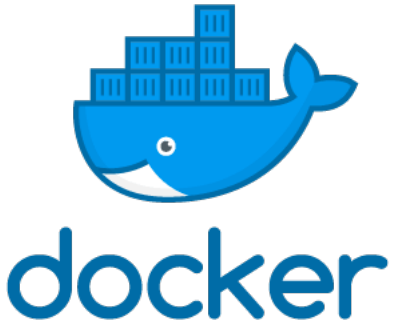


Docker: Criando containers sem dor de cabeça

VI Cimatech

Prof. MSc. Lucas G. Nadalete

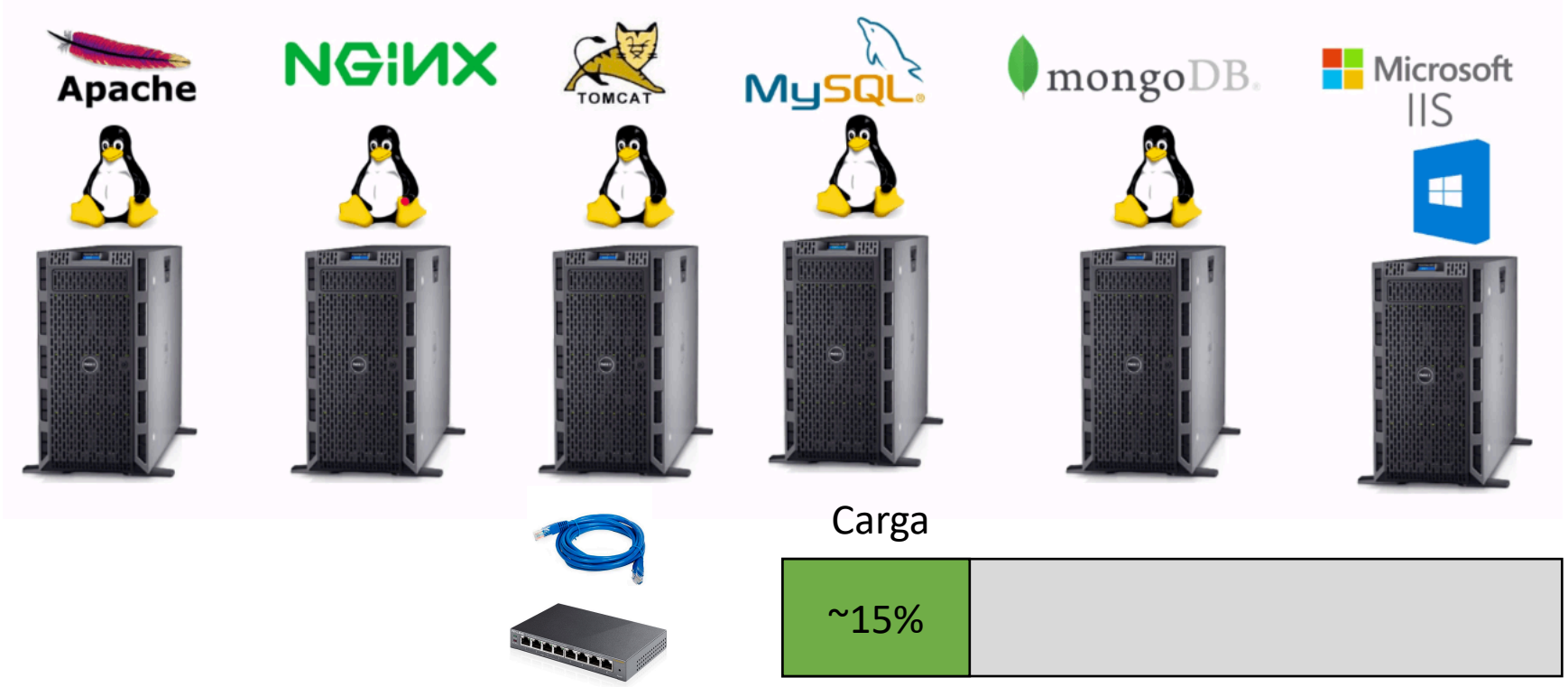
lucas.nadalete@fatec.sp.gov.br

Evolução do host de aplicações

- Várias aplicações, rodando em vários servidores
- Custo de rede
- Custo de energia
- Demais custos inerentes a manutenção de uma infraestrutura

docker

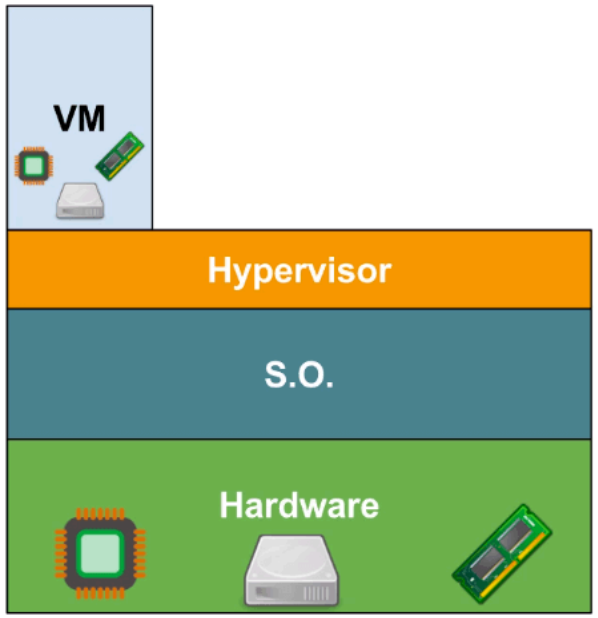
Capacidade pouco aproveitada



- Muito tempo ocioso!
- Muitos recursos desperdiçados!

docker

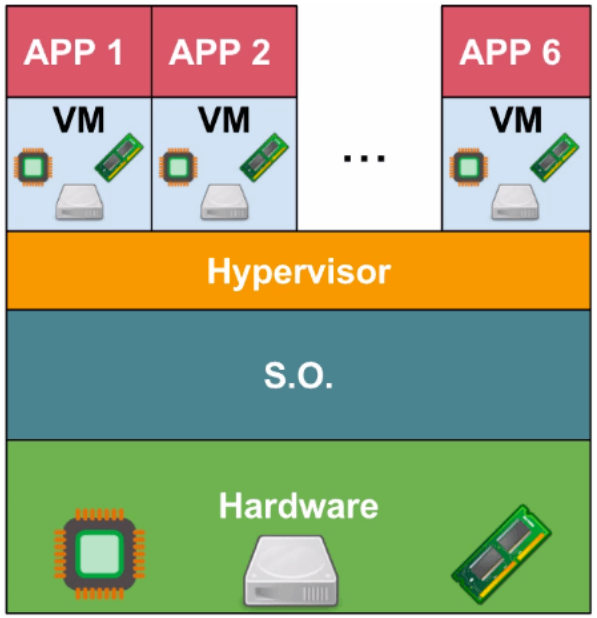
Virtualização como solução



Hypervisor – Tecnologia que roda sobre o sistema operacional, permitindo a virtualização dos seus recursos físicos

docker

Virtualização como solução

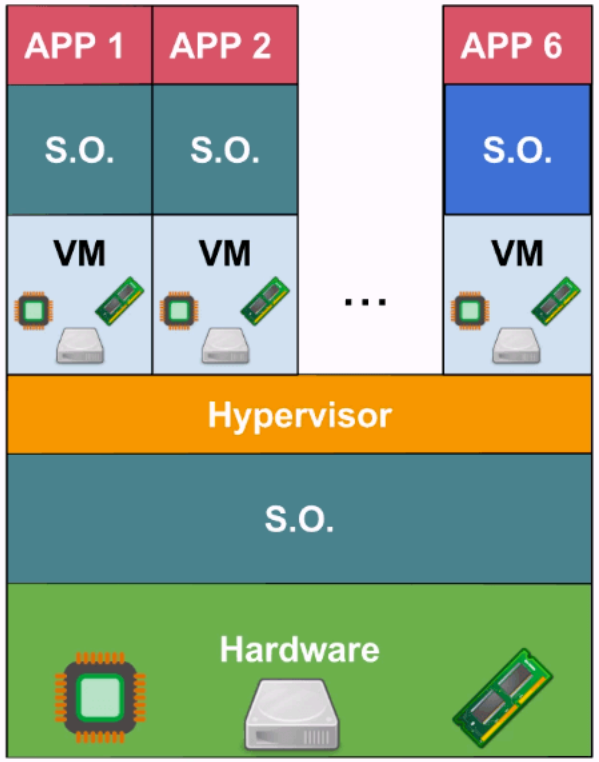


Hypervisor – Tecnologia que roda sobre o sistema operacional, permitindo a virtualização dos seus recursos físicos

docker

Problemas das máquinas virtuais

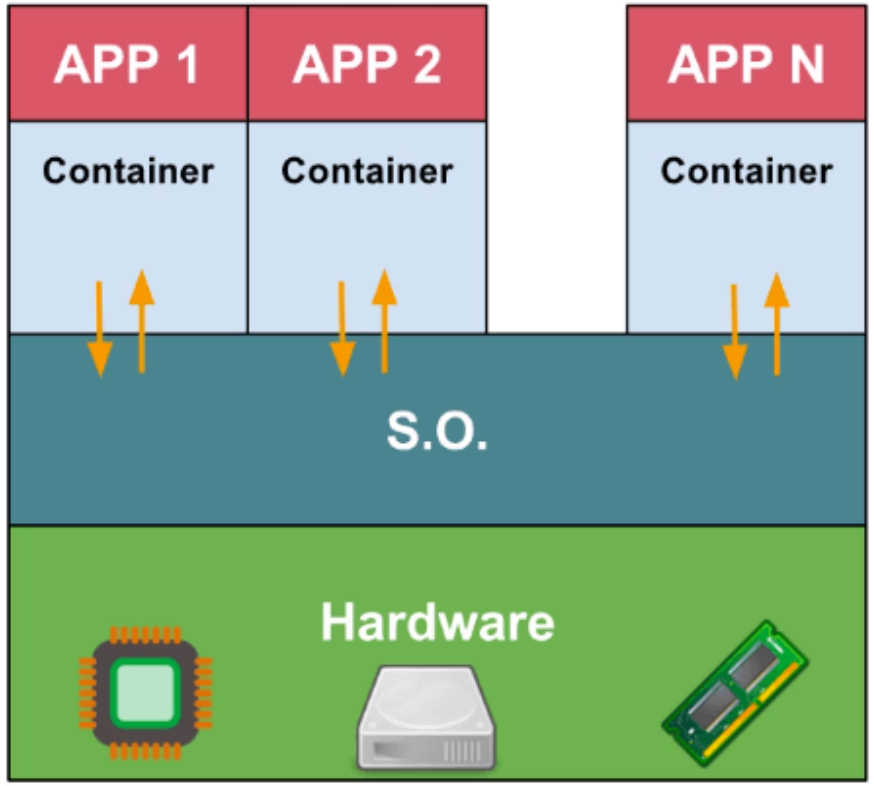
- Cada app necessita de um SO
- Cada SO executado possui seus próprios recursos de RAM, disco e processamento
- Precisamos de uma capacidade minima para rodar os recursos
- Custo de configuração e manutenção



docker

Como melhorar a situação?

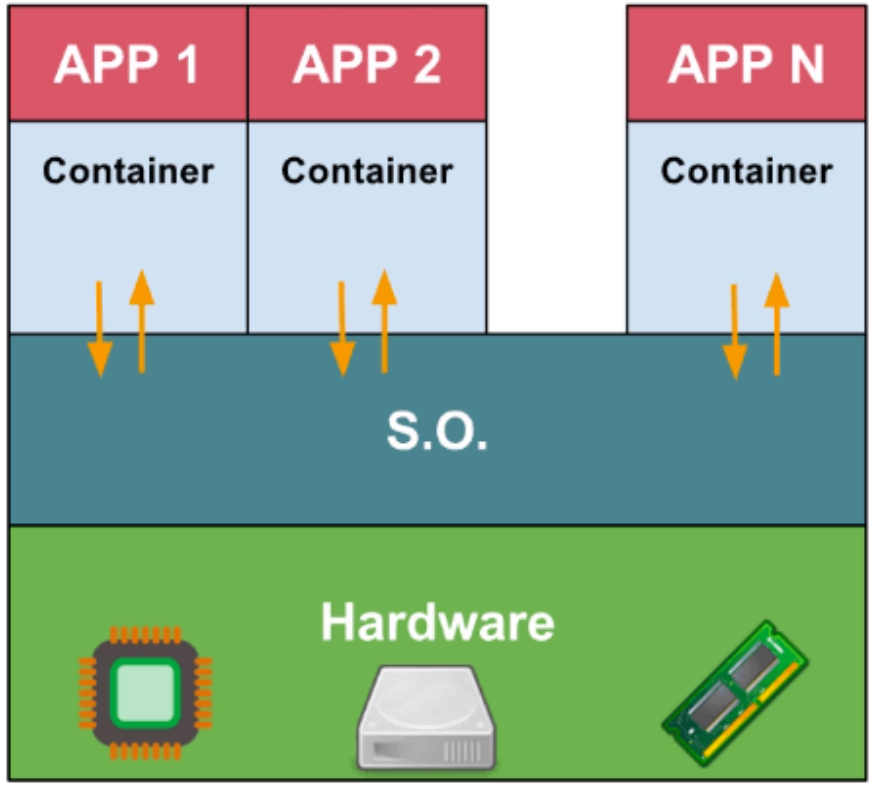
- **Container**
- Não temos mais um SO para cada aplicação!



docker

Vantagens de *containers*

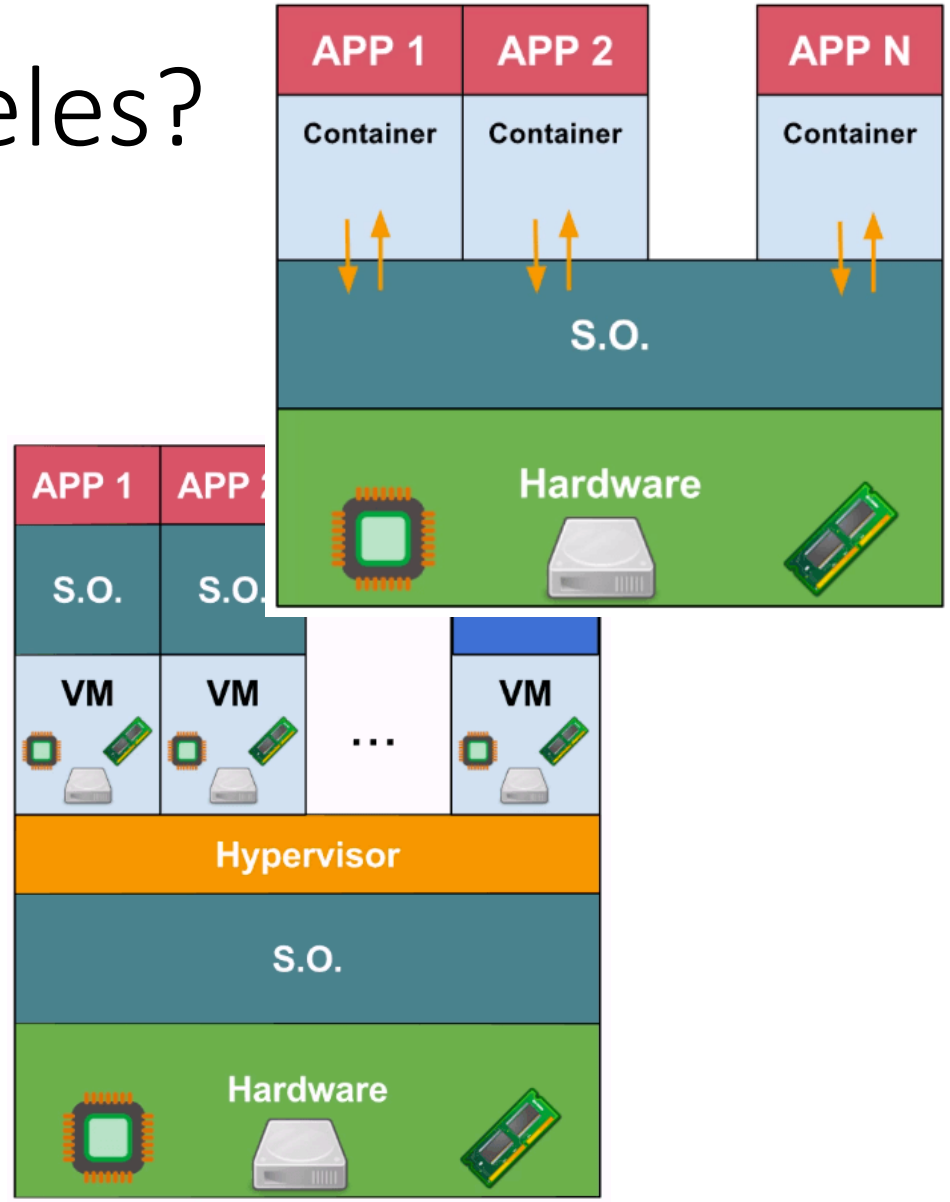
- Mais leve
- Não possui custo de manutenção de múltiplos SOs
- Mais rápido (*up/down*)
- Solução para múltiplas máquinas virtuais em um mesmo hardware físico



docker

Mas por que precisamos deles?

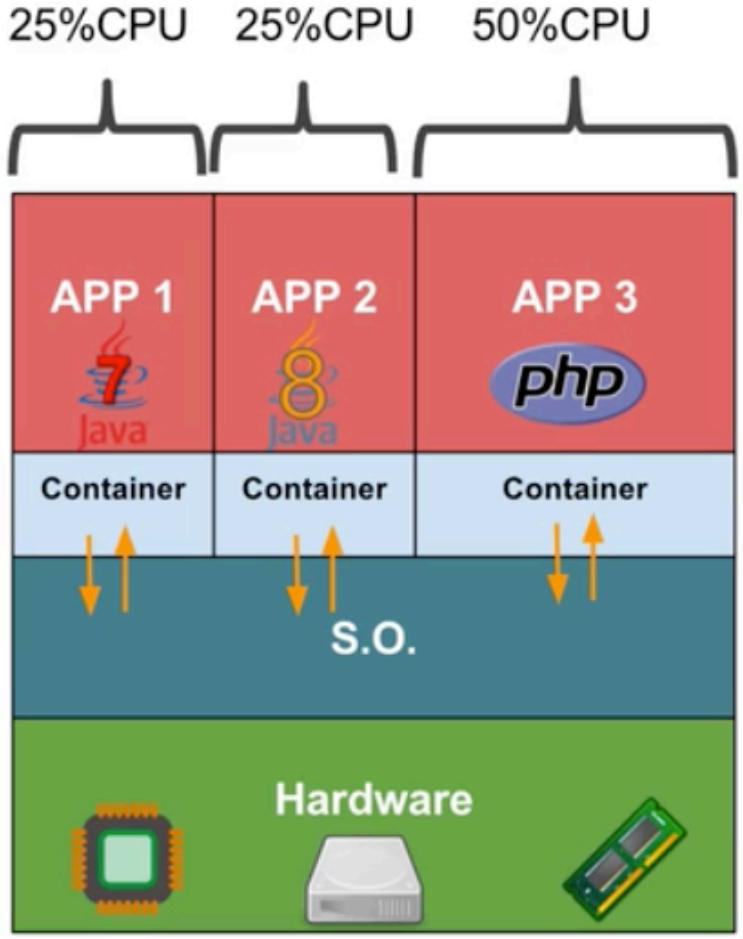
- Dois apps utilizando a mesma porta de rede?
- E se um app começar a consumer muito de um recurso, como a CPU?
- E se cada app precisar de uma versão específica de uma linguagem?
- E se um app congelar todo o SO?



docker

Utilizando *containers*

- Ganhamos
  - Melhor controle sobre o uso de cada recurso (CPU, Disco, Rede e outros)
  - Agilidade na hora de criar ou remover um container
  - Maior facilidade na hora de trabalhar com diferentes versões de linguagens e bibliotecas
  - Mais leves que as VMs



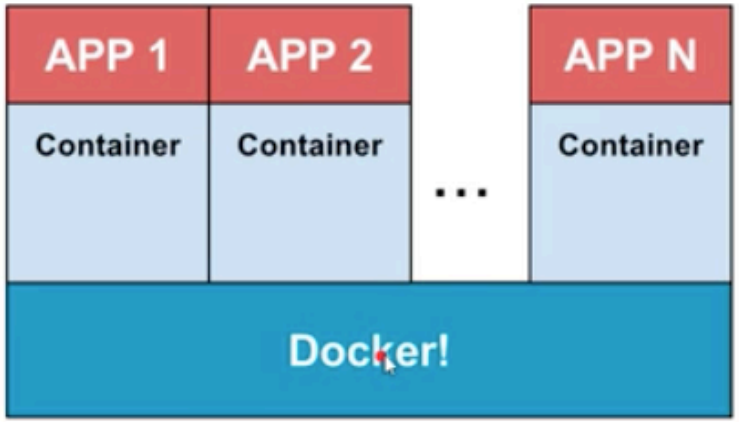
docker

Docker, Inc.

- No início → dotCloud – PaaS (Heroku, Microsoft Azure e outros)
- dotCloud → AWS como infraestrutura
- dotCloud introduziu o conceito de “containers” na hora de subir uma aplicação
- dotCloud criou o Docker
- dotCloud tornou-se a Docker, Inc.

Docker – A(s) tecnologia(s)

- Tecnologias de containers para prover ferramentas modernas para ***deployar*** e rodar aplicações



Docker Engine

docker

Outras tecnologias do Docker

- **Docker Compose**: Um jeito fácil de definer e orquestrar múltiplos containers.

- **Docker Swarm**: Uma ferramenta para colocar múltiplos Dockers Hosts para trabalharem juntos em um cluster

docker

Outras tecnologias do Docker

- **Docker Hub**: Um repositório com mais de 250 mil imagens diferentes para os seus *containers*

- **Docker Machine**: Uma ferramenta que nos permite instalar e configurar em host virtuais

Instalação - Windows

https://store.docker.com/editions/community/docker-ce-desktop-windows

- Docker CE (Community Edition) - 32 ou 64 bits
- O Docker é executado em cima de uma micromáquina virtual, chamada **Alpine Linux**, onde será executada a sua Docker Engine
- O Docker precisa utilizar uma tecnologia chamada de Hyper-V, que é um Hypervisor. O problema disto é que o Hyper-V só está presente nas versões Professional, Education e Enterprise, ou seja, a maioria dos usuários comuns, que utilizam a versão Home Edition, não poderão instalar o Docker pelo modo tradicional, e terá que utilizar o Docker Toolbox
  - Docker Engine
  - Docker Compose
  - Docker Machine
  - Docker Kitematic

Instalação - Mac OS e Ubuntu Linux

https://store.docker.com/editions/community/docker-ce-desktop-mac

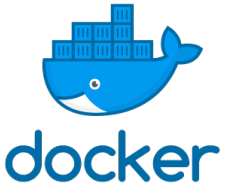
- Mas para criar máquinas virtuais, o Docker precisa utilizar uma tecnologia chamada de HyperKit, que é um Hypervisor. O problema disto é que o HyperKit só está presente na versão OS X El Capitan 10.11 ou mais recente Mas uma alternativa é instalar o Docker Toolbox

https://docs.docker.com/install/linux/docker-ce/ubuntu

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io



Play with Docker – Ferramenta Alternativa

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

docker

Testando o Docker

```
$ docker version
```

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

Container Ubuntu

```
$ docker run ubuntu

$ docker ps

$ docker ps -a
```

Container Ubuntu – Executando um comando

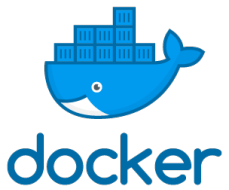
```
$ docker run ubuntu echo “Ola Mundo”
```

```
$ docker run –it ubuntu (CTRL + D ou exit)
```

Abrir um novo terminal

```
$ docker start <id>
```

```
$ docker stop <id>
```



Container Ubuntu – Executando um comando

```
$ docker run ubuntu echo “Ola Mundo”
```

```
$ docker run –it ubuntu (CTRL + D ou exit)
```

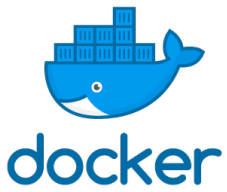
Abrir um novo terminal

```
$ docker start <id>
```

```
$ docker stop <id>
```

```
$ docker start -a -i <id>
```

```
-it  - interactive
-a   - attached
-i   - interactive
```



Container Ubuntu – Executando um comando

Remoção de container

```
$ docker rm <id>
```

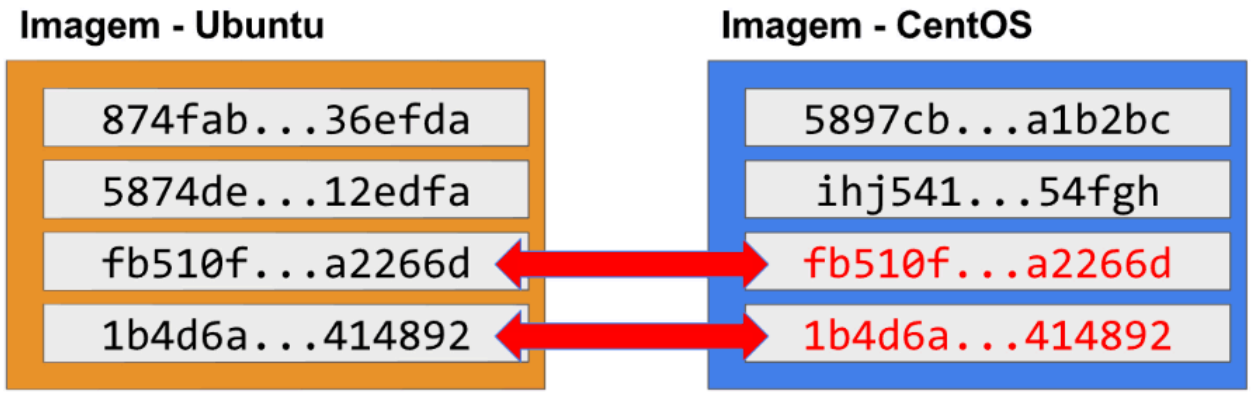
```
$ docker container prune
```

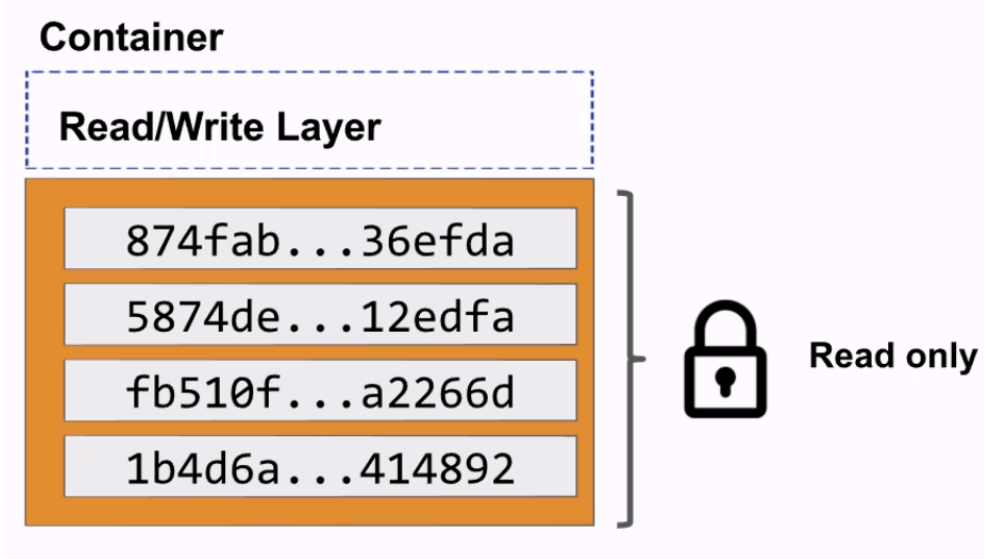
Remoção de imagens

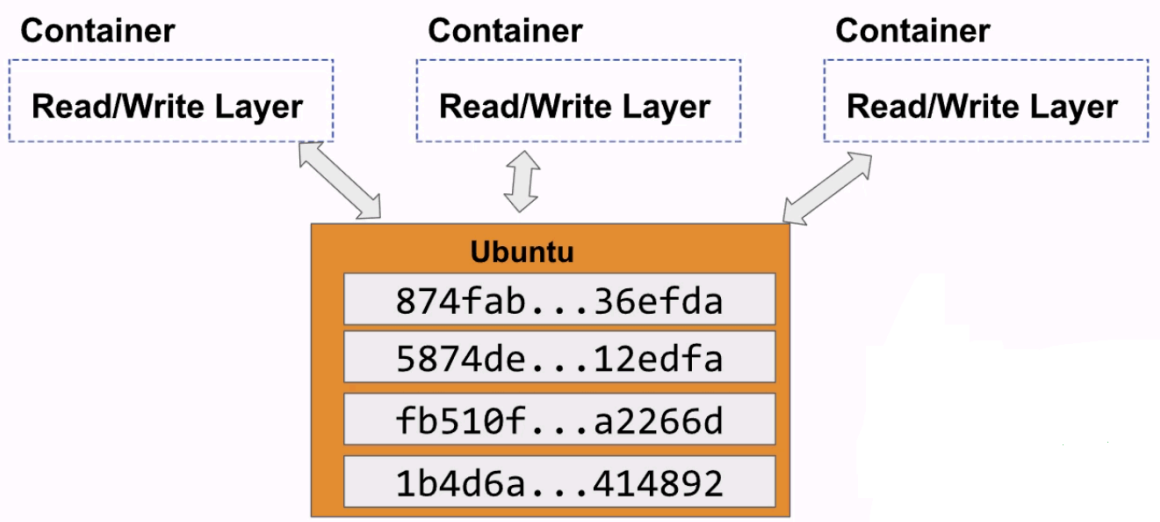
```
$ docker images
```

```
$ docker rmi <image>
```

Docker - *Layered File System*





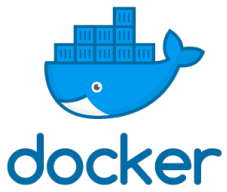


docker

Container Ubuntu – Atribuindo porta

```
$ docker run –d dockersamples/static-site

$ docker ps

$ docker run -d –P dockersamples/static-site

$ docker port <id>
```

-d - daemon
-P – random port



Container Ubuntu – Nomeando container

```
$ docker run –d –P –name meu-site
dockersamples/static-site

$ docker stop meu-site
```

Container Ubuntu – Definindo porta de rede

```
$ docker run –d –p 12345:80
dockersamples/static-site

$ docker port <id>
```

Container Ubuntu – Variável de ambiente

```
$ docker run –d –P –e AUTHOR=“Nadalete”
dockersamples/static-site
```

Container Ubuntu – Parando todos os *containers*

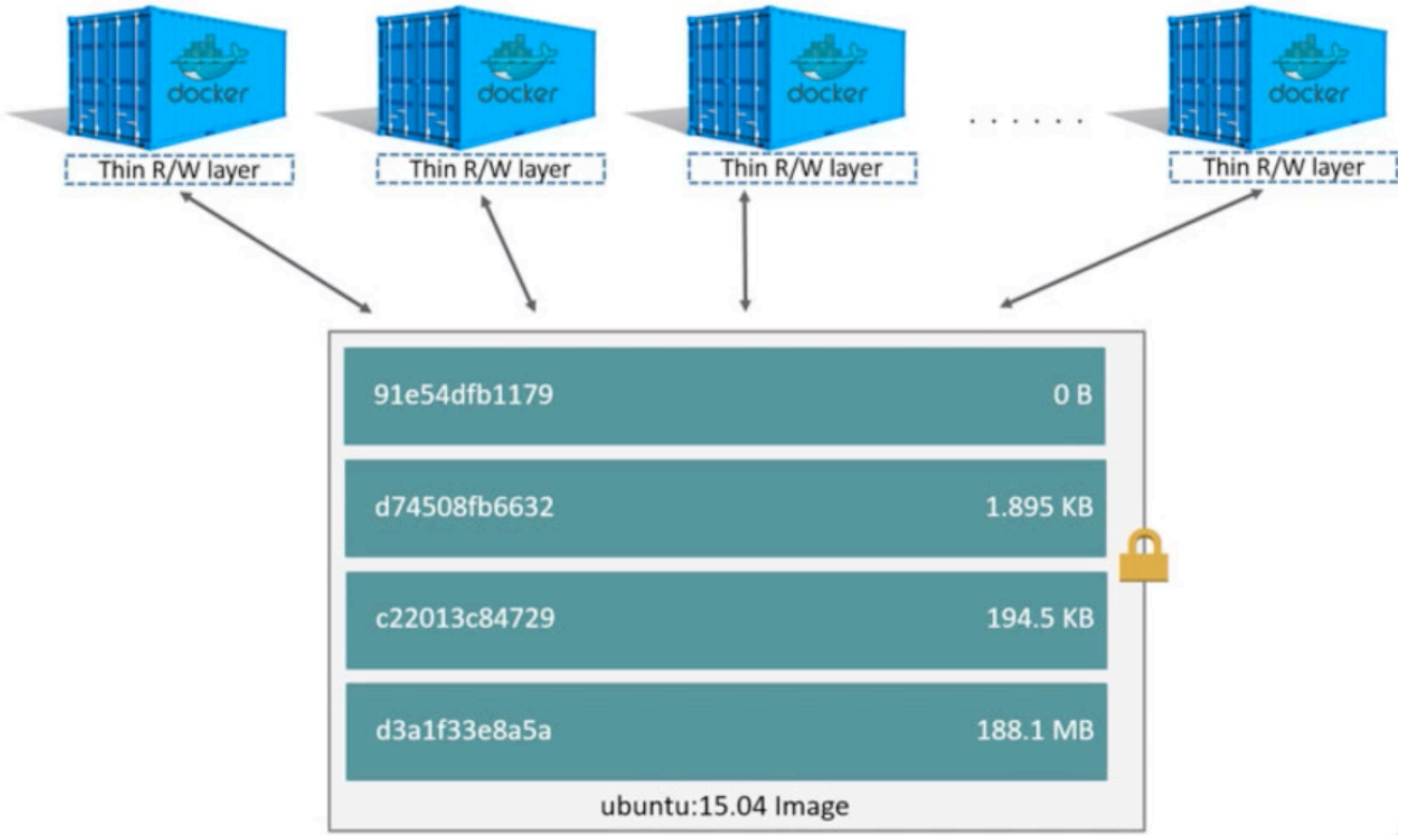
```
$ docker stop $(docker ps –q) – timeout 10s
```

```
$ docker stop –t 0 $(docker ps –q)
```

Container Ubuntu – Remove todos os *containers* parados

```
$ docker container prune
```

Docker – Volumes (Docker Host)

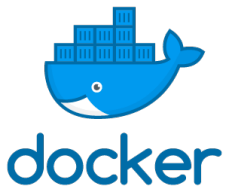


docker

Container Ubuntu – Trabalhando com volumes

```
$ docker run –v “/var/www” ubuntu

$ docker inspect <id>

$ docker run –it –v “/tmp/minicurso:/var/www”
ubuntu

$ cd /var/www
$ touch novo-arquivo.txt
$ echo “Texto de teste” > novo-arquivo.txt
```

-v - volume



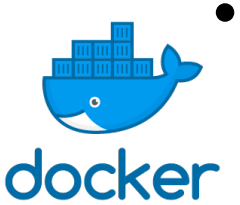
Container Ubuntu – Rodando um projeto

Projeto

- http://shorturl.at/bdBQ7
- `cd /tmp`
- `wget -O volume-exemplo.zip` http://shorturl.at/bdBQ7
- `unzip volume-exemplo.zip`

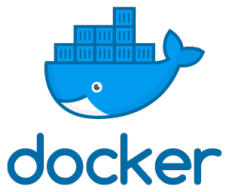
Objetivo

- Usar o código do projeto do link como base para rodarmos uma aplicação em NodeJS



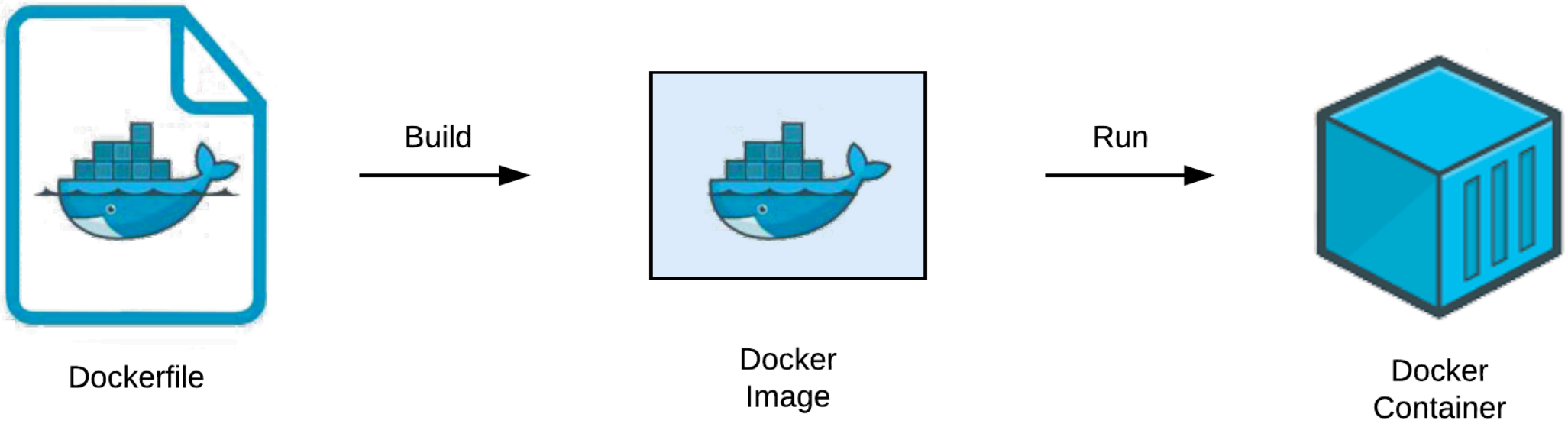
Projeto

```
$ docker run node #baixa a imagem node:latest

$ docker run –p 8080:3000 –v “/tmp/volume-
exemplo:/var/www” node npm start

$ docker run –p 8080:3000 –v “/tmp/volume-
exemplo:/var/www” –w “/var/www” node npm start
```



-w - work dir

Criando nossa própria Imagem - Dockerfile

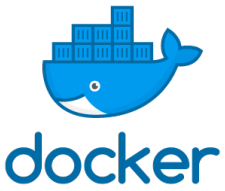


Criando nossa própria Imagem - Dockerfile

Na pasta **volume-exemplo**

```
$ touch Dockerfile #node.dockerfile
```

```
FROM node:latest
MAINTAINER <seu_nome>
ENV PORT=3000
COPY . /var/www
WORKDIR /var/www
RUN npm install
ENTRYPOINT npm start
EXPOSE $PORT
```



Criando nossa própria Imagem - Dockerfile

Na pasta **volume-exemplo**

```
$ docker build –f Dockerfile -t lnadalete/node
```

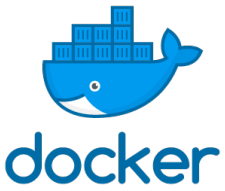
ou

```
$ docker build –t lnadalete/node .
```

```
$ docker images
```

Para executar

```
$ docker run -d –p 8080:3000 lnadalete/node
```

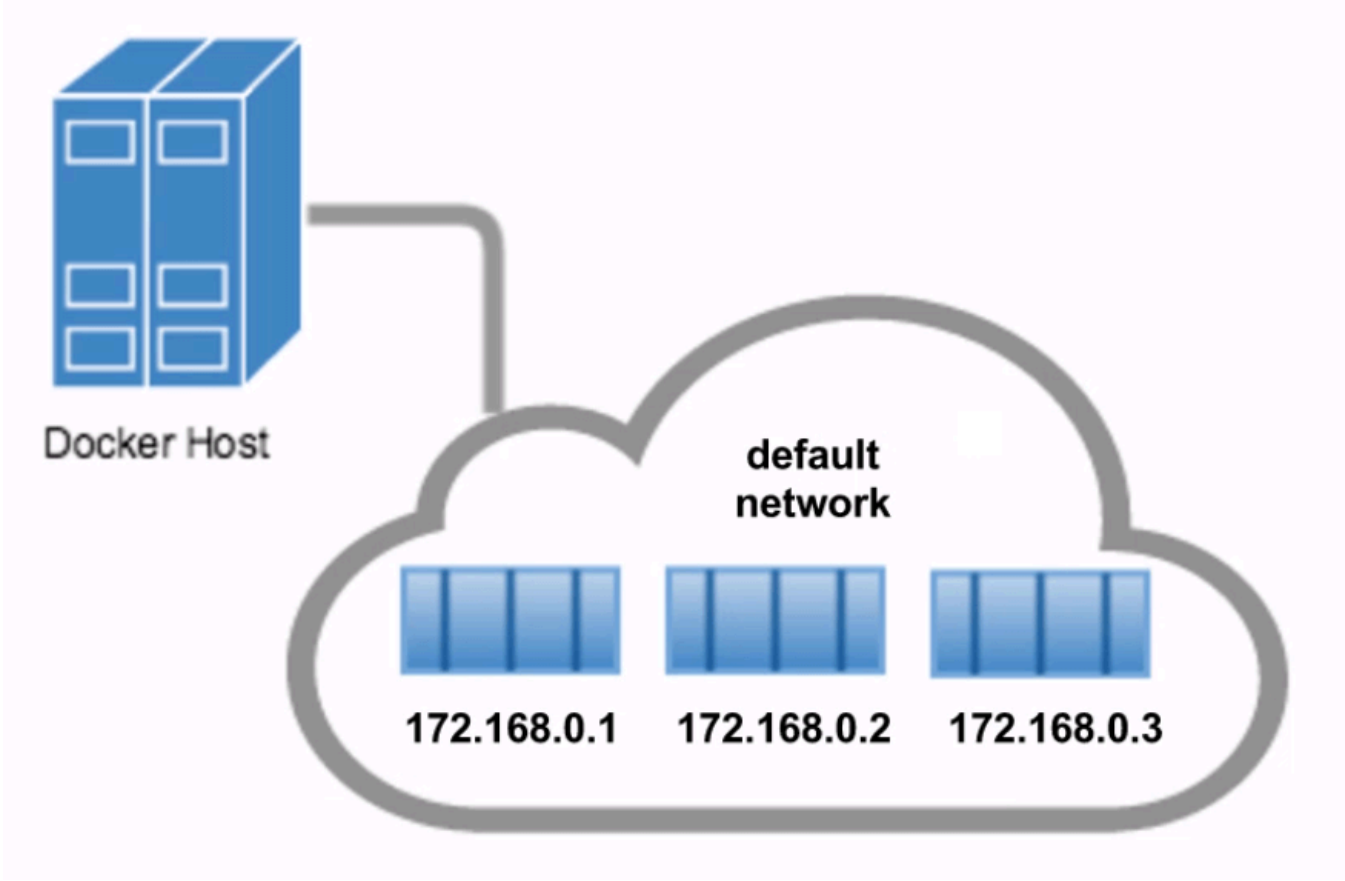


Analisando outro Dockerfile

```
$ cd /tmp
```

```
$ git clone
https://github.com/lucasnadalete/testlink-docker.git
```

Redes com o Docker



docker

Redes com o Docker

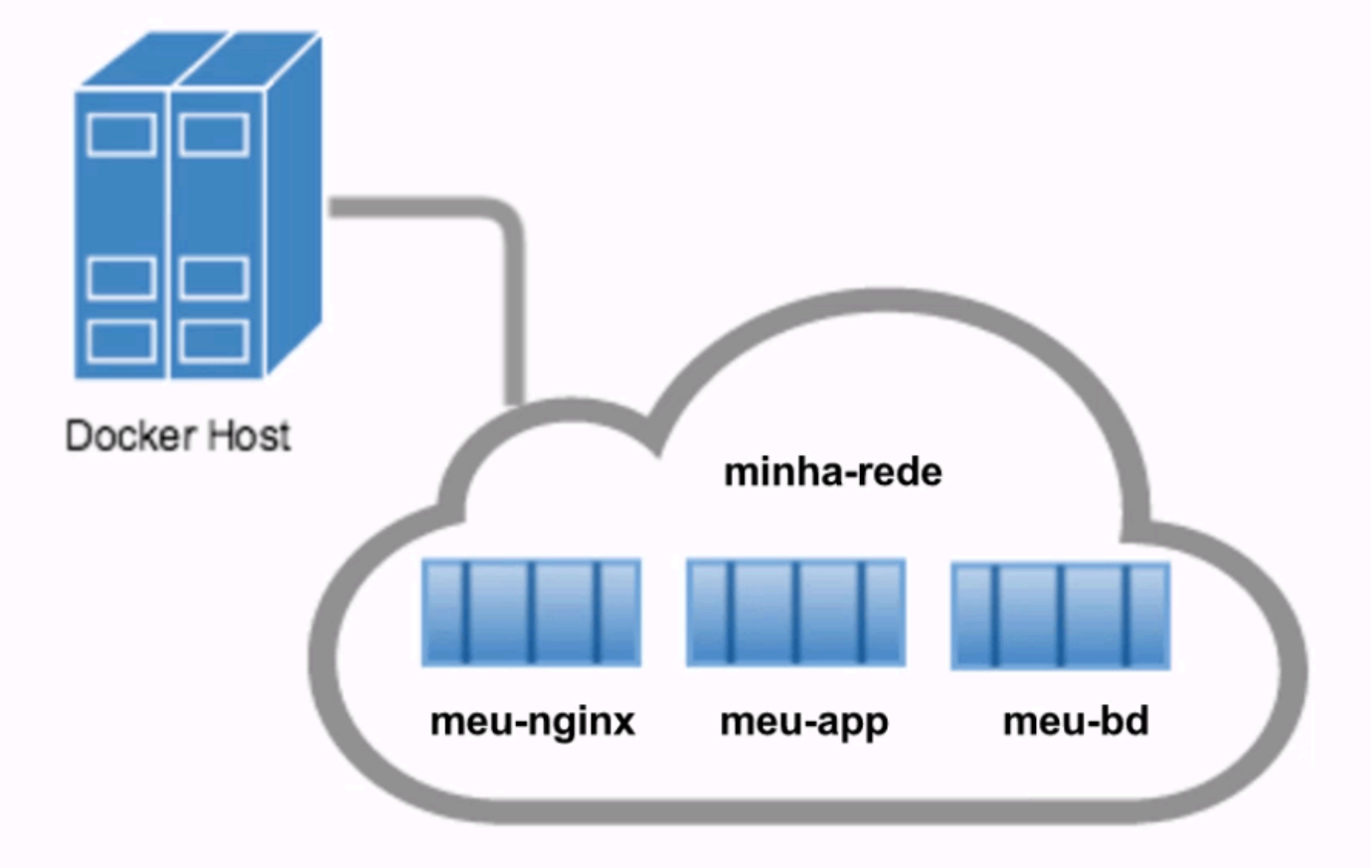
```
T1 $ docker run –it --name dmcu ubuntu

T1 $ hostname -i

T2 $ docker run –it ubuntu

T2 $ hostname –i
```

Redes com o Docker

```
T2 $ apt-get update ; apt-get install iputils-
ping
```

```
T2 $ ping <ip_T1>
```

Redes com o Docker



docker

Redes com o Docker

```
T1 $ docker network create –driver bridge
minha-rede

T1 $ docker run –it --name pmcu –network
minha-rede ubuntu

T3 $ docker run –it --name smcu –network
minha-rede ubuntu
```

Redes com o Docker

```
T3 $ apt-get update ; apt-get install iputils-
ping
```

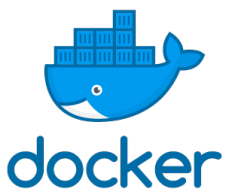
```
T3 $ ping dmcu #deu certo? ☹
```

```
T3 $ ping smcu #deu certo? ☺
```

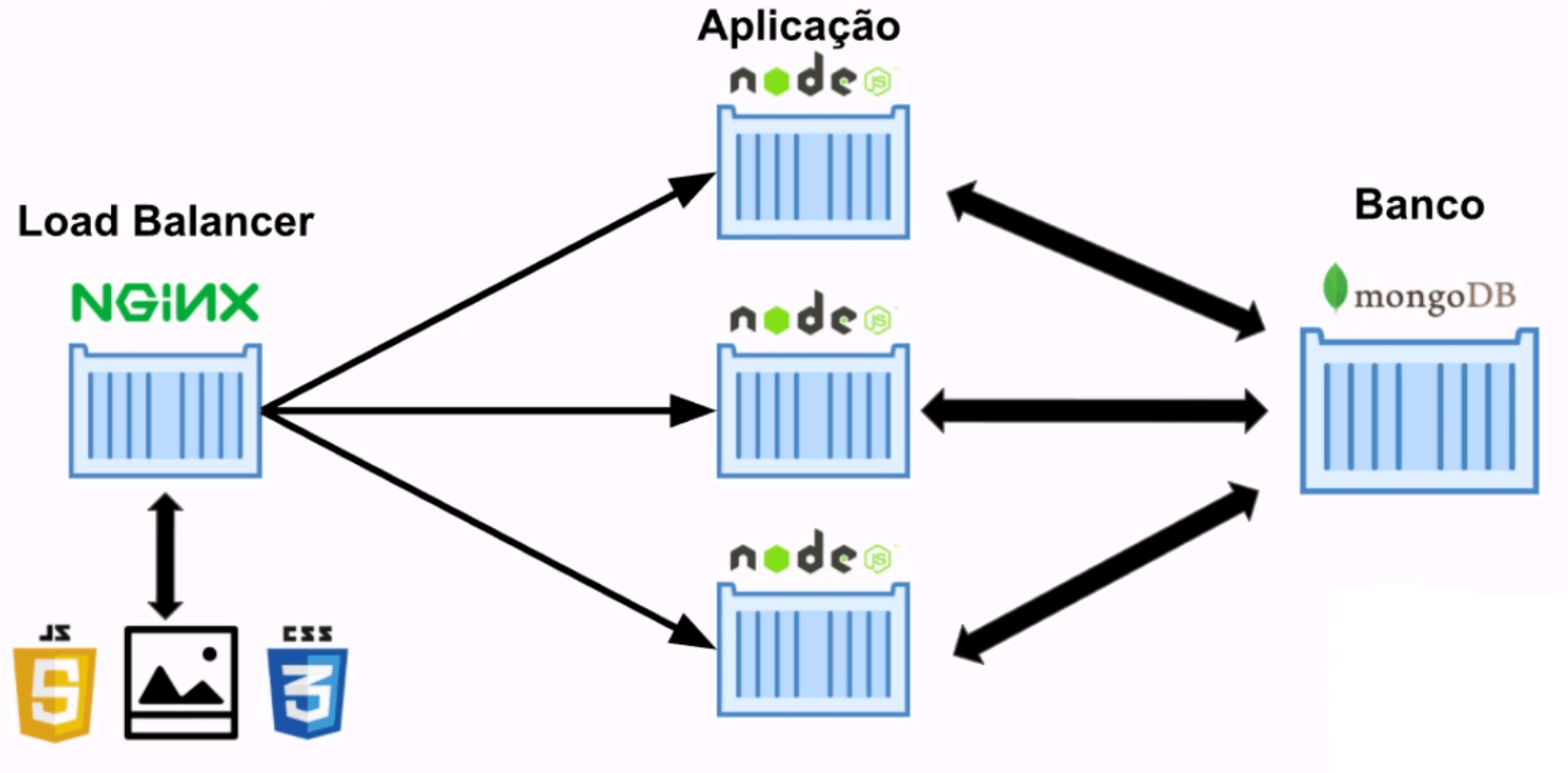
Pegando os dados de um banco em outro container

Criando nossa própria Imagem - Dockerfile

```
$ docker pull douglasq/alura-books:cap05

$ docker pull mongo

$ docker run –network minha-rede -d -p
8080:3000 douglasq/alura-books:cap05

$ docker exec -i -t <id> /bin/bash

$ docker run -d --name meu-mongo --network
minha-rede mongo
```

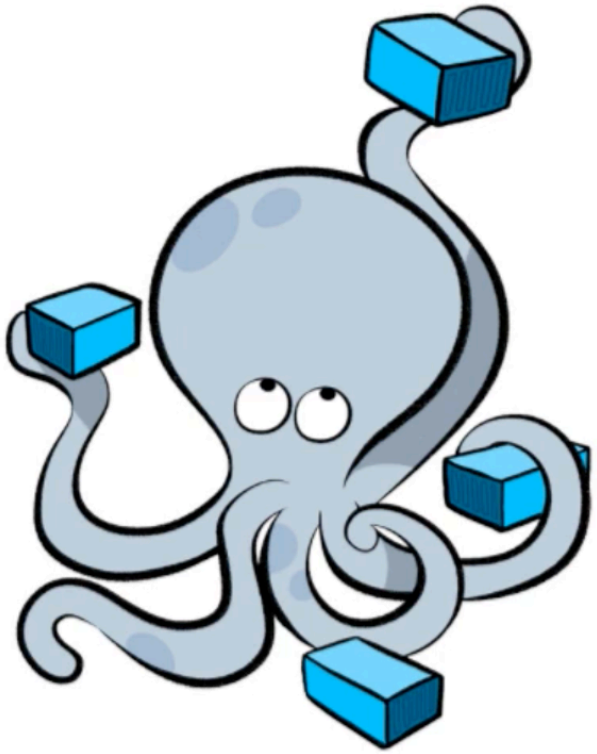


Docker Compose



docker

Subir 5 containers na mão? Não!



Docker Compose salva nossa vida!

docker-compose.**yml**

docker

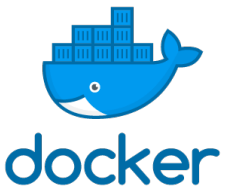
Docker Compose in Action!

Download Projeto:

- http://shorturl.at/btIRW
- `cd /tmp`
- `wget -O alura-docker-cap06.zip http://shorturl.at/btIRW`
- `unzip alura-docker-cap06.zip.zip`

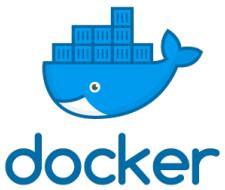
Download do arquivo Docker Compose

- http://shorturl.at/oISU9



Construindo e Executando o Projeto!

```
$ docker-compose build

$ docker-compose up –d

$ docker-compose ps      $

$ docker exec -it alura-books-1 ping alura-books-2

$ docker exec -it alura-books-1 ping node2
```



Acessando o projeto

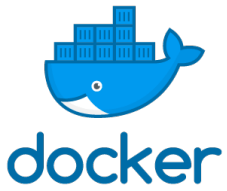
Projeto:

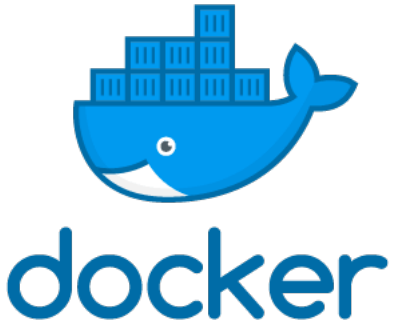
- http://localhost:80
- http://localhost:80/seed
- http://localhost:80

Construindo e Executando o Projeto!

```
$ docker-compose down
```

```
lucasnadalete@Lucass-MacBook-Pro ~/temp/alura-docker-cap06 docker-compose down
Stopping nginx                          ... done
Stopping alura-books-1                  ... done
Stopping alura-books-2                  ... done
Stopping alura-books-3                  ... done
Stopping alura-docker-cap06_mongodb_1 ... done
Removing nginx                          ... done
Removing alura-books-1                  ... done
Removing alura-books-2                  ... done
Removing alura-books-3                  ... done
Removing alura-docker-cap06_mongodb_1 ... done
Removing network alura-docker-cap06_production-network
```





Docker: Criando containers sem dor de cabeça

VI Cimatech

Prof. MSc. Lucas G. Nadalete

lucas.nadalete@fatec.sp.gov.br