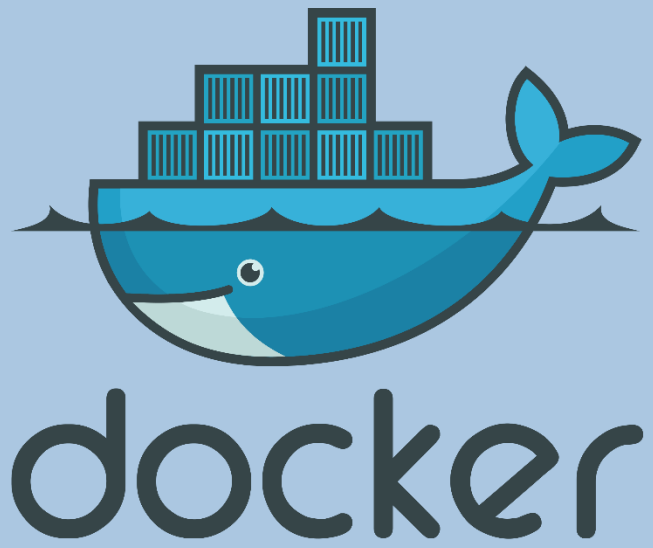


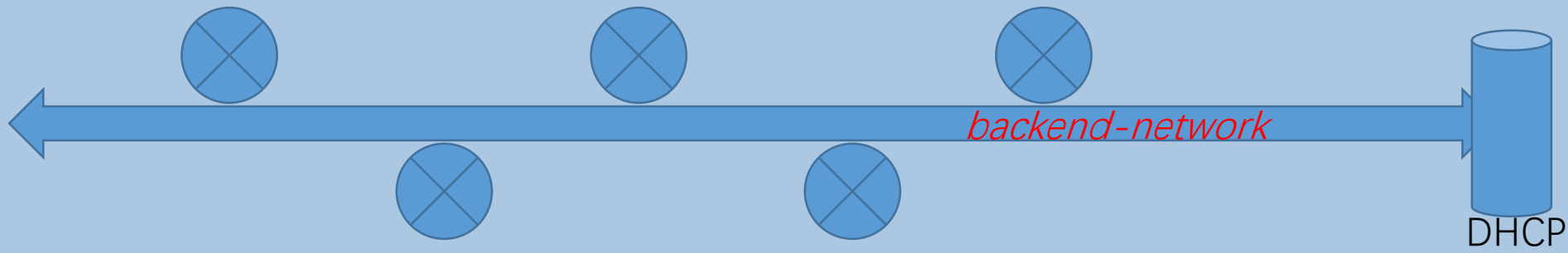
Docker Tutorials

Docker Networks

Step 1 - Create Network❶

The first step is to create a network using the CLI. This network will allow us to attach multiple containers which will be able to discover each other.

In this example, we're going to start by creating a *backend-network*. All containers attached to our backend will be on this network.



ShenHengheng

Step 1 - Create Network❷

Task1: Create Network

- To start with we create the network with our predefined name.
  - *docker network create backend-network*

Task2: Connect To Network

- When we launch new containers, we can use the *--net* attribute to assign which network they should be connected to.
  - *docker run -d --name=redis --net=*
  - *backend-network redis*

Step 2 - Network Communication❶

- Network Communication
  - links
  - docker network

- Unlike using links, *docker network* behave like traditional networks where nodes can be *attached/detached*.

Step 2 - Network Communication❷

- The first thing you'll notice is that Docker no longer assigns environment variables or updates the hosts file of containers. Explore using the following two commands and you'll notice it no longer mentions other containers.
  - *docker run --net=backend-network alpine env*
  - *docker run --net=backend-network alpine cat /etc/hosts*

```
$ docker run --net=backend-network alpine env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=7e92be91a957
HOME=/root
$ docker run --net=backend-network alpine cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.19.0.3      6013dc397bbb
```

ShenHengheng

Step 2 - Network Communication❸

- Instead, the way containers can communicate via an Embedded DNS Server in Docker. This DNS server is assigned to all containers via the IP 127.0.0.11 and set in the *resolv.conf* file.
  - *docker run --net=backend-network alpine cat /etc/resolv.conf*
  - ```
    $ docker run --net=backend-network alpine cat /etc/resolv.conf
    nameserver 127.0.0.11
    options ndots:0
    ```

- When containers attempt to access other containers via a well-known name, such as Redis, the DNS server will return the IP address of the correct Container. In this case, the fully qualified name of Redis will be *redis.backend-network*.
  - *docker run --net=backend-network alpine ping -c1 redis*
    ```

Step 3 - Connect Two Containers

- Docker supports multiple networks and containers being attached to more than one network at a time.
  - *docker network create frontend-network*
- When using the *connect* command it is possible to attach existing containers to the network.
  - *docker network connect frontend-network redis*
- When we launch the web server, *given it's attached to the same network* it will be able to communicate with our Redis instance.
  - *docker run -d -p 3000:3000 --net=frontend-network katacoda/redis-node-docker-example*

Step 4 - Create Aliases❶

- Links are still supported when using docker network and provide a way to define an Alias to the container name. This will give the container an extra DNS entry name and way to be discovered. When using *--link* the embedded DNS will guarantee that localised lookup result only on that container where the --link is used.

- The other approach is to provide an alias when connecting a container to a network.

Step 4 - Create Aliases❷

Connect Container with Alias

- The following command will connect our Redis instance to the frontend-network with the alias of db.
  - *docker network create frontend-network2*
  - *docker network connect --alias db frontend-network2 redis*

- When containers attempt to access a service via the name db, they will be given the IP address of our Redis container.
  - *docker run --net=frontend-network2 alpine ping -c1 db*

Step 5 - Disconnect Containers

- The following command will list all the networks on our host.
  - *docker network ls*
- We can then explore the network to see which containers are attached and their IP addresses.
  - *docker network inspect frontend-network*
- The following command disconnects the redis container from the *frontend-network*.
  - *docker network disconnect frontend-network redis*

Reference

- https://www.katacoda.com/courses/docker/networking-intro