INTRODUCTION FOR BUSY PEOPLE

DOCKER FOR THE BRAVE

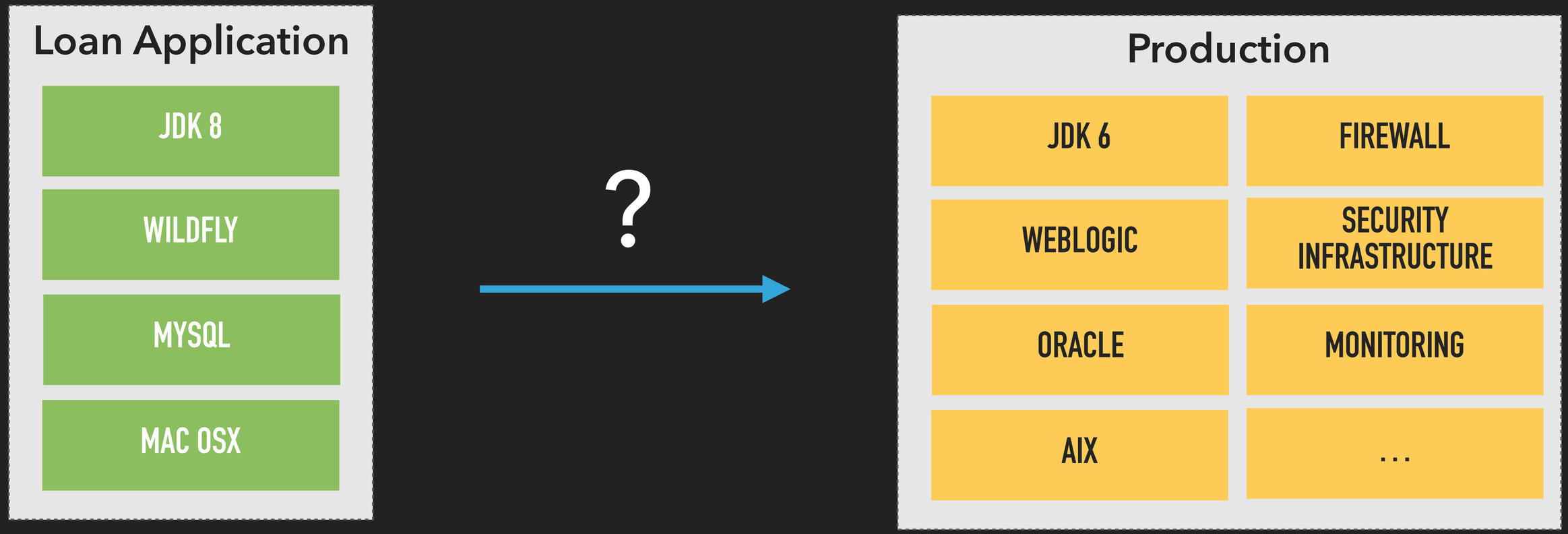
AGENDA

- 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

TRADITIONAL SOFTWARE DELIVERY SUFFERS FROM GAP BETWEEN DEV AND PROD



Magic Operations Guide Template

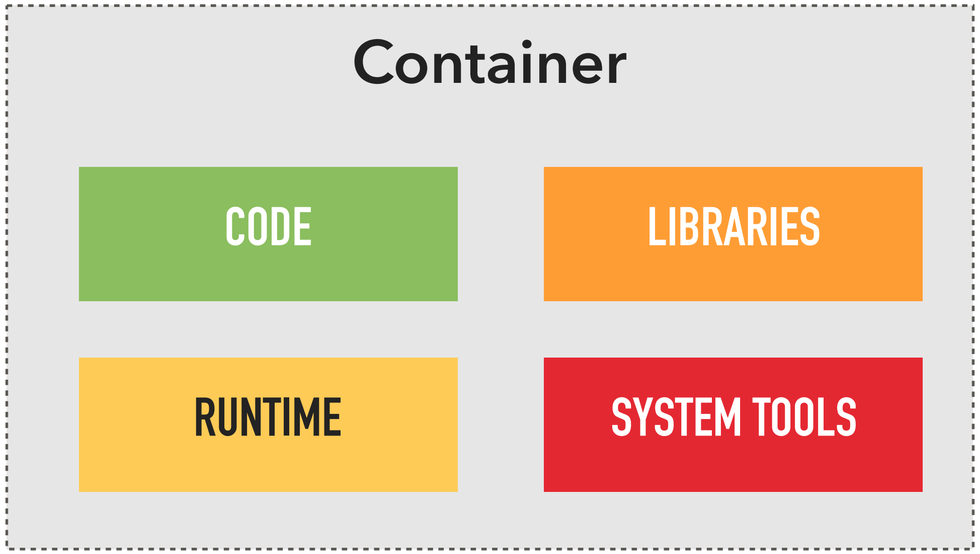

GOOD JOB
GOOD JOB
GOOD JOB

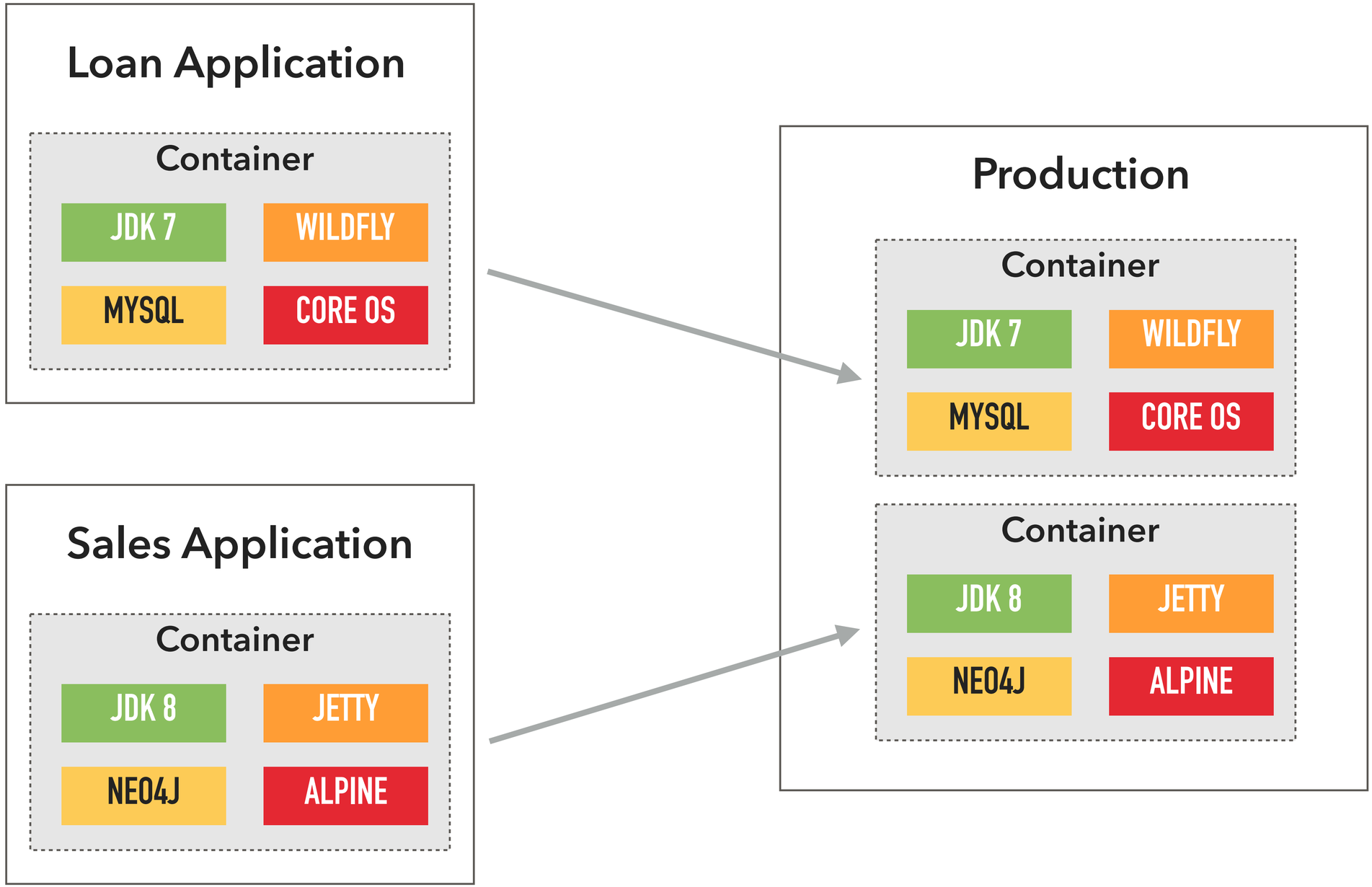
Unicorns do not exist!

CONTAINERIZED APPLICATIONS OFFER A WAY TO NARROW THE GAP

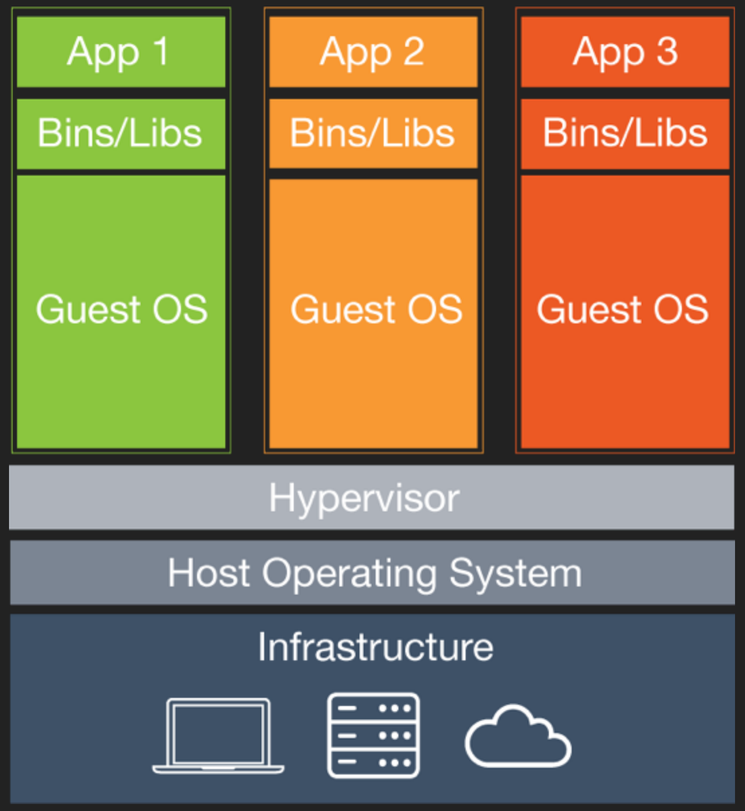
- Immutable containers reduce human errors
- Instead of “works on my machine” you deploy the machine
- Think diapers

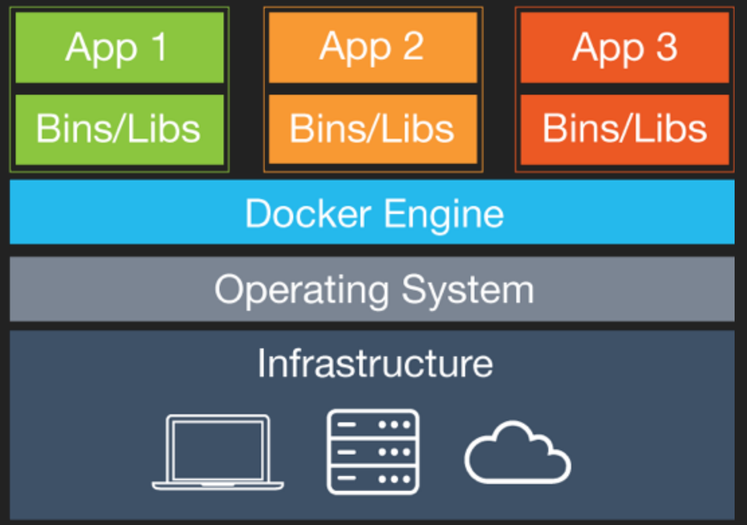
PACKAGE ONCE
DEPLOY ANYWHERE




Loan Application
Container
JDK 7
WILDFLY
MYSQL
CORE OS
Sales Application
Container
JDK 8
JETTY
NEO4J
ALPINE
Production
Container
JDK 7
WILDFLY
MYSQL
CORE OS
Container
JDK 8
JETTY
NEO4J
ALPINE

VIRTUALIZATION VS. CONTAINERS





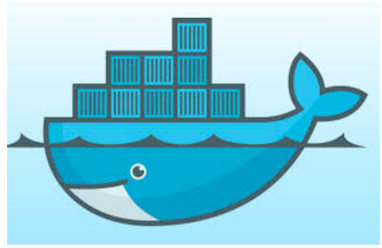
SPEED

FOOTPRINT

SIMPLICITY

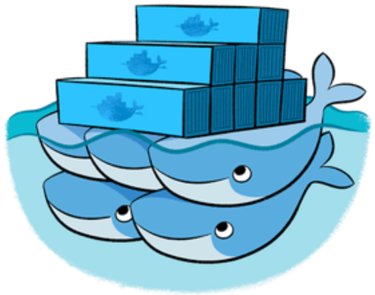
DOCKER BOOTCAMP

THE DOCKER ECOSYSTEM CONSISTS OF FIVE ESSENTIAL PARTS

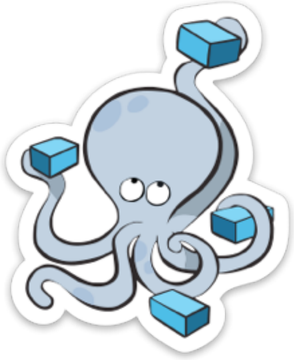


Docker

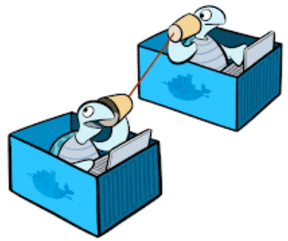
Machine



Swarm



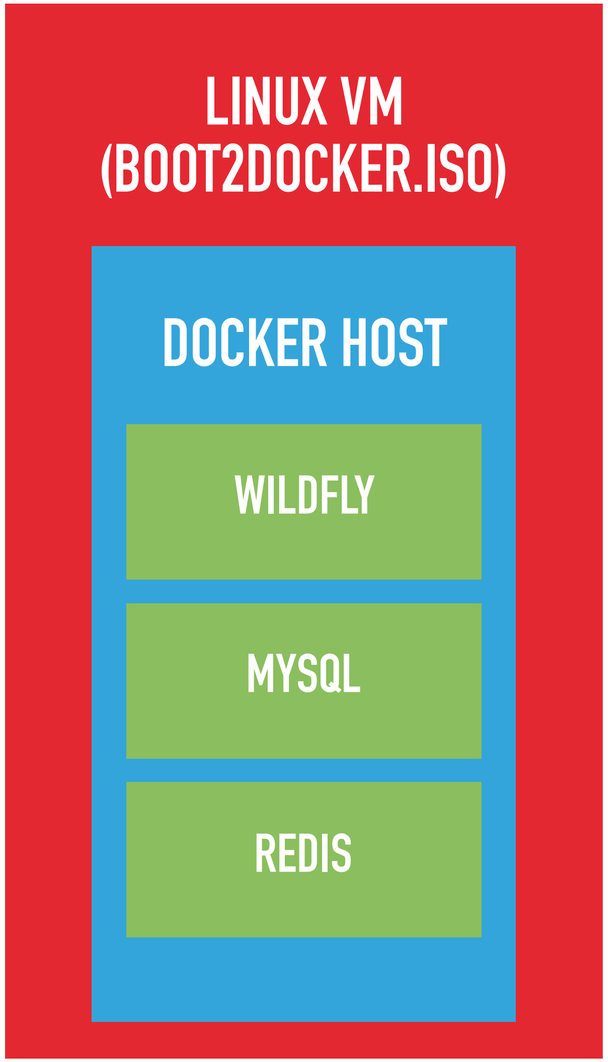
Compose



Network

DOCKER-MACHINE HELPS GETTING STARTED ON NON LINUX BOXES

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

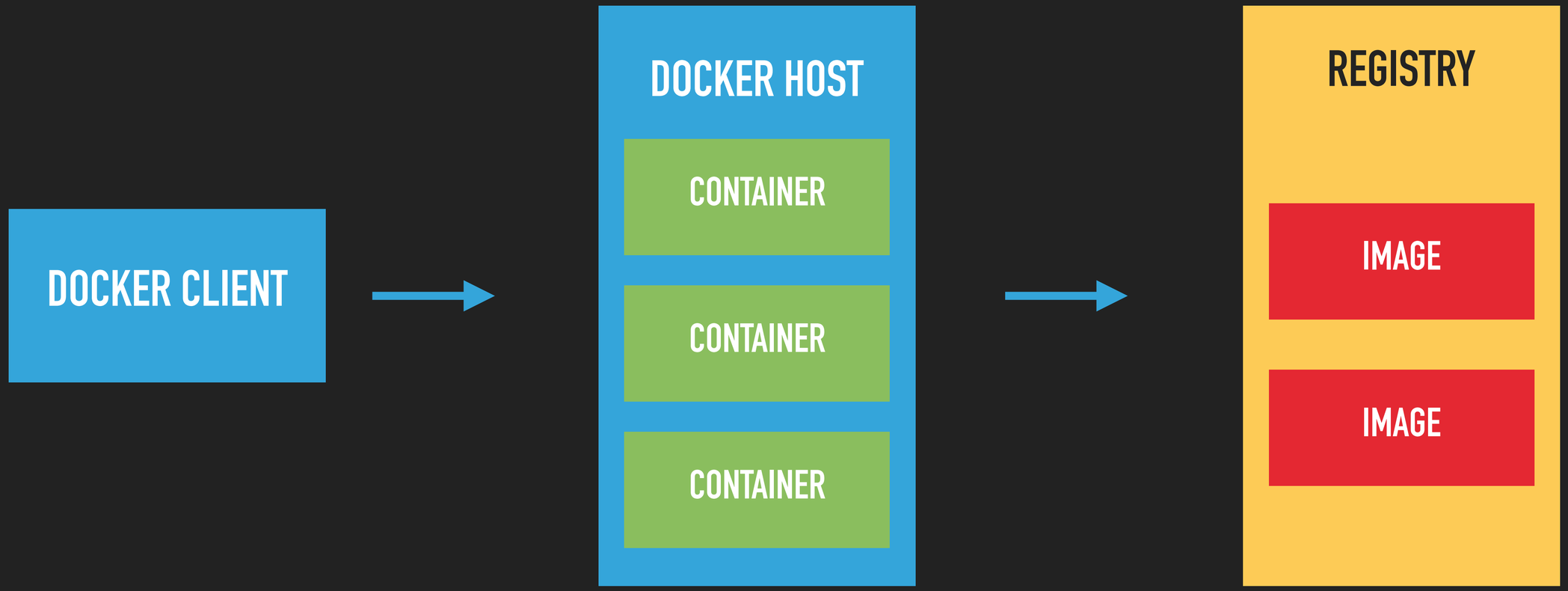


DOCKER TOOLBOX TO THE RESCUE

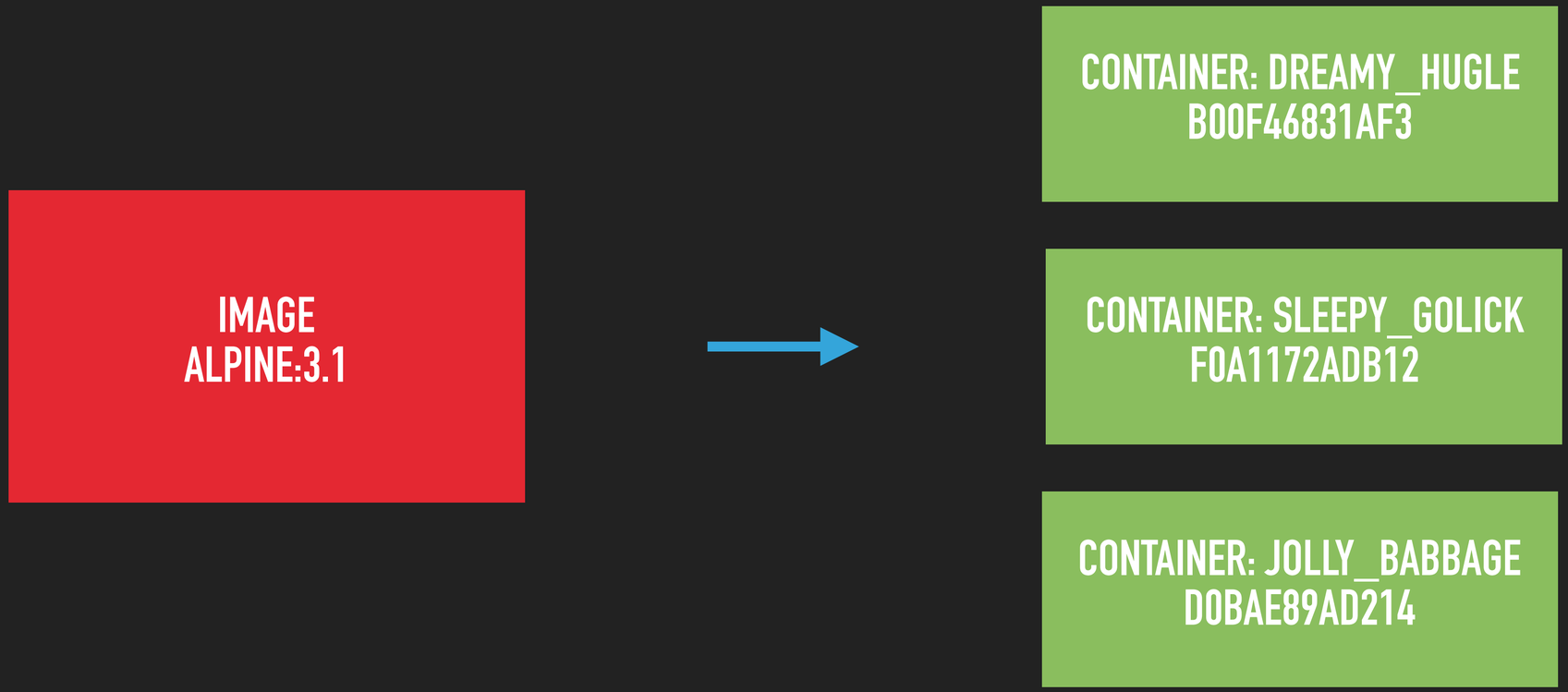
- Offers a one stop Docker setup for non-Linux machines
- Easiest way to get started on Mac and Windows
- Includes everything covered in theses slides

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

DOCKER IN A NUTSHELL

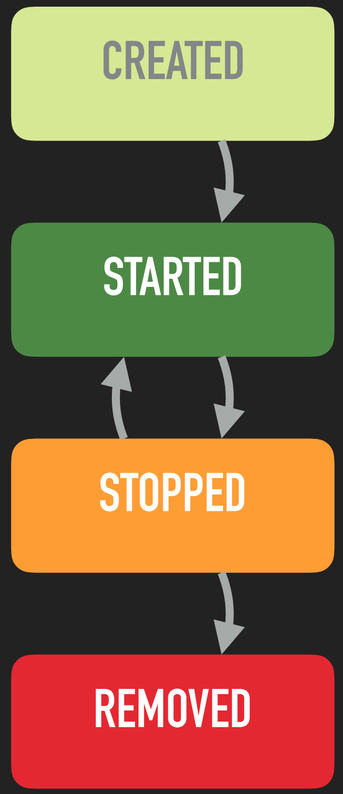


IMAGES AND CONTAINERS




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

CONTAINER LIFECYCLE




```
$ docker run -ti --name mycontainer alpine:3.1 sh
```

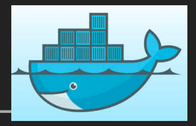
```
$ docker start mycontainer
```

```
$ docker stop mycontainer
```

```
$ docker rm  mycontainer
```

BUILDING A LOCAL TWITTER

- Bootstrap demo
- Basic container usage



EXPOSING SERVICES

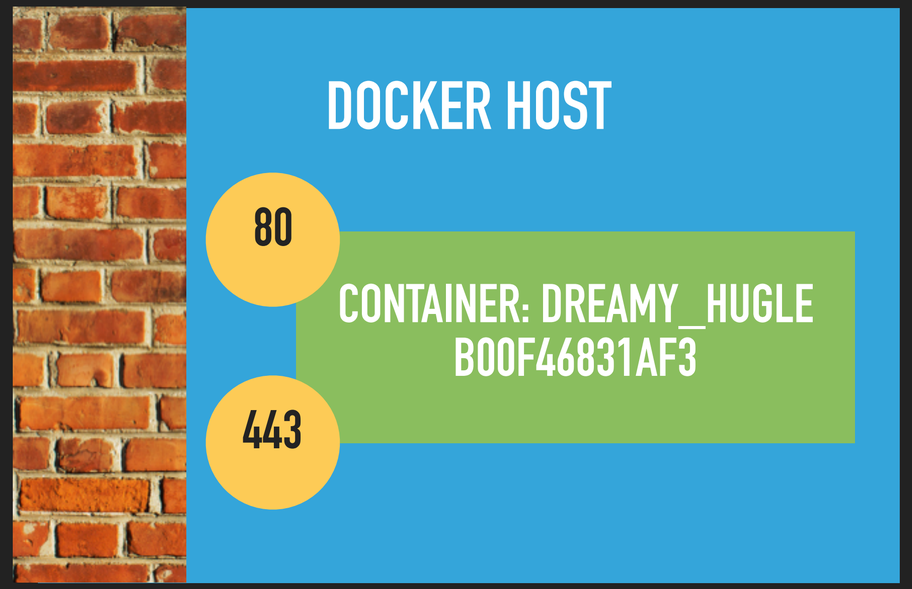
```
$ docker run -ti --rm=true nginx

$ 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                  CREATED   STATUS   PORTS
b5344700e038 nginx "nginx -g 'daemon off"   ...       ...      80/tcp, 443/tcp
```

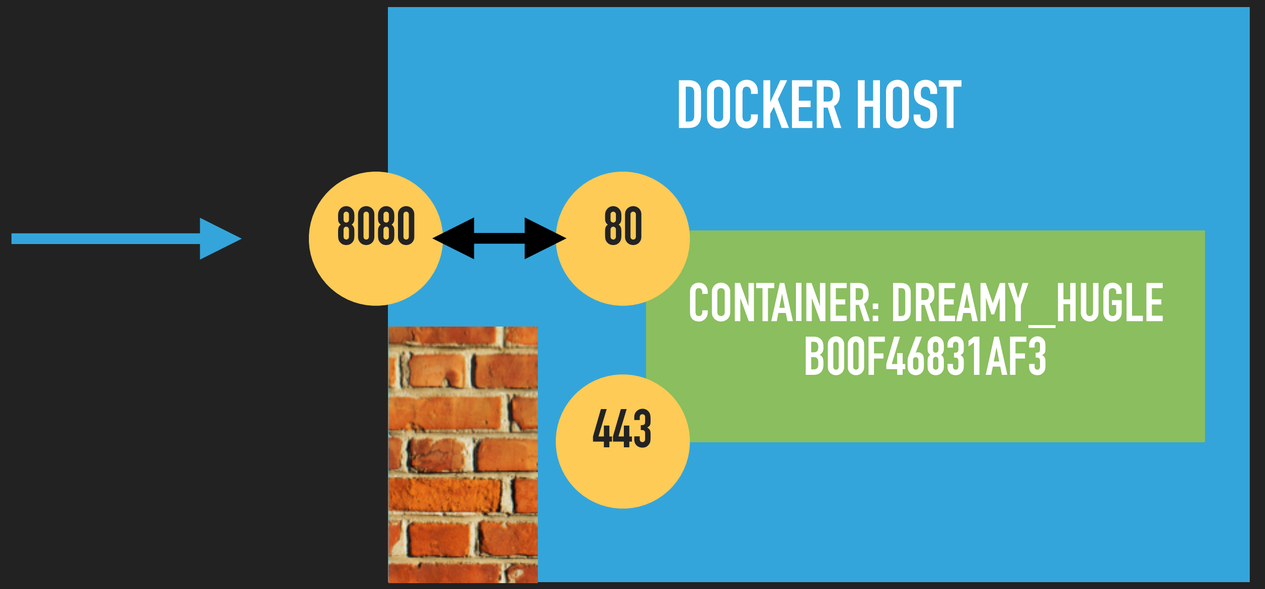


CONTAINER PORTS MUST BE MAPPED TO EXTERNALLY VISIBLE PORTS

```
$ 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
```

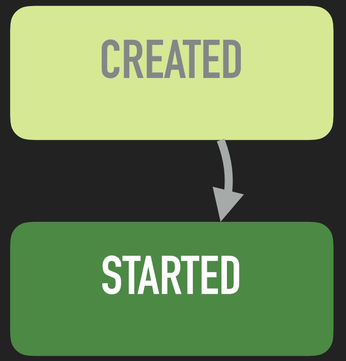


BUILDING A LOCAL TWITTER

- ~~Bootstrap demo~~
- ~~Basic container usage~~
- Running ngnix
- Exposing ports

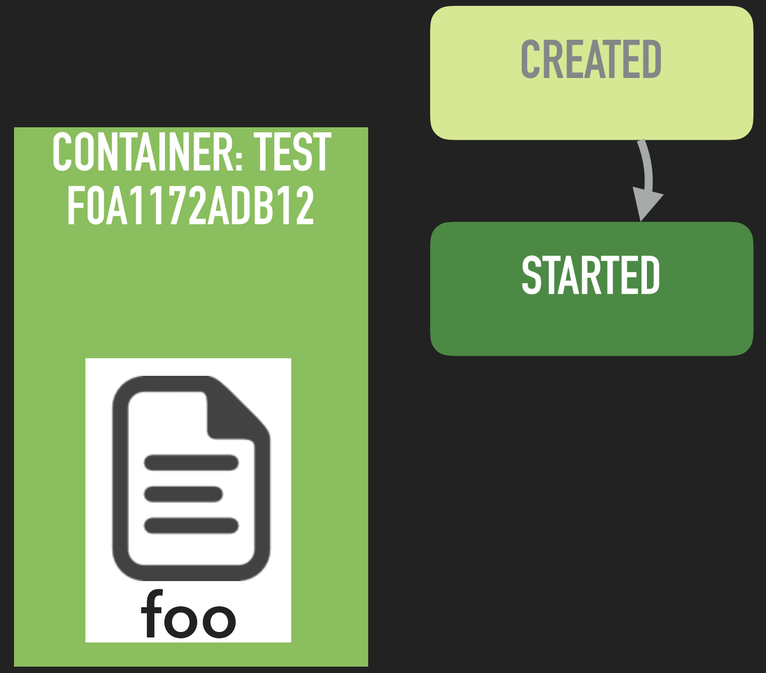
THE CONTAINER FILESYSTEM IS VERSIONED

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



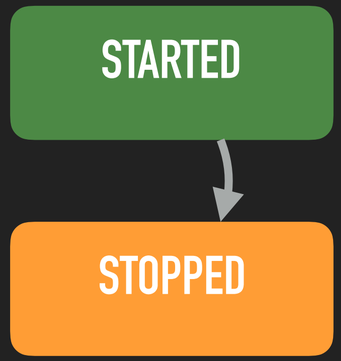
THE CONTAINER FILESYSTEM IS VERSIONED

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



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
```

THE CONTAINER FILESYSTEM SURVIVES RESTARTS

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

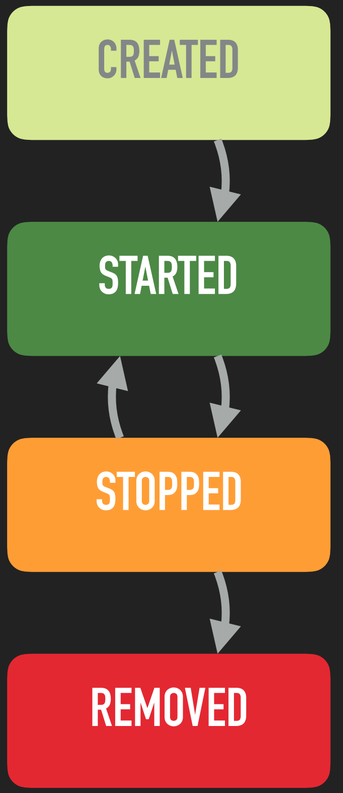
STARTED

THE CONTAINER FILESYSTEM IS VERSIONED

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

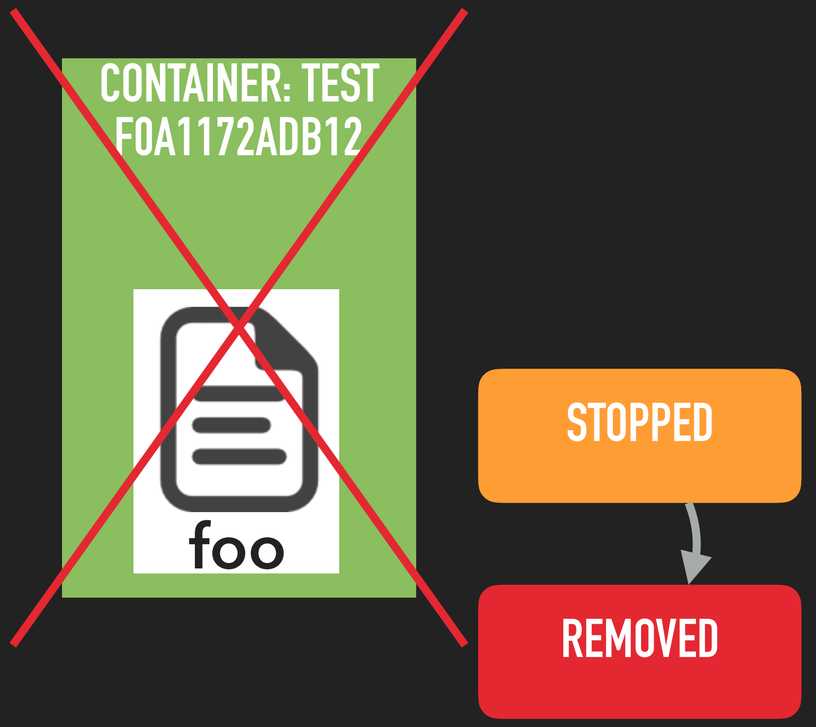
```
/ # echo I was here > foo
/ # cat foo
I was here
```

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



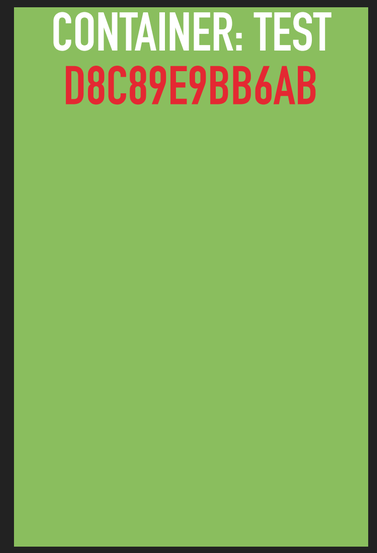
THE CONTAINER FILESYSTEM IS TRANSIENT

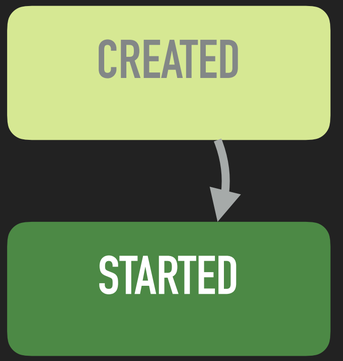
```
$ docker rm test
```



THE CONTAINER FILESYSTEM IS TRANSIENT

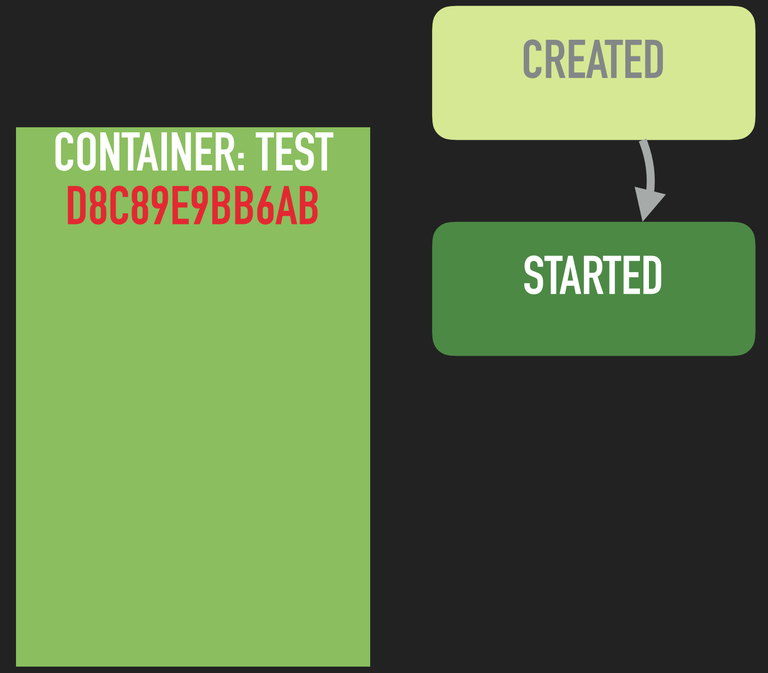
```
$ docker run -ti  --name=test alpine:3.1 sh
```





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

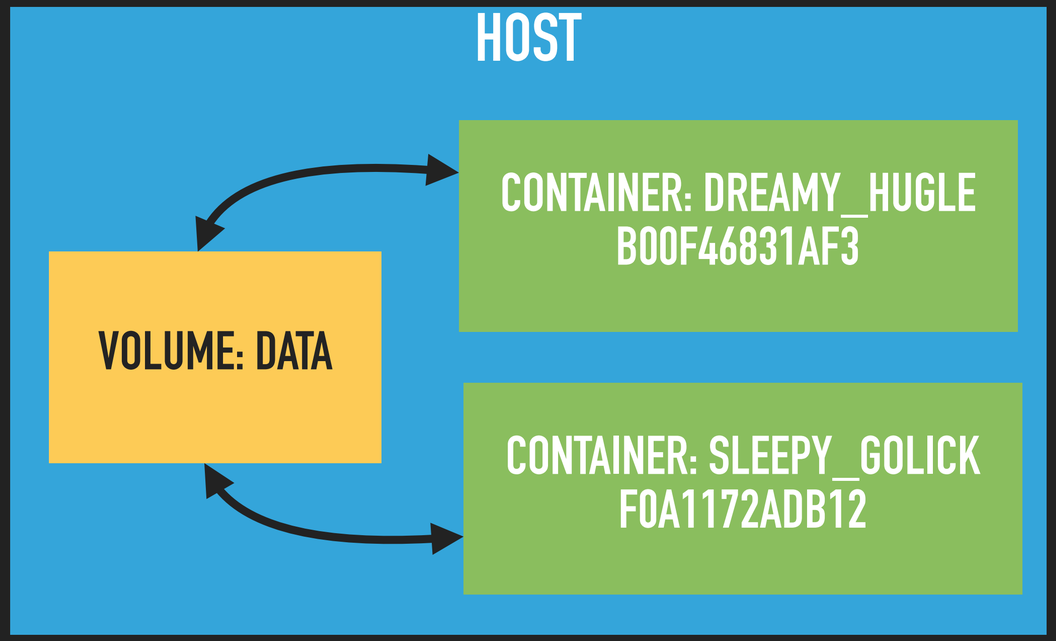
HOST

```
$ ls /usr/
X11
X11R6
bin
include
lib
libexec
local
sbin
share
standalone
```

CONTAINER: DREAMY_HUGLE
B00F46831AF3

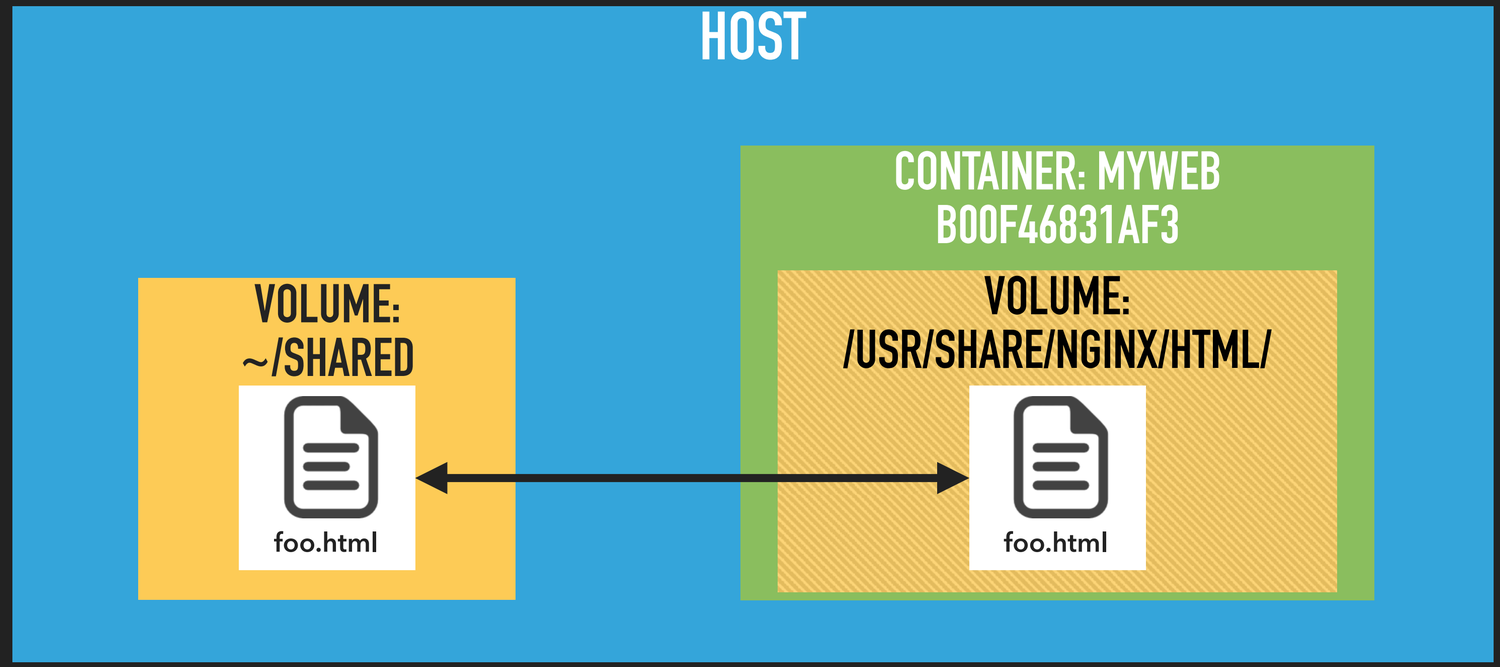
```
/ # ls /usr/
bin
lib
local
sbin
share
```

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 ngnix~~
- ~~Exposing ports~~
- Sharing volumes and data
- Monitoring

THE DOCKERFILE IS THE RECIPE FOR AN IMAGE

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

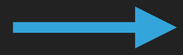
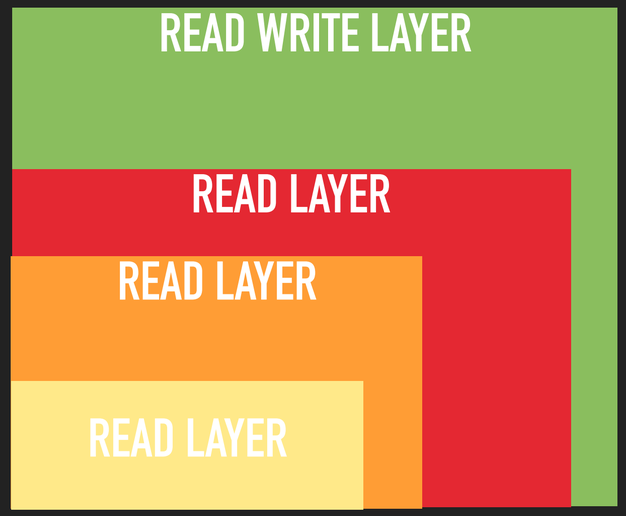


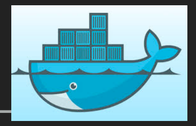
IMAGE
KOENIGHOTZE/JDK8

Dockerfile

```
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

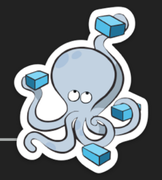
- CMD ["BASH", "-L"]
- VOLUME [$SRC_VOLUME]
- ...
- MAINTAINER DAVID SCHMITZ
- FROM GLIDERLABS/ALPINE:3.3

BUILDING SECURE DOCKER IMAGES

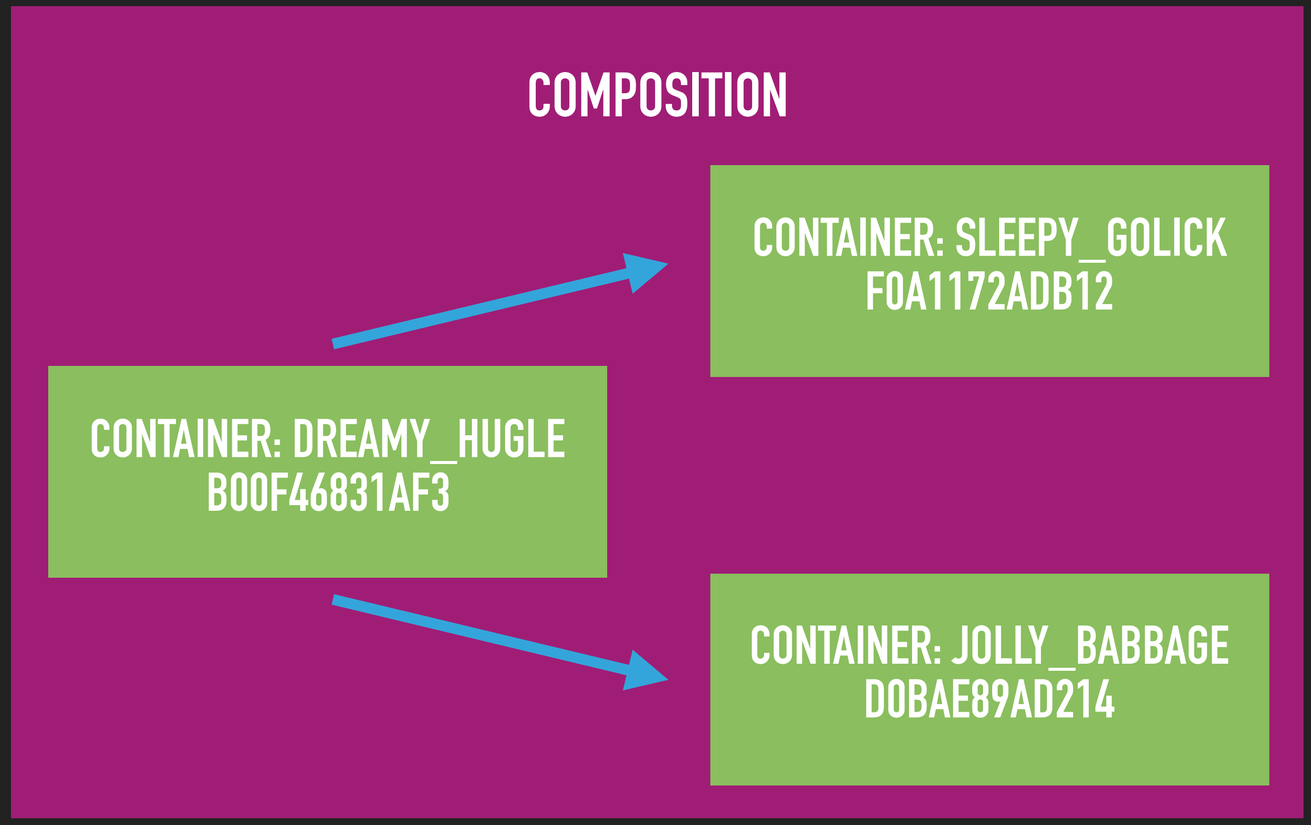
SIDENOTE

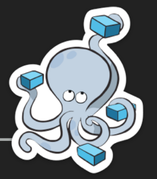
! TO DO !

COMPLEX SCENARIOS USING COMPOSE



MANAGING DEPENDENCIES WITH DOCKER-COMPOSE

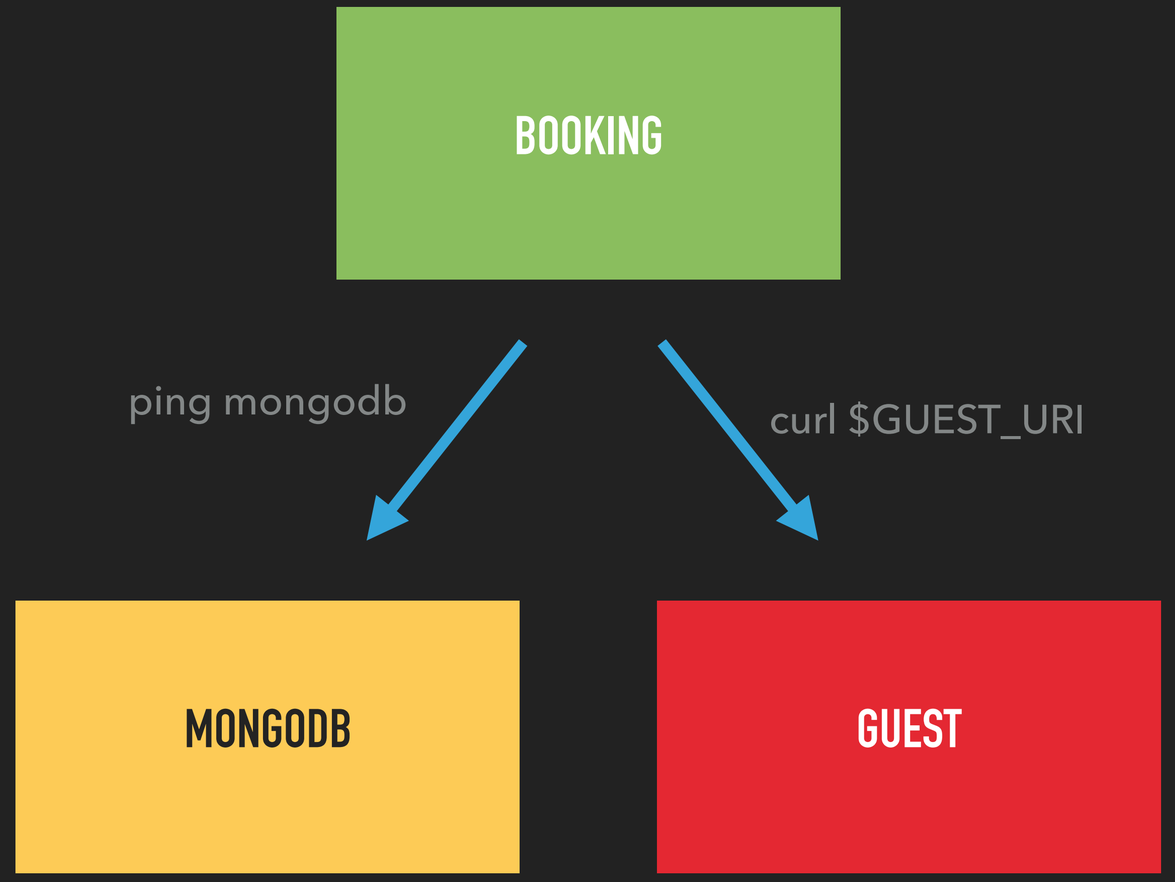




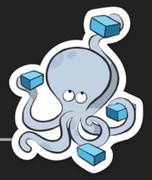
USING DOCKER-COMPOSE FOR DEPENDENCIES*

```
booking:
  image: koenighotze/jee7hotelbooking
  ports:
    - 8080:
    - 9990:
  links:
    - mongo
    - guest
  environme
    - MONGO
    - GUEST_
guest:
  image: koenighotze/jee7hotelguest
  ports:
    - 8081:8080
  expose:
    - 8080
mongodb:
  container_name: jee7hotelmongo
  image: mongo:3.1.9
  expose:
    - "27017"
```

ANPASSEN



*Note: Compose V1 Syntax



DOCKER SINGLE HOST NETWORKING USING LINKS

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

ANPASSEN

MOVIE CONTAINER

MOVIE.WAR

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

ACTOR CONTAINER

ACTOR.WAR

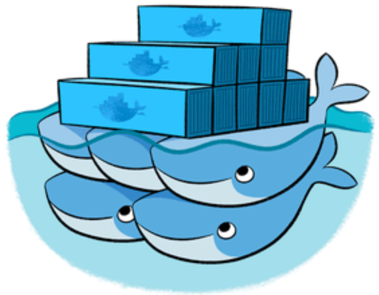
```
@Named
@Path("actor")
@Transactional
public class ActorService {
```

CREATING A SIMPLE COMPOSITION

- 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

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

Loan Application