

Docker - Networking



Multi-Container Apps

- The power of containerization is realized when you decompose an app into microservices, where each runs in its own container – multi-container.
- Container engines provide networking support that allows **complex network namespace setups**.

Docker Networking

- By default Docker comes with the following network setup:

```
$ docker network ls
```

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

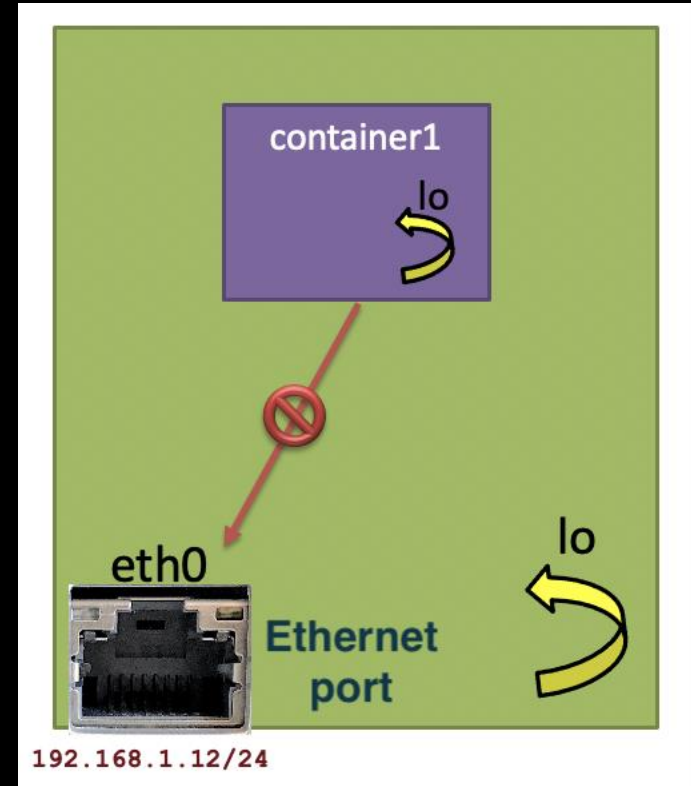
- Network Driver (Types):
 1. **Bridge (default)** – its container are attached to a “bridge” (virtual bridge interface).
 2. **None** - Its **container has no networking capabilities.**
 3. **Host** - Its **containers are attached to the host’s network.**

The “none” Network

```
$ docker run \
```

```
--network=none ...
```

- **Container is in a new network namespace.**
- **Local loopback is enabled.**
- **No other network interfaces are supplied.**



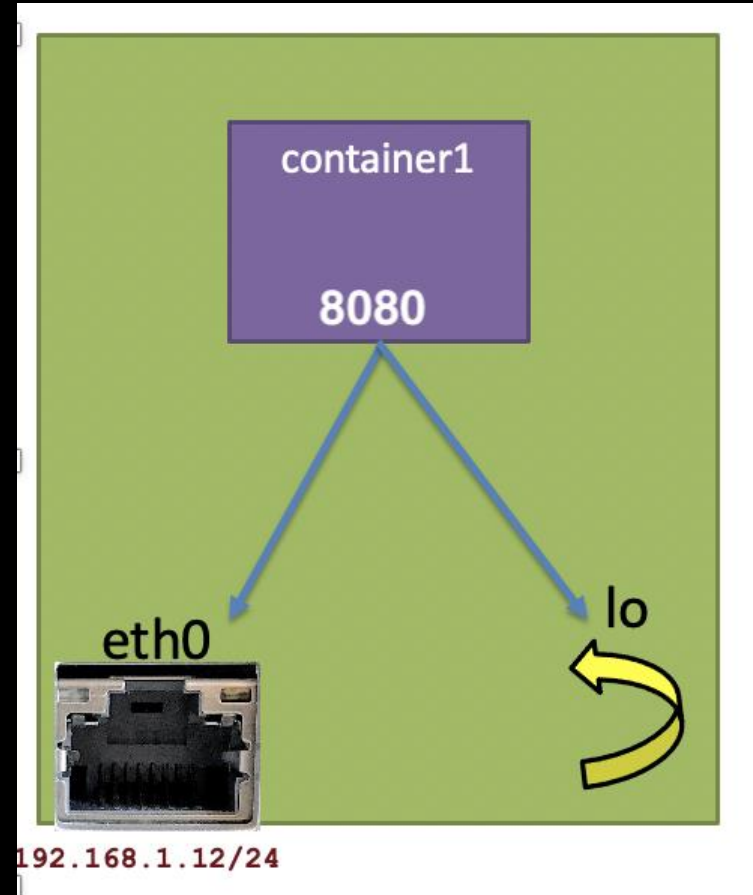
The “host” Network

- Used to add a custom service to the host machine.

\$ docker run \

--network=host ...

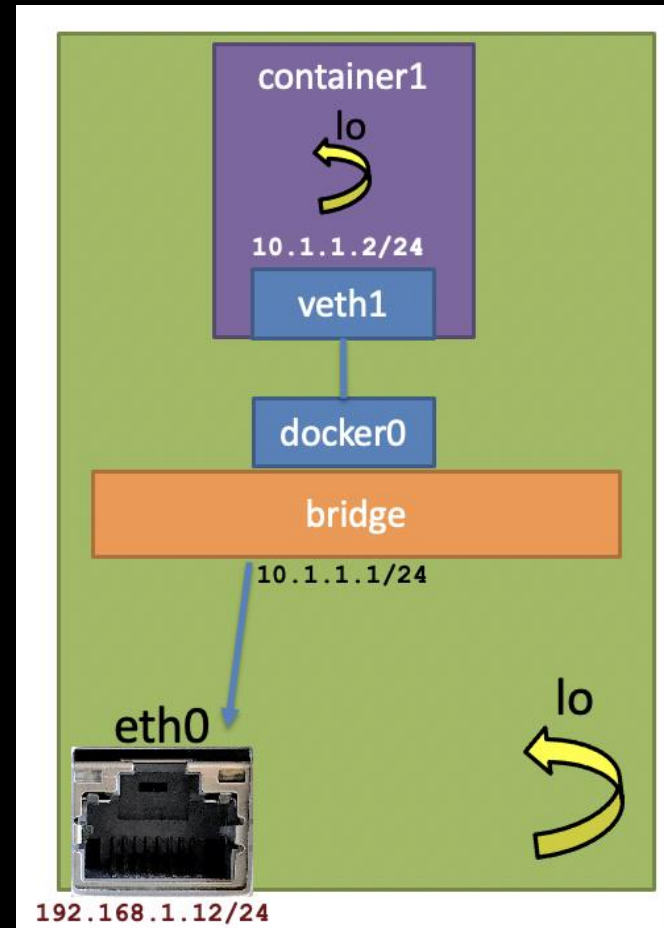
- **Container is in the host's network namespace.**
- **Processes inside container have same access to network resources as those outside.**



A “bridge” Network

```
$ docker run -it --rm  
  --network=bridge ...
```

- **Default network type.**
- **Network assigned a subnet IP address range.**
- **Virtual bridge (docker0) acts as a gateway.**
- **Each container:**
 1. **has a virtual ethernet i/f.**
 2. **Assigned an IP in the range (private)**



Custom Docker Networks

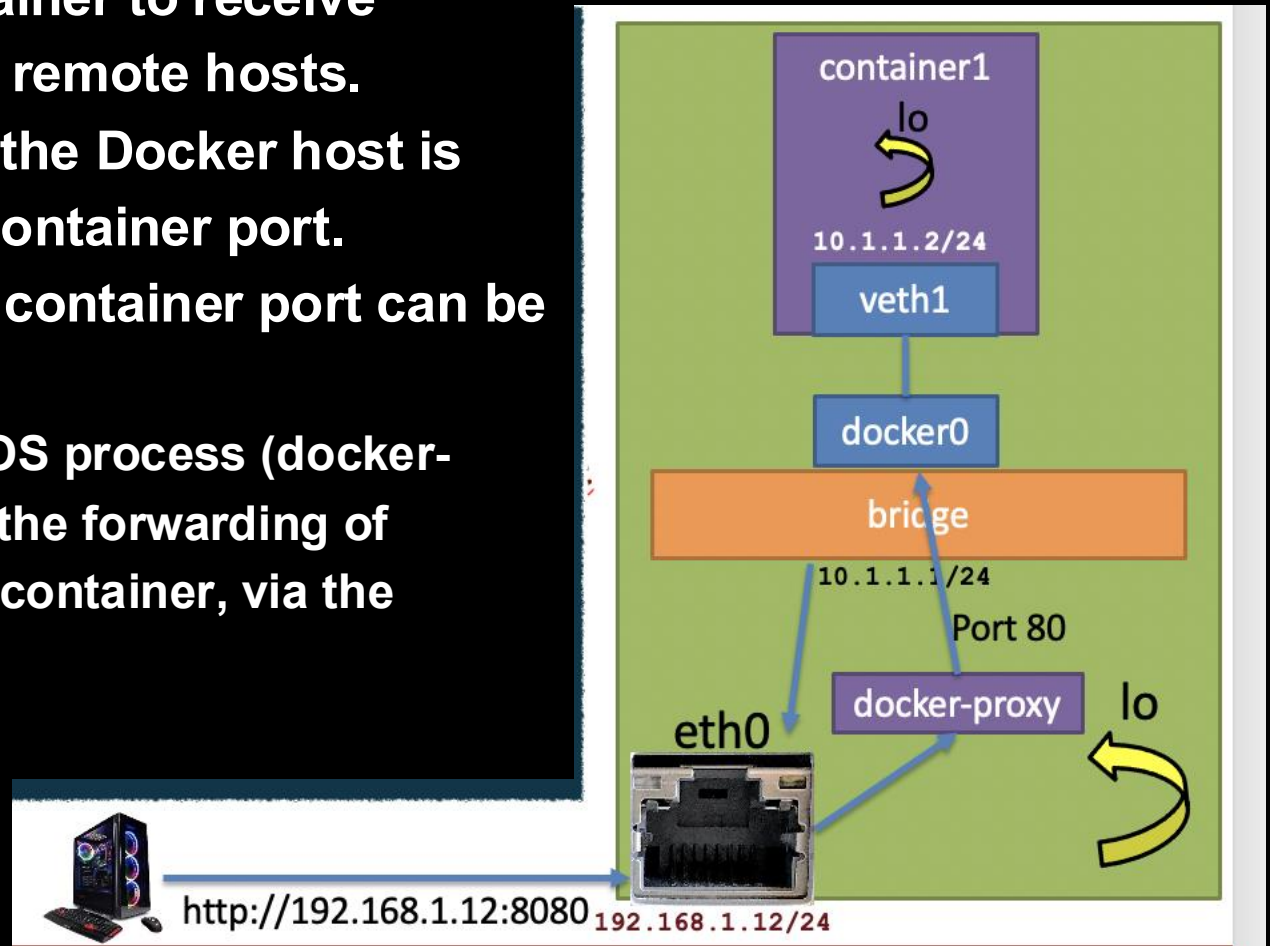
- There can be multiple bridge networks.
 - Each assigned a different IP subnet range

```
mybridge1
diarmuidoconnor docker network create mybridge1 bash at
4018c1071c50409865d00dc2e47f44698933c993bcc2a14cb06e825f4274a05b
diarmuidoconnor docker network create mybridge2
6763c4d030c210722e125396756ff10162eb16e752dd089f81312841564a1820
diarmuidoconnor docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
60964a6b734c    bridge    bridge       local
8537f98578cd    host      host         local
4018c1071c50    mybridge1 bridge       local
6763c4d030c2    mybridge2 bridge       local
a72c3511f663    none      null         local
diarmuidoconnor docker network remove mybridge1    in ba
mybridge1
diarmuidoconnor
```

- Inspect a network:
 - \$ docker network inspect <net_name>
 - Returns lots of JSON-formatted information.

Port Forwarding.

- Allows a container to receive requests from remote hosts.
- A TCP port of the Docker host is mapped to a container port.
- Host port and container port can be different.
- A special host OS process (docker-proxy) handles the forwarding of requests to the container, via the bridge.





Volumes

Volumes (General).

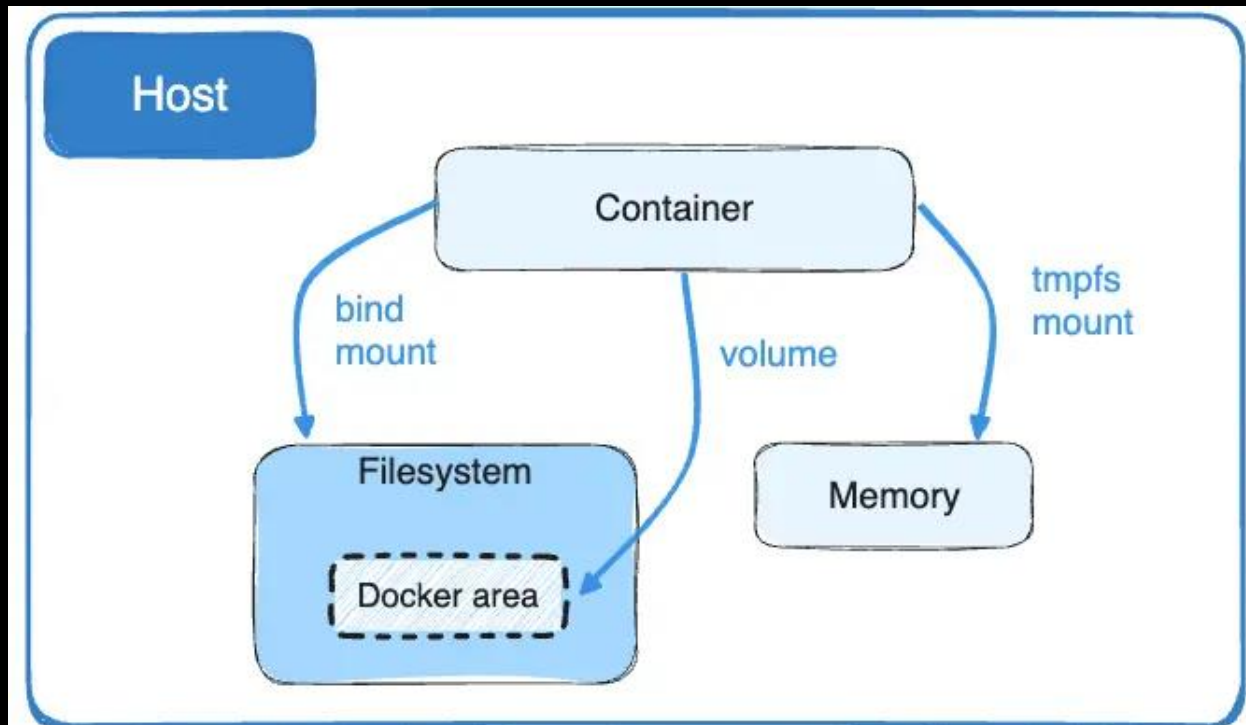
- A Volumes maps a part of the host's filesystems on to the container's filesystems, e.g.

```
$ docker run ... -v host-path:container-path ...
```

- It allows data to persist longer than the lifecycle of the container, e.g. databases, configuration.
 - When a container is removed, its local filesystem is lost.

Volumes (General).

- Two types:
 1. Bind mount (Original).
 2. Named volume.



Bind mounts

- A file or directory on the host is mounted on to a container.
 - Host folder is created on demand if necessary.
- Advantages:
 - Convenient when debugging, due to their accessibility
- Disadvantages:
 - They rely on the host filesystem having a specific structure.
 - Can't manage them directly with Docker CLI commands.
 - Break the isolation principle.

Creating Bind mounts

- Two formats:
 - (1) `$ docker run -d -v source-path:target-path:ro \`
 `--name mycontainer image-name`
 - Creates the source (host) file/folder if it does not exist.
 - Target-path – container path
 - Options, e.g. ro (read only).
 - (2) `$ docker run -d --name mycontainer \`
 `--mount type=bind,source=source-path,`
 `target=target-path,readonly \`
 `image-name`
 - **Throws an error if source path does not exist.**

Named Volumes.

- Volumes live inside Docker; they're not visible on the host filesystem.
- Advantages:
 1. Can be managed with Docker CLI commands.
 2. Higher performance than bind mounts on Docker Desktop.
 3. Volume drivers let you store volumes on remote hosts or cloud providers, and support encryption.

Managing Named Volumes.

- Created and managed using the CLI.

```
$ docker volume create my-vol
```

- Anonymous volume name - generated by Docker Deamon.

```
$ docker volume ls.      list all volumes.
```

```
$ docker volume inspect my-vol
```

```
$ docker volume rm my-vol      remove/delete
```

```
$ docker volume prune          delete all unused volumes (not  
linked to a container).
```

- **Start a container that uses a named volume:**

```
$ docker run -d --name devtest \
```

```
    -v my-vol:target-path image-name
```

- **Can use the `--mount` form as well.**