

Docker: сети, docker-compose

План

- Использование docker-compose
- Работа с сетями в Docker
- Тестирование в docker

Подготовка

- Работа будет осуществляться в репозитории `microservices`
- Создайте новую ветку `"homework-03"`
- Проверьте, создана ли машина `docker-host`:
 - › `docker-machine ls`

Подготовка

Если хоста нет, то создайте хост в GCP с помощью docker-machine. В качестве id проекта подставьте **свой**. (ссылка на [gist](#))

```
> docker-machine create --driver google \  
  --google-project docker-181710 \  
  --google-zone europe-west1-b \  
  --google-machine-type g1-small \  
  --google-machine-image https://www.googleapis.com/compute/v1/projects/model-  
nexus-175717/global/images/nested-ubuntu-1604-lts \  
docker-host
```

Инициализируем переменные окружения для работы с docker-engine на созданной машине:

```
> eval $(docker-machine env docker-host)
```



Работа с сетью в Docker

План

- Разобраться с работой сети в Docker
 - none
 - host
 - bridge

None network driver

Запустим контейнер с использованием none-драйвера.

В качестве образа используем **joffotron/docker-net-tools**.

Делаем это для экономии сил и времени, т.к. в его состав уже входят необходимые утилиты для работы с сетью: пакеты bind-tools, net-tools и curl.

В контейнере будет запущен процесс sleep с таймером на 100 секунд.

После истечения 100 секунд контейнер будет остановлен и удален (флаг --rm).

Запустим: (ссылка на [gist](#))

```
> docker run --network none --rm -d --name net_test  
joffotron/docker-net-tools -c "sleep 100"
```

None network driver

Выполним:

```
> docker exec -ti net_test ifconfig
```

```
lo          Link encap:Local Loopback  
            inet addr:127.0.0.1  Mask:255.0.0.0  
            UP LOOPBACK RUNNING  MTU:65536  Metric:1  
            RX packets:6 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:6 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:504 (504.0 B)  TX bytes:504 (504.0 B)
```


None network driver

В результате, видим:

- что внутри контейнера из сетевых интерфейсов существует только loopback.
- сетевой стек самого контейнера работает (ping localhost), но без возможности контактировать с внешним миром.
- Значит, можно даже запускать сетевые сервисы внутри такого контейнера, но лишь для локальных экспериментов (тестирование, контейнеры для выполнения разовых задач и т.д.)

Host network driver

Запустим контейнер в сетевом пространстве docker-хоста
(ссылка на [gist](#))

```
> docker run --network host --rm -d --name net_test  
joffotron/docker-net-tools -c "sleep 100"
```

Сравните выводы команд:

```
> docker exec -ti net_test ifconfig  
  
> docker-machine ssh docker-host ifconfig
```

Host network driver

Запустите несколько раз (2-4) (ссылка на [gist](#))

```
> docker run --network host -d nginx
```

Каков результат? Что выдал docker ps? Как думаете почему?

Остановите все запущенные контейнеры:

```
docker kill $(docker ps -q)
```



Docker networks

На docker-host машине выполните команду: (ссылка на [gist](#))

```
> sudo ln -s /var/run/docker/netns /var/run/netns
```

Теперь вы можете просматривать существующие в данный момент net-namespaces с помощью команды:

```
> sudo ip netns
```

Задание:

Повторите запуски контейнеров с использованием драйверов none и host и посмотрите, как меняется список namespace-ов.

Прим: `ip netns exec <namespace> <command>` - ПОЗВОЛИТ ВЫПОЛНЯТЬ КОМАНДЫ В выбранном namespace



Bridge network driver

Создадим bridge-сеть в docker (флаг `--driver` указывать не обязательно, т.к. по-умолчанию используется bridge).

```
> docker network create reddit --driver bridge
```

Запустим наш проект reddit с использованием bridge-сети. (ссылка на [gist](#))

```
> docker run -d --network=reddit mongo:latest
```

```
> docker run -d --network=reddit <your-login>/post:1.0
```

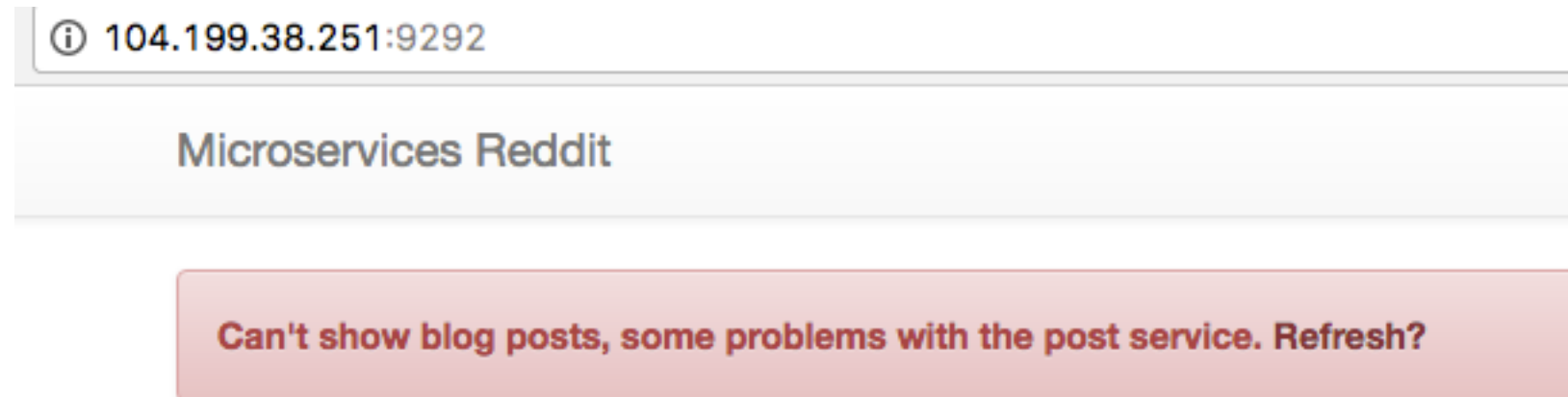
```
> docker run -d --network=reddit <your-login>/comment:1.0
```

```
> docker run -d --network=reddit -p 9292:9292 <your-login>/ui:1.0
```

Зайдем на адрес `http://<your-machine>:9292`

Bridge network driver

Что то пошло не так...



На самом деле, наши сервисы ссылаются друг на друга по dns-именам, прописанным в ENV-переменных (см Dockerfile). В текущей инсталляции встроенный DNS docker не знает ничего об этих именах.

Решением проблемы будет присвоение контейнерам имен или сетевых алиасов при старте:

`--name <name>` (можно задать только 1 имя)

`--network-alias <alias-name>` (можно задать множество алиасов)

Bridge network driver

- Остановим старые копии контейнеров

```
> docker kill $(docker ps -q)
```

- Запустим новые (ссылка на [gist](#))

```
> docker run -d --network=reddit --network-alias=post_db --network-alias=comment_db  
mongo:latest
```

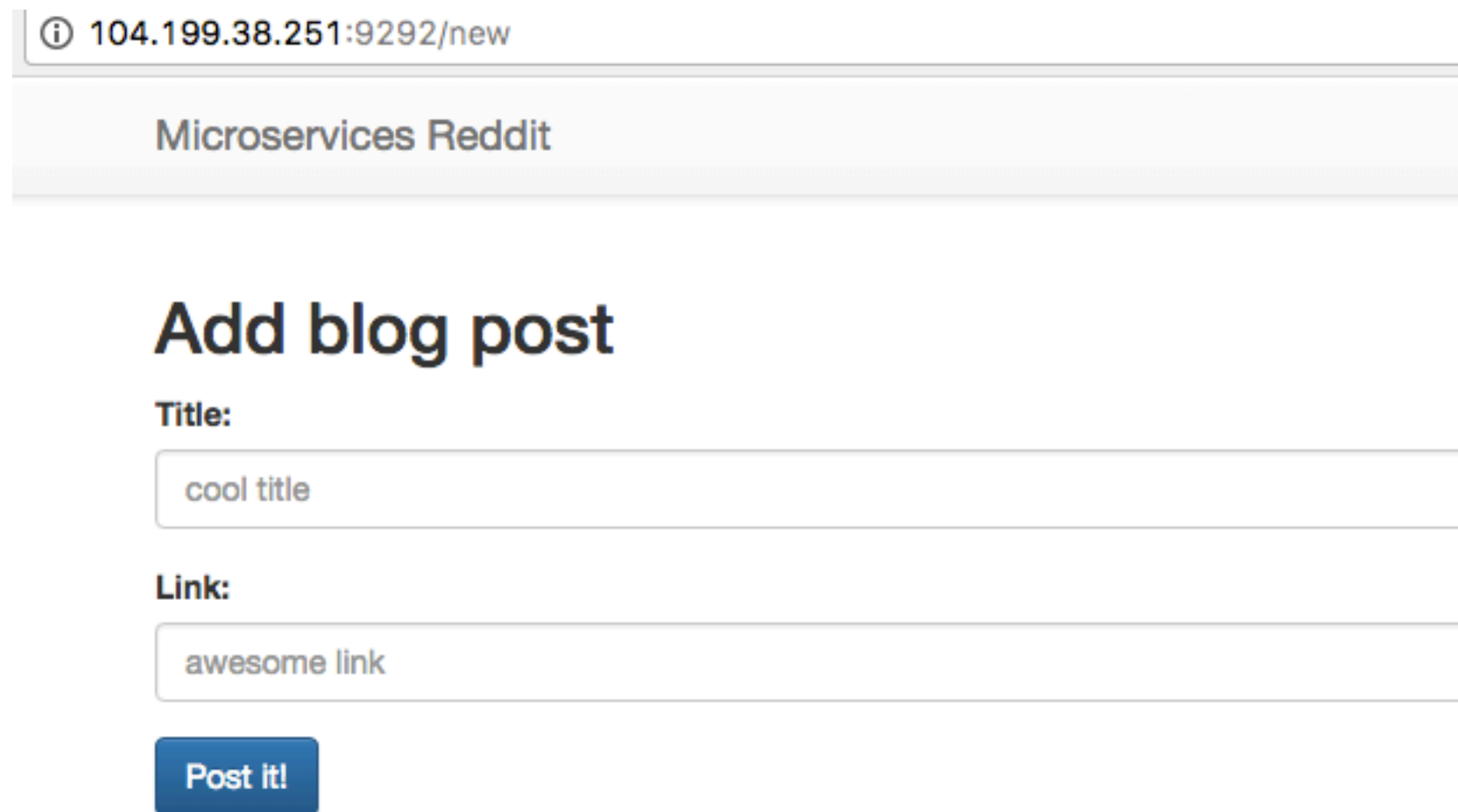
```
> docker run -d --network=reddit --network-alias=post <your-login>/post:1.0
```

```
> docker run -d --network=reddit --network-alias=comment <your-login>/comment:1.0
```

```
> docker run -d --network=reddit -p 9292:9292 <your-login>/ui:1.0
```

Bridge network driver

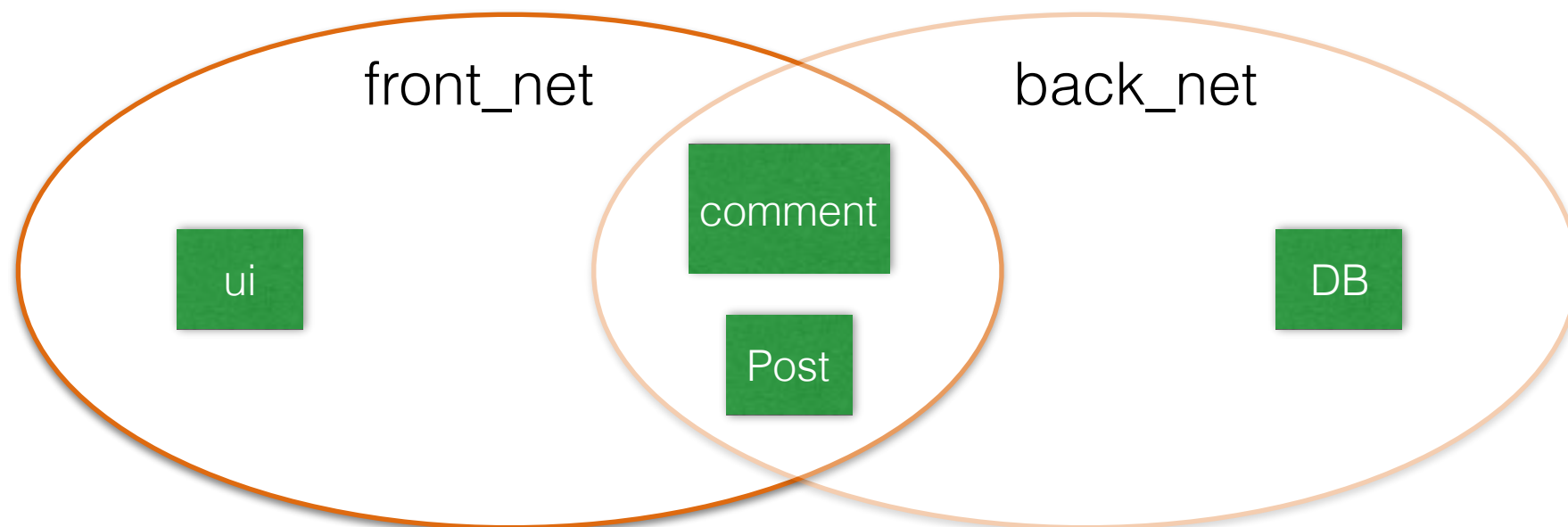
- Зайдем на адрес `http://<your-machine>:9292`



The screenshot shows a web browser window with the address bar displaying `104.199.38.251:9292/new`. The page title is 'Microservices Reddit'. The main content area features a heading 'Add blog post' followed by two input fields: 'Title:' with the placeholder text 'cool title' and 'Link:' with the placeholder text 'awesome link'. Below these fields is a blue button labeled 'Post it!'.

Bridge network driver

Давайте запустим наш проект в 2-х bridge сетях. Так , чтобы сервис ui не имел доступа к базе данных в соответствии со схемой ниже.



Bridge network driver

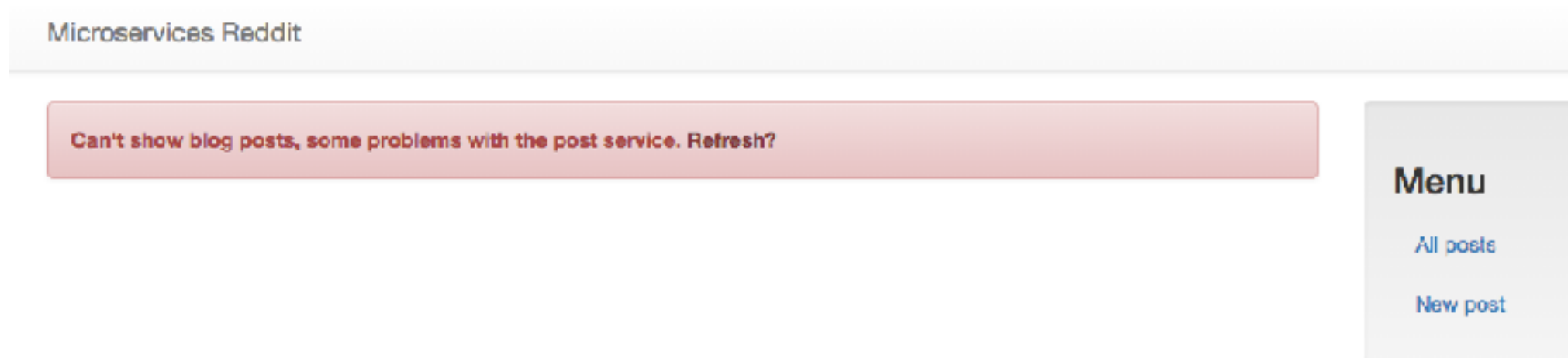
- Остановим старые копии контейнеров
 - > `docker kill $(docker ps -q)`
- Создадим docker-сети (ссылка на [gist](#))
 - > `docker network create back_net --subnet=10.0.2.0/24`
 - > `docker network create front_net --subnet=10.0.1.0/24`

Bridge network driver

- Запустим контейнеры (ссылка на [gist](#))

```
> docker run -d --network=front_net -p 9292:9292 --name ui <your-login>/ui:1.0  
> docker run -d --network=back_net --name comment <your-login>/comment:1.0  
> docker run -d --network=back_net --name post <your-login>/post:1.0  
> docker run -d --network=back_net --name mongo_db --network-alias=post_db --  
network-alias=comment_db mongo:latest
```

- Зайдем на адрес `http://<your-machine>:9292`



Bridge network driver

- Что пошло не так?

Docker при инициализации контейнера может подключить к нему только 1 сеть.

При этом контейнеры из соседних сетей не будут доступны как в DNS, так и для взаимодействия по сети.

Поэтому нужно поместить контейнеры **post** и **comment** в обе сети.

Дополнительные сети подключаются командой:

```
> docker network connect <network> <container>
```

Bridge network driver

- Подключим контейнеры ко второй сети (ссылка на [gist](#))
 - > `docker network connect front_net post`
 - > `docker network connect front_net comment`
- Зайдем на адрес `http://<your-machine>:9292`

Microservices Reddit

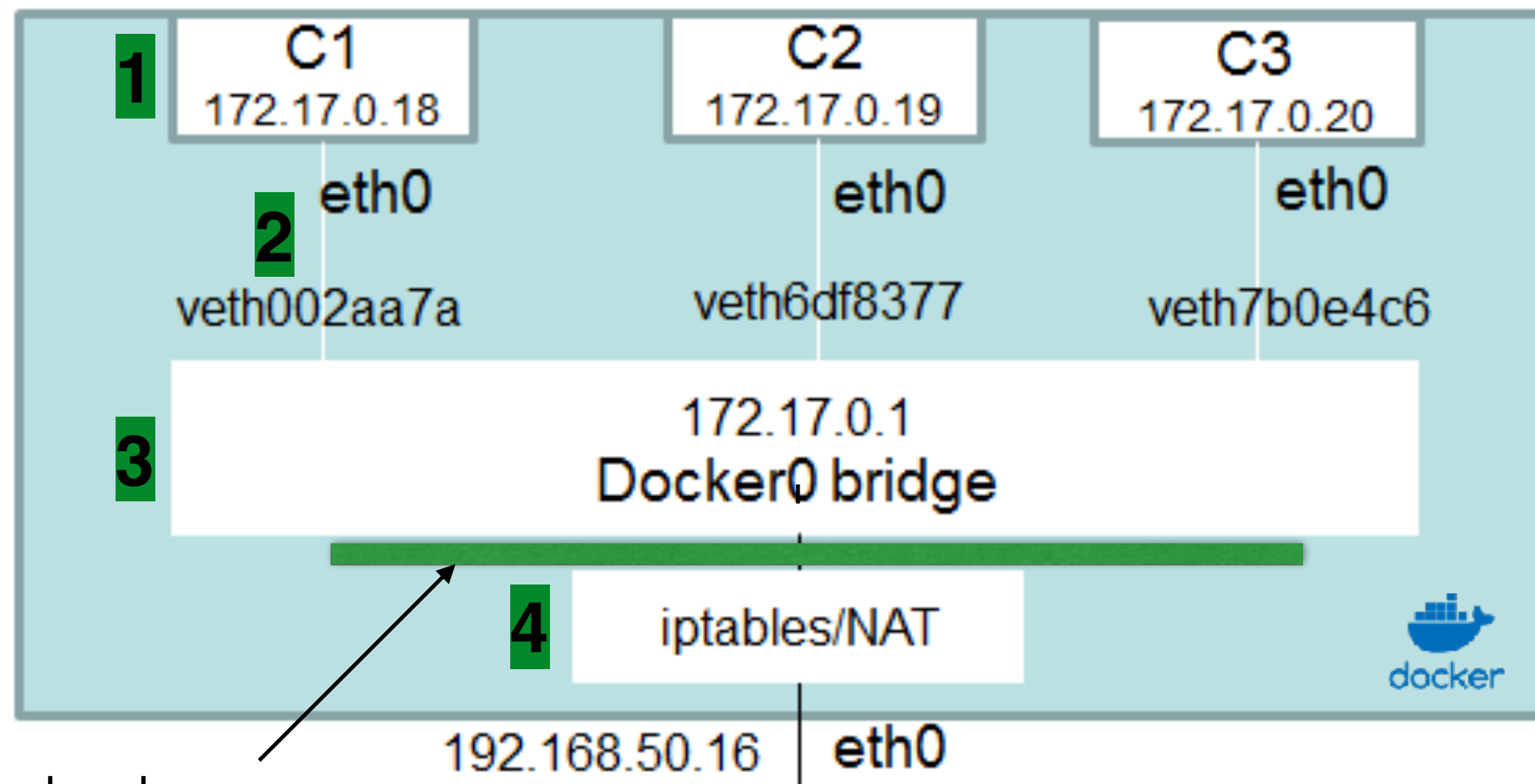
Menu

[All posts](#)

[New post](#)



Bridge network driver



docker-proxy



Bridge network driver

Давайте посмотрим как выглядит сетевой стек Linux в текущий момент, опираясь на схему из предыдущего слайда:

- 1) Зайдите по ssh на docker-host и установите пакет bridge-utils (ссылка на [gist](#)) :

```
> docker-machine ssh docker-host  
> sudo apt-get update && sudo apt-get install bridge-utils
```

- 2) Выполните:

```
> docker network ls
```

- 3) найдите ID сетей, созданных в рамках проекта.



Bridge network driver

4) Выполните:

```
> ifconfig | grep br
```

5) Найдите bridge-интерфейсы для каждой из сетей. Просмотрите информацию о каждом.

6) Выберите любой из bridge-интерфейсов и выполните команду. Ниже пример вывода:

```
> brctl show <interface>
```

bridge name	bridge id	STP enabled	interfaces
br-4ac81d1bf266	8000.0242ae9beade	no	vethaf41855 vethe115d8d

Отображаемые veth-интерфейсы - это те части виртуальных пар интерфейсов (2 на схеме), которые лежат в сетевом пространстве хоста и также отображаются в ifconfig. Вторые их части лежат внутри контейнеров





Bridge network driver

7) Давайте посмотрим как выглядит iptables. Выполним:

> `sudo iptables -nL -t nat` (флаг -v даст чуть больше инфы)

Обратите внимание на цепочку **POSTROUTING**. В ней вы увидите нечто подобное

Chain **POSTROUTING** (policy ACCEPT)

target	prot	opt	source	destination	
MASQUERADE	all	--	10.0.2.0/24	0.0.0.0/0	
MASQUERADE	all	--	172.18.0.0/16	0.0.0.0/0	
MASQUERADE	all	--	172.17.0.0/16	0.0.0.0/0	
MASQUERADE	tcp	--	172.18.0.2	172.18.0.2	tcp dpt:9292

Выделенные правила отвечают за выпуск во внешнюю сеть контейнеров из bridge-сетей



Bridge network driver

8) В ходе работы у нас была необходимость публикации порта контейнера UI (9292) для доступа к нему снаружи.

Давайте посмотрим, что Docker при этом сделал.

Снова взгляните в iptables на таблицу nat.

Обратите внимание на цепочку DOCKER и правила DNAT в ней.

```
DNAT      tcp  --  0.0.0.0/0          0.0.0.0/0          tcp dpt:9292 to:
172.18.0.2:9292
```

Они отвечают за перенаправление трафика на адреса уже конкретных контейнеров.

9) Также выполните:

```
> ps ax | grep docker-proxy
```

Вы должны увидеть хотя бы 1 запущенный процесс docker-proxy.

Этот процесс в данный момент слушает сетевой tcp-порт 9292.



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

План

- Установить docker-compose на локальную машину
- Собрать образы приложения reddit с помощью docker-compose
- Запустить приложение reddit с помощью docker-compose

docker-compose

Установка

- MacOS - идет в docker-бандле
(<https://docs.docker.com/docker-for-mac/install/>)
- Windows - идет в docker-бандле
(<https://docs.docker.com/docker-for-windows/install/>)
- Linux - (<https://docs.docker.com/compose/install/#install-compose>)
либо

```
> pip install docker-compose
```

docker-compose.yml

В директории с проектом создайте файл docker-compose.yml.
Важно, чтобы папки ui, comment, post находились рядом. (ссылка на [gist](#))

docker-compose.yml

```
version: '3.3'
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
```

продолжение далее

docker-compose.yml

docker-compose.yml

```
...
post:
  build: ./post-py
  image: ${USERNAME}/post:1.0
  networks:
    - reddit
comment:
  build: ./comment
  image: ${USERNAME}/comment:1.0
  networks:
    - reddit
volumes:
  post_db:

networks:
  reddit:
```



docker-compose

Отметим, что docker-compose поддерживает интерполяцию(подстановку) переменных окружения.

В данном случае это переменная USERNAME.

Поэтому перед запуском необходимо экспортировать значения данных переменных окружения.

Выполните:

- > `export USERNAME=<your-login>`
- > `docker-compose up -d`
- > `docker-compose ps`

docker-compose

В выводе вы должны увидеть похожую картину

Name	Command	State	Ports
hw17_comment_1	puma	Up	
hw17_mongo_db_1	docker-entrypoint.sh mongod	Up	27017/tcp
hw17_post_1	python3 post_app.py	Up	
hw17_ui_1	puma	Up	0.0.0.0:9292->9292/tcp

Зайдите на <http://<your-machine>:9292/>
и убедитесь, что проект работает правильно

docker-compose.yml

Задание:

- 1) Изменить docker-compose под кейс с множеством сетей, сетевых алиасов (стр 18).
 - 2) Параметризуйте с помощью переменных окружений:
 - порт публикации сервиса ui
 - версии сервисов
 - возможно что-либо еще на ваше усмотрение
 - 3) Параметризованные параметры запишите в отдельный файл с расширением .env
 - 4) Без использования команд **source** и **export** docker-compose должен подхватить переменные из этого файла. Проверьте
- P.S. В git лучше коммитить файл с расширением вроде .env.example, в будущем от него продюцировать файл с расширением .env

docker-compose

При запуске проекта вы увидите, что все создаваемые docker-compose сущности имеют одинаковый префикс (Напр: hw17_ui_1).

hw17 - базовое имя проекта.

Задание:

Узнайте как образуется базовое имя проекта. Можно ли его задать? Если можно то как?

Ответ присылайте в slack-чат пользователю **chromko**.

Проверочное задание

- Создайте PR с вашими наработками
- Пригласите одного из следующих преподавателей на review вашего PR:
 - chromko