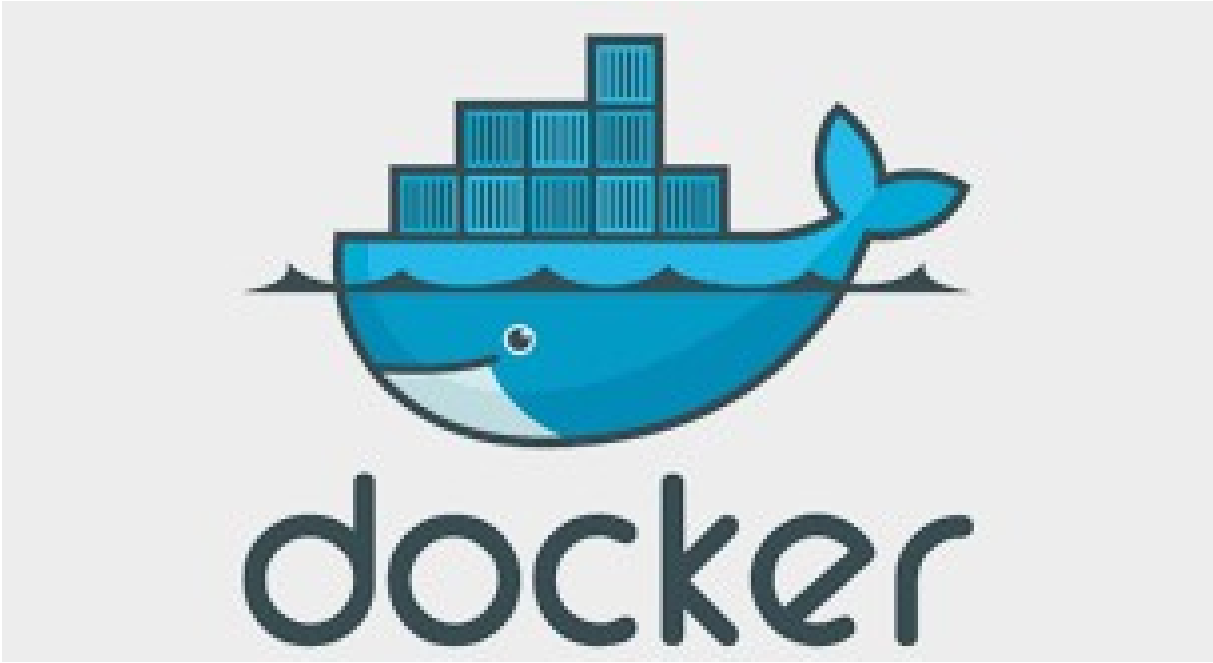
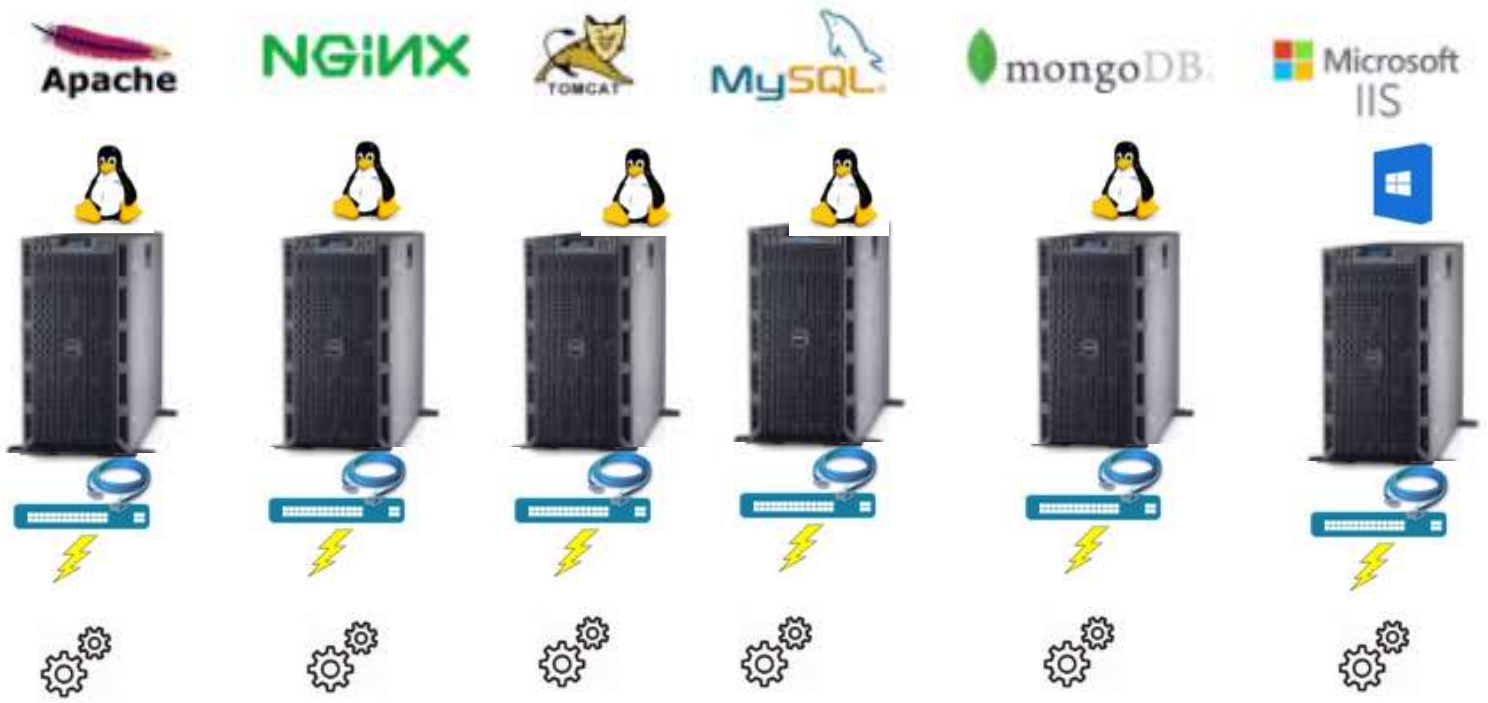


Luciano Cordeiro

Agenda

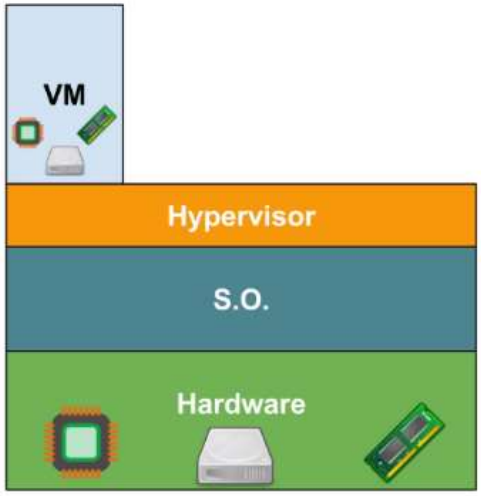
- Contexto
- O que é o Docker ?
- Tecnologias
- Comandos
- Imagens e Containers
- Volumes
- Construa suas próprias imagens
- Redes com Docker

Contexto



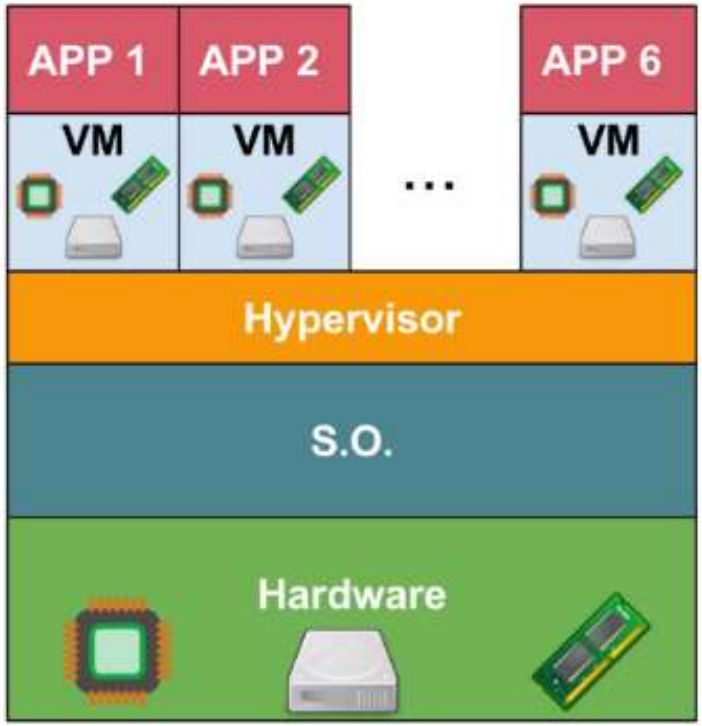
O Problema das Máquinas Virtuais

- A virtualização foi uma solução. Ou seja a utilização de **Máquinas Virtuais**. Isso só foi possível, graças a uma tecnologia chamada **Hypervisor.** Permitindo a virtualização de recursos físicos do nosso sistema operacional.



O Problema das Máquinas Virtuais

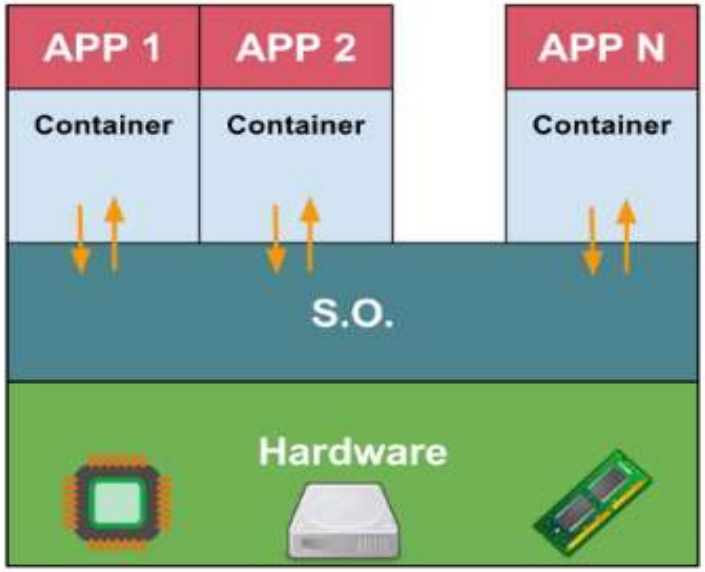
- A partir daí foi possível colocar nossas aplicações nessas VMs.



- Problemas nessa arquitetura:
  - Cada aplicação precisa de um S.O. específico
  - Custo alto de configurações
  - Tempo alto de manutenção dessas máquinas virtuais. Ao invés de se gastar tempo com o Core da empresa, gasta-se com softwares de infraestrutura

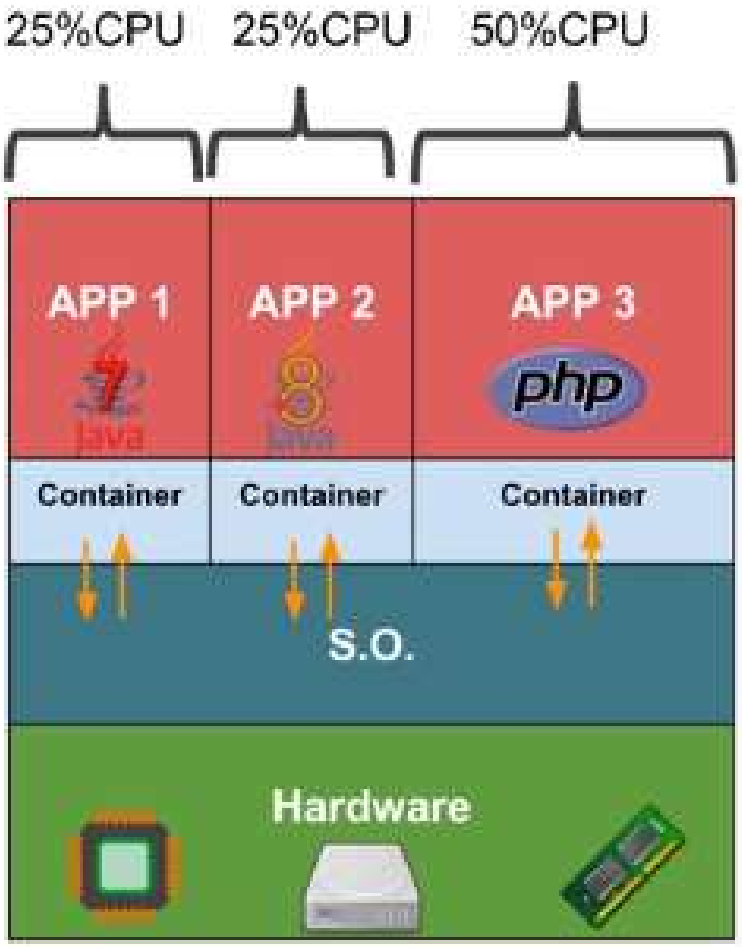
Containers

- Um container funciona junto do nosso S.O. e receberá a nossa aplicação, ou seja, a aplicação funcionará dentro dele



- Vantagens
  - Por não possuir um S.O. é muito mais leve e não possui o custo de vários S.O.s. Muito mais rápido pra subir
  - Agilidade na hora de subir ou parar um container

Containers

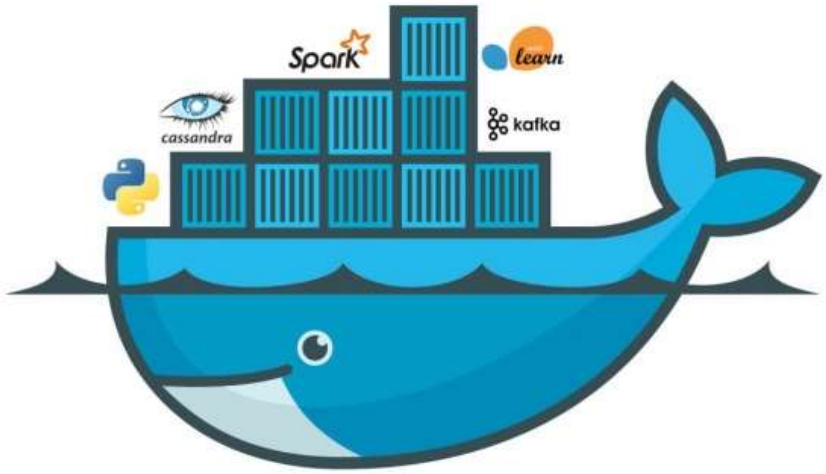


- Vantagens
  - Imagine numa mesma máquina 2 sistemas que utilizam a mesma porta de rede.
  - E se uma aplicação consumir toda a CPU de outra aplicação?
  - Uma aplicação usa Java 7 e outra usa Java 8
  - Sem contar que uma única aplicação pode congelar um ambiente
- Desvantagens
  - ????

O que é o Docker?

docker

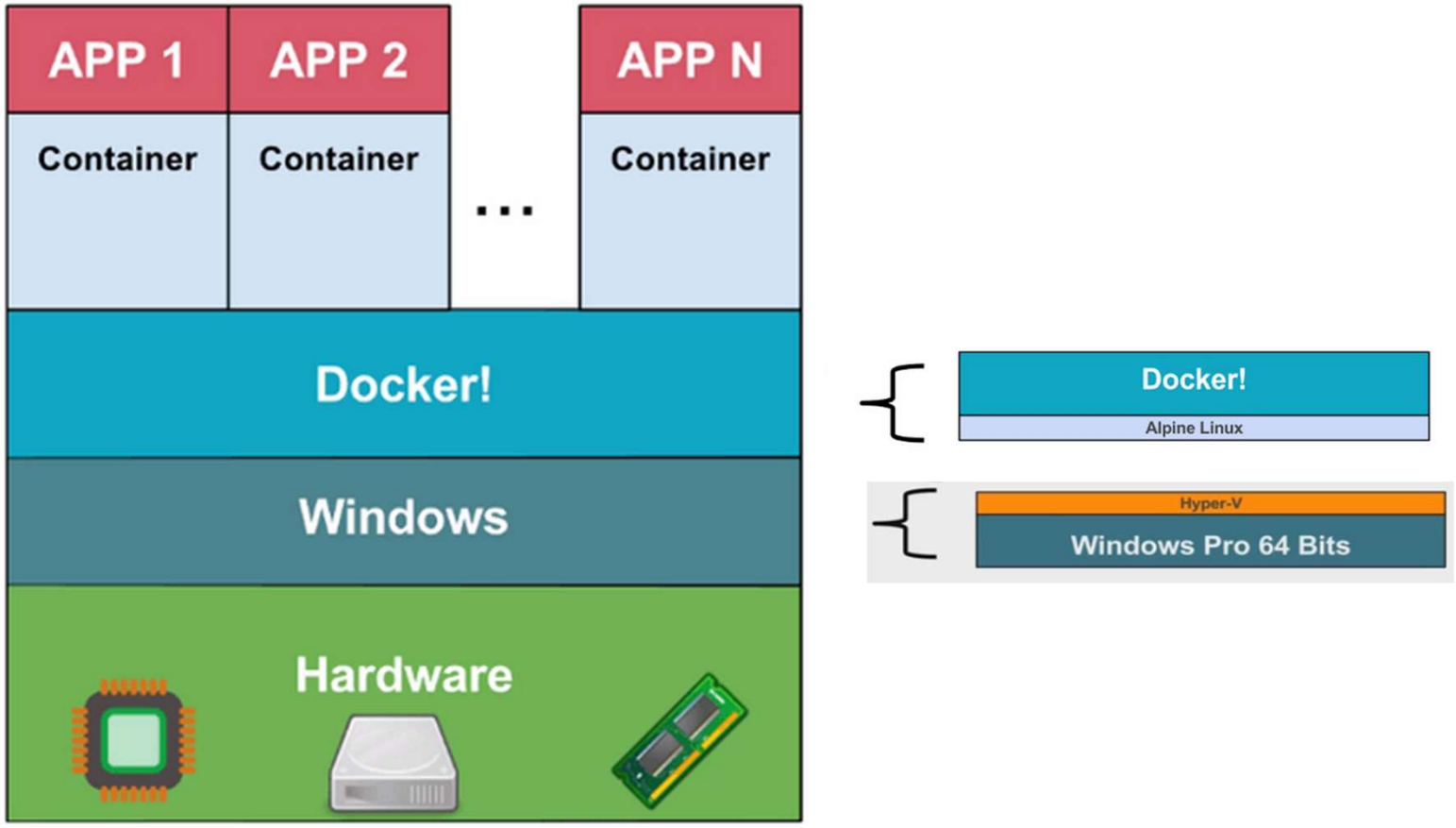
Docker Inc.
A empresa



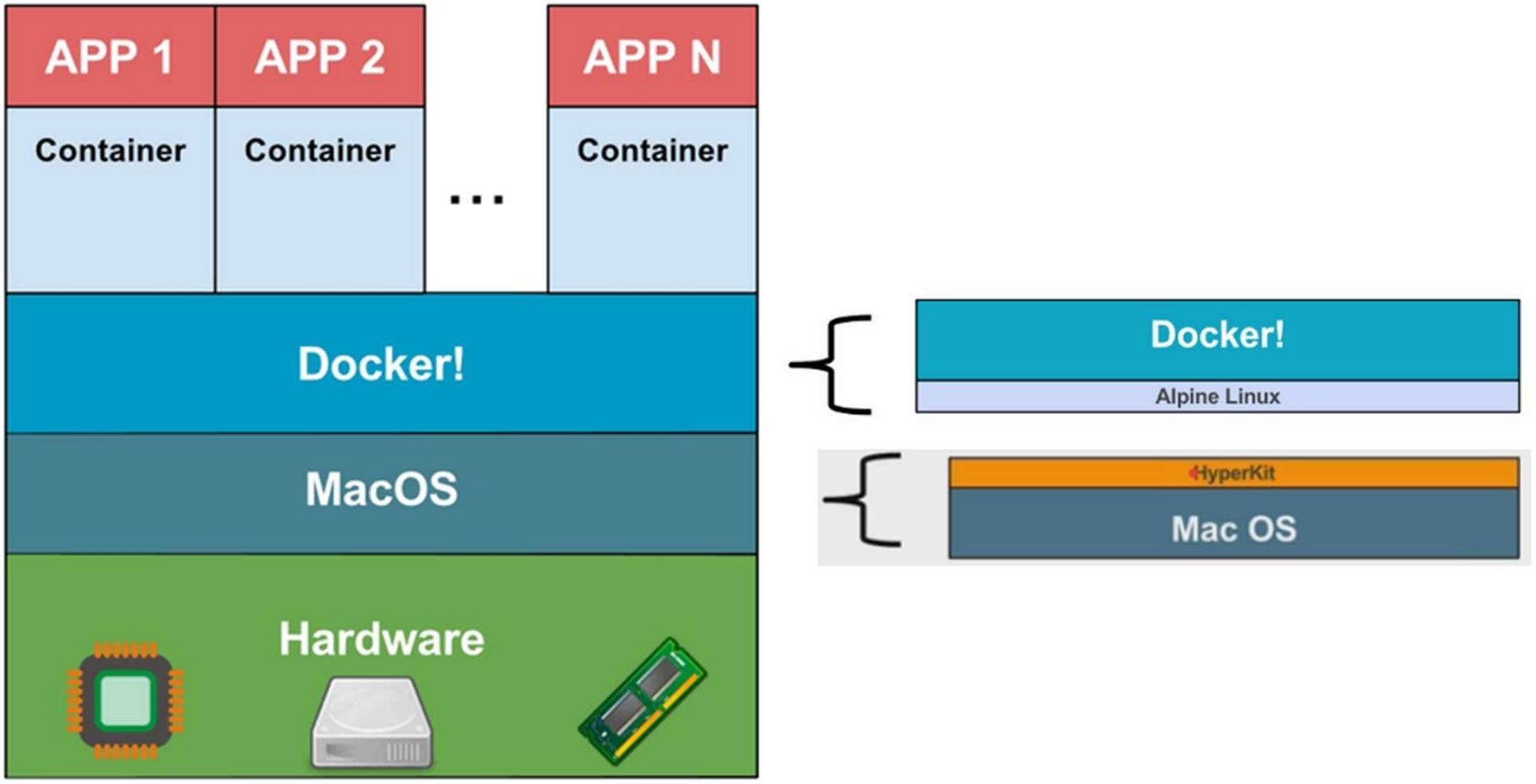
Play with Docker

https://labs.play-with-docker.com/

O que é o Docker?



O que é o Docker?



Tecnologias Docker

- **Docker Compose –** uma forma simples de orquestrar múltiplos containers.

- **Docker Swarm –** uma ferramenta para colocar múltiplos docker engines para funcionarem juntos num cluster.

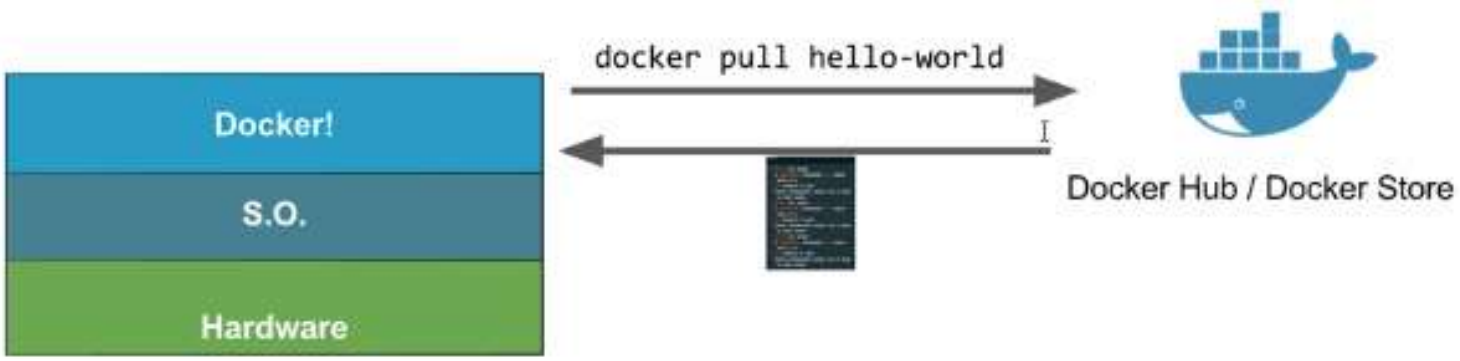
- **Docker Hub –** um repositório com mais de 250 mil imagens diferentes para os nossos containers.

- **Docker Machine –** uma ferramenta que nos permite gerenciar o Docker num Host Virtual.

https://github.com/docker

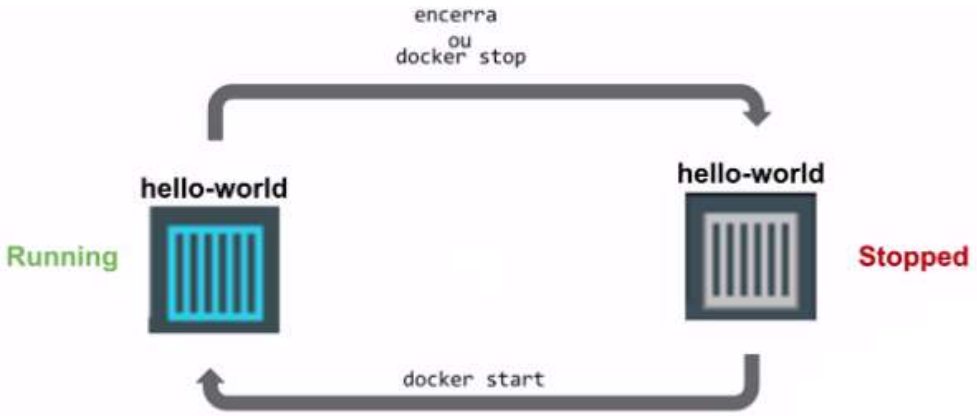
Imagens e Containers

A **imagem** é como se fosse uma receita de bolo, uma série de instruções que o Docker seguirá para criar um ***container***, que irá conter as instruções da imagem.



Comandos Básicos

- docker run ubuntu echo “ Olá Mundo!”
- docker run -it ubuntu - **Trabalhar dentro do Container**
- docker start –help - **Ajuda sobre comandos**
- docker start 4139842e283a – **Iniciar um container**
- docker stop 4139842e283a – **Parar um container**
- docker start –a –i 4139842e283a – **Iniciar um container e já acessá-lo (-a attach** - integrar os terminais **/ -i interactive** – interagir com o terminal**)**
  - root@4139842e283a



Comandos Básicos

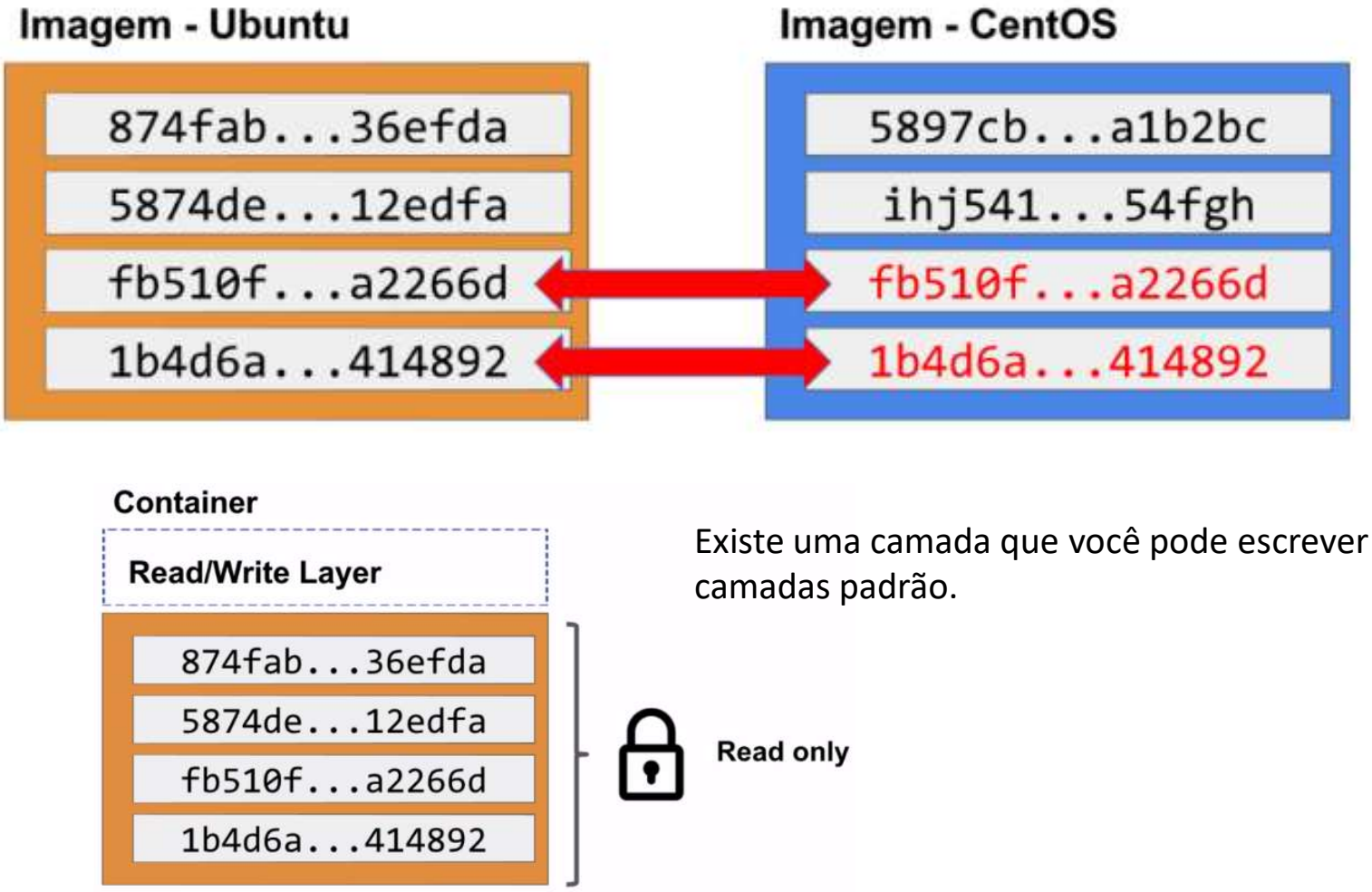
Alguns comandos :

- Remover um Container
  docker rm 9daa6a5cd330

- Remover todos os containers parados
  docker container prune

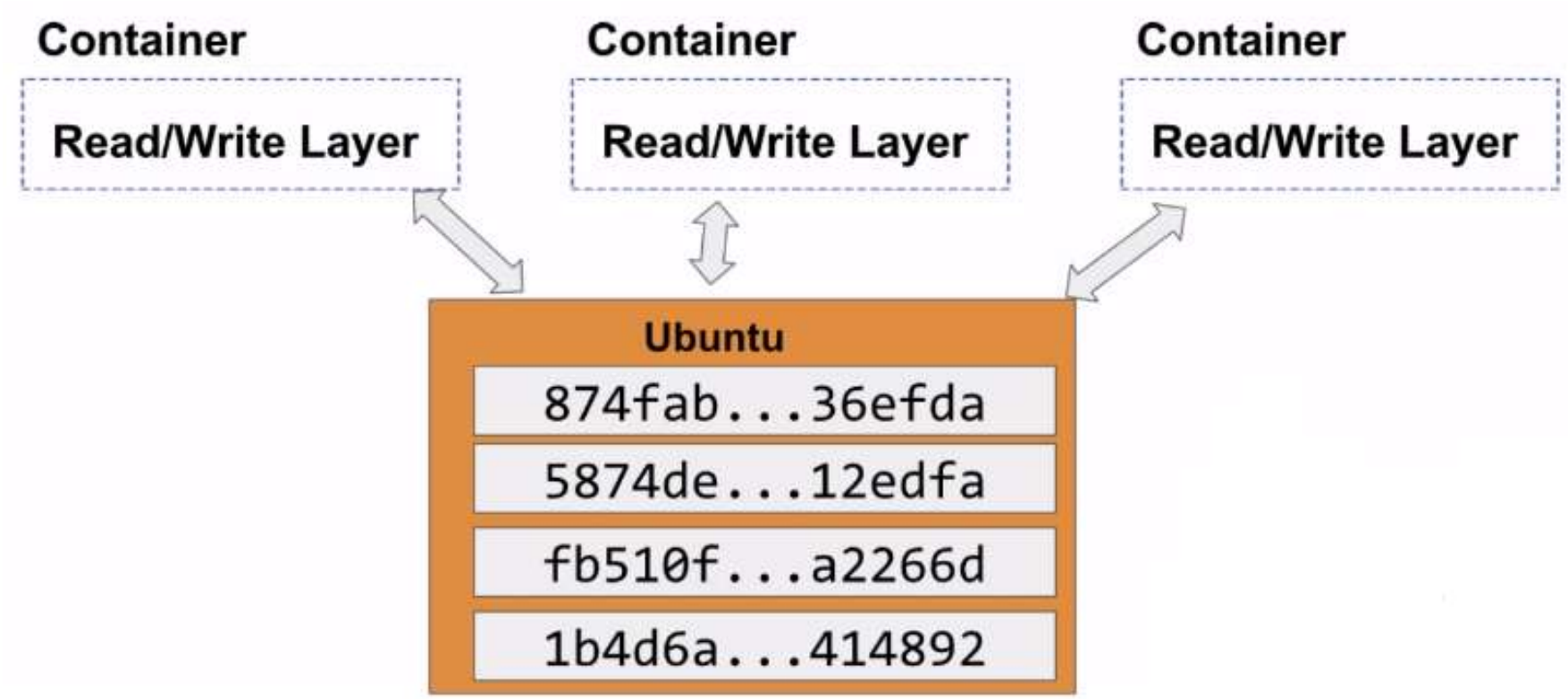
- Remover uma Imagem
  docker rmi hello-world

Layered File System



Existe uma camada que você pode escrever em cima das camadas padrão.

Layered File System



Comandos do Docker

- Vamos criar um Container que dará suporte a Site Estático:

  - **Executando em modo Detached ->** docker run **-d** dockersamples/static-site
    Agora o container fica executando em segundo plano.

Mas como fazer pra acessar o Site Estático?

Comandos do Docker

- Qual porta acessar? Para isso podemos usar a flag **–P** que atribuirá portas aleatórias que farão com que o mundo externo(nossa máquina) se comunique com o container
  - docker run **–d –P** dockersamples/static-site
- Podemos visualizar também as portas através do comando: **docker port container**
  - docker **port** 989e4d7d3638

    PORTS
    0.0.0.0:9001->80/tcp, 0.0.0.0:9000->443/tcp
- ***IMPORTANTE***: Se você está utilizando o Docker no Windows pelo **Docker ToolBox**, ele está rodando em cima de uma Máquina Virtual. Portanto, não será o IP da máquina local (localhost) e sim da VM. Para isso, rode o comando:
  - **docker-machine ip**
    - **192.168.98.112**

http://192.168.98.112:32769/

Comandos do Docker

- ***Nomeando um container – Para facilitar a localização do container posteriormente***
  - docker run –d –P **--name meu-site** dockersamples/static-site
  - Exemplo: Facilitaria para parar esse container posteriormente. -> **docker stop meu-site**

- ***Definindo uma porta específica***
  - docker run –d **–p 12345:80** dockersamples/static-site

- ***Atribuindo uma variável específica***
  - docker run –d –p 12345:80 **-e AUTHOR="Luciano Cordeiro"** dockersamples/static-site

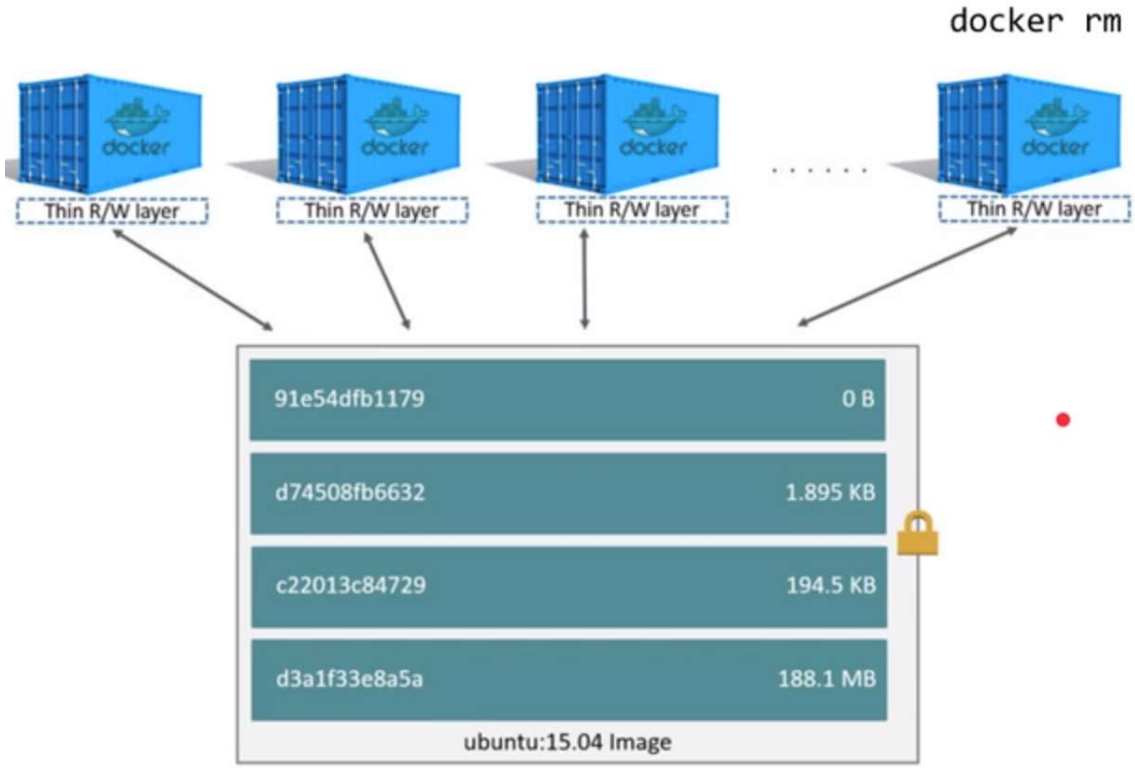
http://192.168.98.112:12345/ 

- ***Parando todos os containers de uma vez***
  - docker stop –t 0 $(docker ps –q)

Revisão Comandos do Docker

- **docker ps** - exibe todos os containers em execução no momento.
- **docker ps -a** - exibe todos os containers, independentemente de estarem em execução ou não.
- **docker run -it NOME_DA_IMAGEM** - conecta o terminal que estamos utilizando com o do container.
- **docker start ID_CONTAINER** - inicia o container com id em questão.
- **docker stop ID_CONTAINER** - interrompe o container com id em questão.
- **docker start -a -i ID_CONTAINER** - inicia o container com id em questão e integra os terminais, além de permitir interação entre ambos.
- **docker rm ID_CONTAINER** - remove o container com id em questão.
- **docker prune** - remove todos os containers que estão parados.
- **docker rmi NOME_DA_IMAGEM** - remove a imagem passada como parâmetro.
- **docker run -d -P --name NOME dockersamples/static-site** - ao executar, dá um nome ao container.
- **docker run -d -p 12345:80 dockersamples/static-site** - define uma porta específica para ser atribuída à porta 80 do container, neste caso 12345.
- **docker run -d -e AUTHOR="Fulano" dockersamples/static-site** - define uma variável de ambiente AUTHOR com o valor Fulano no container criado.

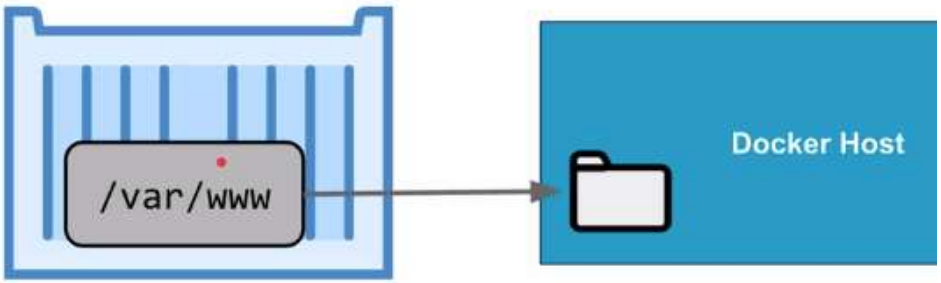
Volumes



- Se você escrever algo nesse container e este for removido, todo o conteúdo será perdido. Por isso é interessante que você utilize o conceito de **Volumes**

Volumes

- A solução de Volumes faz com que ao se criar um conteúdo numa pasta, por exemplo, “/var/www” esse conteúdo vá para o **Docker Host,** fazendo com que este conteúdo não se perca se o container for apagado.



- **docker run -v "/var/www" ubuntu**
  - Dessa forma vc precisa rodar um **docker inspect <código container>** para verificar o “Mounts”


```
"Mounts": [{
        "Type": "volume",
        "Name": "5e1cbfd48d07284680552e56087c9d5196659600ccd6874bfa3831b51ddd0576",
        "Source": "/var/lib/docker/volumes/5e1cbfd48d07284680552e56087c9d5196659600ccd6874bfa3831b51ddd0576/_data",
        "Destination": "/var/www",
        "Driver": "local",
        "Mode": "",
        "RW": true,
        "Propagation": ""
    }
]
```

Volumes

- A pasta que é gerada no Docker Host pode ser configurada

  - docker run -it -v "C:\Users\Luciano\Desktop:/var/www" ubuntu
  - root@abd0286c0083:/#

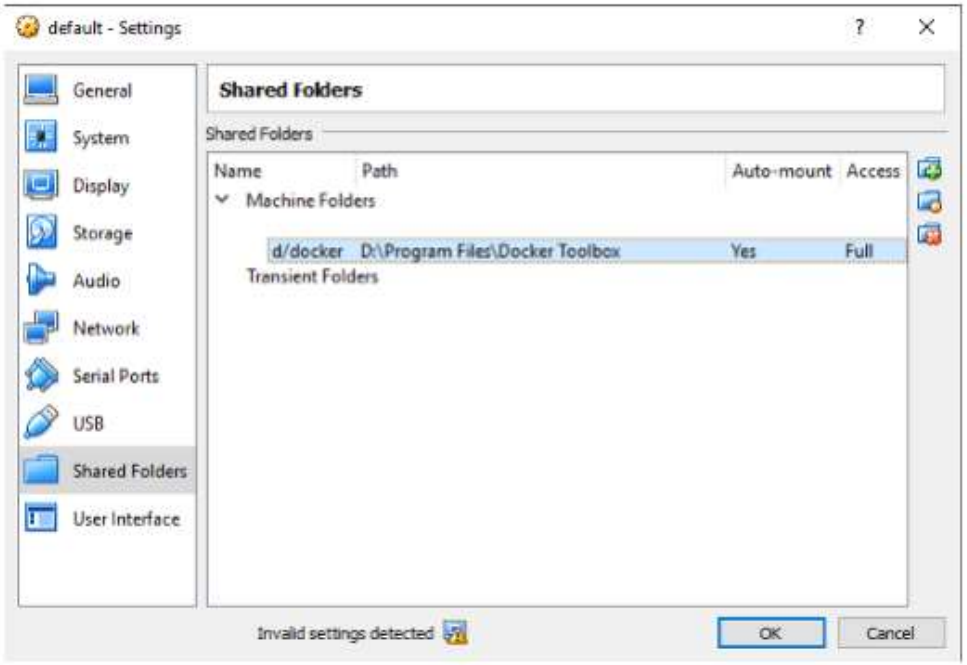
  - root@6ec39d5e9175:/# cd /var/www/
  - root@6ec39d5e9175 :/var/www# touch arquivo1.txt
  - root@6ec39d5e9175 :/var/www# echo "Este arquivo foi criado dentro de um volume" > arquivo1.txt

Volumes no Docker Toolbox

- Detalhe Importante: Se você estiver rodando no Docker Toolbox

  - docker run -it -v "//c/Users/Luciano/Desktop:/var/www" ubuntu
    - (https://medium.com/@Charles_Stover/fixing-volumes-in-docker-toolbox-4ad5ace0e572)

- Além disso é necessário mapear na sua Virtual Box a pasta que será compartilhada com o volume do Container



Rodando código em um Container

- Um exemplo que roda em **Node.js** e não possuímos o Node na máquina.
  - docker run -d -p 8080:3000 -v "//c/Users/<<Seu caminho local>>/volume-exemplo:/var/www" -w "/var/www" node npm start

Ou

  - docker run -d -p 8080:3000 -v “$(pwd):/var/www" -w "/var/www" node npm start

Você conseguirá executar o código em http://localhost:8080

- Nessa pasta volume-exemplo existe um pequeno projeto **Node.js**
- **“-d”** – executa em modo dettached e libera o terminal
- **“-w”** – “Working Directory” informa em qual pasta o comando deve ser executado dentro do container

Lembrando que se você estiver rodando num Windows com uma máquina virtual e rodando o Docker Toolbox, o ip será descoberto pelo “**docker-machine ip” e você executará o código com o endereço ip encontrado e a porta configurada, nesse caso a 8080.**

http://192.168.99.123:8080/

Construir as Próprias Imagens

- ***Criando um Dockerfile –*** O Dockerfile define comandos para executar instalações complexas e com características específicas.

```
# Poderia ter outros nomes como: node.dockerfile, se vc tivesse vários Dockerfiles
FROM node:latest
MAINTAINER Luciano Cordeiro
COPY . /var/www
WORKDIR /var/www
RUN npm install
ENTRYPOINT npm start
EXPOSE 3000
```

- <u>Lembrando:</u>
  - *As imagens são sempre read-only*
  - *Um container é uma instância de uma imagem*
  - *Para guardar as alterações a docker engine cria uma nova layer em cima da última layer da imagem*

Revisão Comandos sobre Criações de Imagens

- **docker build –f Dockerfile** – cria uma imagem a partir de um Dockerfile.
- **docker build –f CAMINHO_DOCKERFILE/Dockerfile –t NOME_USUARIO/NOME_IMAGEM** – constrói e nomeia uma imagem não oficial informando o caminho para o Dockerfile.
  - ➢ docker build -f Dockerfile -t lacsousa/node .

    - Depois do Build executado, podemos ver a imagem criada com o "**docker images**".
    - Podemos também executar essa imagem rodando como um container

  - ➢ docker run –d –p 8080:3000 lacsousa/node

- Após a criação da imagem inicia o processo para "subir" a imagem no Docker Hub
- **docker login** – inicia o processo de login no Docker Hub.
- **docker push NOME_USUARIO/NOME_IMAGEM** – envia a imagem criada para o Docker Hub.
- **docker pull NOME_USUARIO/NOME_IMAGEM** – baixa a imagem criada do Docker Hub.

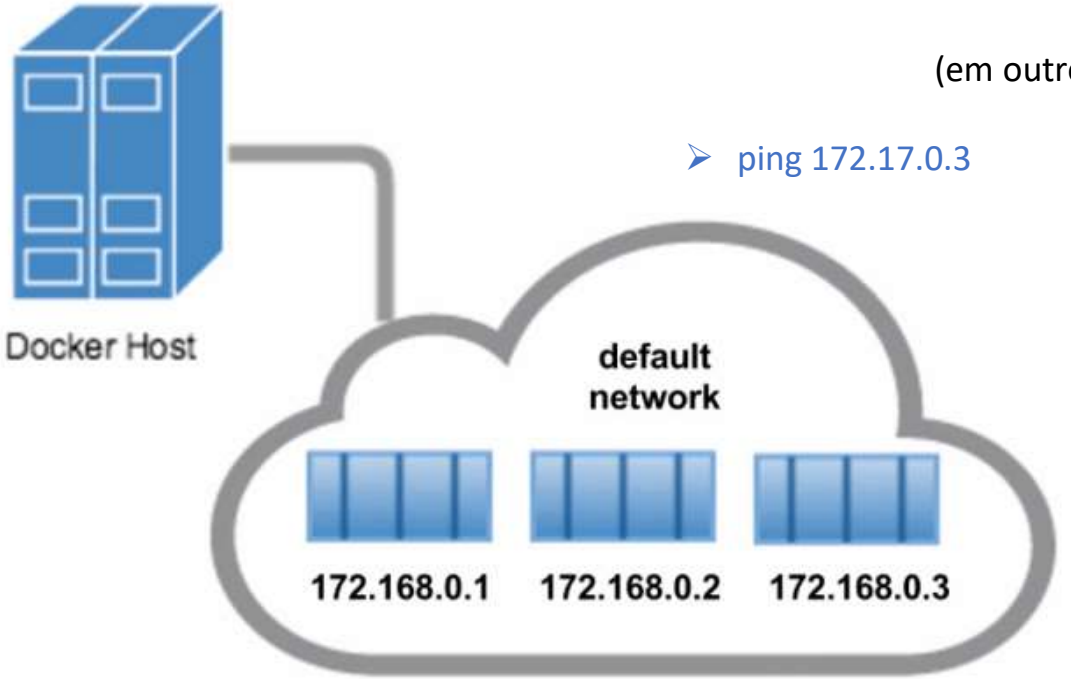
Redes com Docker

- Por padrão o Docker já cria uma **default network**.

  - docker run -it ubuntu
  - root@d11e0d244c29:/# hostname -i
    172.17.0.3
  - root@e6b45f6e15d1:/# apt-get update && apt-get install iputils-ping

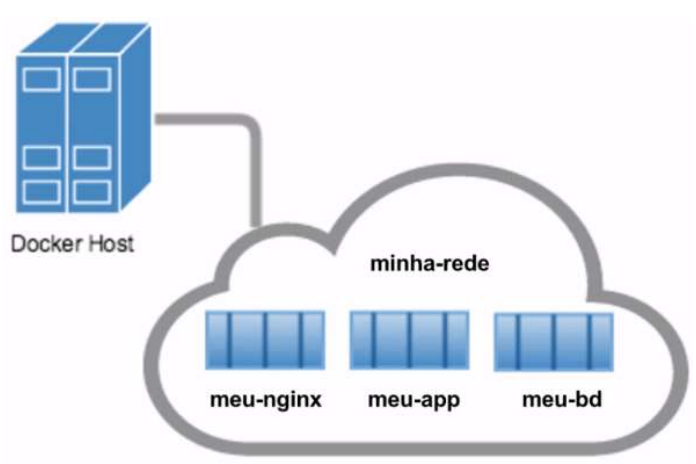
(em outro terminal posso dar um ping para um IP específico

  - ping 172.17.0.3



Redes com Docker

- O problema é que o Docker cria um IP dinâmico todas as vezes que você "subir" um container. É Interessante "nomear" sua rede e criar sua própria rede.
  - P.S. – Só posso efetuar ping entre containers em redes criadas por mim



➢ C:\>docker network ls

NETWORK ID	NAME	DRIVER	SCOPE
65d5d9207ba6	bridge	bridge	local
da7810516736	host	host	local
d09a467d9ef4	none	null	local

➢ C:\>docker network create --driver bridge minha-rede

64e168dcaade06f06e53db5f4e32693fa0cebcdac01a0075e9f5747b404a5594

➢ C:\>docker network ls

NETWORK ID	NAME	DRIVER	SCOPE
65d5d9207ba6	bridge	bridge	local
da7810516736	host	host	local
64e168dcaade	minha-rede	bridge	local
d09a467d9ef4	none	null	local

➢ C:\>

Redes com Docker

- Quando você disponibilizar uma aplicação pelo Docker é interessante que você coloque os containers na mesma rede, através da flag **--network**.

  - docker run -it --name meu-container-de-ubuntu --network minha-rede ubuntu

- Agora, se executarmos o comando **docker inspect meu-container-de-ubuntu**, podemos ver em **NetworkSettings** o container está na rede **minha-rede**. E para testar a comunicação entre os containers na nossa rede, vamos abrir outro terminal e criar um segundo container:

  - docker run -it --name segundo-ubuntu --network minha-rede ubuntu

- Agora, no segundo-ubuntu, instalamos o ping e testamos a comunicação com o meu-container-de-ubuntu:

  - root@00f93075d079:/# ping meu-container-de-ubuntu

```
PING meu-container-de-ubuntu (172.18.0.2) 56(84) bytes of data.
64 bytes from meu-container-de-ubuntu.minha-rede (172.18.0.2): icmp_seq=1 ttl=64 time=0.210 ms
64 bytes from meu-container-de-ubuntu.minha-rede (172.18.0.2): icmp_seq=2 ttl=64 time=0.148 ms
64 bytes from meu-container-de-ubuntu.minha-rede (172.18.0.2): icmp_seq=3 ttl=64 time=0.138 ms
--- meu-container-de-ubuntu ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.138/0.165/0.210/0.033 ms
```

Referências Bibliográficas

- Curso Alura : **Docker: Criando containers sem dor de cabeça**