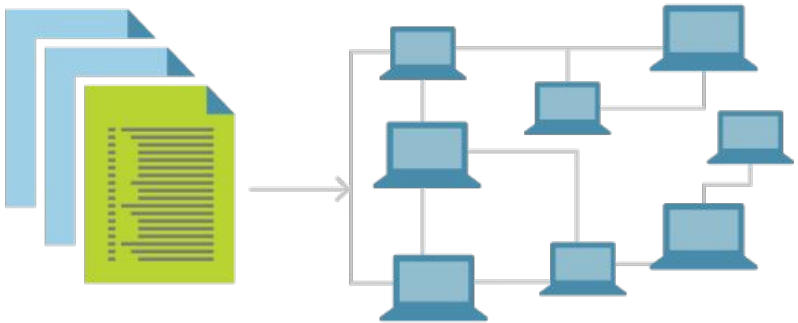
Isolando o seu ambiente

Isolando o seu ambiente

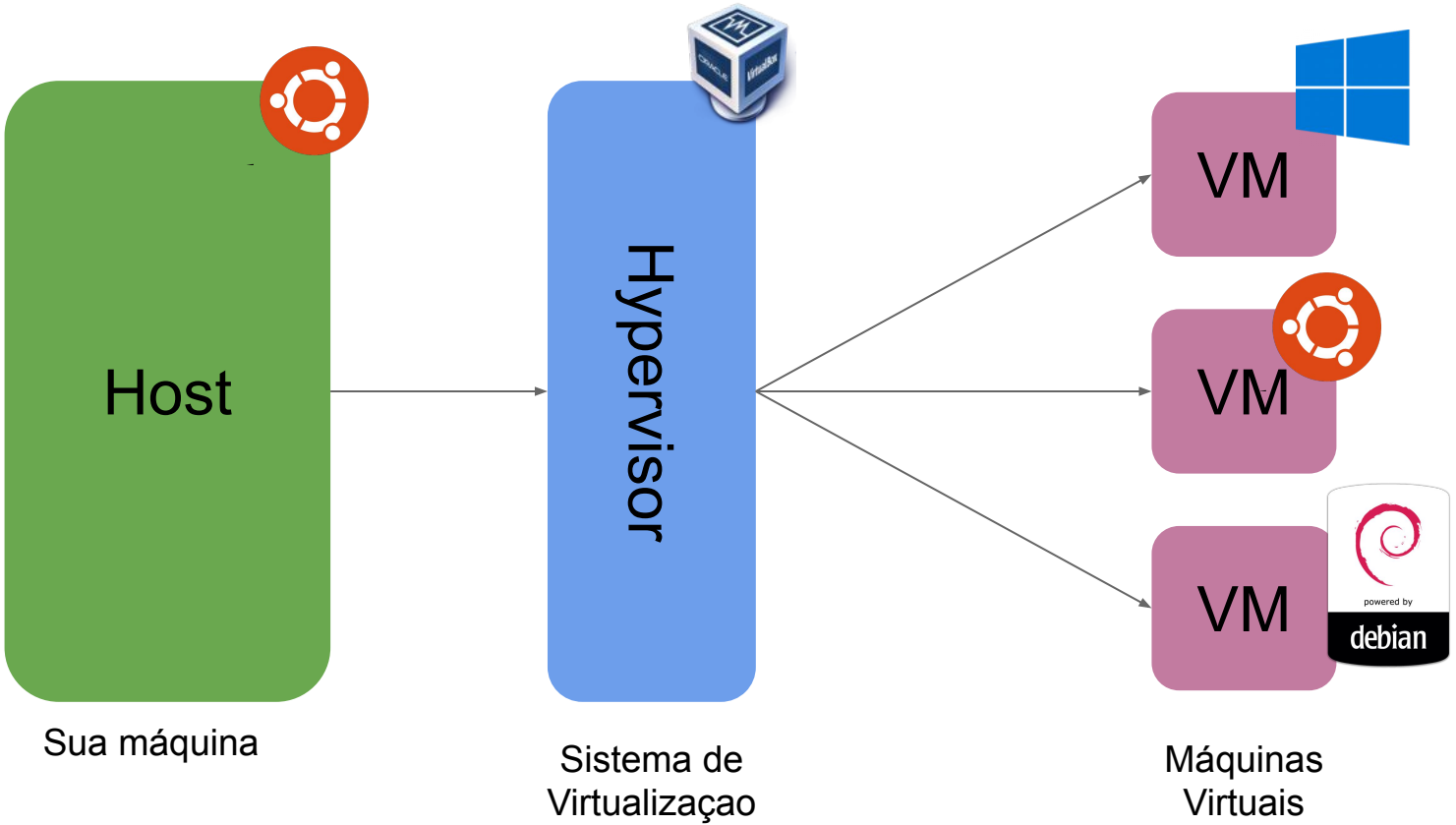
- Reduzir acoplamento entre o ambiente pessoal e o ambiente de trabalho
- Ambiente igual para toda a equipe
  - Evita “Funciona no meu computador”
- IaC - Infrastructure as Code
  - Maior manutenibilidade
  - Maior controle para os desenvolvedores
  - Início com a computação em nuvem
    - IaaS - Infrastructure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


Host
Hypervisor
VM
VM
VM
powered by
debian
Sua máquina
Sistema de
Virtualizaçao
Máquinas
Virtuais

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

E o vagrant?

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

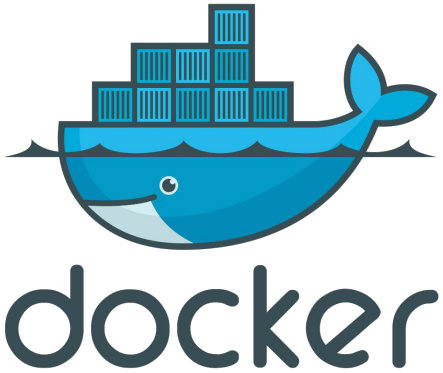
```
Vagrant.configure("2") do |config|
  ## https://atlas.hashicorp.com/boxes/search

  config.vm.box = "ubuntu/trusty64"

end
```

Outras boxes

Nome da box escolhida

Box padrão do vagrant

Outros Providers



Outros Provisioners

CHEF

ANSIBLE

SALTSTACK

Multiplas VMs

Provision Global

Config da vm web

Config da vm db

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

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

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

Fluxo de Trabalho

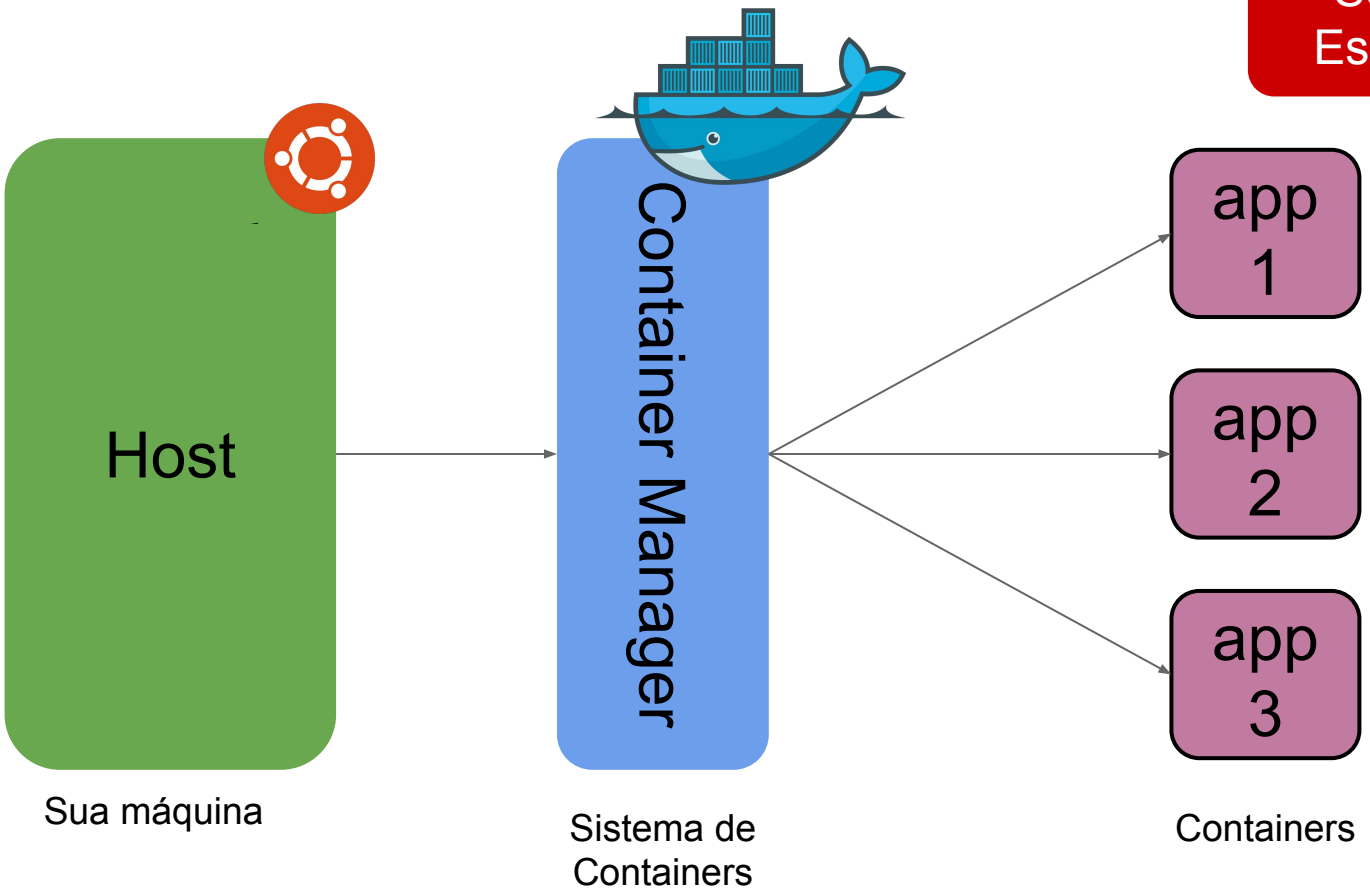
1. vagrant init
2. Editar vagrant file
3. vagrant up
4. vagrant ssh
5. Depois que sair da vm
6. vagrant halt

Features adicionais do Vagrant

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 aplicação

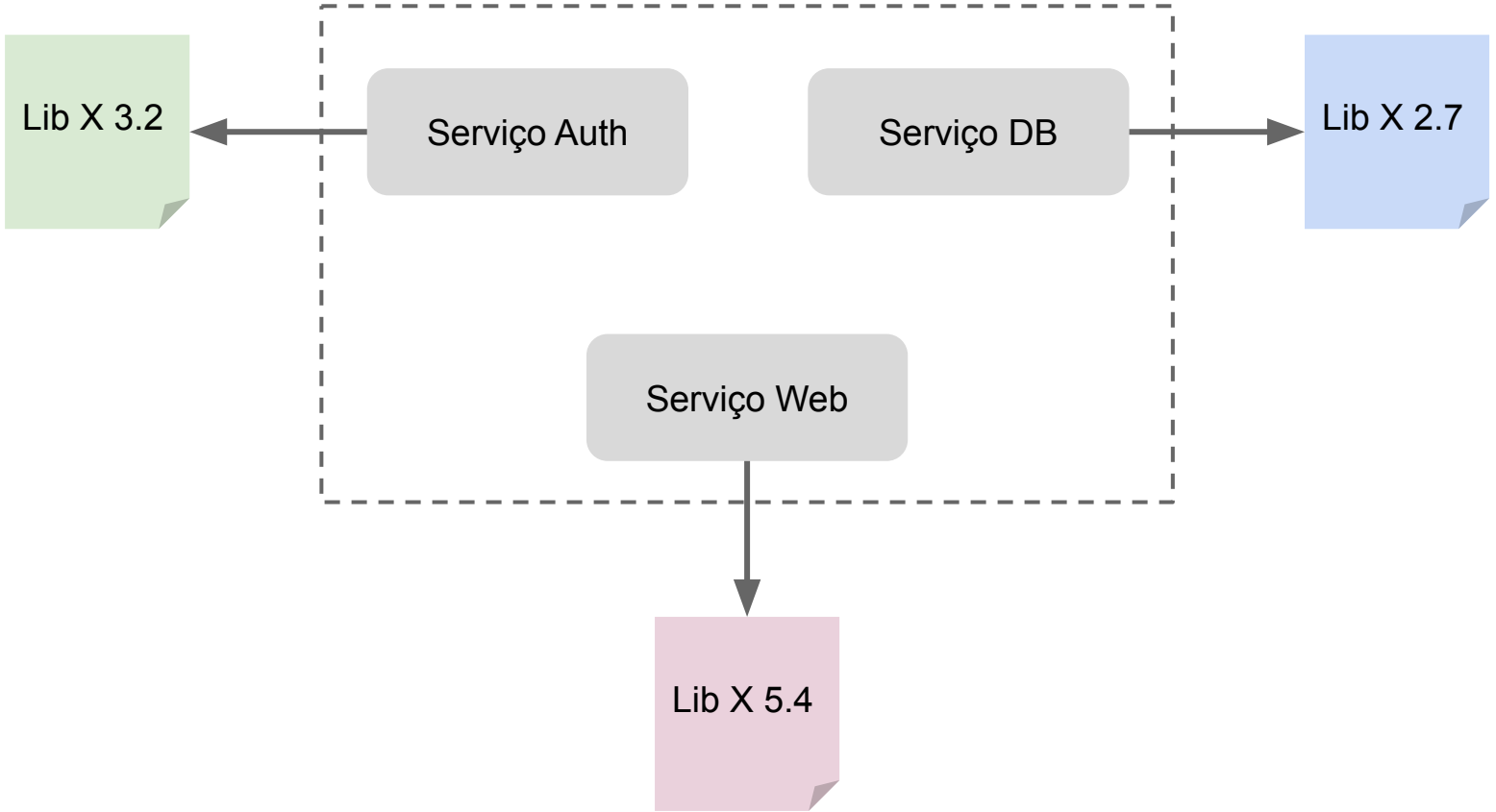

Sem SOs Específicos
Host
Container Manager
app 1
app 2
app 3
Sua máquina
Sistema de Containers
Containers

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

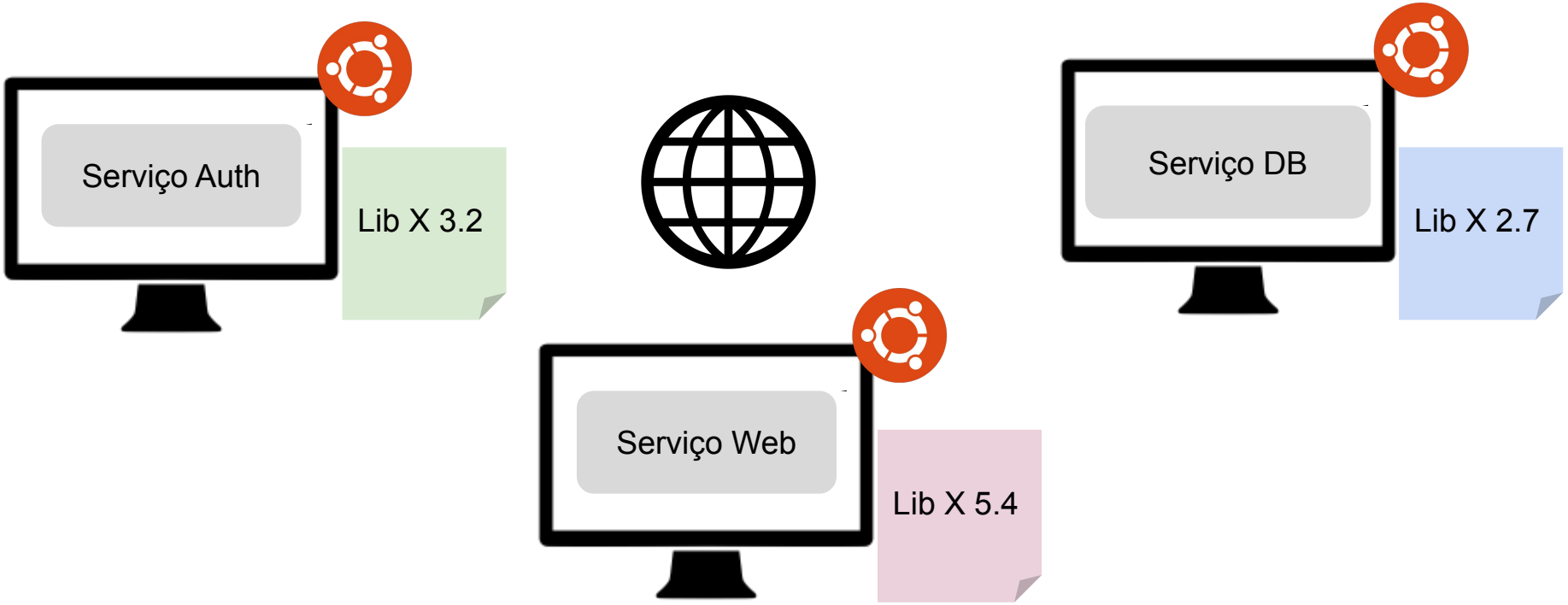
Máquina completa com o sistema

Aplicação e suas dependências

Aplicação empacotada em um executável

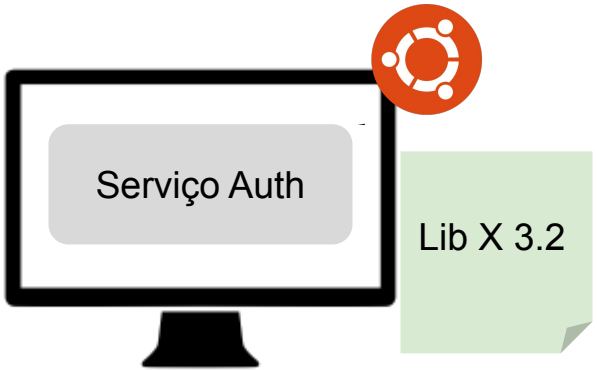

Lib X 3.2
Serviço Auth
Serviço DB
Lib X 2.7
Serviço Web
Lib X 5.4

Solução com VMs

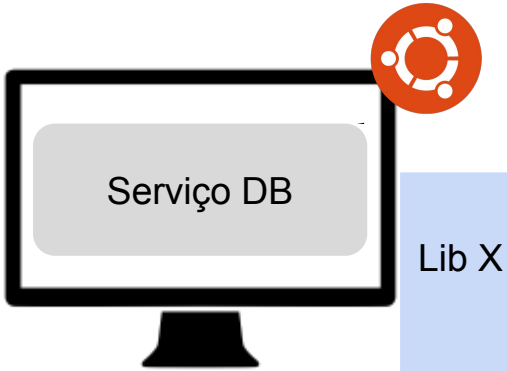


Solução com VMs

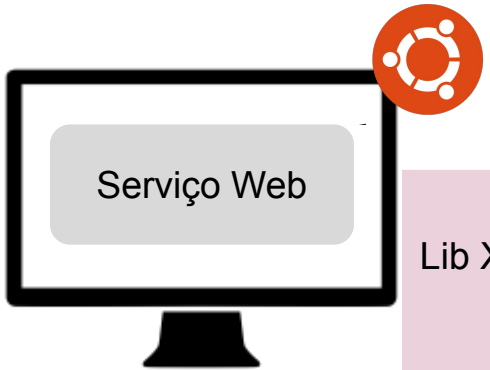
Deploy de 3 Máquinas



Lib X 3.2



Lib X 2.7

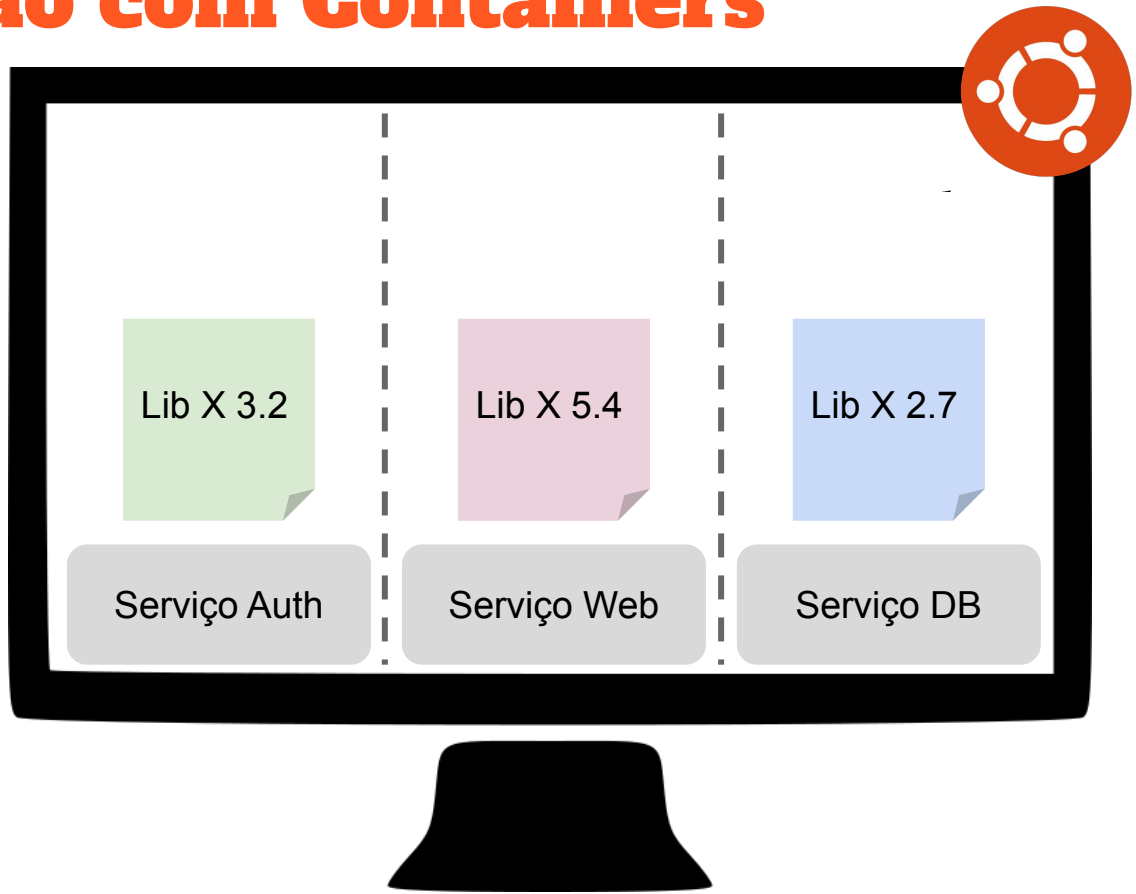


Lib X 5.4

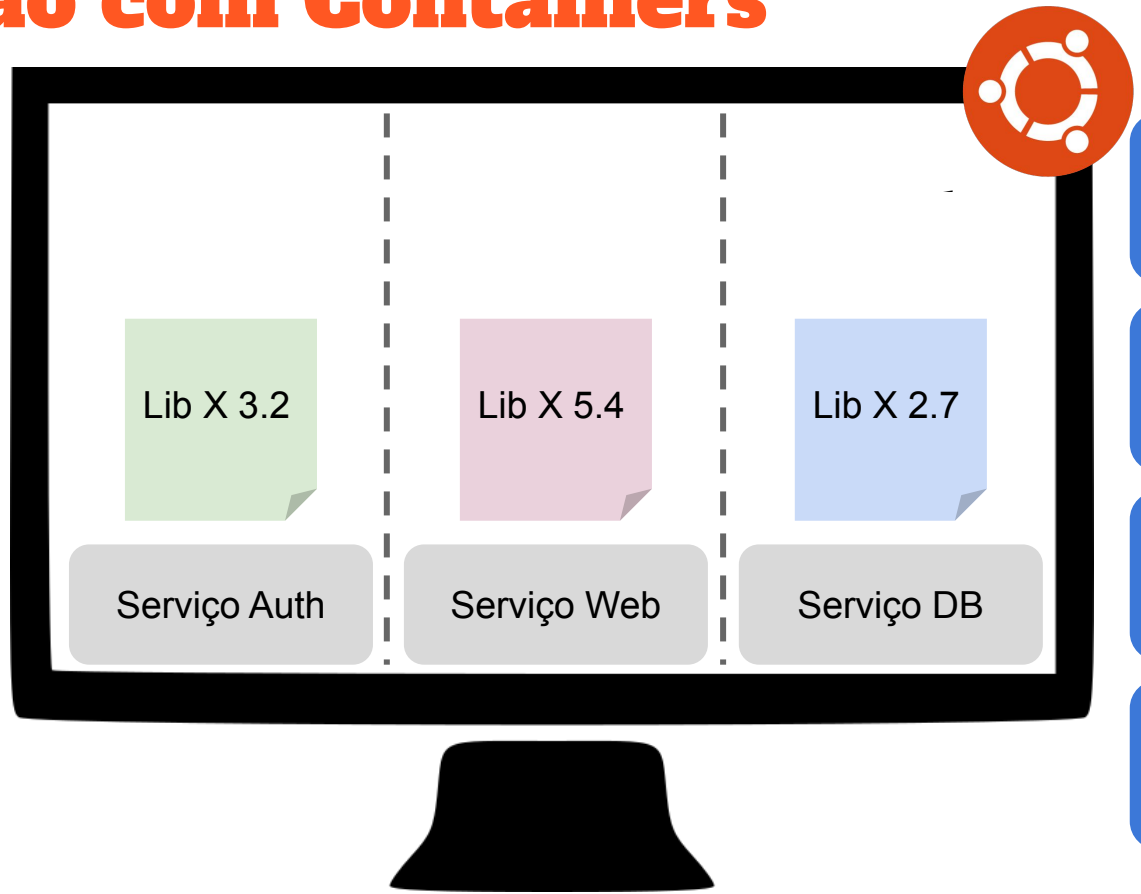
SOs Iguais só mudam a versão da Lib X

Requer 3 máquinas dedicadas a cada serviço

Solução com Containers



Solução com Containers



Containers isolados com diferentes versões da Lib X

Deploy de apenas 1 máquina

O host não possui a Lib X

Mesmo SO base com pequenas alterações

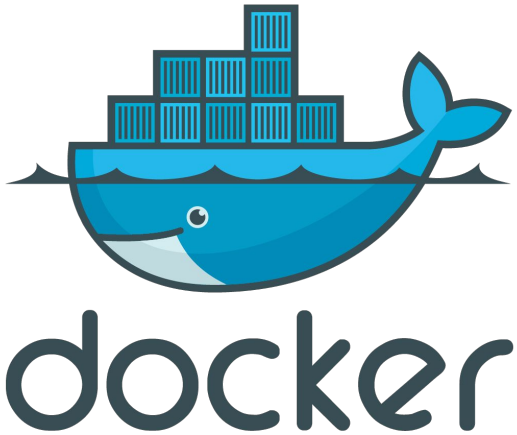
Vantagens

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

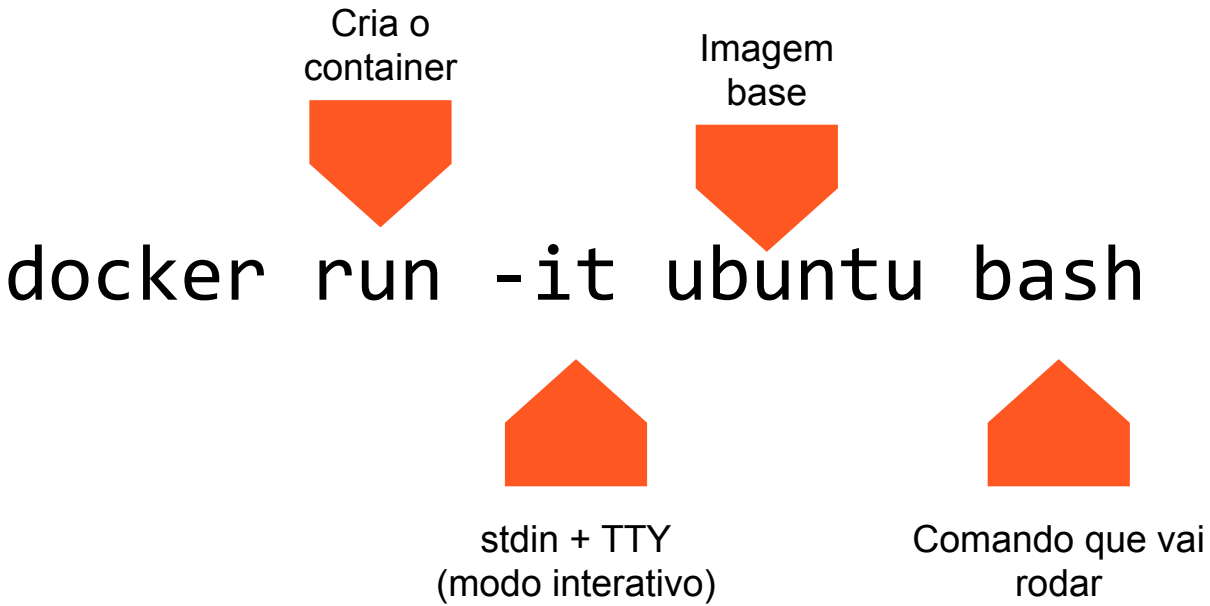


Docker

- 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


```
docker rm 001aac04ad8f
```

```
docker rm naughty_panini
```

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 carlarocha/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 carlarocha/my_ubuntu

docker pull carlarocha/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: .    # Imagem pelo Dockerfile local
    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?