

# Docker

---

Conteinerização de Aplicações na Prática

Vamberto Rocha Jr

[vambertojr@gmail.com](mailto:vambertojr@gmail.com)

# Sobre

- **Graduado em Ciências da computação pelo UNIPÊ**
- **Pós-Graduado em Segurança da Informação pela I2P**
- **Experiências Profissionais**
  - 2008 - Administrador de Rede na Lojas Maia**
  - 2010 - Administrador de Sistemas na CTIS**
  - 2014 - Analista de TI na UFRPE**



# Agenda

- Motivação
- Conceitos básicos
- Componentes do Docker
  - Docker CLI
  - Imagem
  - Docker Hub
  - Dockerfile
  - Contêiner
  - Docker-Compose
  - Certificação
- Execícios

# Motivação

- Padronização
- Portabilidade
- Escalabilidade
- Agilidade
- Economia de recursos

# Docker

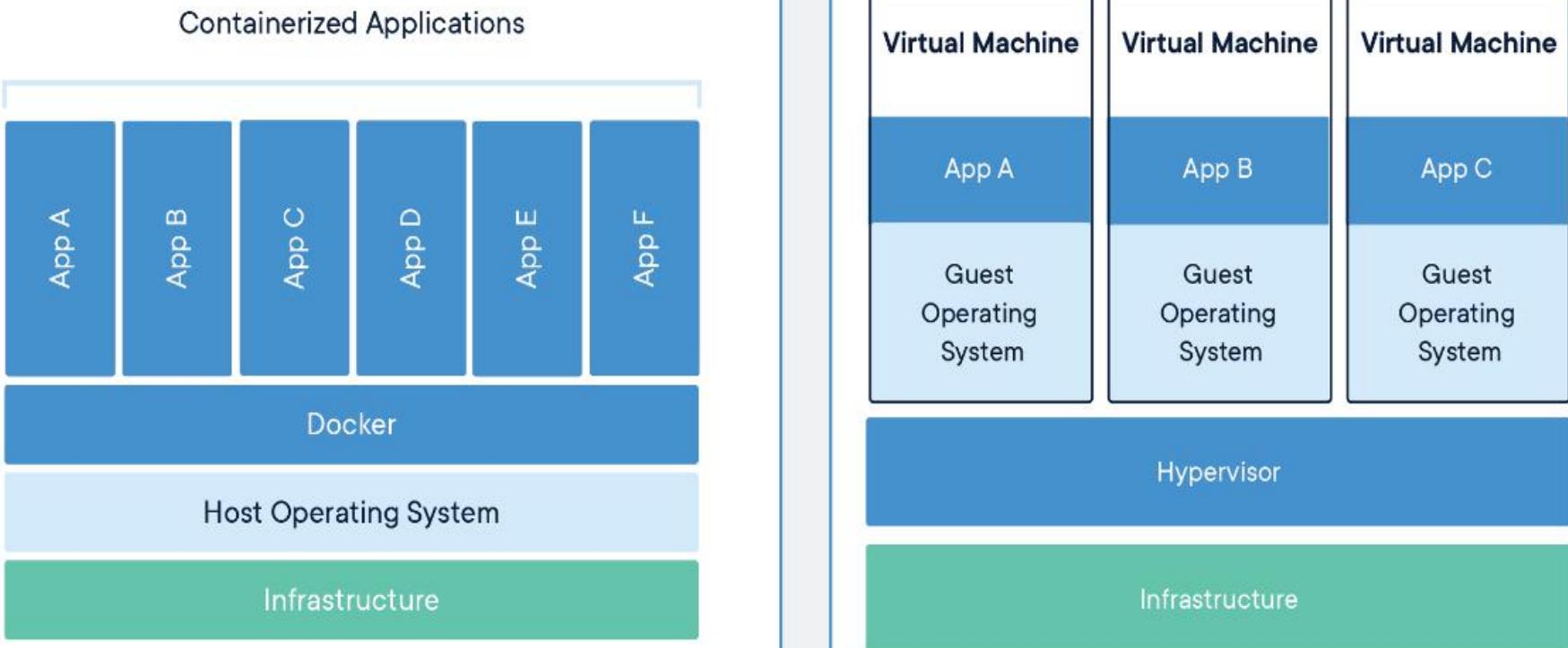
- É uma tecnologia desenvolvida para tornar fácil a criação, deploy, e a execução de aplicações fazendo o uso de contêineres
- Criado por Solomon Hykes em 2013 e atualmente mantido pela Docker Inc
- Pode ser usado por desenvolvedores e administradores de sistemas para facilitar e agilizar a execução de aplicações.

# Docker

- Com o Docker, é possível lidar com os contêineres como se fossem máquinas virtuais modulares e extremamente leves.
- Pode ser utilizado em diversos sistemas operacionais: Linux, Windows, MacOS, Cloud
- Para instalar no Linux  
`wget -qO- https://get.docker.com/ | sh`
- Documentação sobre a instalação  
<https://docs.docker.com/install/>

# Docker

## Containerização X Virtualização



# Docker

- Possui 2 versões: Docker Enterprise Edition e Docker Enterprise Edition

## Community Edition (CE)

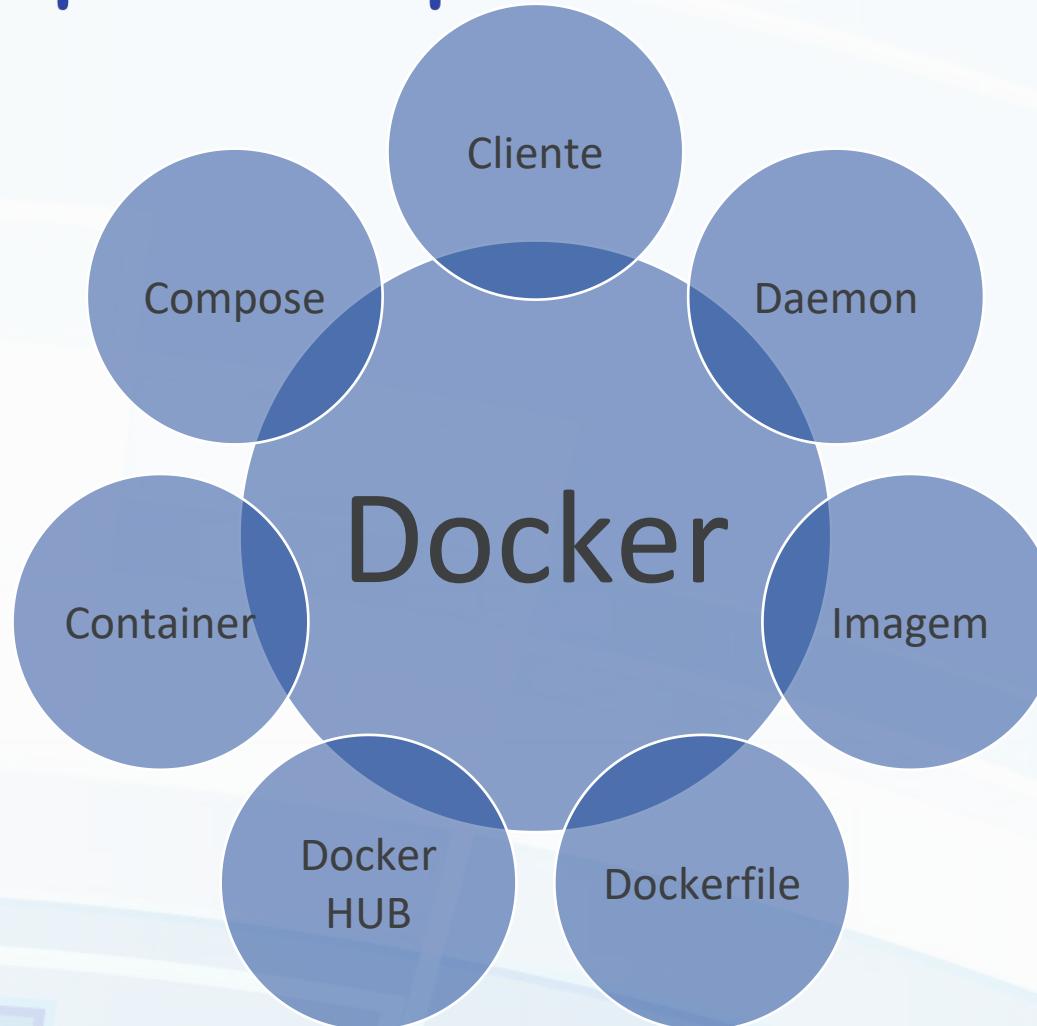
- Plataforma Docker grátis para o "faça você mesmo" dev e ops .
- Release mensal com recursos mais recentes para desenvolvedores.
- Versão trimestral com manutenção para operações.

## Enterprise Edition (EE)

- CaaS que permite assinatura da plataforma (orquestração de container integrada, gerenciamento e segurança) .
- Suporte Enterprise.
- Releases trimestrais, suportados por um ano cada com correções e hotfixes.
- Tecnologia Certificada: Infraestrutura, Plug-ins, Containers.

# Docker

## ■ Principais componentes



# Docker

## ■ Namespaces e cgroups (isolamento)

pid

- independent PIDs

network

- Own IP addresses
- routing

mnt

- Mount points

ipc

- Inter Process Communication

uts

- Isolate Kernel

cgroups

- Limits CPU, memory, disk IO

# Certificação

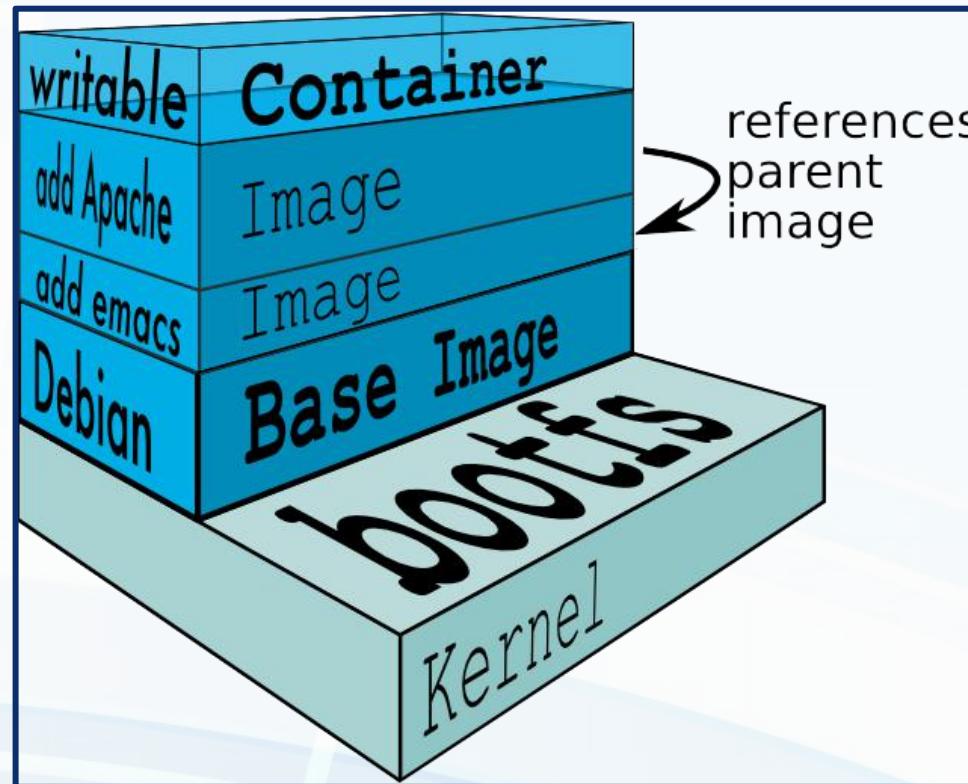
- São 55 questões de multipla escolha em 90 minutos de prova
- Necessário ter um computador com Windows ou IOS
- Prova em inglês
- USD \$195 ou Euro €175
- Resultados liberados imediatamente
- Possui 2 anos de validade

# Imagen

- Uma Imagem estabelece a base de contêineres Docker
- A partir de imagens conseguimos criar nossos contêineres
- Podemos criar diversos contêineres a partir de uma única imagem
- As imagens podem ser baixadas do Docker Hub ou criadas com o Dockerfile

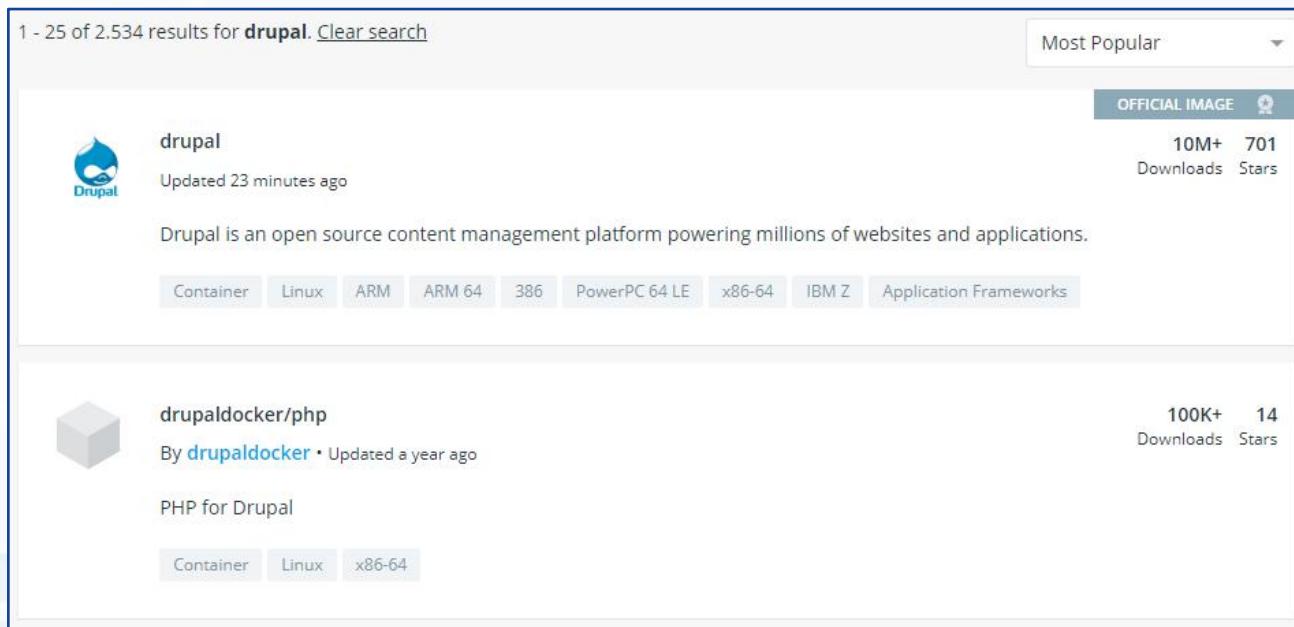
# Imagen

- As imagens são constituídas em camadas, sendo a camada 0 conhecida como imagem base



# Docker Hub

- O Docker Hub é um repositório baseado em nuvem no qual usuários e parceiros do Docker criam, testam, armazenam e distribuem imagens Docker
- Possui diversos repositórios oficiais e não oficiais



- <https://hub.docker.com/>

# Docker Hub

 drupal ☆

[Docker Official Images](#)

Drupal is an open source content management platform powering millions of websites and applications.

 10M+

Container Linux ARM ARM 64 386 PowerPC 64 LE x86-64 IBM Z Application Frameworks

Official Image

Copy and paste to pull this image

`docker pull drupal` 

[View Available Tags](#)

DESCRIPTION	REVIEWS	TAGS
<h3>Supported tags and respective Dockerfile links</h3> <ul style="list-style-type: none"><li>8.7.10-apache , 8.7-apache , 8-apache , apache , 8.7.10 , 8.7 , 8 , latest</li><li>8.7.10-fpm , 8.7-fpm , 8-fpm , fpm</li><li>8.7.10-fpm-alpine , 8.7-fpm-alpine , 8-fpm-alpine , fpm-alpine</li><li>8.6.18-apache , 8.6-apache , 8.6.18 , 8.6</li></ul>		

Be the first to give insight into your experience by rating and reviewing the product.

Add Product Review

Select a product tier 

[Start Your Review](#)

# Docker Hub

- As TAGs no Docker definem informações sobre a versão e os atributos de uma imagem

## Supported tags and respective Dockerfile links

- 8.7.10-apache , 8.7-apache , 8-apache , apache , 8.7.10 , 8.7 , 8 , latest
- 8.7.10-fpm , 8.7-fpm , 8-fpm , fpm
- 8.7.10-fpm-alpine , 8.7-fpm-alpine , 8-fpm-alpine , fpm-alpine
- 8.6.18-apache , 8.6-apache , 8.6.18 , 8.6
- 8.6.18-fpm , 8.6-fpm

# Docker Hub

- Caso uma TAG não seja definida no momento do download da imagem, a TAG “latest” será utilizada

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
drupal	latest	358d46777a73	6 days ago	449MB
php	7.2-apache	c4901971dbc3	12 days ago	410MB
traefik	v2.0	1d7c7c165196	2 weeks ago	69.3MB
remote_host	latest	cbc5674fb47	4 weeks ago	284MB
gitlab/gitlab-ce	latest	6d75d1ad2f82	5 weeks ago	1.8GB
jenkins/jenkins	lts	5721a6cac43c	2 months ago	572MB
nginx	alpine	4d3c246dfef2	2 months ago	21.2MB
centos	7	67fa590cfcc1c	3 months ago	202MB
google/cadvisor	latest	eb1210707573	12 months ago	69.6MB

# Docker Hub

- Alguns Reppositórios oficiais do Docker Hub

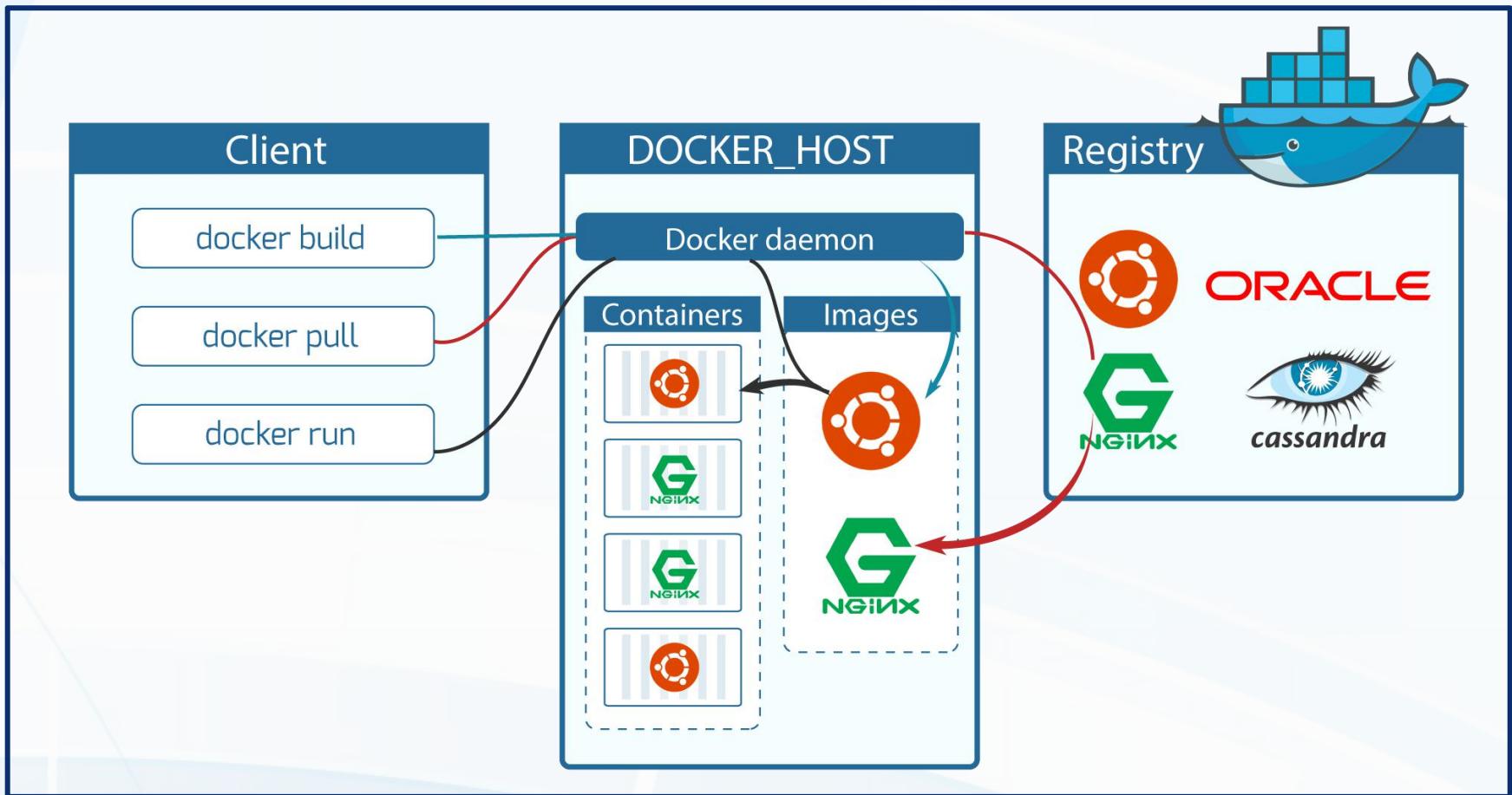


# Contêineres

- **Contêineres são instâncias criadas à partir de imagens**
- **Compartilham algumas partes do kernel do host onde o Docker está instalado**
- **Um contêiner é um ambiente isolado**
- **Por ser extremamente leve, se torna muito rápido para inicializar as aplicações**

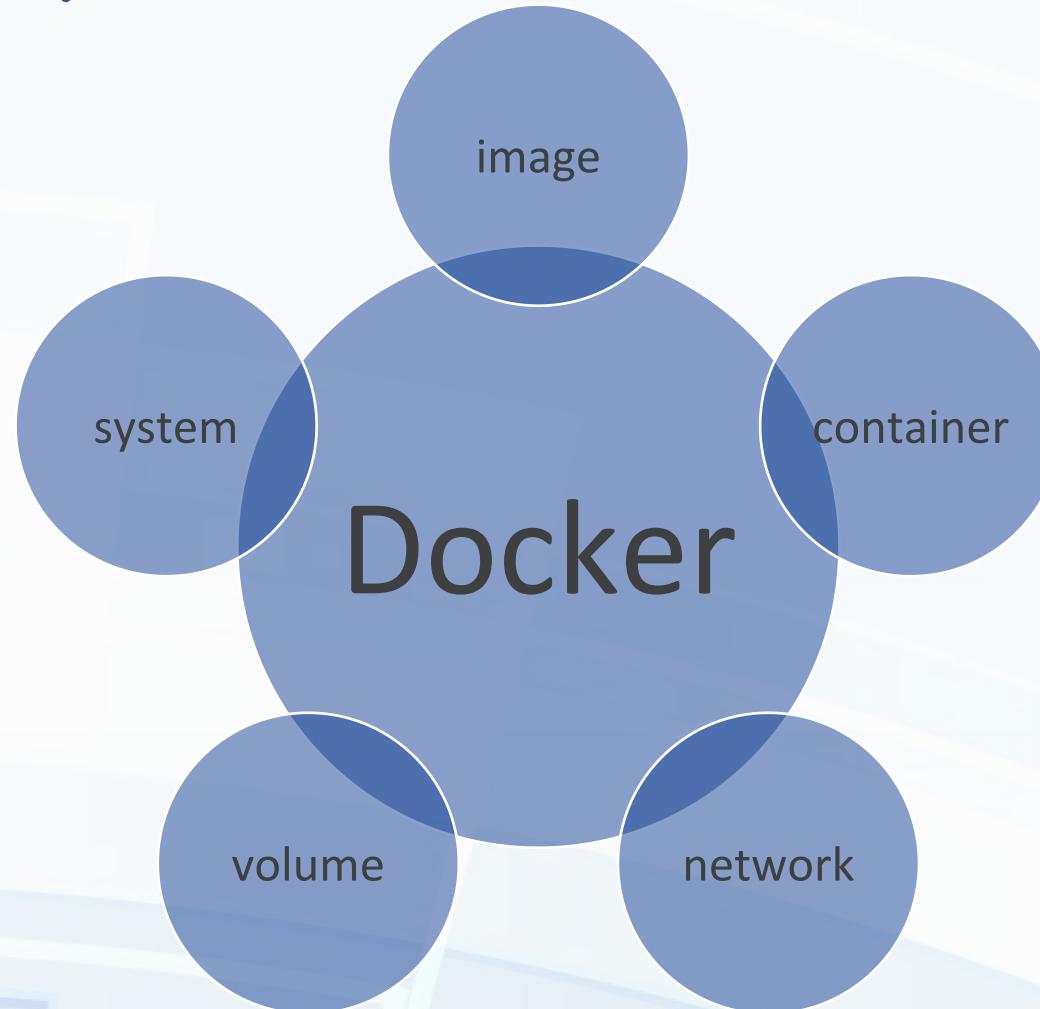
# Contêineres

- Criando um contêiner no Docker



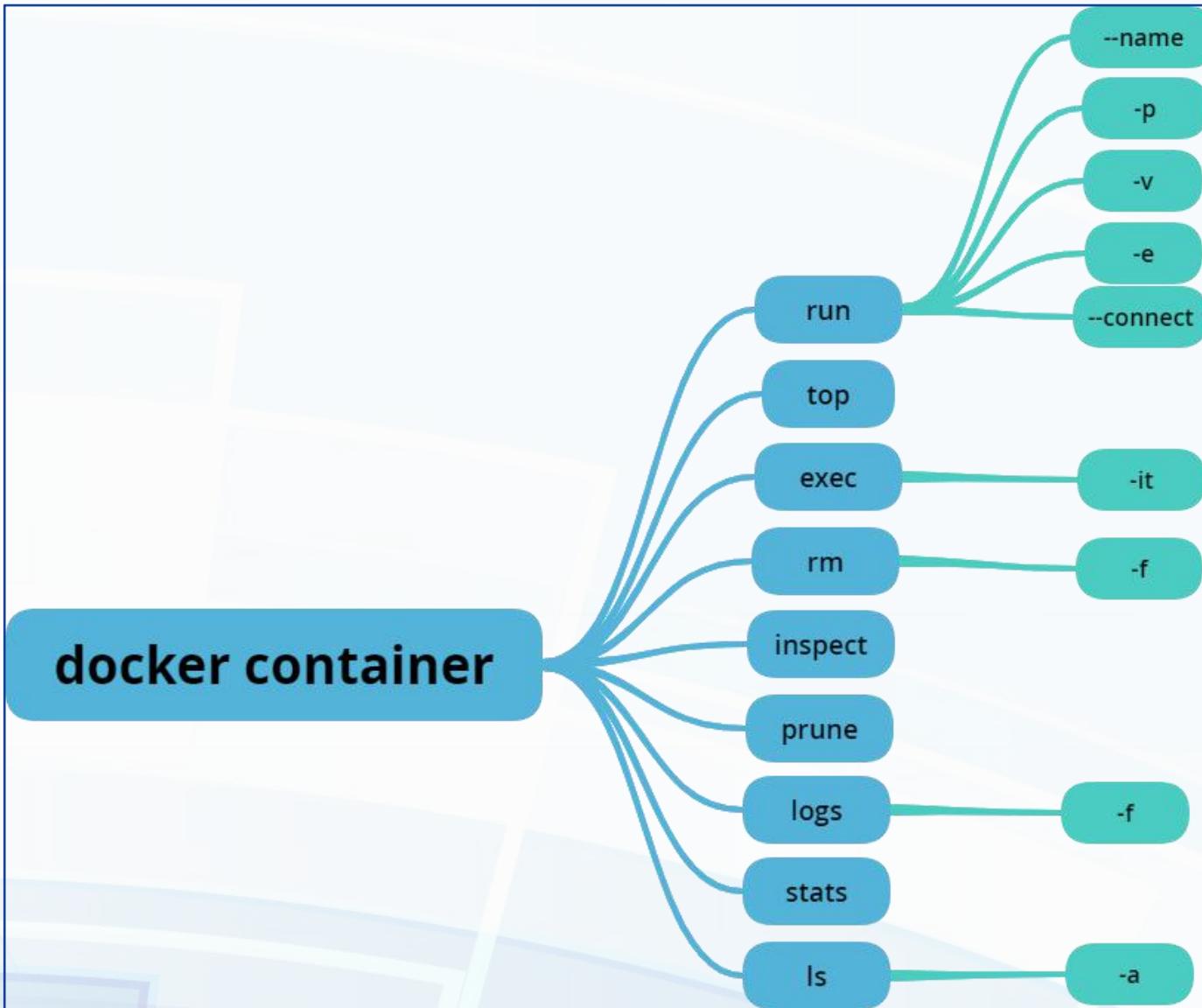
# Docker CLI

## ■ Principais comandos



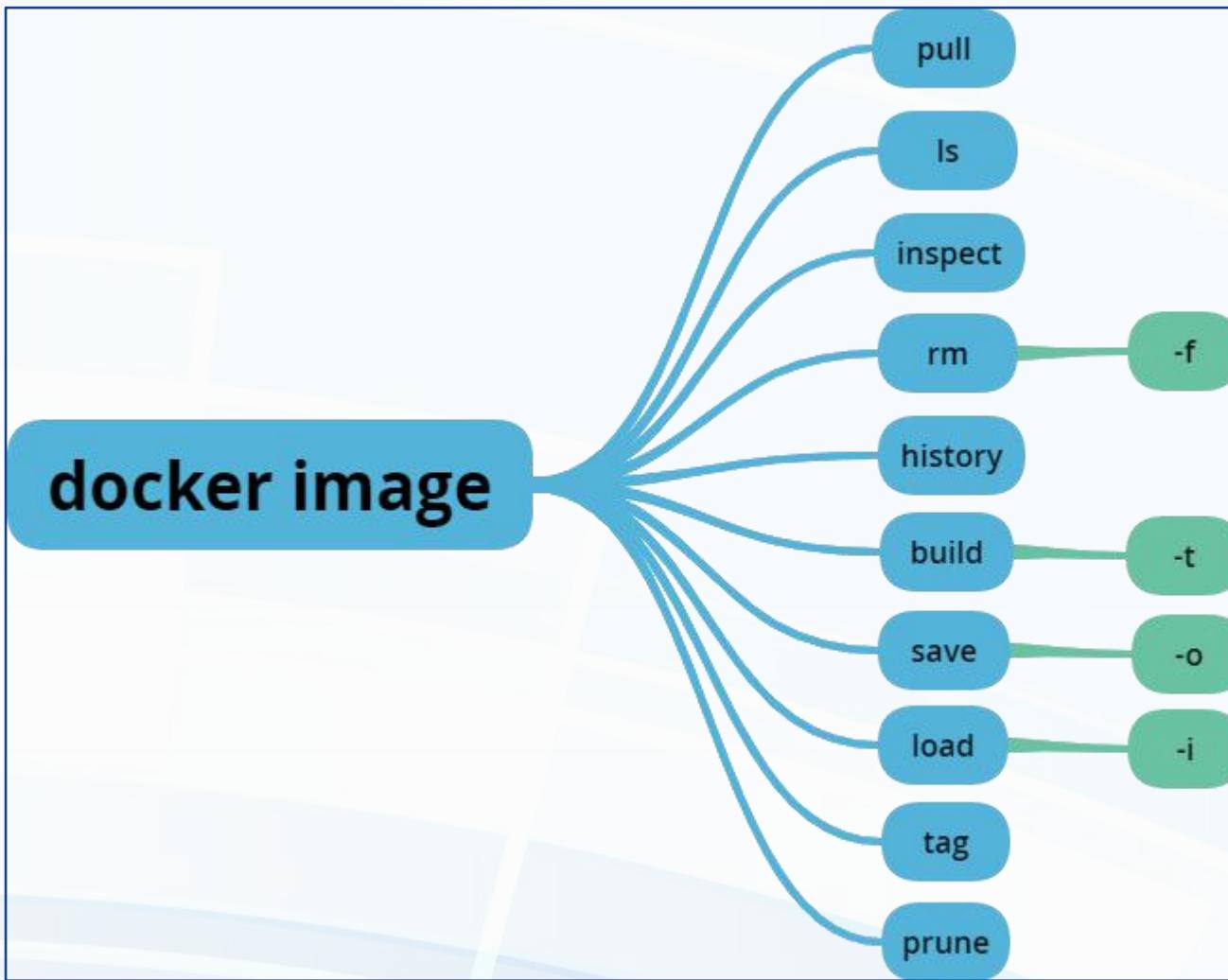
# Docker CLI

## contêiner



# Docker CLI

## image



# **Exercício 1**

## **Instale Docker e Docker-Compose**

- 1. Instale o Docker**
- 2. Instale o Docker-Compose**
- 3. Execute comando “`docker version`” para verificar versão do Docker instalada**
- 4. Execute comando “`docker-compose version`” para verificar versão do Docker-Compose instalada**

**\*Verificar páginas 2 e 3 da apostila**

# Exercício 2

## Executando o primeiro contêiner

`docker contêiner run hello-world`

**docker:** Executa algum comando Docker

**contêiner:** Especifica que o comando será utilizado sobre um contêiner

**run:** Executa um comando em um novo contêiner

1. `docker image pull` - baixa imagem caso não exista
2. `docker contêiner create` - cria um contêiner
3. `docker contêiner start` - inicializa o contêiner
4. `docker contêiner exec` - executa o comando no contêiner em execução

**hello-world:** Nome da imagem utilizada

# Exercício 3

## Crie uma imagem a partir de um contêiner

1. Criar contêiner a partir da imagem oficial do ubuntu.
2. Acessar o contêiner criado e instalar o pacote iutils-ping
3. Criar uma nova imagem a partir do contêiner
4. Criar um contêiner a partir da nova imagem e testar o comando ping
5. Apague a imagem e o contêiner criados  
Verificar as páginas 4, 5 e 6 da apostila

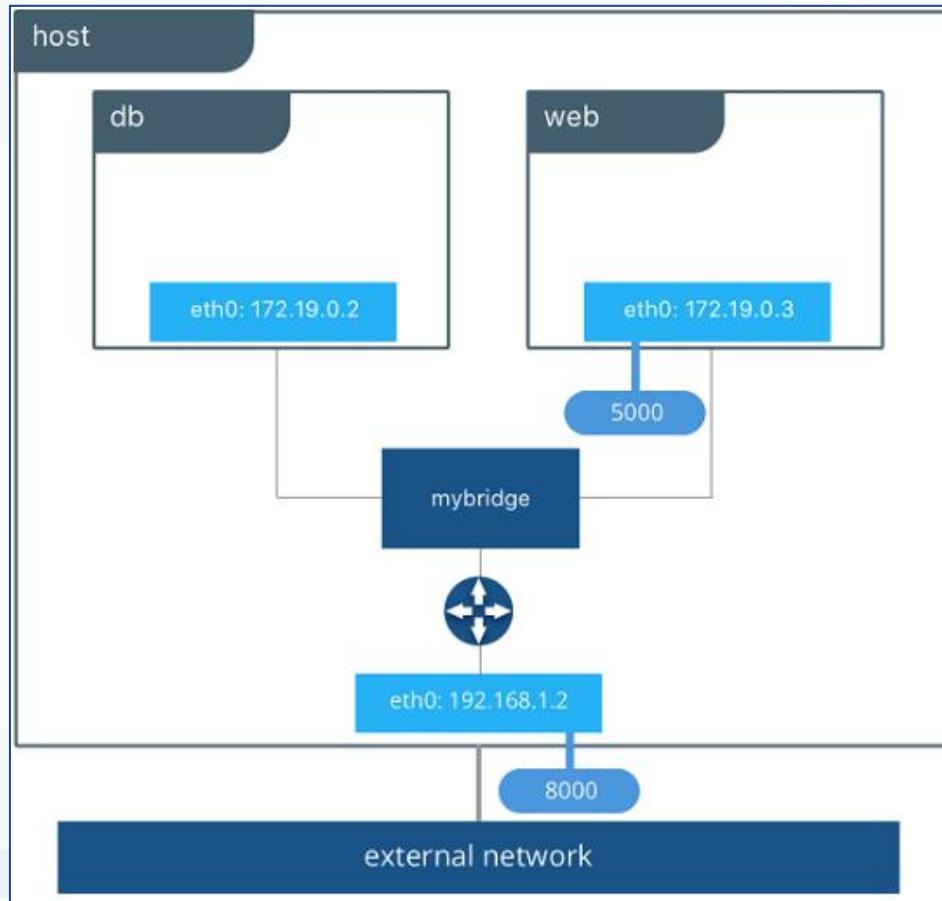
# Docker Network

■ Os drivers de rede nativos que o Docker possui são:

- Bridge (default)
- Host
- Overlay
- Macvlan
- None

# Docker Network

## ■ Bridge Network



# Docker Network

## ■ Principais comandos para manipular redes no Docker

Comando	Descrição
<code>docker container run -p 80:80 apache</code>	mapear porta <host>:<contêiner>
<code>docker network create -d &lt;driver&gt; &lt;nome&gt;</code>	criar uma nova rede
<code>docker network rm &lt;nome&gt;</code>	apagar uma rede
<code>docker network inspect &lt;nome&gt;</code>	exibir informações detalhadas
<code>docker container run -p 80:80 --network &lt;nome&gt; apache</code>	instacia um contêiner e o conecta a uma rede específica
<code>docker container run -p 80:80 --link &lt;db:db&gt; apache</code>	Linkar 2 contêineres --link <nome:alias>

# **Exercício 4**

## **Acesse o serviço apache**

- 1. Crie uma rede bridge chamada ex\_apache**
- 2. Crie um contêiner a partir da imagem oficial do apache, conecte-o a rede ex\_apache e mapeie a porta 90 do host para a porta 80 do contêiner.**
- 3. Tente acessar através de um browser o endereço http://localhost:90**
- 4. Apague o contêiner e a rede criados**

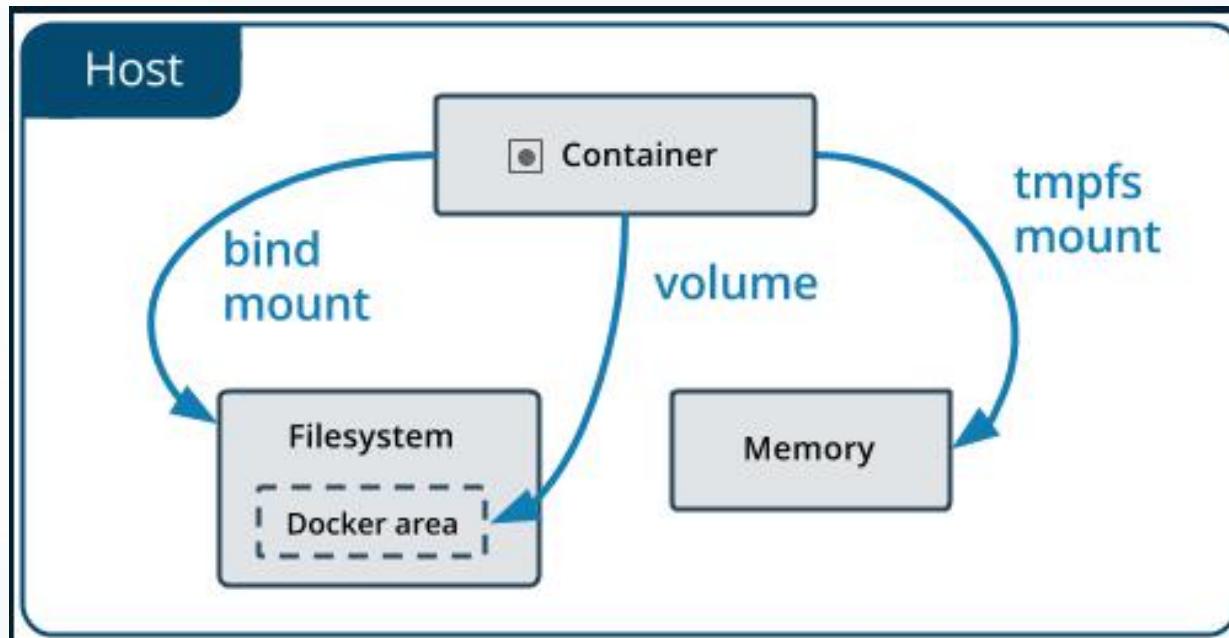
**Verificar as páginas 5,6,7 e 8 da apostila**

# Persistência dos Dados

- Os dados não persistem quando o contêiner não existe mais
- As duas principais formas para tratar a persistência dos dados no Docker são: Volumes e Bind Mounts
- Volumes são criados e gerenciados pelo Docker e armazenam os dados em: `/var/lib/docker/volumes/<volume>`
- Bind Mounts espelha uma pasta/arquivo do host para uma pasta/arquivo do contêiner

# Persistência dos Dados

- Principais formas de tratar a persistência dos dados no Docker são:



# Persistência dos Dados

## ■ Principais comandos para gerenciar a persistência dos dados no Docker

Comandos para gerenciar a persistência dos dados no Docker	
<code>docker volume create &lt;nome&gt;</code>	Criar um volume
<code>docker volume ls</code>	Listar volumes
<code>docker volume inspect &lt;nome&gt;</code>	Informações detalhadas
<code>docker volume rm &lt;nome&gt;</code>	apagar um volume
<code>docker container run --name teste -v &lt;nome&gt;:/app nginx</code>	Iniciar contêiner com um volume
<code>docker container run --name teste2 -v &lt;pasta/arq_host&gt;:&lt;pasta/arq_contêiner&gt; nginx</code>	iniciar contêiner com um bind mount

# Exercício 5

## Criando Volume

1. Crie um volume chamado teste
2. Crie um contêiner a partir da imagem oficial do ubuntu de nome testevolume1 e monte o volume criado na pasta /teste do contêiner
3. No contêiner, crie o arquivo /teste/teste.txt com o conteúdo “testando volumes no Docker”
4. Saia do contêiner e verifique se o arquivo foi criado no sistema de arquivos do host
5. Apague o contêiner testevolume1

# Exercício 5

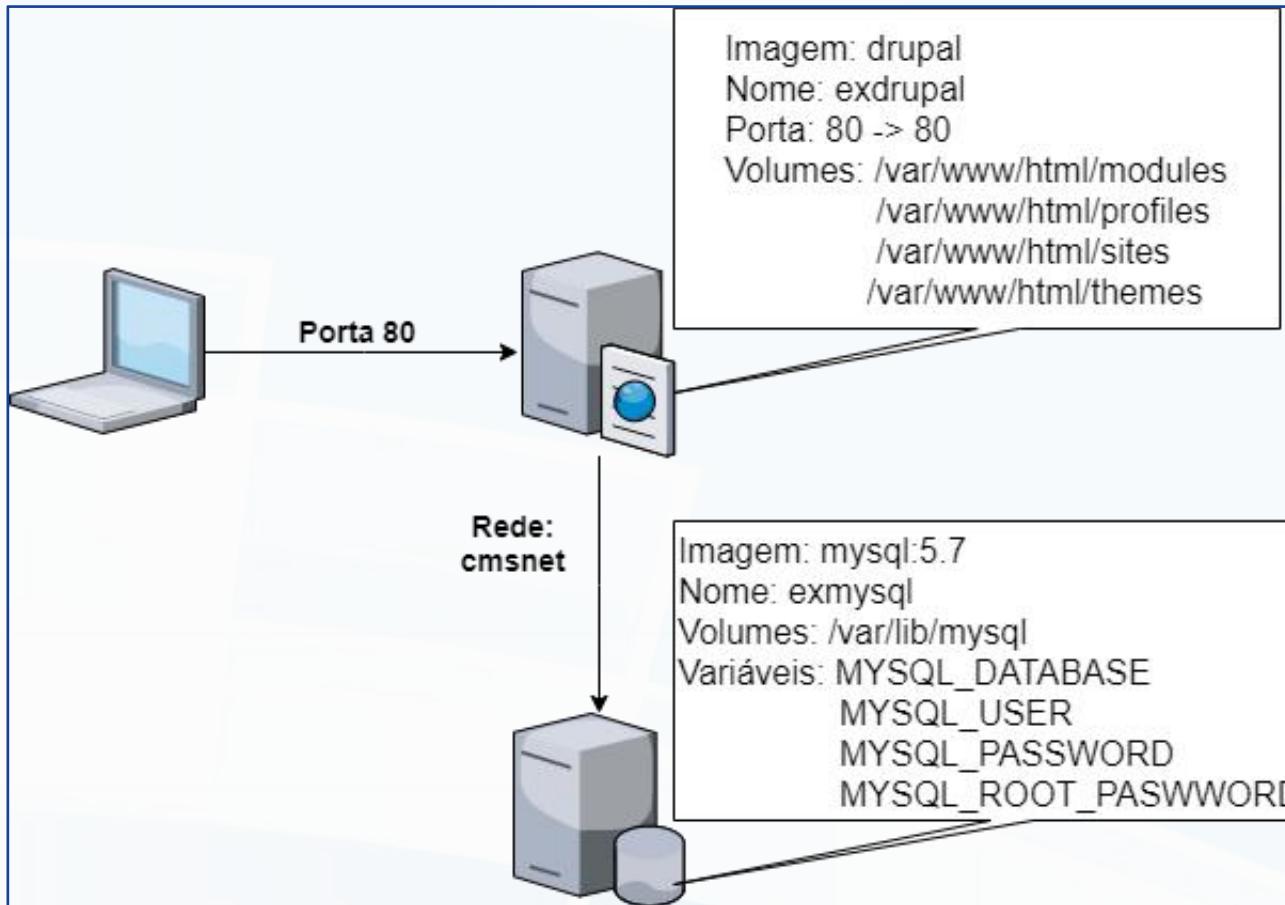
## Criando Volume

6. Crie um contêiner a partir da imagem oficial do ubuntu de nome testevolume2 e monte o volume criado na pasta /teste2 do contêiner
7. Verifique o conteúdo do arquivo /teste2/teste.txt
8. Saia do contêiner, remova o volume teste e remova o contêiner testevolume2

Verificar as páginas 5,6 e 7 da apostila

# Exercício 6

## CMS Drupal



\* Remova os contêineres, volumes e a rede após o término do exercício

# Portainer

- É uma interface web de gerenciamento leve que permite gerenciar facilmente o host do Docker ou o cluster do Swarm.
- Não necessita de instalação, é executado como um contêiner
  1. `docker volume create portainer_data`
  2. `docker run -d -p 9000:9000 -p 8000:8000 --name portainer --restart always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer`

# Portainer

portainer.io

my account log out

Home PRIMARY

Dashboard

App Templates

Stacks

Containers

Images

Networks

Volumes

Events

Host

SETTINGS

Extensions

Users

Endpoints

Registries

Settings

portainer.io 1.22.2

## Endpoints

Latest News From Portainer x dismiss

We are pleased to announce our new Portainer Membership program. This free program is designed to provide a range of extra resources, content and ways of engaging with the Portainer Team. As a Portainer member you will get instant access to a growing range of configuration guides, member only webinars with the founders and more. Details are [here](#)

If you haven't done so already, please take the opportunity to upgrade your Portainer instance to 1.22.2. This release offers fixes to a range of security issues that have come to light in our ongoing testing. All Portainer users are strongly recommended to upgrade at their earliest opportunity. Details are available [here](#)

Thank you for your ongoing support



Endpoints

Refresh

Search by name, group, tag, status, URL...

 primary <span>up</span> 2019-11-20 15:27:29	Group: Unassigned <span>edit</span>
1 stack	5 containers - 5 0
2 3.1 GB	21 volumes 5 images
No tags	

Group: Unassigned edit

Standalone 19.03.4

/var/run/docker.sock

Items per page 10 ▾

# Exercício 7

## Portainer

1. Crie um contêiner executando o portainer
2. Explore a interface gráfica do portainer
3. Utilizando o portainer, crie a mesma estrutura do exercício anterior
4. Apague todos os contêineres, volumes e rede criados

# Dockerfile

- Arquivo responsável por criar as imagens no Docker
- Funciona como um script com comandos e instruções que você executaria manualmente na construção de uma imagem
- O arquivo texto “deverá” ser salvo com o nome Dockerfile para que a imagem seja criada utilizando o comando build

# Dockerfile

## ■ Principais instruções do arquivo Dockerfile

Dockerfile	
FROM	LABEL
RUN	VOLUME
EXPOSE	WORKDIR
ENV	USER
COPY	CMD
ADD	ENTRYPOINT

# Dockerfile

```
FROM centos:latest
```

```
RUN yum -y install httpd
```

```
ENTRYPOINT ["/usr/sbin/httpd", "-D", "FOREGROUND"]
```

```
EXPOSE 80
```

```
docker image build -t apachelmts .
```

- Guia de referência do Dockerfile

<https://docs.docker.com/engine/reference/builder/>

# Dockerfile

## Melhores Práticas

- Utilize sempre que possível imagens oficiais
- Não instale pacotes desnecessários
- Sempre combine RUN apt-get update com apt-get install na mesma instrução RUN
- Sempre limpe o cache dos gerenciadores de pacotes após instalar um ou mais pacotes
  - yum -y clean all
  - rm /var/lib/apt/lists/\*

# Dockerfile

## Melhores Práticas

- Evite utilizar os comandos “yum update” e “apt upgrade”
- Crie o menor número possível de camadas
- As instruções RUN, COPY e ADD criam uma nova camada na imagem, use-as o menor número de vezes possível

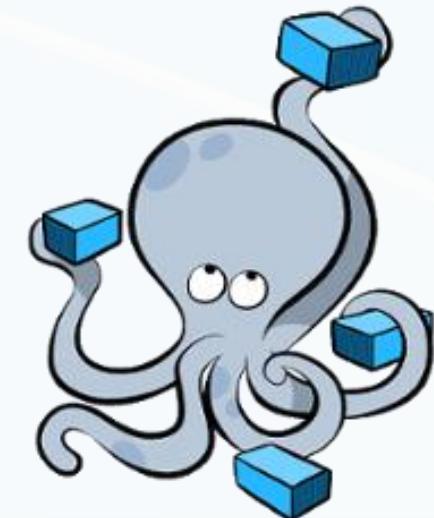
# Exercício 8

Crie uma imagem utilizando Dockerfile que apresente a página de informação do PHP

1. Crie o arquivo Dockerfile
2. Crie um volume que irá armazenar o arquivo index.php
3. Crie o arquivo index.php com o código: <?php  
phpinfo(); ?>
4. Crie uma imagem com nome phplmts
5. Crie um contêiner a partir da imagem phplmts
6. Teste o app
7. Verificar as páginas 4, 5, 6, 8, 9 e 10 da apostila

# Docker Compose

- É o orquestrador de contêineres do Docker
- Permite instanciar diversos contêineres de uma vez
- É escrito em Ain't Markup Language (YAML)
- É instalado separadamente
- Instalação e guia de referência do Docker Compose



<https://docs.docker.com/compose/install/#install-compose>

<https://docs.docker.com/compose/compose-file/>

# Docker-Compose

## ■ Principais instruções do arquivo docker-compose.yml

docker-compose.yml	
version	links
services	volumes
image	expose
build	restart
contêiner_name	network
labels	ports
enviroment	depends_on

## ■ Verificar a página 10 da apostila

# Docker Compose

## Exemplo do arquivo docker-compose.yml

```
1 version: '3'
2 services:
3   mediawiki:
4     container_name: wiki
5     image: mediawiki
6     restart: always
7     ports:
8       - 7070:80
9     links:
10    - database
11   volumes:
12     - /var/www/html/images
13
14   database:
15     image: mariadb
16     restart: always
17     environment:
18       MYSQL_DATABASE: my_wiki
19       MYSQL_USER: wikiuser
20       MYSQL_PASSWORD: example
21       MYSQL_RANDOM_ROOT_PASSWORD: 'yes'
```

# **Exercício 9**

## **Orquestrar ambiente CMS Drupal**

- 1. Crie o arquivo docker-compose.yml com os serviços Drupal, Mysql e Portainer.**
- 2. Verifique o consumo de recursos dos serviços criados**
- 3. Remova os contêineres, rede e volumes criados**

**Verificar as páginas 5,10,11 e 12 da apostila**

# Sugestão de Estudos

- Aprimorar os estudos sobre o Docker, Docker-Compose e Dockerfile
- Praticar Docker no lab, endereço:  
<https://labs.play-with-docker.com/>
- Estudar Cluster Docker utilizando Swarm
- Estudar Cluster Docker utilizando Kubernetes

# Docker



**Perguntas ???**