

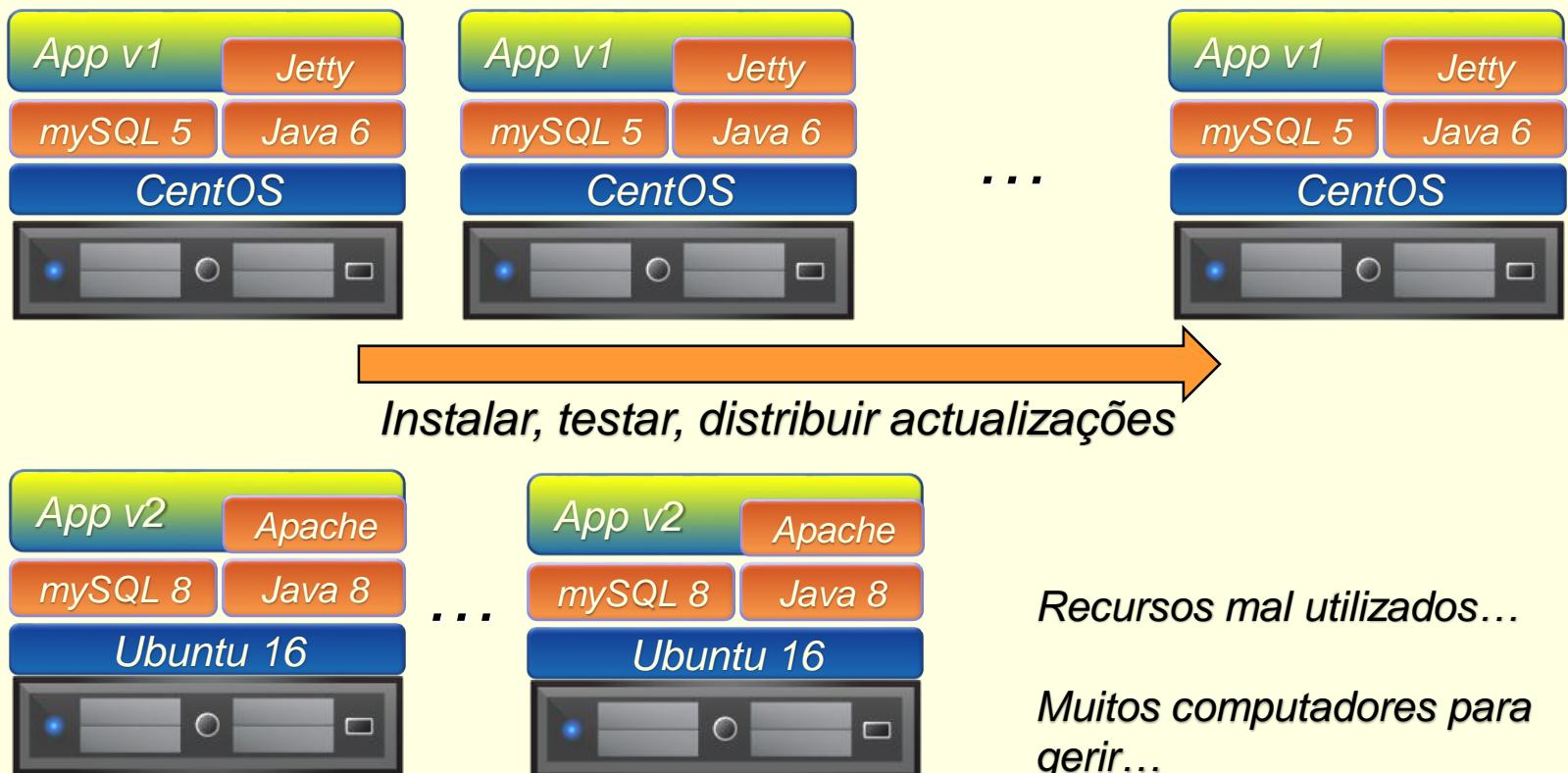
# *Fundamentos de Sistemas de Operação*

*Unix* *Windows NT* *Netware* *MacOS* *DOS/VMS* *Vax/VMS*  
*Linux Solaris* *HP/UX* *AIX* *Mach*  
*Chorus*

*A idade do ágil:*  
*1- Containers, Parte I*

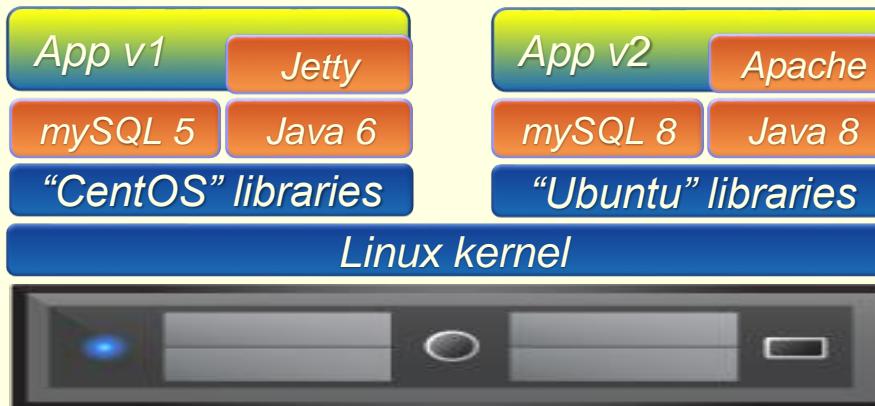
# O problema

- Muitos: desenvolvedores, versões, “softwares”, ...



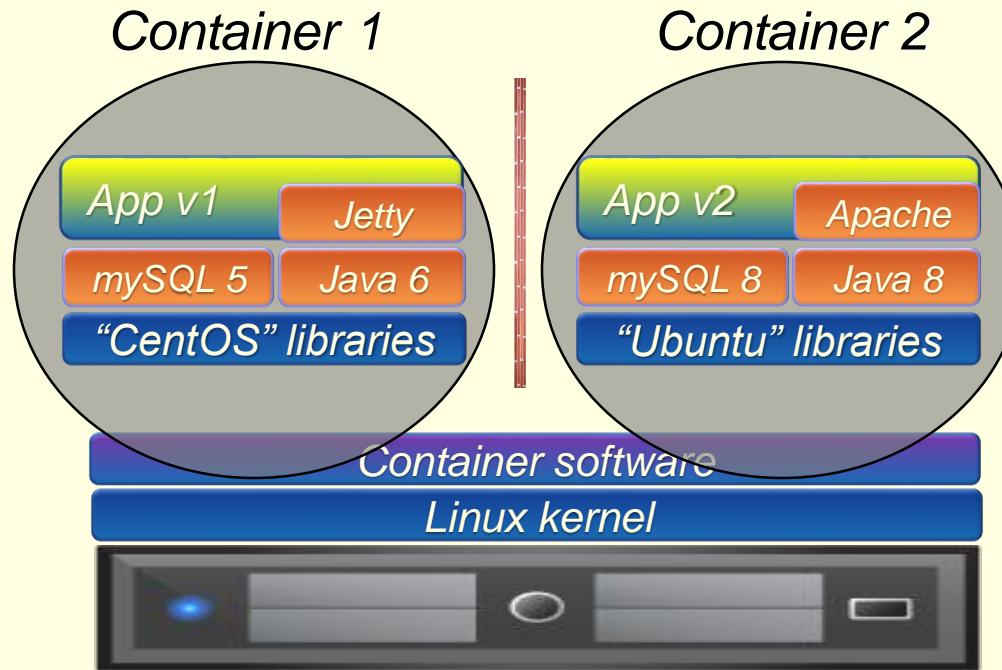
# *Uma solução pouco “prática”*

- *Partilha do hardware (melhora a utilização de recursos)*
- *Problemas:*
  - *Difícil de gerir uma série de “pacotes” software de diferentes versões mas instalados na mesma máquina – umas versões necessitam de uns, outras de outros, situação complexa, sujeita a erros...*
  - *Difícil garantir que os “gestores” e os desenvolvedores das duas Apps não interferem uns com os outros (mesmos discos, mesmos endereços IP, ...)*



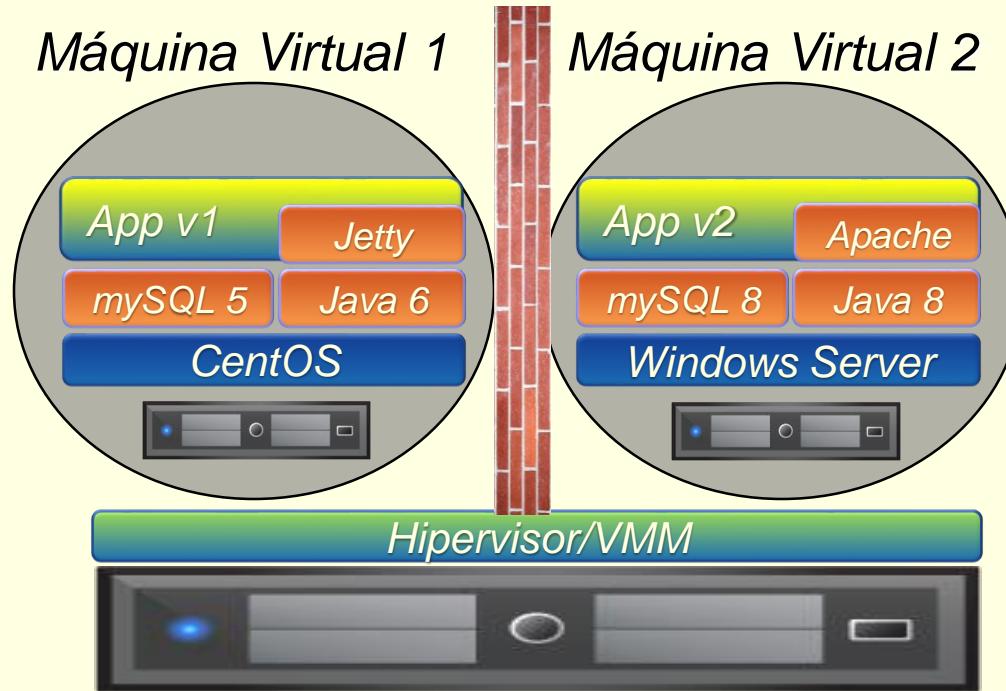
# *Uma solução melhor...*

- *Partilha do hardware (melhora a utilização de recursos)*
  - Mas... alguma capacidade de isolar os contentores uns dos outros...



# *Uma solução “extrema”...*

- *Partilha do hardware* (*melhora a utilização de recursos*)
  - *Isolamento “máximo”*



# *Containers* (1)

## □ Definição

- Mecanismo de virtualização **ao nível do SO**, que permite executar grupos de processos que partilham o núcleo do SO mas que têm espaços separados para outros recursos tais como rede, sistema de ficheiros, etc..

## □ A separação é possível ao nível de

- Espaço de nomes de processos: um container tem um grupo de processos com PIDs privados (i.e., que não “vêem” os processos nos outros containers)
- Espaço de UIDs: um container pode ter as suas próprias “contas” de utilizadores - UIDs

(cont.)

# *Containers* (2)

- A separação é possível ao nível de (cont.)
  - *Espaço de endereços (IP) de rede: um container pode ter o seu próprio endereço e hostname, podendo funcionar como um “computador separado”*
  - *Espaço de ficheiros: um container pode ter o seu repositório de ficheiros privado*
- Mecanismos no kernel Linux
  - Existem ao nível do kernel mecanismos que suportam a abstracção “container”:
    - Namespaces (PIDs, UIDs, IP, mountpoints,...)
    - Cgroups (gestão de grupos de processos, controle da afectação de recursos)

# Docker (1)

Unix Windows NT Netware Mac OS DOS/V Windows VMS  
Linux Solaris HP/UX AIX Mach Chorus

## □ O Docker é...

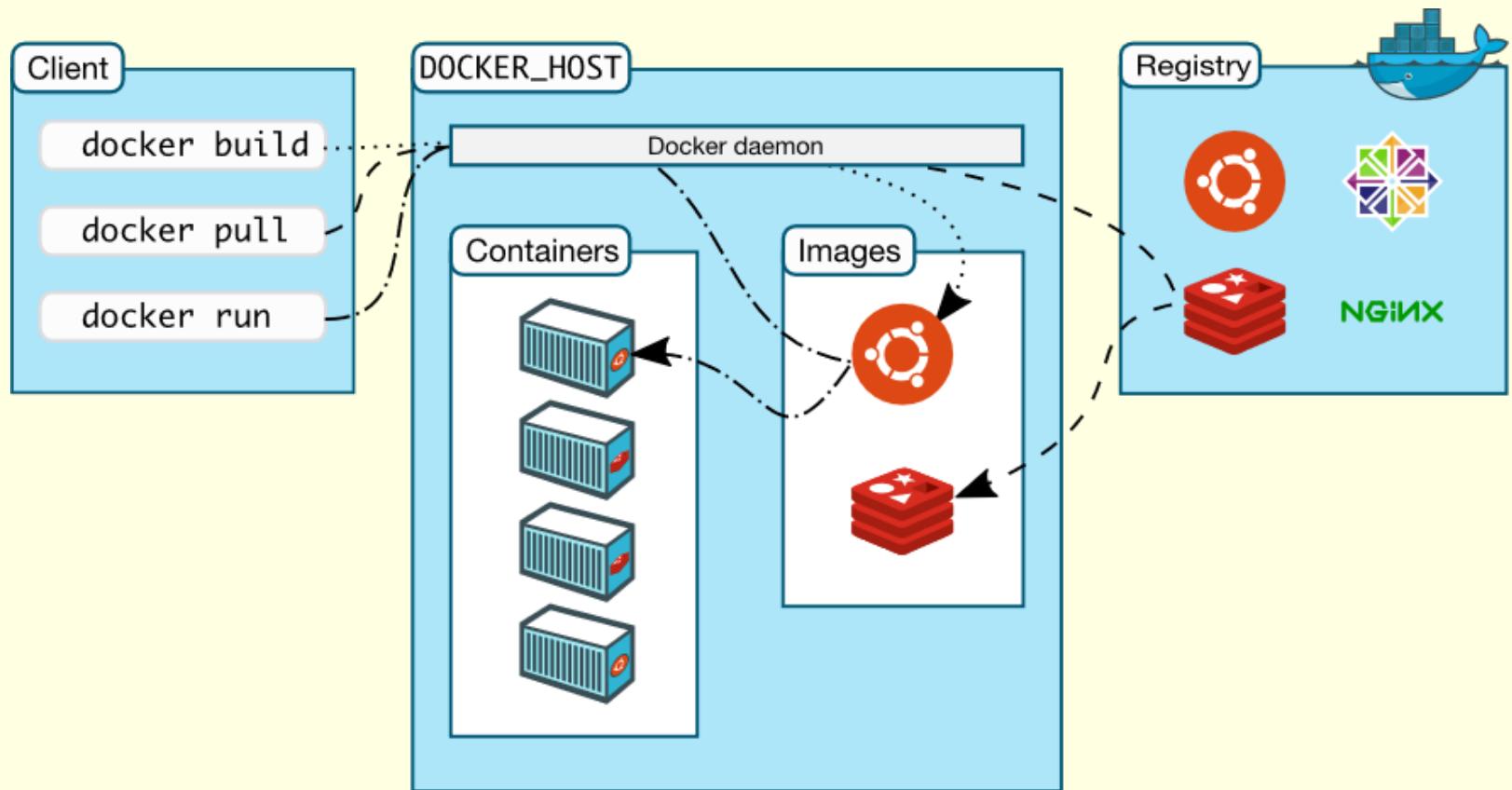
- *Uma tecnologia que usa o suporte do SO (e.g., no Linux usa os cgroups e namespaces, no Windows usa outros mecanismos) para correr código (user-level) “contentorizado”. [Outras tecnologias Linux, e.g., LXC, também usam cgroups e namespaces...]*

## □ Porque razão é tão popular?

- Porque a empresa Docker, Inc., desenvolveu um conjunto de ferramentas que tornam o processo de gestão (criação, alteração/actualização, destruição) de imagens muitíssimo simples e expedito...
- E hoje existe um ecosistema Docker com dúzias de “serviços” (de facto, alguns acham que se tornou imensamente complexo)

# Arquitectura Docker (1)

Unix Windows NT Netware Mac OS DOS/VMS Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus



# Arquitectura Docker (2)

## □ O que é uma imagem?

- *Um ficheiro contendo uma pilha de software que queremos executar no interior de um “contentor” – uma execução constitui uma instância de um contentor, que se pode designar abreviadamente por contentor*

## □ O que é um “registry”?

- É um (serviço) repositório de imagens; pode ser local (como os que existem nos Docker hosts), público (como o oferecido pela empresa Docker Inc. em <https://hub.docker.com/>), ou privado (mantido por uma organização para uso interno).

## □ O que é um “Docker host”?

- É um computador que contém o software necessário para administrar e executar imagens e contentores; em particular, o “Docker daemon” é a peça fundamental que interage com o cliente e com o SO (no Linux, interage com os namespaces, cgroups)

# *Arquitectura Docker (3)*

## □ *O que é um “Docker client”?*

- É um computador que contém o software necessário para um utilizador se poder conectar a um Docker host e neste administrar e executar imagens...

# *Fundamentos de Sistemas de Operação*

Unix Windows NT Netware Mac OS DOS/V/VS Vax/VMS  
Linux Solaris HP/UX AIX Mach Chorus

**Demo:**

*Instalar e testar o Docker*

# *Instalando o Docker (1)*

## □ Usar a VM LUbuntu64-16.04.6

- VM desligada: criar um snapshot “Antes do update”
- Actualizar o software

```
fso@fso-VM64:~$ sudo apt-get update  
fso@fso-VM64:~$ sudo apt-get upgrade  
fso@fso-VM64:~$ reboot
```

- Testar login, ver se corre tudo bem. Shutdown
- VM desligada: criar um snapshot “Antes do Docker”
- Instalar o Docker (usando a versão standard do Ubuntu)

```
fso@fso-VM64:~$ sudo apt-get docker.io
```

# *Testando o Docker (1)*

## □ Primeiro teste:

```
fso@fso-VM64:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest:sha256:4df8ca8e309c256d60d7971ea14c27672fc0d10c...f8cc17ff
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your application appears to be working correctly.

Está a aceder ao repositório público “Docker Hub”

(cont.)

# *Testando o Docker (2)*

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

```
https://hub.docker.com/
```

For more examples and ideas, visit:

```
https://docs.docker.com/get-started/
```

```
fso@fso-VM64:~$
```

# *Testando o Docker (3)*

## □ Segundo teste:

*Criar um container a correr um “ubuntu”, que só tem um bash. “Ligamo-nos” a ele por um terminal (“tipo” consola)*

```
fso@fso-VM64:~$ sudo docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
7ddbc47eeb70: Pull complete
...
45d437916d57: Pull complete
Digest: sha256:6e9f67fa63b0323e9a1e587fd71c561ba48a0...5d
Status: Downloaded newer image for ubuntu:latest
root@ad63b4326fce:/#
```

*Estamos a correr uma shell no interior de um container!!!*

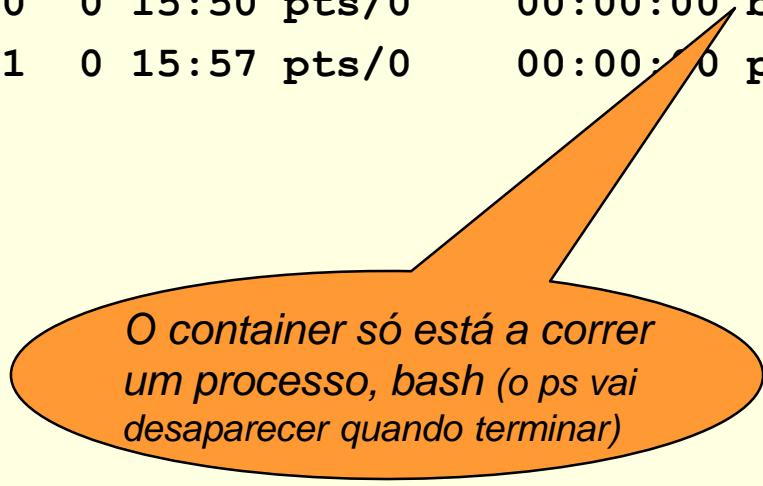
# *Testando o Docker (4)*

## □ Segundo teste:

```
root@ad63b4326fce:/# ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	15:50	pts/0	00:00:00	bash
root	10	1	0	15:57	pts/0	00:00:00	ps -ef

```
root@ad63b4326fce:/#
```



O container só está a correr um processo, bash (o ps vai desaparecer quando terminar)

# Servidor Web “contentorizado” (1)

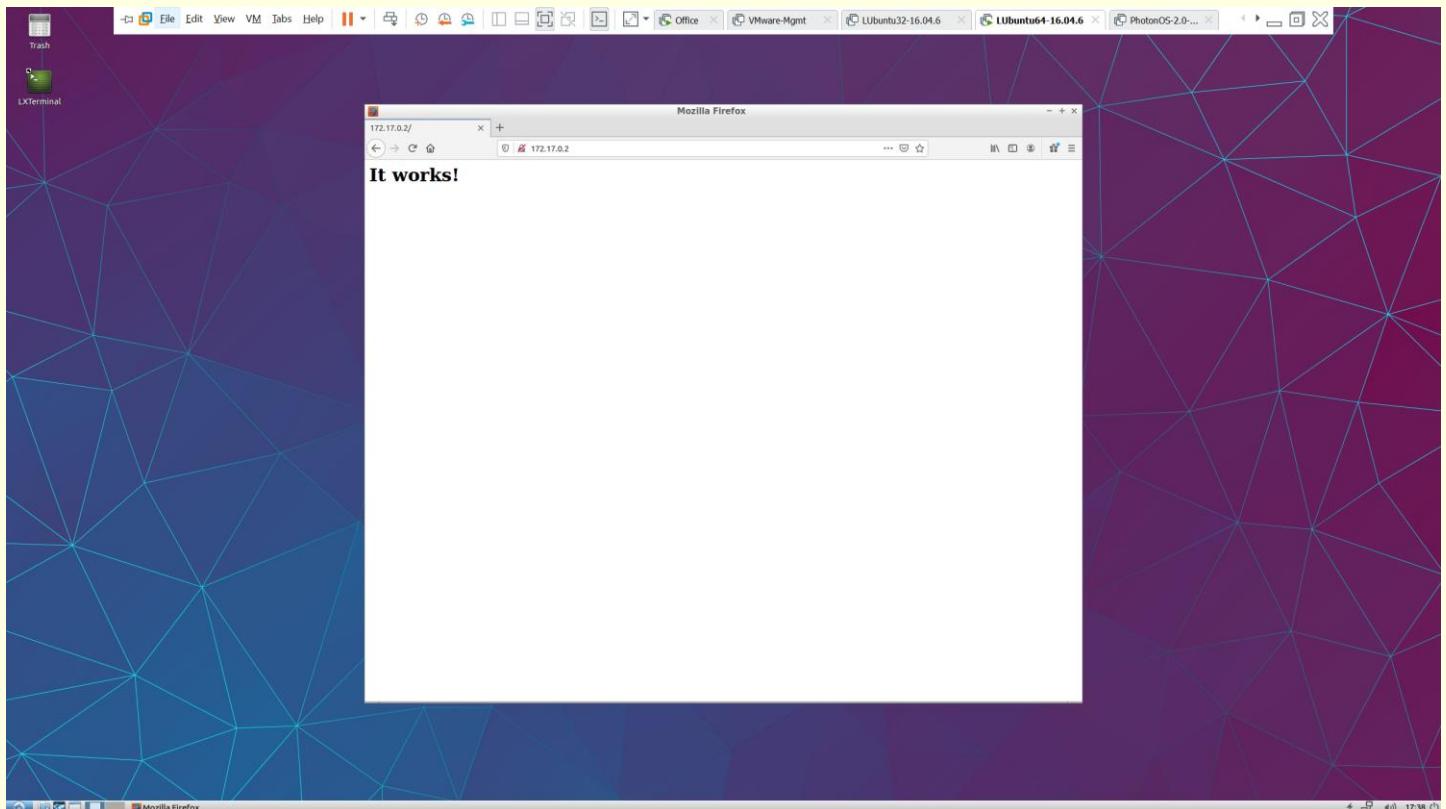
## □ Lançando o container

```
fso@fso-VM64:~$ sudo docker run -i -t httpd
AH00558: httpd: Could not reliably determine the server's fully
qualified domain name, using 172.17.0.2. Set the 'ServerName'
directive globally to suppress this message
...
...
```



# *Servidor Web “contentorizado”* (1)

- *Teste no browser da “fso-VM64”*



# *Algumas “notas”* (1)

## □ *sudo*

- *Em vez de estar constantemente a escrever sudo “isto” sudo “aquilo”, crie uma sessão root, mas cuidado... asneiras pagam-se caras ☺*

```
fso@fso-VM64:~$ sudo su - root  
[sudo] password for fso:  
root@fso-VM64:~#
```

# *Algumas “notas”* (2)

- Lançar um container que fica a correr “em background”

```
root@fso-VM64:~# docker run -d httpd
```

- Que containers estão a correr?

```
root@fso-VM64:~# docker ps
```

- Que imagens estão no repositório local?

```
root@fso-VM64:~# docker images
```

- Help...

```
root@fso-VM64:~# docker --help
```