



OFICINA DE INTRODUÇÃO AO DOCKER



Um dos mais interessantes e poderosos projetos open-source a ganhar vida nos últimos anos. Docker pode ajudá-lo em tantas coisas que é injusto tentar resumir as suas capacidades em uma frase.

(Artigo Digital Ocean)



O QUE VAMOS VER

- O que é Docker
- Arquitetura
- Instalação
- Containers / Contêineres
- Imagens
- Dockerfile
- Compose



O QUE É DOCKER

Entrando no mundo da containerização

**WORKED FINE IN
DEV**

OPS PROBLEM NOW



Containerização

Em 26 de Abril de 1956, acontecia o primeiro embarque de contêineres em navios, um navio carregando 58 contêineres deixou Newark, partindo para Houston.

A emergência do transporte de carga containerizada foi um facilitador crucial para o desenvolvimento do comércio mundial. Contêineres tornaram-se uma rápida, confiável e econômica maneira de se transportar produtos diversos.

Malcolm P. McLean é conhecido como o “Pai da Containerização”.



CONTAINERIZAÇÃO E DOCKER

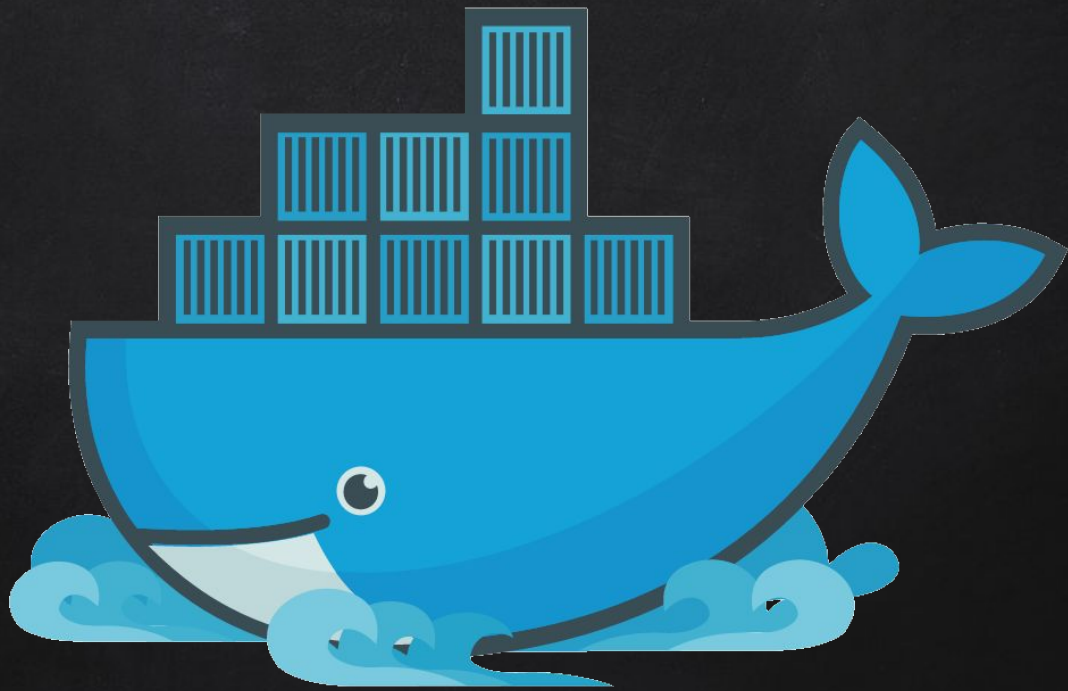
O primeiro anúncio de Docker foi em uma conferência de Python em 2013 por Solomon Hykes, CEO na época da dotCloud.

Docker foi criado, inicialmente, para que a gerência do PaaS (Platform as a Service) pudesse ser de forma mais simples, a ideia é que em vez de máquinas virtuais por debaixo dos panos, tudo rodaria em contêineres Linux.

Para a parte de containerização, inicialmente, Docker utilizava a antiga tecnologia LXC (Linux Containers) que existe em toda distribuição Linux desde 2008.

A partir da versão 1 o Docker substituiu o LXC pela libcontainer e a partir de junho de 2015 passou a utilizar o runC (Runtime Container), evolução da libcontainer, doada posteriormente ao OCI (Open Container Initiative).

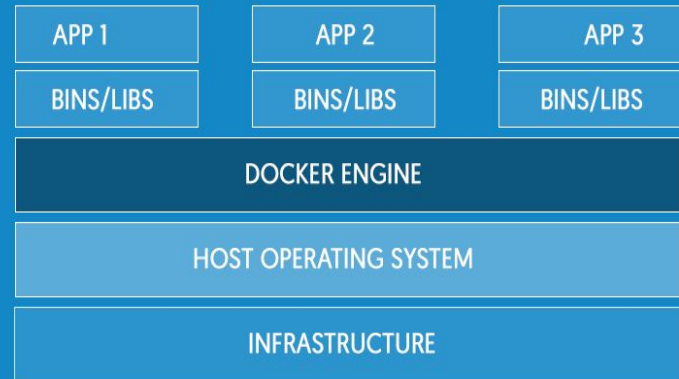
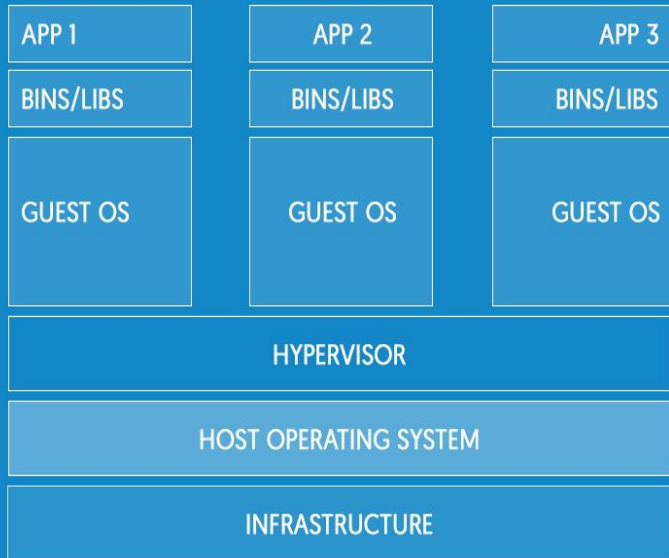
BUILD, SHIP, RUN

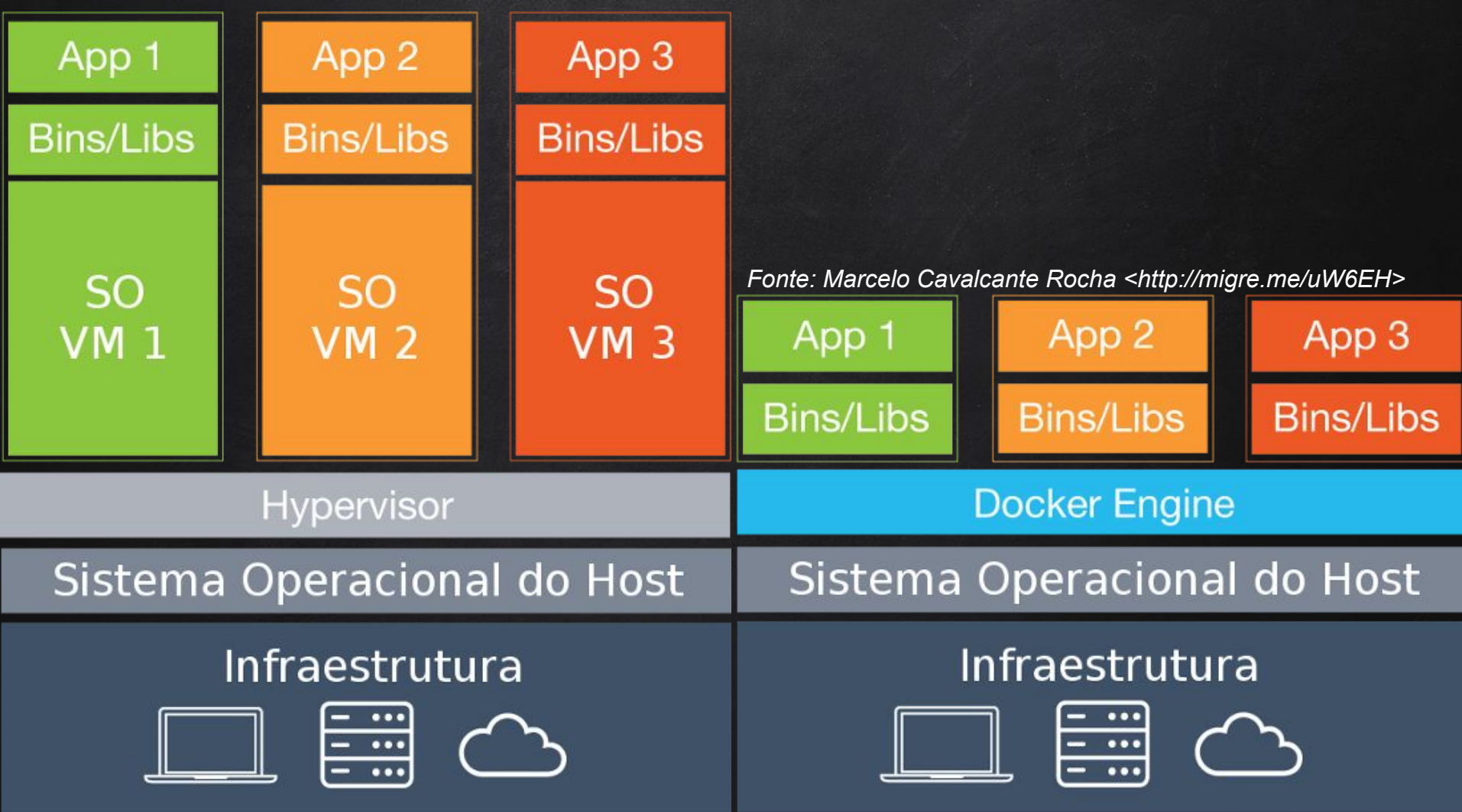


Docker é uma plataforma
open source usada para
construir, executar e
distribuir ambientes

Que é uma forma de empacotamento/
containerização de infraestrutura
portável e simples, na qual constitui
facilmente várias "máquinas"
executando no mesmo kernel, porém
isoladas logicamente, usando as
tecnologias runC.

Tudo isso é possível por conta da Docker Engine...







ARQUITETURA DOCKER

Como funciona por debaixo dos “panos”



PARTES PRINCIPAIS DOCKER

docker daemon

usado para
gerenciar os
containers docker
no host onde ele
roda

docker CLI

usado para
comandar e se
comunicar com o
docker daemon

docker registry

um repositório
(público ou
privado) para as
imagens do
docker

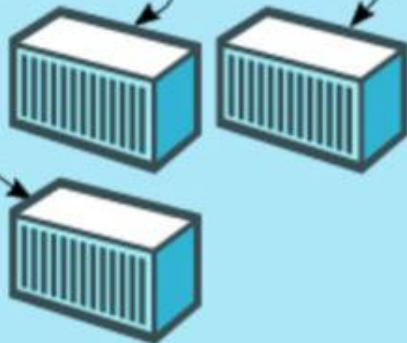
OS X

Docker client

DOCKER_HOST

Linux VM

Docker daemon



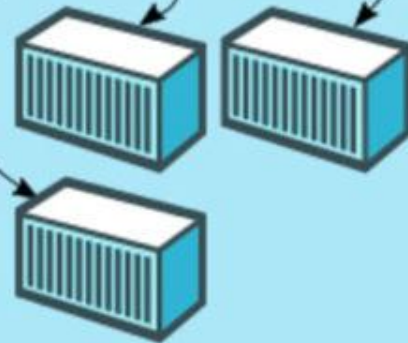
Windows

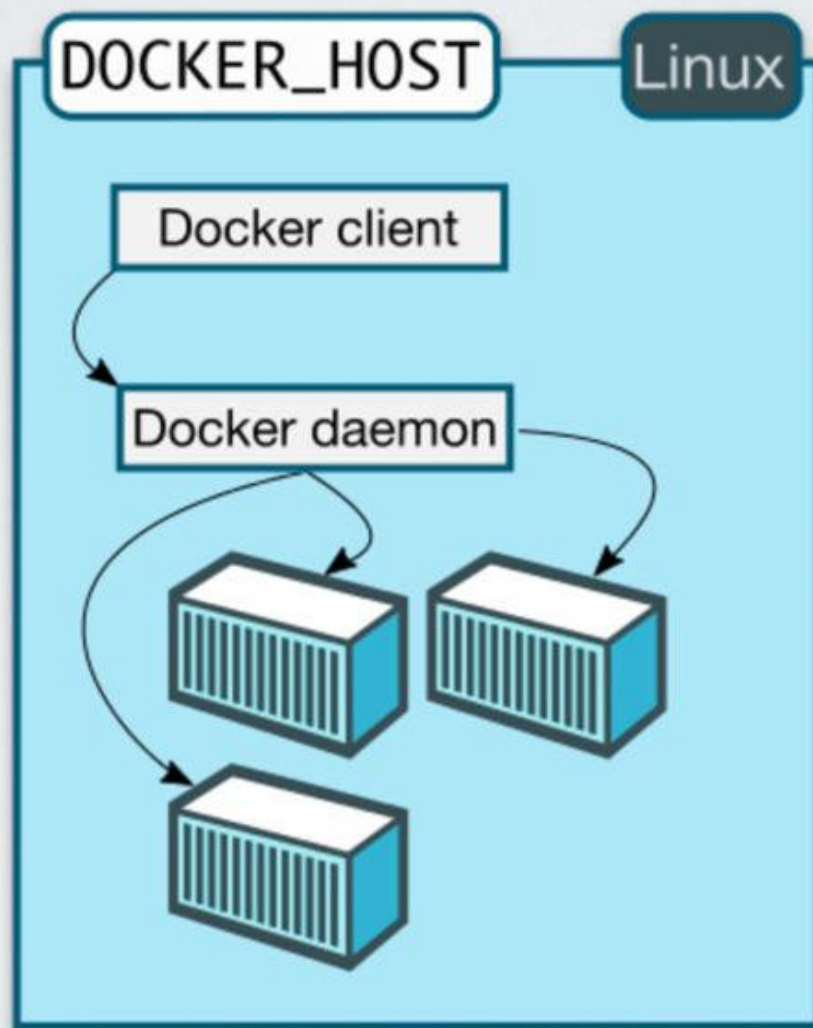
Docker client

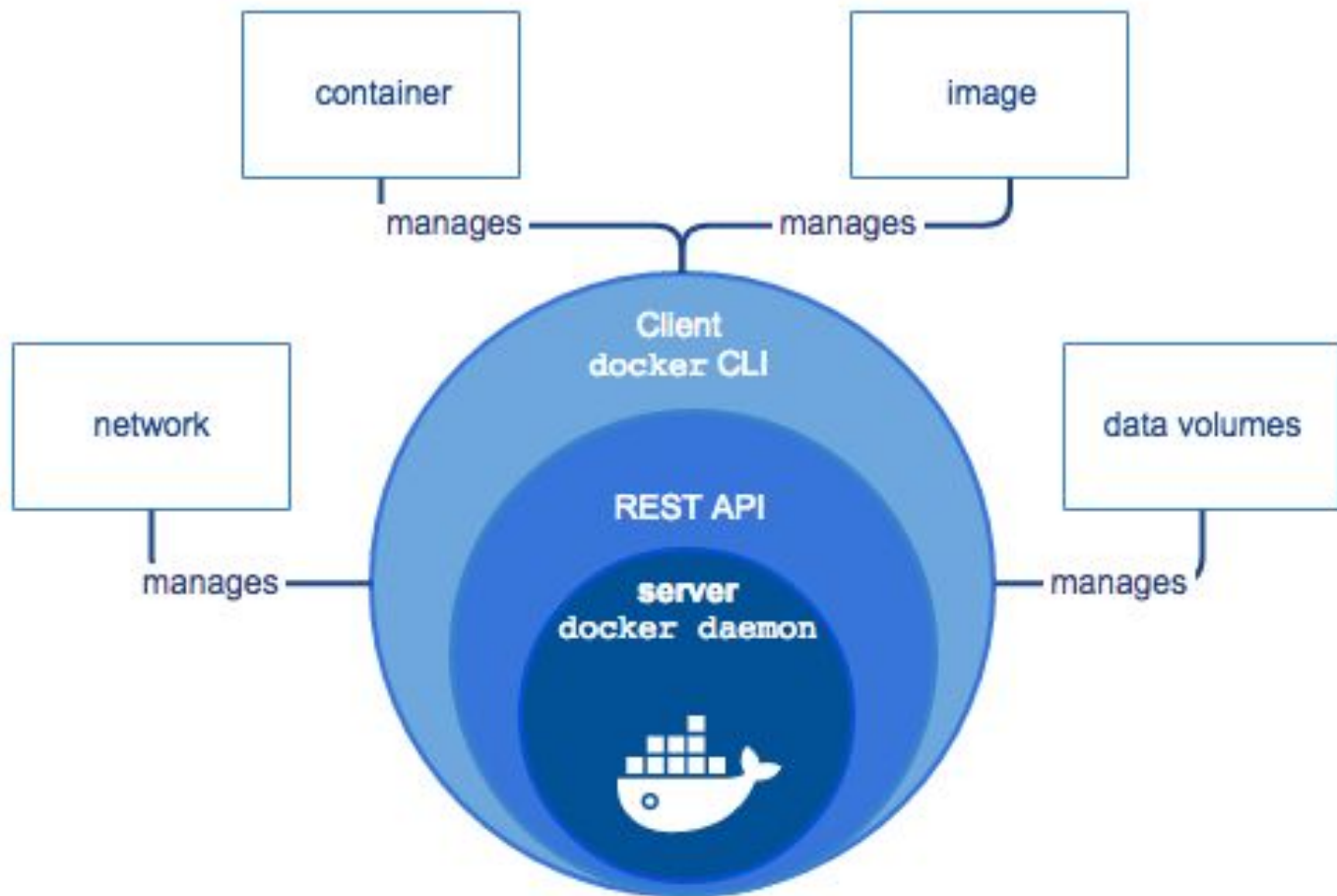
DOCKER_HOST

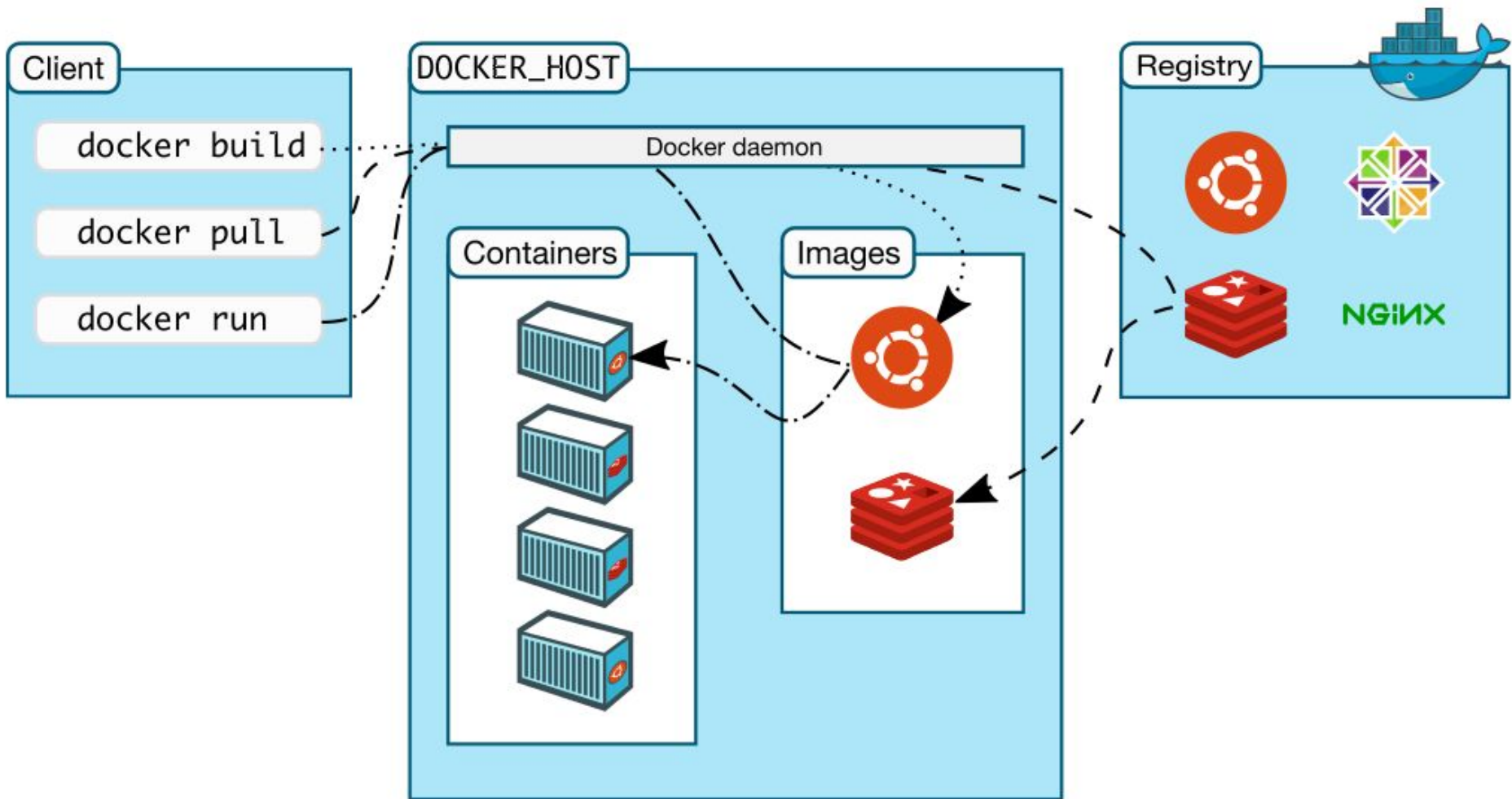
Linux VM

Docker daemon











ELEMENTOS PRINCIPAIS DOCKER

docker images

imagens

instantâneas dos
contêineres ou do
S.O. básico
(Ubuntu por
exemplo)

docker container

diretórios

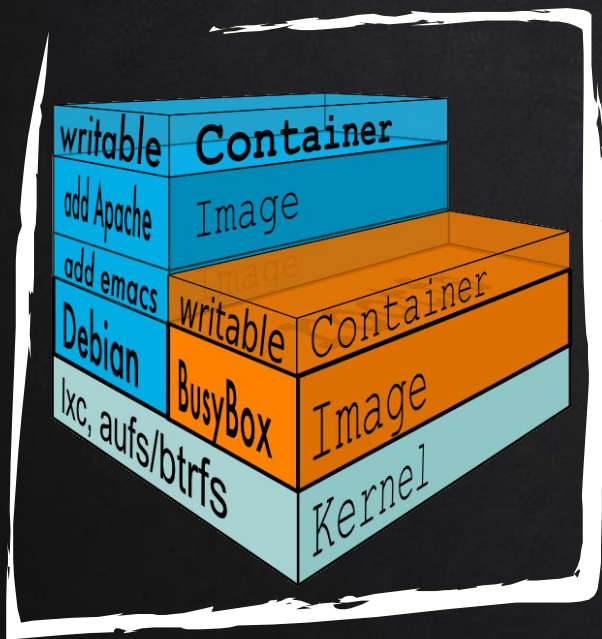
contendo tudo
que constitui sua
aplicação.

Dockerfiles

scripts que
automatizam o
processo de
construção de
imagens



DOCKER IMAGE



é uma especificação do
que o container vai
possuir quando for
executado.



DOCKER CONTAINER

Depende de uma imagem

Geram novas imagens

Conectividade com o host e outros containers

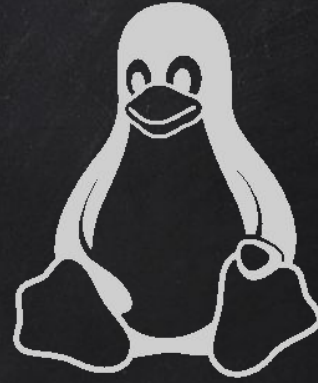
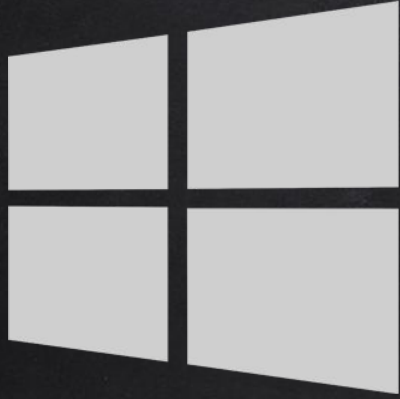
Execução controlada - CPU, RAM, I/O, etc

Descartáveis ou persistentes



INSTALAÇÃO DOCKER

Configuração do ambiente onde o docker irá rodar



amazon
web services



Docker for Windows

Docker for Mac

ou

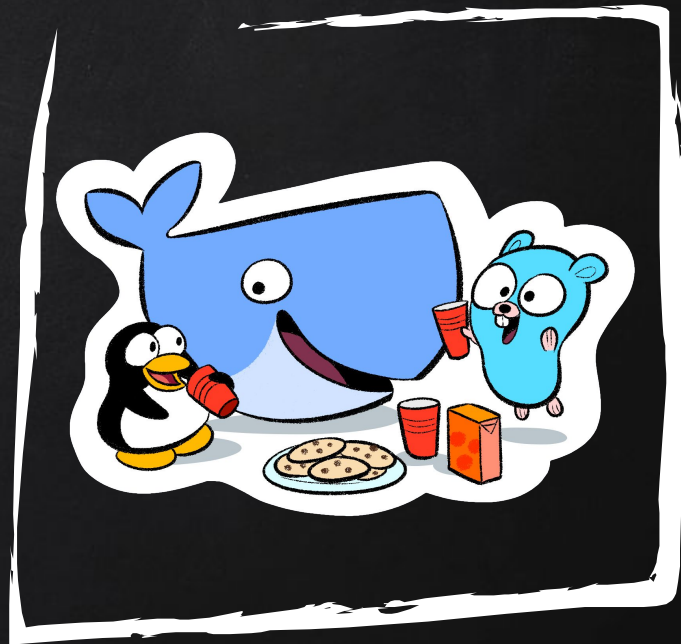
Docker Toolbox





INSTALAÇÃO NO LINUX

Apenas uma forma,
simples e funciona!



Na instalação no linux, você pode criar um grupo chamado docker e adicionar seu usuário, isso faz com que você tenha acesso e não precise ficar pedindo permissão com o sudo.

```
$ sudo groupadd docker
```

```
$ sudo usermod -aG docker $(whoami)
```

```
$ docker info
```




COMANDOS DOCKER

Trabalhando na parte CLI

Em um terminal
vamos executar:

docker version ou
docker -v
para ver a versão do
docker
instalada.

```
morvanabonin@deathnote ~$ docker -v
Docker version 1.12.0, build 8eab29e
morvanabonin@deathnote ~$ docker version
Client:
Version:      1.12.0
API version:  1.24
Go version:   go1.6.3
Git commit:   8eab29e
Built:        Thu Jul 28 22:11:10 2016
OS/Arch:      linux/amd64

Server:
Version:      1.12.0
API version:  1.24
Go version:   go1.6.3
Git commit:   8eab29e
Built:        Thu Jul 28 22:11:10 2016
OS/Arch:      linux/amd64
morvanabonin@deathnote ~$
```

docker ps

para ver os containers que estão rodando.

```
morvanabonin@deathnote ➤ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
morvanabonin@deathnote ➤						

docker ps -a

Para ver todos os containers.

Vamos rodar o comando

`docker run hello-world`
para criar um container a
partir da imagem
`hello-world`.

Tells your operating
system you are using
the **docker** program

docker

Tells Docker which
image to load into
the container

hello-world

run

A *subcommand* that
creates & runs a
Docker *container*

docker info

retorna uma lista de todos os containers, imagens, versão do docker, o drive do storage, entre outras informações.

docker start id do container ou nome do container
inicia um container

docker stop id do container ou nome do container
para um container

docker rm id do container ou nome do container
remove um container

docker build
docker commit
docker create
docker exec
docker history
docker logs
docker images

docker network
docker pull
docker restart
docker rmi
docker rename
docker search

Para ver todos os comandos que o docker possui, utilize o docker help

```
$ docker help
```

A maioria dos comandos também possuem parâmetros, e para vê-los você pode utilizar o comando

```
$ docker run --help
```



DOCKER RUN

O comando `docker run` é o comando principal do docker para criar containers a partir de imagens. Há duas formas de executar o comando `docker run`, em primeiro plano (foreground) e em segundo plano, ou como alguns chamam, desanexado (detached).

Entre as várias opções, parâmetros que o run aceita, temos dois que iremos utilizar bastante, principalmente para testes.

-i ou **--interactive** - mantém o STDIN aberto, possibilitando ver os outputs que estão acontecendo dentro do container.

-t ou **-tty** - aloca de pseudo terminal TTY para processo rodando no container.

É bem comum as opções -i e -t virem juntas.

Outras opções, parâmetros:

- rm - remove o container quando ele finaliza a execução.
- name - atribui de nome a de container.
- net - define o tipo de rede que será usada no container.
- m ou --memory - define o máximo de memória alocada para o container.
- e ou --env - define variáveis de ambiente e valores dentro do container.
- h ou --hostname - define um host específico para o container.
- u ou user - define de usuário para o container.
- w ou --workdir - define em qual diretório do container serão executados os scripts; por padrão é o /.



DOCKER VOLUME

-v ou --volume é a opção/ parâmetro ao qual é possível compartilhar de volume entre o docker host e o container.

O parâmetro --volumes-from permite montar volumes de outros containers.

Não é recomendado o uso do comando docker commit, é recomendado a construção de imagens usando o arquivo **Dockerfile** e o comando docker build.



DOCKERFILE

A receita para a criação de imagens

Dockerfile é um arquivo que contém um conjunto de instruções necessárias para se criar uma imagem Docker

```
#####  
# Dockerfile para construir uma imagem de exemplo  
# Baseada no Ubuntu  
#####  
  
# Set the base image to Ubuntu  
FROM ubuntu  
  
# Descrição de quem manterá a imagem  
MAINTAINER Morvana Bonin <morvanabonin@gmail.com>  
  
# Update nas listas do repositório  
RUN apt-get update  
  
# Upgrade no sistema  
RUN apt-get -y upgrade  
  
# Instalação do traceroute  
RUN apt-get install traceroute
```




COMANDOS DOCKERFILE

ADD

CMD

ENTRYPOINT

ENV

EXPOSE

FROM

MAINTAINER

RUN

USER

COPY

VOLUME

WORKDIR

CMD - define de comando a ser executado quando de container é criado. Pode ser sobrescrito.

ENTRYPOINT - parecido com o CMD, a diferença do comando Entrypoint é que ele não será sobrescrito.

WORKDIR - seta o diretório a ser trabalhado quando o container for instanciado.

ENV - a instrução ENV serve para setar variáveis durante o processo de construção da imagem.

USER - essa instrução serve para setar de usuário específico que a imagem deve rodar

VOLUME - adiciona de volume para qualquer container criado a partir da imagem.

ADD - adiciona arquivos ou diretórios dentro da nossa imagem.

RUN - Essa instrução é responsável por executar comandos no shell.

EXPOSE - serve para exportar as portas do container para o host.

LABEL - define label para agregar mais informações para o container.



DOCKER COMPOSE

Orquestrando seus containers



é uma ferramenta
para a criação e
execução de
múltiplos containers
de aplicação.



Com o Compose, você irá utilizar um único arquivo do tipo yaml para definir como será o ambiente de sua aplicação e usando um único comando você criará e iniciará todos os serviços definidos.

docker-compose

build

config

create

down

events

exec

help

kill

logs

pause

port

ps

pull

restart

rm

run

scale

start

stop

unpause

up

version



REFERÊNCIAS E DICAS

Documentação, livros, dicas



REFERÊNCIAS

Docker for mac <<https://docs.docker.com/docker-for-mac>>

Docker for windows <<https://docs.docker.com/docker-for-windows>>

Docker Toolbox <<https://www.docker.com/products/docker-toolbox>>

Docker fo Linux

<https://docs.docker.com/engine/getstarted/linux_install_help>



DOCKER PARA DESENVOLVEDORES



Livro sendo escrito por Rafael Gomes (@Gomex), é um livro no estilo pague o quanto quiser, ou mesmo nada. Se encontra no LeanPub.



APRENDENDO DOCKER

Livro escrito por
Wellington Figueira da
Silva, referência em
Docker, publicado pela
editora Novatec.





CONTAINERS COM DOCKER

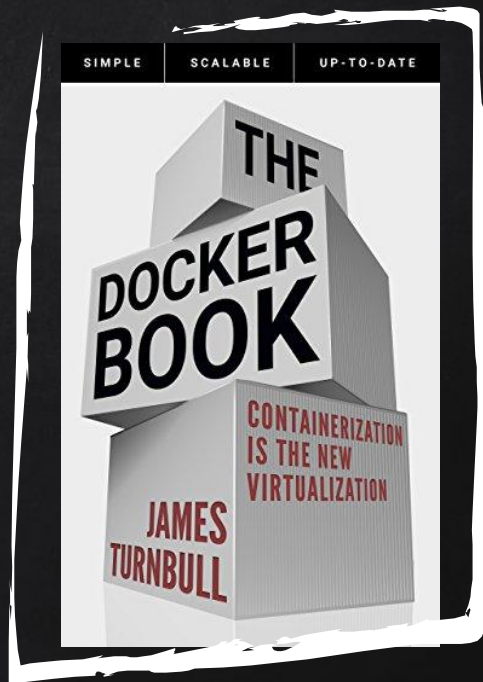


Livro escrito por Daniel Romero, publicado pela Casa do Código.



THE DOCKER BOOK

Livro escrito por James
Turnbull e vendido pela
Amazon.





DICAS

Documentação Docker [En]

<<https://docs.docker.com/>>

Awesome Docker

<<https://github.com/veggie-monk/awesome-docker>>

Lista do Telegram <<https://telegram.me/dockerbr>>