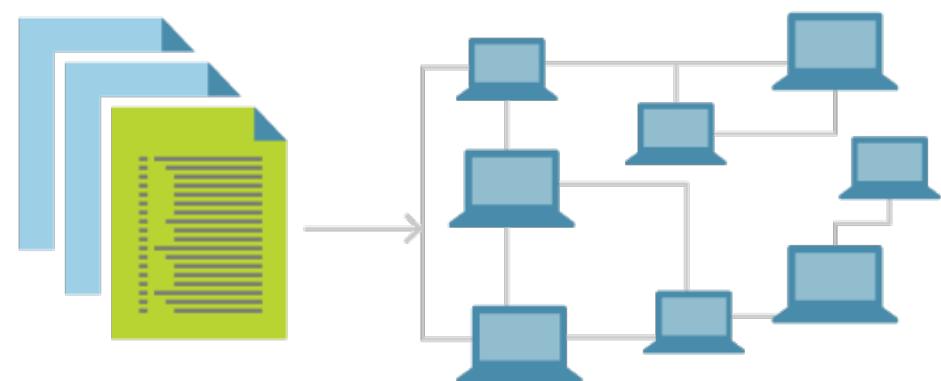


Isolando o seu ambiente

Prof. Matheus Sousa Faria

Isolando o seu ambiente

- Reduzir acoplamento entre o ambiente pessoal e o ambiente de trabalho
- Ambiente igual para toda a equipe
 - Evita “Funciona no meu computador”
- IaC - Infrastructure as Code
 - Maior manutenibilidade
 - Maior controle para os desenvolvedores
 - Início com a computação em nuvem
 - IaaS - Infrastructure as Service
- Descrição e documentação do ambiente

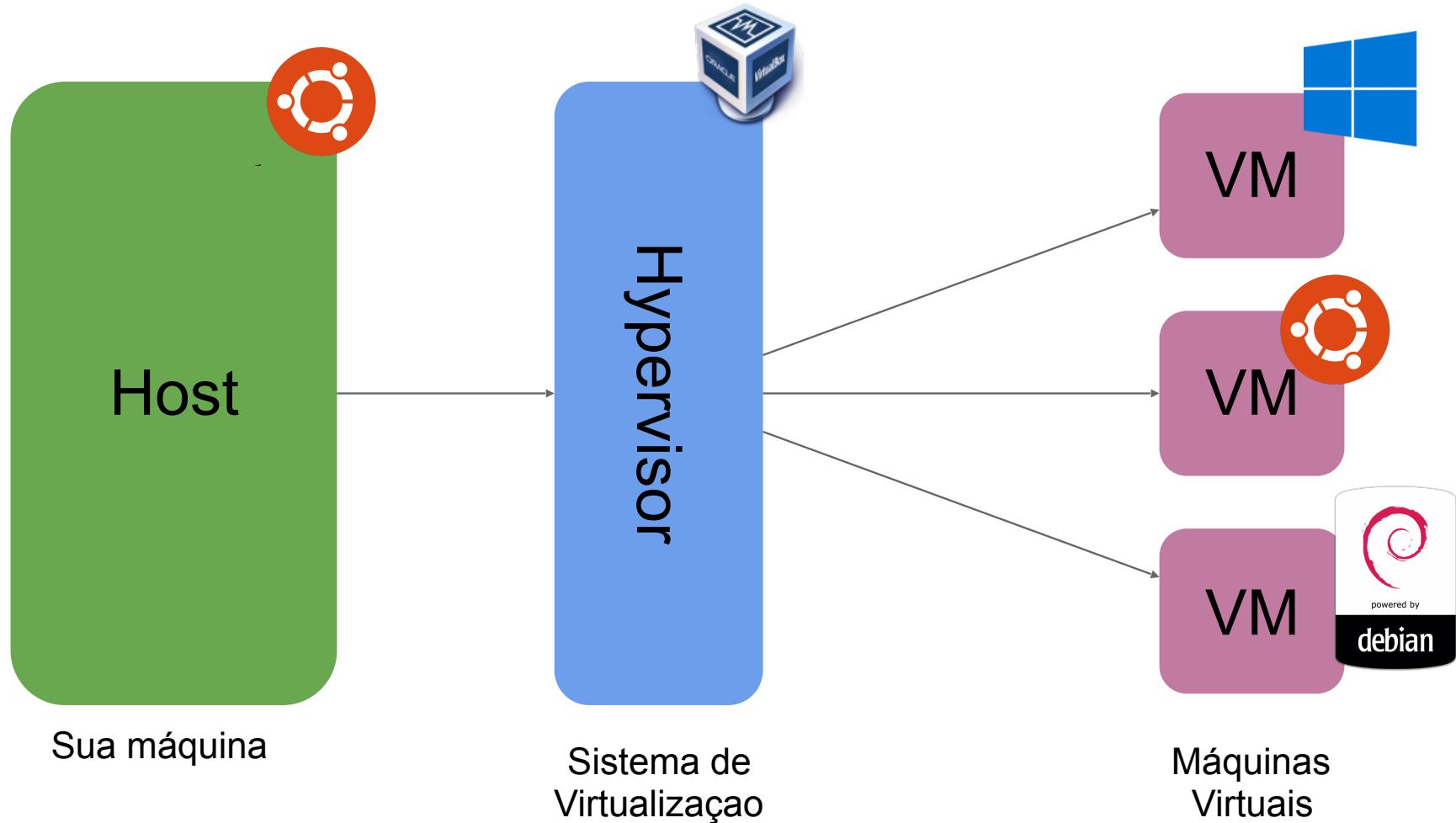


Virtualização

O que é?

- Abstração do HW a nível de SW
- Cria Computadores Virtuais
 - Virtual Machines (VMs)
 - Guest Machine
- Tipos:
 - Full Virtualization
 - Paravirtualization
- Hypervisor: Programa que faz virtualização





Vantagens

1. Criação de ambientes facilitada
2. Ambiente 100% isolado
3. Snapshots e backups de forma fácil

Desvantagens

1. Perda de performance
 2. Ambiente pesado para se manter
 3. Consome muitos recursos
-

Ferramentas



VirtualBox



|| Parallels®

E o vagrant?



- **Não** é um virtualizador/hypervisor
- Ferramenta para gerência de vms
 - Criar
 - Remover
 - Acessar
 - **Configurar**
- Boa ferramenta para trabalhar com máquinas virtuais
- Abstrai configurações de específicas de cada hypervisor

Comandos Básicos

- | | |
|-------------------|------------------------|
| vagrant init | - Criar um Vagrantfile |
| vagrant up | - Iniciar/Criar uma vm |
| vagrant halt | - Desligar uma vm |
| vagrant destroy | - Deletar uma vm |
| vagrant provision | - Configurar uma vm |
| vagrant ssh | - Acessar uma vm |

Vagrantfile

```
Vagrant.configure("2") do |config|
  ## https://atlas.hashicorp.com/boxes/search

  config.vm.box = "ubuntu/trusty64"

end
```

Outras boxes

Nome da box
escolhida

Box padrão do vagrant

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box_url = "https://
mydomain.com"
  config.vm.box = "xpto"
end
```

Box de uma origem customizada

Vagrantfile

Encaminhamento de portas

```
config.vm.network "forwarded_port", guest: 80, host: 8080
```

IP de acesso a VM

```
config.vm.network "private_network", ip: "192.168.33.10"
```

Sincronização de pasta

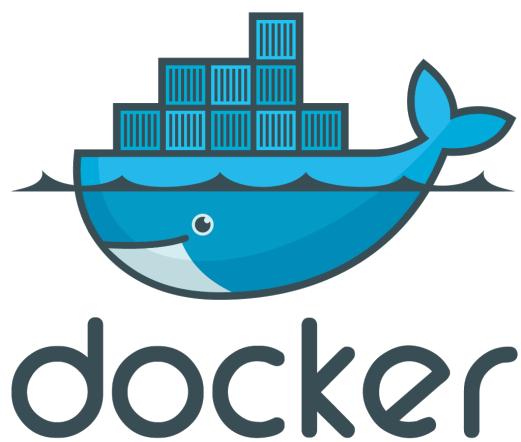
```
config.vm.synced_folder "../data", "/vagrant_data"
```

Vagrantfile

provider == hypervisor

```
config.vm.provider "virtualbox" do |vb|
  vb.name = "gcs"
  vb.gui = false
  vb.memory = "1024"
  vb.cpus = 2
end
```

Outros Providers



Microsoft
Hyper-V



Vagrantfile

Comandos executados
pelo usuário root

```
config.vm.provision "shell", inline: <<-SHELL
  apt-get update
  apt-get install -y vim git
SHELL
```

Automação da
configuração ambiente

Outros Provisioners



ANSIBLE



SALTSTACK



CHEF

Multiplas VMs

Provision Global

```
vagrant.configure("2") do |config|  
  config.vm.provision "shell", inline: "echo Hello"
```

Config da vm web

```
  config.vm.define "web" do |web|  
    web.vm.box = "apache"  
  end
```

Config da vm db

```
  config.vm.define "db" do |db|  
    db.vm.box = "mysql"  
  end  
end
```

Fluxo de Trabalho

1. vagrant init
 2. Editar vagrant file
 3. vagrant up
 4. vagrant ssh
 5. Depois que sair da vm
 6. vagrant halt
-

Features adicionais do Vagrant

Verificar todas as VMs

```
vagrant global-status
```

Retorna o id, estado e localização de
cada VM utilizada no seu host

Salvando estado atual

`vagrant snapshot`

Salva o estado atual da VM

Criar uma pilha de snapshot

Similar ao “git stash”

Permite restauração de um ponto específico

Criando sua box

```
vagrant package --base my-  
box
```

Cria um box a partir do estado atual da sua vm

Gerenciando as boxes

vagrant box list
vagrant box add
vagrant box
update

Permite a adição e atualização de boxes

Customizadas ou padrões

Evita tempo de download

Validando seu Vagrantfile

```
vagrant validate
```

Checa se a syntax do seu Vagrantfile está correta

Evita execuções desnecessárias

Sincronização de Pastas

```
config.vm.synced_folder "../data", "/vagrant_data"
```

```
vagrant rsync
```

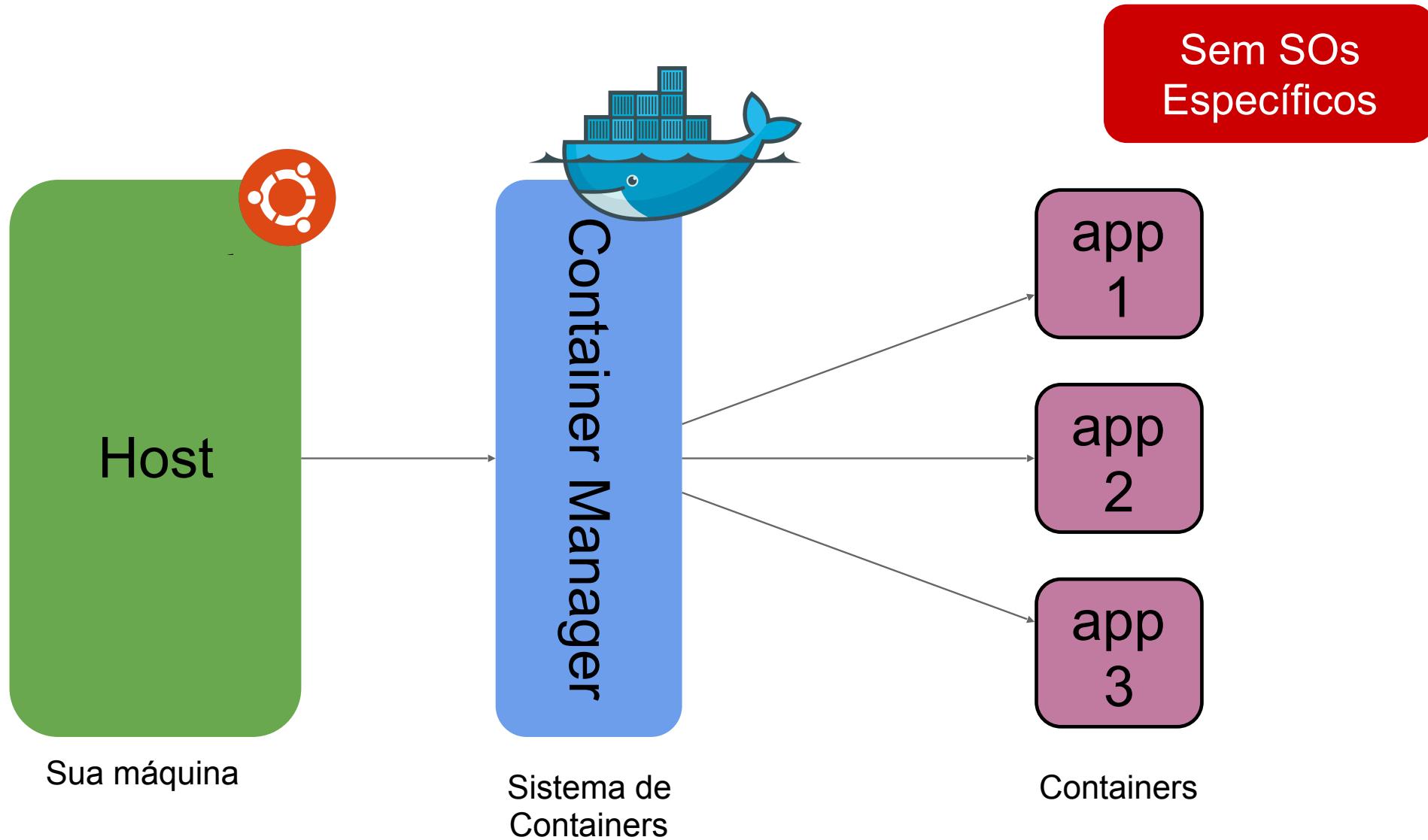
Força a ressincronização das pastas marcadas para sincronização

Containers

O que são?

- Carregam tudo que é necessário para o funcionamento da sua app
- Instancias isoladas de um sistema
- Containers
 - Jails
 - Virtualization Engines
- Criam ambientes mínimos e isolados para a sua aplicacão





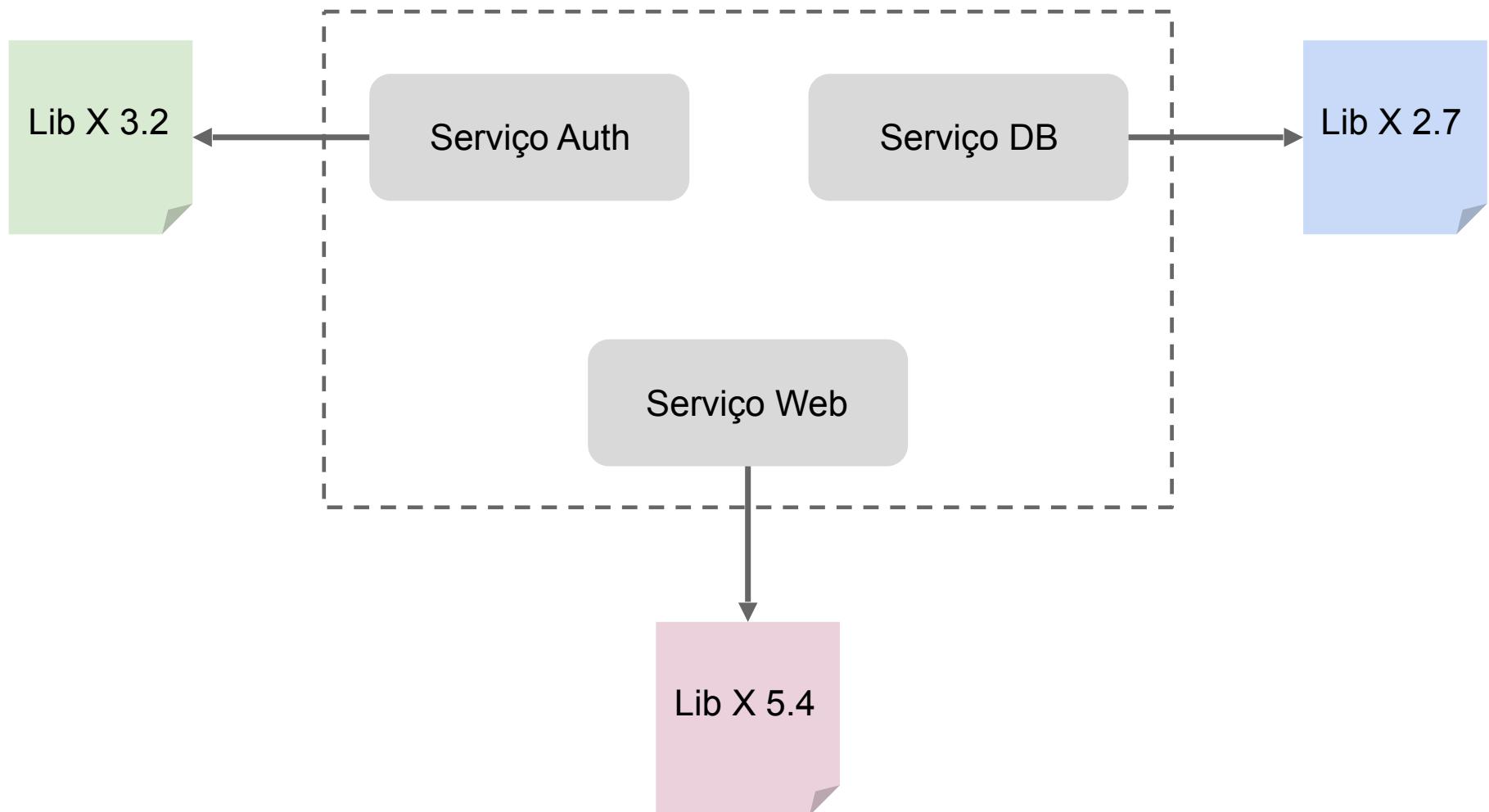
Isola o ambiente?
Mas as vms não
fazem isso já?



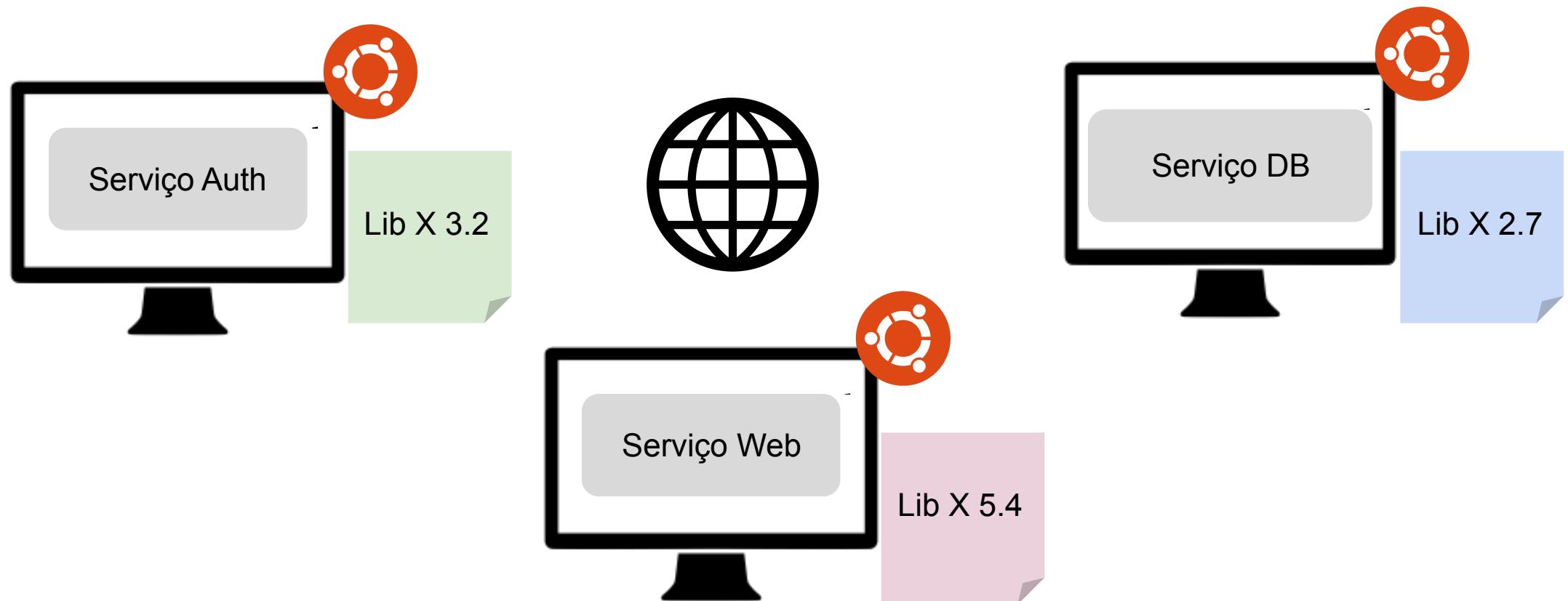
Máquina completa
com o sistema

Aplicação e suas
dependências

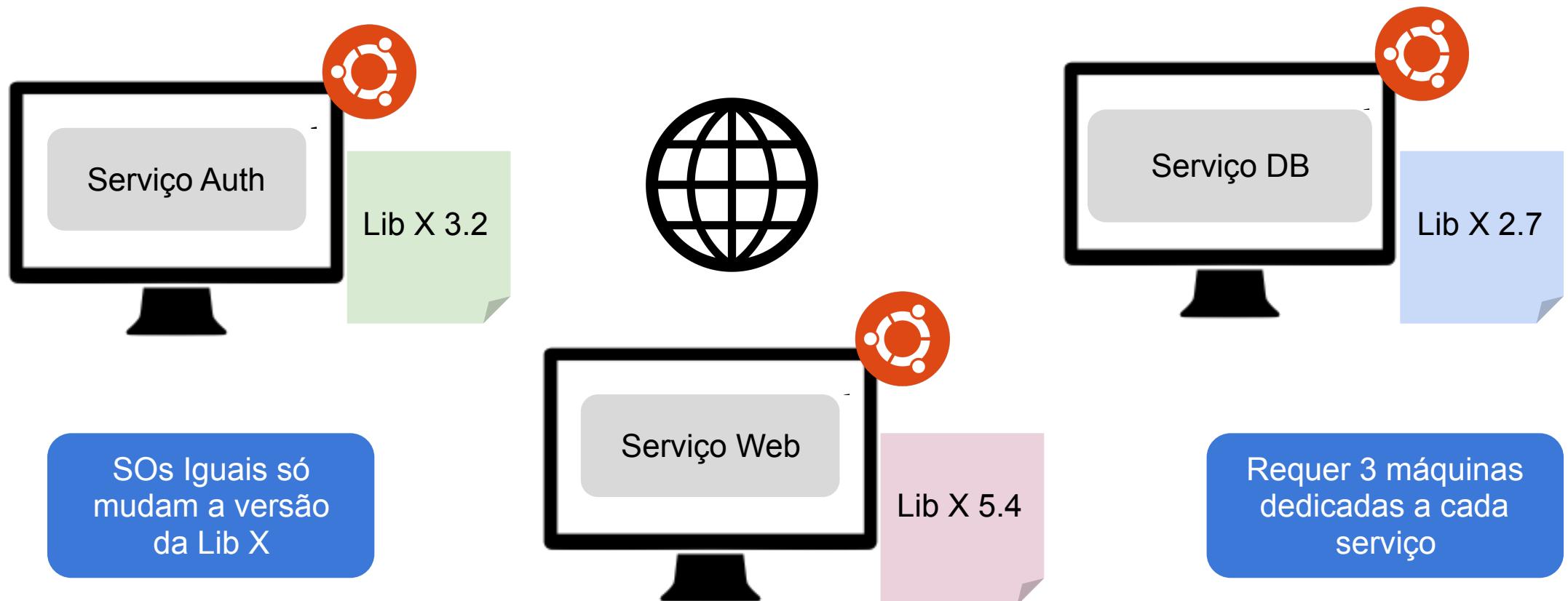
Aplicação
empacotada em
um executável



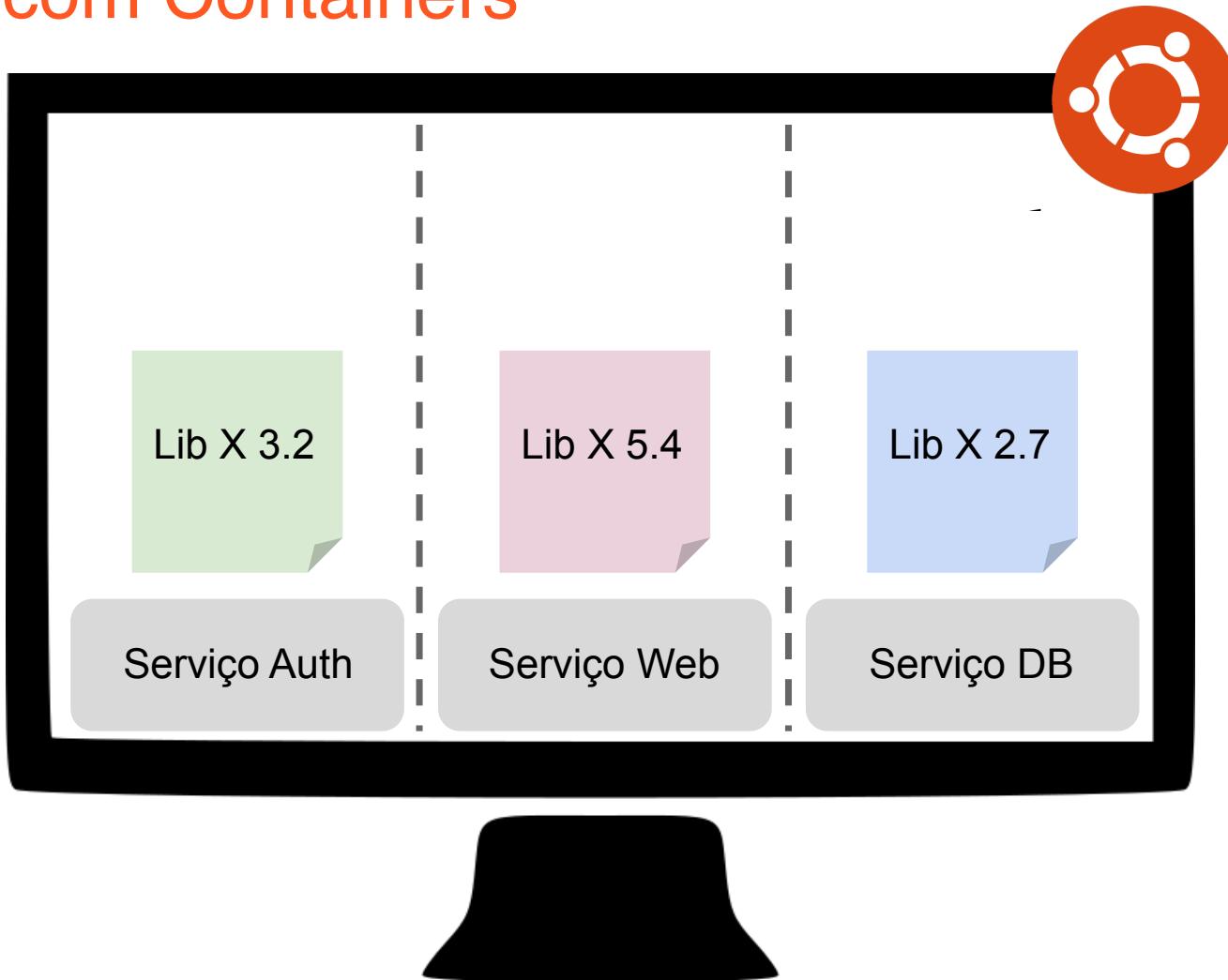
Solução com VMs



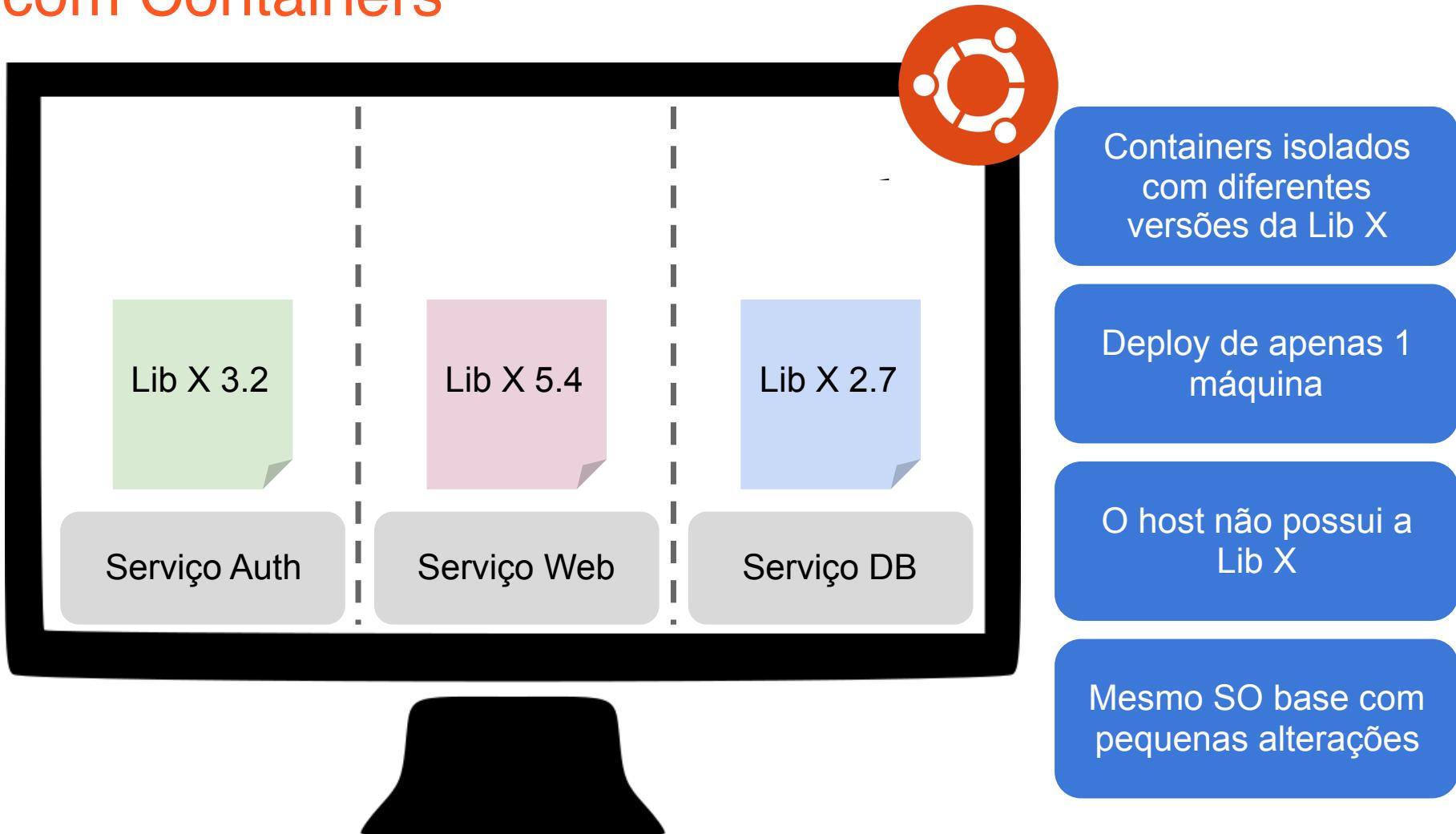
Solução com VMs



Solução com Containers



Solução com Containers



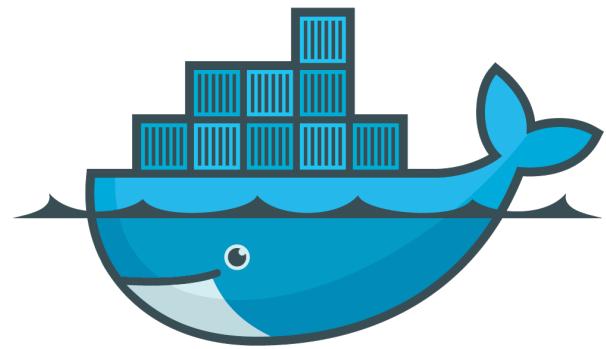
Vantagens

1. Leves em comparação com VMs
2. Deploy e Execução Rápidas
3. Deploy facilitado
4. Carregam só o necessário
5. Fácil de escalar

Desvantagens

1. Menos seguros que máquinas virtuais
 2. Dependem muito do Host
-

Ferramentas



docker



Docker

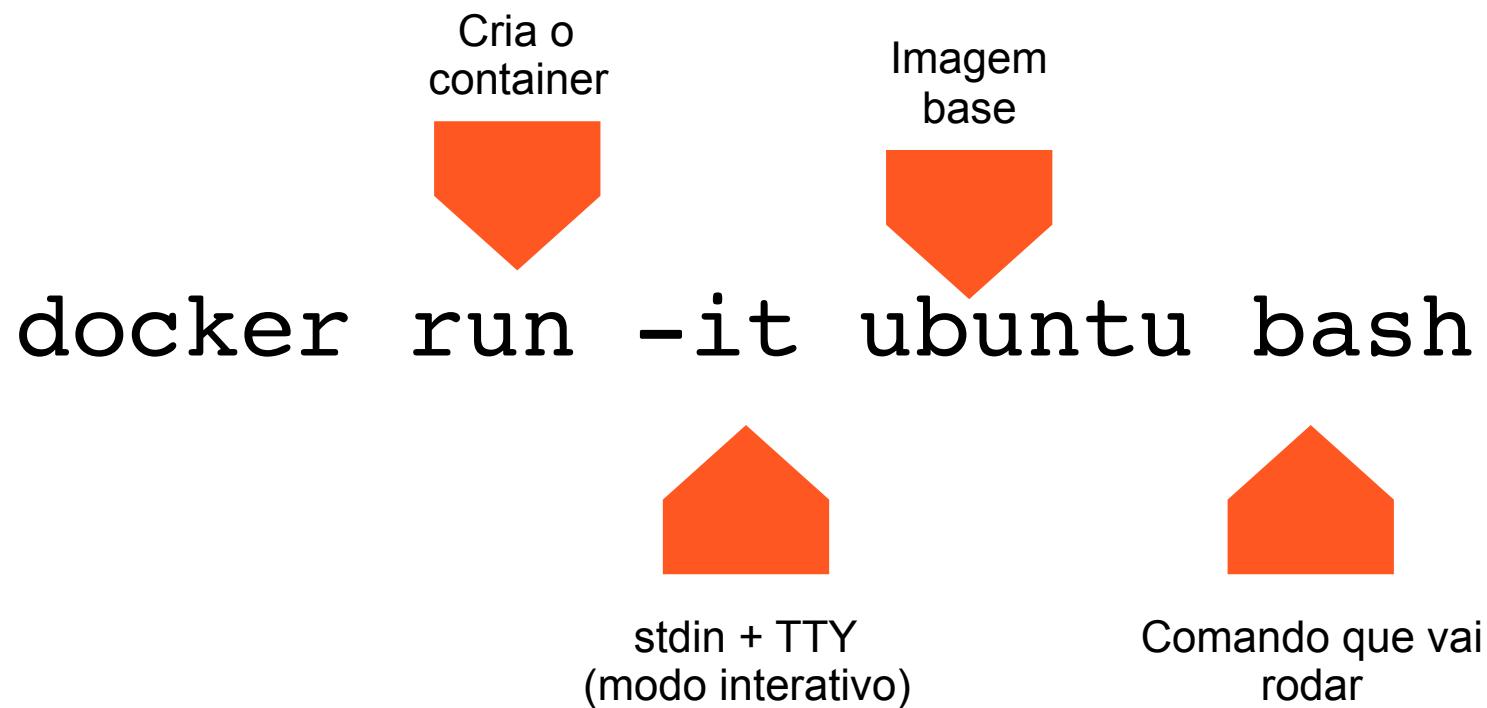


- Gerenciador de Containers
 - Criar, Remove, Instancia
- Images: Aglomerado executável com tudo que o seu SW precisa
 - Binários
 - SO mínimos e “incompletos”
 - Ponto de partida para a criação
- Containers: Instancias das imagens
 - Compartilha o kernel com o host
- Orientado a processos
- Trata o container como um estado
 - GIT-like (parecido com o git)



- Trata o container como um estado
 - GIT-like (parecido com o git)
- Fork do sistema de arquivos
 - Não o seu atual
 - Sistema da imagem base
- Grava as diferenças das imagens
- Push/Pull Docker Hub

Criando container (imagem pronta)



Listando os containers

```
docker ps -a
```



Inclui containers
que estão
desligados

Excluindo Containers

ID do
container



```
docker rm 001aac04ad8f
```

```
docker rm naughty_panini
```



Nome do
container

Criando container (imagem pronta)

```
docker run -it --name my_ubuntu ubuntu bash
```



Especifica um
nome para o
container

Diff do container

```
docker diff my_ubuntu
```



Nome do
container

Criando imagens (containers alterados)

```
docker commit my_ubuntu matheusfaria/my_ubuntu
```



Nome do
container OU ID



Nome da imagem
a ser criada
Boa prática:
username/nome

Listando e removendo imagens

`docker images`

`docker rmi ubuntu`



Nome da imagem

Interagindo com o Docker Hub

```
docker login
```

```
docker push matheusfaria/  
my_ubuntu
```

```
docker pull matheusfaria/  
my_ubuntu
```

Salva os diffs das imagens,
tornando o processo mais
otimizado

Ligando e Desligando containers

```
docker stop my_ubuntu  
docker start my_ubuntu
```



Nome do
container

Encaminhamento de portas

```
docker run -it --name my_ubuntu -p 8003:8000 ubuntu bash
```



8000 do container
para
8003 do host

Sincronização de pastas (volumes)

```
docker run -it --name my_ubuntu -p 8003:8000 -v /tmp/www:/code ubuntu bash
```



/tmp/www do host
para
/code do container

Dockerfile

Configuração de uma
imagem em um arquivo

```
FROM python:2           # Imagem base

WORKDIR /app            # Muda o diretório de Trabalho
ADD . /app              # Copiando tudo do diretório atual para o app

RUN pip install -r requirements.txt # Rodando um comando

EXPOSE 80               # Expondo port 80 para fora do container

ENV NAME World          # Criando variavel de ambiente
CMD [ "python", "app.py" ] # Comando de execução
```

Criando imagens (dockerfile)

```
docker build -t my_ubuntu_img ~/Desktop/
```



Nome da imagem
a ser criada



Caminho pro
Dockerfile



- Facilita / Orquestra o uso de múltiplos containers
- Replica as flags da linha de comando
- Criar uma rede pré configurada entre os containers
- docker-compose.yml
 - Syntax YAML



```
version: '3'      # Versão mais atual

services:          # Descrição dos containers

  web:             # Container web
    build: .
    ports:          # Encaminhamento de portas
    - "5000:5000"
    volumes:        # Sincronização de pastas
    - .:/code
    depends_on:    # Ordem de geração
    - redis

  redis:           # Container redis
    image: "redis:alpine" # Imagem pronta do docker
```

hub



- Sub Rede de containers

O container **web** vai conhecer o redis como **redis**

Devem utilizar as portas expostas

Perguntas?