Docker 101 Workshop

DockerCon EU October 2017

Your Instructors

Mike Coleman

Michael Irwin

John Zaccone

Agenda

Part 1

- Running Containers
- Images
- Dockerfiles
- Bind Mounts
- Port Mapping

Part 2

- Understanding the Docker Filesystem
- Understanding Volumes

Part 3

- Docker Networking
- Docker Swarm Intro

dockercon17 EU

Part 1

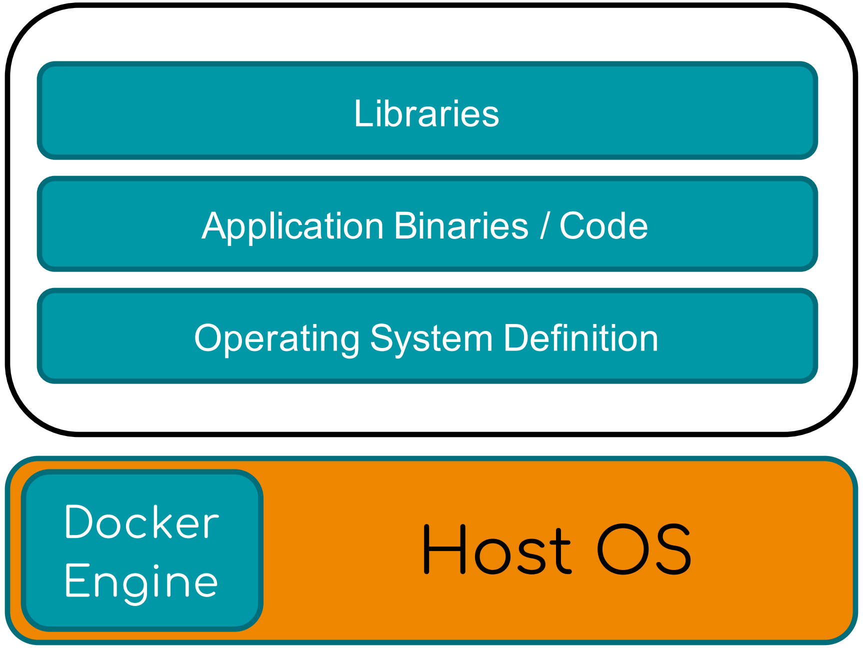
Running containers, Dockerfiles, Bind mounts

Containers are Not VMs?

VMs

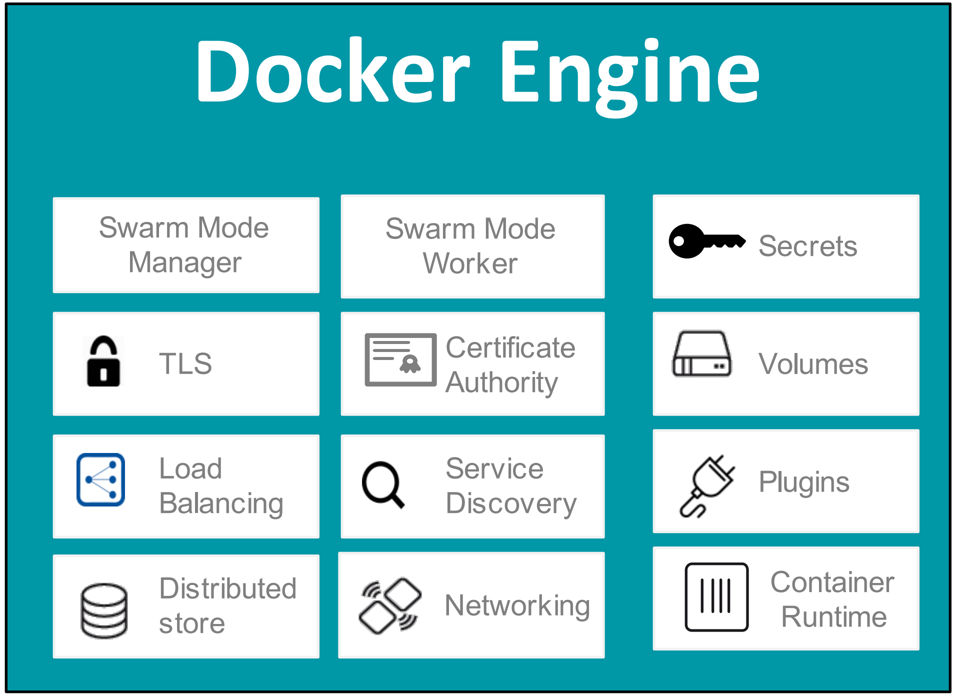
Containers

What is a Container?



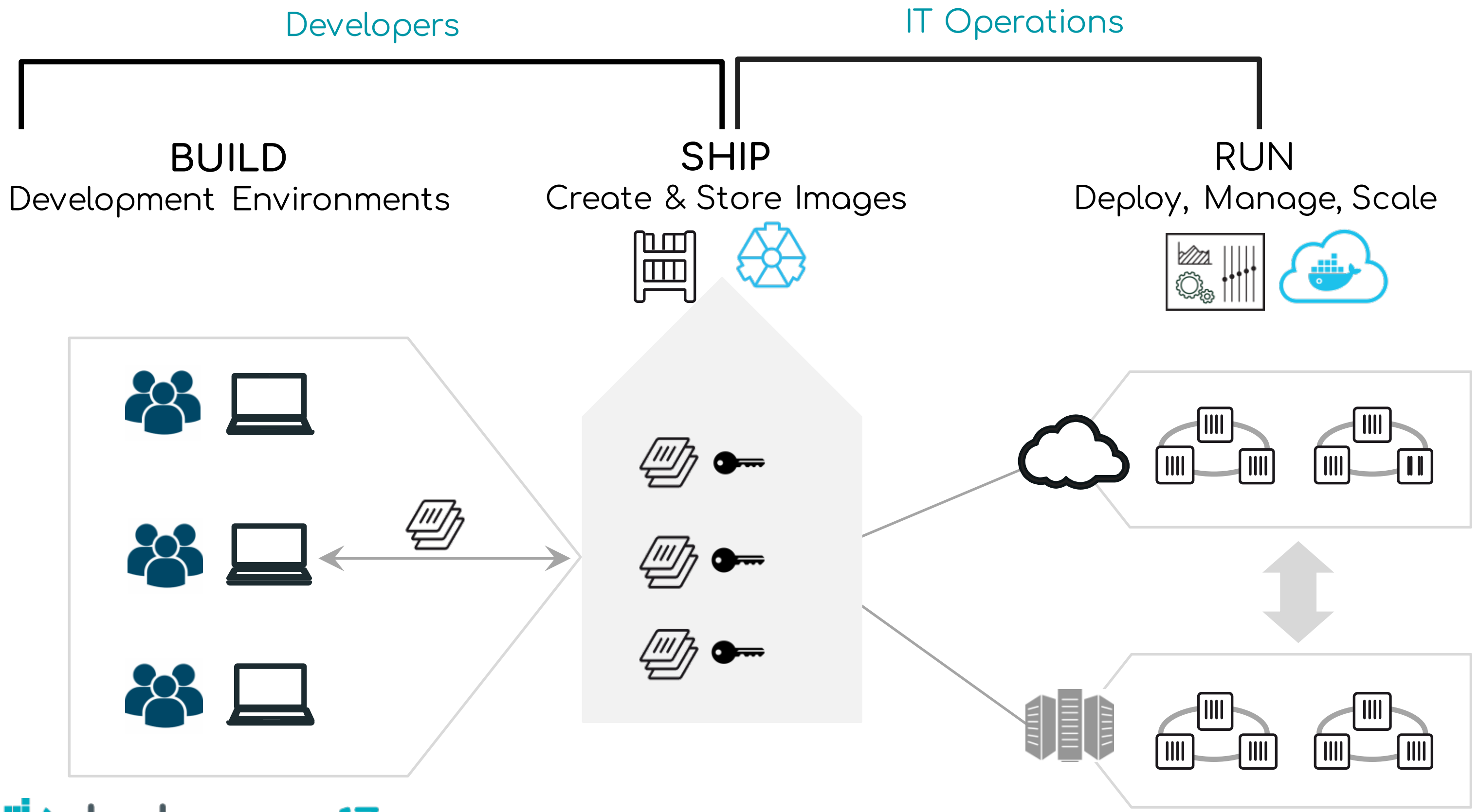
- Isolated Operating System Process
- Includes Everything The App Needs to Run
- Shares Underlying OS Kernel
- Inherently Portable
- Managed by Docker Engine

Docker Engine



- Powerful yet simple, built in orchestration
- Declarative app services
- Built in container centric networking
- Built in default security
- Extensible with plugins, drivers and open APIs

Build, Ship, Run



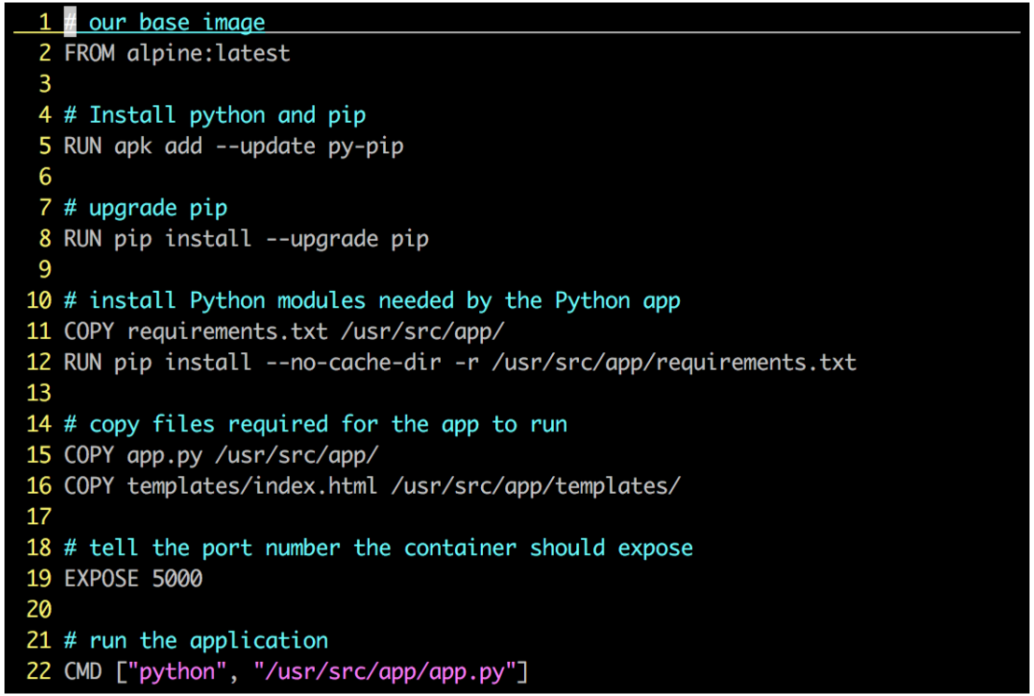
Docker Images

- Read only
- Build-time artifact
- Basis for running containers
- Built using Dockerfile
- Stored on a registry

Managing Images

- Images are pushed and pulled from registries
- Registries can be SaaS / public or on-prem
- Tags can be applied to images to denote versions
- Effective Dockerfiles are extremely important

Dockerfile Example



```
# our base image
FROM alpine:latest

# Install python and pip
RUN apk add --update py-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

- Instructions on how to build a Docker image
- Looks very similar to “native” commands
- Important to optimize your Dockerfile

dockercon17 EU

Types of Running Containers

Single task:

```
$ docker container run alpine hostname
```

Background:

```
$ docker container run --detach alpine top
```

Interactive:

```
$ docker container run –interactive -tty alpine bash
```

Bind Mounts

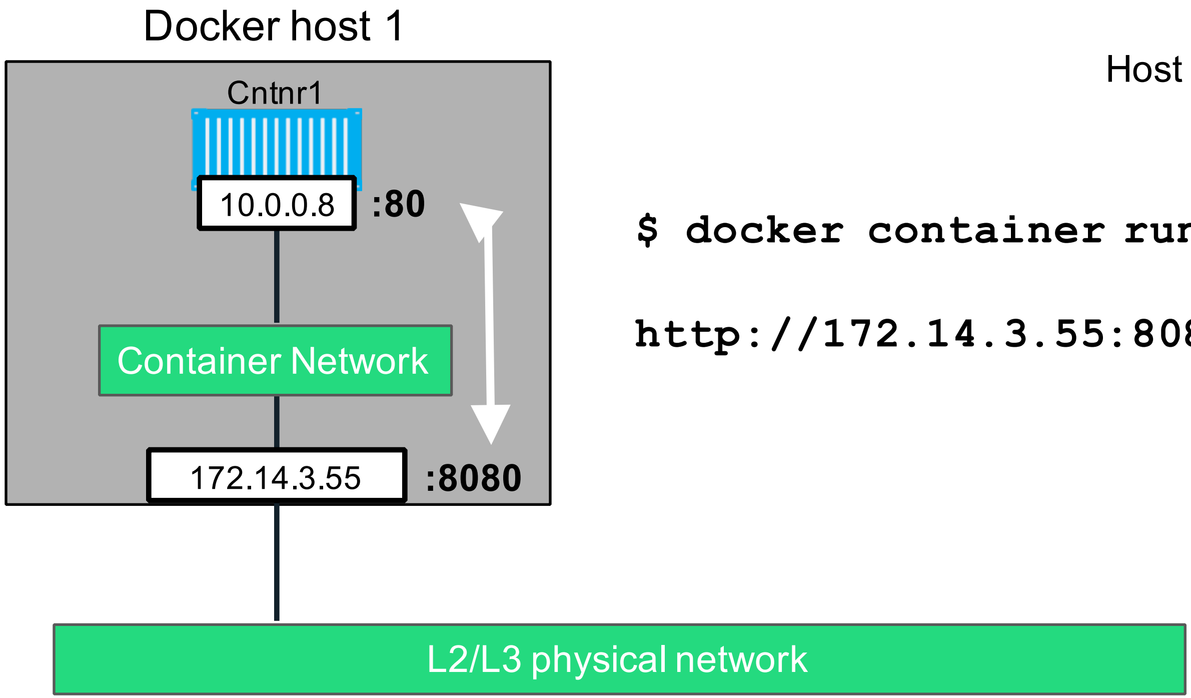
- Mount a directory on the host into the running container
- Good for source code
- Changes can be immediately reflected
- Not a volume

```
$ docker container run -v $(pwd):/usr/src/app webfrontend
```

Exposing Ports

- A host can only expose a given port once
- Some uses cases require the same port multiple times
- Docker uses port mapping to achieve this

Port Mapping



```
$ docker container run -p 8080:80 ...

http://172.14.3.55:8080
```

docker

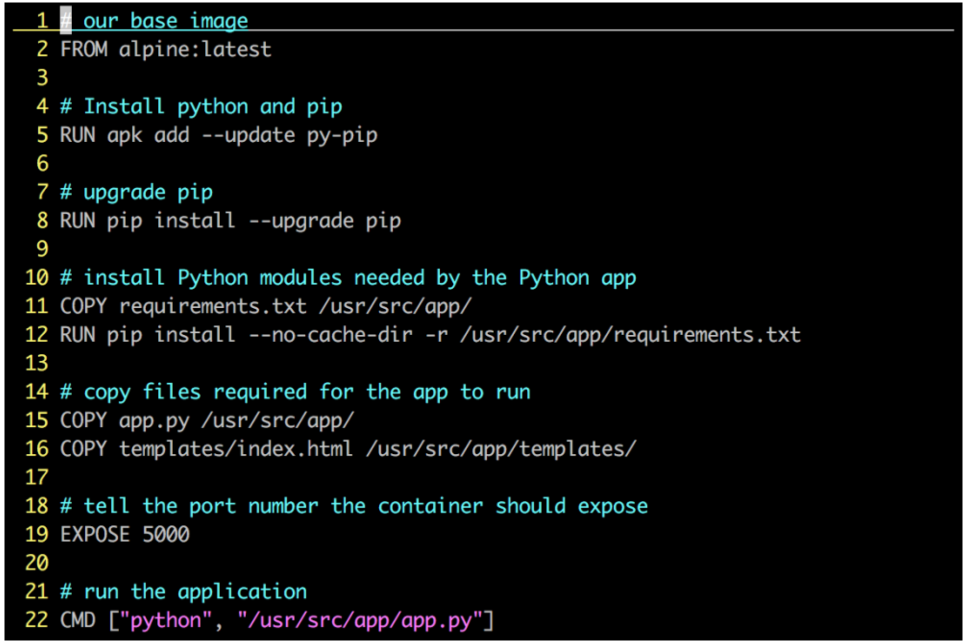
Lab: Part 1

https://github.com/mikegcoleman/docker101-linux

Part 2

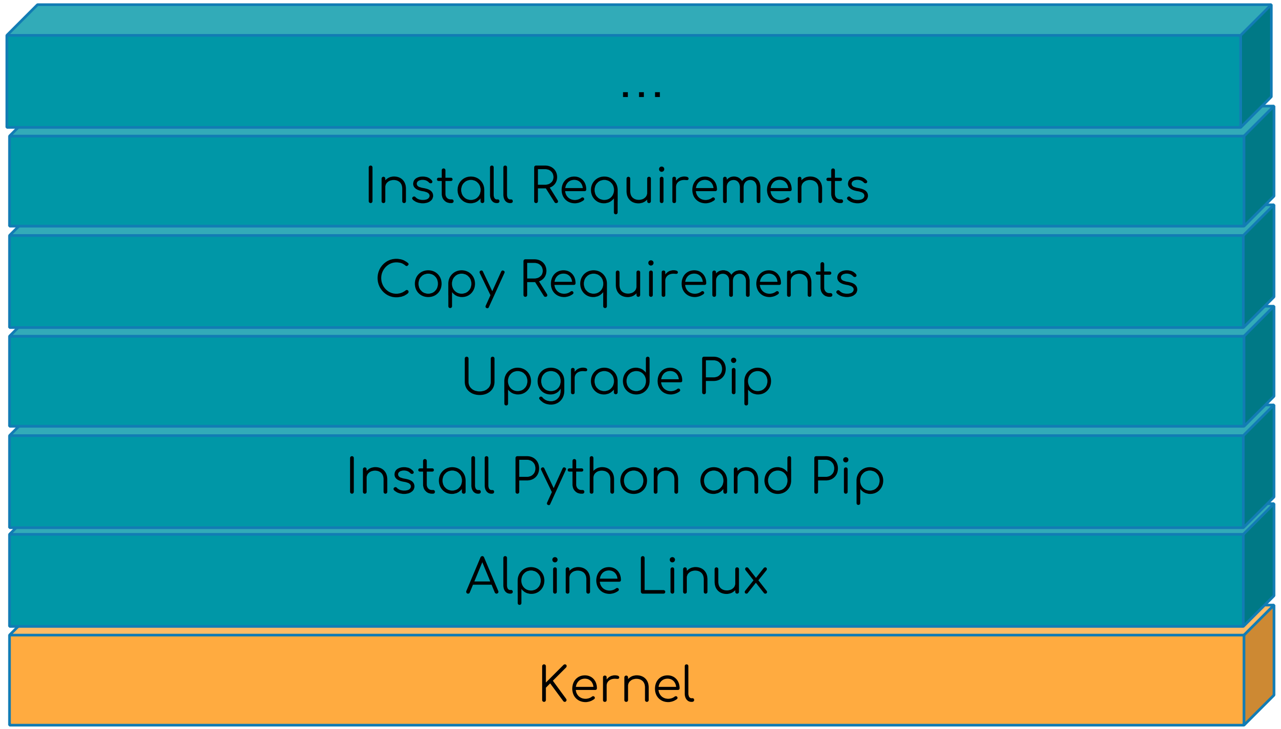
Docker filesystem, Volumes,

Let’s Go Back to Our Dockerfile



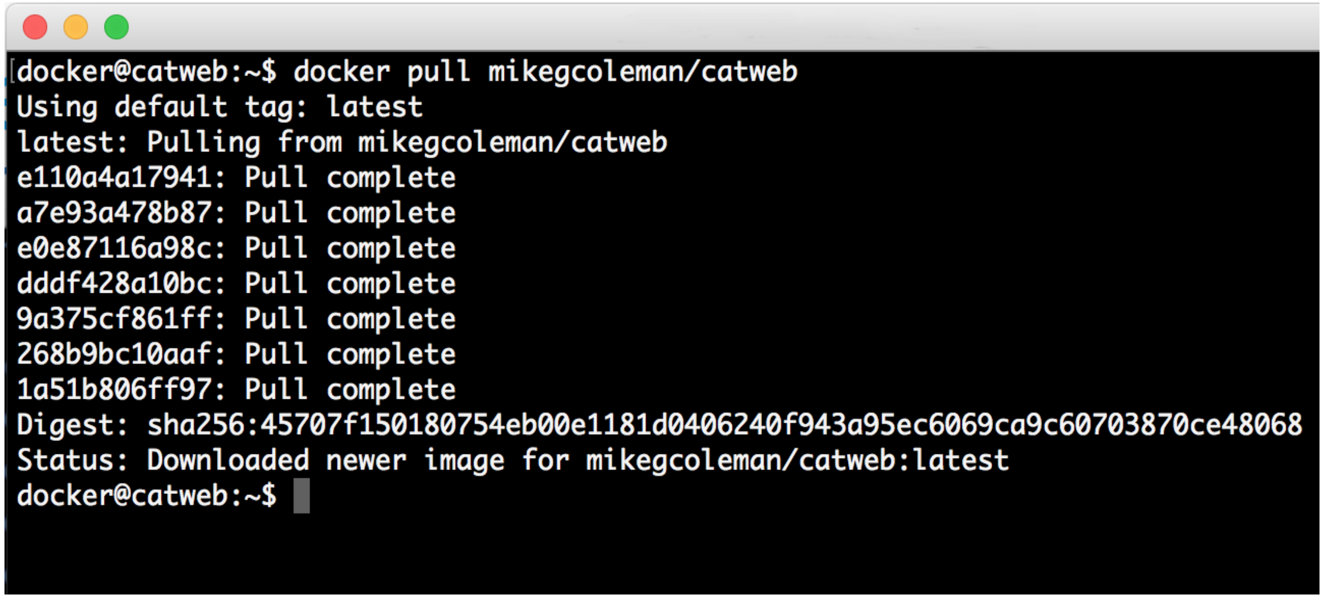
```
# our base image
FROM alpine:latest

# Install python and pip
RUN apk add --update py-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

Each Dockerfile Command Creates a Layer



dockercon17 EU

Pulling layers



```
docker@catweb:~$ docker pull mikegcoleman/catweb
Using default tag: latest
latest: Pulling from mikegcoleman/catweb
e110a4a17941: Pull complete
a7e93a478b87: Pull complete
e0e87116a98c: Pull complete
dddf428a10bc: Pull complete
9a375cf861ff: Pull complete
268b9bc10aaf: Pull complete
1a51b806ff97: Pull complete
Digest: sha256:45707f150180754eb00e1181d0406240f943a95ec6069ca9c60703870ce48068
Status: Downloaded newer image for mikegcoleman/catweb:latest
docker@catweb:~$
```

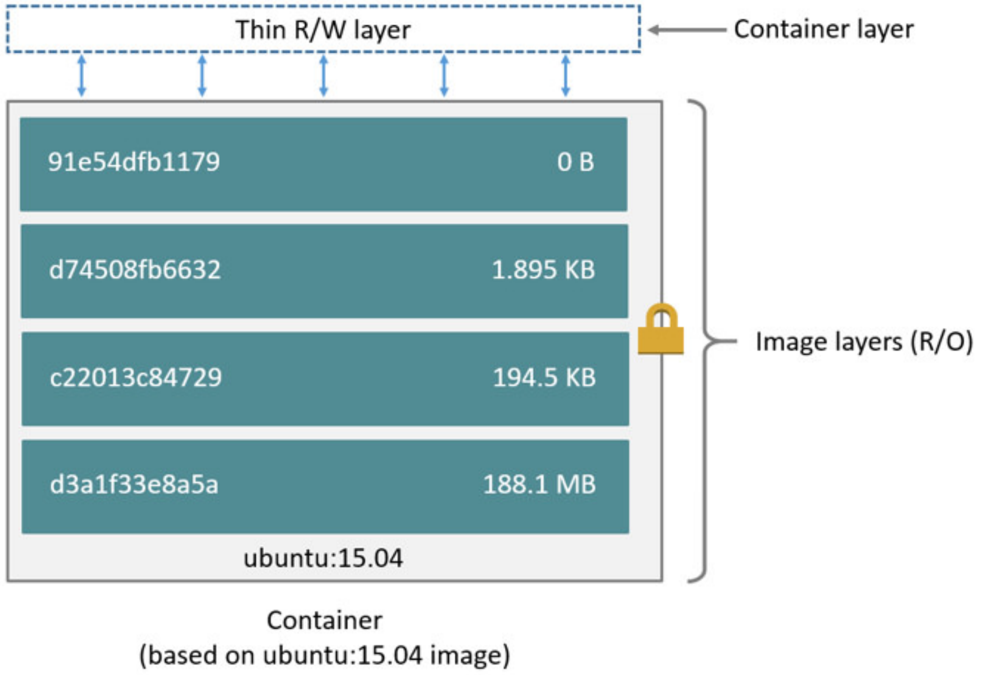
Docker Storage Drivers

- Union file system (UFS)
- Aggregates multiple FS primitives into a single logical FS in the image
- Several different drivers available

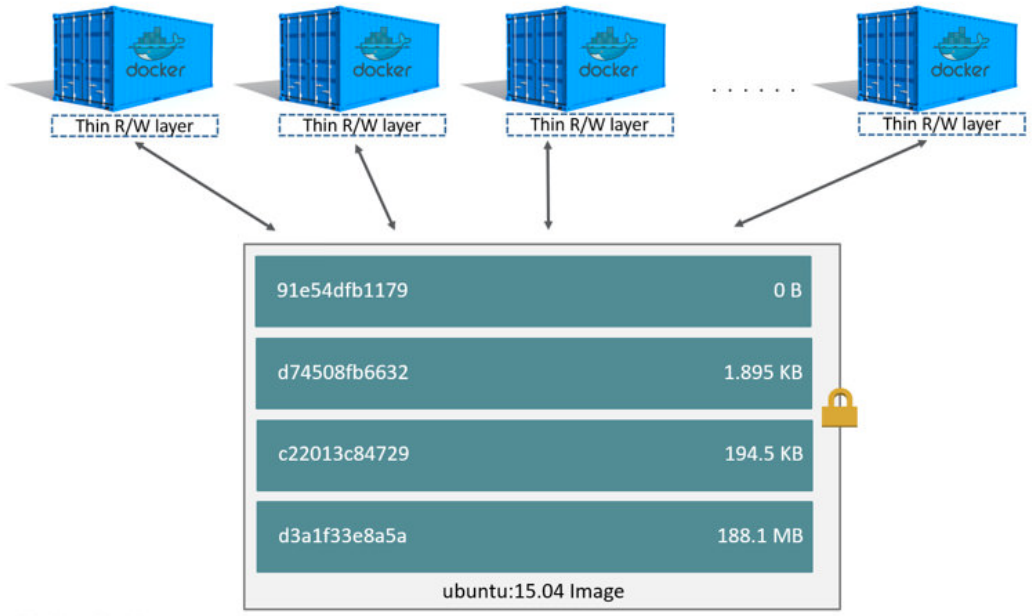
Copy on Write

- Super efficient:
  - Sub second instantiation times for containers
  - New container can take <1 Mb of space
- Containers appears to be a copy of the original image
- But, it is really just a link to the original shared image
- If someone writes a change to the file system, a copy of the affected file/directory is "copied up"

Containers vs. Images



Efficient Storage Utilization



Docker Volumes

- Volumes mount a directory on the host file system into the container at a specific location
- Volume directory structure is not managed by the Docker storage drive
- Can be created in via a Dockerfile, Docker Compose or CLI
- Named vs. Anonymous
- Use cases
  - Persistence
  - Performance

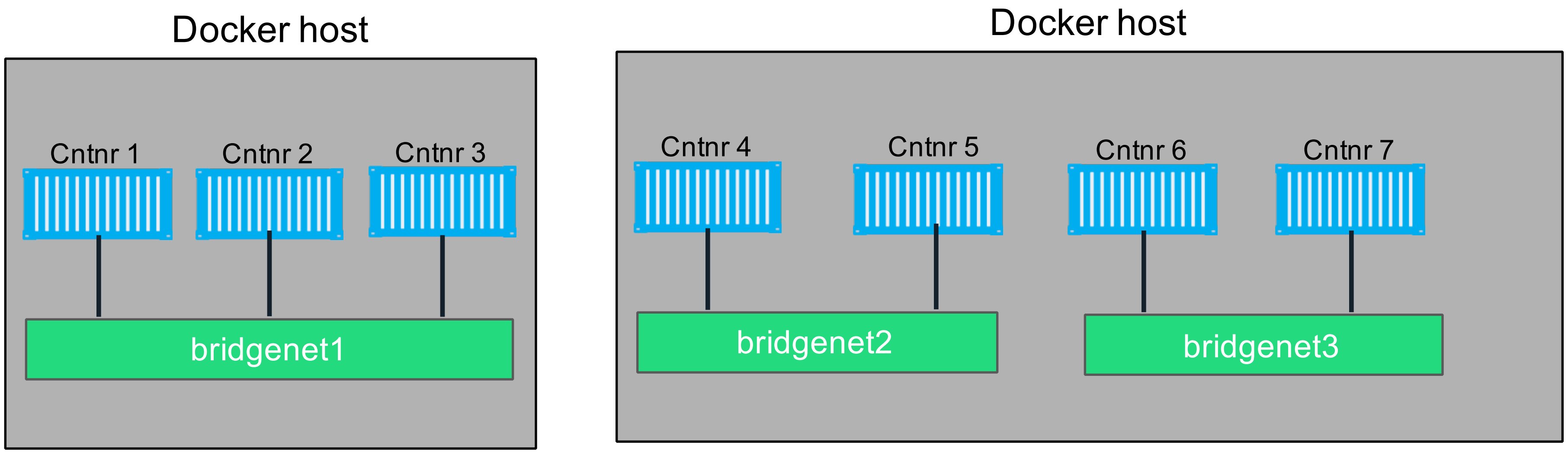
Lab: Part 2

https://github.com/mikegcoleman/docker101-linux

Part 3

Networking and Swarm

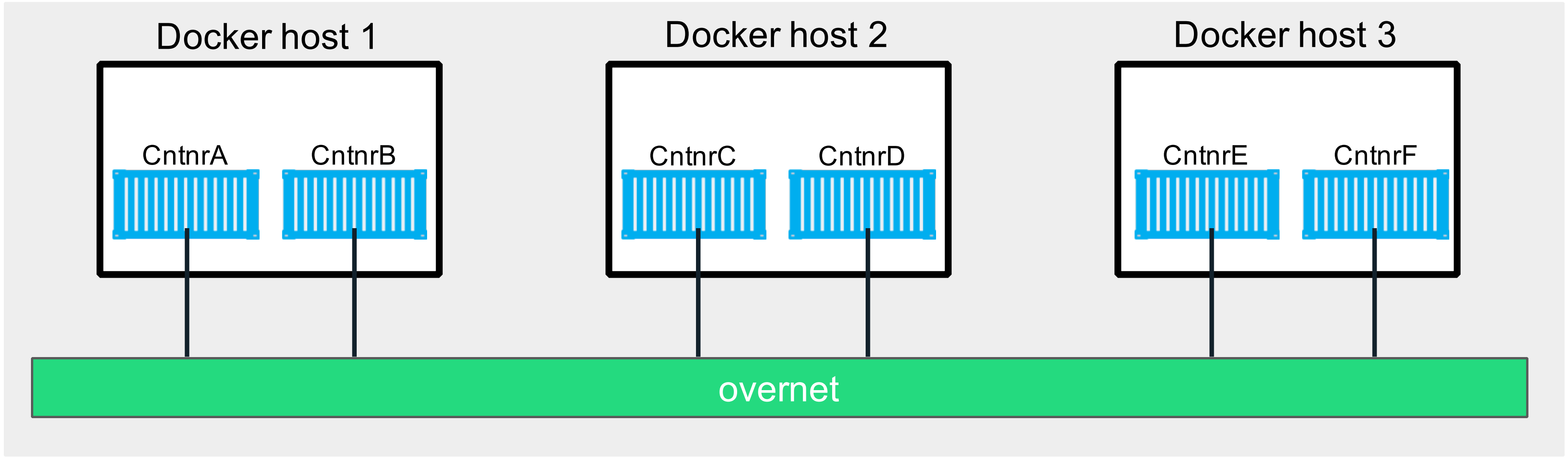
What is Docker Bridge Networking



```
docker network create -d bridge --name bridgenet1
```

What is Docker Overlay Networking

The **overlay** driver enables simple and secure **multi-host** networking



```
docker network create -d overlay --name overnet
```