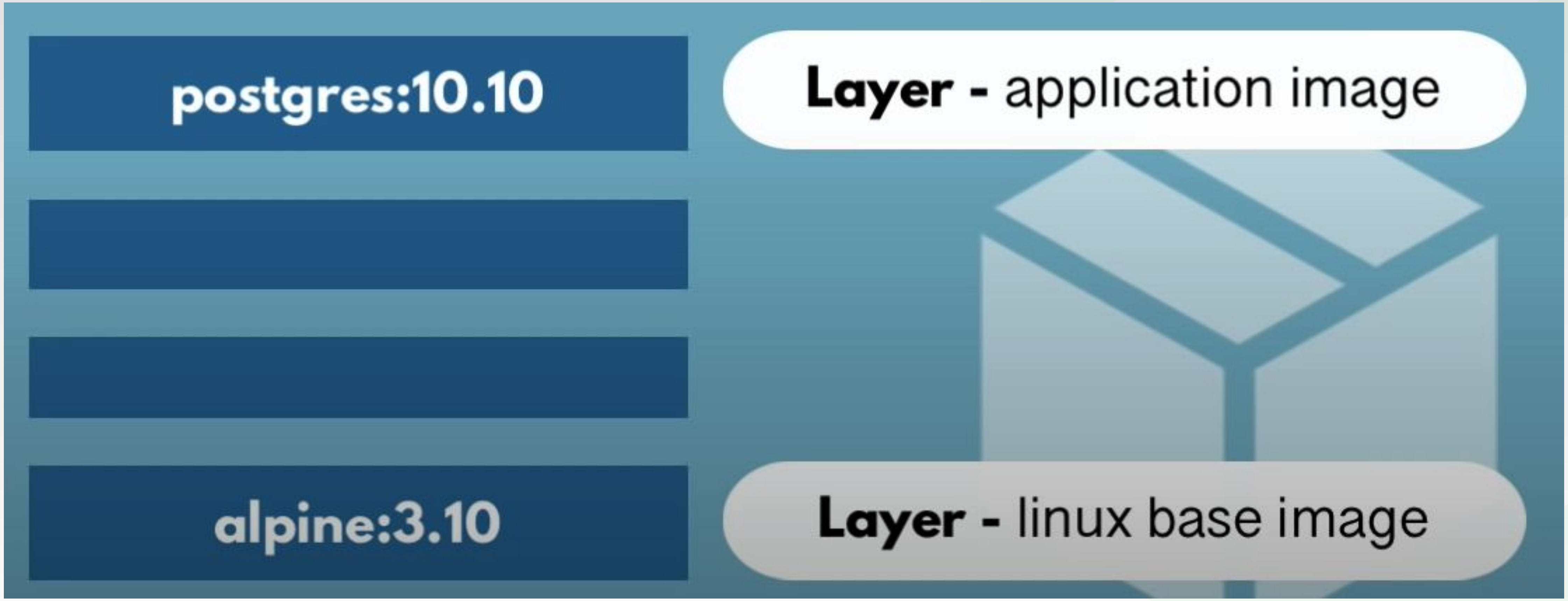
BATCH	:	B107 AWS-DevOps
LESSON	:	**Docker**
DATE	:	13.04.2023
SUBJECT	:	**Networking**

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ

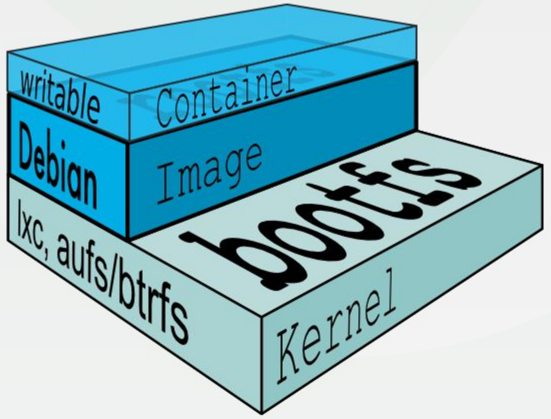
Review

Docker Image

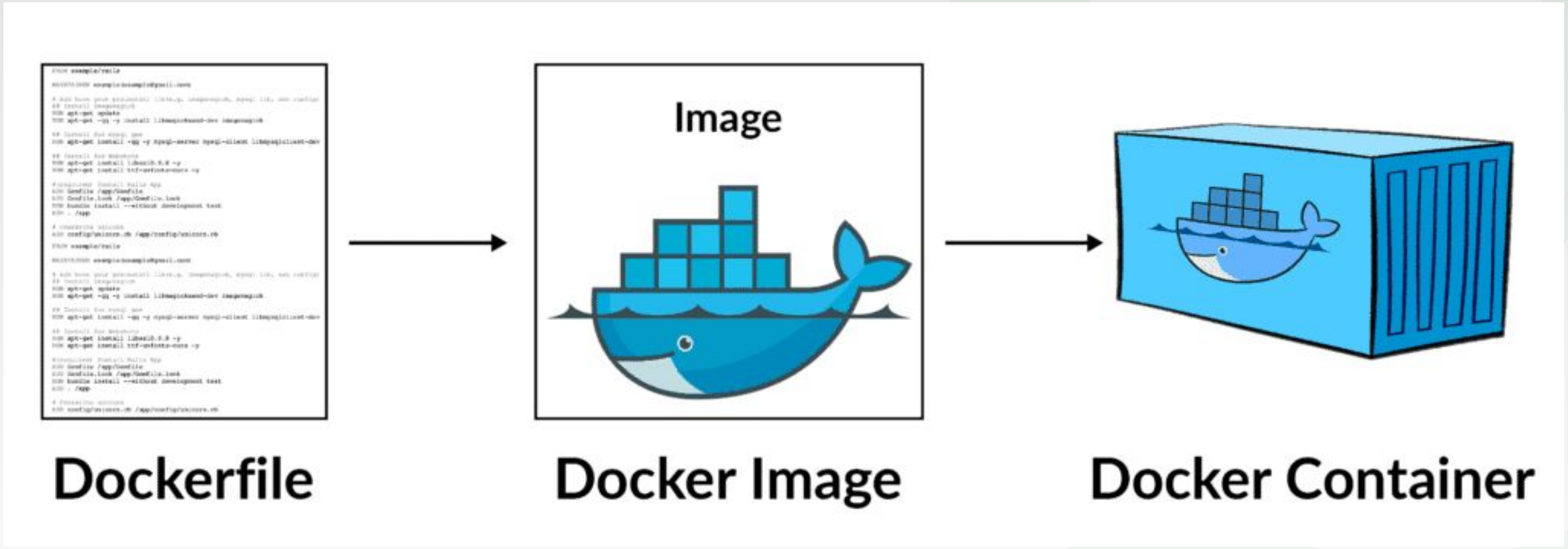


Docker Image

- An image is a collection of files and some metadata
- Images are comprised of multiple layers referencing another image
- Each image contains source code or software that you want to run
- Every image starts from a base image
- Layers are immutable or read only

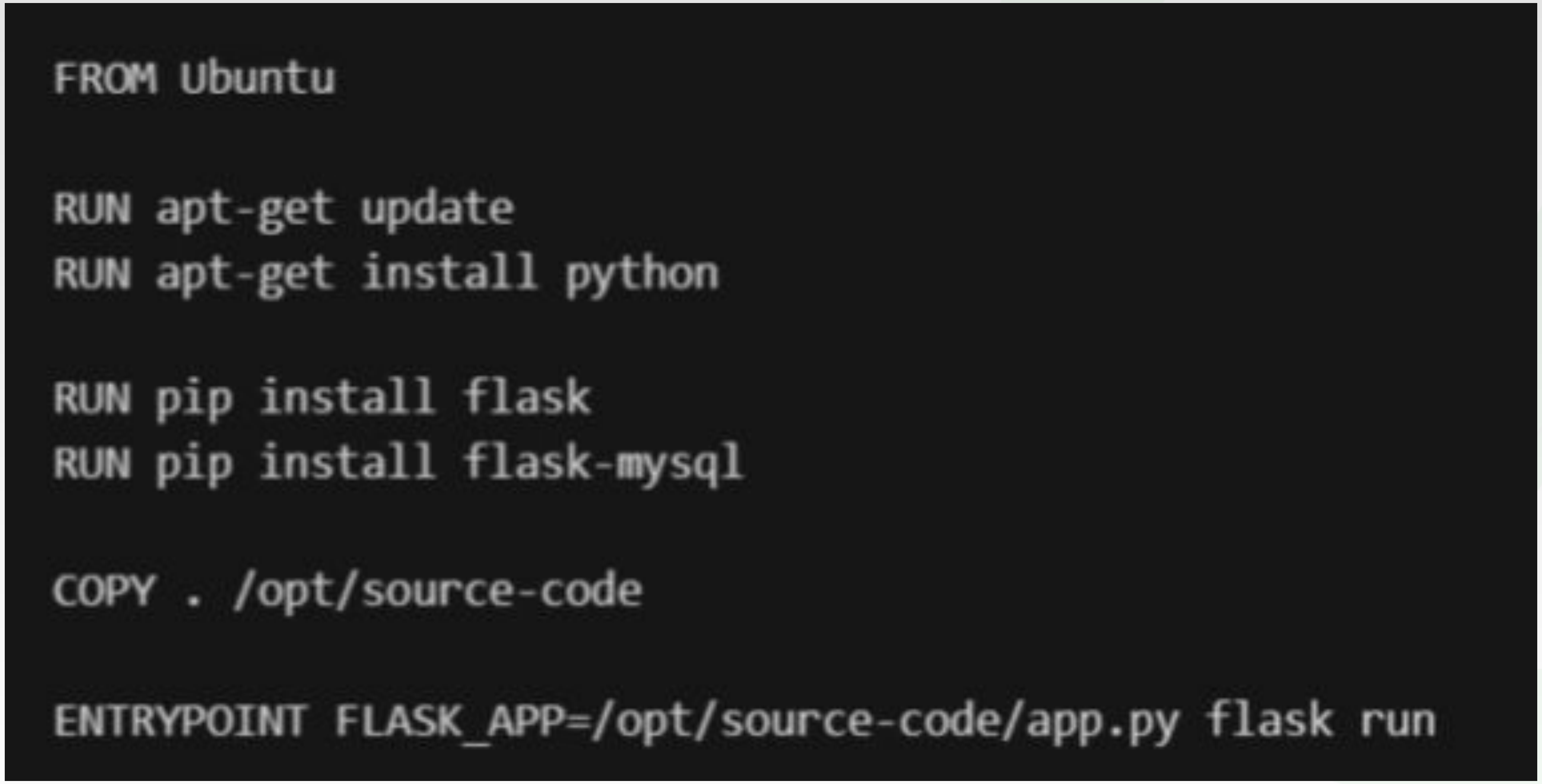


Dockerfile



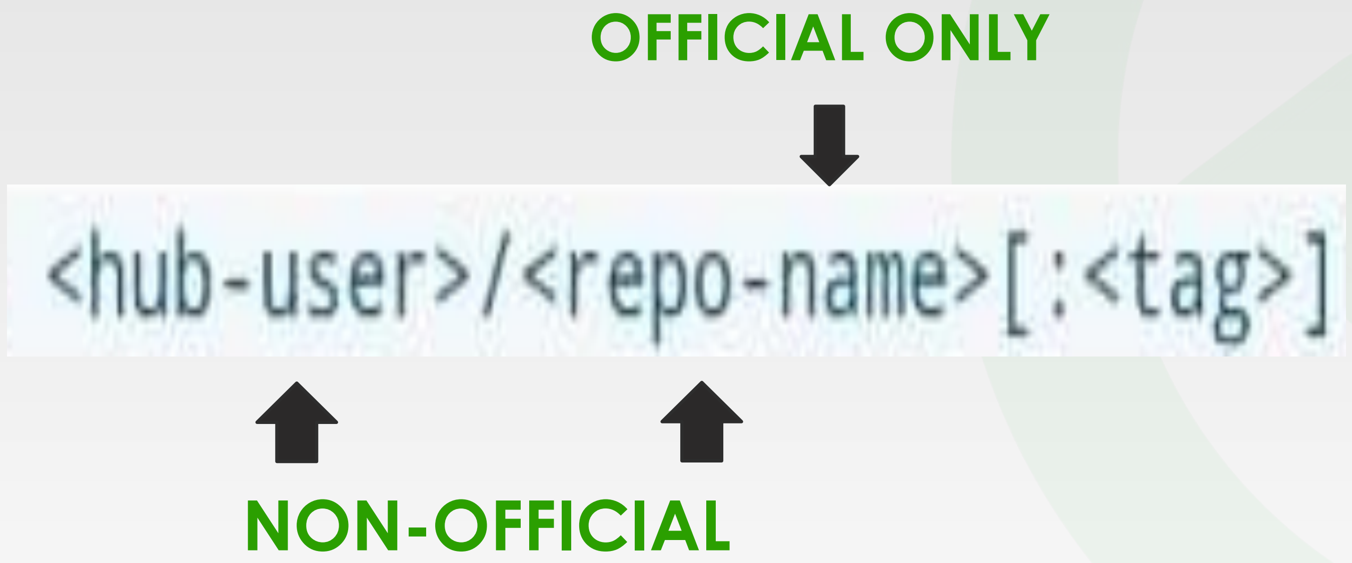
```
FROM example/rails

MAINTAINER example <example@gmail.com>

# [illegible]
## Install imagemagick
RUN apt-get update
RUN apt-get -qq -y install libmagickwand-dev imagemagick

## Install for mysql gem
RUN apt-get install -qq -y mysql-server mysql-client libmysqlclient-dev

## Install for [illegible]
RUN apt-get install [illegible] -y
RUN apt-get install ttf-unfonts-core -y

# [illegible] Install Rails App
ADD Gemfile /app/Gemfile
ADD Gemfile.lock /app/Gemfile.lock
RUN bundle install --without development test
ADD . /app

# [illegible]
ADD config/unicorn.rb /app/config/unicorn.rb
FROM example/rails

MAINTAINER example <example@gmail.com>

# [illegible]
## Install imagemagick
RUN apt-get update
RUN apt-get -qq -y install libmagickwand-dev imagemagick

## Install for mysql gem
RUN apt-get install -qq -y mysql-server mysql-client libmysqlclient-dev

## Install for [illegible]
RUN apt-get install [illegible] -y
RUN apt-get install ttf-unfonts-core -y

# [illegible] Install Rails App
ADD Gemfile /app/Gemfile
ADD Gemfile.lock /app/Gemfile.lock
RUN bundle install --without development test
ADD . /app

# [illegible]
ADD config/unicorn.rb /app/config/unicorn.rb
```

Dockerfile



```
FROM Ubuntu

RUN apt-get update
RUN apt-get install python

RUN pip install flask
RUN pip install flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Docker Image Naming Convention



```
<hub-user>/<repo-name>[:<tag>]
```

Docker Image Creation Commands

To build image

- docker build -t myimage:tag .

To build another version of the image

- docker commit modifiedContainer newimage

Docker Networking

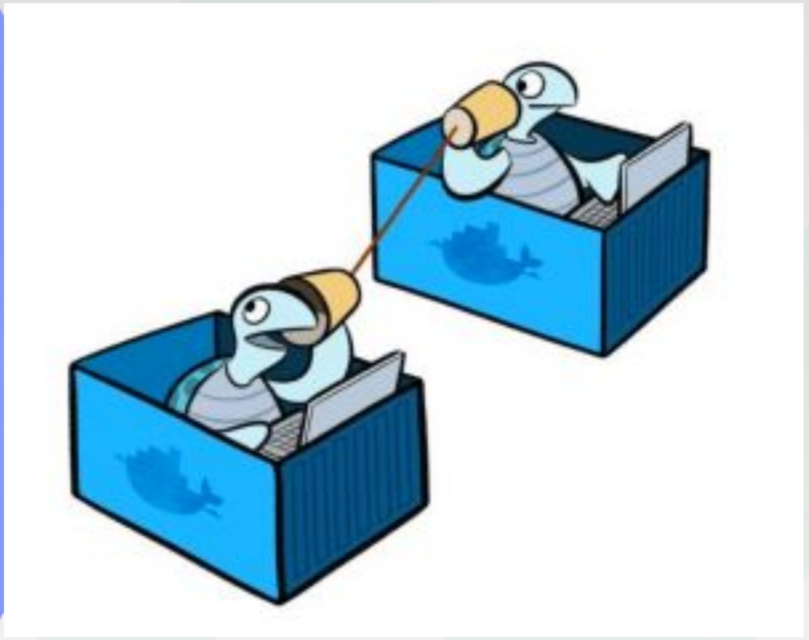
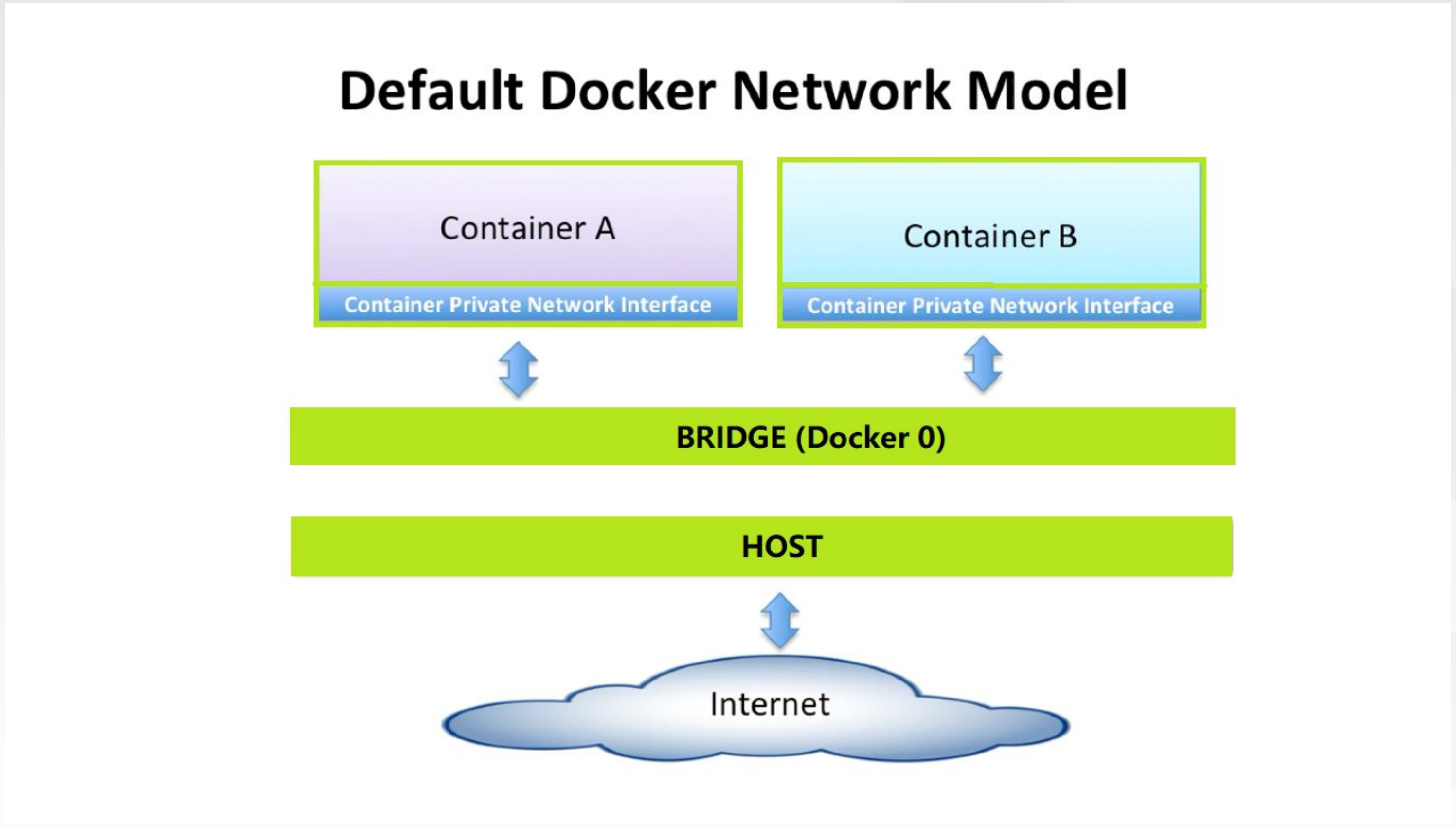


Table of Contents

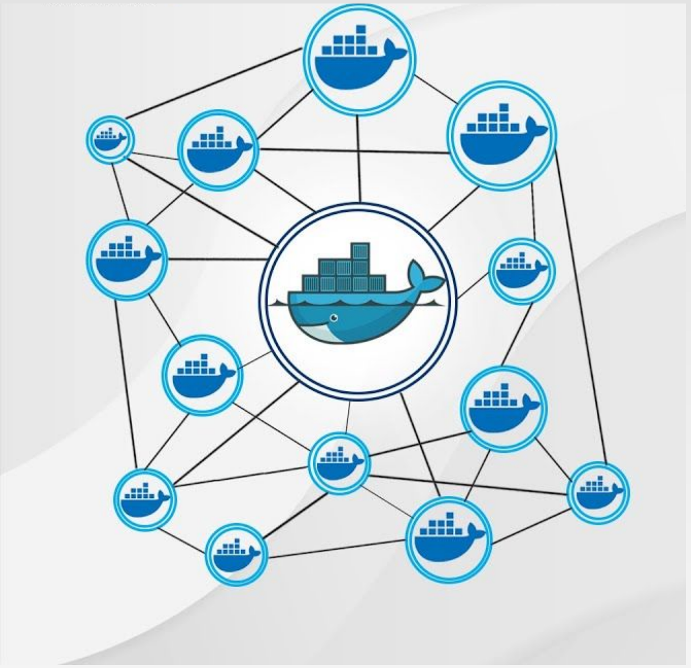
Networking Overview

Networking Overview

A **network** is two or more computer systems linked together by some form of the transmission medium.


Default Docker Network Model
Container A
Container Private Network Interface
Container B
Container Private Network Interface
BRIDGE (Docker 0)
HOST
Internet

Networking Overview



- One of the reasons Docker containers and services are so powerful is that you can connect them together, or connect them to non-Docker workloads.

- Whether your Docker hosts run linux, Windows, or a mix of the two, you can use Docker to manage them in a platform-agnostic way.

Network Drivers

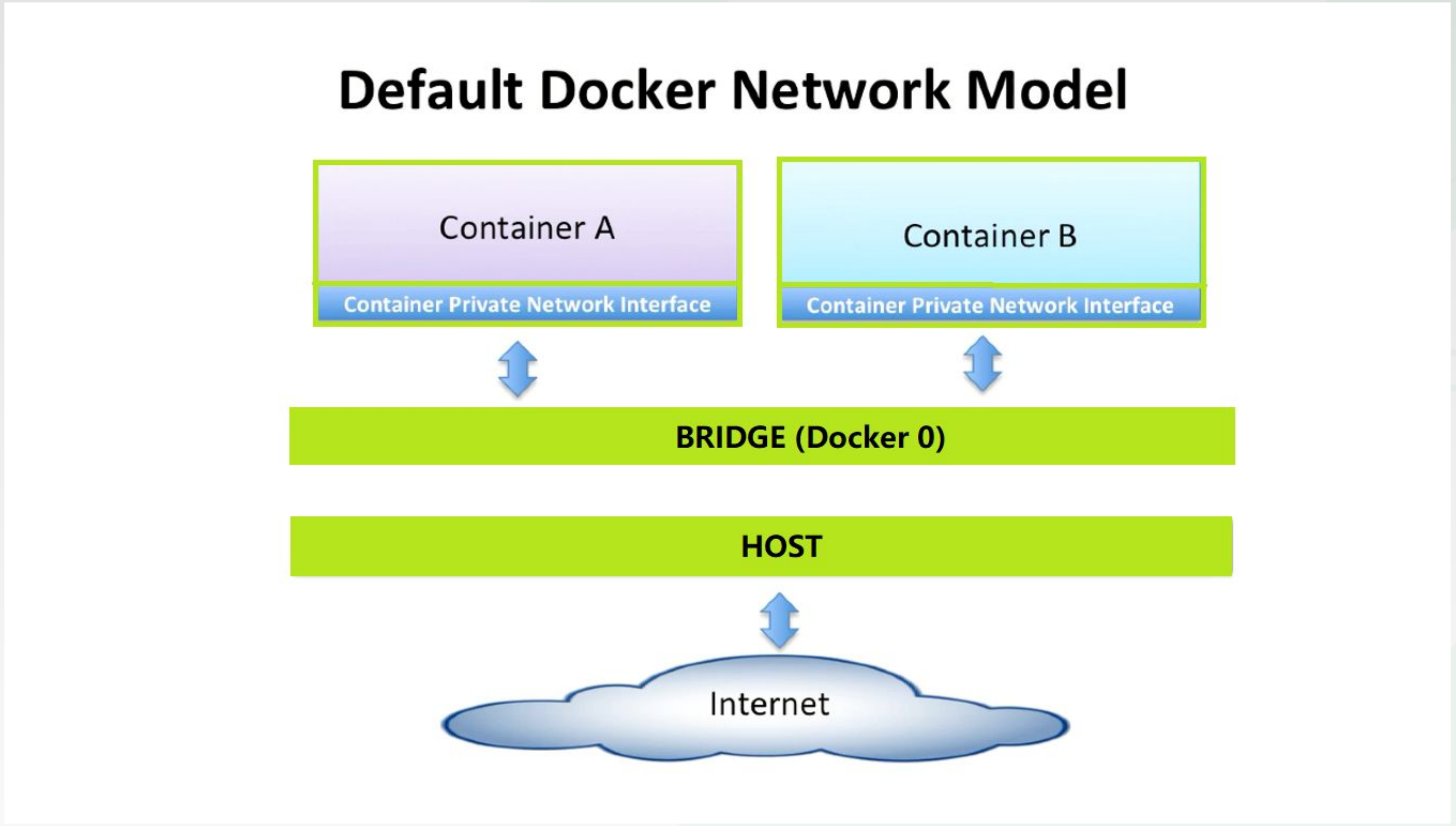
Network Drivers

As default, docker has three network drivers.

- Bridge
- Host
- None

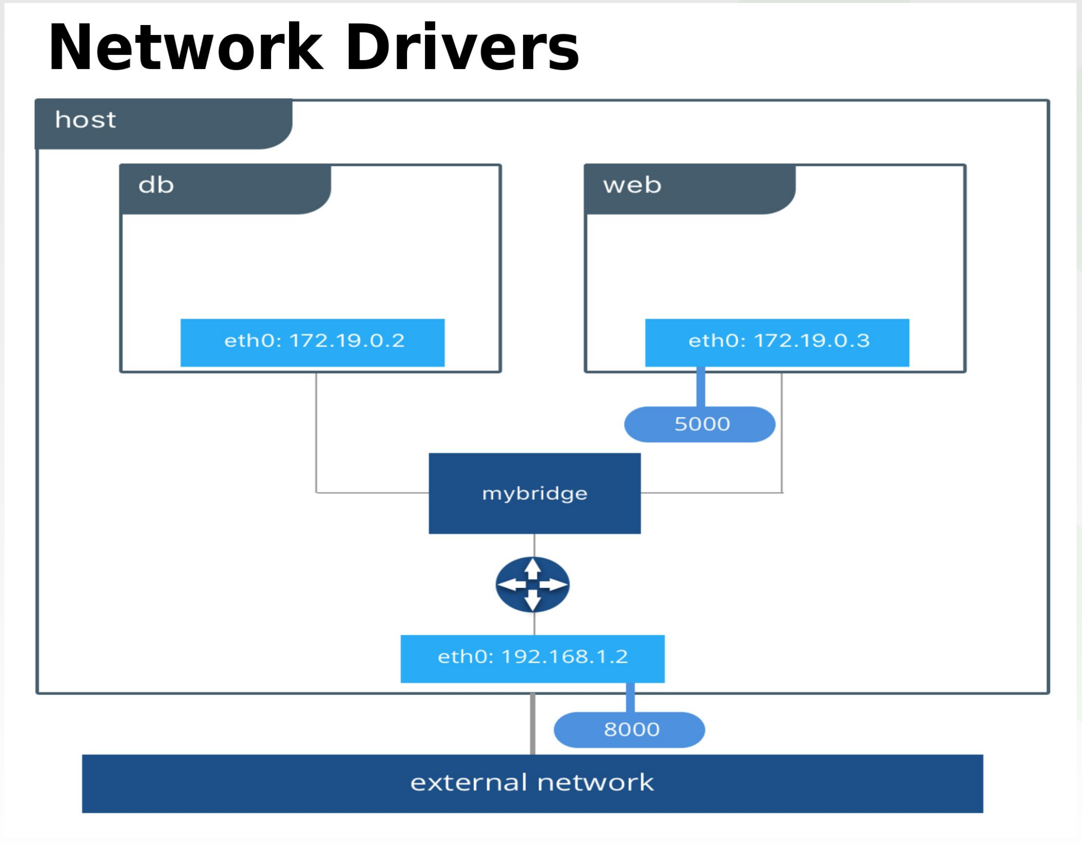
Network Drivers

- **Bridge** is the private default network driver. If we don't specify a driver, this is the type of network we are creating.


Default Docker Network Model
Container A
Container Private Network Interface
Container B
Container Private Network Interface
BRIDGE (Docker 0)
HOST
Internet

Network Drivers

- When we create containers, it will automatically attach to the bridge driver.



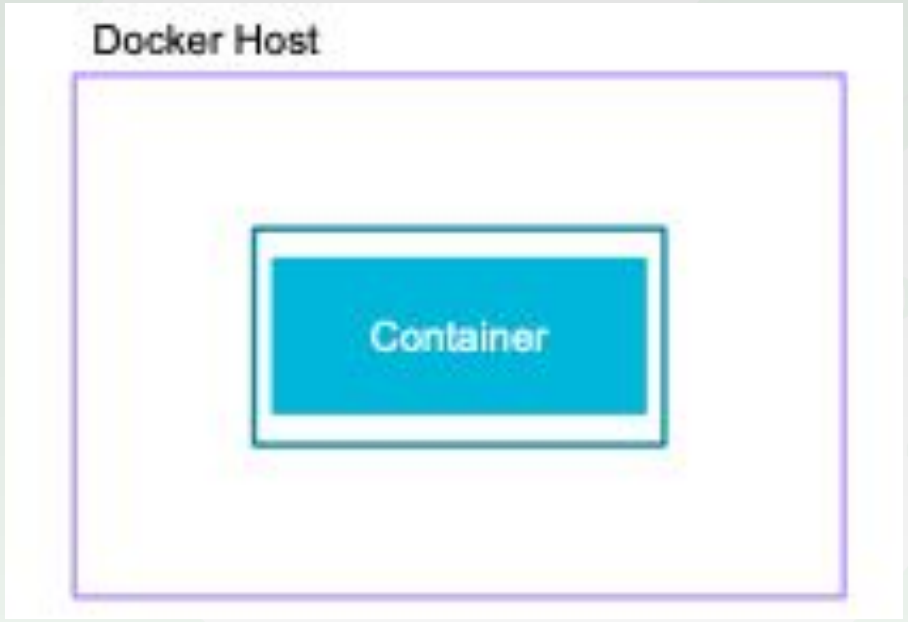
Network Drivers

- **Host** removes network isolation between the docker host and docker containers. It uses the host's networking directly.

- Host networks are best when the network stack should not be isolated from the Docker host, but we want other aspects of the container to be isolated.

Network Drivers

- **None** network driver disables all networking of containers.
- **None** network driver will not configure any IP for the container and doesn't have any access to the external network as well as to other containers.
- It is used when a user wants to block the networking access to a container.



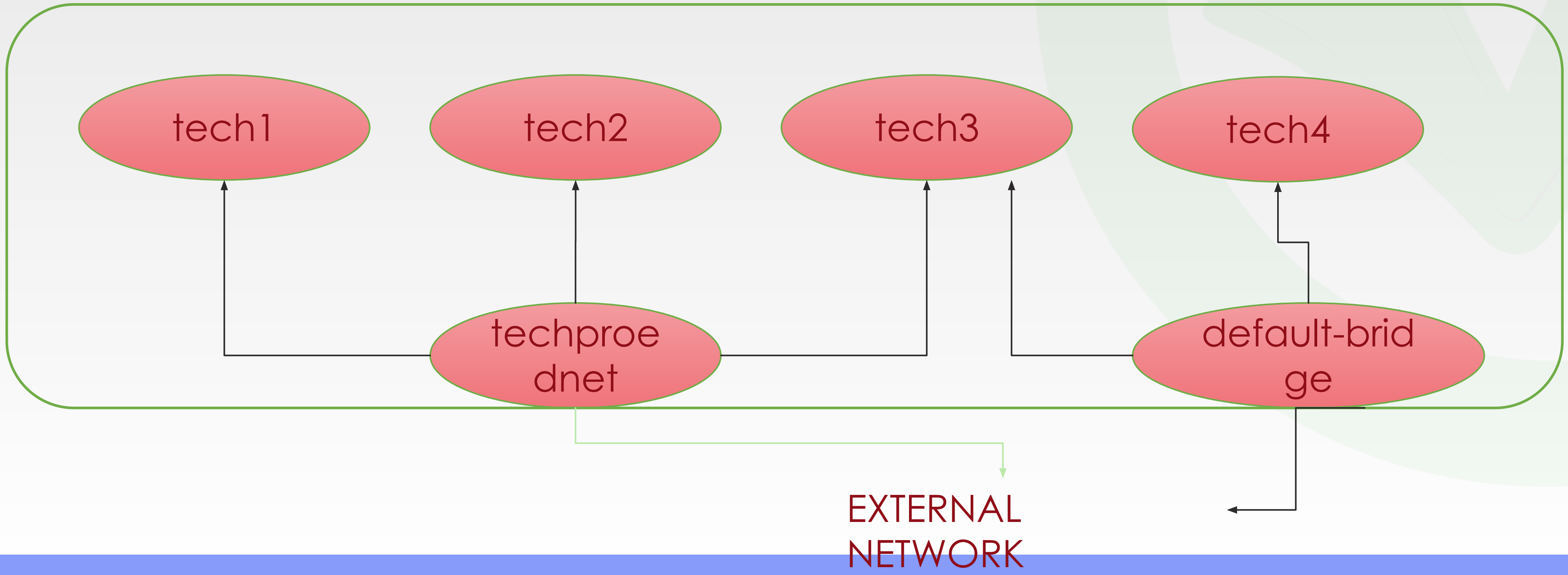
User-defined bridge networks

•

User-defined Networks

- In addition to the default networks, users can create their own networks called user-defined networks of any network driver type.

$ docker network create --driver bridge techproednet

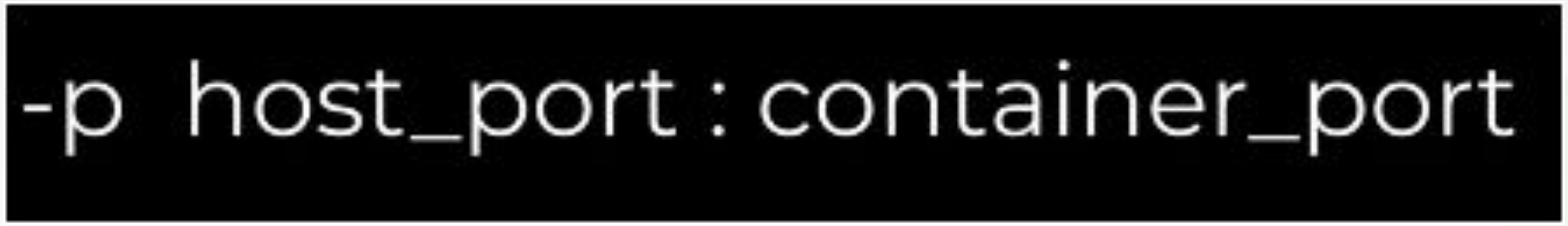


Run – Port Mappings

•

Run – Port Mappings

- By default, when you create a container, it does not publish any of its ports to the outside world. To make a port available to services outside of Docker, or to Docker containers which are not connected to the container's network, use the **--publish** or **-p flag.**

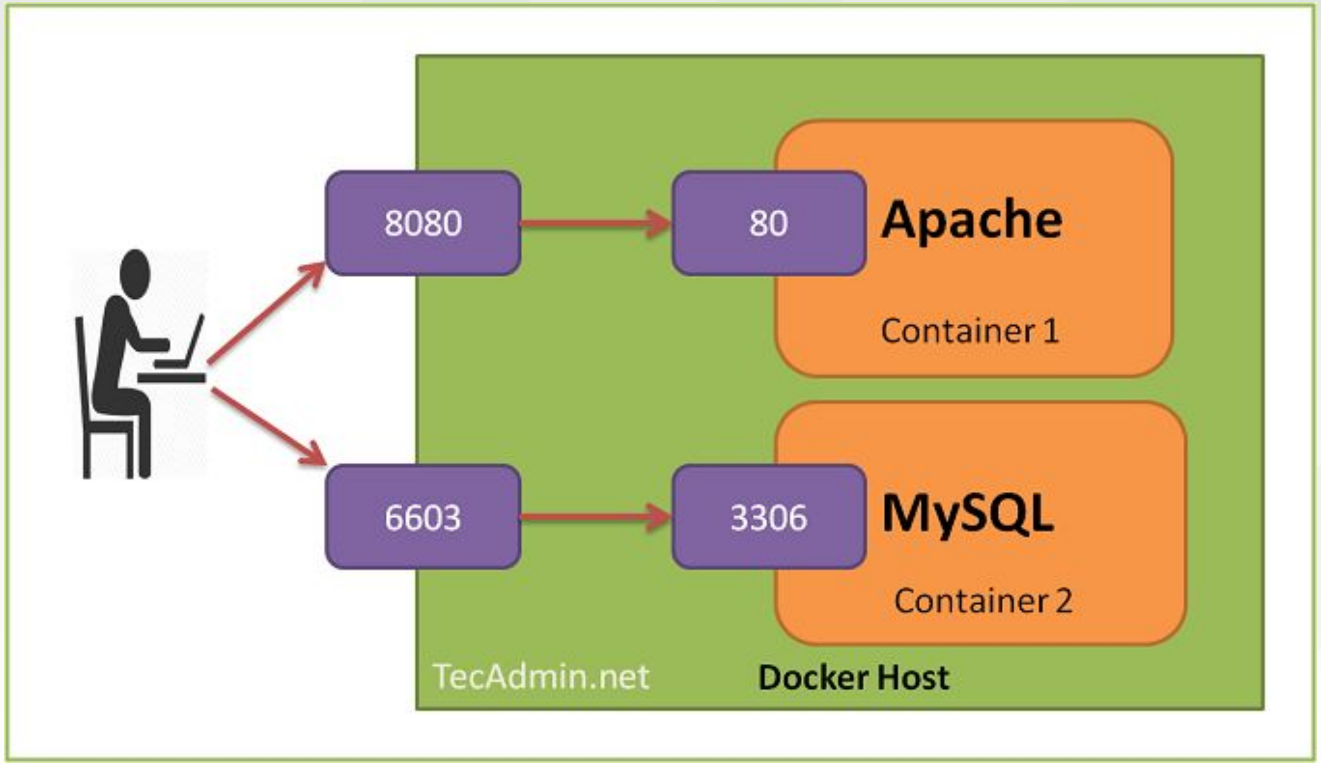


```
-p host_port : container_port
```

-P -> Random Ports

Run – Port Mappings

```
$ docker run-d-p 8080:80 apache_image

$ docker run d -p 6603:3306 mysql_image
```

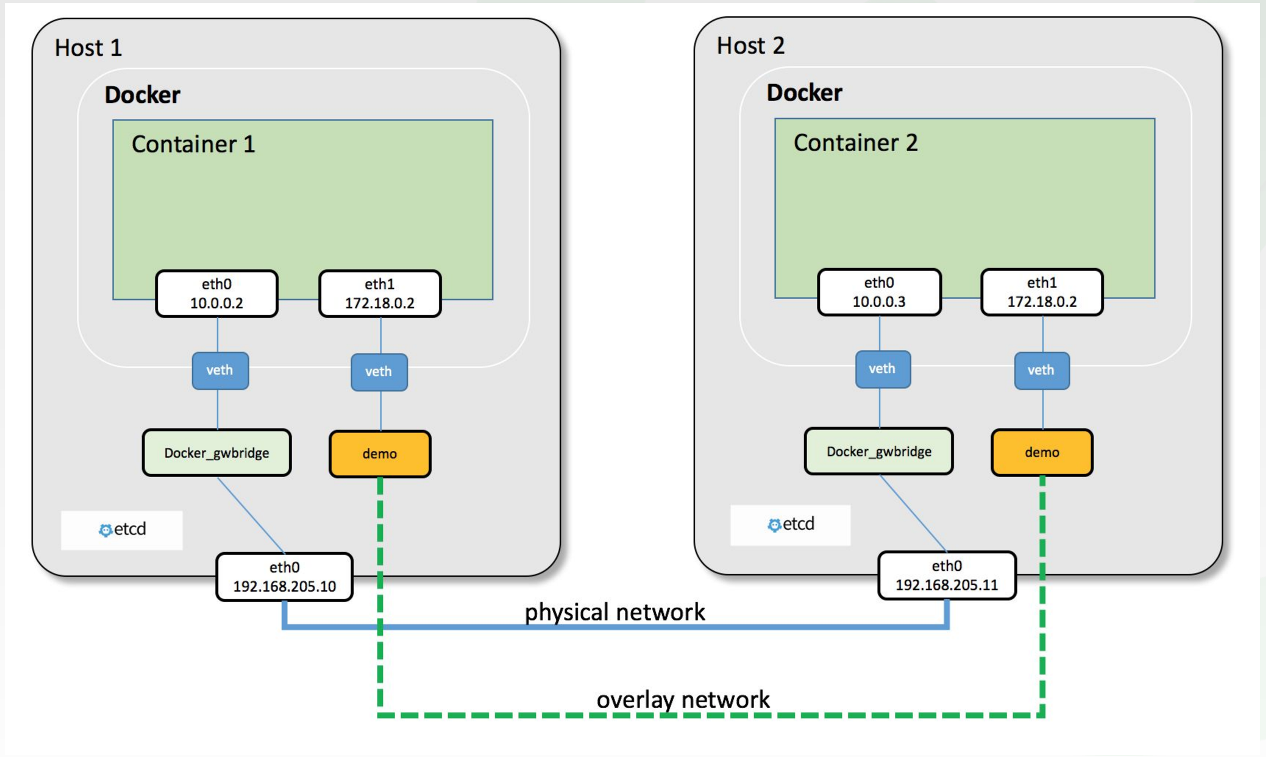


Other Network Drivers

•

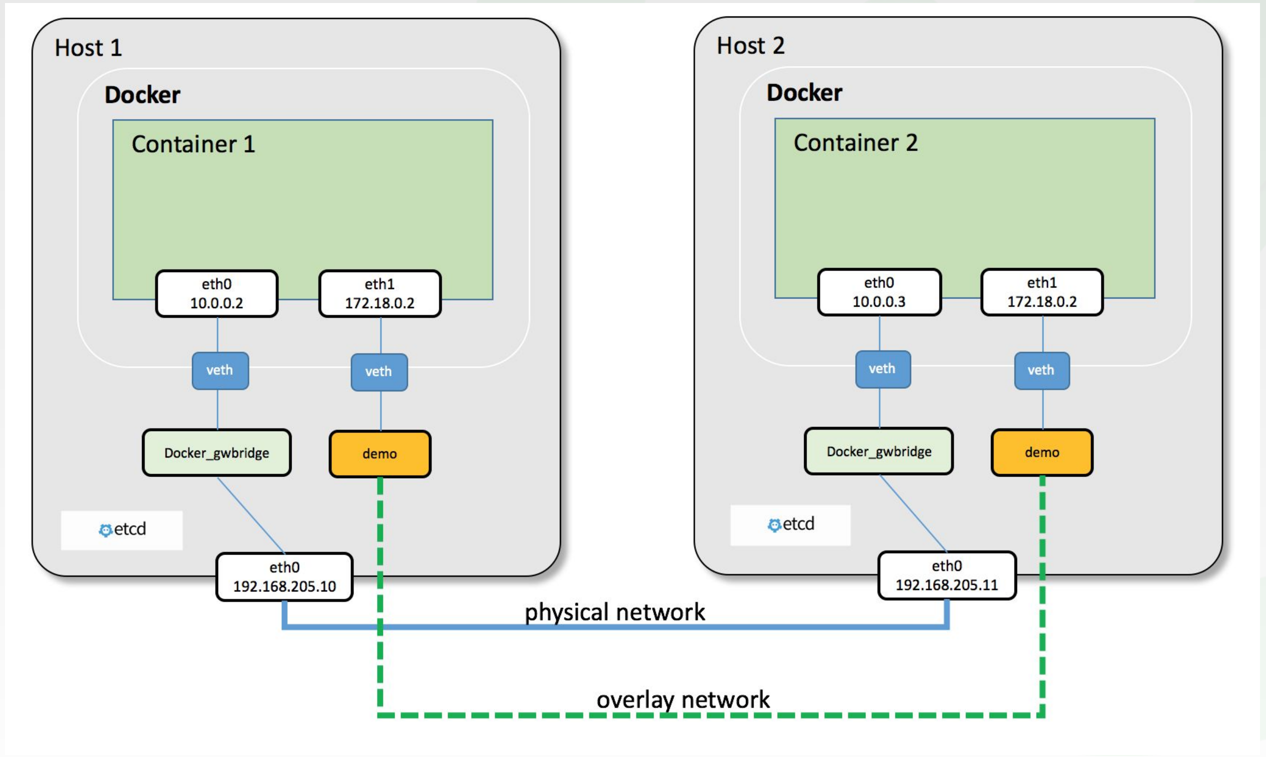
Network Drivers

- **The Overlay network driver** creates a distributed network among **multiple Docker deamon hosts.**



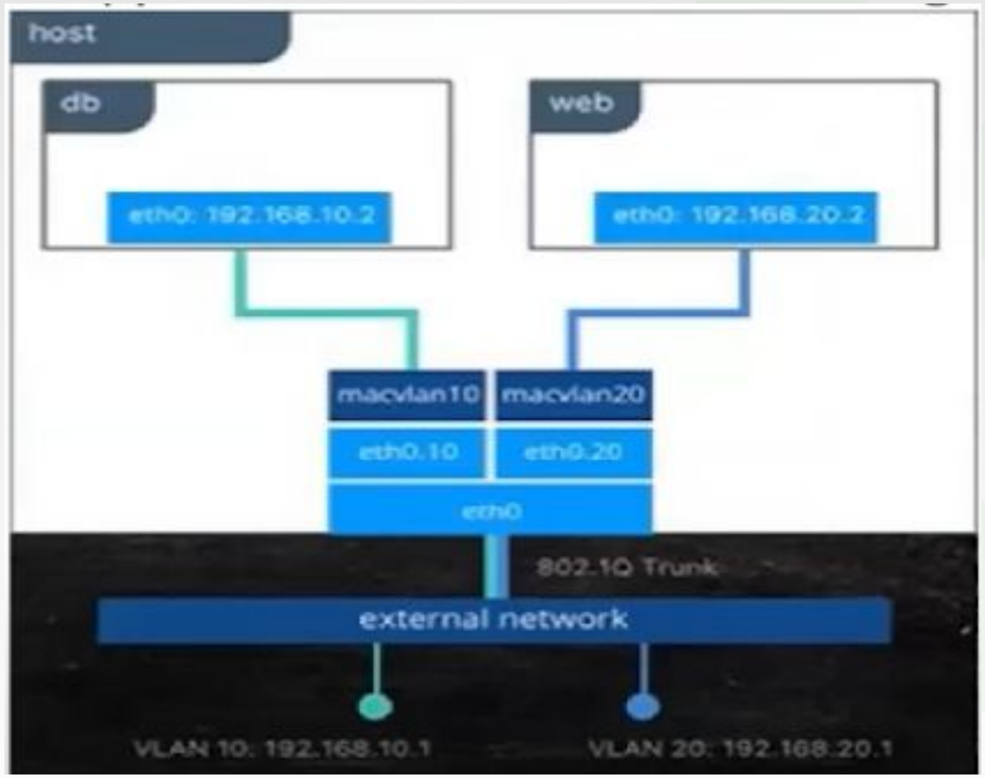
Network Drivers

- Overlay networks connect multiple Docker daemons together and enable swarm services to communicate with each other.



Network Drivers

- **MacVlan network** driver supplies the containers networking as if they have physical NICs.



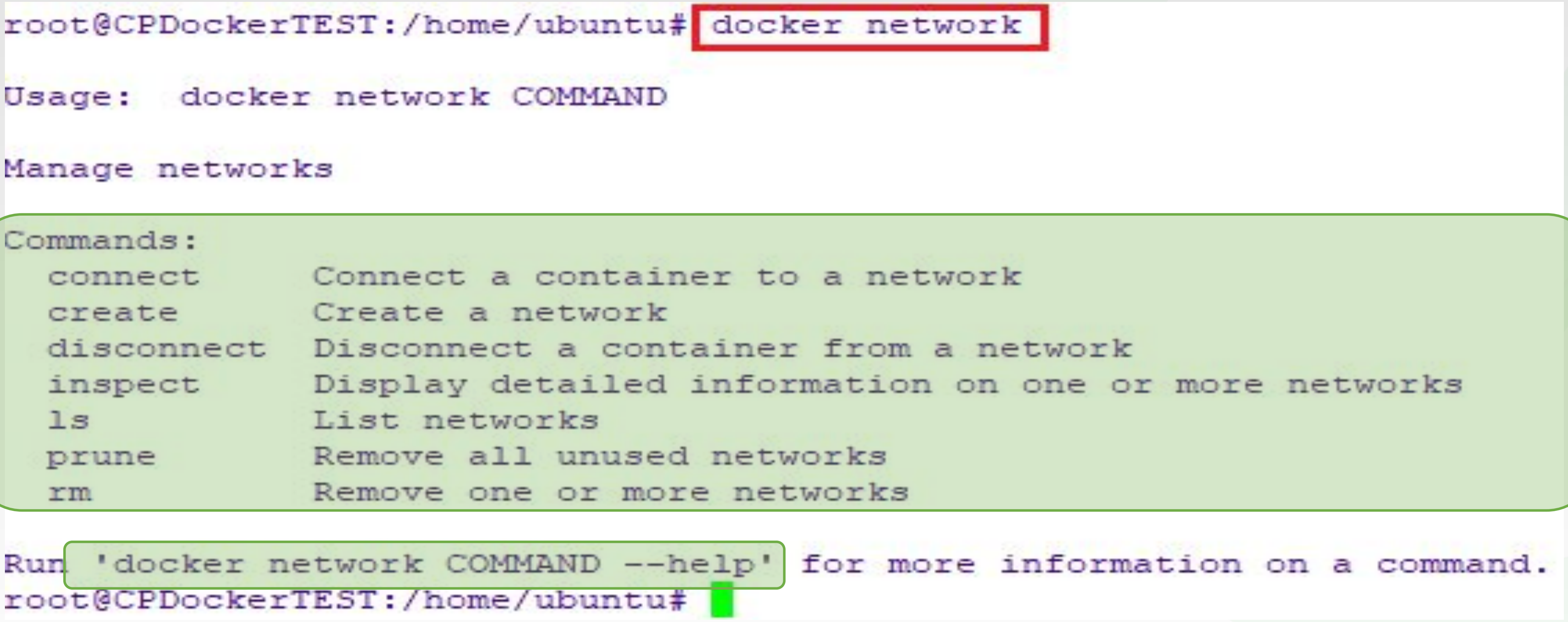
Network Drivers

- Macvlan networks allow you to assign a MAC address to a container, making it appear as a physical device on your network.

- Using the macvlan driver is sometimes the best choice when dealing with legacy applications that expect to be directly connected to the physical network, rather than routed through the Docker host's network stack.

Docker Network Commands

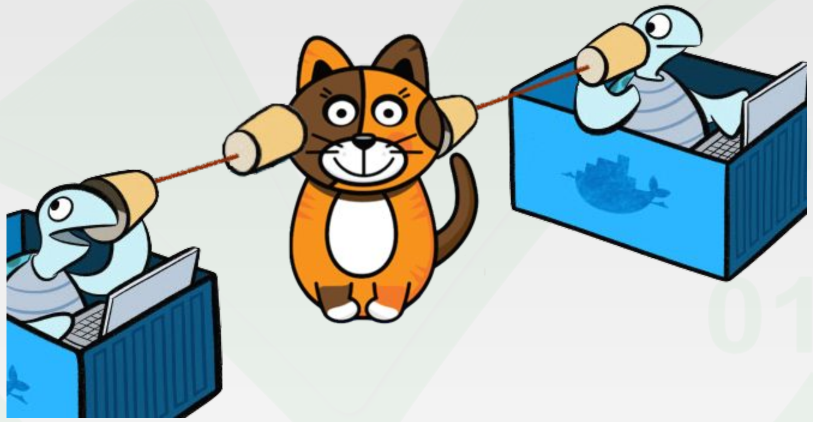
Docker Network Commands

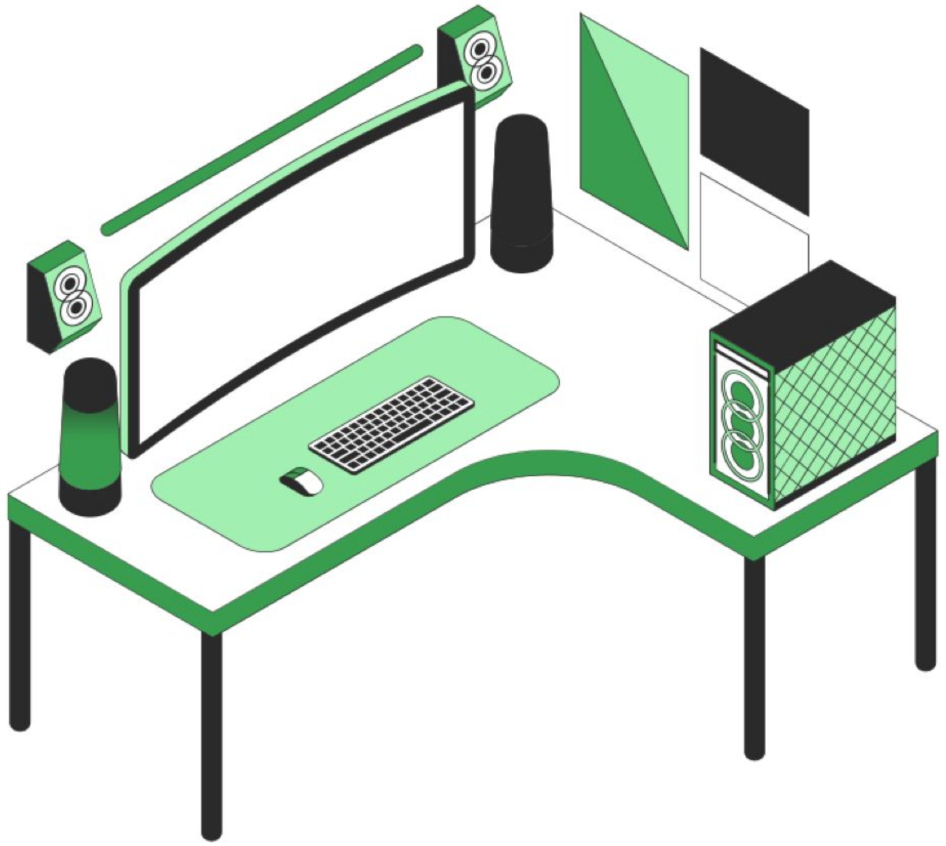


```
root@CPDockerTEST:/home/ubuntu# docker network

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

Run 'docker network COMMAND --help' for more information on a command.
root@CPDockerTEST:/home/ubuntu#
```

Docker Networking Summary

- User-defined bridge networks are best when you need multiple containers to communicate on the same Docker host.
- Host networks are best when the containers should not be isolated from the Docker host, but you want other aspects of the container to be isolated.
- Overlay networks are best when you need containers running on different Docker hosts to communicate, or when multiple applications work together using swarm services.
- Macvlan networks are best when you are migrating from a VM setup or need your containers to look like physical hosts on your network, each with a unique MAC address.
- Third-party network plugins allow you to integrate Docker with specialized network stacks.





Do you have any questions?

Send it to us! We hope you learned something new.