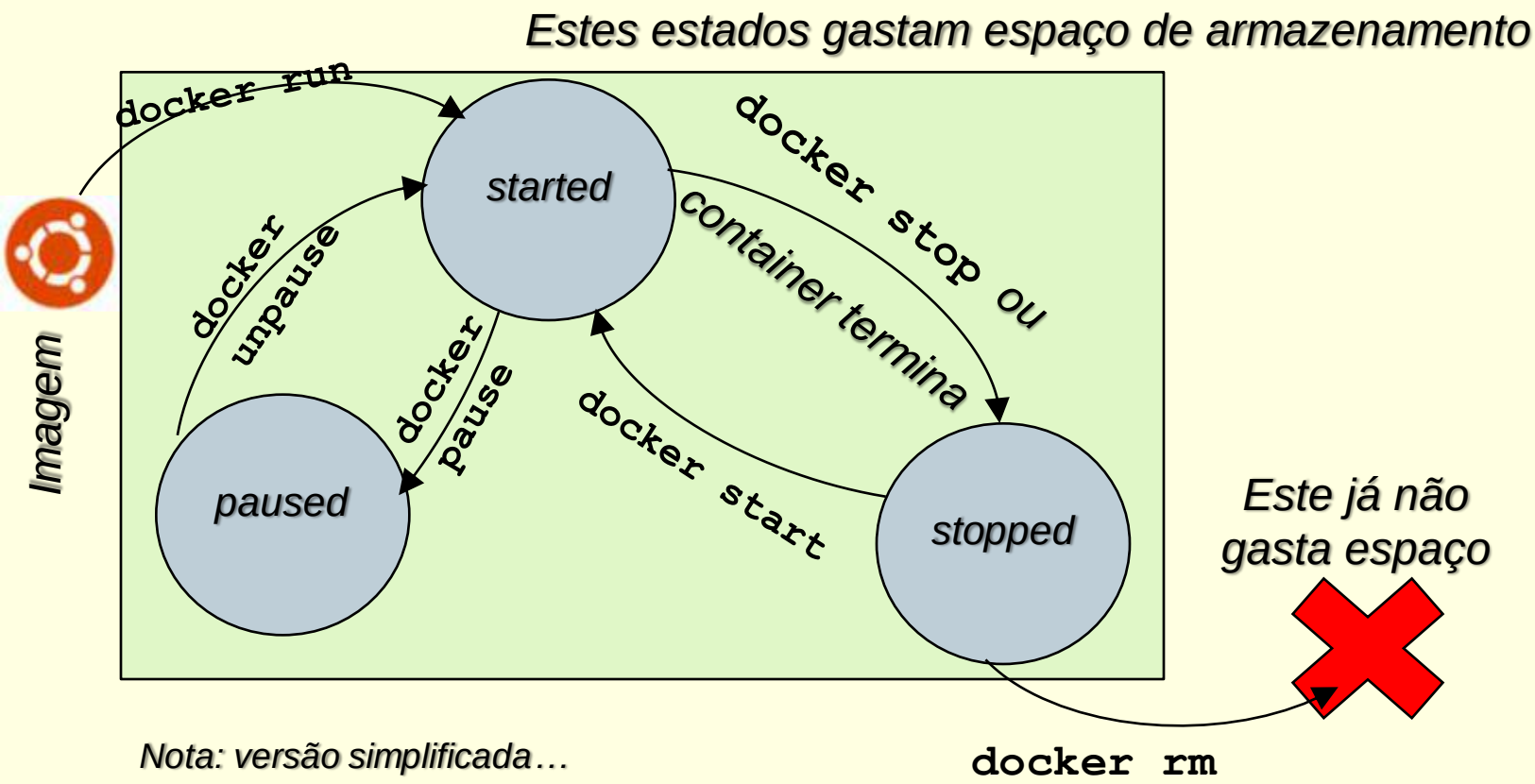
Fundamentos de Sistemas de Operação

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

A idade do ágil:

1. Containers, Parte II

Docker containers: estados (1)



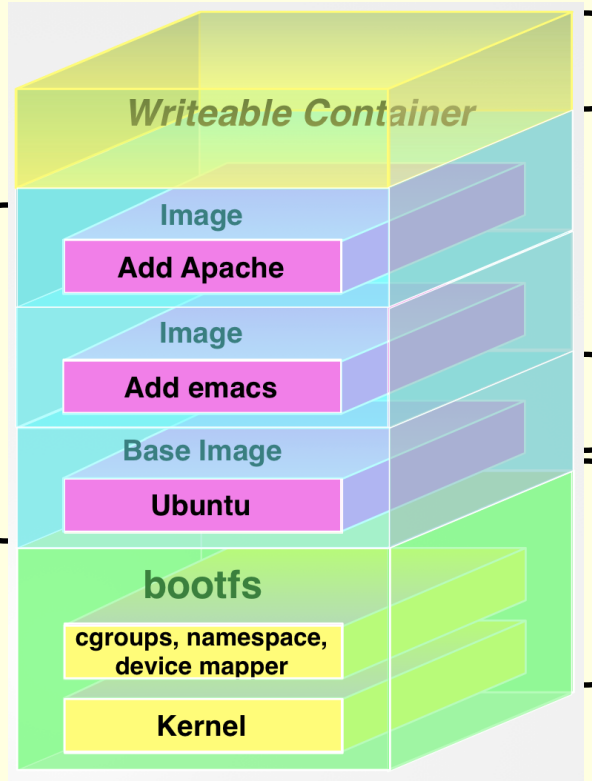
Nota: versão simplificada…

Docker containers: estados (2)

- *A partir de uma imagem,*
  - *cria-se e executa-se o container:*
    - **`docker run [opções] <imagem>`** – *Esta operação cria no SF um "ficheiro" que tem o conteúdo – executáveis, dados, configurações – que vai ser carregado em memória e executado. O "ficheiro" representa a instância.*
    - *Quando a execução termina o "ficheiro" acima referido continua a existir; as alterações efectuadas no SF são preservadas. A instância está parada.*
- *A partir de uma instância parada,*
  - *pode retomar-se a sua execução:*
    - **`docker [opções] start <stopped-container-id>`** – *Relança-se a execução da instância, usando o estado guardado no "ficheiro". Os processos são lançados de novo – não estavam suspensos, tinham mesmo terminado!*
  - *pode também remover-se a instância (o "ficheiro")*
    - **`docker rm <stopped-container-id>`**

Docker: imagens (1)

- *Em termos de espaço, muito eficientes. Porquê?*



Para funcionamento da instância

Adicionadas pelo utilizador

A imagem-base é geralmente escolhida, e não "adicionada"

Código de bootstrap da Docker, Inc.

Fonte: The Docker Book
James Turnbull (gratuito)

Docker: imagens (2)

- *Uma imagem é constituída por camadas*
  - *A camada de boostrap tem código proprietário da Docker*
  - *A camada-base é um "flavour" de SO (Debian, CentOS, Ubuntu,…). Normalmente disponível no Docker Hub – diz-se não ser fácil de criar de raíz ☺*
- *As camadas são "read-only"* e de tipo "união"*
  - *Quando numa camada N existe um ficheiro mais recente que outro numa camada "mais abaixo", o primeiro "tapa" o segundo (modelo COW do UnionFS – sistema de ficheiros de tipo "união" – ou SF tipo "overlay").*
- *A última camada é "read-write"*
  - *e suporta todas as escritas que alterem ficheiros. Existe enquanto existir a instância, e desaparece quando esta for apagada…*

* *Excepto a última*

Docker: criar nova imagem (1)

1.1 Ponto de partida: imagem existente

- *Alterar "manualmente", i.e., interactivamente*
  - **`docker run -it <imagem>`** – *lançar um container a partir de uma imagem escolhida por ser "a mais próxima" do que queremos*
  - *Carregar o software adicional (Ubuntu, apt-get; CentOS, yum), configurar os ficheiros, etc.*
- *Parar o container (stop)*

1.2 Criar a nova imagem

- *A partir do container "stopped"*
  - **`docker commit <container-stopped> <friendly-name>`** *Salvaguardar a imagem dando-lhe um nome "amigável"*

Docker: criar nova imagem (2)

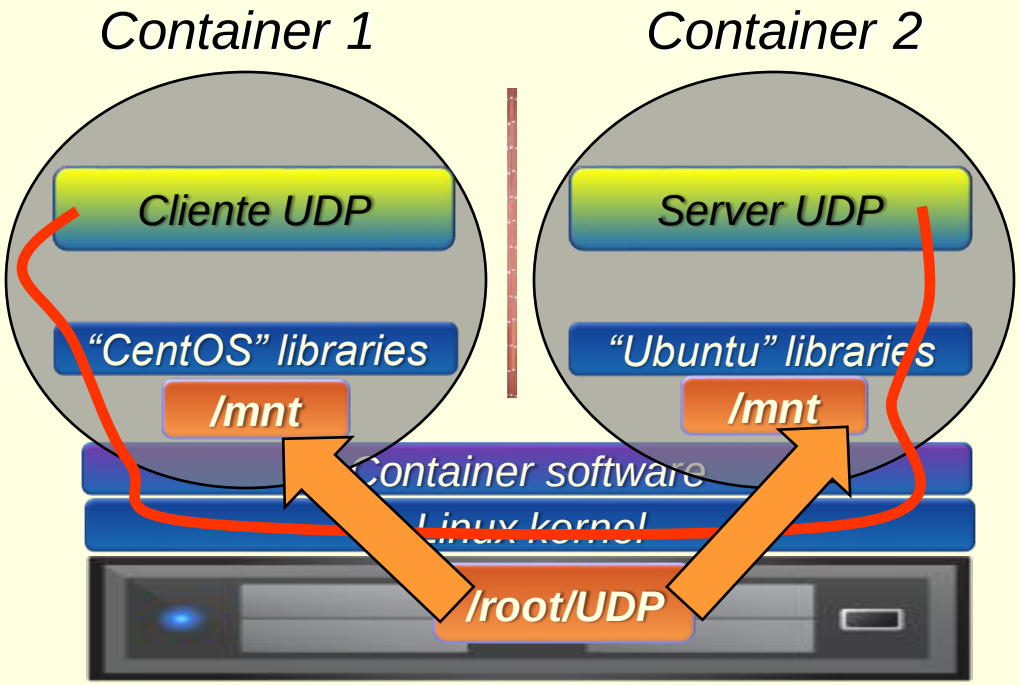
2. *Ponto de partida: imagem existente*
   - *"Criar" a partir de um script, usando ferramentas Docker*
     - *No script especificar as adições, actualizações, alterações de configurações em ficheiros, etc. num ficheiro de instruções, o* **`Dockerfile`**
     - *Podem também especificar-se os processos a lançar na instância, ports de comunicação, etc.*
     - *Executar a tool de criação da nova imagem usando o* **`Dockerfile`** *como especificação.*

Docker: volumes (1)

- *O Docker host pode "exportar" volumes para os containers*
  - *A ideia é haver armazenamento persistente, mesmo que os containers sejam destruídos e (eventualmente) recriados*
- *Um volume*
  - *É uma directoria no Docker host*
  - *É um "mount point" no(s) container(s)*
  - *Pode ser "read-only" (ro) ou "read-write" (rw)*
  - *Pode ser partilhado por vários containers*
  - *Só pode ser especificado no comando docker run*
- *Formato*
  - **`-v <dir-no-host>:<mount-point-no-container>[:ro|rw]`**

Docker: volumes (2)

- *Exemplo na demo: Cliente-Servidor UDP*
  - **`docker run -it -v /root/UDP:/mnt:ro ubuntu /bin/bash`**



Fundamentos de Sistemas de Operação

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

Demo: Docker…