Docker

Tiago Heinrich

UniSociesc Joinville

30/04/2020

Birth of Docker

- Solomon Hykes fundador da dotCloud
- Em uma conferência de desenvolvedores do Python
- California em March 15, 2013
- Encapsular o processo de criação de um artefato distribuível para qualquer aplicativo (escalável e otimizado)
- Desenvolvido em Go

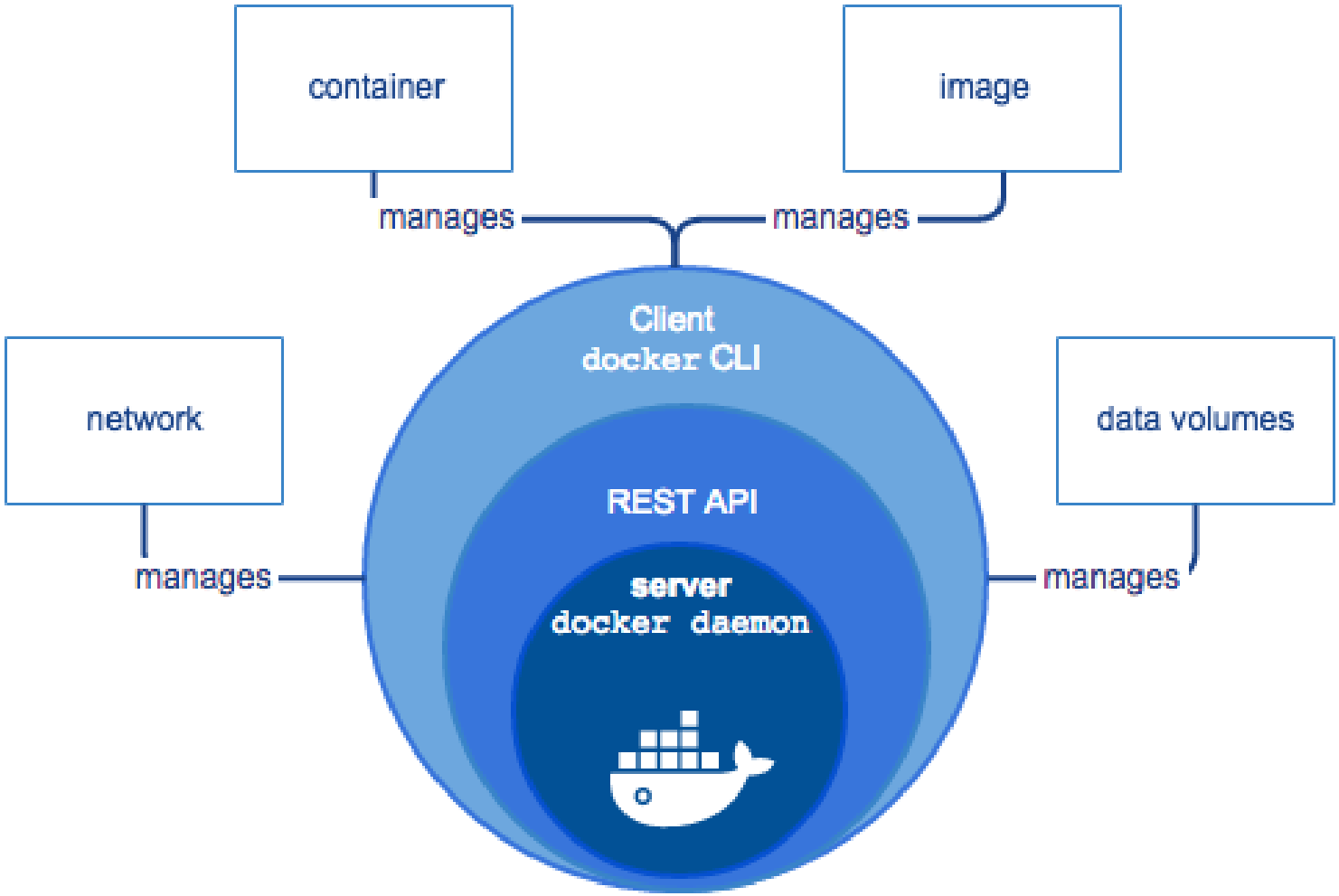
Introdução

- O Docker é uma plataforma aberta para desenvolvimento, envio e execução de aplicativos
- Permite que você separe seus aplicativos da sua infraestrutura
- Execute um aplicativo em um ambiente isolado chamado contêiner
- Os contêineres são leves porque não precisam da carga extra de um hipervisor

Aplicações

- Desenvolvimento e compartilhamento de código
- Ambiente de teste e rotinas automatizadas
- Para a remoção de bugs, basta enviar a imagem atualizada para o ambiente de produção
- Implantação e dimensionamento responsivos

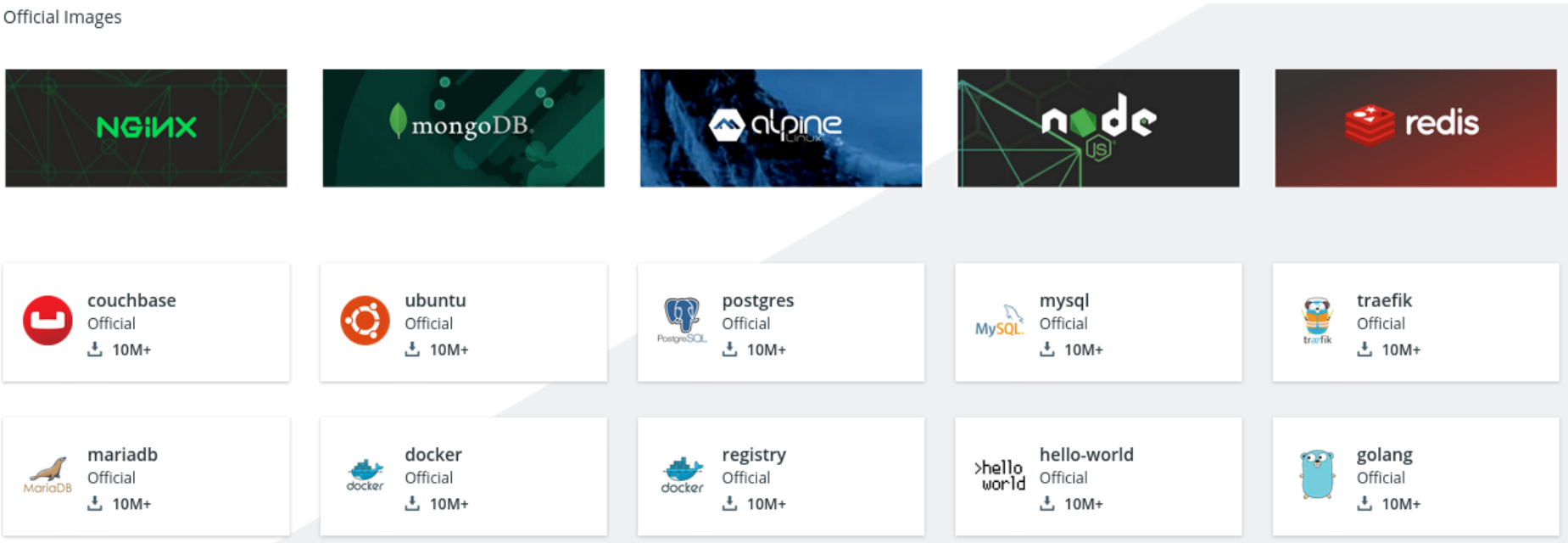
Docker Engine



Docker Engine

- Docker *registry* armazena imagens do Docker (Docker Hub)

```
docker pull ; docker run
docker push
```



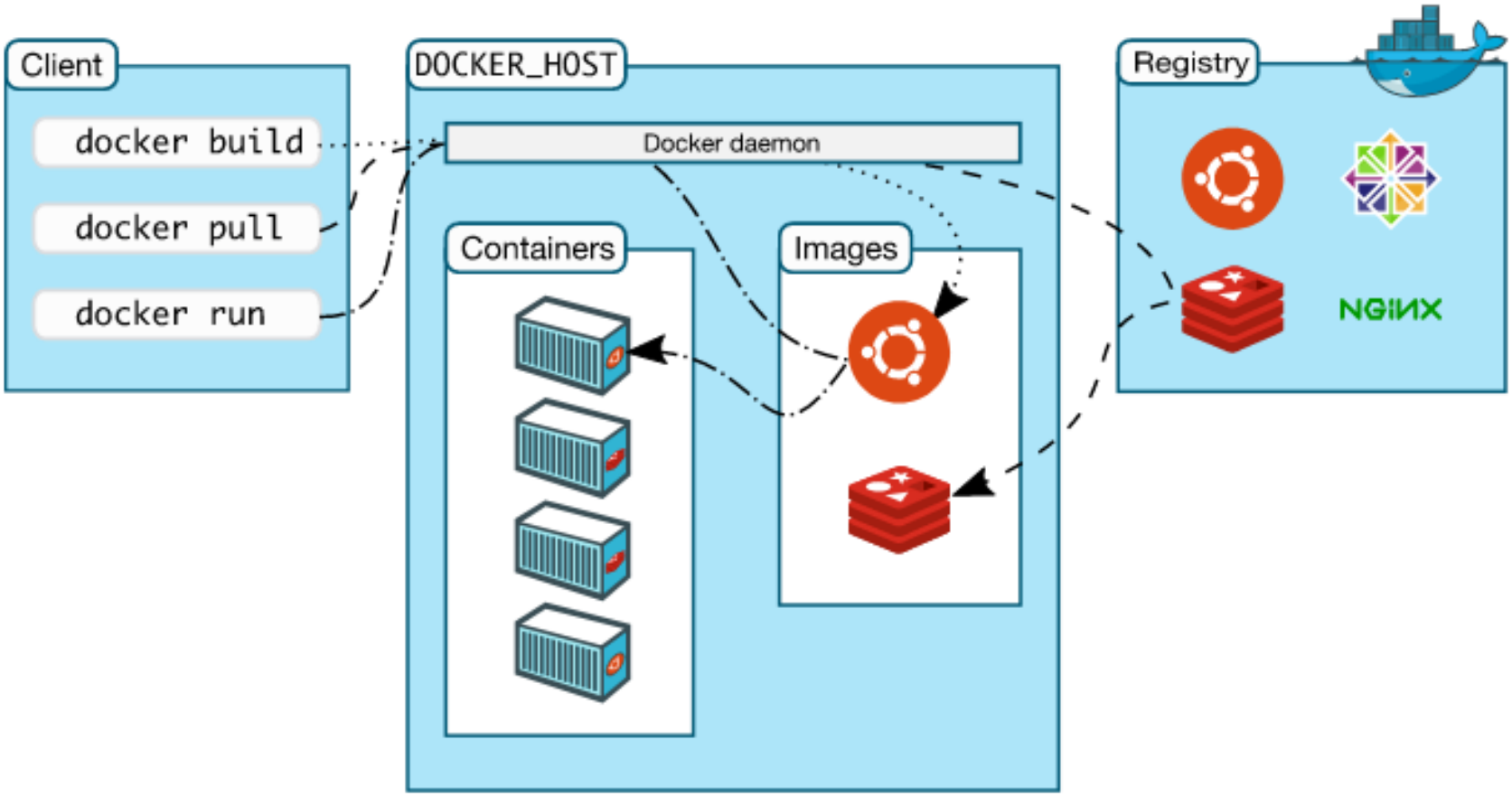
Docker objects

- **Images** é um modelo somente leitura com instruções para criar um contêiner do Docker

  `Por exemplo, você pode criar uma imagem baseada na imagem do ubuntu, mas instala o servidor da web Apache e seu aplicativo`
- **Containers** é uma instância executável de uma imagem. Você pode criar, iniciar, parar, mover ou excluir um contêiner
- **Services** permitem dimensionar contêineres em vários daemons do Docker, que funcionam juntos como um *swarm*

```
sudo docker run --rm -ti ubuntu:latest
    /bin/bash
```

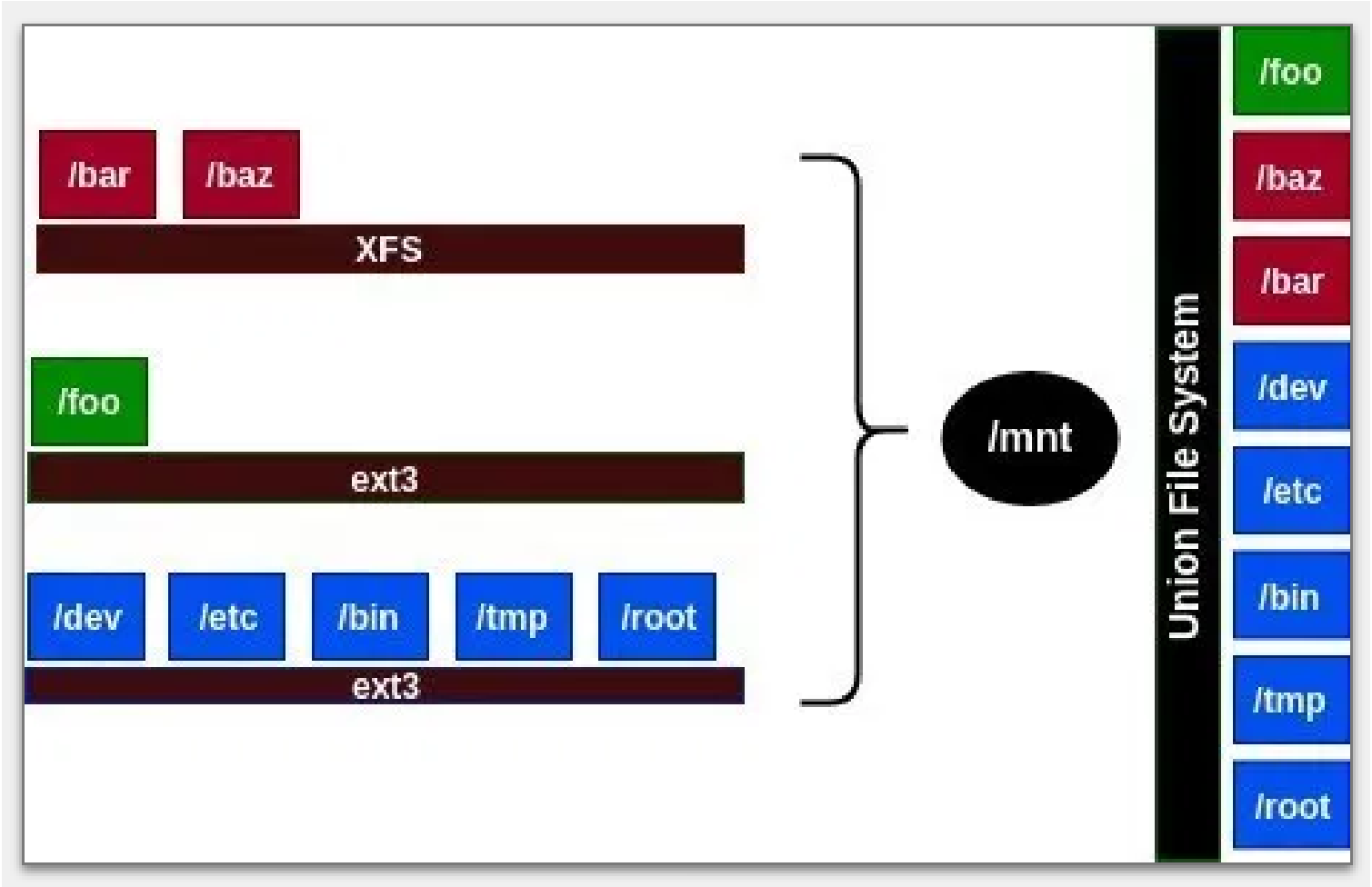
Docker architecture



Docker objects

- O acesso é limitado a esse espaço para o **Namespaces** (pid, net, ipc, mnt e uts)
- **Control groups** (cgroups) limita um aplicativo a um conjunto específico de recursos
- **Union file systems** (UnionFS)

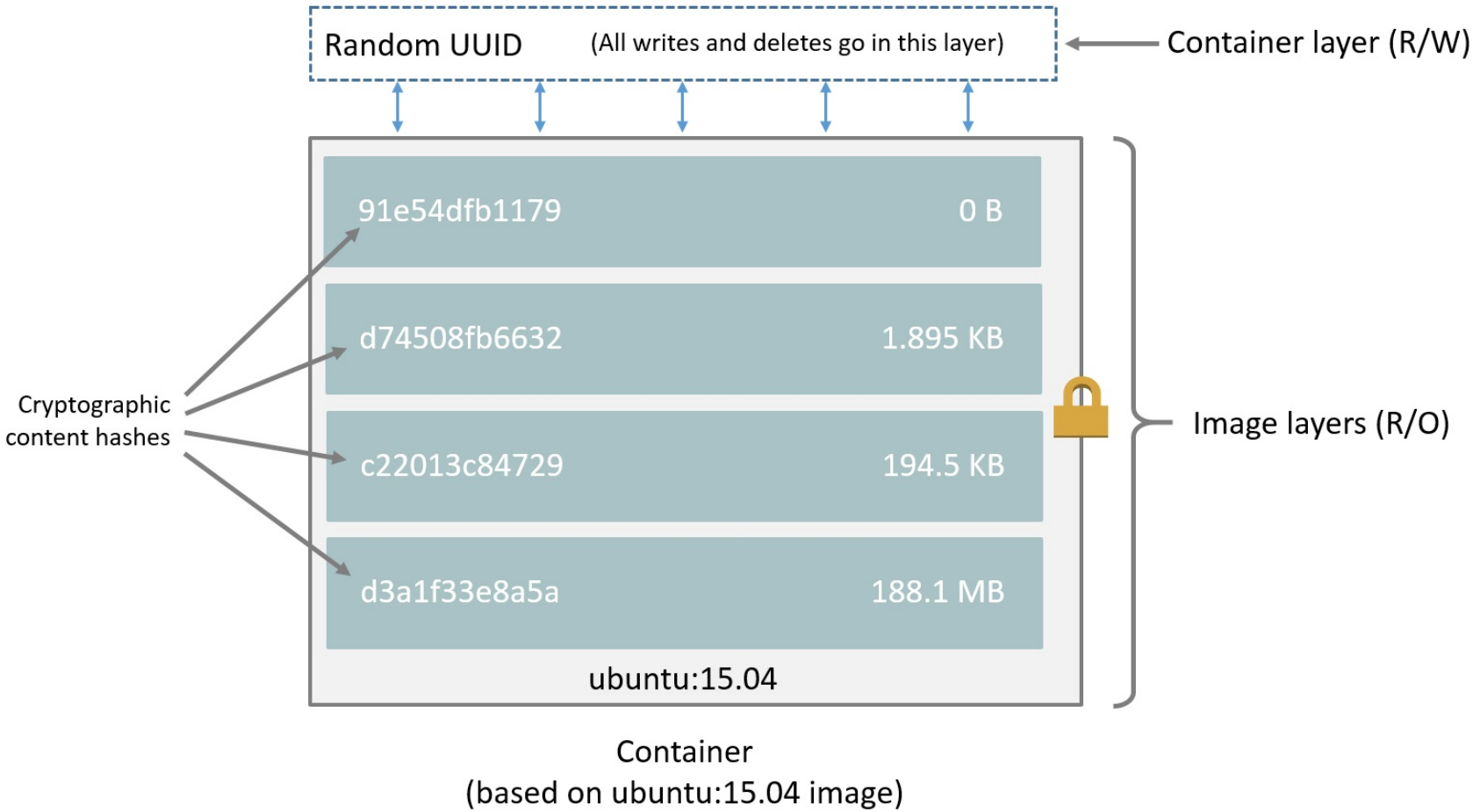
UnionFS


/bar
/baz
XFS
/foo
ext3
/dev
/etc
/bin
/tmp
/root
ext3
/mnt
Union File System
/foo
/baz
/bar
/dev
/etc
/bin
/tmp
/root

Propriedades

1. Logical merge of multiple layers
2. Read-only lower layers, writable upper layer
3. Start reading from the upper layer than defaults to lower layers
4. Copy on Write (CoW)
5. Simulate removal from lower directory through whiteout file

Layers Image



Docker Isn't

- Plataforma de virtualização corporativa (VMware, KVM, etc.)
- Plataforma em nuvem (Openstack, CloudStack, etc.)
- Estrutura de implantação (Capistrano, Fabric, etc.)
- Ferramenta de Gerenciamento de Carga de Trabalho (Mesos, Frota, etc.)
- Ambiente de desenvolvimento (Vagrant, etc.)

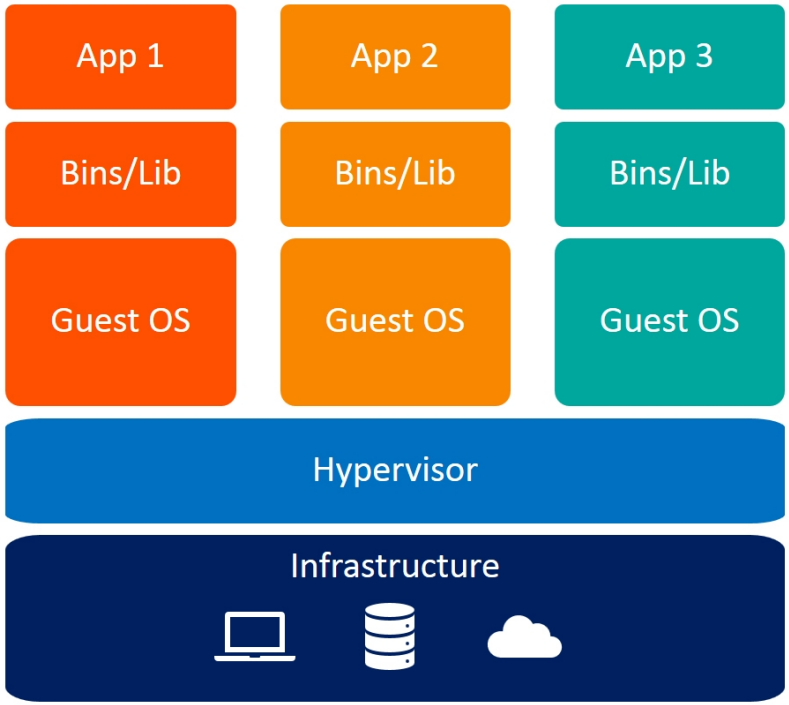
Docker overview

- Empacotar software de uma maneira que aproveite as habilidades que os desenvolvedores já possuem
- Agrupando software de aplicativo e sistemas de arquivos de SO necessários em um único formato de imagem padronizado
- Usando artefatos empacotados para testar e entregar exatamente o mesmo artefato para todos os sistemas em todos os ambientes
- Abstraindo aplicativos de software do hardware sem sacrificar recursos

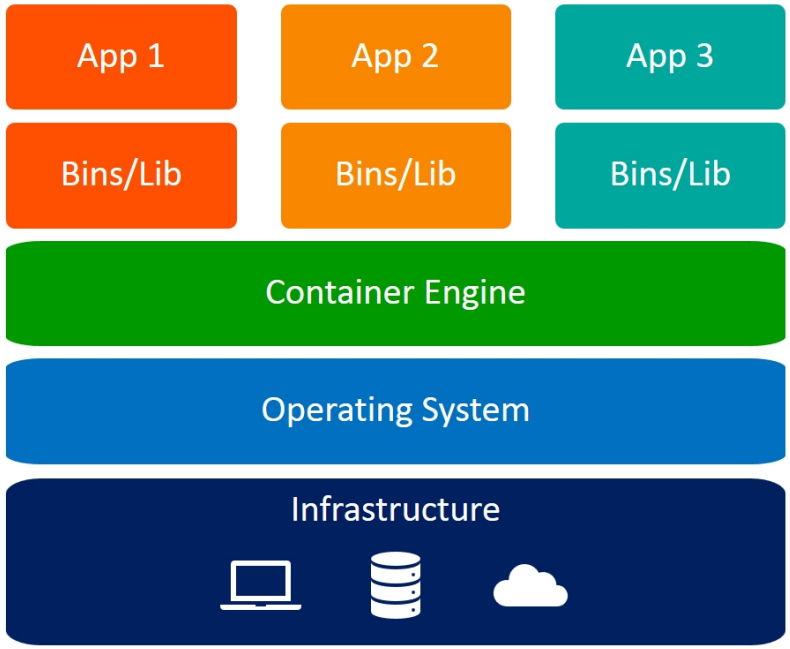
Limited Isolation

- Recipientes são “isolados” um do outro
- Possibilita colocar limites em seus recursos (CPU e memória), mas estes recursos serão executados ao lado de aplicações no *Host* físico
- Contêineres podem competir por recursos, da mesma forma que você esperaria de processos Unix
- Os limites de CPU e uso de memória são possíveis através do Docker, MAS não são similares ao que seria de uma máquina virtual

Docker vs VM



Virtual Machines



Containers

Docker Prática

Install

```
sudo apt-get install apt-transport-https
    ca-certificates curl gnupg-agent
    software-properties-common

curl -fsSL https://download.docker.com/
    linux/ubuntu/gpg | sudo apt-key add -

sudo apt-key fingerprint 0EBFCD88

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

Basic Commands

- Criar um container (docker create)
- Inspecionar Imagens (docker images)

```
docker create -t -i fedora bash
docker start -a -i ID....
(yum dnf...)
```

Basic Build

- Inspecionar Imagens
- Inspecionar Containers

```
docker images
docker create --name nginxmy -p 80:80
    nginx:alpine
docker ps -a
docker start ID
lynx IP
```

Web Page

```
touch index.html
docker cp index.html mmm:/usr/share/nginx
   /html/index.html

docker exec -it ID ls /usr/share/
```

NTP Server

```
sudo systemctl status ntpd/ntp
sudo systemctl stop ntpd/ntp

docker pull cturra/ntp

docker run --name=ntp                    \
           --restart=always              \
           --detach=true                 \
           --publish=123:123/udp         \
           --cap-add=SYS_TIME            \
           cturra/ntp

docker exec ntp chronyc tracking
```

NTP Server

```
ip a ;  ntpdate -q IP_DOCKER

docker restart ID
docker exec ntp chronyc sources
docker exec ntp chronyc sourcestats
```

Iperf3

```
docker run networkstatic/iperf3

docker run -it --rm --name=iperf3-server -
    p 5201:5201 networkstatic/iperf3 -s

docker run -it --rm networkstatic/iperf3 -
    c 172.17.0.1
```