



INTRODUCTION FOR BUSY PEOPLE DOCKER FOR THE BRAVE

AGENDA

AGENDA

- ▶ The need for containers
- ▶ Overview: Docker and the Docker Ecosystem
- ▶ Basic Docker Workflows

“Well, it worked on my machine!”

TRADITIONAL SOFTWARE DELIVERY SUFFERS FROM GAP BETWEEN DEV AND PROD

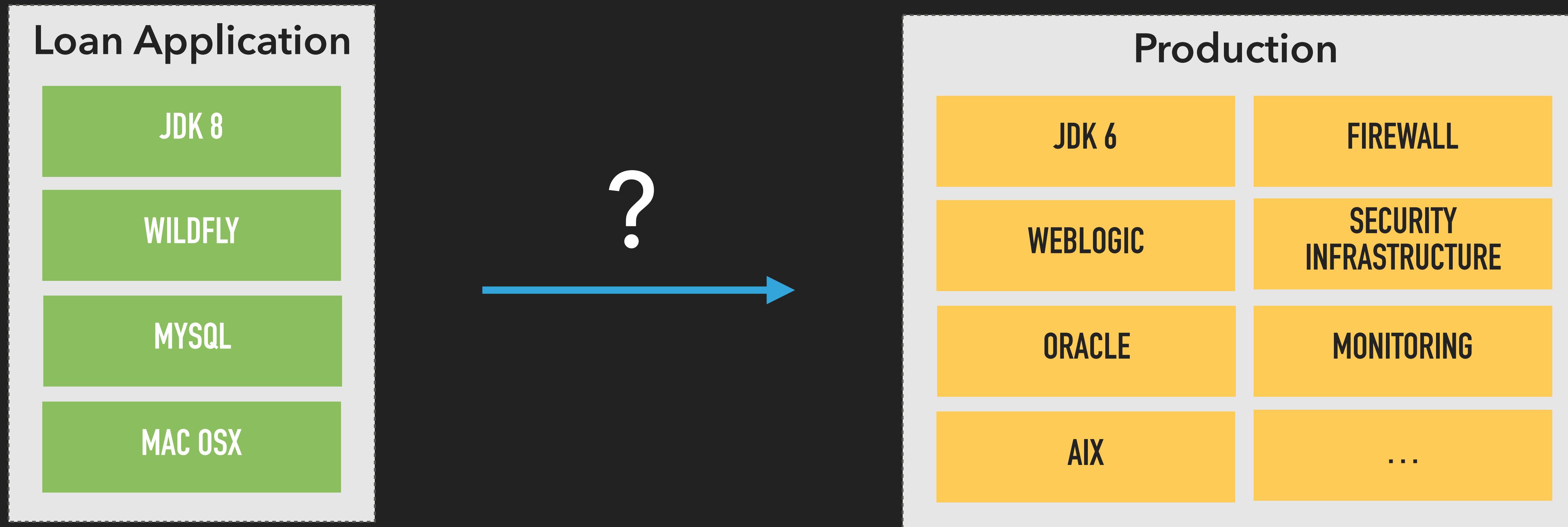
Development

FLEXIBILITY

Production

STABILITY AND ROBUSTNESS

TRADITIONAL SOFTWARE DELIVERY SUFFERS FROM GAP BETWEEN DEV AND PROD





Magic Operations Guide Template



Unicorns do not exist!

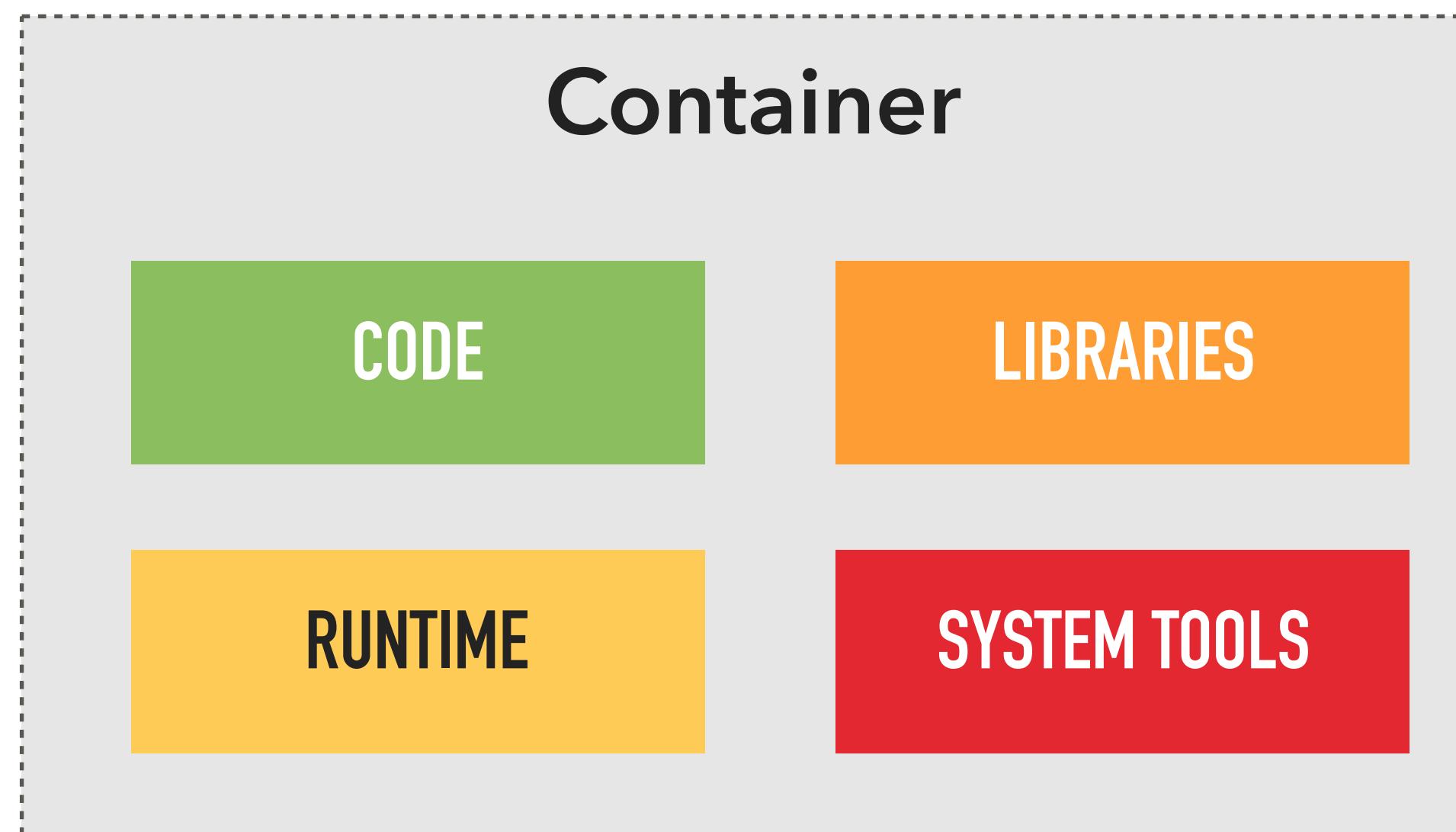
CONTAINERIZED APPLICATIONS OFFER A WAY TO NARROW THE GAP

- ▶ Immutable containers reduce human errors
- ▶ Instead of “works on my machine” you deploy the machine
- ▶ Think diapers

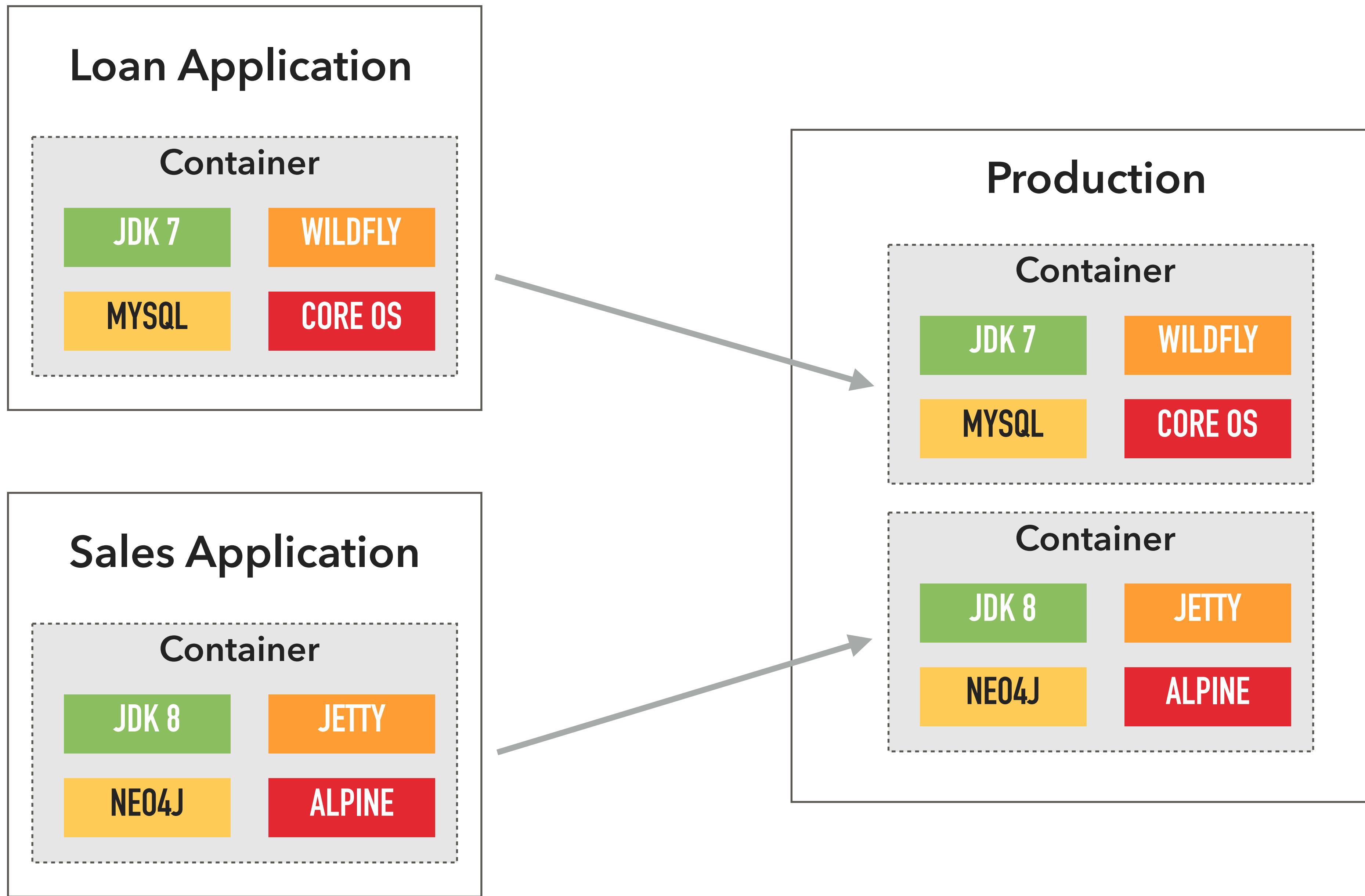


THE NEED FOR CONTAINERS

PACKAGE ONCE
DEPLOY ANYWHERE

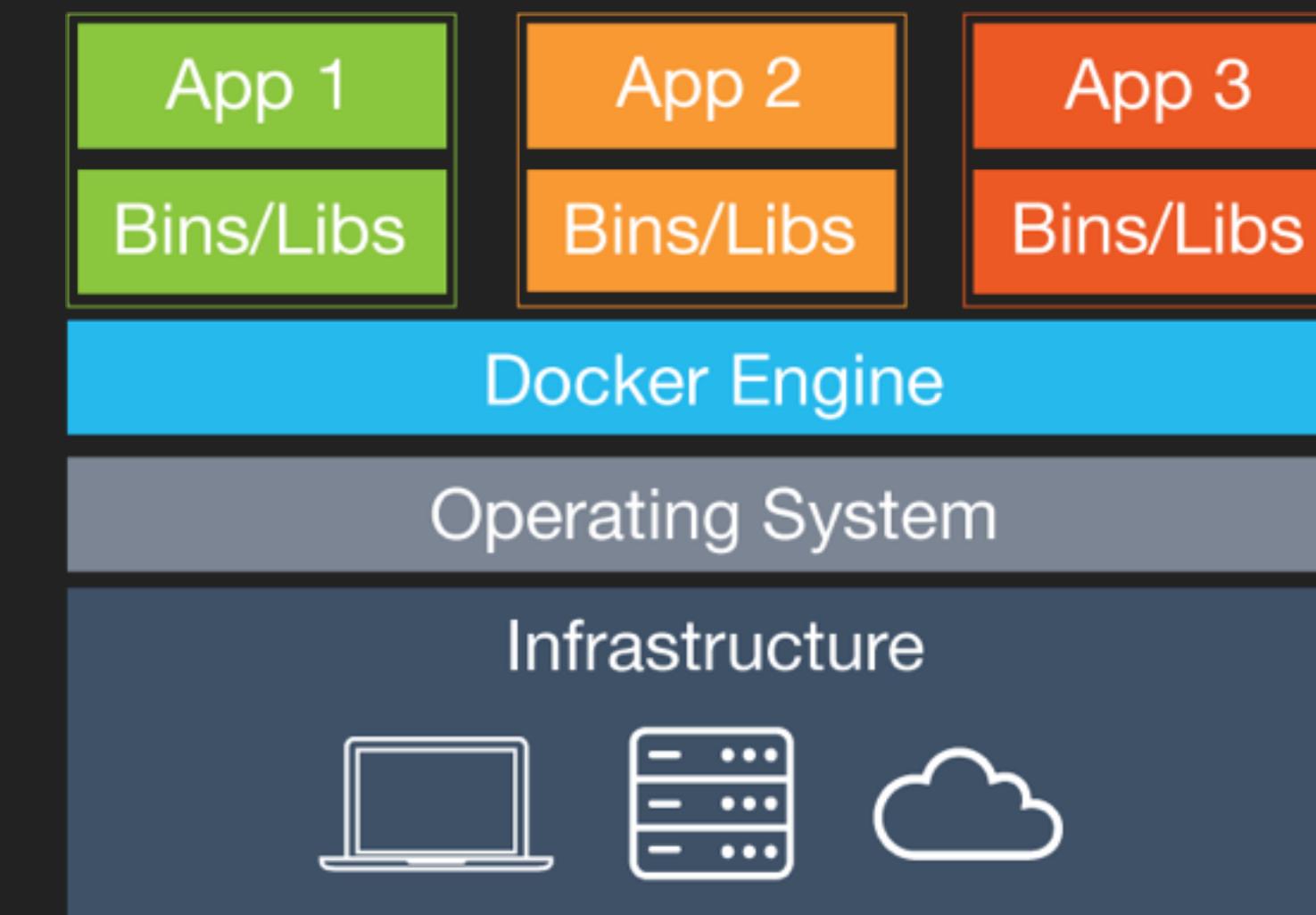
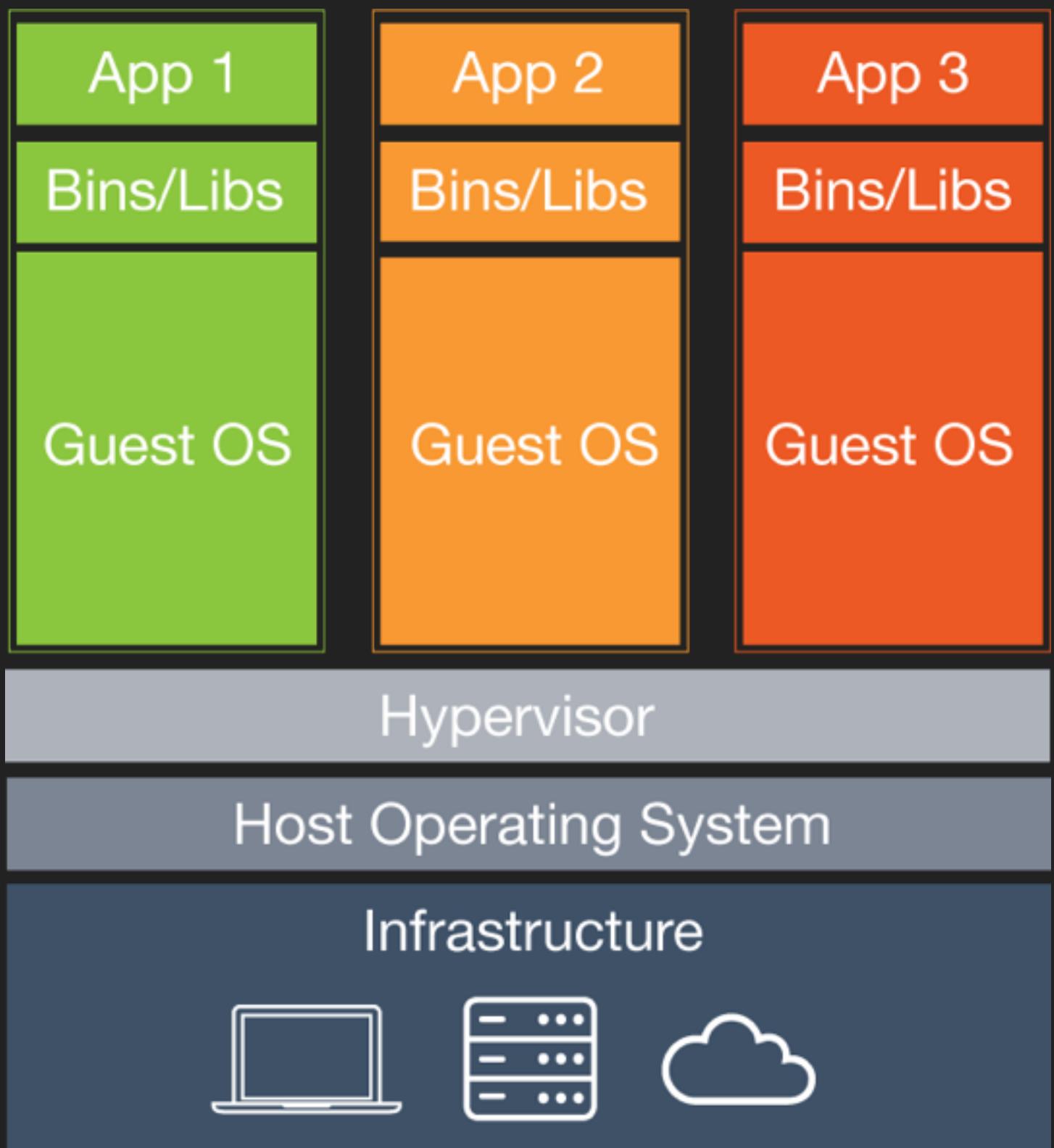


THE NEED FOR CONTAINERS



THE NEED FOR CONTAINERS

VIRTUALIZATION VS. CONTAINERS



SPEED

FOOTPRINT

SIMPLICITY

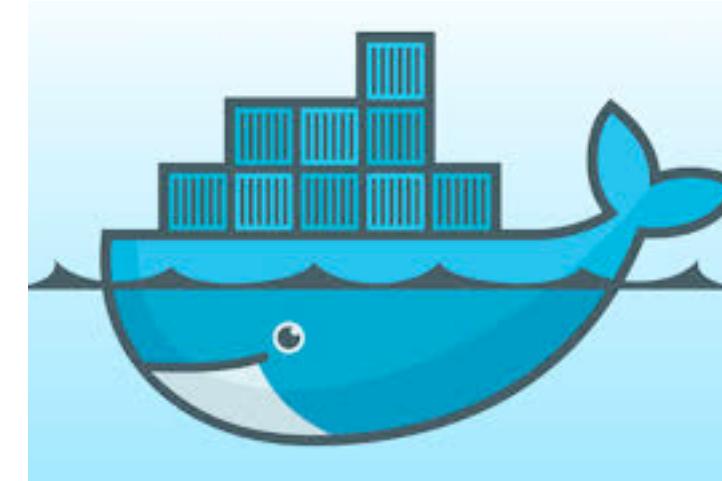


DOCKER BOOTCAMP

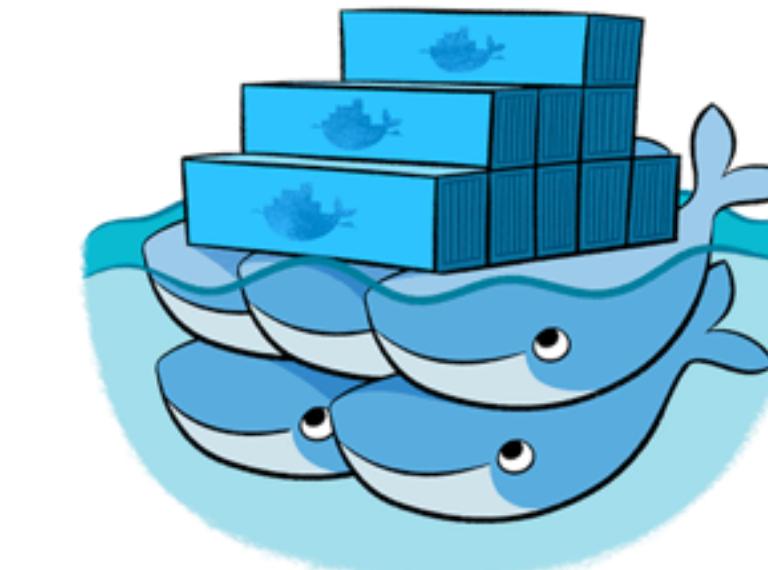
THE DOCKER ECOSYSTEM CONSISTS OF FIVE ESSENTIAL PARTS



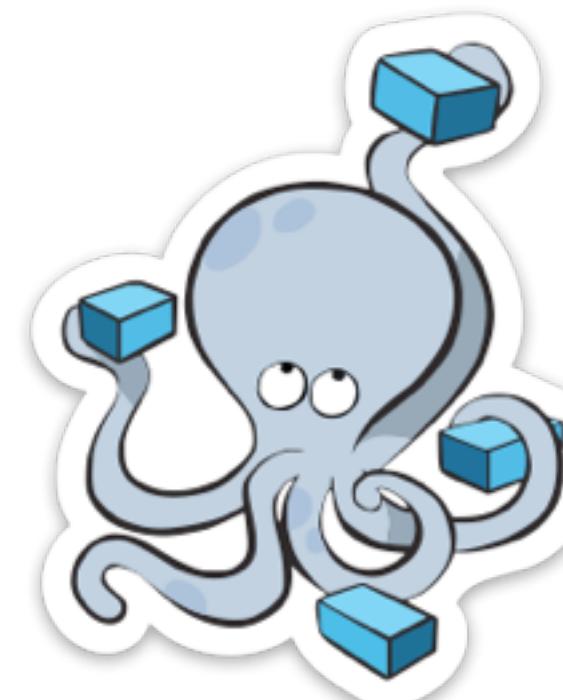
Machine



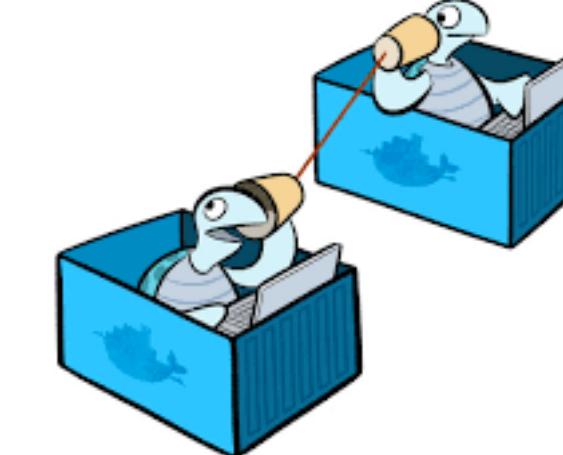
Docker



Swarm



Compose

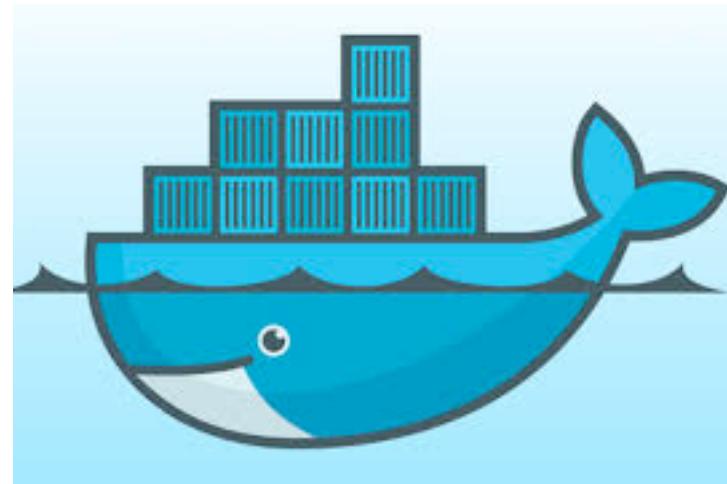


Network

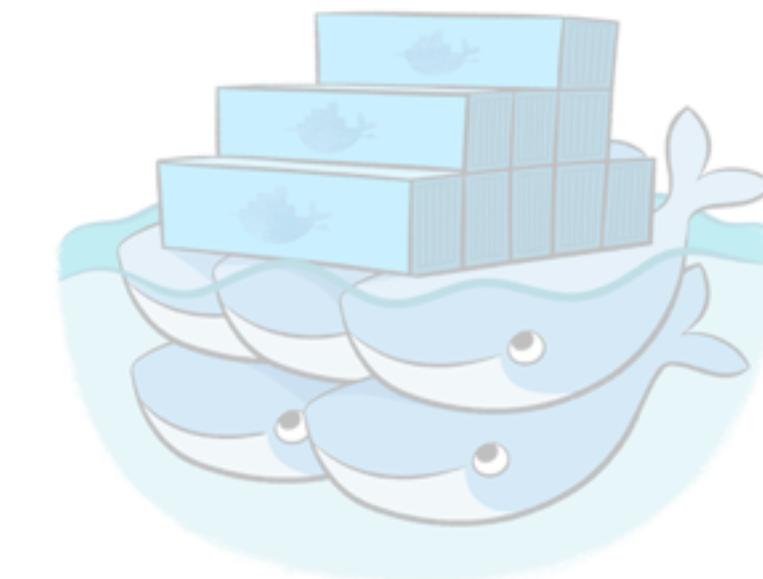
WE'LL ONLY COVER THE BASICS



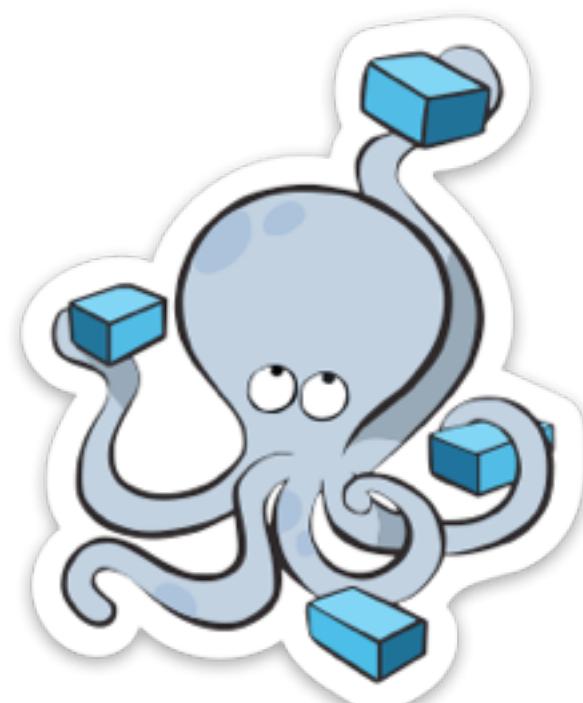
Machine



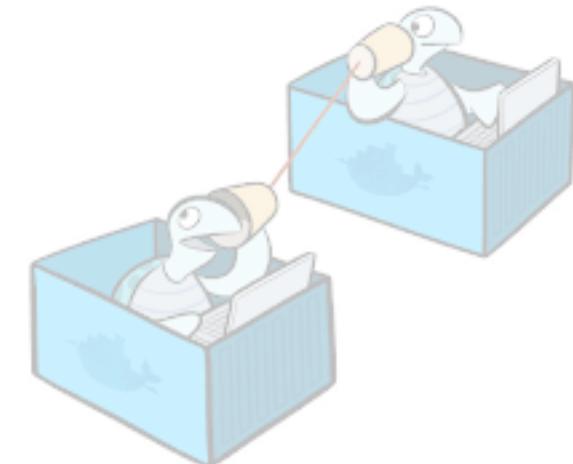
Docker



Swarm



Compose

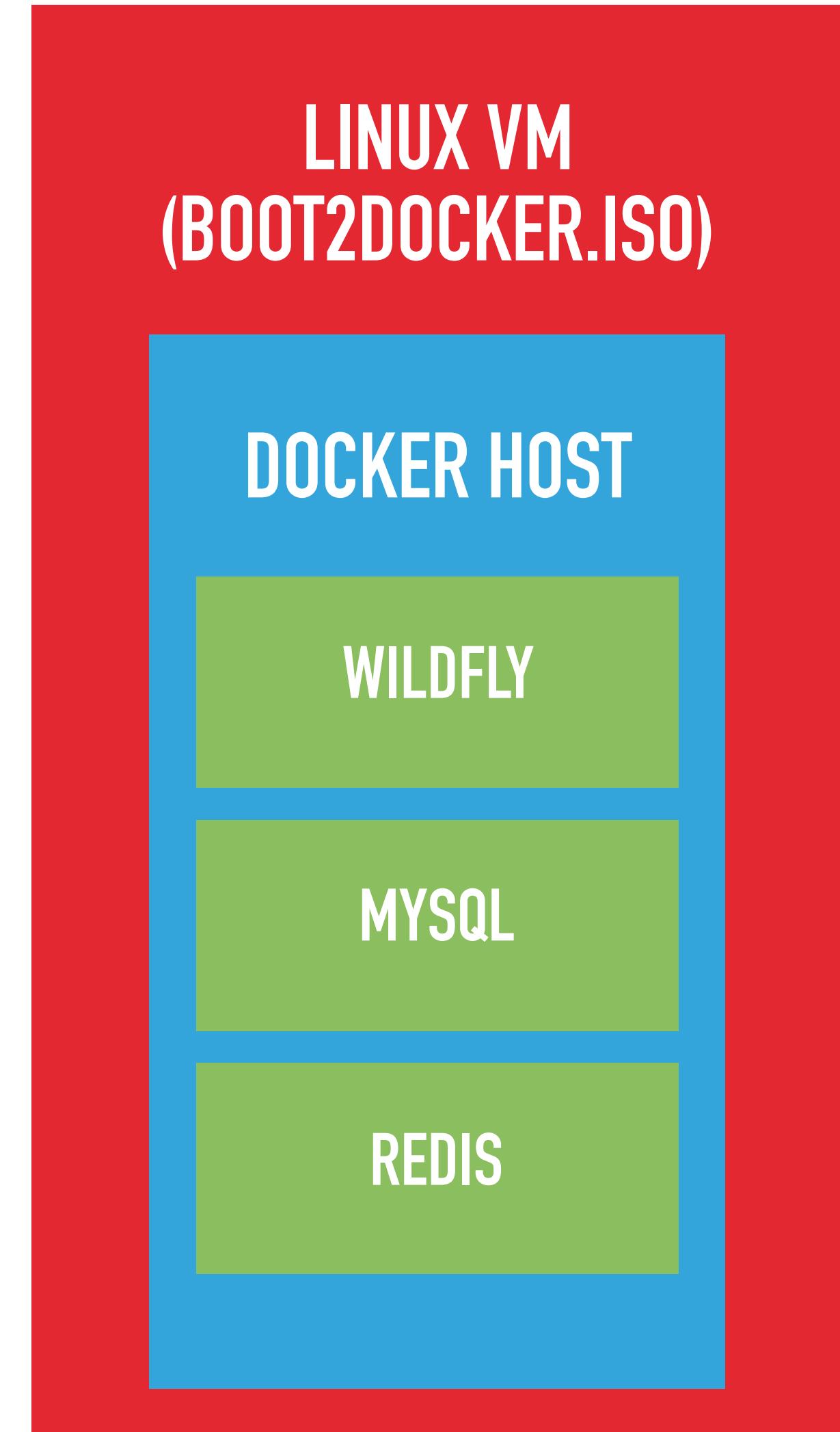


Network



DOCKER-MACHINE HELPS GETTING STARTED ON NON LINUX BOXES

```
$ docker-machine create --driver=virtualbox test  
$ eval $(docker-machine env test)
```

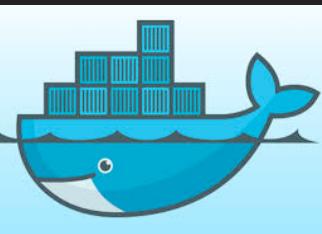


DOCKER TOOLBOX TO THE RESCUE

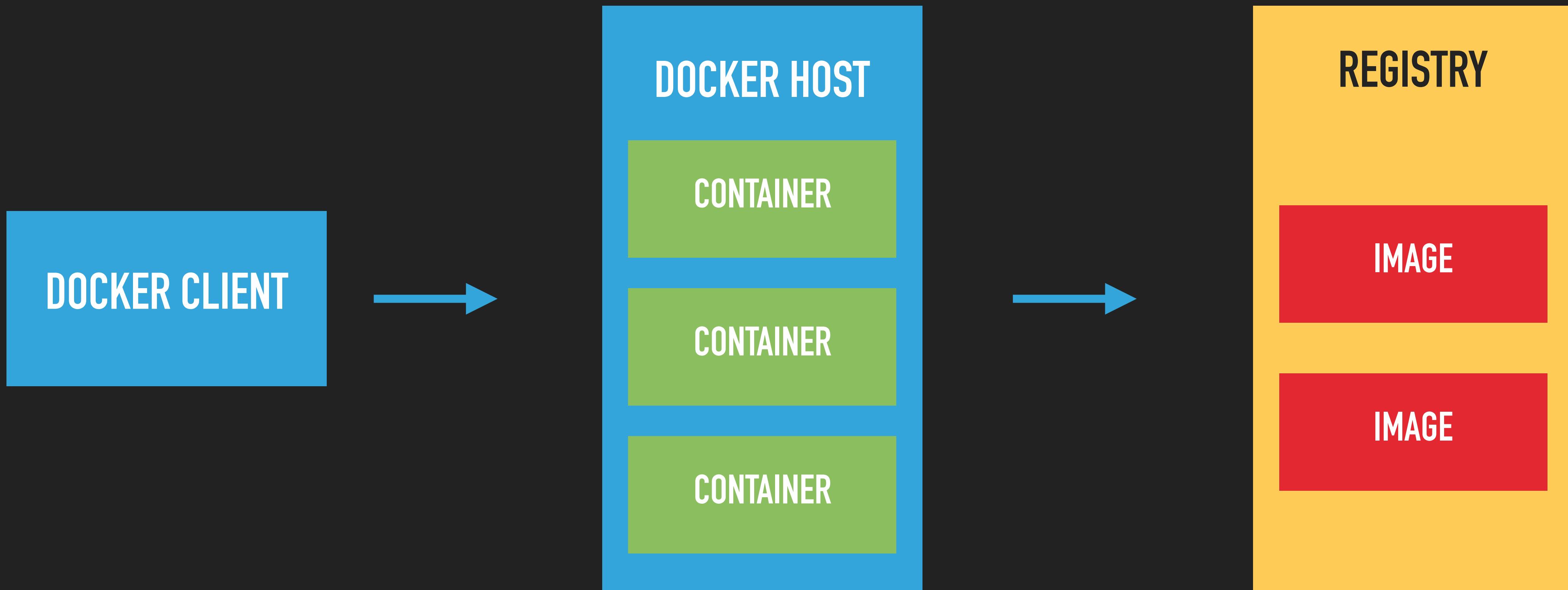
- ▶ Offers a one stop Docker setup for non-Linux machines
- ▶ Easiest way to get started on Mac and Windows
- ▶ Includes everything covered in these slides

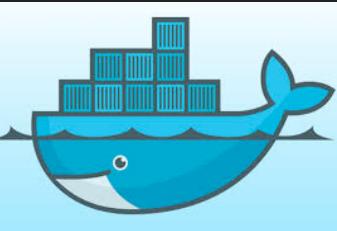


<https://www.docker.com/docker-toolbox>



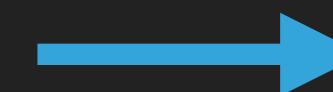
DOCKER IN A NUTSHELL





IMAGES AND CONTAINERS

IMAGE
ALPINE:3.1

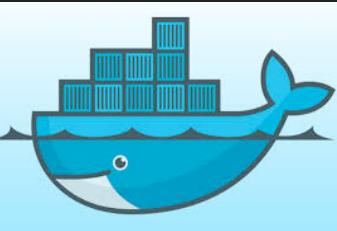


CONTAINER: DREAMY_HUGLE
B00F46831AF3

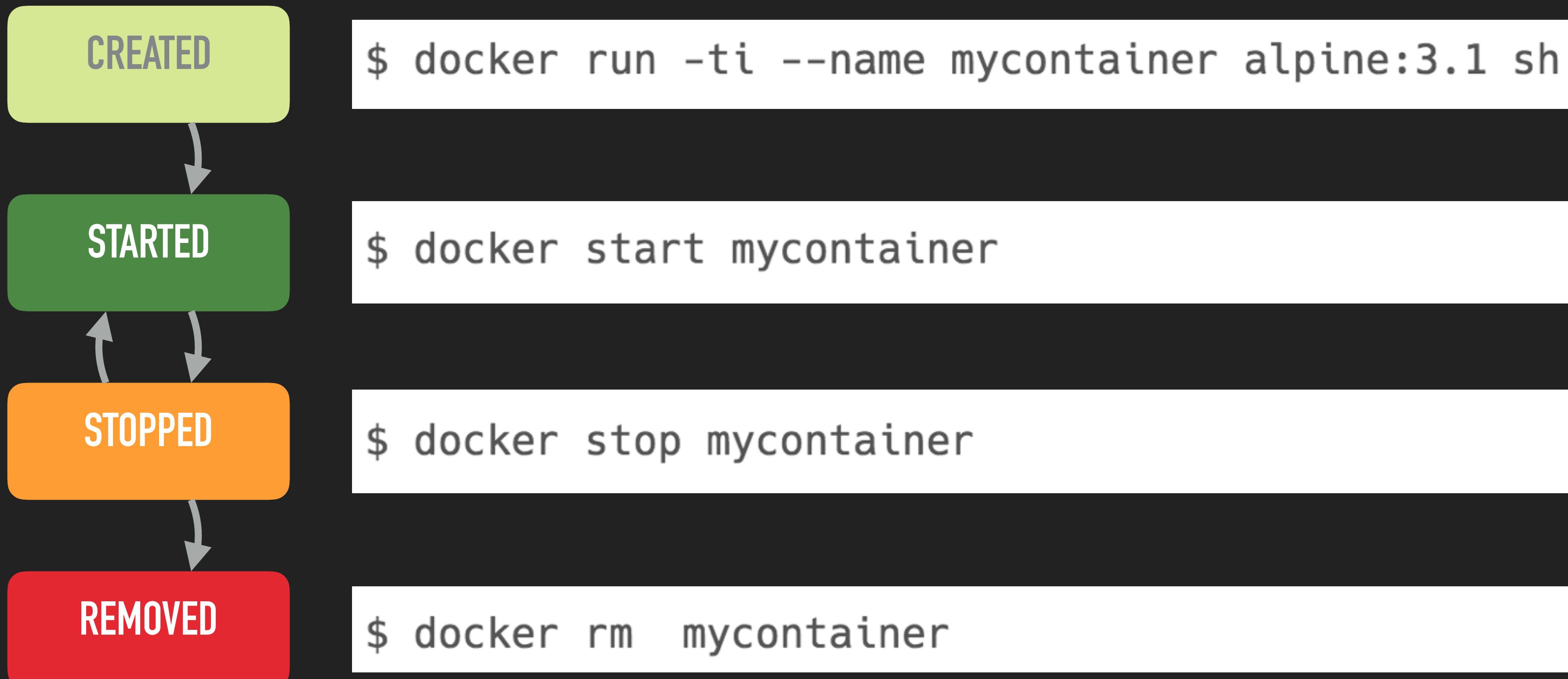
CONTAINER: SLEEPY_GOLICK
F0A1172ADB12

CONTAINER: JOLLY_BABBAGE
DOBAE89AD214

```
$ docker run alpine:3.1 echo "Hello World"  
Hello World
```

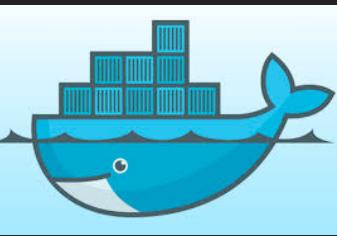


CONTAINER LIFECYCLE



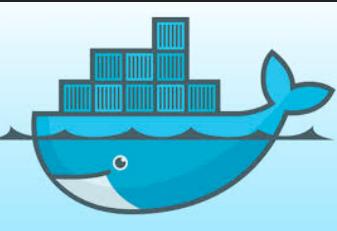
BUILDING A LOCAL TWITTER

- ▶ Bootstrap demo
- ▶ Basic container usage



EXPOSING SERVICES

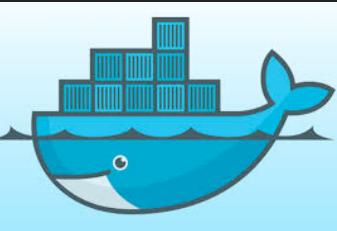
```
$ docker run -ti --rm=true nginx  
  
$ curl http://$(docker-machine ip default)  
Gateway Timeout: can't connect to remote host
```



CONTAINER PORTS ARE NOT VISIBLE OUTSIDE OF THE DOCKER HOST

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
b5344700e038	nginx	"nginx -g 'daemon off'"	80/tcp, 443/tcp

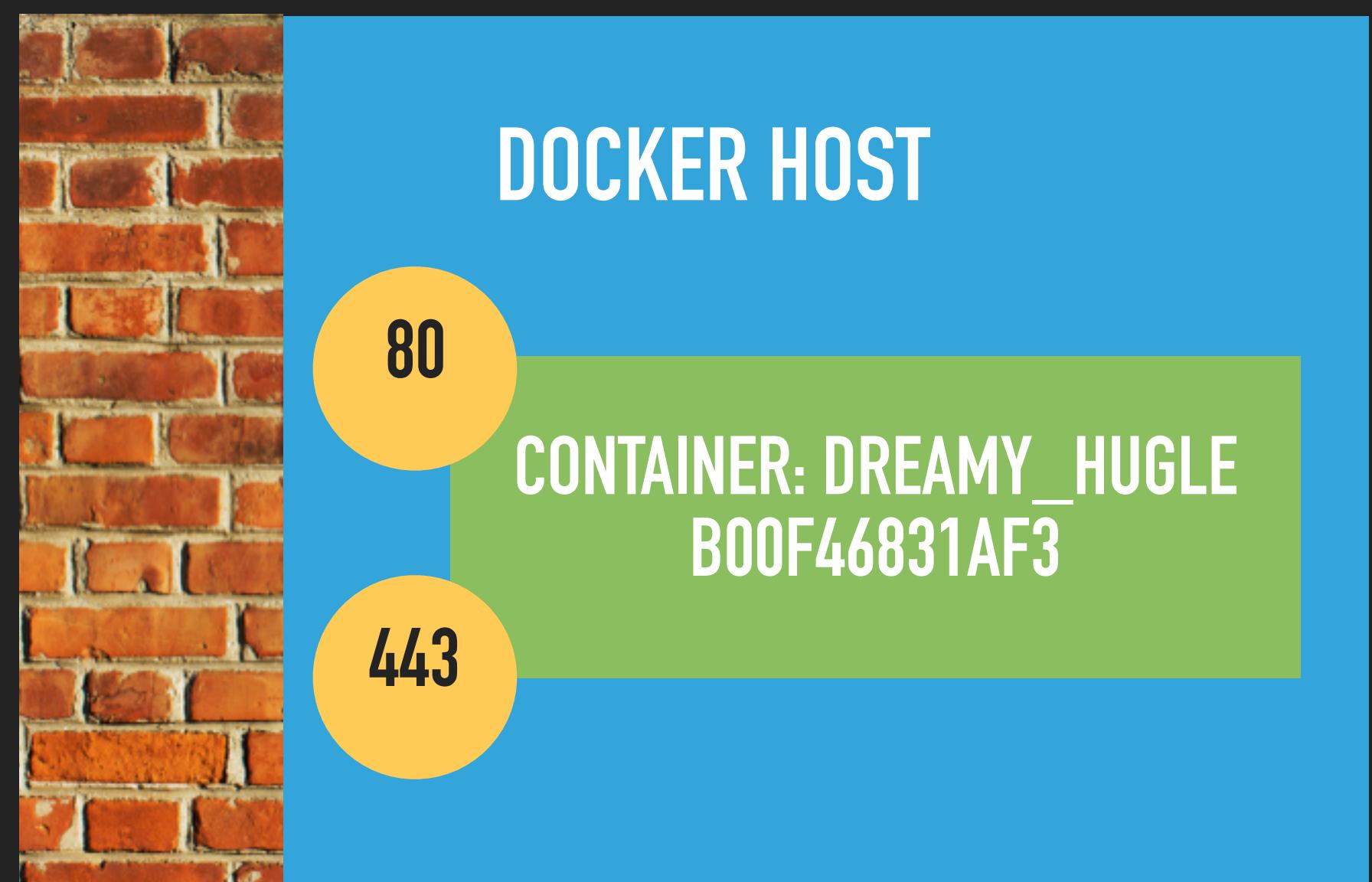


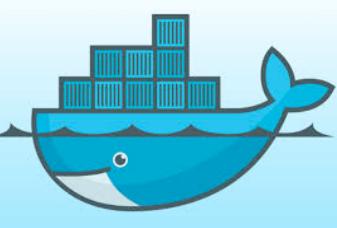
CONTAINER PORTS ARE NOT VISIBLE OUTSIDE OF THE DOCKER HOST

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND
b5344700e038	nginx	"nginx -g 'daemon off'"

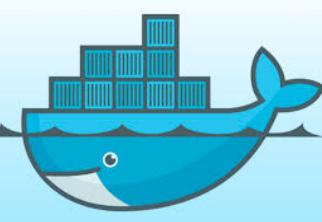
CREATED	STATUS	PORTS
...	...	80/tcp, 443/tcp





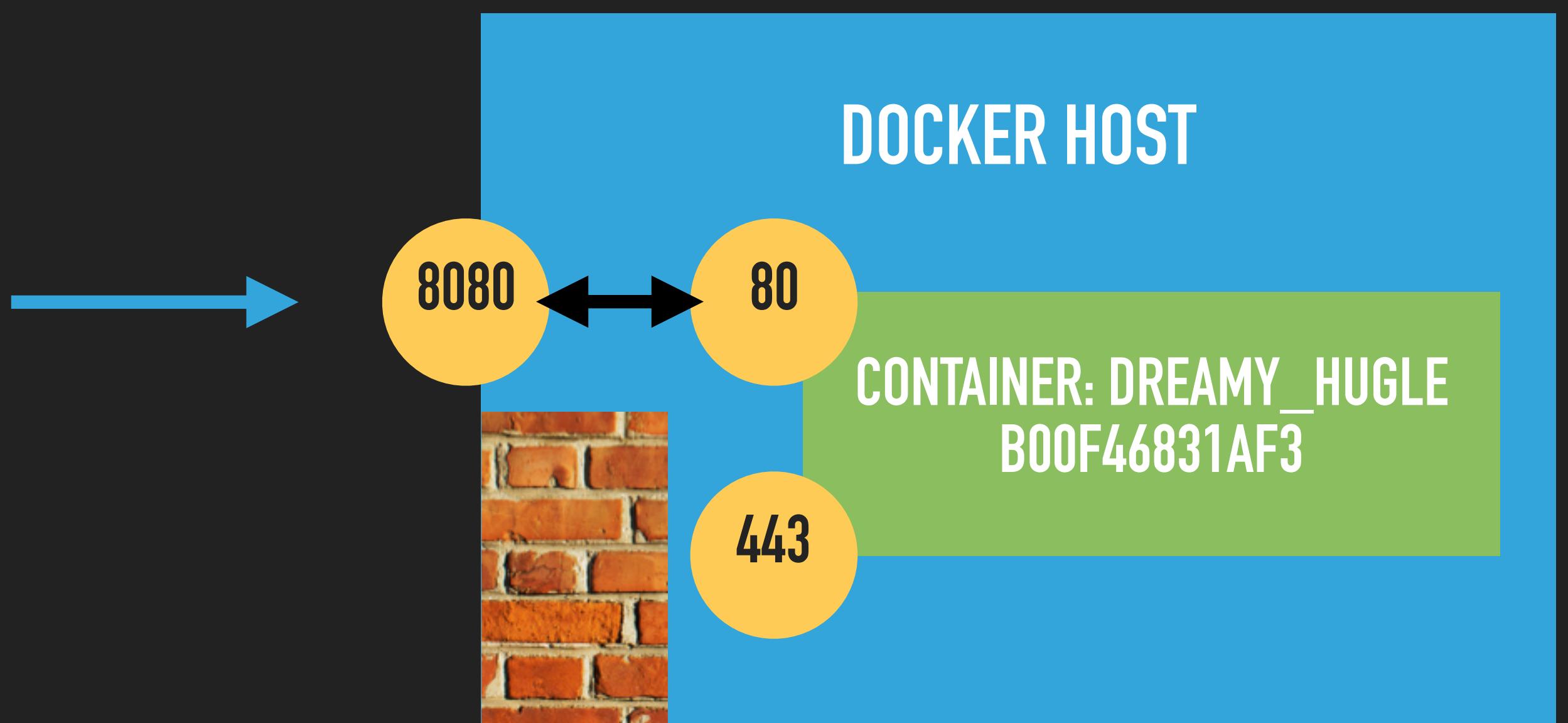
CONTAINER PORTS MUST BE MAPPED TO EXTERNALLY VISIBLE PORTS

```
$ docker run -ti --rm=true -p 8080:80 nginx
```



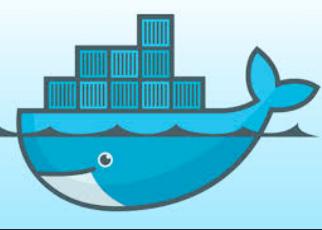
CONTAINER PORTS MUST BE MAPPED TO EXTERNALLY VISIBLE PORTS

```
$ docker run -ti --rm=true -p 8080:80 nginx
```



BUILDING A LOCAL TWITTER

- ▶ ~~Bootstrap demo~~
- ▶ ~~Basic container usage~~
- ▶ Running nginx
- ▶ Exposing ports



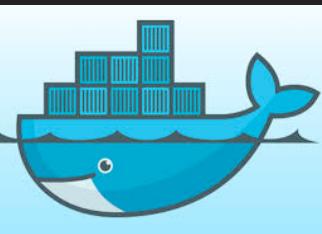
THE CONTAINER FILESYSTEM IS VERSIONED

```
$ docker run -ti --name=test alpine:3.1 sh
```

CONTAINER: TEST
FOA1172ADB12

CREATED

STARTED



THE CONTAINER FILESYSTEM IS VERSIONED

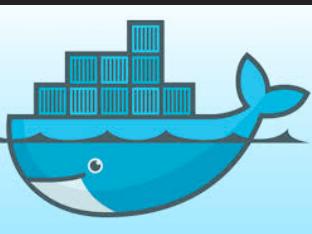
```
/ # echo I was here > foo  
/ # cat foo  
I was here
```

CONTAINER: TEST
FOA1172ADB12



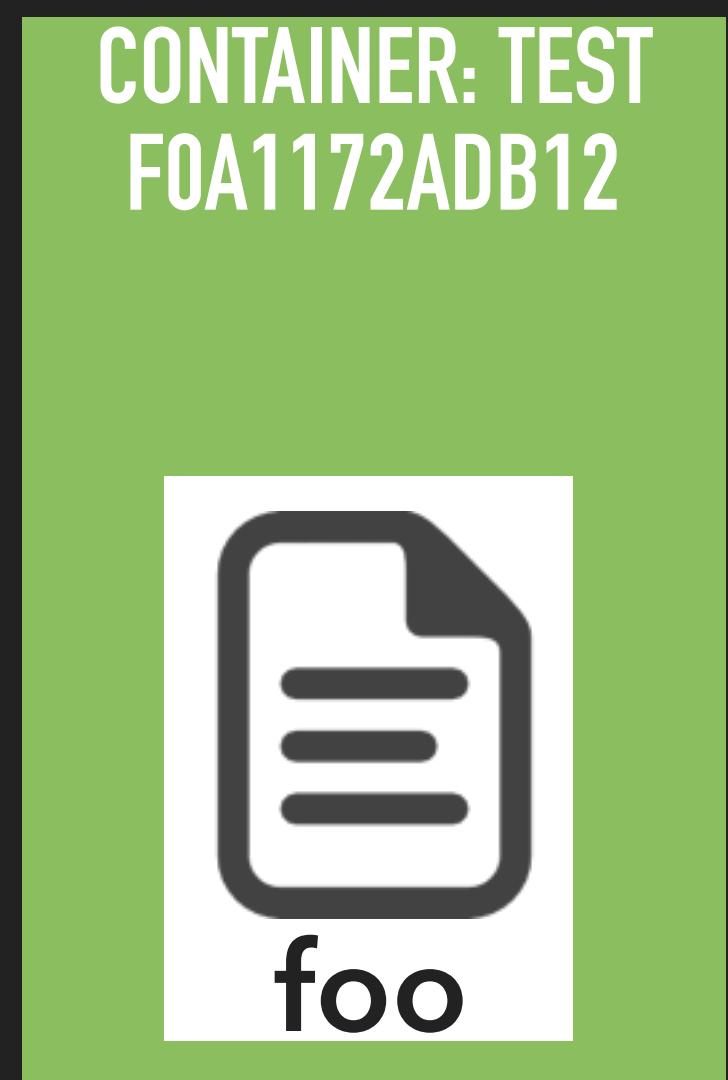
CREATED

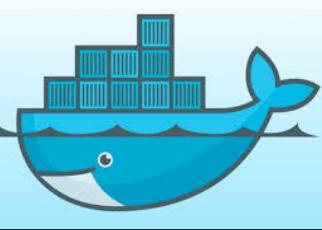
STARTED



THE CONTAINER FILESYSTEM IS VERSIONED

```
$ docker diff test
A /foo
C /root
A /root/.ash_history
```





THE CONTAINER FILESYSTEM SURVIVES RESTARTS

```
$ docker start test  
$ docker attach test
```

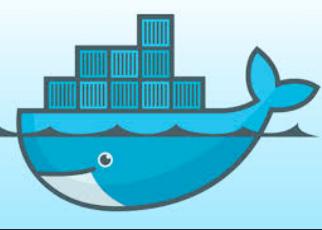
CONTAINER: TEST
FOA1172ADB12



STARTED

STOPPED





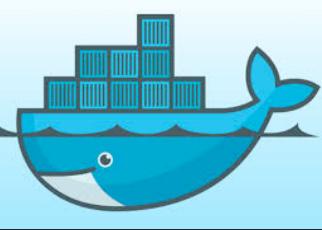
THE CONTAINER FILESYSTEM SURVIVES RESTARTS

```
/ # cat foo  
I was here
```

CONTAINER: TEST
FOA1172ADB12



STARTED



THE CONTAINER FILESYSTEM IS VERSIONED

```
$ docker run -ti --name=test alpine:3.1 sh
```

```
/ # echo I was here > foo  
/ # cat foo  
I was here
```

```
$ docker diff test  
A /foo  
C /root  
A /root/.ash_history
```

CONTAINER: TEST
FOA1172ADB12



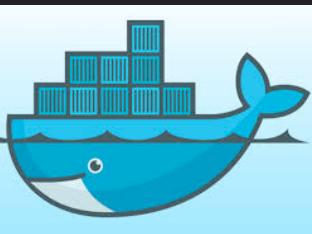
CREATED

STARTED

STOPPED

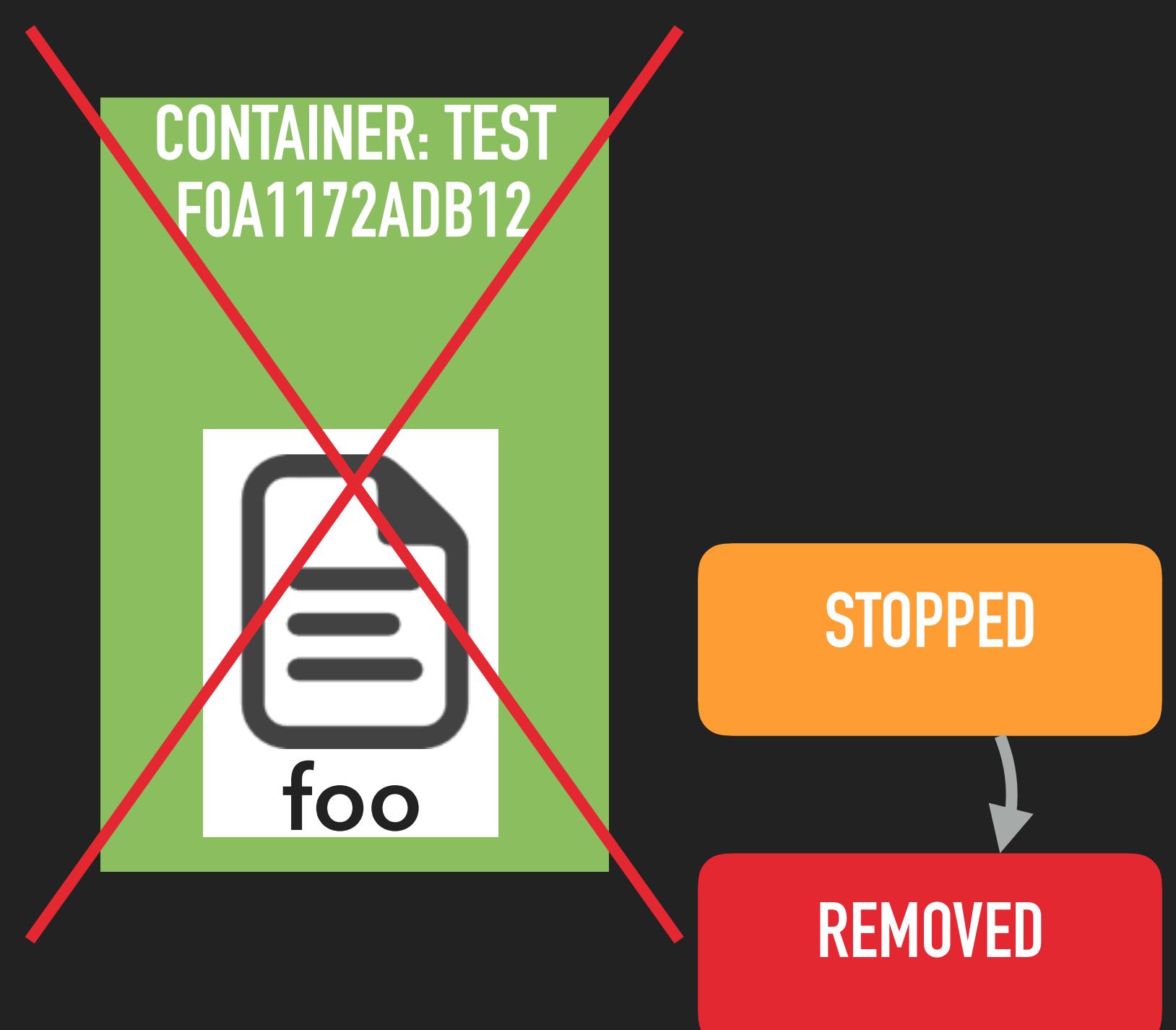
REMOVED

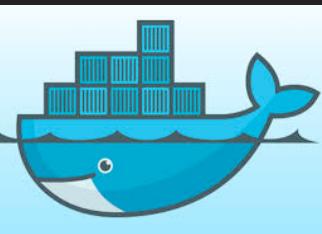




THE CONTAINER FILESYSTEM IS TRANSIENT

```
$ docker rm test
```





THE CONTAINER FILESYSTEM IS TRANSIENT

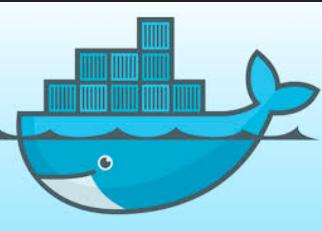
```
$ docker run -ti --name=test alpine:3.1 sh
```

CONTAINER: TEST
D8C89E9BB6AB

CREATED

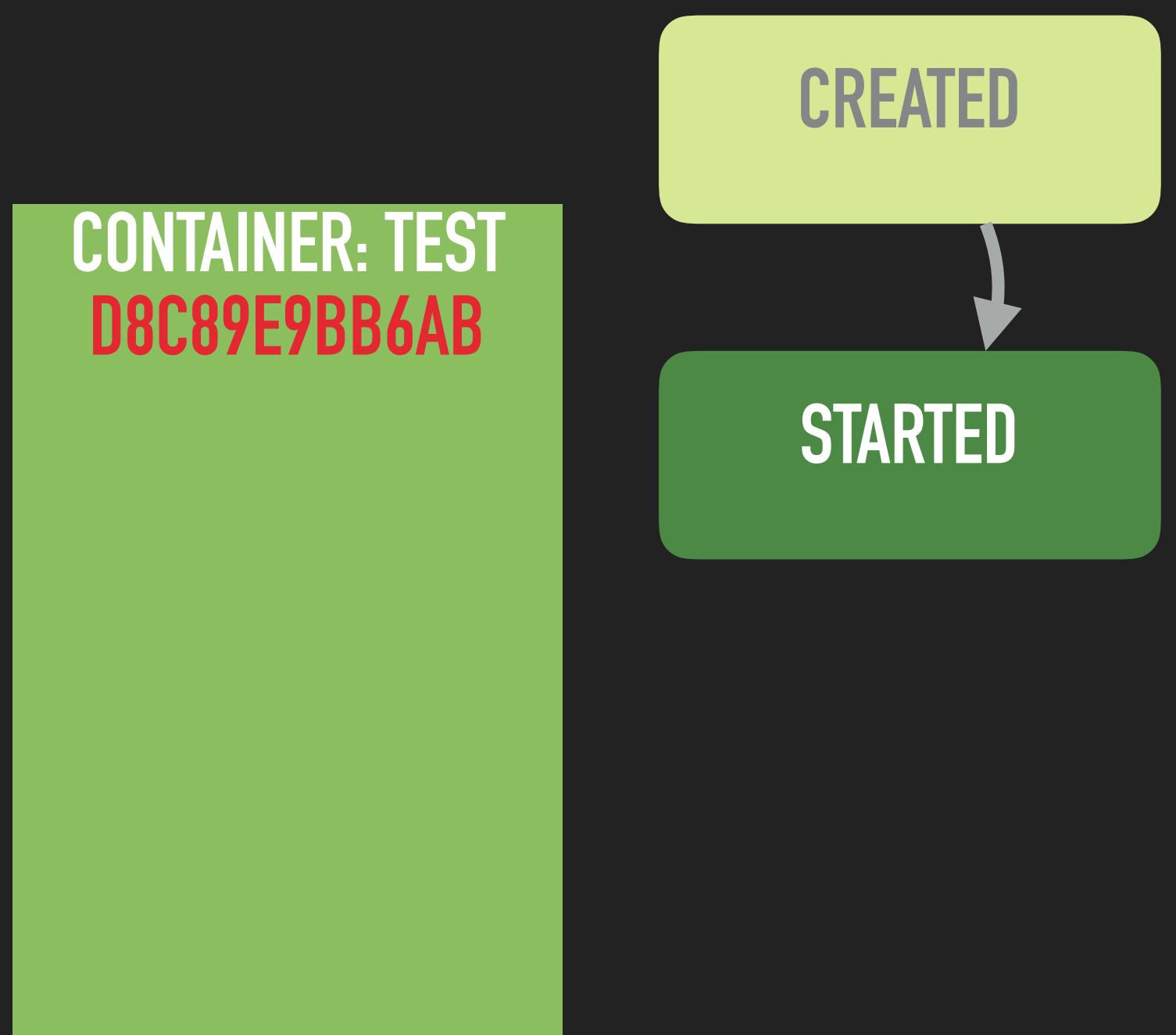
STARTED

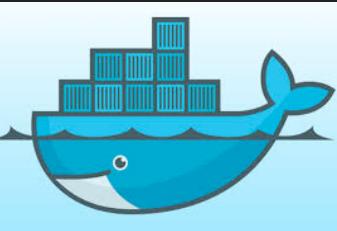




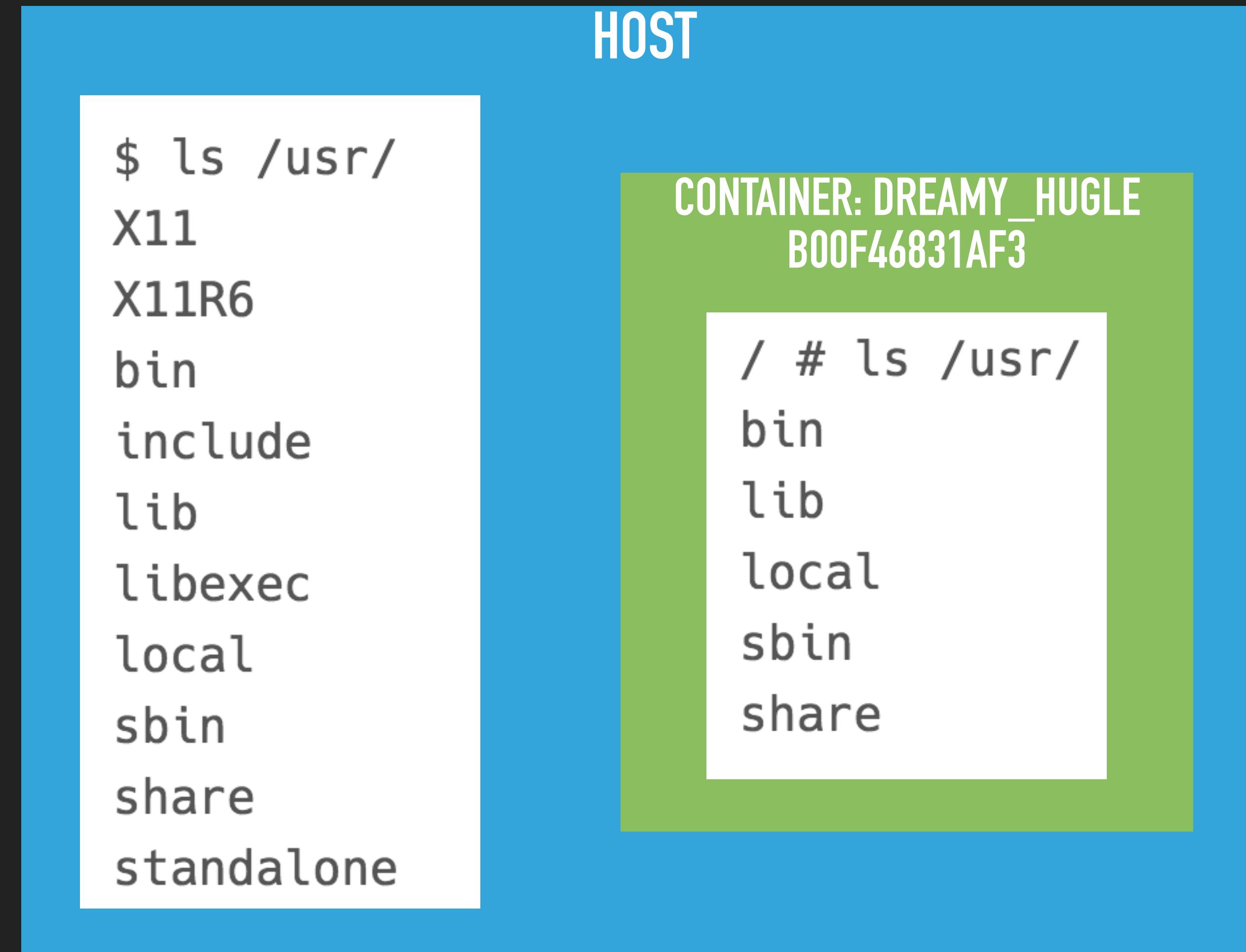
THE CONTAINER FILESYSTEM IS TRANSIENT

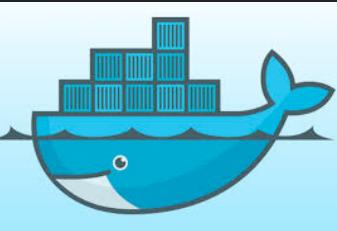
```
/ # cat foo  
cat: can't open 'foo': No such file or directory
```



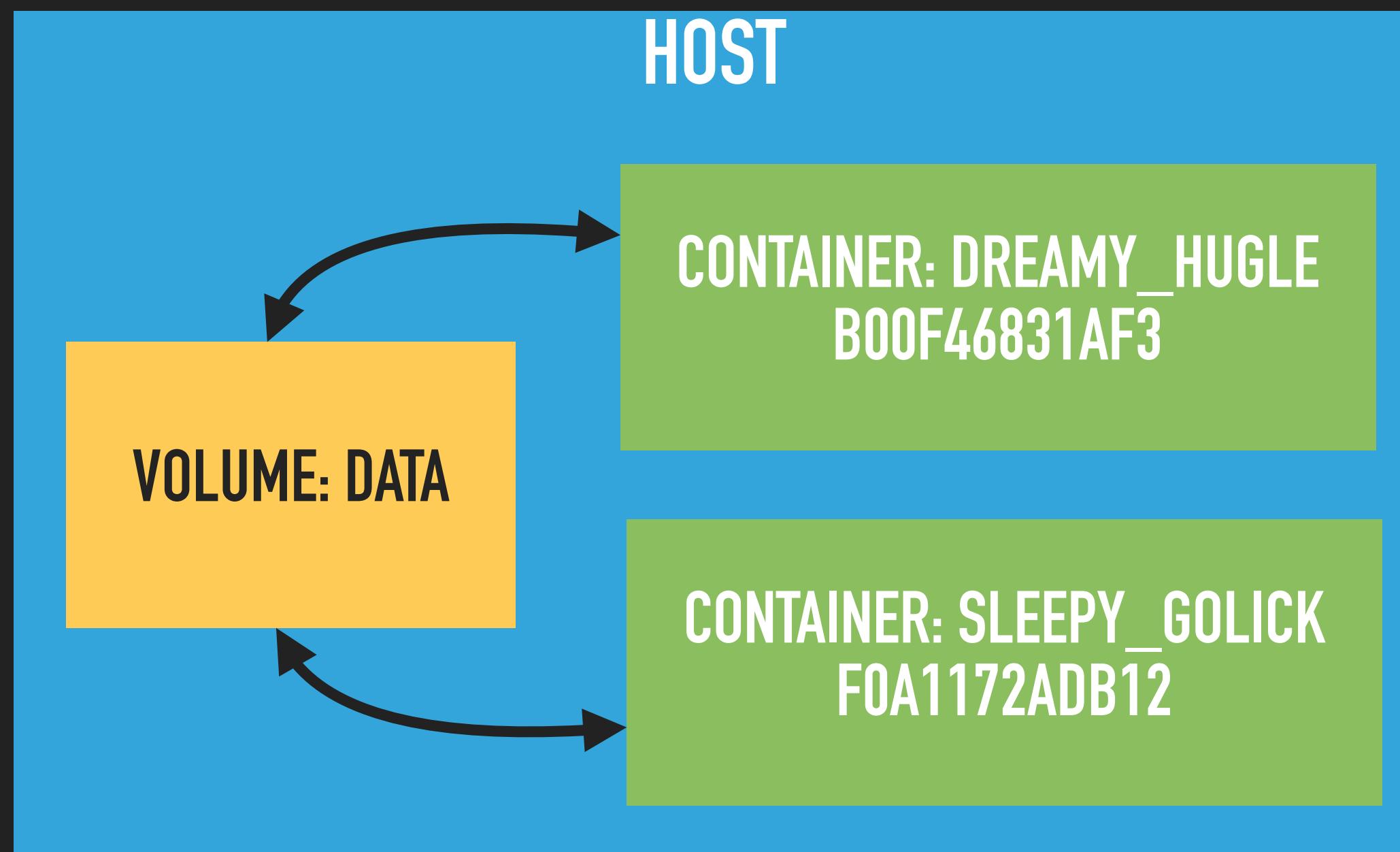


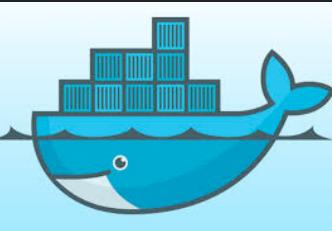
THE CONTAINER FILESYSTEM IS SEPARATED FROM THE HOST ENVIRONMENT





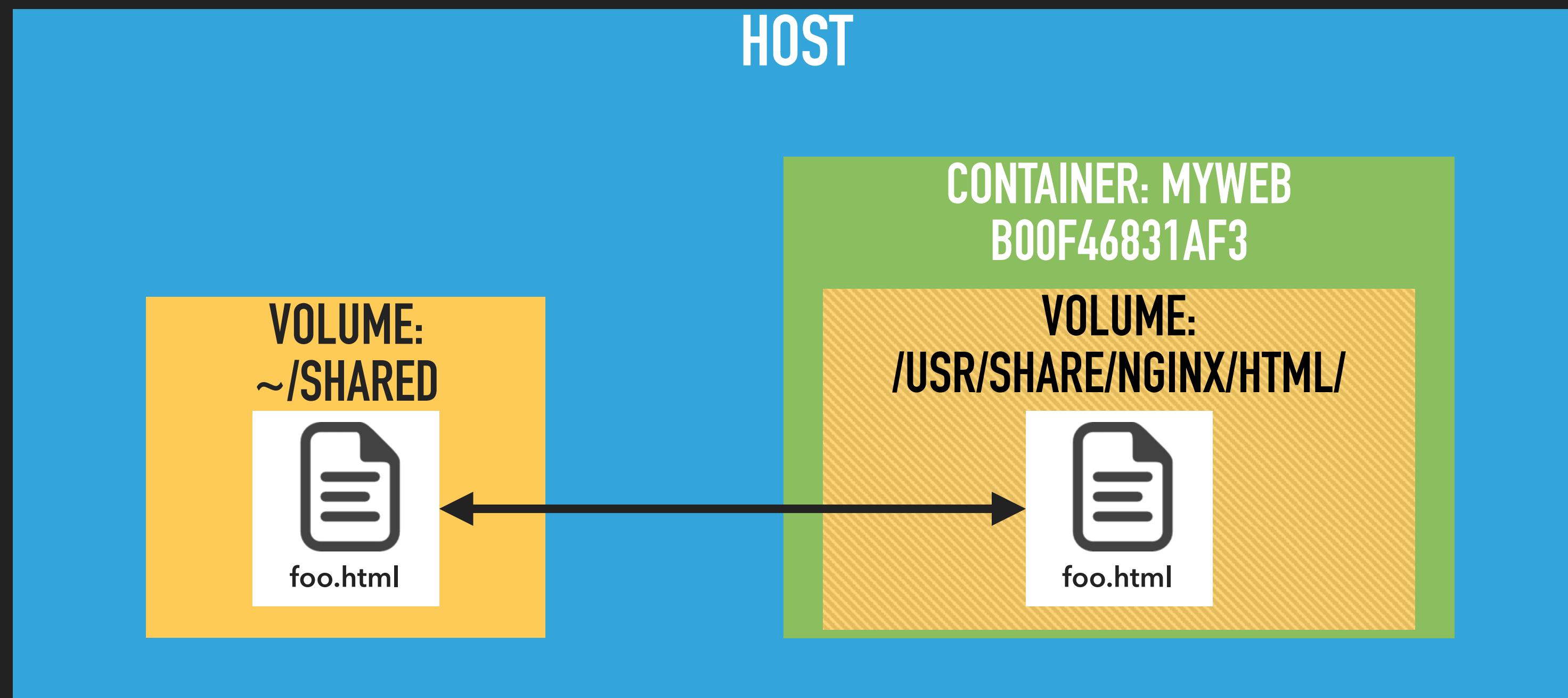
PERSISTENT VOLUMES ARE USED TO SHARE DATA BETWEEN HOST AND CONTAINER





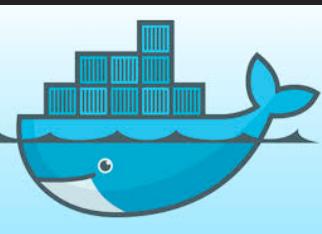
PERSISTENT VOLUMES ARE USED TO SHARE DATA BETWEEN HOST AND CONTAINER

```
$ mkdir ~/shared/  
$ docker run --name=myweb --rm=true -p 8080:80 \  
    -v ~/shared/:/usr/share/nginx/html/ nginx  
$ echo Hello World > ~/shared/foo.html
```



BUILDING A LOCAL TWITTER

- ▶ ~~Bootstrap demo~~
- ▶ ~~Basic container usage~~
- ▶ ~~Running nginx~~
- ▶ ~~Exposing ports~~
- ▶ Sharing volumes and data
- ▶ Monitoring



THE DOCKERFILE IS THE RECIPE FOR AN IMAGE

```
$ docker build -t koenighotze/jdk8 .
```

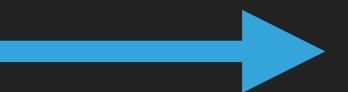
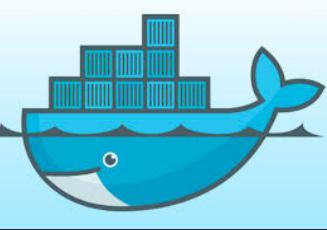


IMAGE
KOENIGHOTZE/JDK8

Dockerfile

OVERVIEW: DOCKER AND THE DOCKER ECOSYSTEM



```
FROM gliderlabs/alpine:3.3
MAINTAINER David Schmitz <koenighotze@yahoo.com>

ENV REFRESHED_AT 2015-12-27

ENV LANG en_US.UTF-8
ENV LANGUAGE en_US:en
ENV LC_ALL en_US.UTF-8

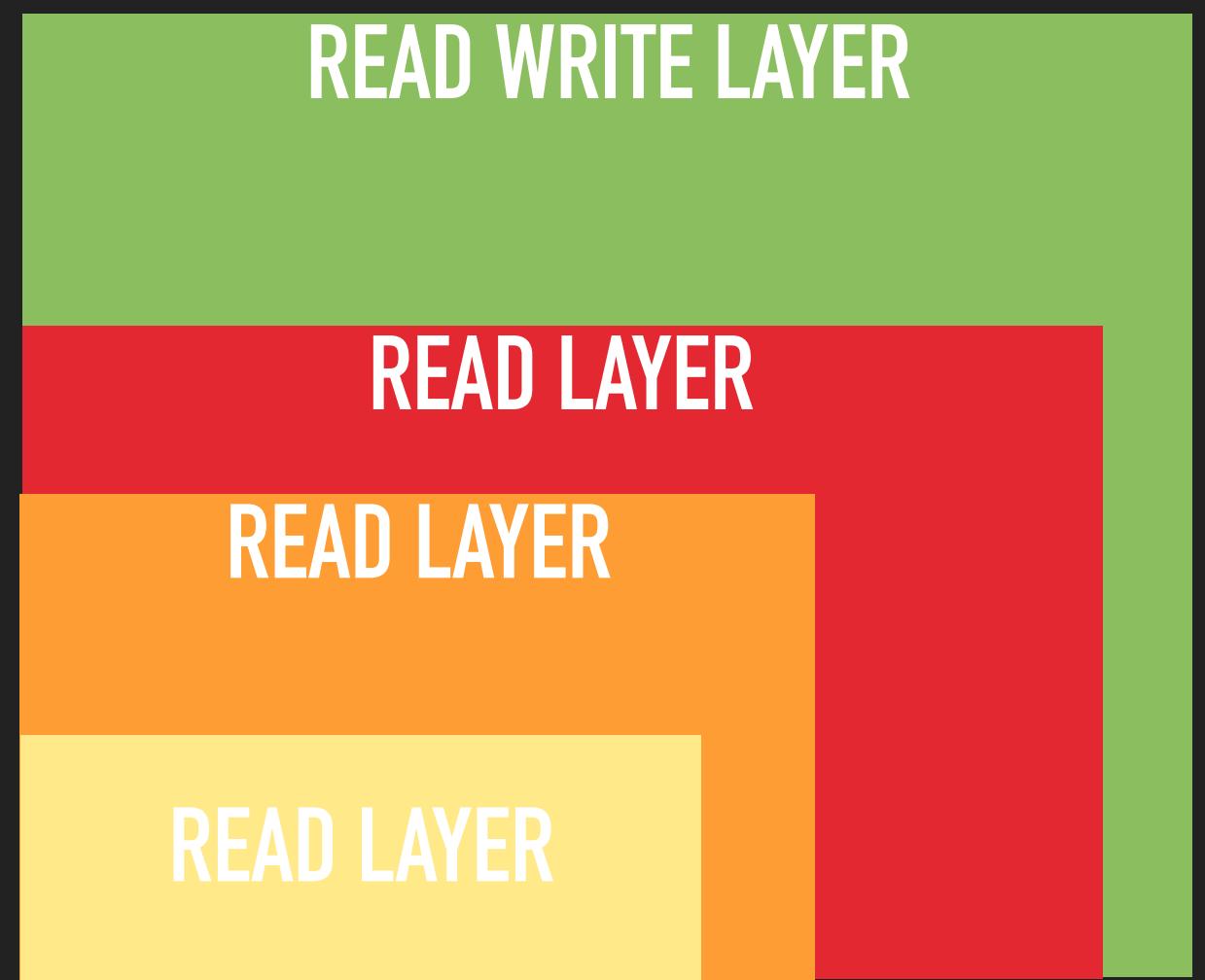
RUN apk-install bash \
    ca-certificates \
    openjdk8

COPY prompt.sh /etc/profile.d/

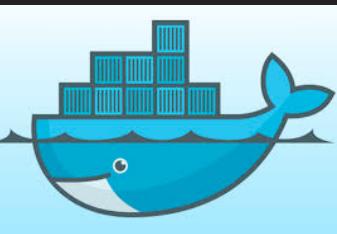
ENV JAVA_HOME /usr/lib/jvm/default-jvm

ENV SRC_VOLUME /volumes/share
VOLUME [$SRC_VOLUME]

CMD [ "bash", "-l" ]
```



OVERVIEW: DOCKER AND THE DOCKER ECOSYSTEM



```
FROM gliderlabs/alpine:3.3
MAINTAINER David Schmitz <koenighotze@yahoo.com>

ENV REFRESHED_AT 2015-12-27

ENV LANG en_US.UTF-8
ENV LANGUAGE en_US:en
ENV LC_ALL en_US.UTF-8

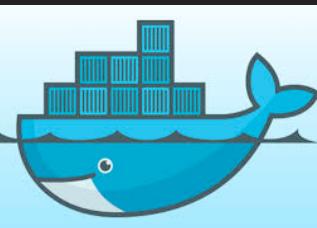
RUN apk-install bash \
    ca-certificates \
    openjdk8

COPY prompt.sh /etc/profile.d/

ENV JAVA_HOME /usr/lib/jvm/default-jvm

ENV SRC_VOLUME /volumes/share
VOLUME [$SRC_VOLUME]

CMD ["bash", "-l"]
```



IMAGES ARE LAYERED; EACH LAYER IS CACHED OR Fetched INDIVIDUALLY

```
$ docker history koenighotze/jdk8
```

IMAGE	CREATED	CREATED BY	SIZE
790ba5309f26	12 minutes ago	/bin/sh -c #(nop) CMD ["bash" "-l"]	0 B
82f472fb0916	12 minutes ago	/bin/sh -c #(nop) VOLUME [[/volumes/share]]	0 B
ec3034064c9e	12 minutes ago	/bin/sh -c #(nop) ENV SRC_VOLUME=/volumes/sha	0 B
0b53191f161a	12 minutes ago	/bin/sh -c #(nop) ENV JAVA_HOME=/usr/lib/jvm/	0 B
cf5edd73df14	12 minutes ago	/bin/sh -c #(nop) COPY file:7a1d0aabc19cb9513	116 B
f9648e9584d7	12 days ago	/bin/sh -c apk-install bash c	142.9 MB
aaf60f9e582d	2 weeks ago	/bin/sh -c #(nop) ENV LC_ALL=en_US.UTF-8	0 B
95f75268c863	2 weeks ago	/bin/sh -c #(nop) ENV LANGUAGE=en_US:en	0 B
bb224f27eb36	2 weeks ago	/bin/sh -c #(nop) ENV LANG=en_US.UTF-8	0 B
ba17a33dd7dd	2 weeks ago	/bin/sh -c #(nop) ENV REFRESHED_AT=2015-12-27	0 B
f79be35aabda	2 weeks ago	/bin/sh -c #(nop) MAINTAINER David Schmitz <k	0 B
24e7cde1dbe9	3 weeks ago	/bin/sh -c #(nop) ADD file:78d479146973ebc076	4.794 MB

KOENIGHOTZE/JDK8

CMD ["BASH", "-L"]

VOLUME [\$SRC_VOLUME]

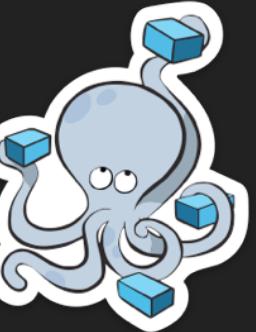
...

MAINTAINER DAVID SCHMITZ

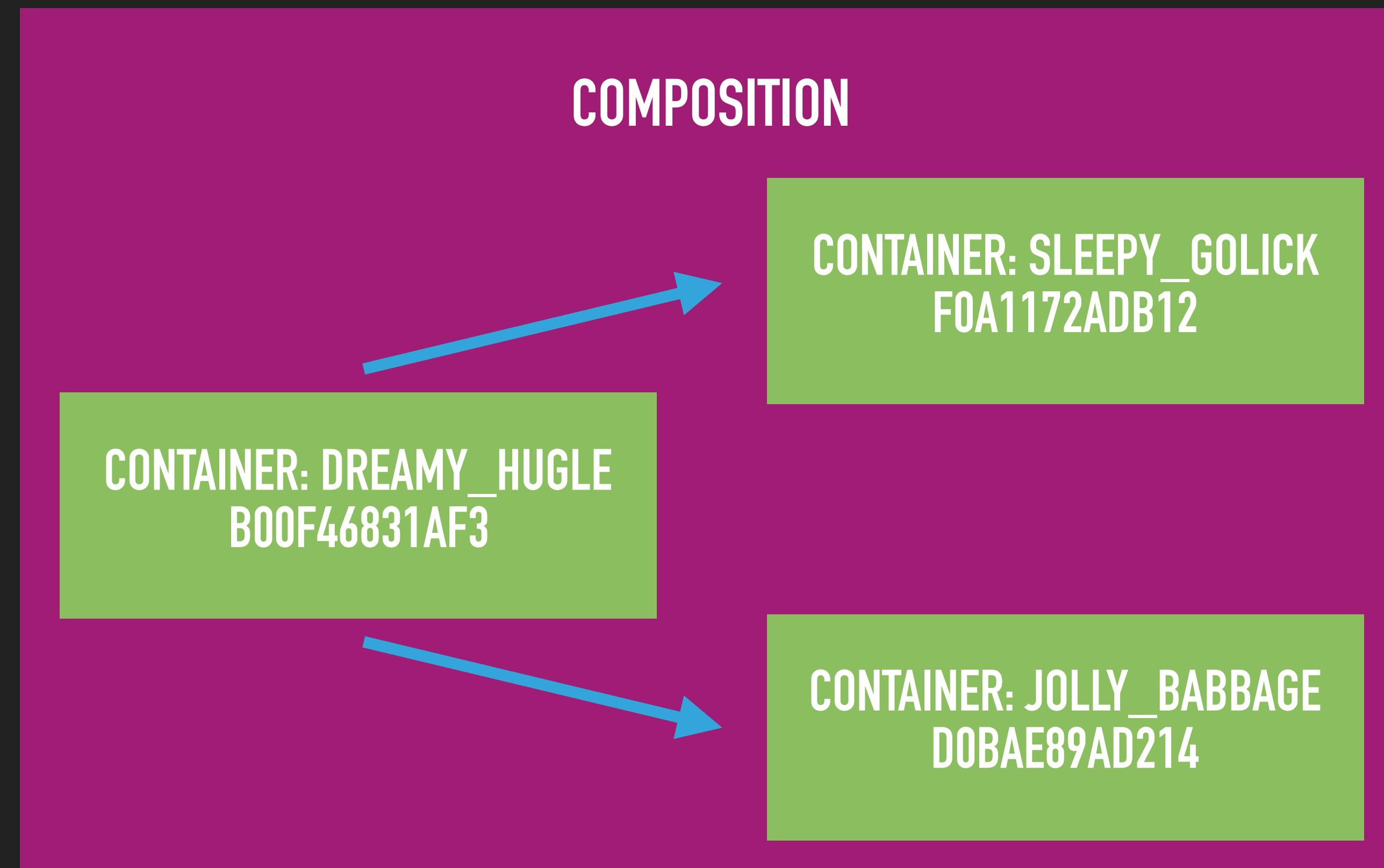
FROM GLIDERLABS/ALPINE:3.3

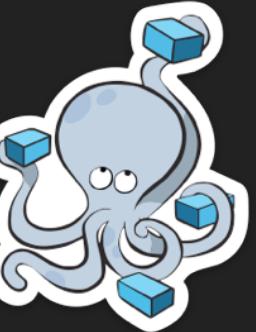


COMPLEX SCENARIOS USING COMPOSE



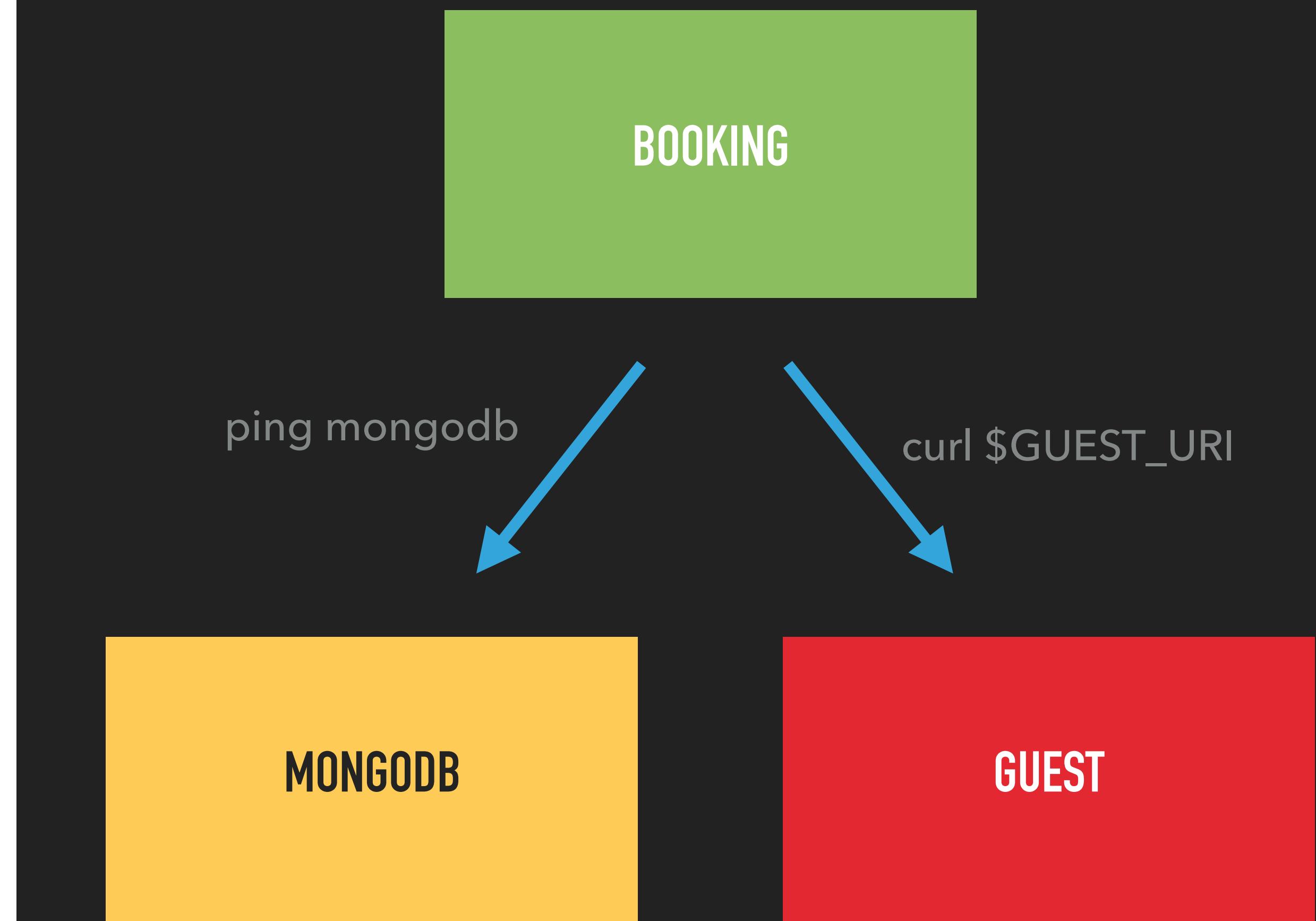
MANAGING DEPENDENCIES WITH DOCKER-COMPOSE

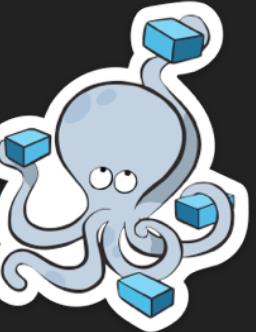




USING DOCKER-COMPOSE FOR DEPENDENCIES*

```
booking:  
  image: koenighotze/jee7hotelbooking  
  ports:  
    - 8080:8080  
    - 9990:9990  
  links:  
    - mongodb  
    - guest  
  environment:  
    - MONGO_URI=mongodb  
    - GUEST_URI=guest:8080  
guest:  
  image: koenighotze/jee7hotelguest  
  ports:  
    - 8081:8080  
  expose:  
    - 8080  
mongodb:  
  container_name: jee7hotelmongo  
  image: mongo:3.1.9  
  expose:  
    - "27017"
```





DOCKER SINGLE HOST NETWORKING USING LINKS

```
movie:  
  image: demo/movie  
  ports:  
    - 8082:8080  
  links:  
    - actor  
actor:  
  image: demo/actor  
  ports:  
    - 8081:8080  
  links:  
    - movie  
cinema:  
  image: demo/cinema  
  ports:  
    - 8080:8080  
  links:  
    - movie
```

MOVIE CONTAINER

MOVIE.WAR

```
Client client = newClient();  
WebTarget target = client.target(new URI("http://actor:8080/actor"));  
JsonObject response = target.request(APPLICATION_JSON)  
    .get(JsonObject.class);
```

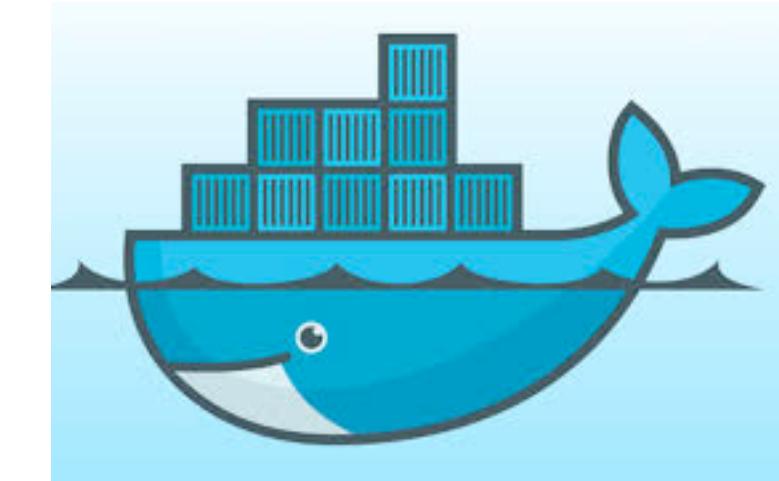
ACTOR CONTAINER

ACTOR.WAR

```
@Named  
@Path("actor")  
@Transactional  
public class ActorService {
```

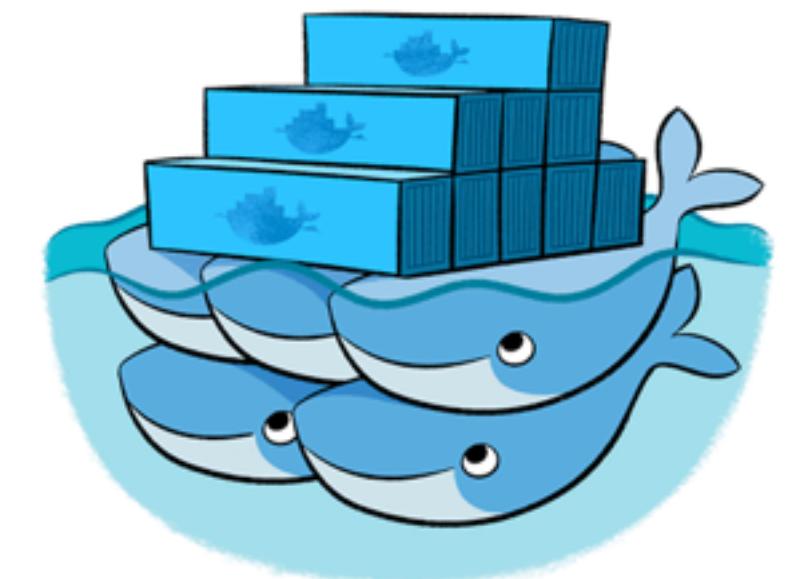
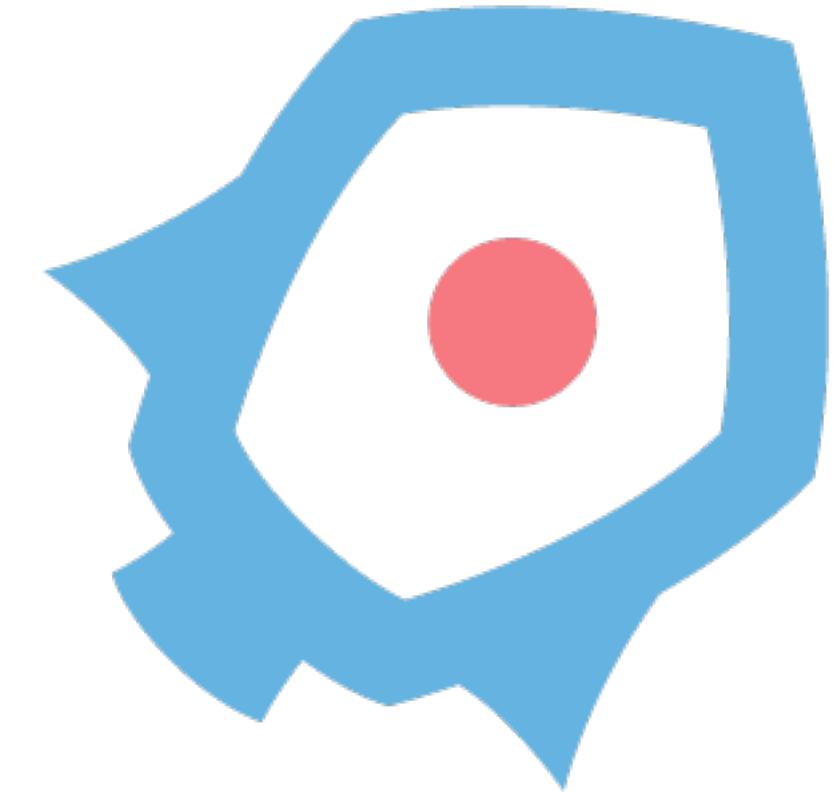
CREATING A SIMPLE COMPOSITION WITH SPRING-BOOT AND DOCKER

- ▶ Create simple web application
- ▶ Add database as container
- ▶ Create compositions file
- ▶ Add volumes



WHAT IS NEXT?

- ▶ CoreOs and rkt
- ▶ Docker Swarm and Kubernetes
- ▶ Unikernels (!)



SUMMARY

- ▶ Docker is production ready
- ▶ Docker-Compose is targeted towards developers and should *not* be used for production
- ▶ Might have some issues with regards to networking and clustering
- ▶ Multi-Cloud deployments possible
- ▶ Community is vital and eager to help

WANT TO KNOW MORE?

TRY ONE OF THESE GREAT VAULTS OF WISDOM

- ▶ Docker Home Page: <https://www.docker.com/>
- ▶ Docker Cheat Sheet: <https://github.com/wsargent/docker-cheat-sheet>
- ▶ James Turnbull, Dockerbook: <http://www.dockerbook.com/>

david.schmitz@senacor.com

THANK YOU!

<>

BACKUP

THE NEED FOR CONTAINERS

Loan Application

