Сети Docker изнутри, связь между контейнерами в Docker Swarm и Overlay-сети

**5 мин**

**60K**

[Системное администрирование\*](https://habr.com/ru/hubs/sys_admin/)[Сетевые технологии\*](https://habr.com/ru/hubs/network_technologies/)[DevOps\*](https://habr.com/ru/hubs/devops/)

Перевод

[Автор оригинала: Abdul Kittana](http://securitynik.blogspot.ru/2016/12/docker-networking-internals-container.html)

В [предыдущей статье](https://habrahabr.ru/post/333874/) я рассказал, как Docker использует виртуальные интерфейсы Linux и bridge-интерфейсы, чтобы установить связь между контейнерами по bridge-сетям. В этот раз я расскажу, как Docker использует технологию vxlan, чтобы создавать overlay-сети, которые используются в swarm-кластерах, а также где можно посмотреть и проинспектировать эту конфигурацию. Также я расскажу, как различные типы сетей решают разные задачи связи для контейнеров, которые запущены в swarm-кластерах.

Я предполагаю, что читатели уже знают, как разворачивать swarm-кластеры и запускать сервисы в Docker Swarm. Также в конце статьи я приведу несколько ссылок на полезные ресурсы, с помощью которых можно будет изучить предмет в деталях и вникнуть в контекст обсуждаемых здесь тем. Опять же, буду ждать ваших мнений в комментариях.

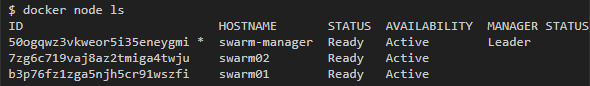
Оглавление

* [Docker Swarm и Overlay-сети](https://habr.com/ru/articles/334004/#docker-swarm-i-overlay-seti)  
  + [Overlay](https://habr.com/ru/articles/334004/#overlay)  
    - [vxlan](https://habr.com/ru/articles/334004/#vxlan)
    - [Шифрование](https://habr.com/ru/articles/334004/#shifrovanie)
    - [Инспектирование интерфейсов vxlan-туннелей](https://habr.com/ru/articles/334004/#inspektirovanie-interfeysov-vxlan-tunneley)
  + [ingress](https://habr.com/ru/articles/334004/#ingress)
  + [Docker\_gwbridge](https://habr.com/ru/articles/334004/#docker_gwbridge)
* [Заключение](https://habr.com/ru/articles/334004/#zaklyuchenie)
* [Ссылки/ресурсы](https://habr.com/ru/articles/334004/#ssylkiresursy)

Docker Swarm и Overlay-сети

Overlay-сети используются в контексте кластеров (Docker Swarm), где виртуальная сеть, которую используют контейнеры, связывает несколько физических хостов, на которых запущен Docker. Когда вы запускаете контейнер на swarm-кластере (как часть сервиса), множество сетей присоединяется по умолчанию, и каждая из них соответствует разным требованиям связи.

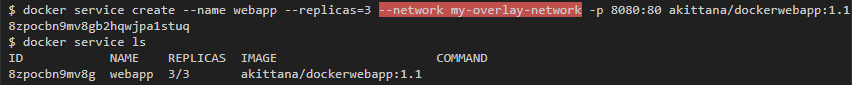
Например, у меня есть 3 ноды docker swarm кластера:



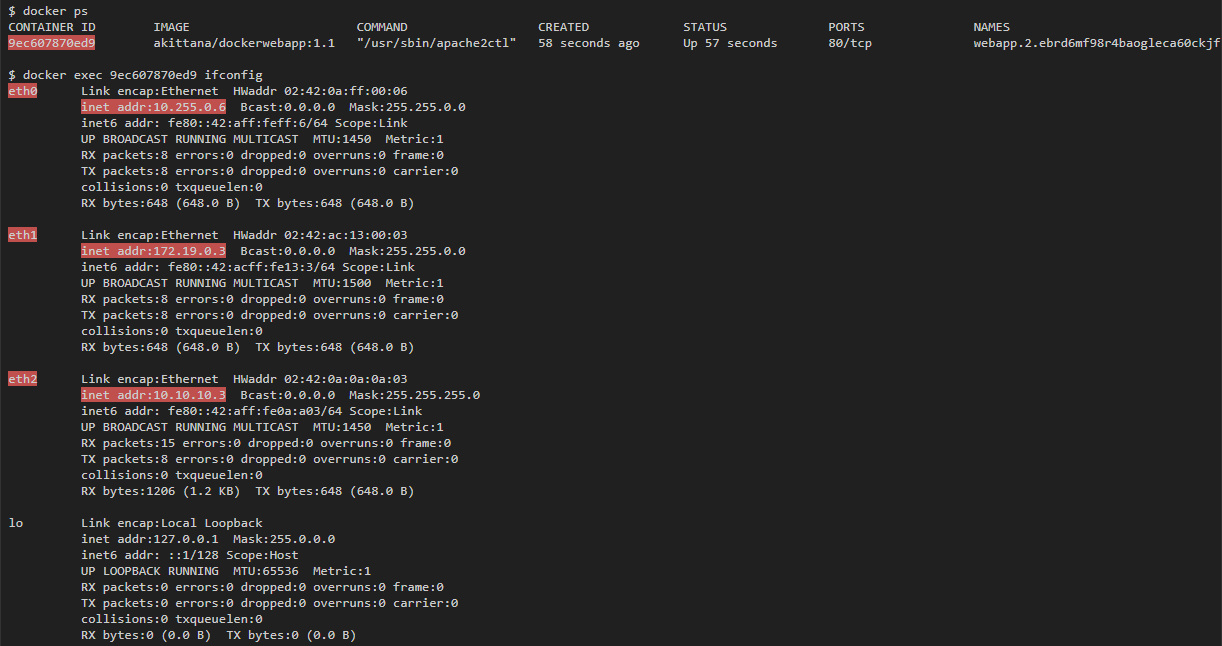
Для начала я создам overlay-сеть под названием my-overlay-network:

docker network create

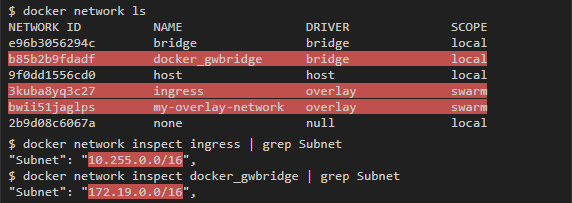
Затем запущу сервис с контейнером, на котором запущен простой веб-сервер, который смотрит портом 8080 во внешний мир. Этот сервис будет иметь 3 реплики, и я отмечу, что он связан только с одной сетью (my-overlay-network):



Если затем вывести список всех интерфейсов, доступных любому запущенному контейнеру, то их будет 3. В то же время, если запустить контейнер на одном хосте, то можно ожидать только 1 интерфейс:



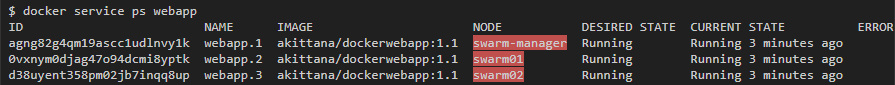
Контейнер связан с my-overlay-network через eth2, что можно понять по IP-адресу. eth0 и eth1 связаны с другими сетями. Если запустить docker network ls, то можно увидеть 2 дополнительные сети, которые добавились: docker\_gwbridge и ingress, а по адресам подсетей можно понять, что они привязаны к eth0 и eth1:



Overlay

Overlay-сеть создает подсеть, которую могут использовать контейнеры в разных хостах swarm-кластера. Контейнеры на разных физических хостах могут обмениваться данными по overlay-сети (если все они прикреплены к одной сети).

Например, для веб-приложения, которое мы запустили, можно увидеть по одному контейнеру на каждом хосте в swarm-кластере:



Я могу получить overlay IP-адрес для каждого контейнера при помощи команды ifconfig eth2 (eth2 — это интерфейс, присоединенный к overlay-сети).

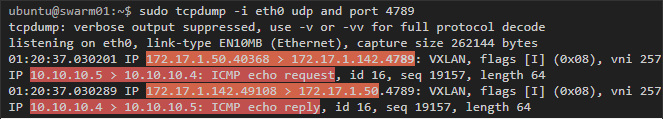
На swarm01:

Потом с контейнера на swarm02 у меня должна быть возможность пингануть 10.10.10.5 (IP контейнера на swarm01):

vxlan

Overlay-сеть использует технологию vxlan, которая инкапсулирует layer 2 фреймы в layer 4 пакеты (UDP/IP). При помощи этого действия Docker создает виртуальные сети поверх существующих связей между хостами, которые могут оказаться внутри одной подсети. Любые точки, которые являются частью этой виртуальной сети, выглядят друг для друга так, будто они связаны поверх свича и не заботятся об устройстве основной физической сети.

Чтобы увидеть этот процесс в действии, можно сделать захват трафика на хостах, которые являются частью overlay-сети. В последнем примере захват трафика на swarm01 или swarm02 выявит icmp-трафик между контейнерами, которые на них запущены (vxlan использует udp port 4789):

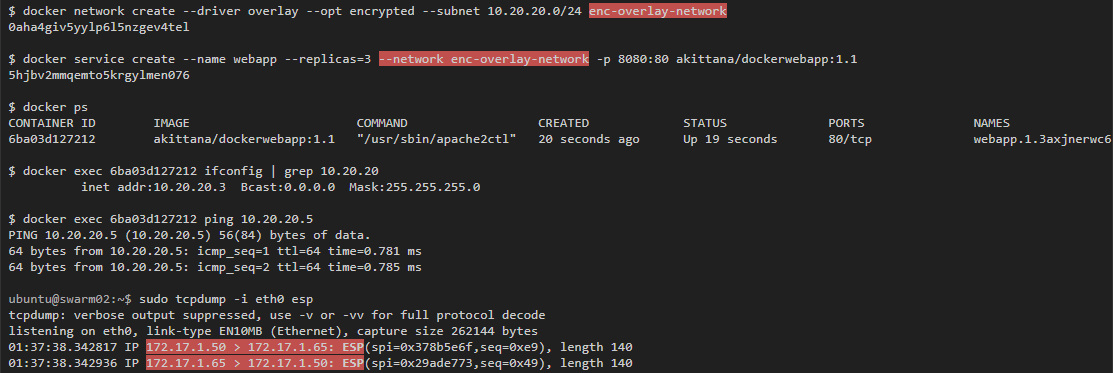


В этом примере в пакетах можно видеть два слоя. Первый — это туннельный трафик udp vxlan между хостами по порту 4789, а внутри можно увидеть второй — icmp трафик с IP-адресами контейнера.

Шифрование

Захват трафика в этом примере показал, что если ты видишь трафик между хостами, то увидишь и трафик внутри контейнеров, проходящий по overlay-сети. Именно поэтому в Docker есть опция шифроования. Можно запустить автоматическое IPSec-шифрование vxlan-туннелей, просто добавив --opt encrypted при создании сети.

Если запустить такой же тест, но с использованием зашифрованной overlay-сети, то можно увидеть только зашифрованные пакеты между хостами:

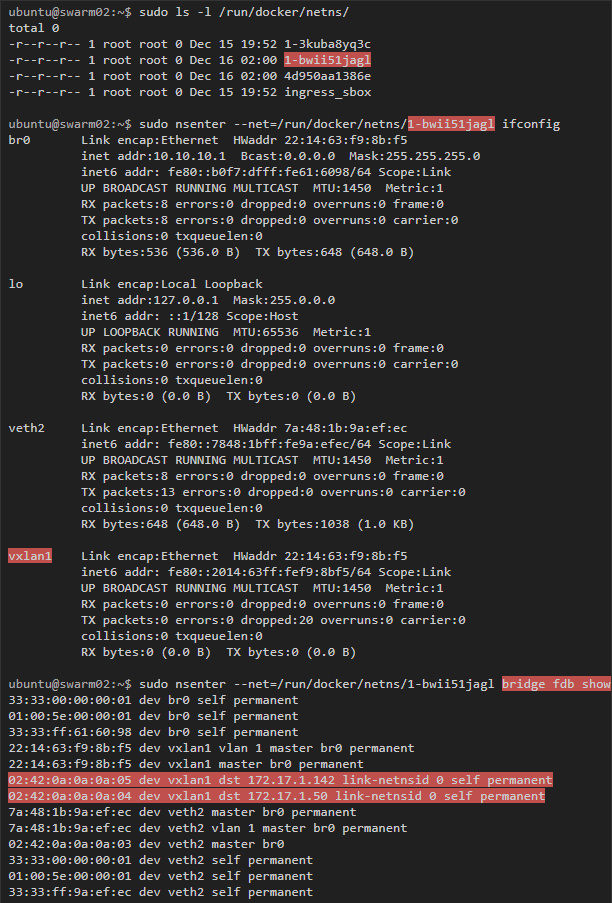


Инспектирование интерфейсов vxlan-туннелей

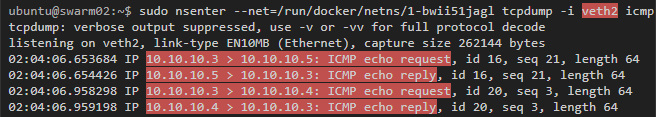
Как и bridge-сети, Docker создает bridge-интерфейс для каждой overlay-сети, который соединяет виртуальные туннельные интерфейсы, выполняющие vxlan туннельную связь между хостами. Впрочем, эти туннельные интерфейсы (bridge и vxlan) создаются не напрямую на туннельном хосте. Они находятся в разных контейнерах, которые Docker запускает для каждой создаваемой overlay-сети.

Чтобы действительно проинспектировать эти интерфейсы, надо использовать nsenter для запуска команд внутри сети контейнера, который управляет туннелями и виртуальными интерфейсами. Эту команду надо запустить на хостах с контейнерами, которые участвуют в overlay-сети.

Также надо отредактировать /etc/systemd/system/multi-user.target.wants/docker.service на хосте и закомментировать MountFlags=slave по инструкции из [этого обсуждения](https://forums.docker.com/t/unable-to-check-docker-overlay-network-namespace/17267/3).



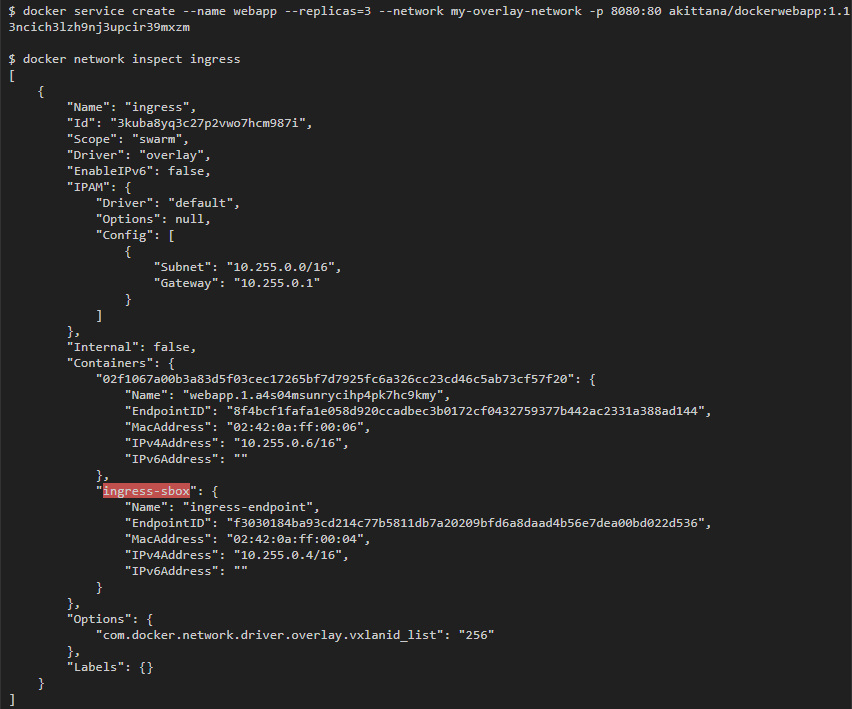
Наконец, если запустить захват трафика на veth-интерфейсе, то мы увидим трафик, который покидает контейнер, но до того, как он будет направлен в vxlan туннель (упомянутый выше пинг все еще работает):



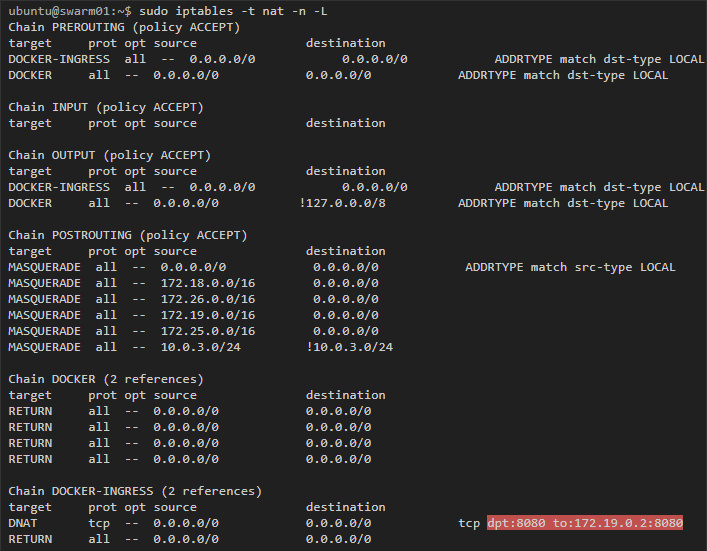
ingress

Вторая сеть, к которой были присоединены контейнеры, — это сеть ingress. Это overlay-сеть, но она устанавливается по умолчанию сразу после запуска swarm-кластера. Эта сеть отвечает за связи, которые устанавливаются с контейнерами со стороны внешнего мира. Также именно в ней происходит балансировка нагрузки, которую предоставляет swarm-кластер.

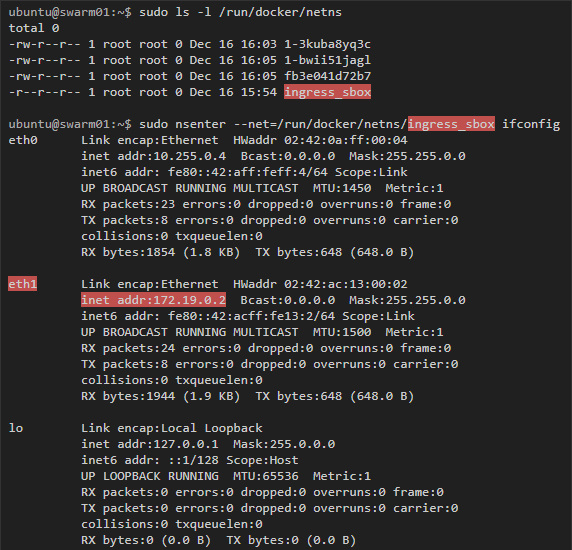
Балансировку нагрузки выполняет IPVS в контейнере, который Docker swarm запускает по умолчанию. Можно увидеть, что этот контейнер прикреплен к ingress-сети (я использовал тот же веб-сервис, что и раньше: он раскрывает порт 8080, который прикрепляется к порту 80 в контейнерах):



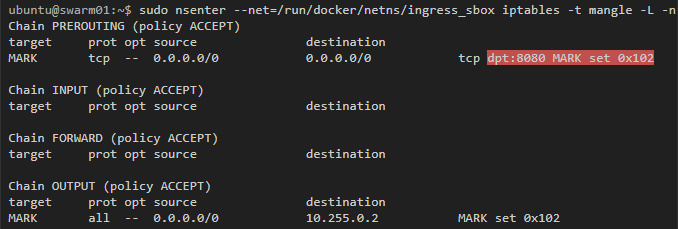
Для начала, взглянем на хост — на любой хост, который участвует в swarm-кластере:



Здесь мы видим правило, по которому трафик, предназначенный для порта 8080, перенаправляется по адресу 172.19.0.2. Этот адрес принадлежит контейнеру ingress-sbox, если проинспектировать его интерфейсы, то мы получим слещующее:



Docker использует mangle-правила iptables, чтобы назначить определенный номер пакетам для порта 8080. IPVS будет использовать этот номер, чтобы балансировать нагрузку в подходящие контейнеры:



Как Docker swarm использует iptables и IPVS для балансировки нагрузки контейнеров, можно более детально изучить по видеоролику [Deep Dive into Docker 1.12 Networking](https://www.youtube.com/watch?v=nXaF2d97JnE).

Docker\_gwbridge

Наконец, поговорим о сети docker\_gwbridge. Это bridge-сеть с соответствующим интерфейсом под названием docker\_gwbridge, который создается на каждом хосте swarm-кластера. Сеть docker\_gwbridge соединяет трафик из контейнеров swarm-кластера с внешним миром. Например, такой трафик получится, если мы направим запрос в Google.

Не буду вдаваться в подробности, поскольку bridge-сети я уже в деталях рассмотрел в [предыдущей статье](https://habrahabr.ru/post/333874/).

Заключение

Контейнер, запущенный на swarm-кластере, по умолчанию может быть соединён с тремя и более сетями. Первая сеть, docker\_gwbridge, позволяет контейнерам поддерживать связь с внешним миром. Сеть ingress нужна только для того, чтобы устанавливать входящие соединения из внешнего мира. И, наконец, сети overlay: их создает сам пользователь и их можно прикрепить к контейнерам. Эти сети служат общей подсетью для контейнеров единой сети, в которой они могут обмениваться данными напрямую (даже если они запущены на разных физических хостах).

Также существуют пространства разных сетей, которые создаются по умолчанию на swarm-кластере. Эти пространства помогают управлять vxlan-туннелями для overlay-сетей и правилами балансировки нагрузки для входящих связей.

Ссылки/ресурсы

* [Deep dive into Docker Networking](https://www.youtube.com/watch?v=nXaF2d97JnE) (видео)
* [Overlay networking and swarm mode](https://docs.docker.com/engine/userguide/networking/get-started-overlay/#/overlay-networking-and-swarm-mode)
* [Docker Swarm overlay security](https://docs.docker.com/engine/userguide/networking/overlay-security-model/)