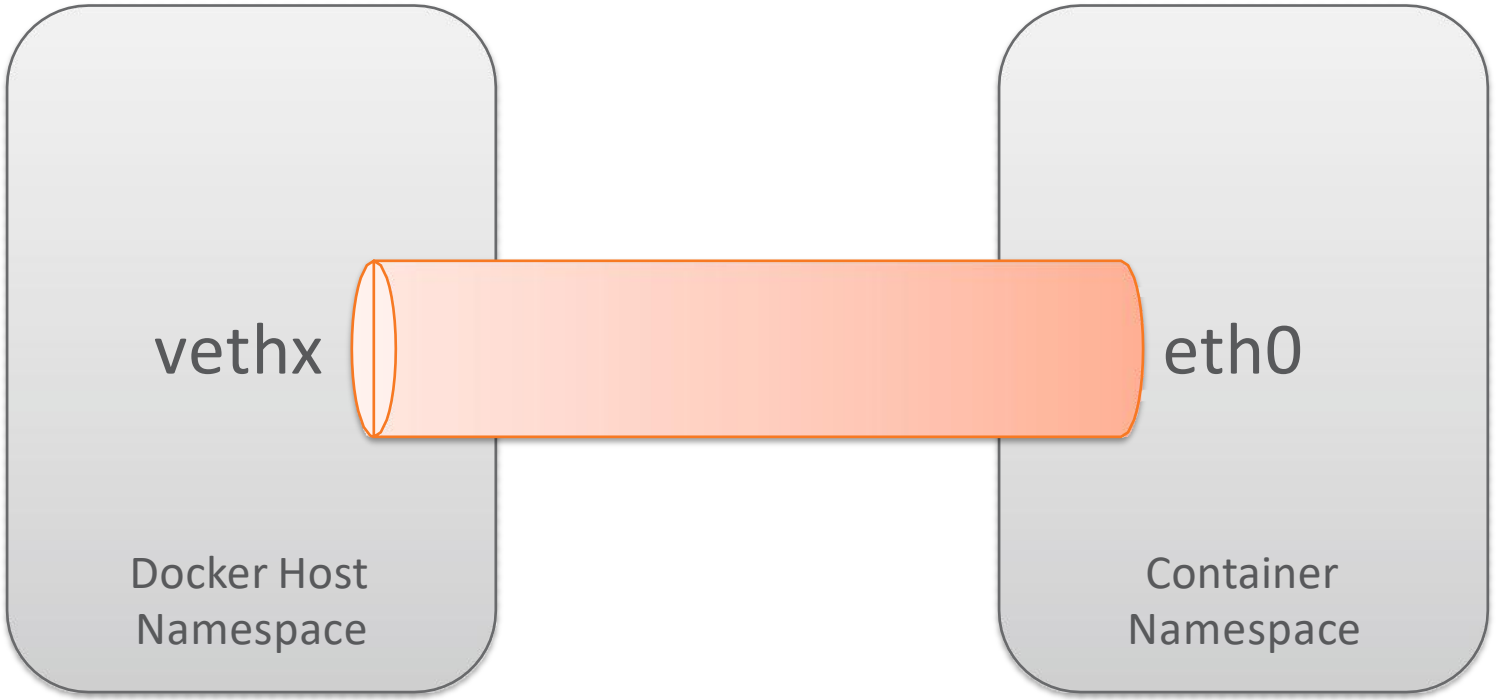
Docker Networking

The “docker0” Virtual Bridge

Virtual Ethernet Interfaces

Exposing Ports

Linking Containers



Default Networks: docker0

Default Networks: docker0

When you install Docker, it creates three networks automatically.

You can see this bridge as part of a host's network stack by using the ifconfig command on the host.

$ ifconfig

The bridge network represents the docker0 network present in all Docker installations. Unless you specify otherwise with the docker run --network=<NETWORK> option, the Docker daemon connects containers to this network by default.

You can list these networks using the docker network ls command:

```
$ docker network ls

NETWORK ID          NAME                DRIVER
7fca4eb8c647        bridge              bridge
9f904ee27bf5        none                null
cf03ee007fb4        host                host
```

These three networks are built into Docker. When you run a container, you can use the --network flag to specify which networks your container should connect to.

The default bridge network

The default bridge network is present on all Docker hosts. If you do not specify a different network, new containers are automatically connected to the default bridge network.

$ docker network inspect bridge

Docker creates three networks automatically on install: bridge, none, and host.

Specify which network a container should use with the --net flag.

If you create a new network my_network (more on this later), you can connect your container (my_container) with:

$ docker run my_container --net=my_network

Run the following two commands to start two busybox containers, which are each connected to the default bridge network.

$ docker run -itd --name=container1 busybox

$ docker run -itd --name=container2 busybox

Inspect the bridge network again after starting two containers. Both of the busybox containers are connected to the network. Make note of their IP addresses, which will be different on your host machine than in the example below.

$ docker network inspect bridge

Containers connected to the default bridge network can communicate with each other by IP address.

You can attach to a running container to see how the network looks from inside the container.
$ docker attach container1
root@0cb243cd1293:/# ifconfig
root@0cb243cd1293:/# ping 8.8.8.8
root@0cb243cd1293:/# traceroute 8.8.8.8

From inside the container, use the ping command to test the network connection to the IP address of the other container.
root@0cb243cd1293:/# ping –w 3 172.17.0.3

Use the cat command to view the /etc/hosts file on the container. This shows the hostnames and IP addresses the container recognizes.

```
root@0cb243cd1293:/# cat /etc/hosts
```

To detach from the container1 container and leave it running, use the keyboard sequence CTRL-p CTRL-q.

Creating a bridge network

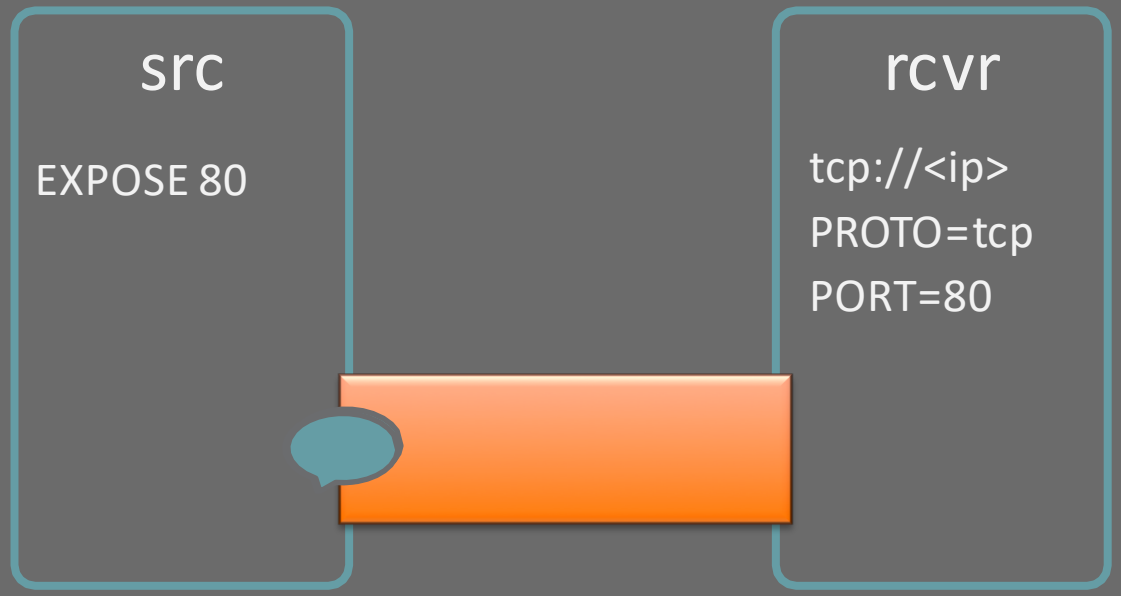
Creating a bridge network

Bridge networks (similar to the default docker0 network) offer the easiest solution to creating your own Docker network.

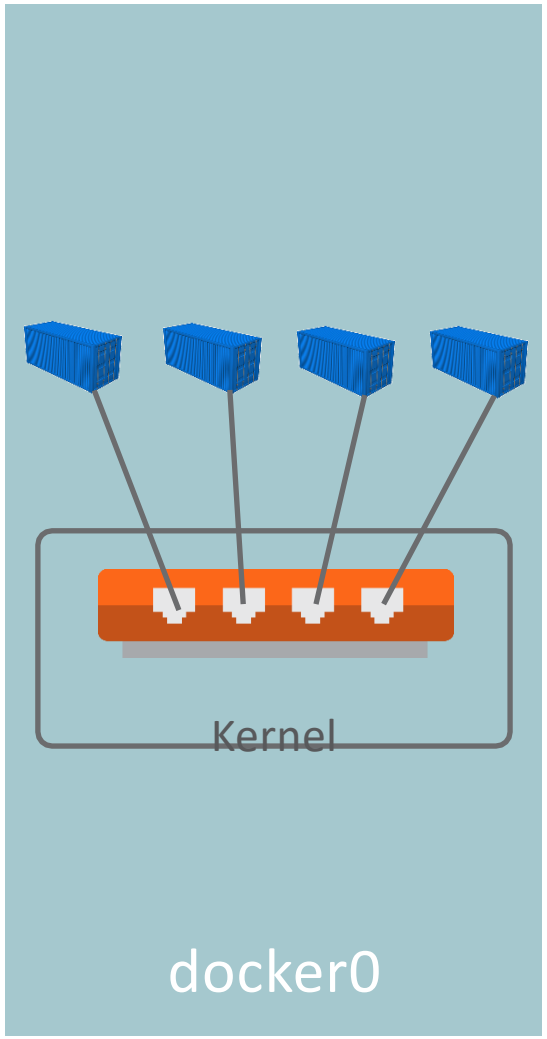
Follow along below to create your own my_isolated_bridge_network and run your Postgres container my_psql_db on that network:

$ docker network create --driver bridge my_isolated_bridge_network

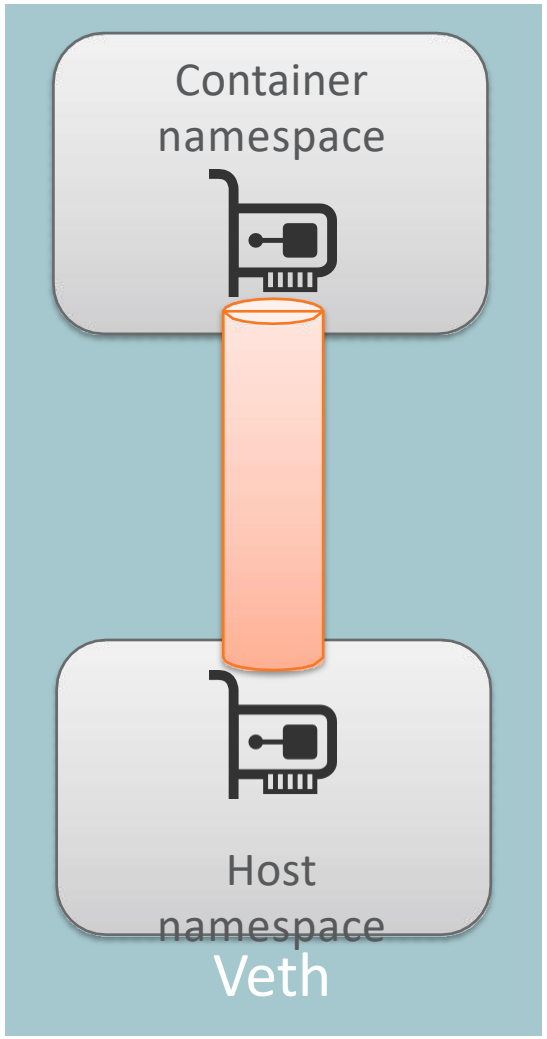
```
$ docker network inspect my_isolated_bridge_network

$ docker network ls

$ docker run --net=my_isolated_bridge_network --name=my_psql_db
postgres

$ docker network inspect my_isolated_brige_network
```



Module Recap



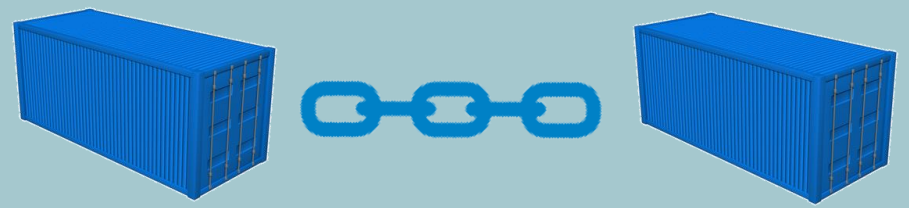
docker0



Veth

```
# docker port web1
80/tcp —> 0.0.0.0:5001
```

Exposing Ports



Linking Containers

Reference

https://goo.gl/fSoiCo
https://goo.gl/88tcpn