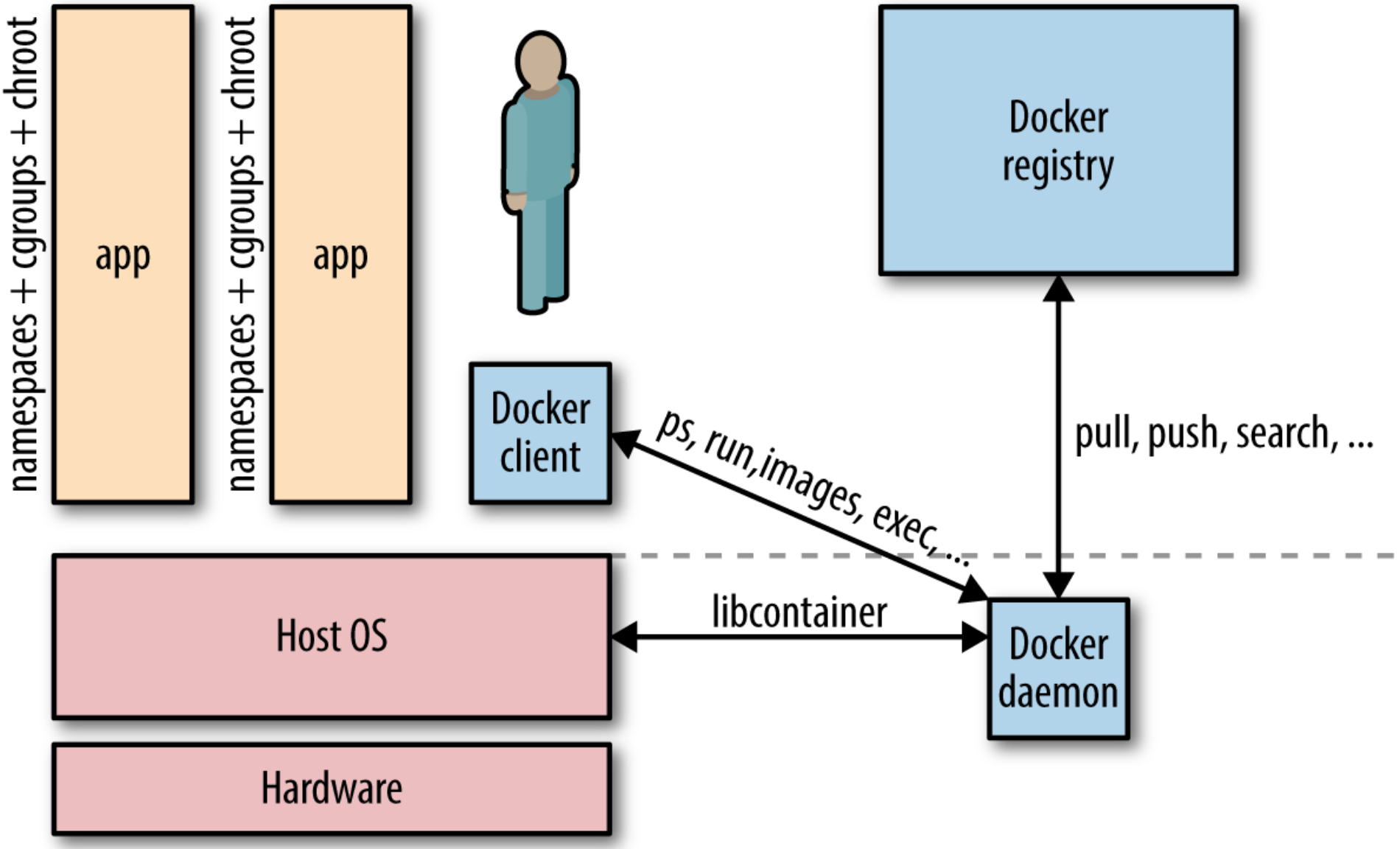
Module 5

Docker Networking

Simplified Docker architecture (single host)



Need of Networking

The relationship between a host and containers is 1:*N*. This means that one host typically has several containers running on it

For most *single-host deployments*, data exchange via a shared volume versus data exchange through networking (HTTP-based or otherwise).

Docker data volume is simple to use,
introduces tight coupling, meaning that it will be harder to turn a single-host deployment into a multihost deployment.
Naturally, the upside of shared volumes is speed.

In *multihost deployments*, you need to consider two aspects:

how are containers communicating within a host and
how does the communication paths look between different hosts.

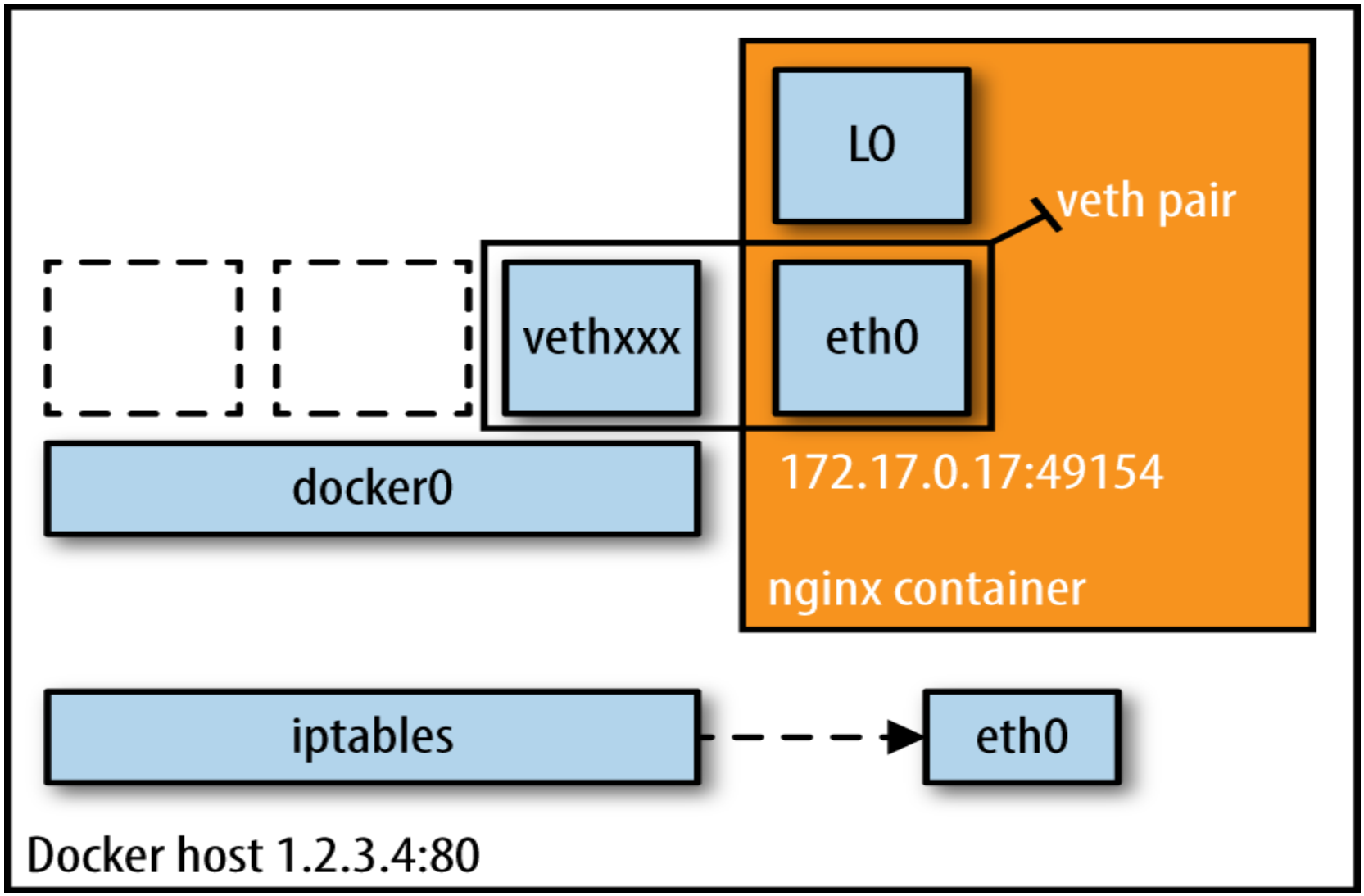
Both performance considerations and security aspects will likely influence your design decisions.

Multihost deployments usually become necessary either when the capacity of a single host is insufficient or when one wants to employ distributed systems such as Apache Spark, HDFS, or Cassandra.

Options for Networking

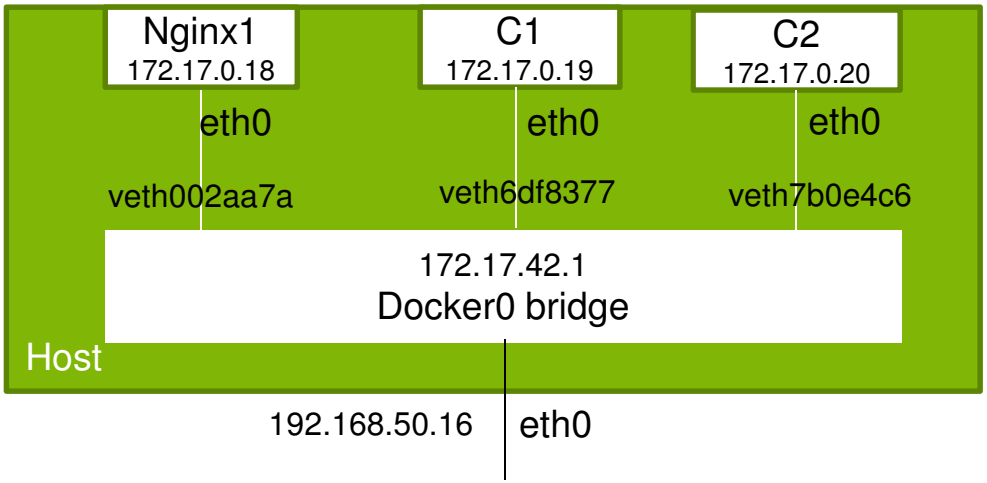
1. Bridge
2. Host
3. Container
4. SDN (Open vSwitch)

Bridge Mode


LO
veth pair
vethxxx
eth0
docker0
172.17.0.17:49154
nginx container
iptables
eth0
Docker host 1.2.3.4:80

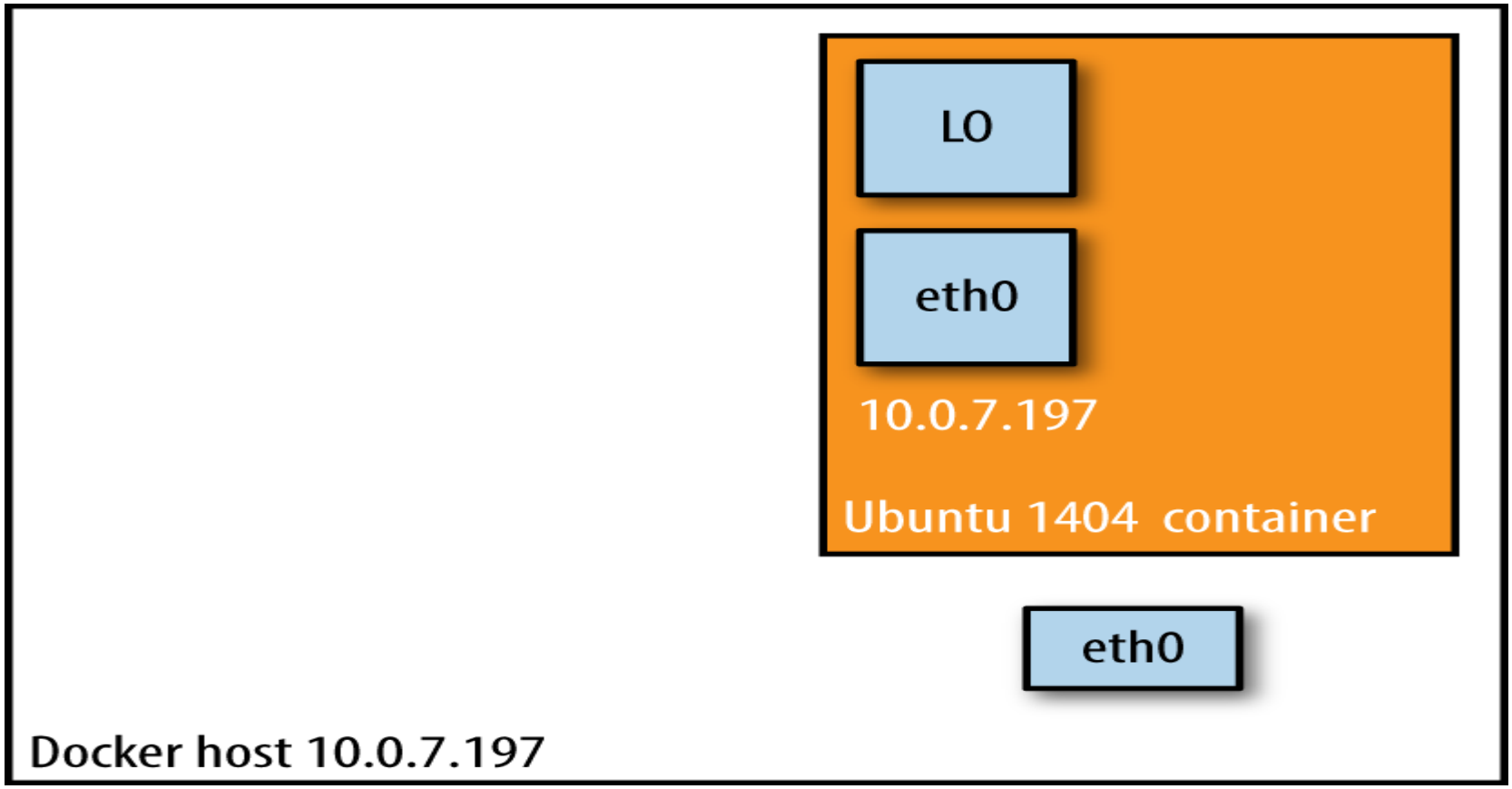
Bridge

- Default network automatically created when no additional options "--net" or "-P" are specified



- Each container is addressed by a static IP address assigned by Docker
- Similar to what we have as default with KVM or VirtualBox
- Host can reach container with IP on the bridge
- But, outside traffic cannot reach the container

Host Mode



This mode effectively disables network isolation of a Docker container. Because the container shares the networking namespace of the host, it is directly exposed to the public network; consequently, you need to carry out the coordination via port mapping.

Host Mode

Give full access of the host network to the container using --net=host option

```
#   docker run --net=host
    --name=c3 -i -d -t base /bin/bash
```

Check network within container using ifconfig command through exec

```
#   docker exec c3 ifconfig eth0
    eth0   Link encap:Ethernet
           HWaddr 52:54:00:0d:3c:9f
           inet addr:192.168.50.16
           Bcast:192.168.50.255
```

Host can talk to container using lo (localhost) interface

Containers can listen on privileged ports (i.e., port numbers < 1024) of host

Other Modes

Container Mode

Docker reuses the networking namespace of another container. In general, this mode is useful when you want to provide custom network stacks. Indeed, this mode is also what Kubernetes networking leverages.

run -it --net=container:CID   ubuntu:14.04 ip addr

Container Mode

Give full access to network of a container XX to the new container YY using --net=container:XX option

```
#  docker run --net=container:nginx1
   --name=c4 -i –d –t base /bin/bash
```

Check network within container using ifconfig command through exec

```
#  docker exec c4 ifconfig eth0
    eth0   Link encap:Ethernet
           HWaddr 02:42:ac:11:00:12
           inet addr:172.17.0.18
           Bcast:0.0.0.0
```

Container XX can talk to container YY using lo (localhost) interface

Open vSwitch (OVS)

- Similar to Linux bridge, but different technology
  - Today, this is not the default with Docker
  - Allows programming with OVSDB and OpenFlow protocols
- Why? OpenvSwitch has many useful features!
  - VxLAN, GRE, VLAN based encapsulation and L2 forwarding
  - Encapsulation allows containers to pick any MAC/IP they want
  - Also possible to do L3 routing, ARP proxy etc, load-balancing
  - Access control, traffic rate limiting and prioritization
  - 10G/s or more packet processing throughput possible
  - 1) kernel, or 2) userspace, with optionally DPDK acceleration

User defined networks

Can create your own user-defined networks that better isolate containers. Docker provides some default network drivers for creating these networks. You can create a new bridge network or overlay network.

You can create multiple networks. You can add containers to more than one network. Containers can only communicate within networks but not across networks.

A container attached to two networks can communicate with member containers in either network

User defined networks

The easiest user-defined network to create is a bridge network. This network is similar to the historical, default docker0 network. There are some added features and some old features that aren't available.

$ docker network create –driver bridge #network-name#

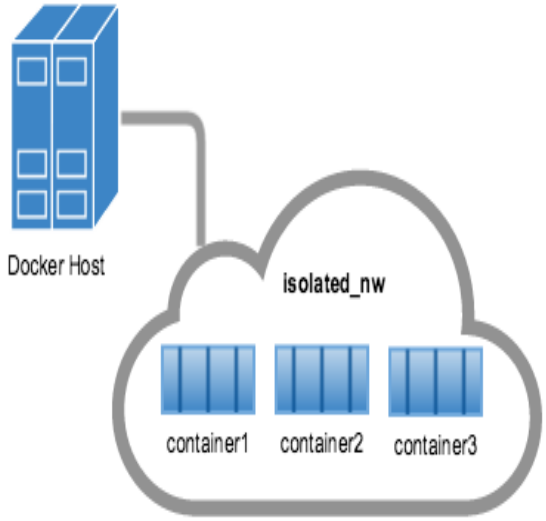
$ docker network inspect #network-name#

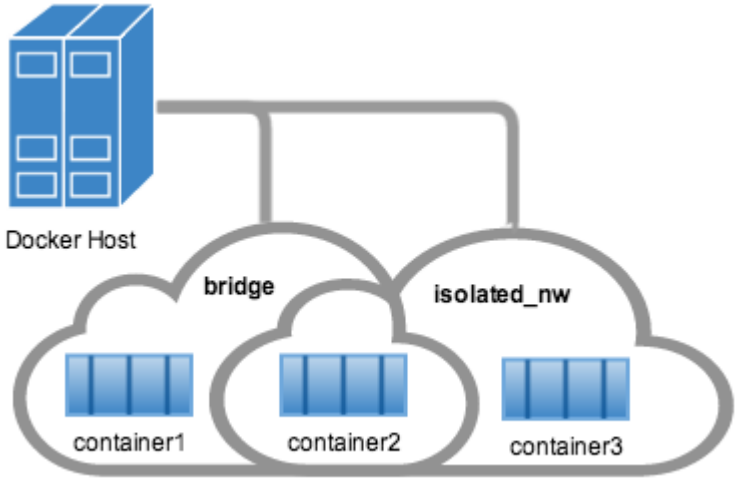
$ docker network ls

$ docker run --network=i#network-name# –itd --name=container3 busybox

User defined networks

The containers you launch into this network must reside on the same Docker host. Each container in the network can immediately communicate with other containers in the network. Though, the network itself isolates the containers from external networks.





Working with networking commands

$ docker network create

$ docker network connect

$ docker network ls

$ docker network rm

$ docker network disconnect

$ docker network inspect