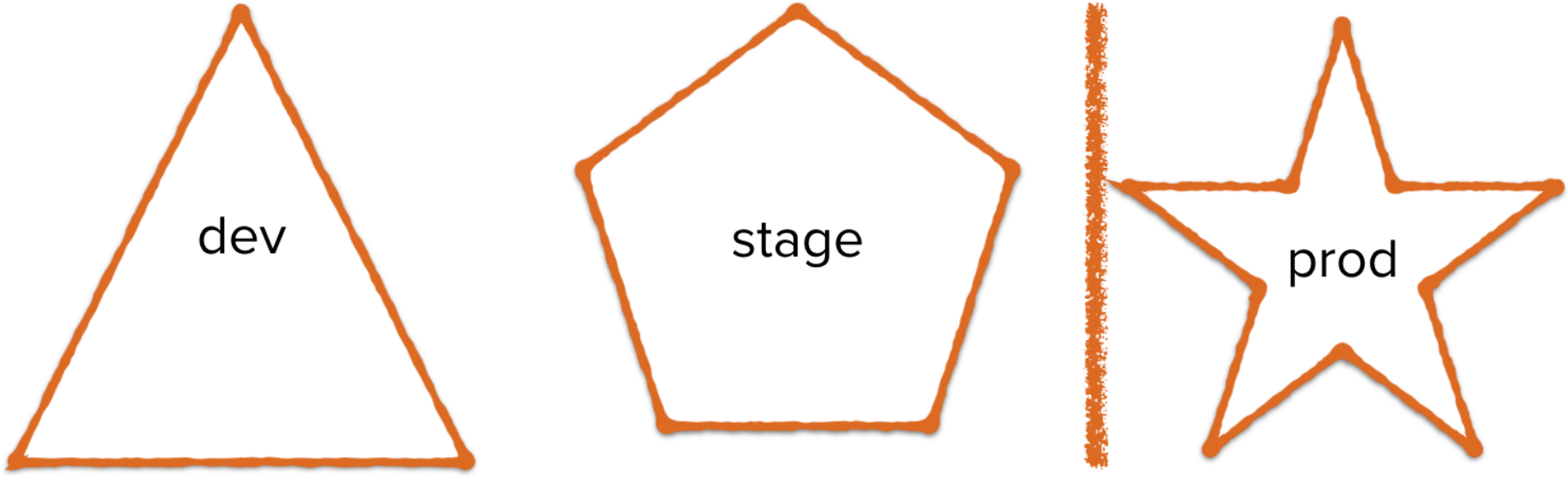
```
networks:
  reddit:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.16.238.0/24
```

Docker и работа с окружениями

Хотим получить одинаковые окружения:

Docker и работа с окружениями

В действительности получаем:



Docker Compose и работа с окружениями

- Один **docker-compose.yml** на все окружения (под проект или приложение)
- Файл **docker-compose.override.yml** с:
  - Произвольными точки монтирования для быстрого изменения кода внутри контейнера
  - ENV переменными, характерными для окружения
  - Заменой command инструкций для образов
  - Перезаписью выводимых наружу портов

Тестирование с Docker

Что тестировать?

- Содержимое образа
- Интеграционное тестирование
- Security

Тестируем образ

- Dockerfile всё еще пишут люди
- Декларативность Dockerfile неоднозначна
- Кто-то проверяет, что внутри Docker образа?

Тестируем образ

- goss - утилита для тестирования инфраструктуры
- dgoss - обертка на bash, запускающая контейнер из данного образа и выполняющая тесты
- Container-structure-test от Google

Тестируем образ

Пример файла goss.yaml

```
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
```

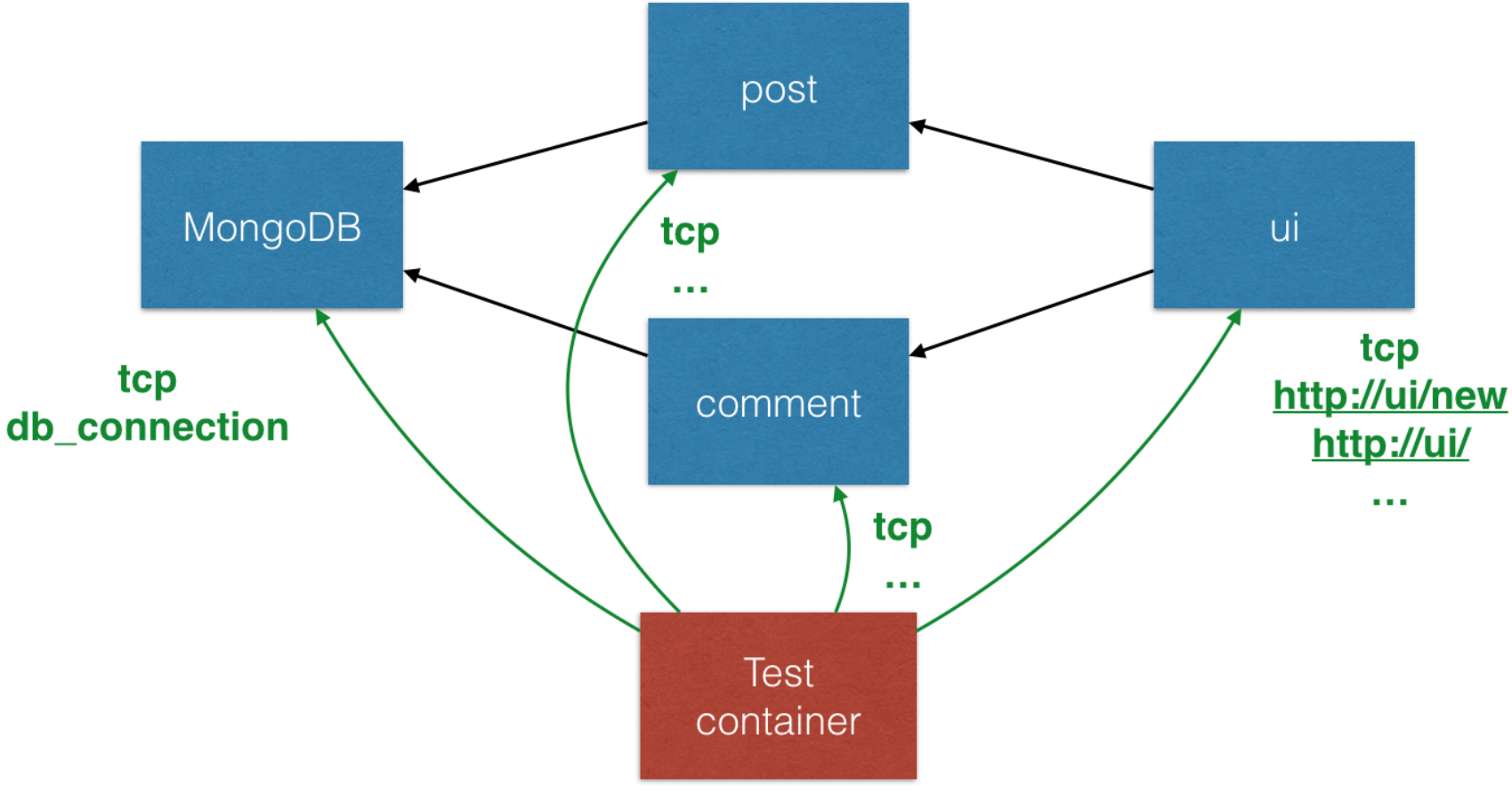
Тестируем образ

Вывод результатов тестирования

```
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



Security

- Параметры Docker
- Как написан Dockerfile
- Как запущены контейнеры
- Что внутри образов

Доклад про безопасность Docker и Kubernetes (mail.ru)

Docker Bench for Security

Healthchecks

Оператор Dockerfile:

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

Инструкция Docker Compose:

```
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost:5000/healthcheck"]
  interval: 1m30s
  timeout: 10s
  retries: 3
```

Healthchecks

```
$ docker ps

IMAGE                          COMMAND        CREATED              STATUS
express42/ui:2.14-alpine       "puma"         3 minutes ago        Up 3 minutes (healthy)
```