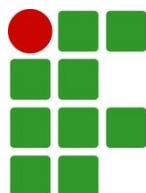




## ENCONTRO DE COMPUTAÇÃO DO SERTÃO

*Docker: Disponibilizando APIs*



# MINICURSO DOCKER

DISPONIBILIZANDO APIs



Ricardo de Sousa

Job

IFPB/Cajazeiras



ricardojob



ricardojob

04.11 QUINTA

14H



ENCONTRO DE COMPUTAÇÃO  
DO SERTÃO

# Este minicurso não é...

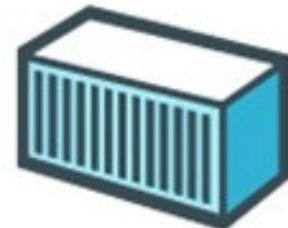


# Dificuldades

- Montar e organizar um ambiente de desenvolvimento
  - No meu computador funciona!
- Entregar o software funcional para diferentes ambientes: desenvolvimento, homologação, produção
  - Sistema operacional
  - Versões e configurações de banco de dados
  - Conflitos de ferramentas e bibliotecas
-

# Contêineres

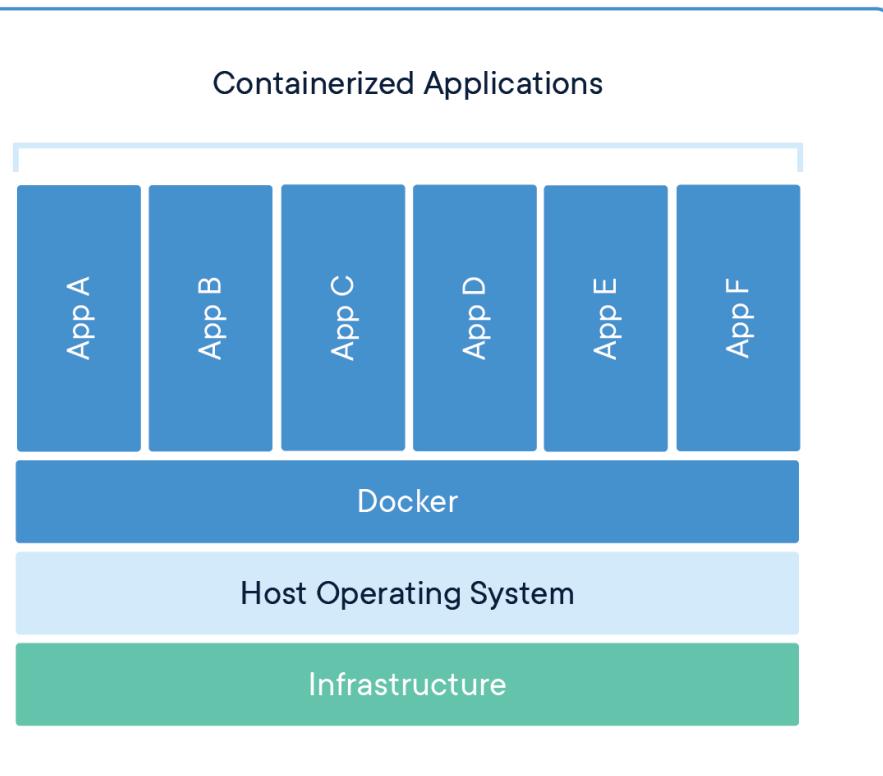
A container is a **standard unit** of software that **packages up code** and all its **dependencies** so the application runs quickly and **reliably** from **one computing environment to another.**



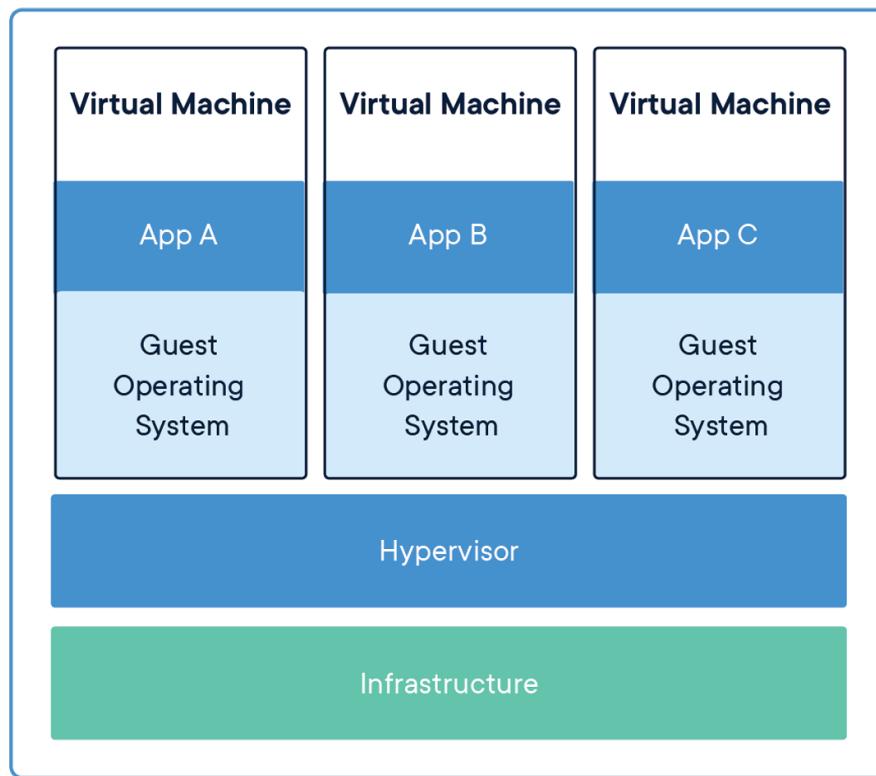
# Contêineres x VM

**Contêineres** é uma *image* em execução.

Os contêineres virtualizam o **sistema operacional**, não o hardware.

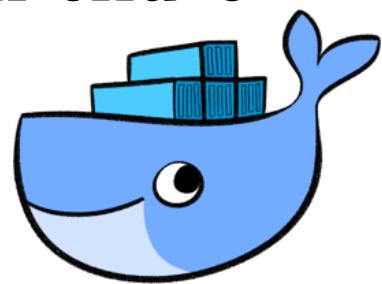


**VMs** proporcionam um **isolamento maior** e uma **sobrecarga**, pois cada **máquina virtual** que é criada executará seu **próprio kernel** e **instância do sistema operacional**.



# Docker

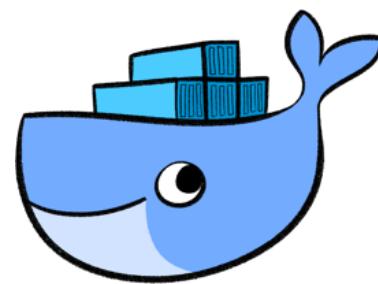
- **Containers** são mais leves, já que não precisam de um ambiente virtual completo, pois o kernel do host proporciona total gerenciamento de memória, I/O, CPU etc.
- **Docker** é uma ferramenta para criar e manter containers, ou seja, ele é responsável por armazenar vários serviços de forma isolada do SO host, como: web server, banco de dados, aplicação, memcached etc. O seu back-end é baseado em LXC (LinuX Containers).



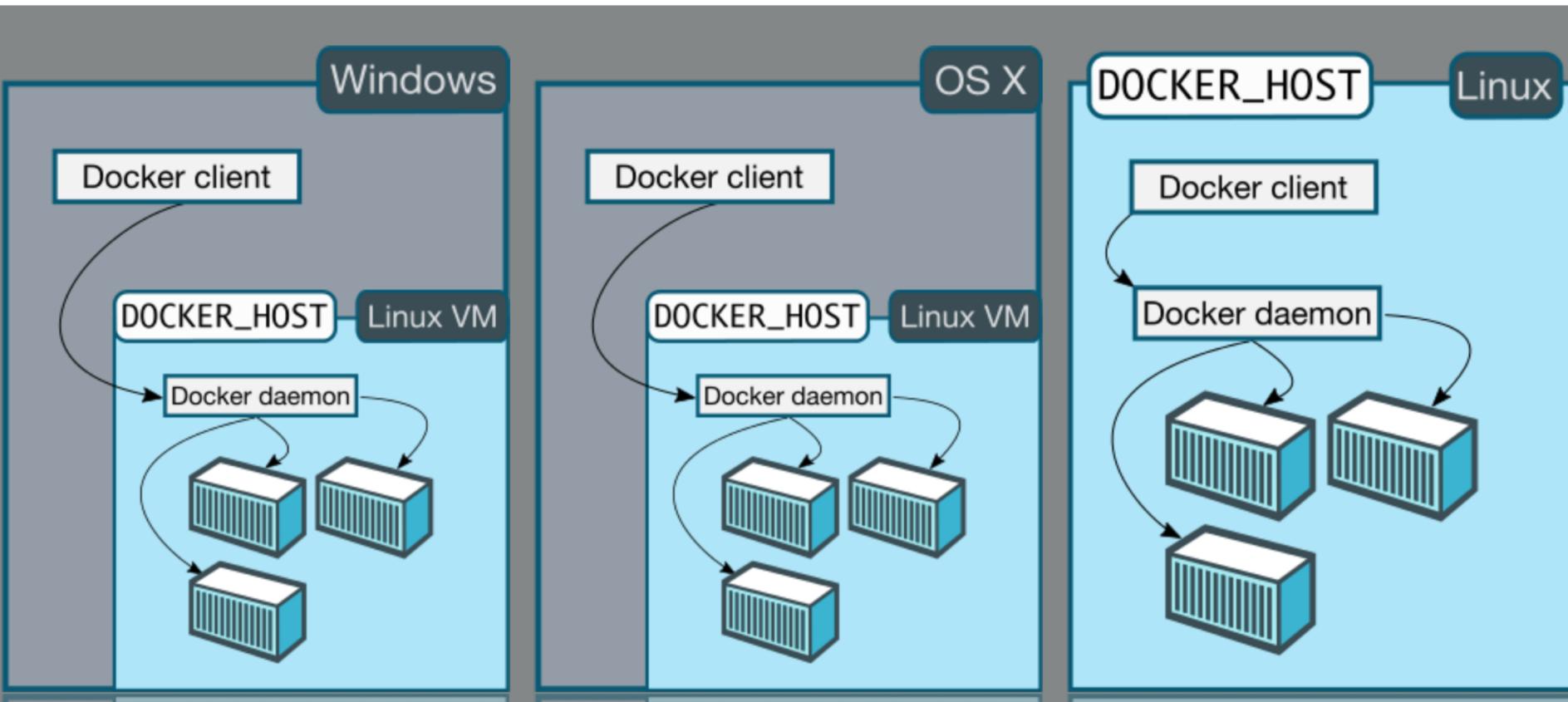
# Docker

**Docker** auxilia na construção de uma camada de isolamento em software e reduz a carga de comunicação.

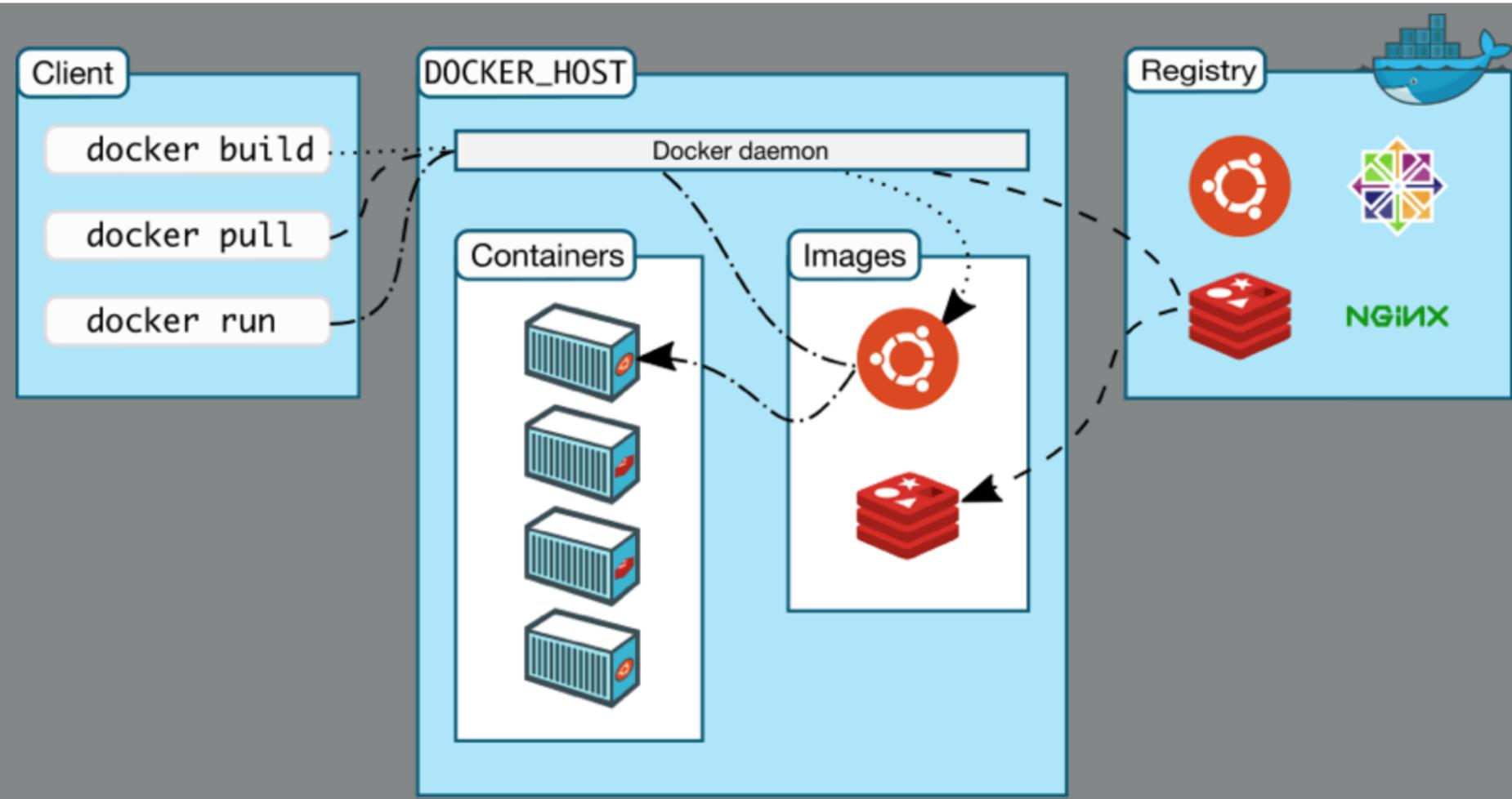
- containers atômicos.
- aplicações portáveis, escaláveis e confiáveis.



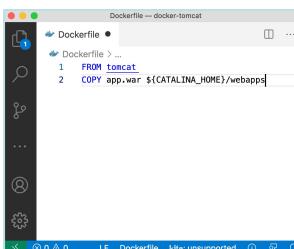
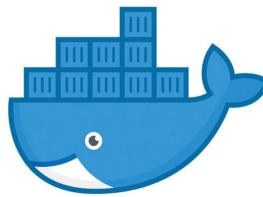
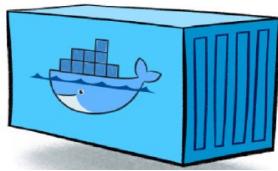
# Docker



# Arquitetura



# Pilares

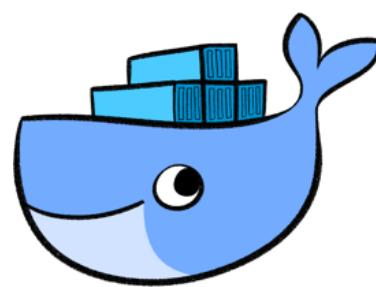


- **Docker Containers** são container que empacotam as aplicações e permitem sua execução em diversos ambientes.
- **Docker Image** é um template *read-only* que contém um conjunto de instruções para a criação de um contêiner.
- **Docker Registry** é um ambiente para catalogar e compartilhar Docker images.
- **Dockerfile** é um documento de texto que descreve os recursos de uma imagem. Ele contém todos os comandos necessários para montar uma Docker image.

# Command Line

## Contêineres

Comando	Descrição
<code>docker container ls</code>	lista os container em execução
<code>docker container run</code>	cria e iniciar (create e start) um container
<code>docker container start</code>	inicia um container
<code>docker container stop</code>	para um container
<code>docker container rm</code>	remove um container
<code>docker container exec</code>	executa um comando em um container



# Nosso primeiro conteiner

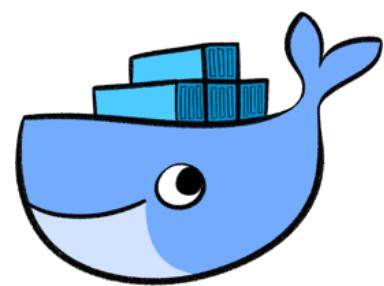
## Etapa 1

- Executar containers

```
docker container run tomcat
```

- Listar os containers em execução

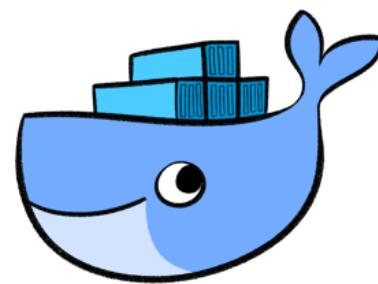
```
docker container ls
```



# Command Line

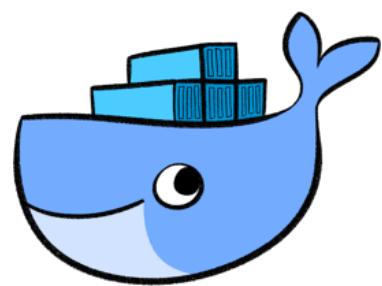
## Image

Comando	Descrição
<code>docker image ls</code>	lista as images
<code>docker image build</code>	constrói uma nova image
<code>docker image push</code>	envia uma image para um registro
<code>docker image pull</code>	baixa uma image para um registro
<code>docker image rm</code>	remove uma image



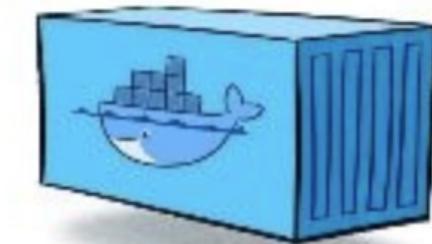
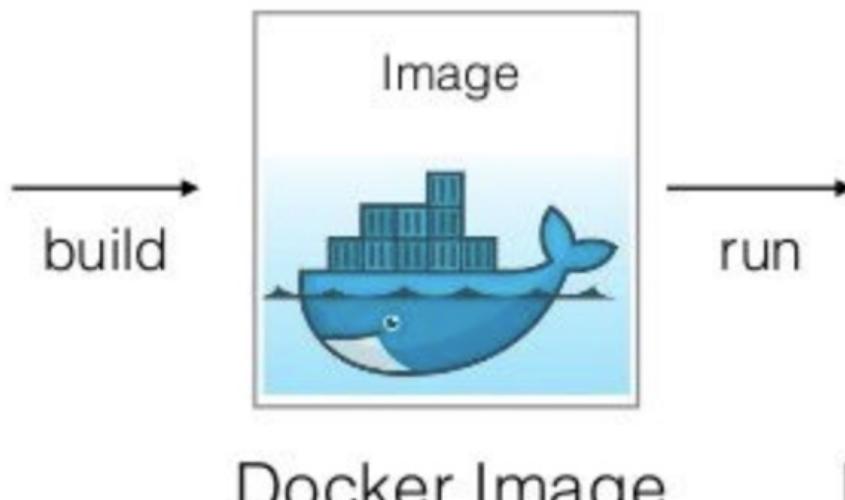
# Dockerfile

- O **Dockerfile** é um documento de texto que **descreve** os recursos de uma imagem.
- Ele contém a sequência de **comandos** necessários para configurar e montar uma Docker image.
- 



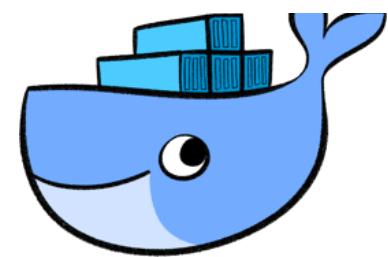
# Dockerfile

## Dockerfile



## Docker Image

# Docker Container



## Etapa 2

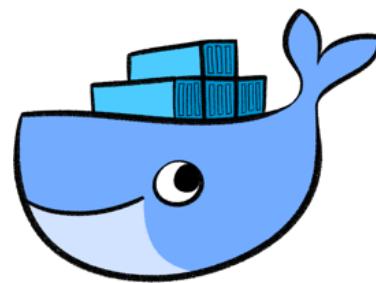
- Criar um arquivo *Dockerfile*

```
FROM payara/server-web
```

```
COPY target/app.war $DEPLOY_DIR/app.war
```

- Criar uma *image*

```
docker image build -t exemplo .
```



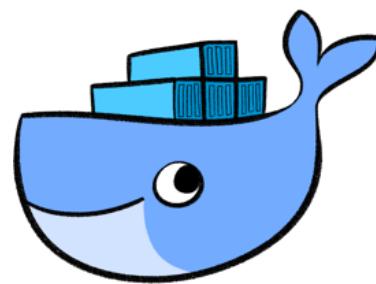
## Etapa 2

- **Listar as *image* disponíveis**

`docker image ls`

- **Executar a imagem criada**

`docker container run -p 8080:8080 exemplo`



# Dockerfile

Configurando o banco de dados **PostgreSQL**.

● Dockerfile ●

exemplo-04 > postgres > ● Dockerfile > ...

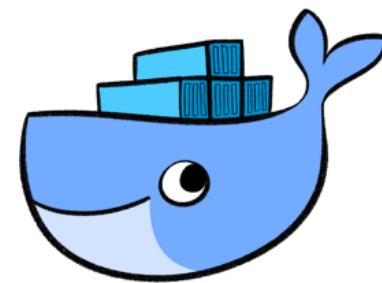
```
1 FROM postgres
2 ENV POSTGRES_DB clientes
3 ENV POSTGRES_USER job
4 ENV POSTGRES_PASSWORD 123
5
6 COPY create.sql /docker-entrypoint-initdb.d/
7 COPY insert.sql /docker-entrypoint-initdb.d/
```

Listando os arquivos de um diretório.

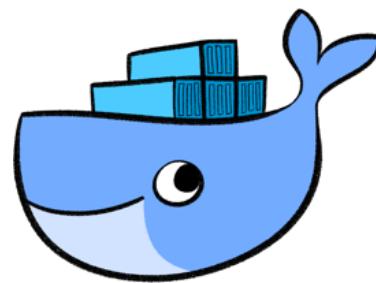
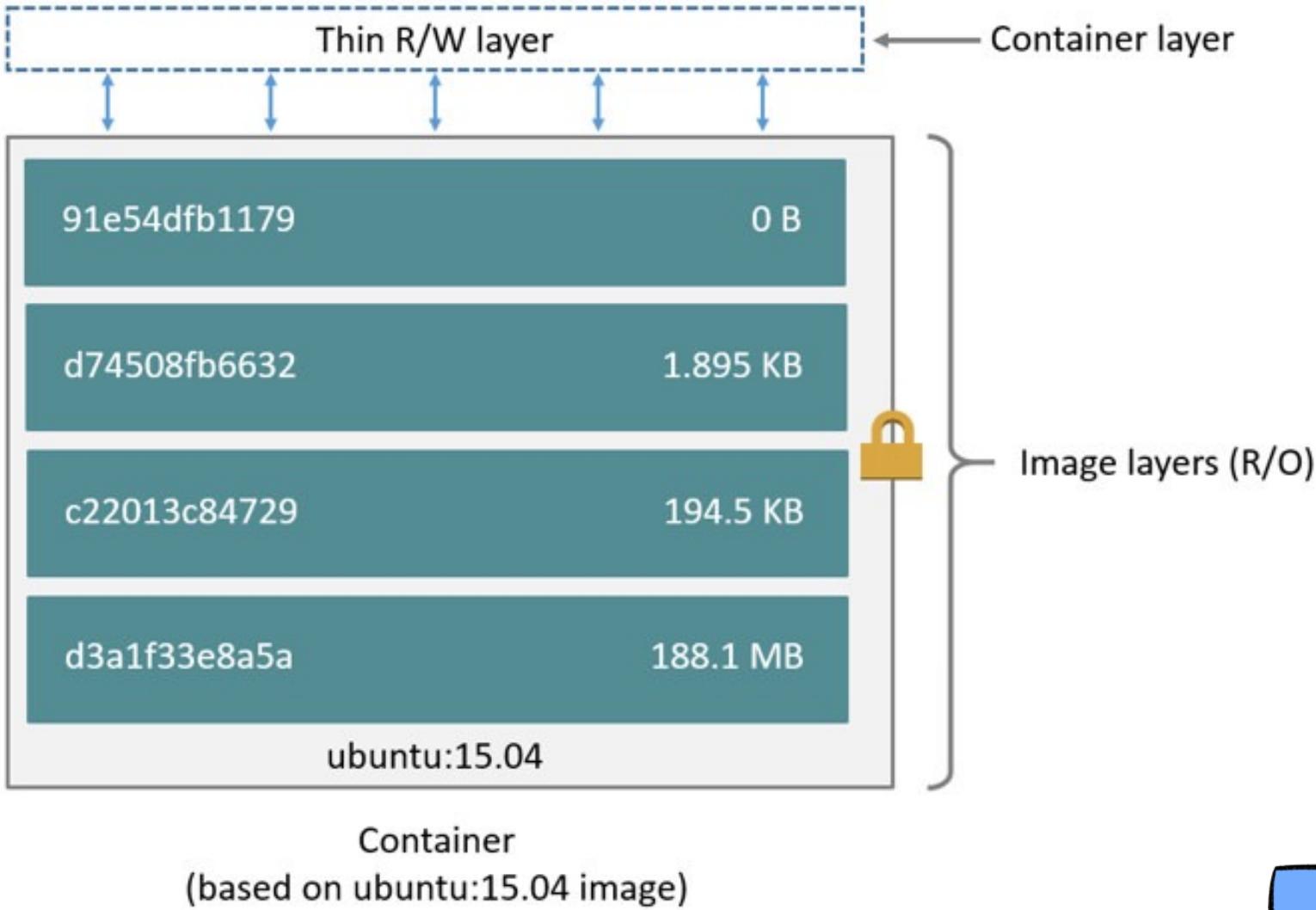
● Dockerfile X

exemplo-03 > ● Dockerfile > ...

```
1 FROM alpine
2 WORKDIR /app/src
3 RUN mkdir data
4 VOLUME [ "data" ]
5 EXPOSE 9090
6 CMD ls data
```



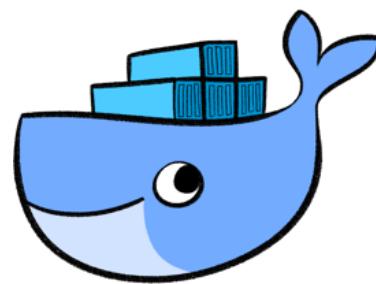
# Dockerfile



# Comandos

## Dockerfile

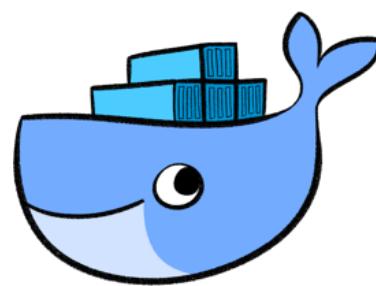
Comando	Descrição
FROM	Informa a partir de qual imagem será gerada a nova imagem.
RUN	Especifica que o argumento seguinte será executado, ou seja, realiza a execução de um comando
ENV	Instrução que cria e atribui um valor para uma variável dentro da imagem
CMD	Define um comando a ser executado quando um container baseado nessa imagem for iniciado
ENTRYPOINT	Informa qual comando será executado quando um container for iniciado. Esse comando será sempre executado.



# Comandos

## Dockerfile

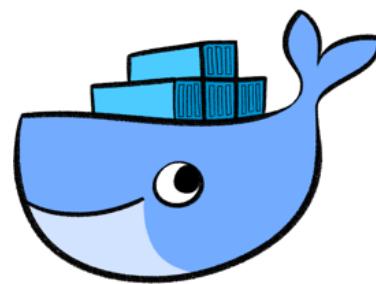
Comando	Descrição
COPY	Copia arquivos ou diretórios locais para dentro da imagem
ADD	Adiciona arquivos locais ou remotos (url), para dentro da imagem.
EXPOSE	Indica uma ou mais portas que o container irá ouvir, ou seja, o container quando iniciado <b>poderá</b> ser acessível através dessas portas.
WORKDIR	Define qual será o diretório de trabalho padrão
VOLUME	Mapeia um diretório do host para ser acessível pelo container.



## Etapa 3

- Criar a *image* do PostgreSQL

```
FROM postgres
ENV POSTGRES_USER postgres
ENV POSTGRES_PASSWORD 12345
ENV POSTGRES_DB aula
COPY create.sql /docker-entrypoint-initdb.d/
COPY insert.sql /docker-entrypoint-initdb.d/
```



# Roteiro

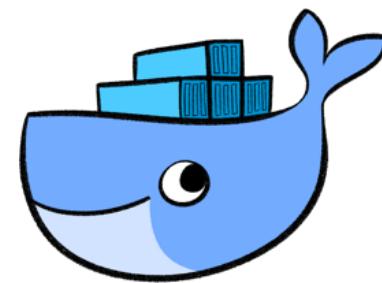
## Etapa 3

- Criar a *image* do PostgreSQL

```
docker build -t ricardojob/banco ./postgres
```

- Executar o container

```
docker container run -p 5433:5432 -d --name  
banco ricardojob/banco
```



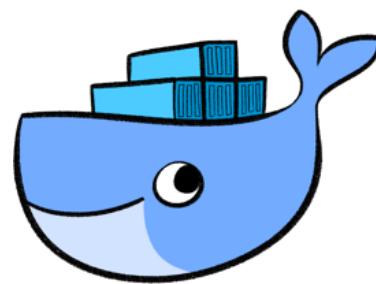
## Etapa 3

- Executar comandos no container

```
docker exec -it banco /bin/bash
```

- Acessar o *database* que definimos no arquivo *Dockerfile*

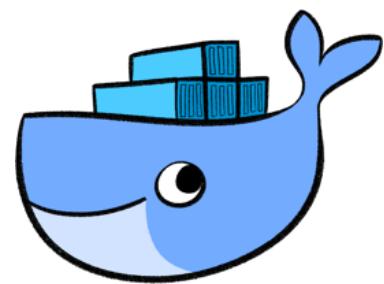
```
psql -U postgres aula
```



# Docker Compose

- **Compose** is a **tool** for **defining** and **running multi-container** Docker applications. With Compose, you use a **YAML file** to **configure** your application's services. Then, with a single command, you **create and start all the services** from your configuration.

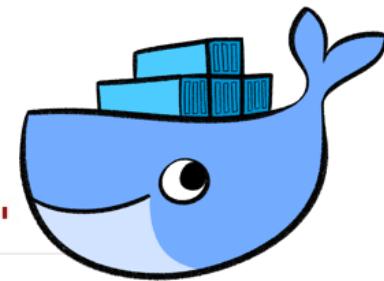
○



# Docker Compose

Configurando uma aplicação que possui **dois** serviços: **banco** (para o bando de dados) e **app** (para a aplicação)

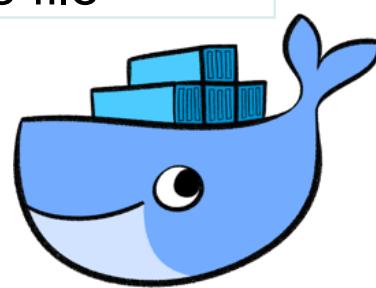
```
❶ docker-compose.yml ×  
exemplo-04 > ❷ docker-compose.yml  
❸ version: '3'  
❹ services:  
❺ banco:  
❻   container_name: banco  
❼   image: ricardojob/banco  
❼   build: ./postgres  
❼   ports:  
❼     - "5433:5432"  
❼ app:  
❼   container_name: app  
❼   image: ricardojob/app  
❼   build: ./app  
❼   ports:  
❼     - "8080:8080"  
❼   depends_on:  
❼     - "banco"  
❼   links:  
❼     - "banco:host-banco"
```



# Command Line

## Docker compose

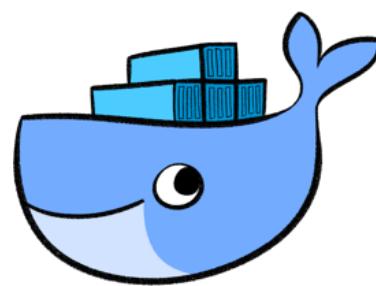
Comando	Descrição
<code>docker-compose ps</code>	lista os container
<code>docker-compose run</code>	inicia um container
<code>docker-compose start</code>	inicia um serviço
<code>docker-compose up</code>	cria e inicia os contêineres
<code>docker-compose stop</code>	para um serviço
<code>docker-compose down</code>	para e remove os contêineres
<code>docker-compose exec</code>	executa um comando em um container
<code>docker-compose config</code>	valida e visualiza o Compose file



# Comandos

## Compose file

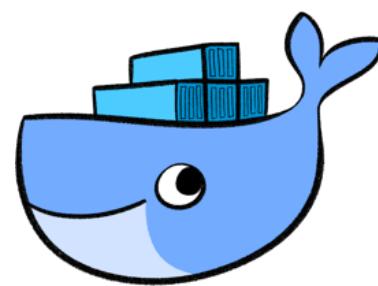
Comando	Uso	Descrição
build	build: ./app	Pode ser especificado como uma <i>string</i> com o caminho para o contexto de criação
context	context: ./dir	Especifica o diretório que contém o <b>Dockerfile</b>
dockerfile	dockerfile: Dockerfile-prod	Especifica um <b>Dockerfile</b> alternativo
args	args: buildno: 1	Adiciona um argumento ao processo de build. Ele deve ter sido especificado no <b>Dockerfile</b> .



# Comandos

## Compose file

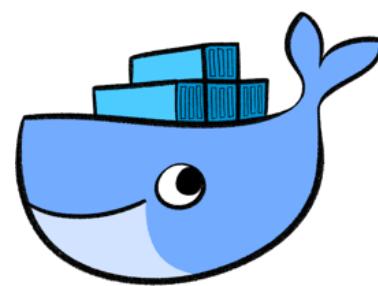
Comando	Uso	Descrição
network	network: custom_network	Especifica a rede que será conectada durante a execução das instruções do comando RUN.
target	target: prod	Constrói um <i>stage</i> específico definido no <b>Dockerfile</b>
container_name	container_name: banco	Especifica um <b>nome</b> ao container
depends_on	depends_on: - banco	Especifica a dependência entre os serviços.



# Comandos

## Compose file

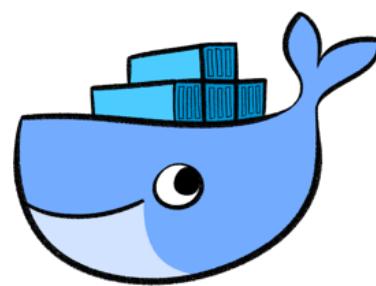
Comando	Uso	Descrição
env_file	env_file: .env	Especifica variáveis definidas no arquivo .env
command	command: bundle exec thin -p 3000	Sobrescreve o comando definido no Dockerfile
entrypoint	entrypoint: /entrypoint.sh	Sobrescreve o entrypoint definido no Dockerfile
environment	environment: - RACK_ENV=development - SHOW=true	Adiciona variáveis de ambientes.



# Comandos

## Compose file

Comando	Uso	Descrição
expose	<code>expose:</code> – "3000" – "8000"	Define as portas, mas sem publicá-las.
image	<code>image: ricardojob/banco</code>	Especifica a image que o container será iniciado.
ports	<code>ports:</code> – "5433:5432"	Publica as portas.
volumes	<code>volumes:</code> – "dbdata:/var/lib/postgresql/data"	Monta um path no host que será compartilhado com o container.



## Previously

```
docker image ls
```

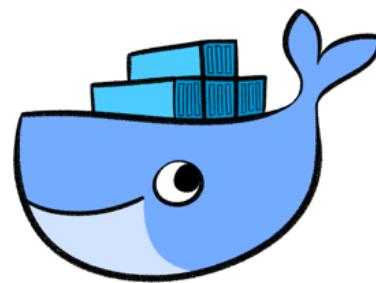
```
docker container ls
```

```
docker image build -t ricardojob/banco ./postgres
```

```
docker container run -p 5433:5432 -d --name banco
```

```
ricardojob/banco
```

```
docker container exec -it banco /bin/bash
```



# Roteiro

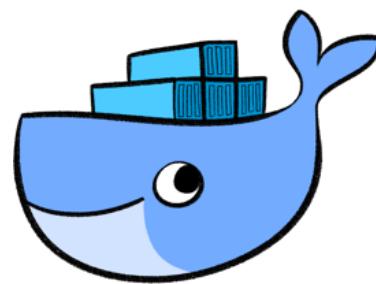
## Previously

```
docker container run -p 8080:8080 -d --name  
aplicacao ricardojob/app
```

--detach, (executa o comando em background)

--publish, (publica uma porta do container para o host)

```
docker container cp aplicacao  
app:/usr/local/tomcat/webapps
```



## **Lista de comandos**

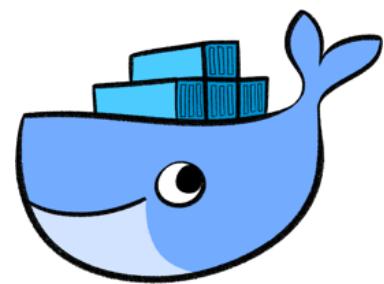
- <https://docs.docker.com/engine/reference/commandline/>

## **Ambiente para treinamentos**

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

## **Exemplos**

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



## Mundo Docker

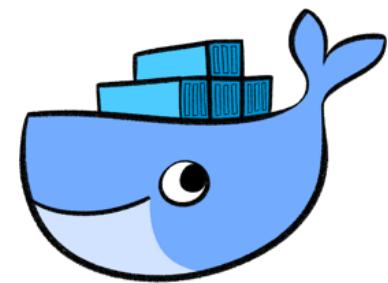
- <https://www.mundodocker.com.br/o-que-e-dockerfile/>

## Lista de comandos

- <https://docs.docker.com/compose/reference/build/>

## Docker Compose

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





## ENCONTRO DE COMPUTAÇÃO DO SERTÃO