Container network

Docker Network

- 컨테이너에 네트워크 연결을 가능하게 하는 기능
- 실제 물리적인 네트워크 장치가 아닌 가상 인터페이스 사용
- 기본 네트워크로 bridge, host, none 세개의 네트워크 생성

- Container ip 주소 확인

```
docker inspect -f "{{ .NetworkSettings.IPAddress }}" <CONTAINER>
```

- 도커 네트워크 목록 확인

```
> docker network ls
> docker network ls --no-trunc
```

- 도커 네트워크 세부 정보 확인

```
> docker network inspect <NETWORK>
❯ docker network inspect bridge
```

- 네트워크 지정하여 컨테이너 실행

```
> docker container run --network <NETWORK --name <CONTAINER> IMAGE:TAG
```

- 도커 네트워크 생성

```
> docker network create --driver=[TYPE] <NETWORK>
```

도커 네트워크 유형

- bridge : 호스트의 docker0 bridg를 사용하여 외부와 연결
- host : 호스트의 네트워크를 그대로 사용
- null : 네트워크 기능 사용하지 않음
- macvlan : 호스트와 같은 네트워크에서 별도의 MAC을 사용하여 IP할당
- container : 다른 컨테이너의 네트워크 설정을 사용
- overlay: 클러스터 환경을 위해 사용하는 네트워크

- 컨테이너에 네트워크 연결

```
$ docker network connect <NETWORK> <CONTAINER>
```

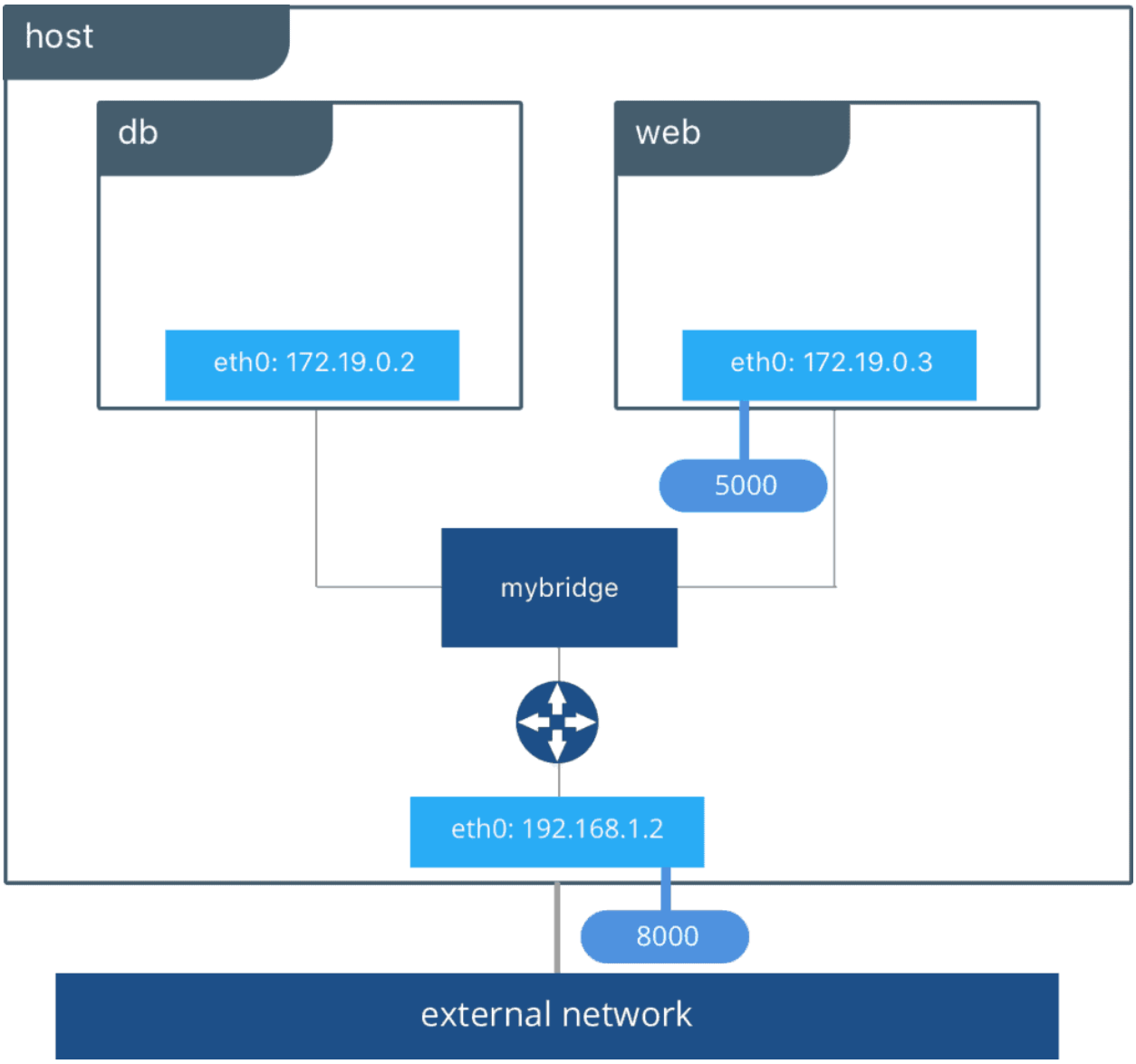
- 컨테이너에서 네트워크 연결 해제

```
$ docker network disconnect <NETWORK> <CONTAINER>
```

- 컨테이너 네트워크 삭제

```
$ docker network disconnect <NETWORK> <CONTAINER>
$ docker network rm <NETWORK>
```

bridge



- Docker 내부에 사설 네트워크를 구성
- Docker Host의 기본 브리지 네트워크 인터페이스 : docker0
- Docker Container의 기본 브리지 인터페이스 : vethXXX
- 호스트의 인터페이스를 사용해서 외부 네트워크와 연결가능
    - 외부로 통신할 때에는 NAPT 처리 필요

- bridge 네트워크로 연결하기 위해서는 포트 포워딩 필요

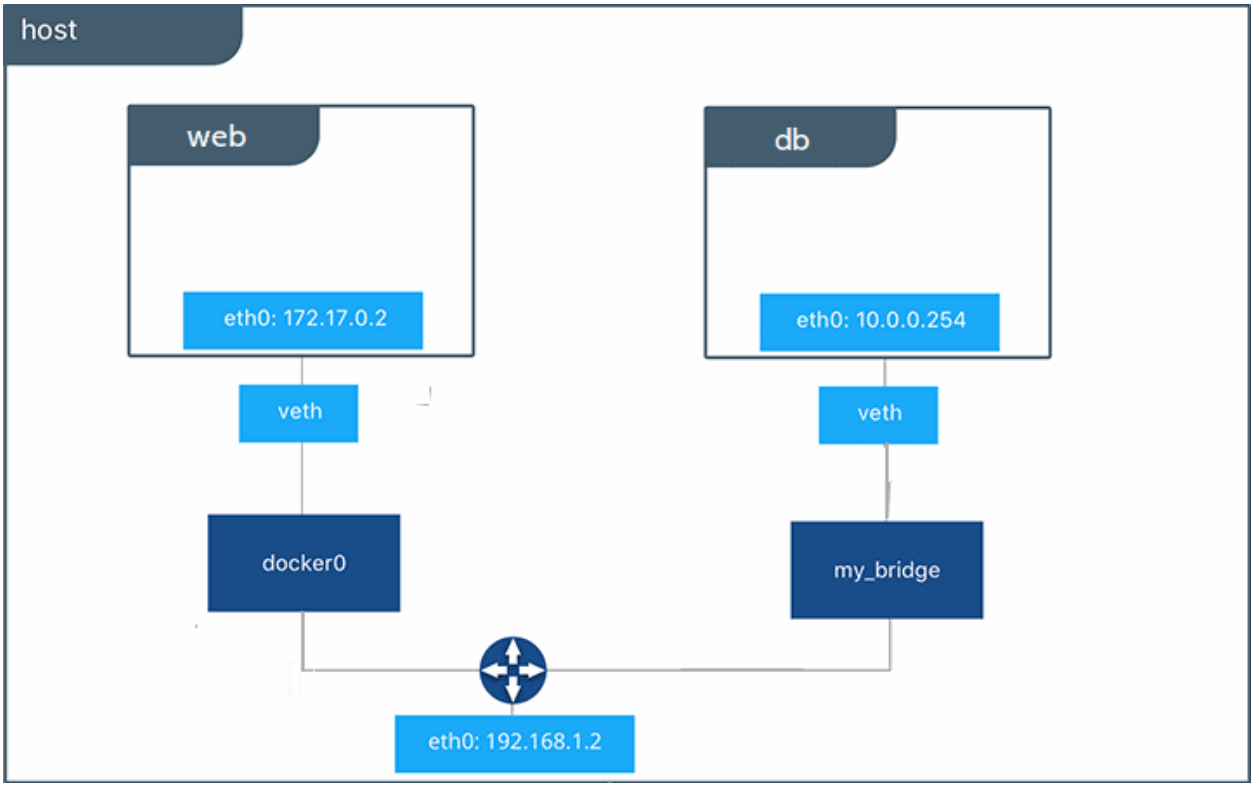
```
❯ ifconfig
        eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.27.13.110  netmask 255.255.240.0  broadcast 172.27.15.255


❯ docker network create --driver=bridge --subnet 172.27.100.0/24 --gateway 172.27.100.254 net1

❯ ifconfig
        br-4d6ca801d4ca: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.27.100.254  netmask 255.255.255.0  broadcast 172.27.100.255

❯ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
4d6ca801d4ca   net1      bridge    local
```

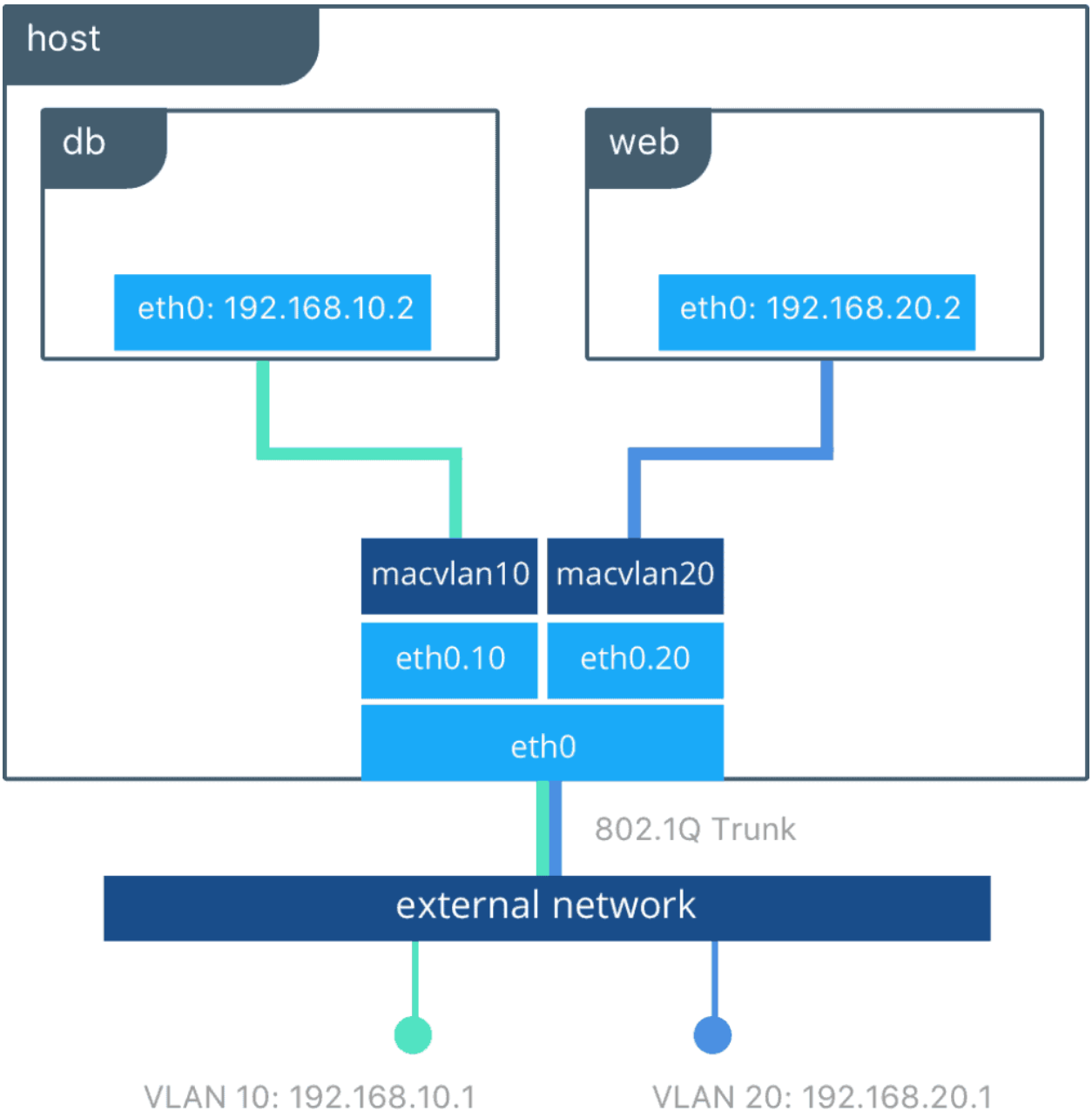
host



- 호스트에서 컨테이너의 네트워크 격리를 해제
- 호스트 에서는 하나의 프로세스로 컨테이너가 동작하는것과 다름없기 때문에 네트워크 정보를 공유할 수 있다.
- 컨테이너의 포트와 호스트 포트 충돌에 주의

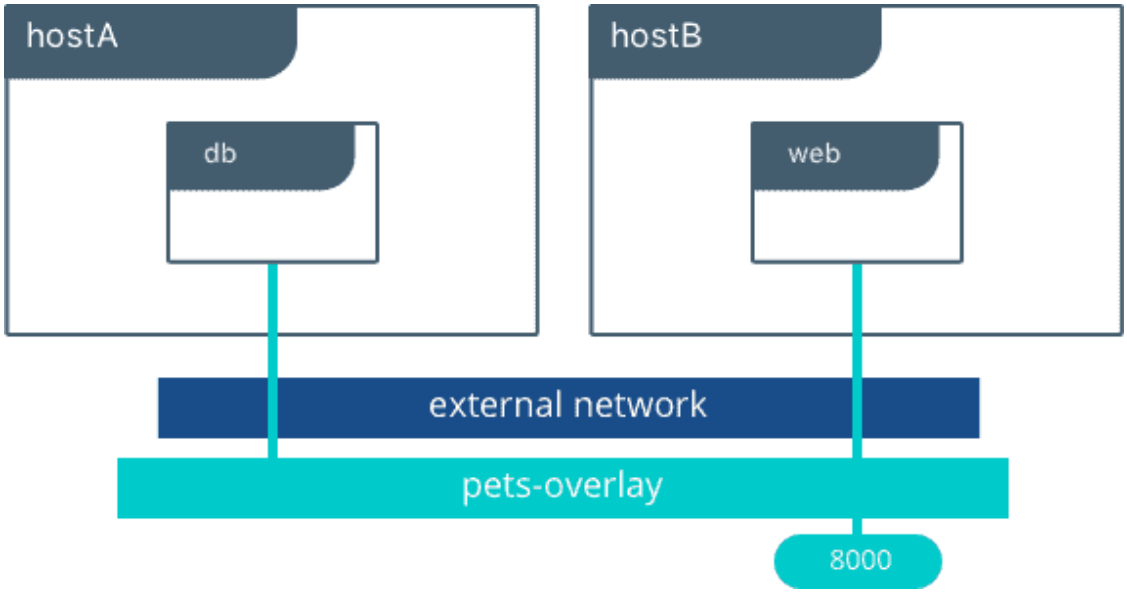
```
❯ docker run -d --name http_server --network host httpd:latest
❯ curl localhost
<html><body><h1>It works!</h1></body></html>
```

macvlan



- 컨테이너의 인터페이스에 MAC 주소를 컨테이너에 할당
- 네트워크 상에 물리적 디바이스로 나타나게 한다.
- dockerd가 트래픽을 컨테이너의 MAC 주소로 라우팅
- 컨테이너가 실제 네트워크에 직접 연결해야 할 때 사용

overlay



- 여러 호스트를 사용하여 Docker를 구동하는멀티 호스트 환경에서 사용 (Swarm)

container

- 다른 컨테이너의 네트워크 설정 공유
- host와 유사하지만 호스트와는 네트워크 설정을 공유하지 않음
- kubernetes의 Pod가 container 유형의 네트워크 사용
- pause 컨테이너 : 네트워크 할당 및 유지

null

- 컨테이너의 네트워크 기능을 사용하지 않음
- 다른 컨테이너 및 외부 네트워크 접근 불가
- 컨테이너에서 네트워킹을 비활성화하려는 경우

```
❯ docker run -it --rm --network none centos:latest ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
```

```
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
```

Contrainer 연결

동일 host 상에 배포된 container 사이는 Private IP 를 이용해 통신이 가능하다. 하지만 Container 의 IP 는 언제든 변할 수 있는 유동적인 성격을 띄고 있기 때문에 고정 IP로 통신하게 되면 container 가 재생성될때마다 IP주소를 변경해줘야 한다

- link

```
> docker run -d --name <CREATE_CONTAINER> --link <TARGET_CONTAINER> <CONTAINER>:<TEG>
```

  - `docker container link` 는 Docker 컨테이너 간에 네트워크 연결을 설정하는 명령어이다.
  - 이를 통해, 여러 개의 컨테이너가 서로 통신할 수 있도록 설정할 수 있다.
  - 보통 `docker container link` 를 사용하여, 하나의 컨테이너에서 다른 컨테이너에서 실행 중인 애플리케이션에 접근하고자 할 때 사용됩니다. 예를 들어, `web` 컨테이너에서 `db` 컨테이너에 접근하기 위해서는 다음과 같은 명령어를 실행할 수 있습니다.

    ```
    > docker container run -d --name db mongo
    > docker container run -d -p 80:80 --link db:db --name web nginx
    ```

  - 위의 명령어에서, `--link db:db` 는 `web` 컨테이너에서 `db` 컨테이너에 접근하기 위해 사용됩니다. 이후 `web` 컨테이너에서 `db` 컨테이너에 접근하기 위해 다음과 같은 코드를 사용할 수 있습니다.

    ```
    > docker container exec -it web bash
    root@web:/# ping db
    ```

- 하지만, `docker container link` 는 현재 더 이상 사용되지 않는 기능입니다. 대신, Docker 네트워크를 사용하여 컨테이너 간에 연결을 설정하는 것이 권장됩니다.
- `docker network create` 명령어를 사용하여 네트워크를 생성하고, `--network` 옵션을 사용하여 컨테이너를 생성하는 방법
- `docker network create` 를 사용하여 네트워크를 생성하면, Docker는 해당 네트워크에 대한 DNS 서버를 자동으로 생성합니다. 이를 통해, 컨테이너 이름을 사용하여 다른 컨테이너에 연결할 수 있습니다. 예를 들어, `web` 컨테이너와 `db` 컨테이너를 `my-net` 이라는 네트워크에 연결하고자 할 때 다음과 같은 명령어를 사용할 수 있습니다.

```
> docker network create my-net
> docker container run -d --network my-net --name db mongo
> docker container run -d -p 80:80 --network my-net --name web nginx
```

- 위의 명령어에서, `--network my-net` 옵션을 사용하여 컨테이너를 `my-net` 네트워크에 연결하고 있습니다. 이후 `web` 컨테이너에서 `db` 컨테이너에 접근하기 위해 다음과 같은 코드를 사용할 수 있습니다.

```
> docker container exec -it web bash
root@web:/# ping db
```

- 포트포워딩

```
> docker run -d --name <CONTAINER_NAME>
```