Docker basic networking

Florent Gluck - Florent.Gluck@hesge.ch

April 13, 2022

Main Docker network drivers

- None
- Host
- Bridge
- (Overlay)

None network driver

- The `none` network driver ensure no network interface is available to the container (except for the local `loopback` interface)
- Example:

```
docker run -it --rm --network none ubuntu:20.4
```

Host network driver

- **Remove network isolation** between container and host
- All network interfaces from the host are available in the container
- Example:

```
docker run -it --rm --network host ubuntu:20.4
```

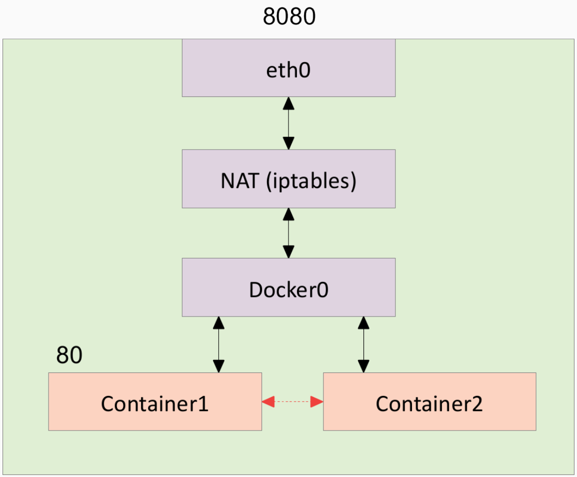
Bridge network driver

- Allow containers connected to the same bridge network to communicate, while providing isolation from containers which are not connected to it
- One can create user-defined custom bridge networks

Default bridge network

- When Docker daemon is started, a default **bridge** network is created automatically
  - Named `bridge` and exposed via the `docker0` interface
- Newly-started containers connect to the **bridge** network unless otherwise specified
- The default **bridge** network is legacy and is not recommended for production use
- Instead, it's recommended to create user-defined custom bridge networks

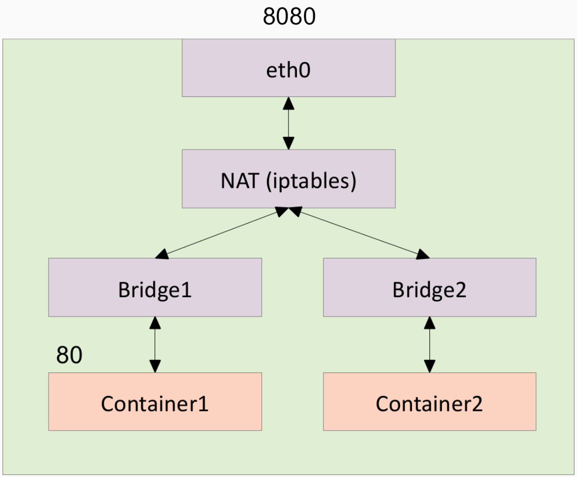
Default Docker bridge



User-defined bridge network

- Containers connected to the same user-defined bridge network effectively expose all ports to each other
- For a port to be accessible to containers or non-Docker hosts on different networks, it must be published with `-p`

User defined bridge



User-defined bridge network: usage

- Create `net1` and `net2` user-defined bridge networks:

```
for i in net1 net2; do docker network create $i; done
```

- Create container `web` and connect it to `net1` network; publish port 80 in the container to port 8080 on the host:

```
docker create --name web -h web --network net1 -p
    8080:80 nginx
```

- Connect `web` container to `net2` network:

```
docker network connect net2 web
```

- Any other container connected to `net1` or `net2` networks has access to all ports on `web`, and vice versa

Network commands

```
Usage:  docker network COMMAND
Manage networks
Commands:
  connect      Connect a container to a network
  create       Create a network
  disconnect   Disconnect a container from a network
  inspect      Display detailed information on one or
               more networks
  ls           List networks
  prune        Remove all unused networks
  rm           Remove one or more networks
```

Container hostname

- By default, a container's hostname is randomly generated
- The container's hostname **is not** set to the container's name
- Use `docker run -h <hostname>` to specify the container's hostname
- **Advice**: set the hostname to match the container's name, e.g.

```
docker run -it --rm --network theforce --name luke
     -h luke ubuntu:20.4
```

User-defined bridge network vs default bridge (1/2)

- User-defined bridges provide better flexibility and interoperability between containerized applications
- User-defined bridges provide **name resolution between containers** conntected to the same bridge
  - Containers on the default bridge network can only access each other by IP addresses

User-defined bridge network vs default bridge (2/2)

- Containers can be attached/detached from user-defined networks **on the fly**
  - To remove a container from the default `bridge` network, it needs to be stopped and recreated with different options
- Each user-defined network creates a configurable bridge
  - Configuring the default `bridge` network happens outside of Docker itself, and requires a restart of Docker

How to list network interfaces?

- `ip a` command, requires `iproute2` package (Ubuntu/Debian)
- `ifconfig` command, requires `net-tools` package (Ubuntu/Debian)
- Inspect the `/proc/net/dev` file (for instance with `cat`)

Resources

- Docker official documentation
  https://docs.docker.com/network/ https://docs.docker.com/network/bridge/