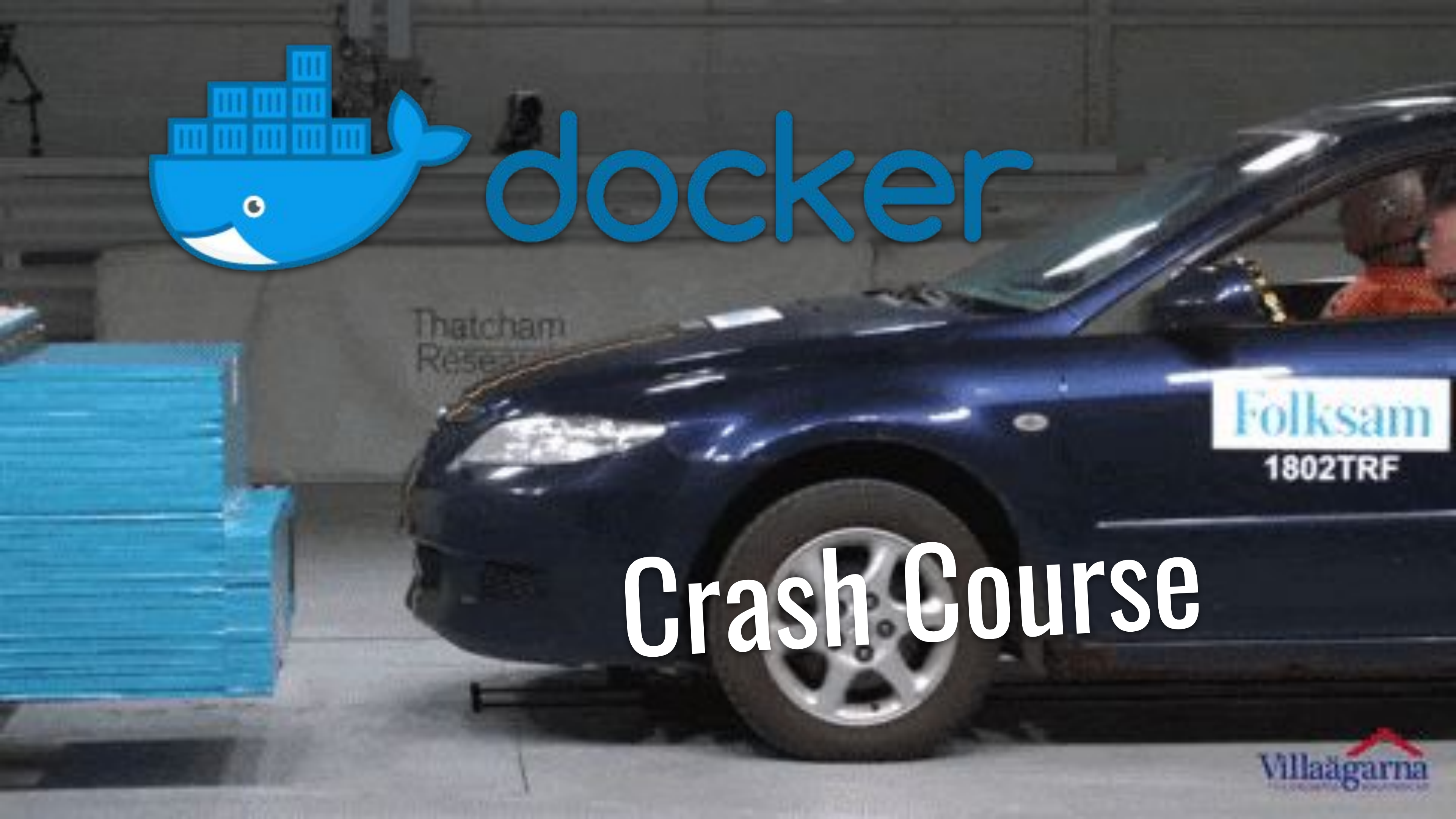
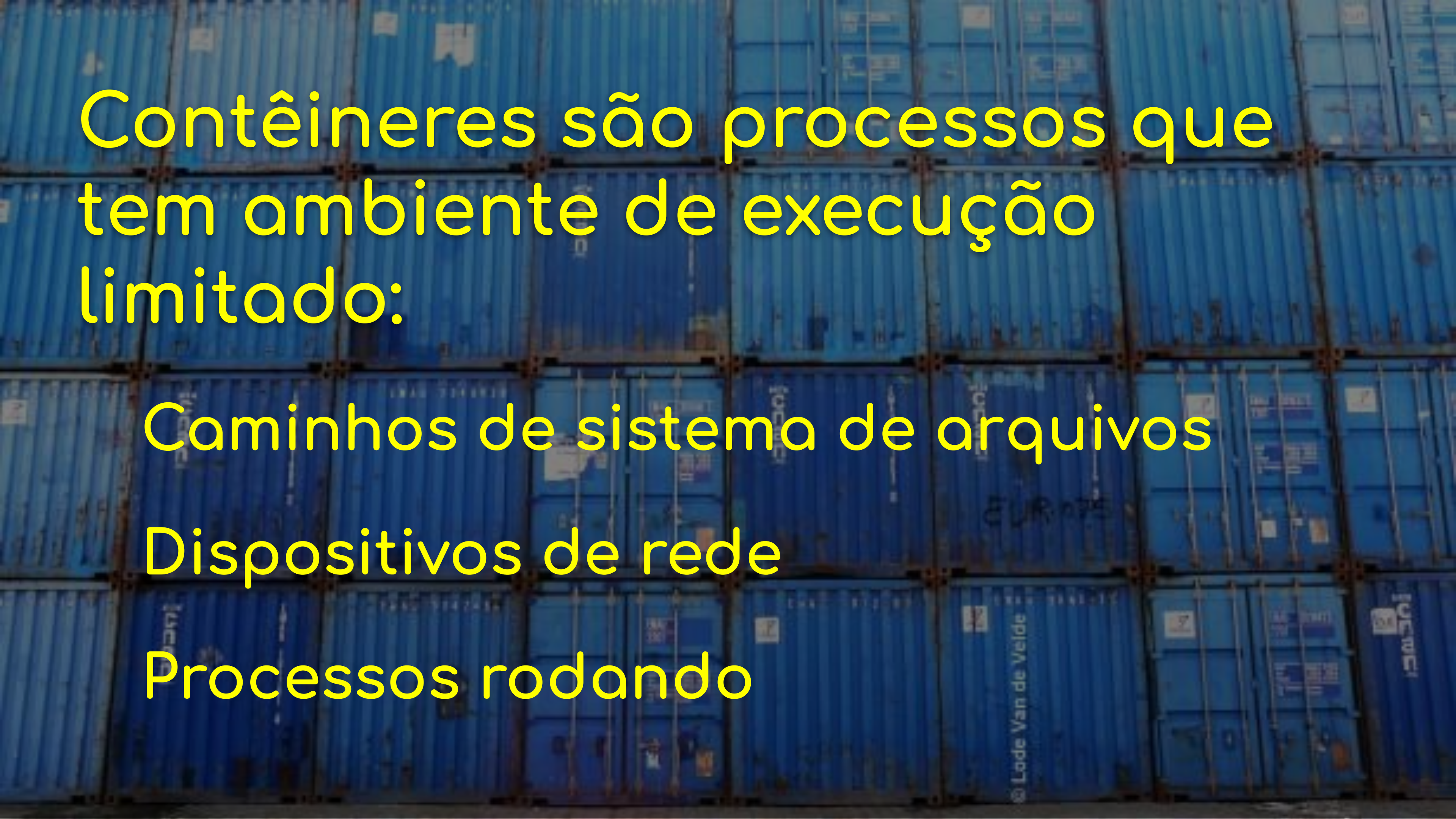
docker Crash Course



Contêineres <u>NÃO</u> são máquinas virtuais!

Contêineres são somente processos.

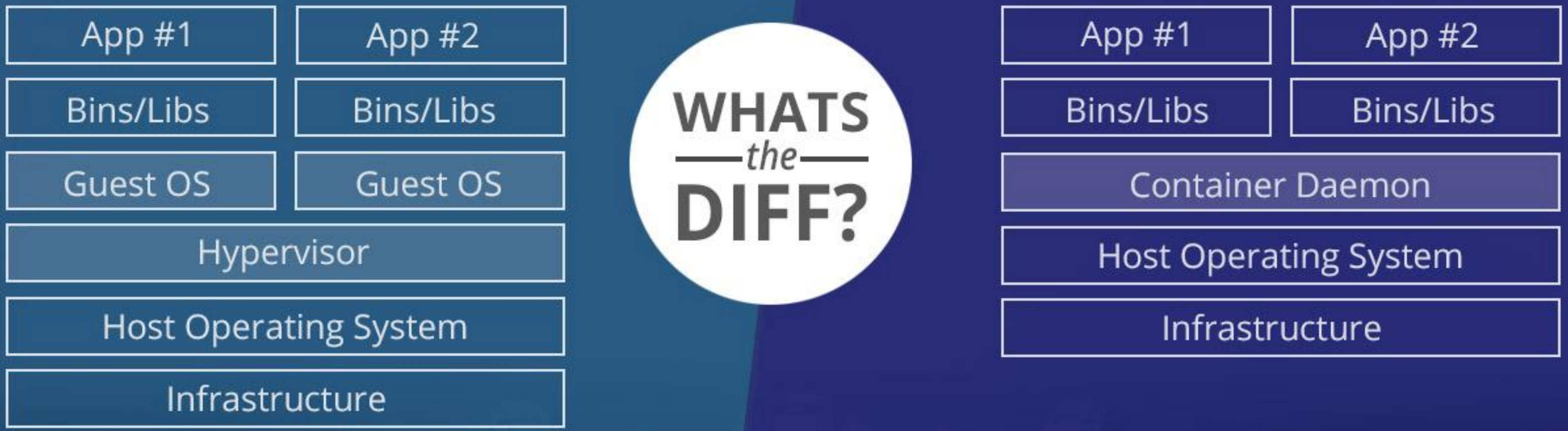


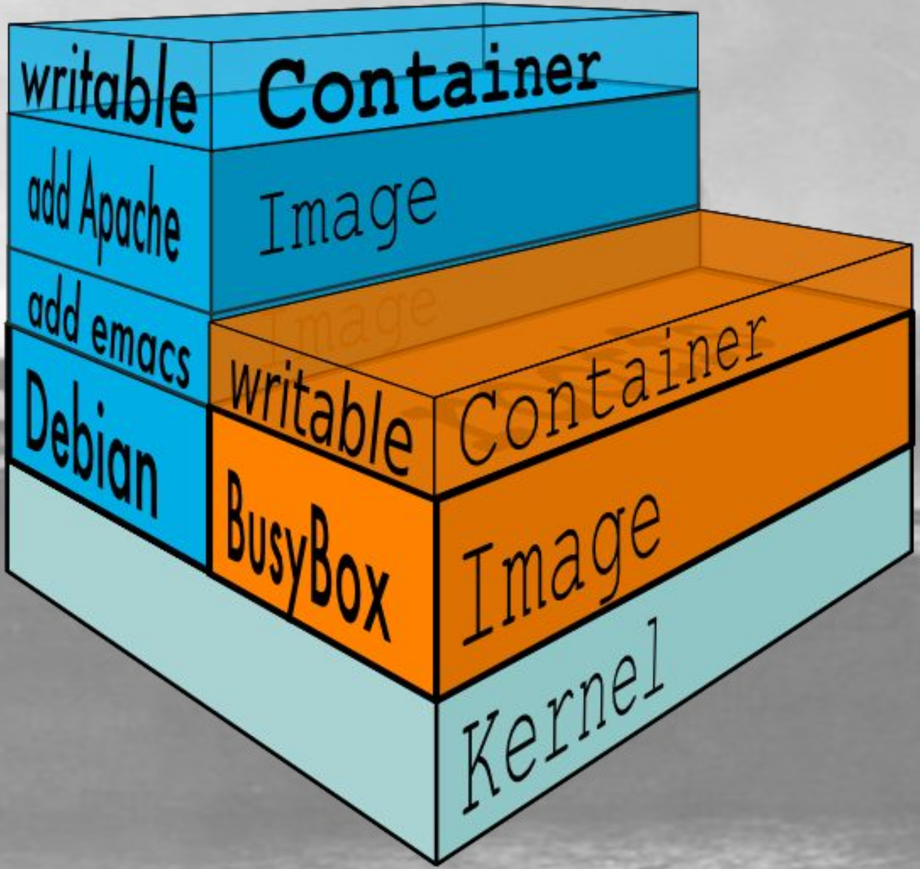
Contêineres são processos que tem ambiente de execução limitado:

- Caminhos de sistema de arquivos
- Dispositivos de rede
- Processos rodando

VIRTUAL MACHINES

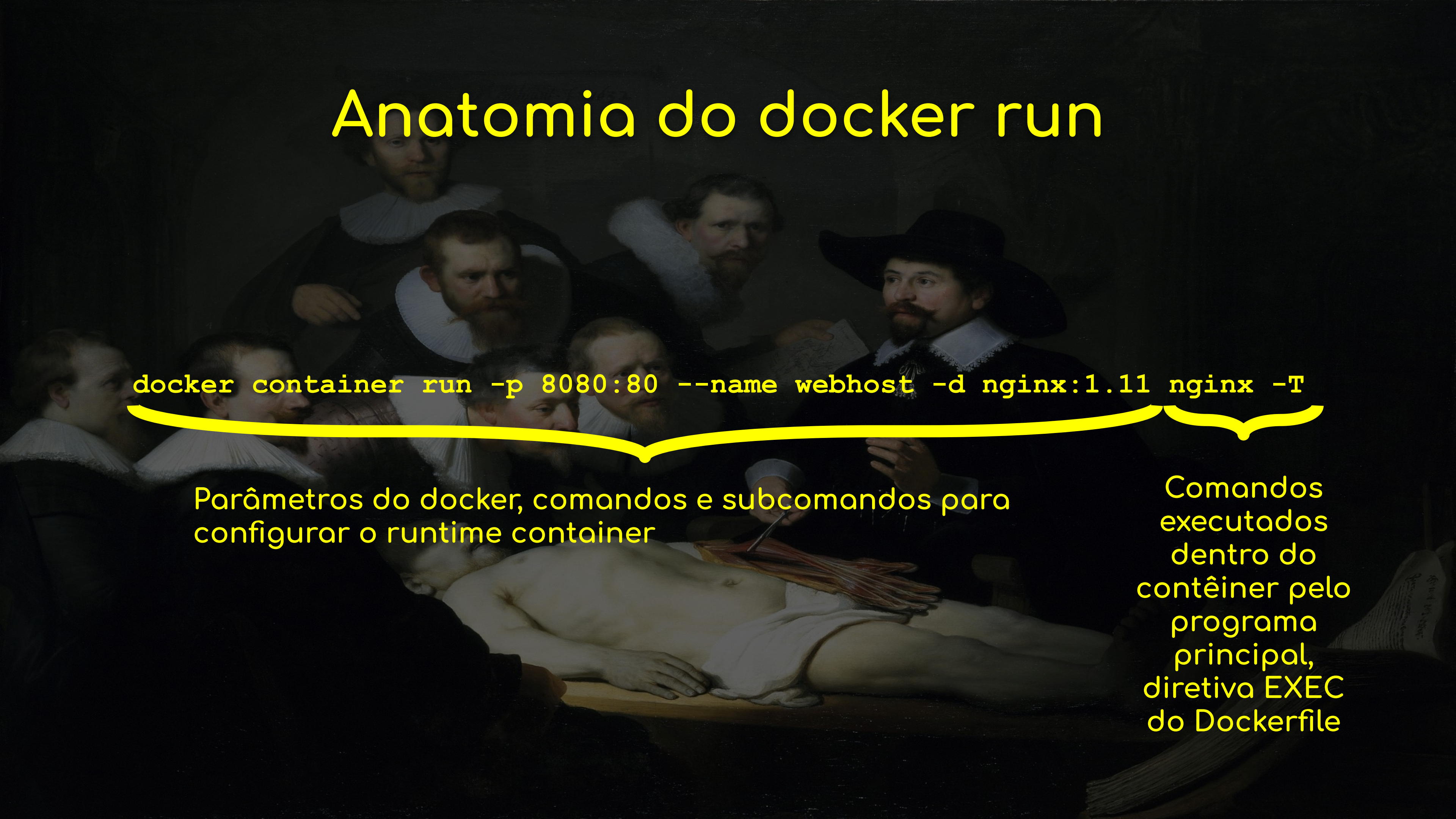
CONTAINERS




writable
Container
add Apache
Image
add emacs
Debian
writable
Container
BusyBox
Image
Kernel

O serviço de hospedagem das imagens chama-se registry

https://hub.docker.com



Anatomia do docker run

```
docker container run -p 8080:80 --name webhost -d nginx:1.11 nginx -T
```

Parâmetros do docker, comandos e subcomandos para configurar o runtime container

Comandos executados dentro do contêiner pelo programa principal, diretiva EXEC do Dockerfile

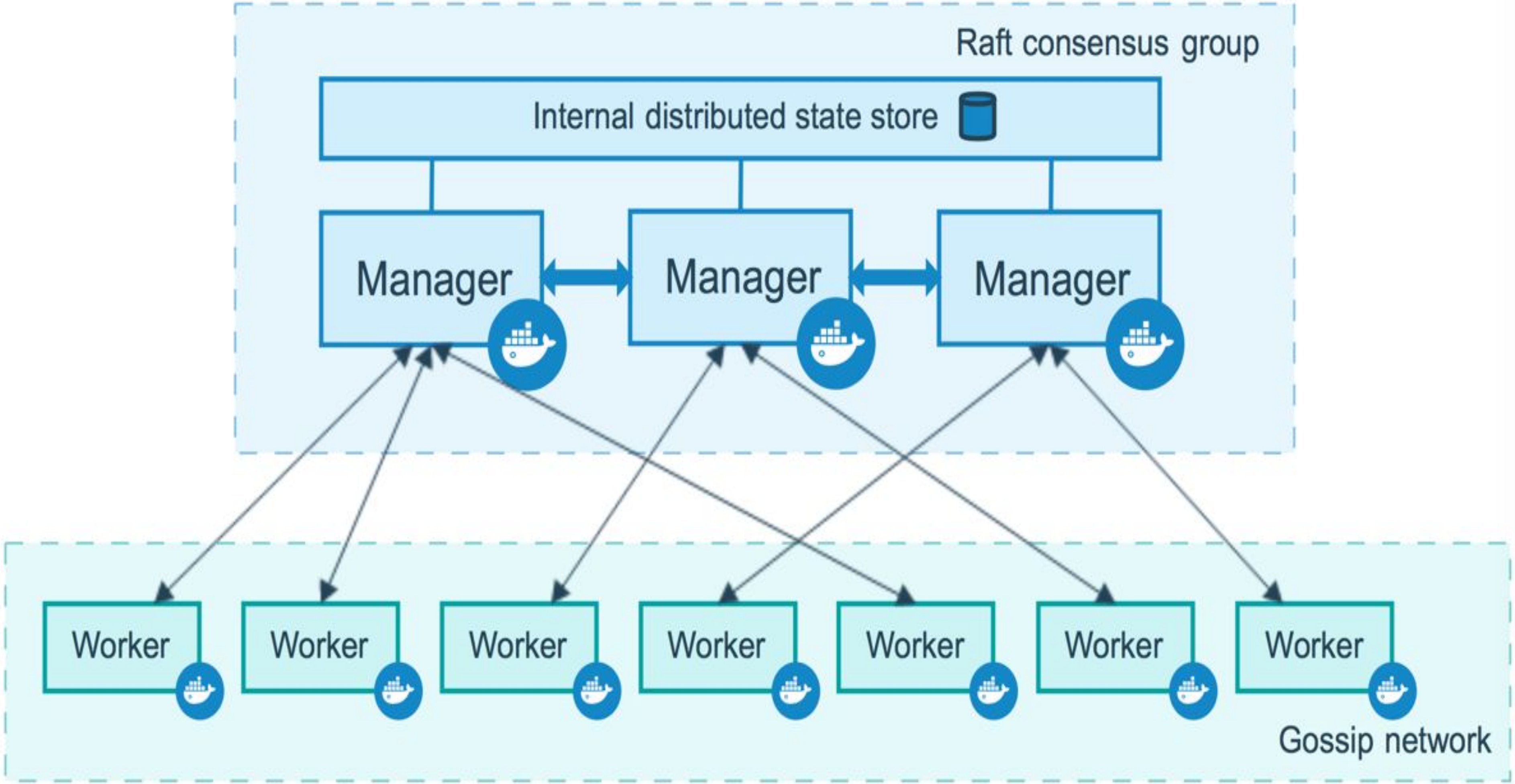
Verificando contêineres:

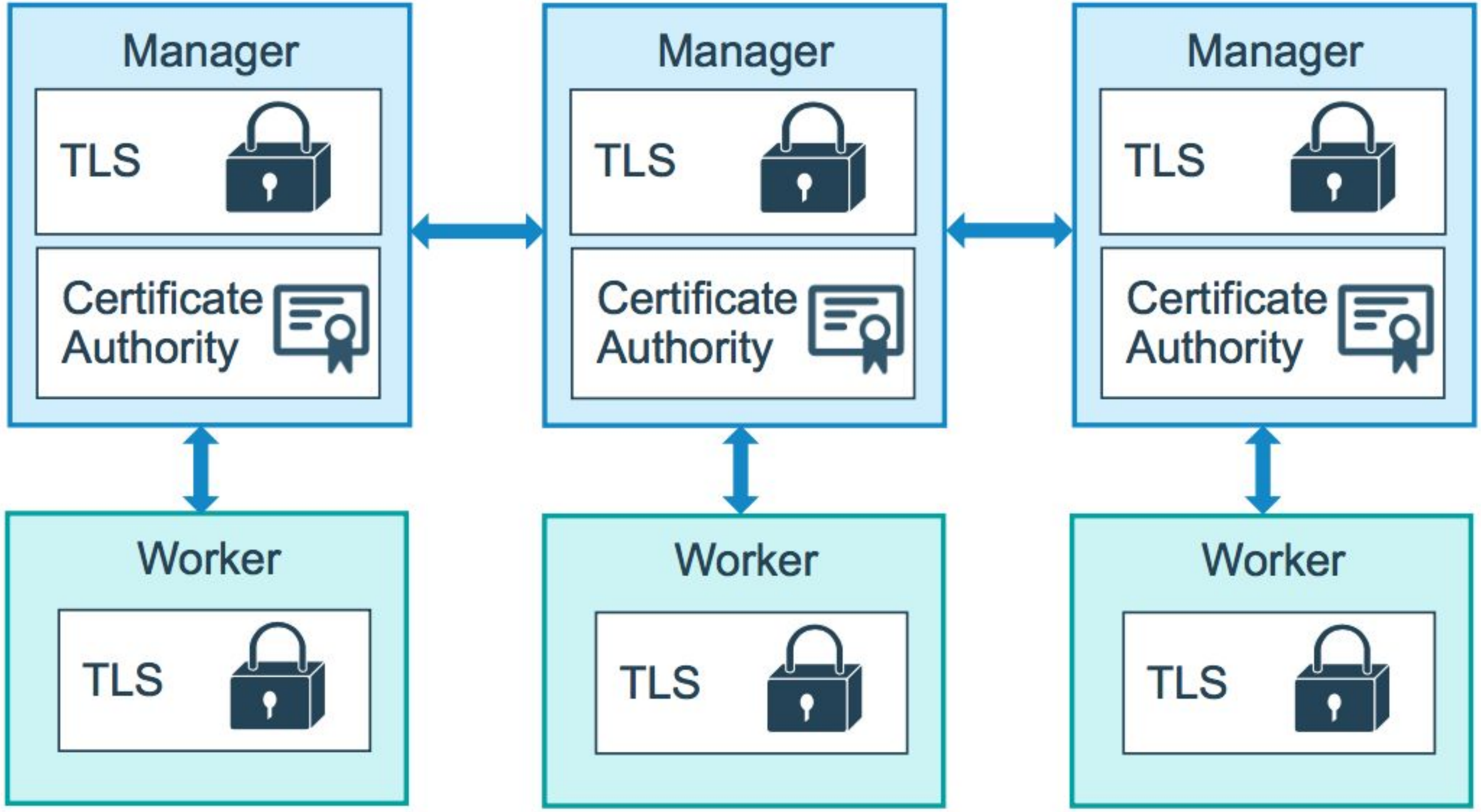
```
docker container ps

docker container ps -a

docker container top <container_id>

docker container inspect <container_id>

docker container stats <container_id>

docker container logs <container_id>

docker container logs -f <container_id>

docker container --it exec <command>

docker container port <container_id>
```

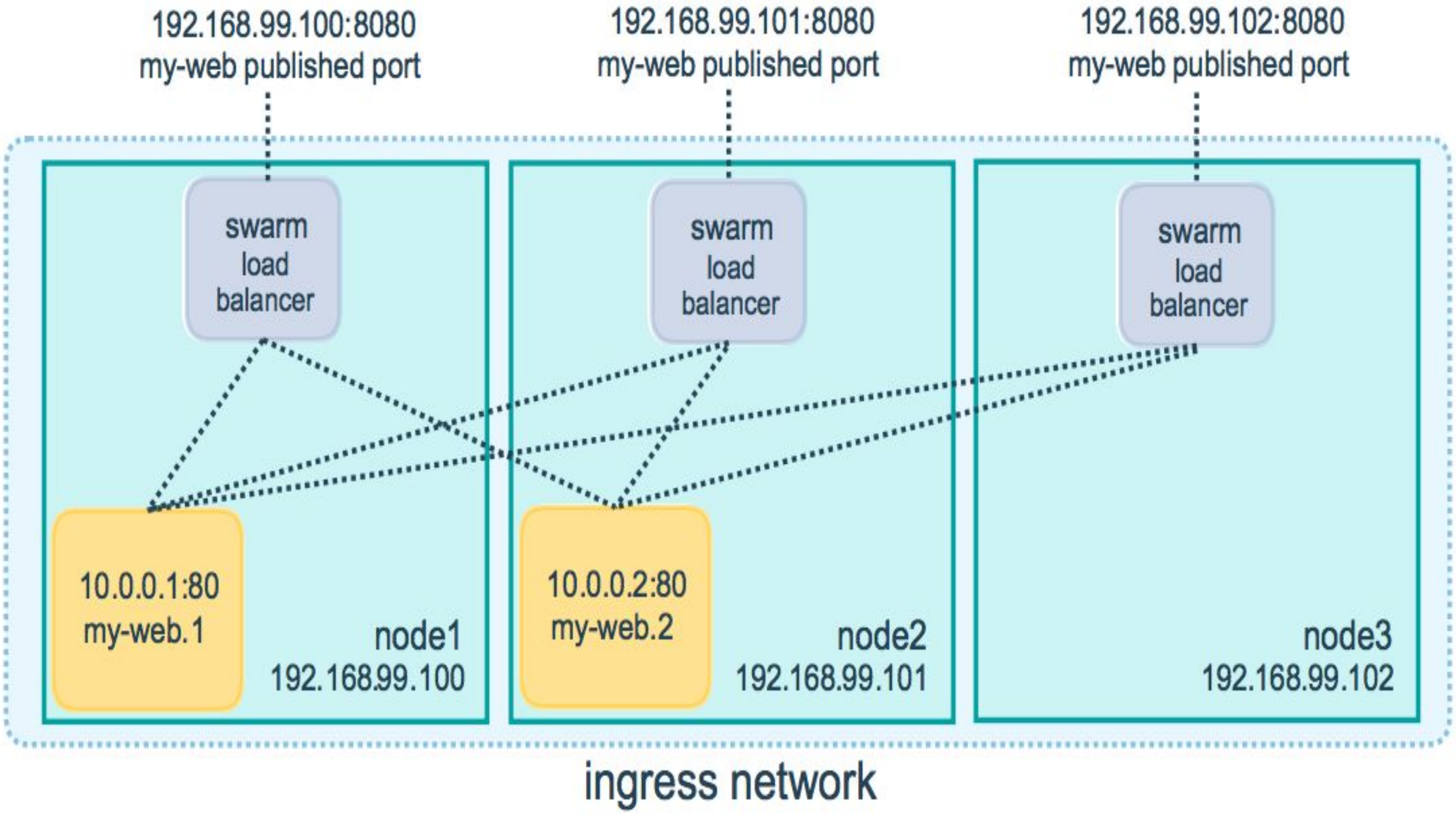
Docker compose

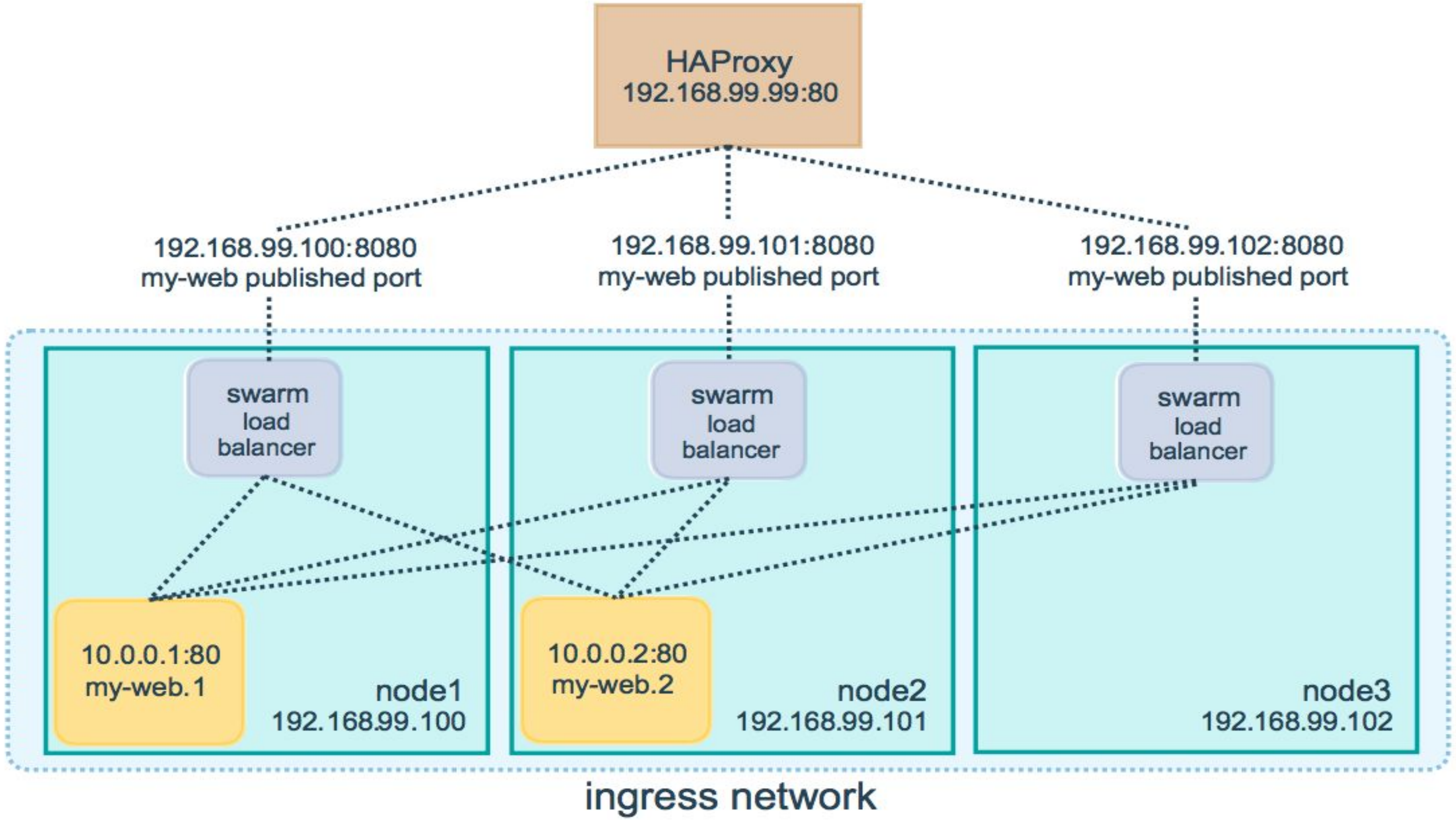
```
version: '3'

services:
 influxdb:
   image: influxdb
   volumes:
     - /opt/docker_compose/influxdb:/var/lib/influxdb
     - /opt/docker_compose/influxdb_config:/etc/influxdb
   ports:
     - "8083:8083"
     - "8086:8086"
     - "2222:22"
   environment:
     - INFLUXDB_DB=jenkins
     - INFLUXDB_ADMIN_USER=admin
     - INFLUXDB_ADMIN_PASSWORD=admin
 jenkins:
   image: jenkins:target
   environment:
     - JAVA_OPTS=-Djenkins.install.runSetupWizard=false
   volumes:
     - /opt/docker_compose/jenkins:/var/jenkins_home
   ports:
     - "8080:8080"
     - "50000:50000"
     - "2223:22"
```

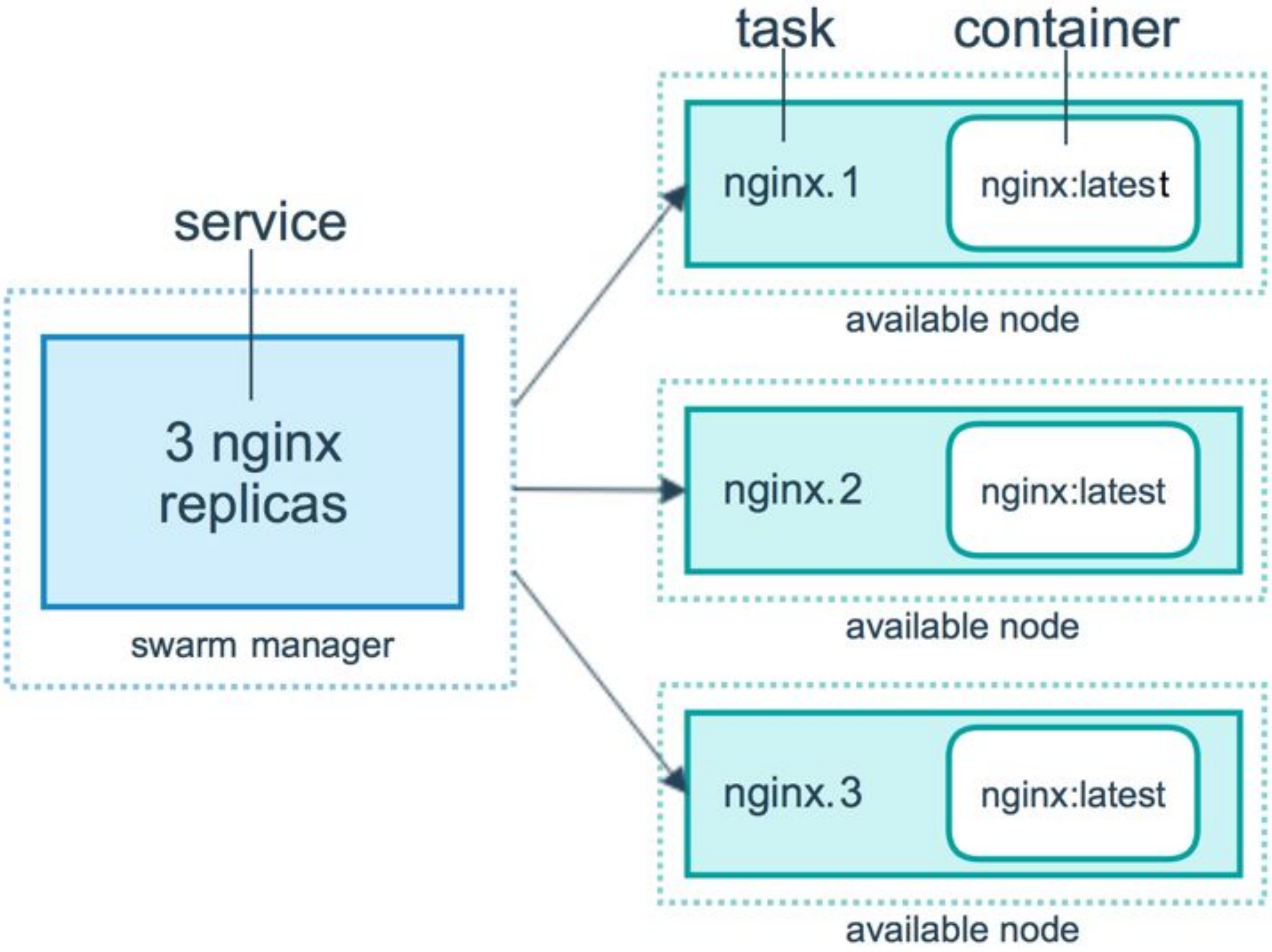
Docker Swarm - rodando contêineres em produção

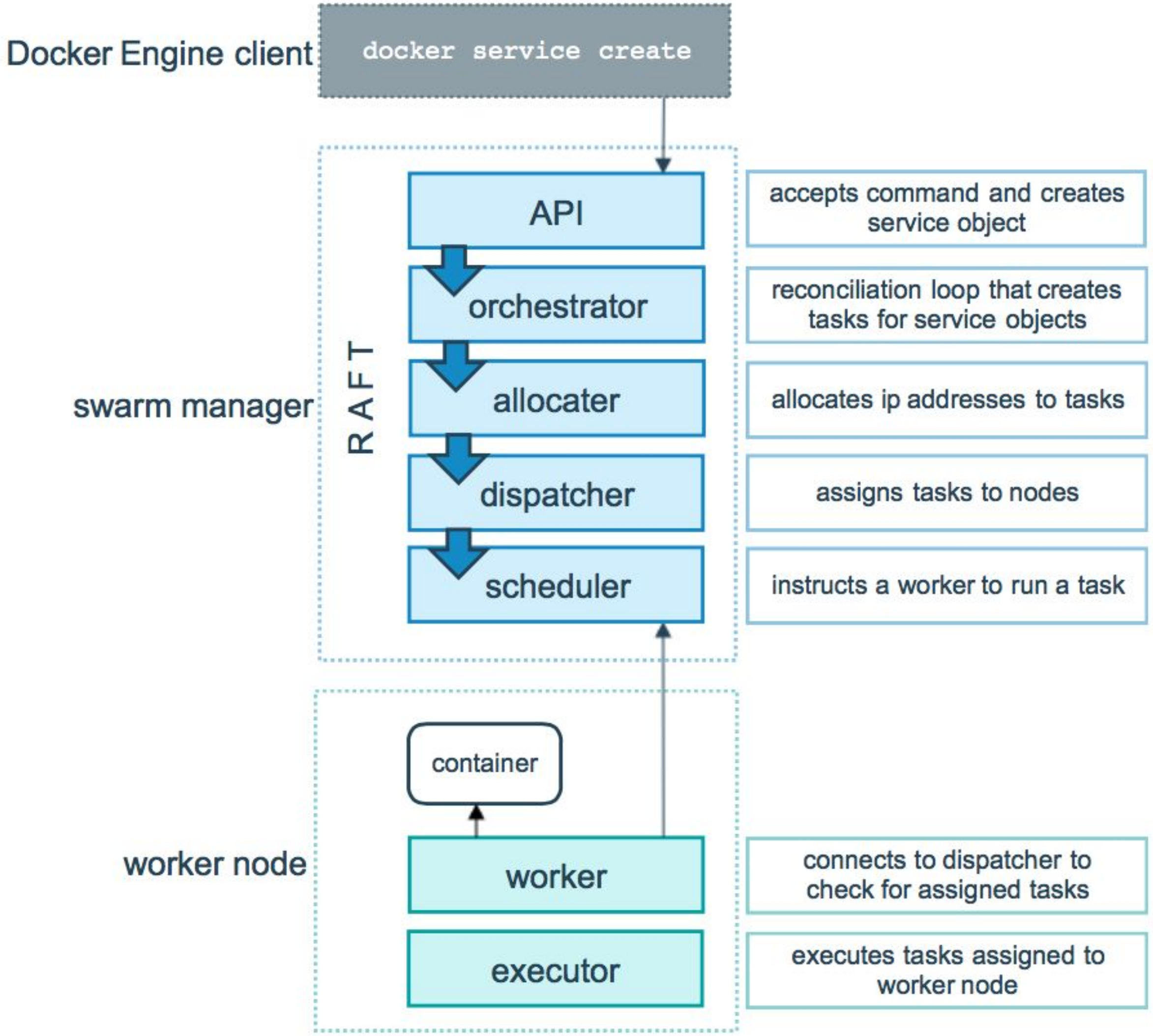

Raft consensus group
Internal distributed state store
Manager
Manager
Manager
Worker
Worker
Worker
Worker
Worker
Worker
Worker
Gossip network

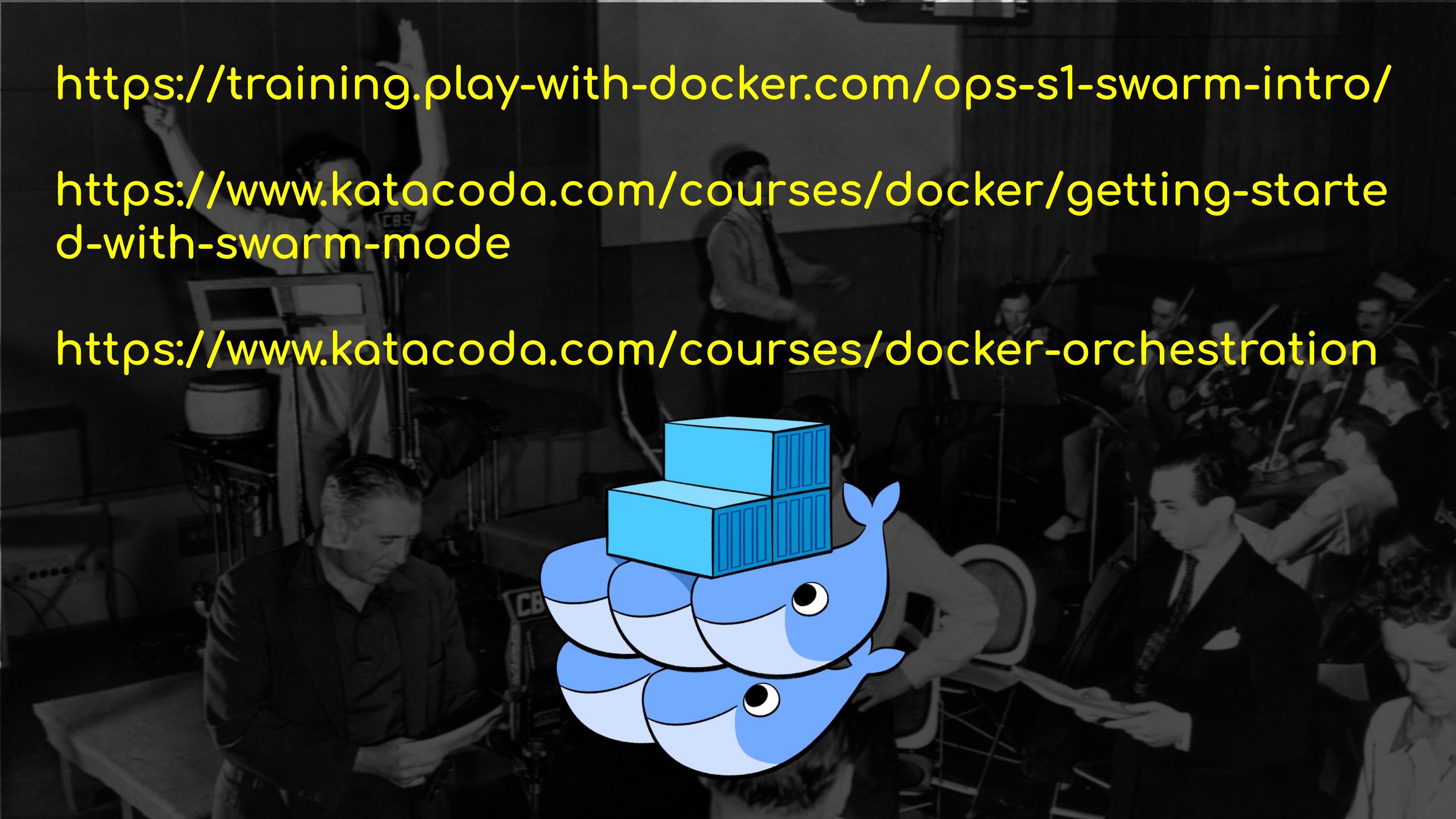

Manager
TLS
Certificate Authority
Manager
TLS
Certificate Authority
Manager
TLS
Certificate Authority
Worker
TLS
Worker
TLS
Worker
TLS


192.168.99.100:8080
my-web published port
192.168.99.101:8080
my-web published port
192.168.99.102:8080
my-web published port
swarm load balancer
swarm load balancer
swarm load balancer
10.0.0.1:80
my-web.1
node1
192.168.99.100
10.0.0.2:80
my-web.2
node2
192.168.99.101
node3
192.168.99.102
ingress network


HAProxy
192.168.99.99:80
192.168.99.100:8080
my-web published port
192.168.99.101:8080
my-web published port
192.168.99.102:8080
my-web published port
swarm load balancer
swarm load balancer
swarm load balancer
10.0.0.1:80
my-web.1
node1
192.168.99.100
10.0.0.2:80
my-web.2
node2
192.168.99.101
node3
192.168.99.102
ingress network


task
container
service
nginx.1
nginx:latest
available node
3 nginx replicas
nginx.2
nginx:latest
swarm manager
available node
nginx.3
nginx:latest
available node


Docker Engine client
docker service create
swarm manager
RAFT
API
orchestrator
allocater
dispatcher
scheduler
accepts command and creates service object
reconciliation loop that creates tasks for service objects
allocates ip addresses to tasks
assigns tasks to nodes
instructs a worker to run a task
worker node
container
worker
executor
connects to dispatcher to check for assigned tasks
executes tasks assigned to worker node



https://training.play-with-docker.com/ops-s1-swarm-intro/

https://www.katacoda.com/courses/docker/getting-started-with-swarm-mode

https://www.katacoda.com/courses/docker-orchestration

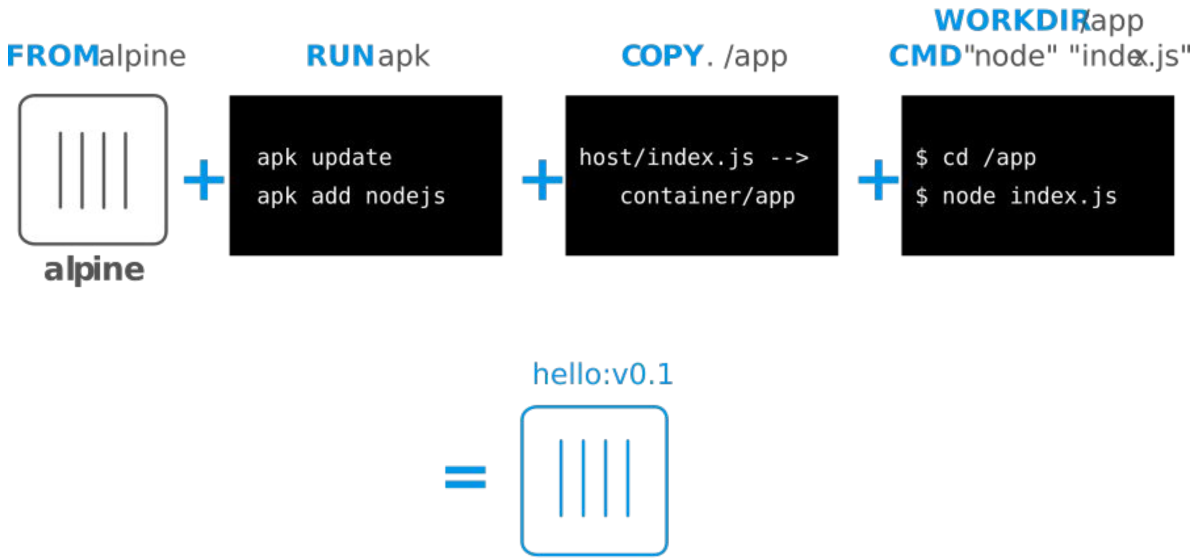
Construindo imagens Docker

```
FROM jenkins/jenkins:alpine

USER root

RUN apk add ca-certificates

ENV JAVA_OPTS="-Djenkins.install.runSetupWizard=false"

COPY plugins.txt /usr/share/jenkins/ref/plugins.txt

RUN for i in $(cat /usr/share/jenkins/ref/plugins.txt|grep -v
^#) ; do echo "########## Installing $i ############";
/usr/local/bin/install-plugins.sh $i; done

RUN chown 1000:1000 /var/jenkins_home -R

RUN apk add --no-cache tzdata

ENV TZ America/Sao_Paulo
```

Dockerfiles

Dockerfile:

```
FROM alpine
RUN apk update && apk add nodejs
COPY . /app
WORKDIR /app
CMD ["node","index.js"]
```

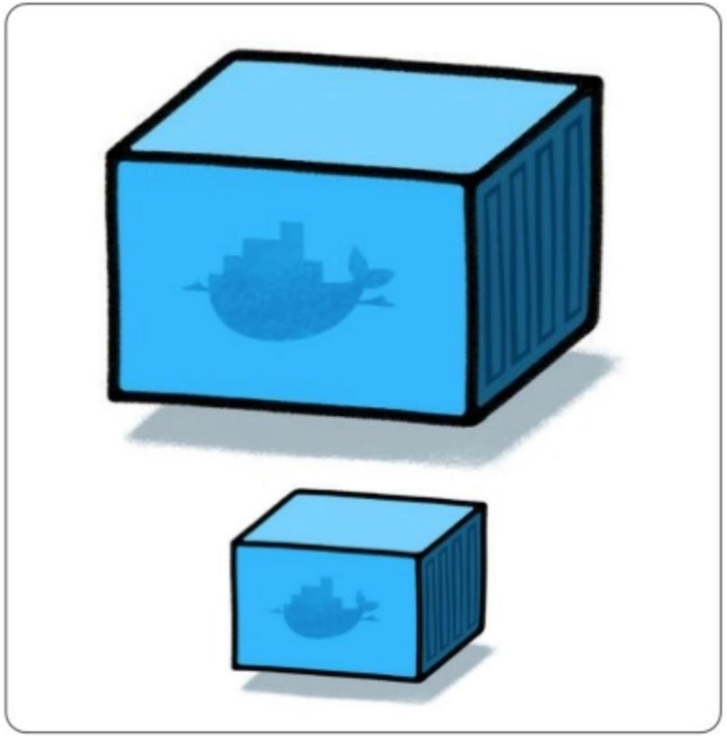


1

```
# Dockerfile
# build stage
FROM buildbase as build
...
...
...
```

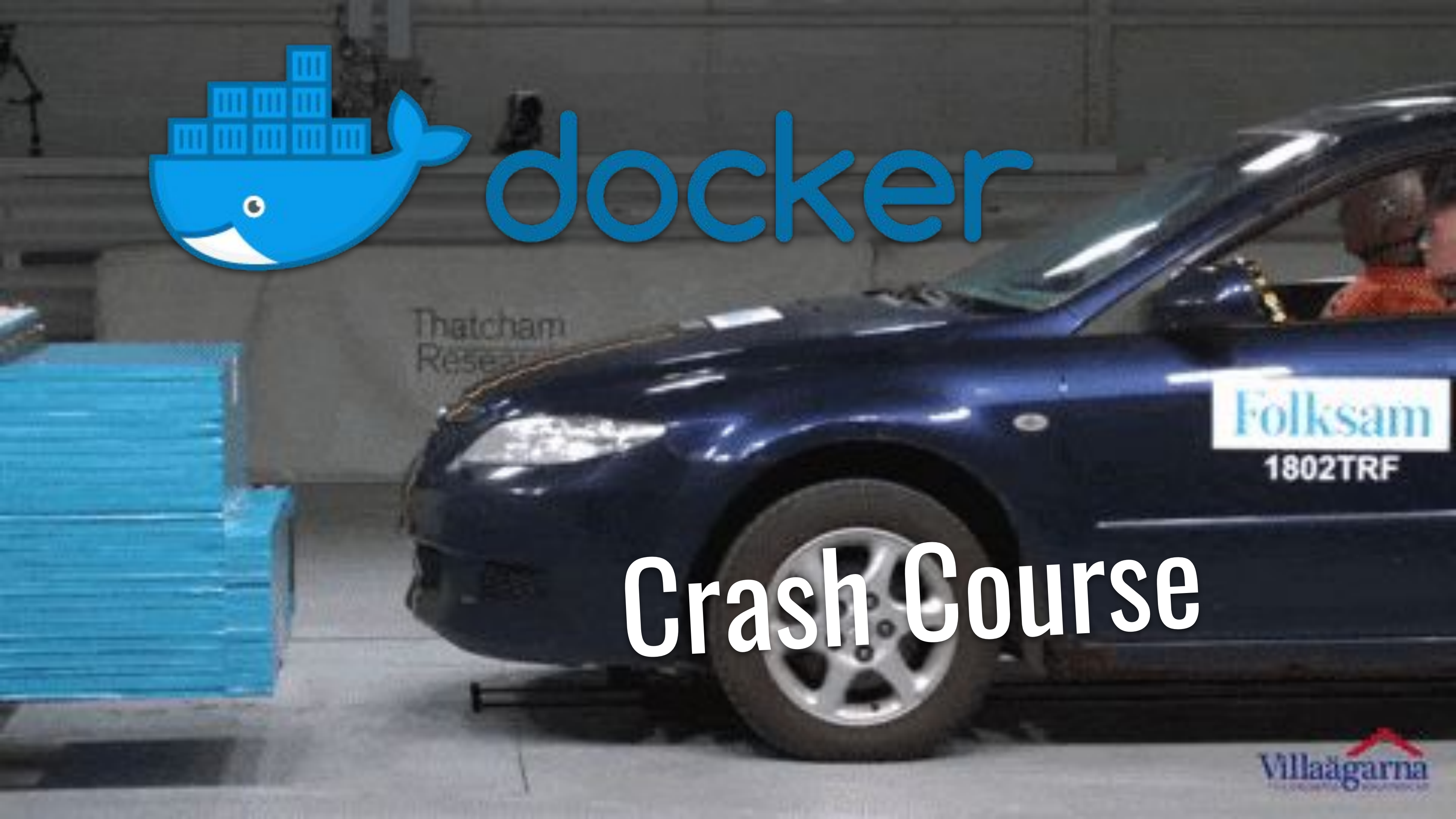
2

```
# production ready stage
FROM runbase
...
COPY --from=build
/artifact /app
```



https://training.play-with-docker.com/multi-stage/

https://www.katacoda.com/courses/docker/multi-stage-builds



Crash Course