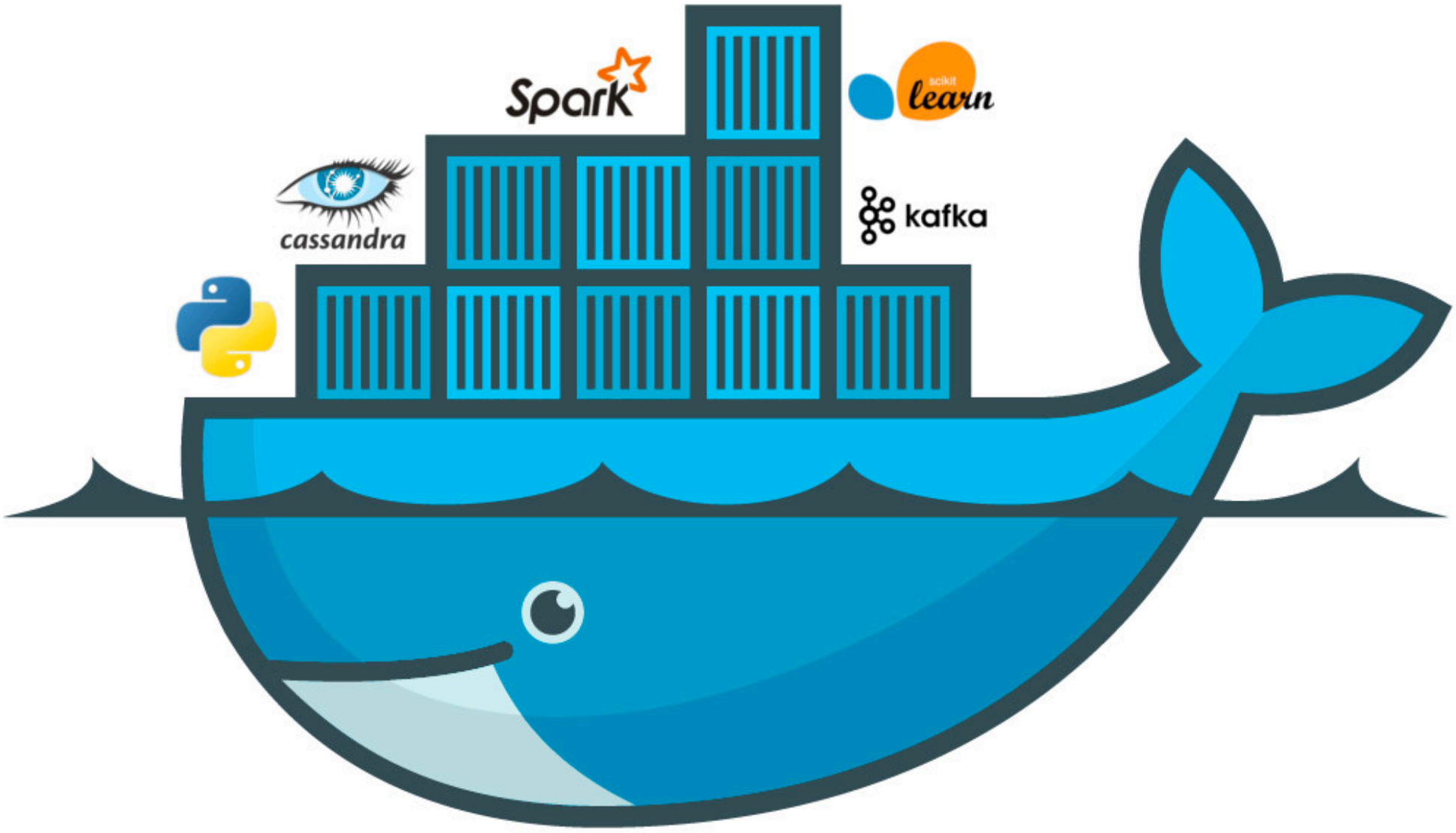
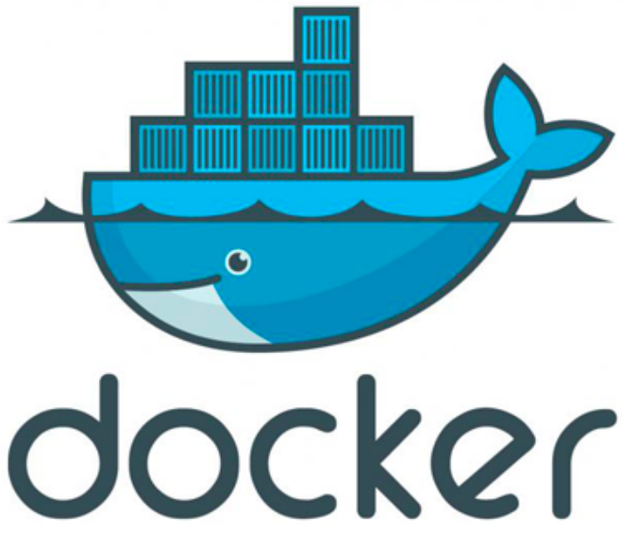


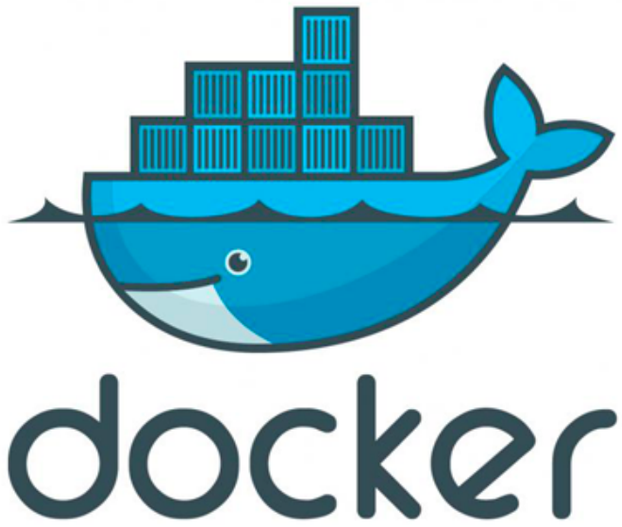
Banco de dados NoSQL - Introdução ao Docker (Parte 1)

Prof. Gustavo Leitão



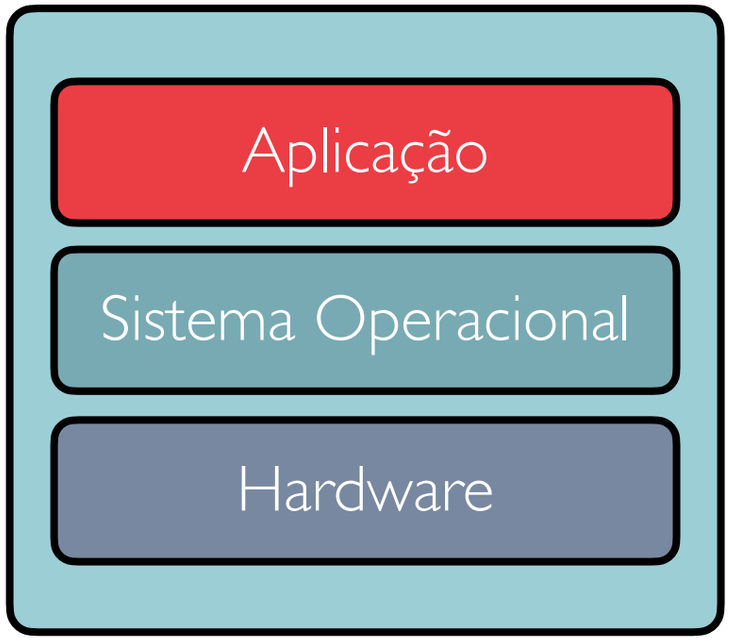
Docker é uma tecnologia de software que fornece contêineres, que fornece uma camada adicional de abstração e automação de virtualização de nível de sistema operacional no Windows e no Linux.

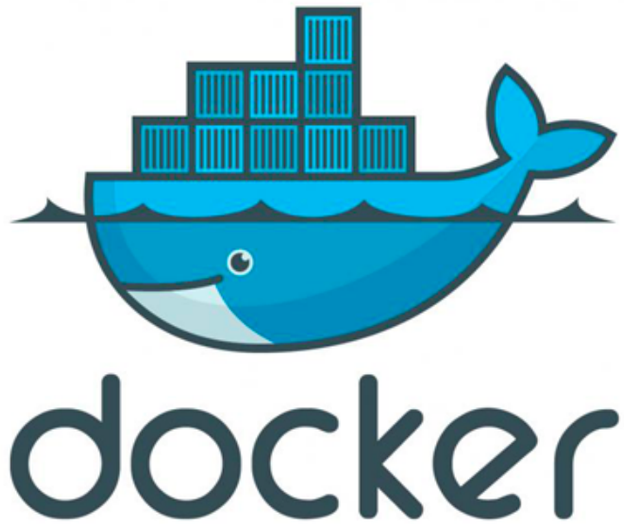
O Docker usa as características de isolamento de recurso do núcleo do
Linux como cgroups e espaços de nomes do núcleo, e um sistema de arquivos com recursos de união, como OverlayFS e outros para permitir "contêineres" independentes para executar dentro de uma única instância Linux, evitando a sobrecarga de iniciar e manter máquinas virtuais (VMs).



Uma aplicação em um servidor físico

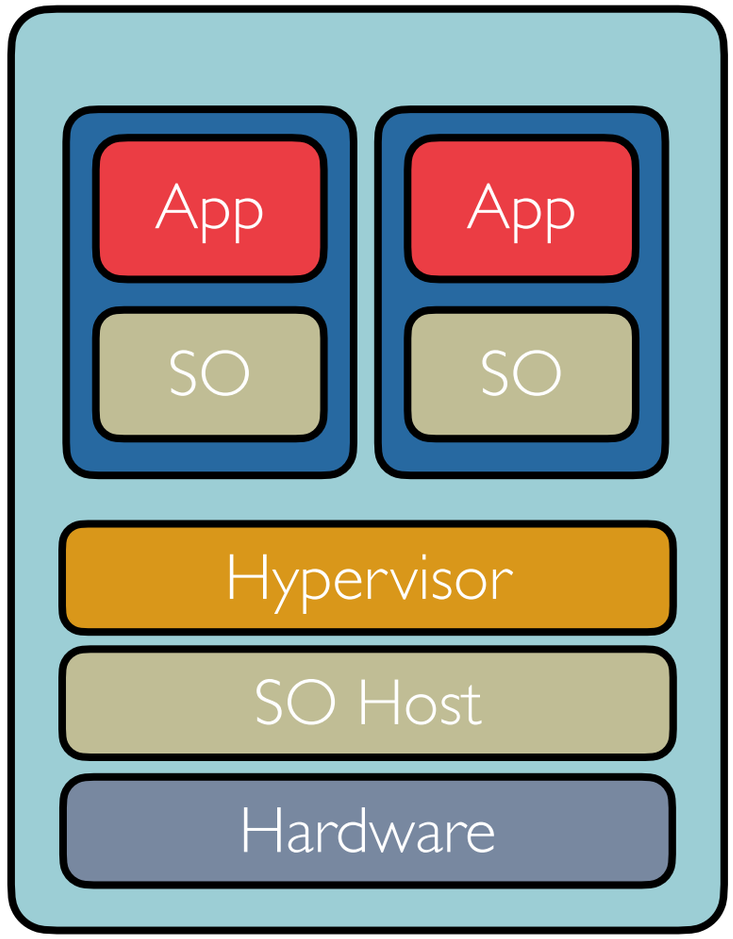
- Lentidão para *deploy*
- Altos custos
- Desperdício de recursos
- Dificuldade para escalar
- Dificuldade para migrar
- Forte dependência do ambiente.

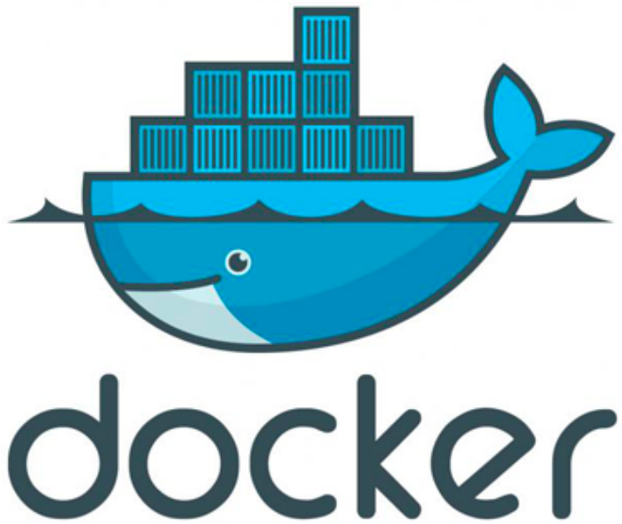




Um servidor físico para múltiplas aplicações

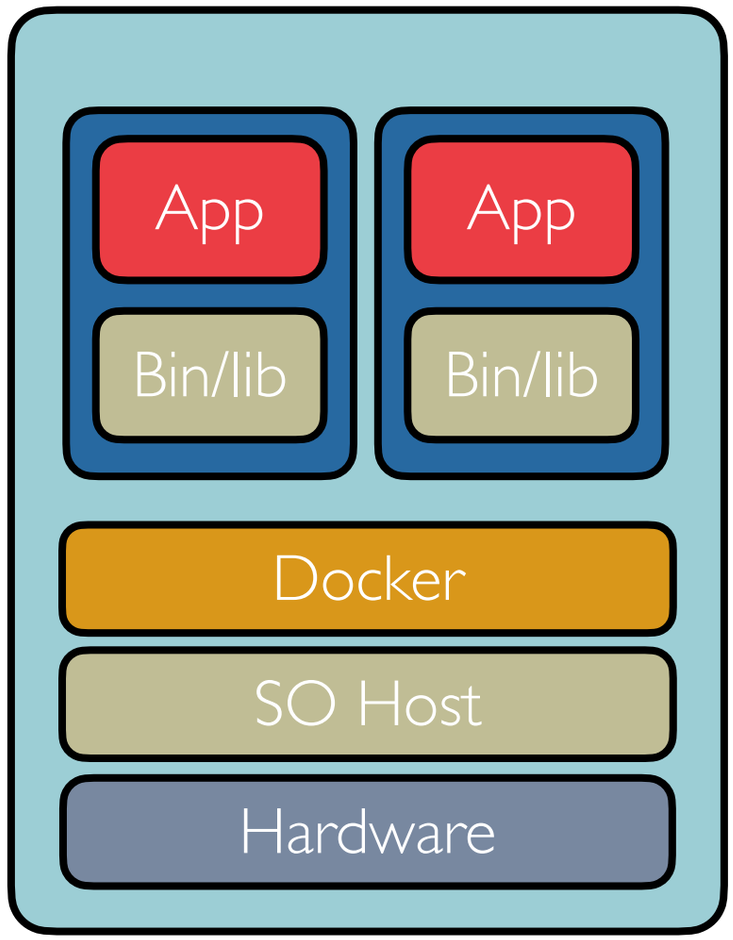
- Melhor gestão dos recursos
- Mais fácil para escalar
- Cada VM requerer pré-alocação de hardware
- Cada VM precisa de um SO.
- Um SO por Vm implica em desperdício de recursos
- Portabilidade da aplicação não é garantido

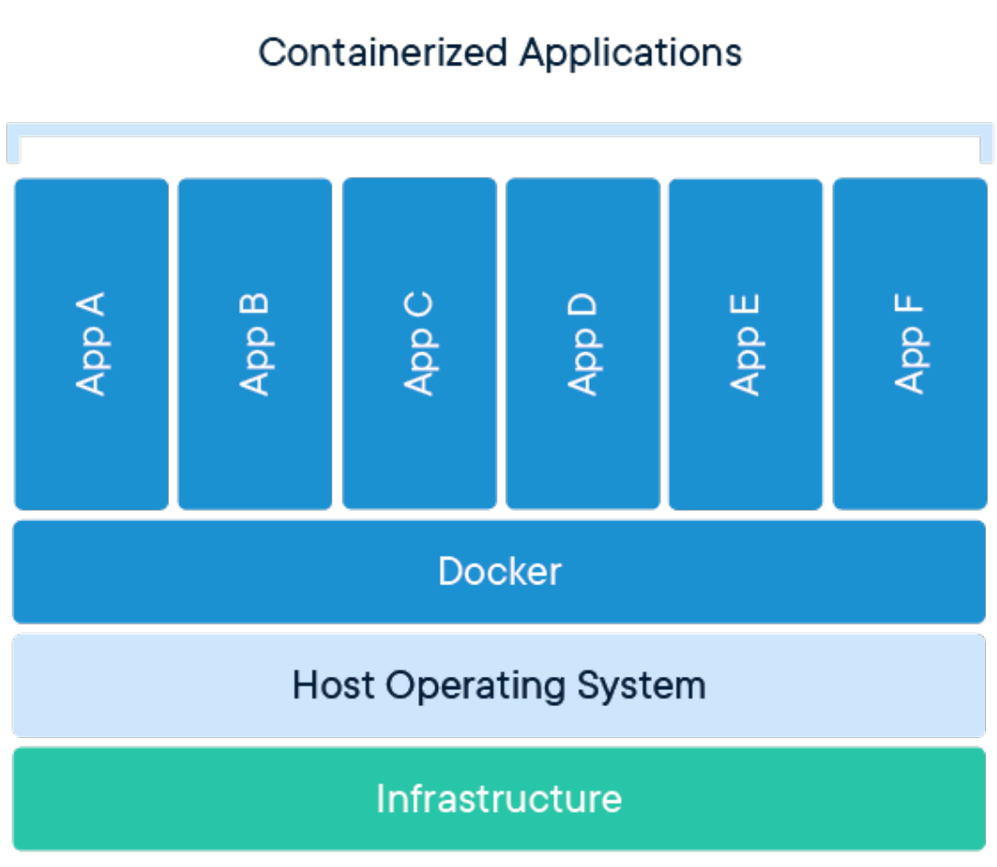




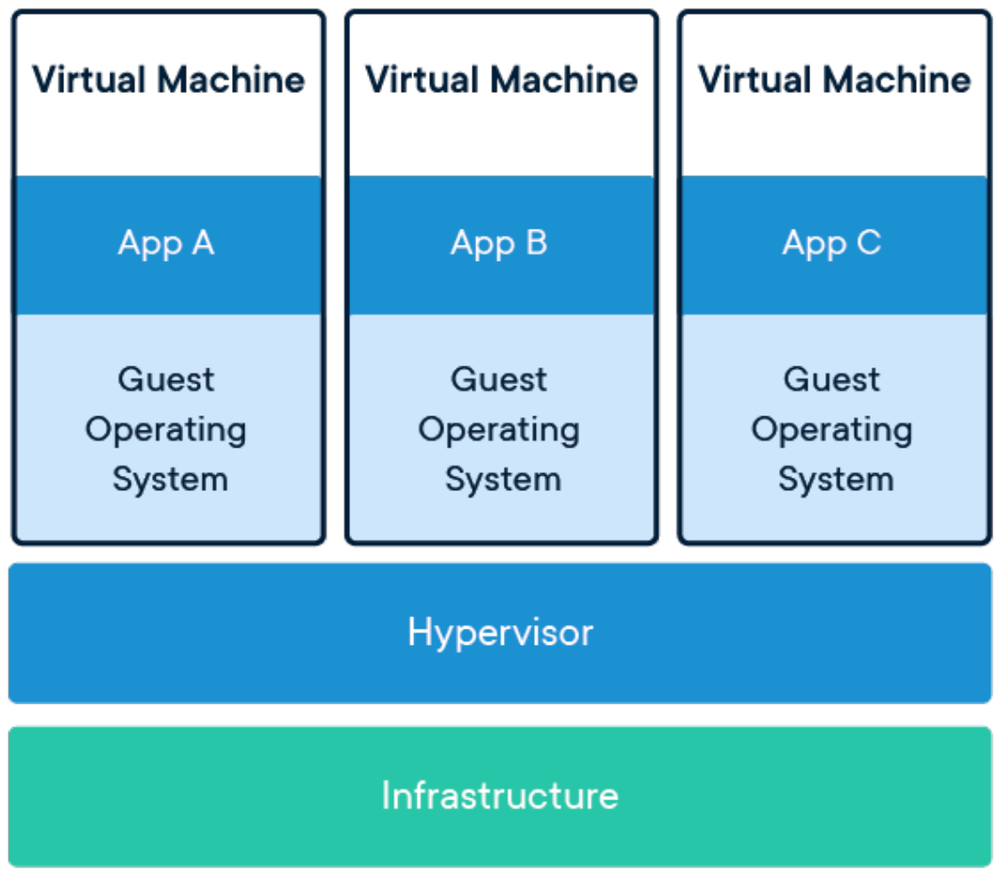
Com Contêineres

- Maior velocidade de boot, pois não existe um SO para iniciar
- Menor dependência (maior facilidade em mudar entre diferentes infraestruturas)
- Maior eficiência: menor uso do OS.





Contêiner são construídos em nível de aplicação



VMs são componentes em nível de infraestrutura.

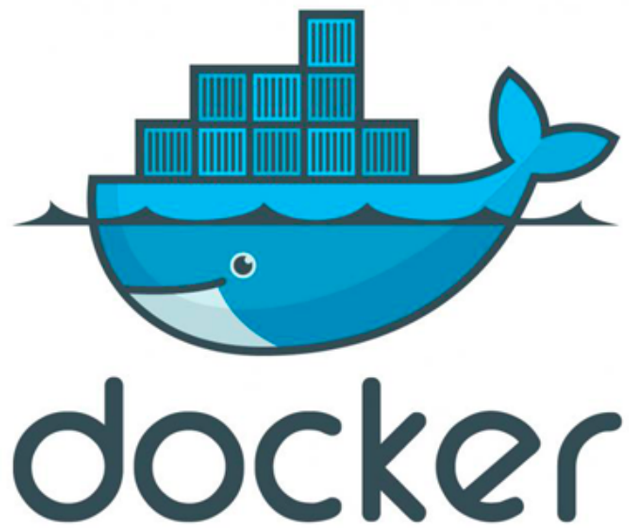


Imagem: A base de um contêiner. Artefato que contém tudo necessários para execução da aplicação.

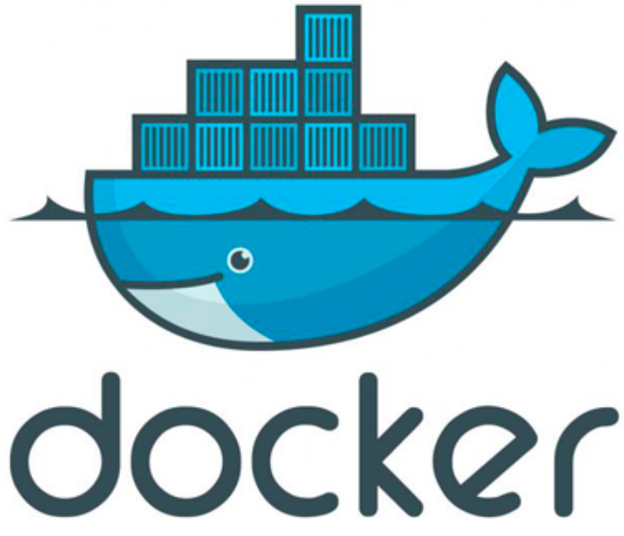
Container: Imagem em execução.

Registry: Armazena, distribui e gerencia imagens Docker.

Volumes: Área de armazenamento de dados de um contêiner

Network: Rede que conecta os contêiner

Docker Hub: Registry público de imagens Docker

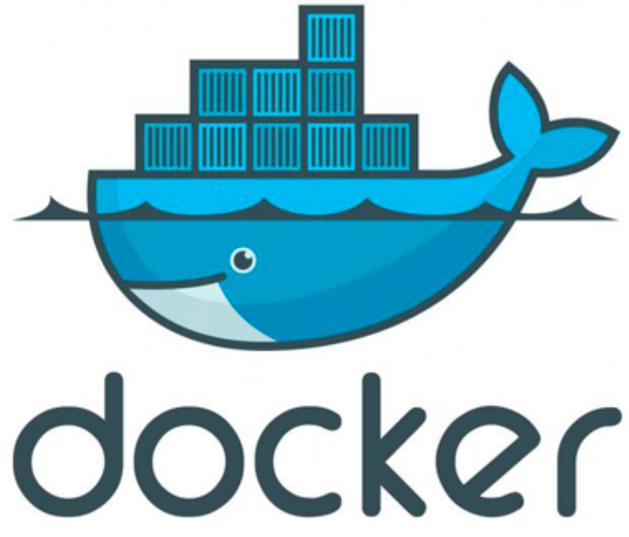


Lista imagens disponíveis localmente

```
$ docker images
```

Lista *contêineres* em execução

```
$ docker ps
```

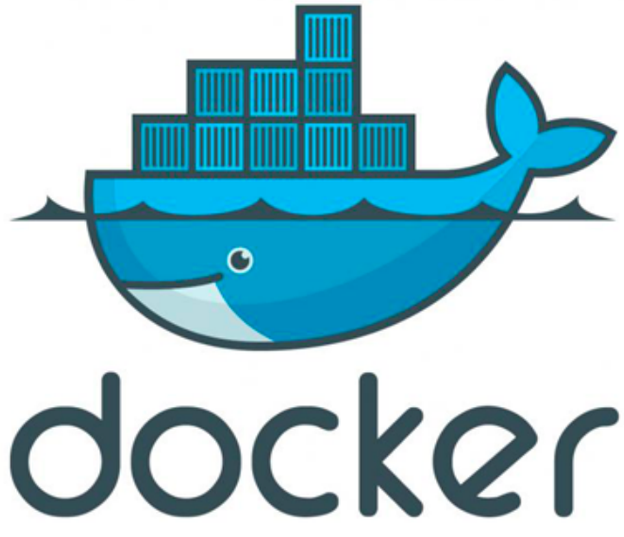


Lista todos os contêineres

```
$ docker ps -a
```

Executa uma imagem (criando novo contêiner)

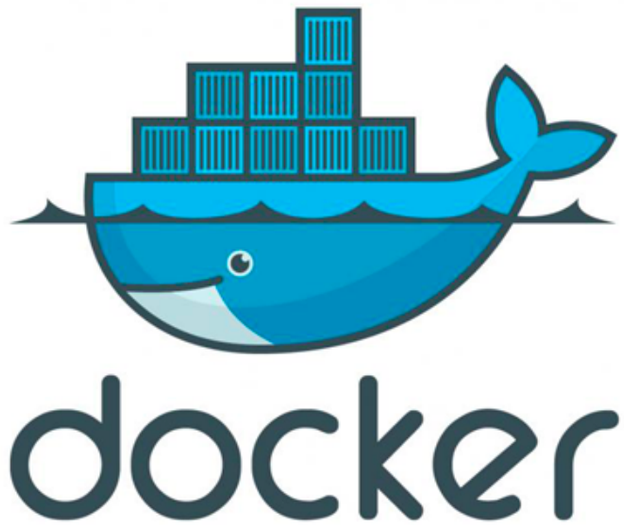
```
$ docker run #nome-imagem:tag
```



Executando primeira imagem

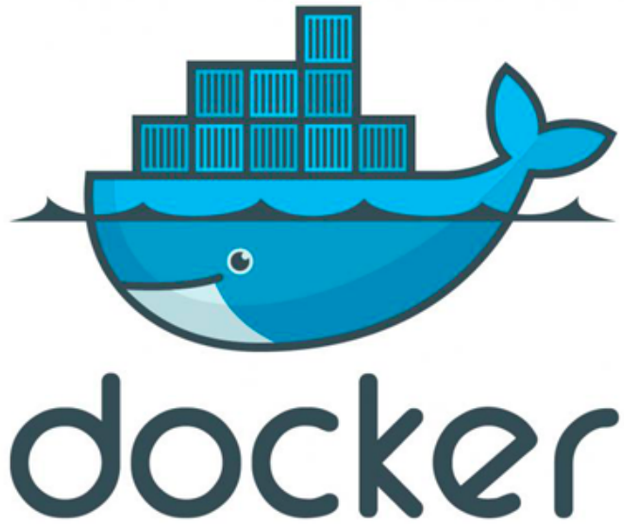
```
$ docker run hello-world
```

Caso não seja encontrado a imagem localmente, o Docker tentará realizar download da imagem no registro público.



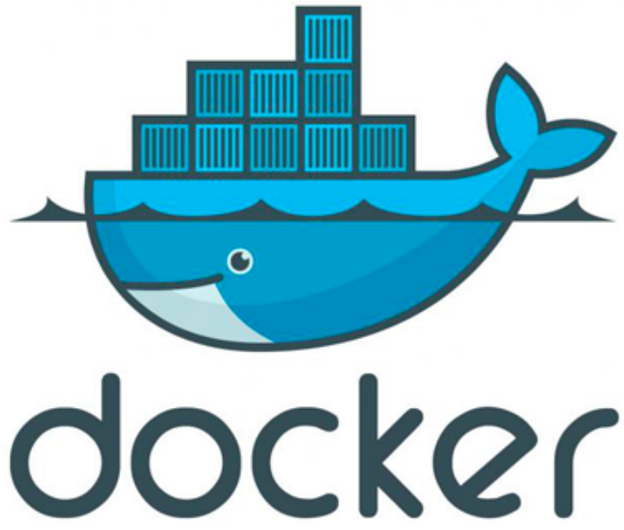
Como criar uma imagem?!

1. Cria um arquivo Dockerfile com descrição para sua aplicação
2. Criar uma imagem através do comando `docker build`



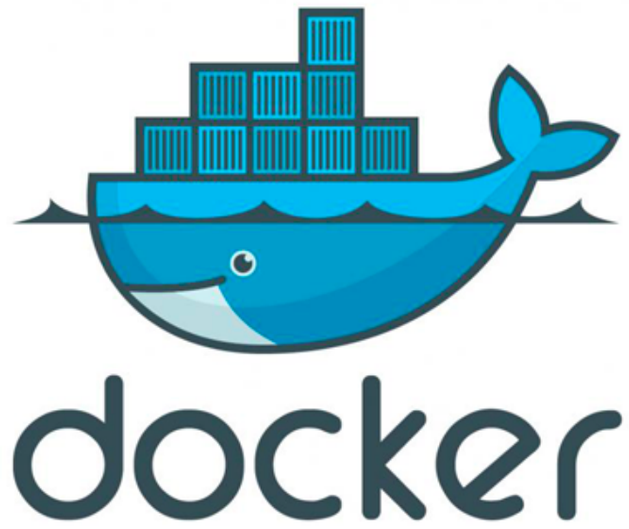
Crie o código de sua aplicação (app.js)

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port $
{port}!`))
```



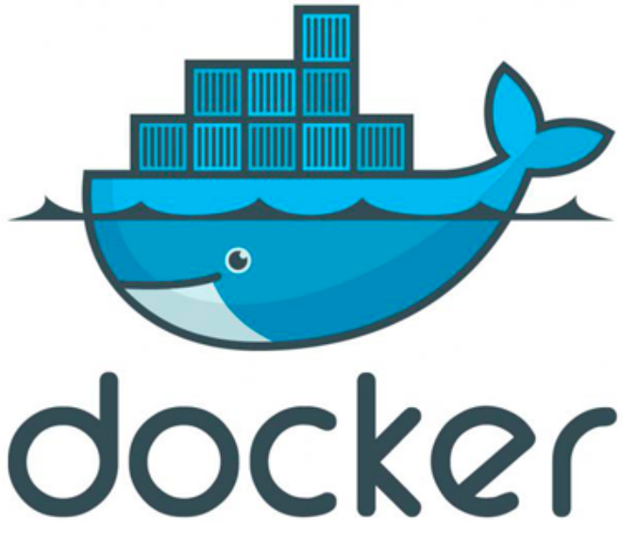
Crie o código de sua aplicação (package.json)

```
{
  "name": "nodejs-image-demo",
  "version": "1.0.0",
  "description": "nodejs image demo",
  "author": "Gustavo Leitão <gustavo.leitao@imd.ufrn.br>",
  "main": "app.js",
  "dependencies": {
    "express": "^4.16.4"
  }
}
```



Crie o Dockerfile

```
FROM node:10
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . /usr/src/app
EXPOSE 3000
CMD [ "node", "/usr/src/app/app.js" ]
```



Crie a imagem

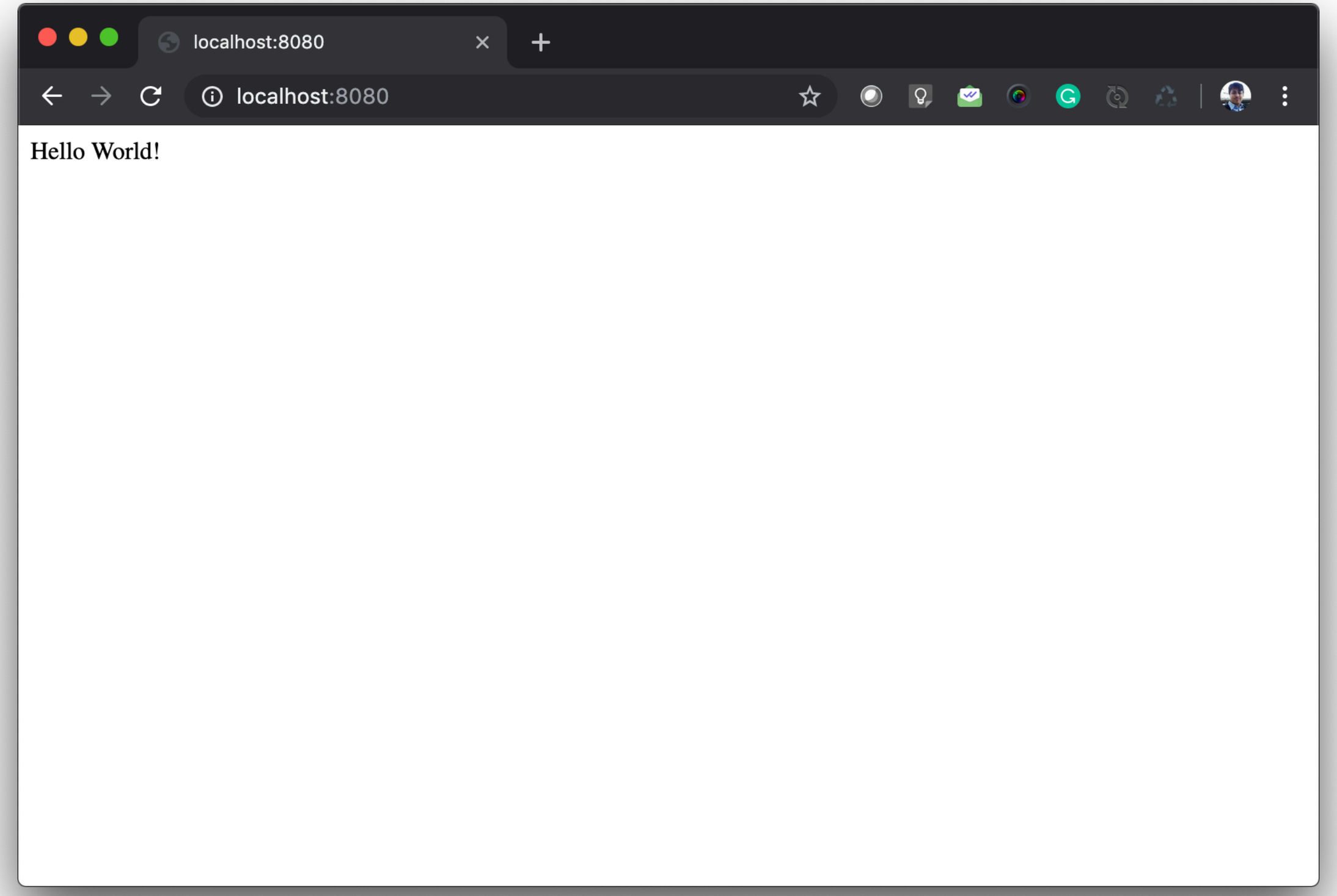
```
$ docker build node-sample:tag .
```

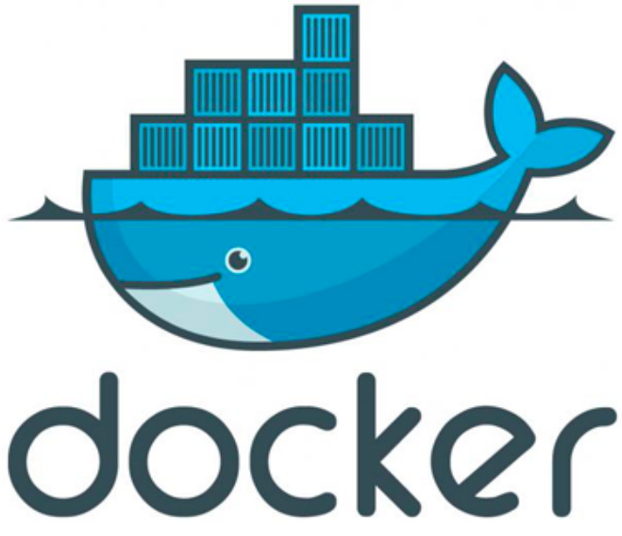
Execute a imagem em *detatch* e com mapeamento de portas

```
$ docker run -p 8080:3000 -d nome-sample:tag
```

-p vai mapear a porta local 8080 na porta interna do contaminar 3000

-d vai executar o container em background


localhost:8080
localhost:8080
Hello World!

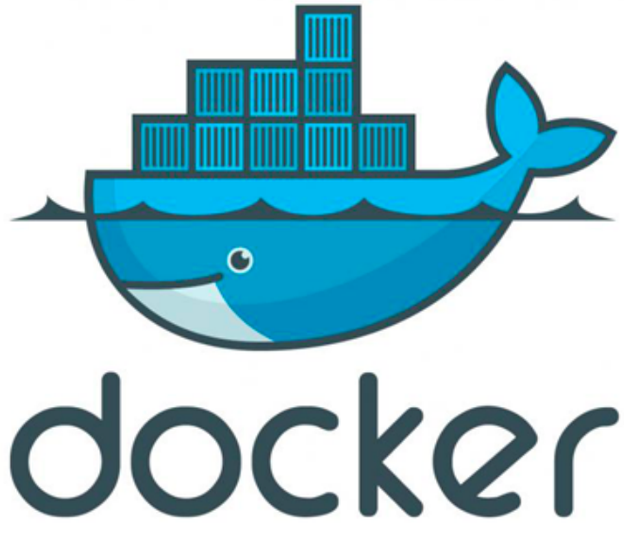


Iniciando um container existente

```
$ docker start #container id | #container name
```

Parando um container em execução

```
$ docker stop #container id | #container name
```

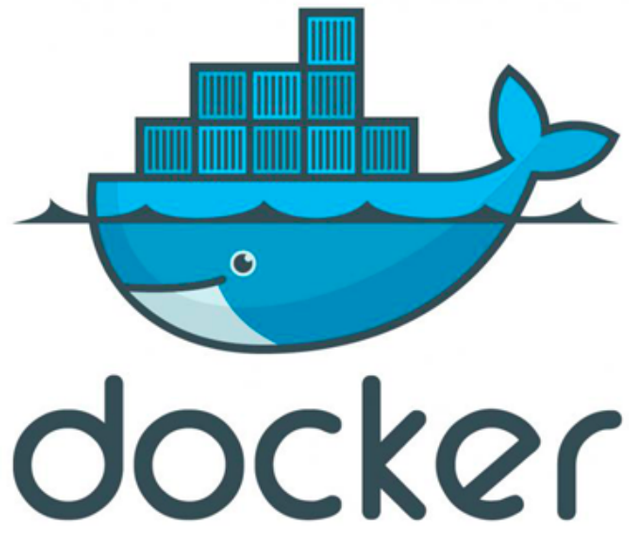


Removendo um container

```
$ docker rm #container id | #container name
```

Removendo uma imagem

```
docker rmi #image id:tag
```



Executando um comando dentro do Container

```
$ docker exec -it #container_name #comando
```

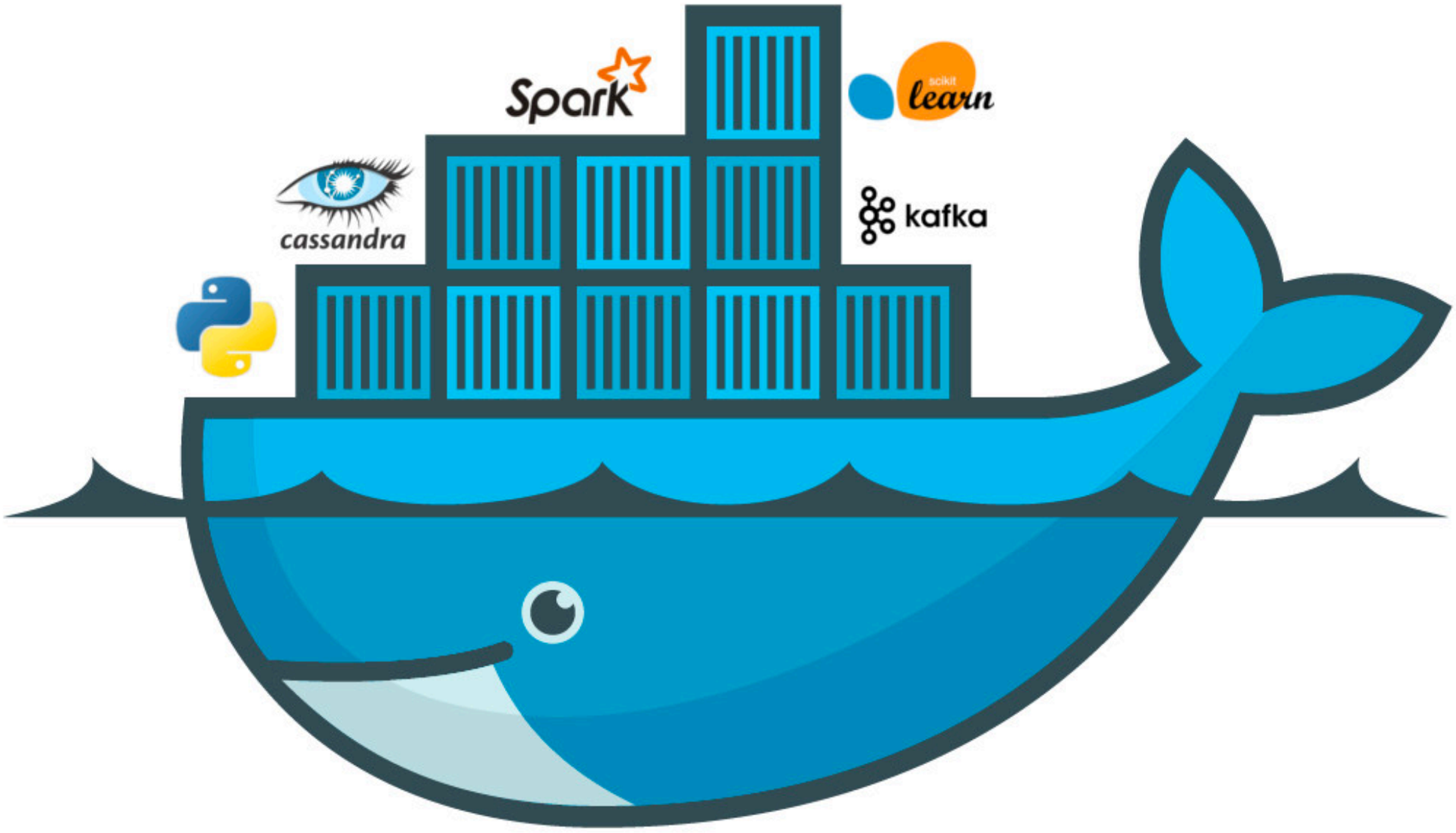
-i: significa interativo

-t: Alocação de um pseudo-tty

Exemplo:

```
docker exec -it node-sample bash
```

Executa bash no container de nome "node-sample"



Obrigado!

Prof. Gustavo Leitão