



Sistemas Operativos

Capítulo 2



Patrício Domingues, ESTG/IPLeiria
2019

O papel de um SO

- ✓ Um computador é um conjunto de recursos orientadas para o movimento, armazenamento e processamento de dados
 - O SO é responsável pela gestão desses recursos



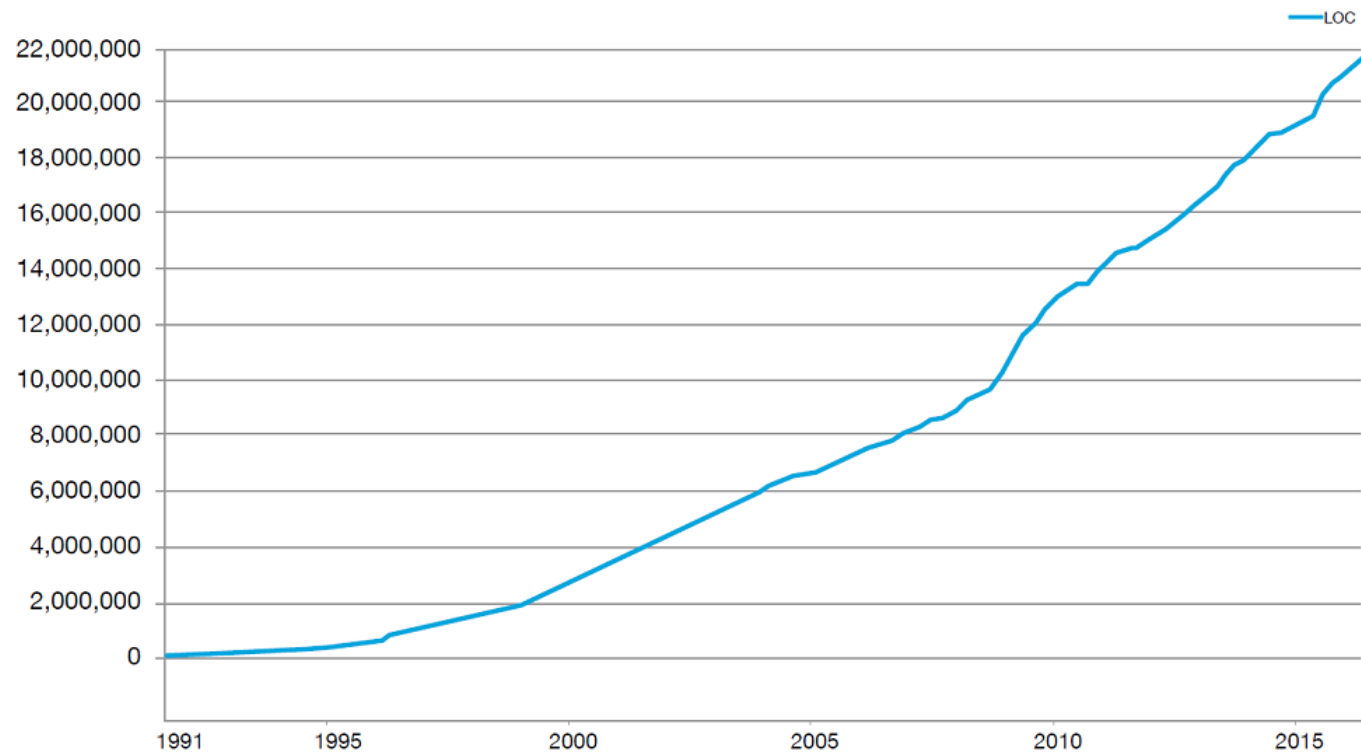
Sistema operativo é...

- ✓ Software que controla a execução das aplicações
 - Na realidade trata-se de um conjunto (vasto) de programas
 - Kernel do Linux 4.x (2015): 22 milhões de linhas...(slide seguinte)
- ✓ Interface entre aplicações e o hardware
 - Exemplo
 - Uma aplicação não manipula diretamente um ficheiro
 - A aplicação recorre a serviços do sistema operativo (através das funções “open”, “close”, “write”, “read”...)
 - Chamadas ao sistema

Kernel do Linux

- ✓ Linhas de código do kernel do Linux
 - Quanto mais linhas de código, mais *bugs*...
 - Fonte: <http://bit.ly/2lesVnr>

Total Lines of Code in the Linux Kernel



✓ Conveniência

- Facilitar a vida ao utilizador/programador

✓ Eficiência

- Uso eficiente dos recursos do sistemas
 - Maximizar o aproveitamento dos recursos (CPU, memória, disco,...)

✓ Capacidade para evoluir

- As plataformas de hardware vão evoluindo (rapidamente!) bem como as exigências dos utilizadores
 - Um sistema operativo deve possibilitar a evolução

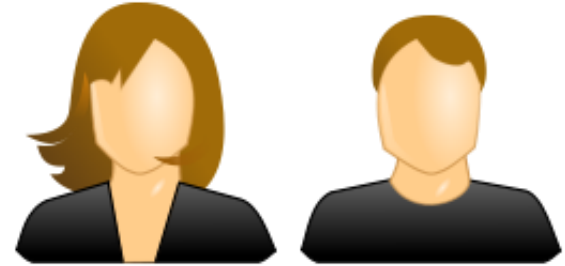


- Desenvolvimento de aplicações
- Execução de programas
- Acesso a dispositivos de E/S
- Acesso controlado a ficheiros
- Acesso ao sistema
- Detecção de erros
- Contabilização de uso, recursos, etc.
- ...

- ✓ Um SO tem quatro classes de utilizadores
 - Utilizador de aplicações
 - Programador de aplicações
 - Programador de sistemas
 - Administrador de sistemas
- ✓ Cada classe de utilizadores tem necessidade diferentes

Próximo slide: descrição das classes de utilizadores >>

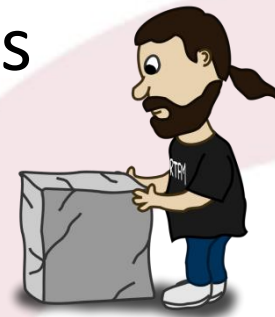
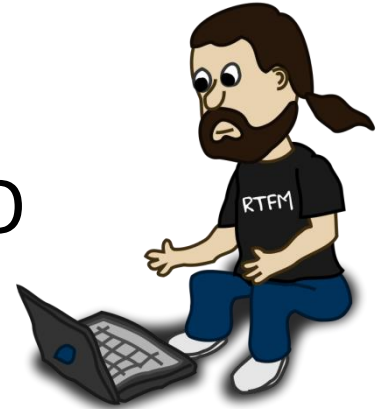
✓ Utilizador de aplicações pretende...



- SO fácil de usar e aprender
 - Idealmente, que nem seja preciso aprender
- SO que se adapte ao utilizador
- SO que responda rapidamente às solicitações
- SO sem surpresas (e.g.: “apagar ficheiro sem avisar” é uma surpresa altamente indesejável...)
- SO com alternativas na realização de tarefas
 - Exemplo:
 - Alguns utilizadores preferem o rato, outros utilizadores preferem o teclado

✓ Programador de aplicações

- Acesso fácil a chamadas de baixo nível do SO
 - Leitura de input via teclado, posição do rato
- Visão consistente do sistema ao programador
- Uso fácil de serviços de alto nível
 - Criação de janelas, acesso a canais de comunicação via rede, código e serviços para cifração de dados, etc. ...
- Portabilidade para outras plataformas



- ✓ Programador de sistemas
 - Desenvolve software para tarefas do sistema
 - Device drivers, código do núcleo,...
- ✓ SO deve possibilitar...
 - fácil criação de programas corretos
 - fácil depuração de programas incorretos
 - fácil manutenção de programas
 - fácil alterar/expandir programas



✓ SO deve possibilitar...

- Fácil acréscimo/remoção de dispositivos como discos, impressoras, ligações de rede, etc.
- Segurança para todos os utilizadores do sistema, dados, etc.
- Fácil atualização do sistema para novas versões
- Fácil gestão de utilizadores
- Boa gestão dos recursos do sistema



✓ Tipos

- SO mono utilizador, mono tarefa
- SO multitarefa
- SO não interativo (*batch mode* – processamento lotes)
- SO multiutilizador / time-sharing
- SO de tempo real
- Sistemas virtualizados
 - Não é bem um sistema operativo

Próximos slides: Análise aos vários tipos de SO >>



- ✓ O SO apenas executa um processo de cada vez
 - Processo: programa em execução
- ✓ SO típico dos primórdios dos computadores pessoais
 - SO simples e limitados
 - Interface de texto sem gráficos (linha de comandos)
- ✓ Exemplos
 - CP/M
 - Microsoft DOS
 - Limitado a 640 KB de memória

```
File Edit View Insert Format Utilities Macro Window Help
README.DOC
This file supplements the printed documentation for
Microsoft Word, version 5.5B.

-----Contents-----
. Other Sources of Information
. Release Information for 5.5B
. Additional Setup Information
. Installing Word To Run Under Windows 3.0
. Using Word Under Windows 3.0 with a Hercules Adapter
. Installing Word To Run Under Windows 2.1
. New Mouse Driver
. Running Word on a Floppy Disk System
. Additional Style Sheet Information
. Additional Macro Information
. Using Word with a Monochrome VGA Adapter
. Using an IBM 8514 Monitor Under OS/2
. Using an IBM PS/2 Model 70 Display Under OS/2 1.21
. Using Word with KEYB.COM
. Using Word with Presentation Manager
. Mouse Support and OS/2

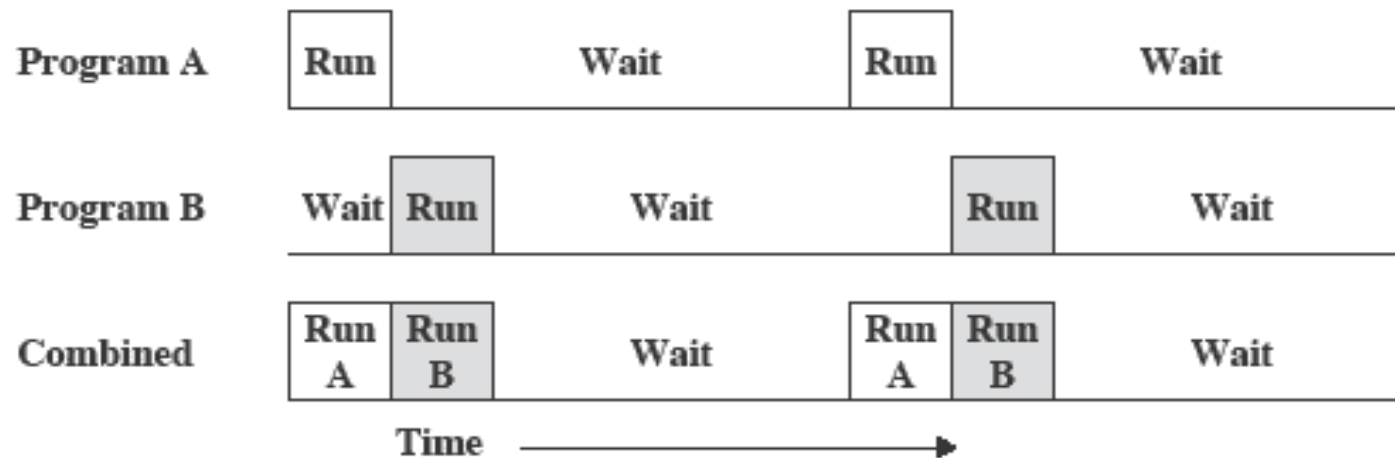
Pg1 Co1      {}      <F1=Help>      MX      Microsoft Word
Edit document or press Alt to choose commands
```

SO multitarefa (1)

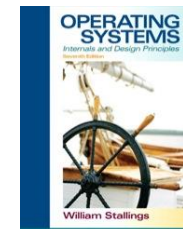
- ✓ SO suporta a execução “simultânea” de várias aplicações
 - Embora um só utilizador interativo
- ✓ Requer a capacidade de comutação de tarefas
 - CPU executa parte de um processo, depois executa parte de outro, etc.
 - Tudo isso num curto intervalo de tempo (e.g. 30 processos por segundo)
 - Utilizador pensa que os processos estão em execução simultânea
- ✓ Qualquer SO recente é multitarefa (“Windows”, Linux, Mac OS X, iOS, Android...)

SO multitarefa (2)

- ✓ O SO vai comutando as tarefas
 - Uma tarefa é substituída por outra quando:
 - Já está há demasiado tempo a executar no CPU (e.g. 50 ms)
 - Requer o uso de um dispositivo de Entrada/Saída



(b) Multiprogramming with two programs



SO não interativo (1)

- ✓ Também designado por processamento em lotes (*“batch”*)
- ✓ Os recursos de um sistema de computação de alto desempenho são caros (e.g., supercomputador)
 - Supercomputador com 1024 CPUs e GPUs



China Sunway BlueLight MPP Supercomputer



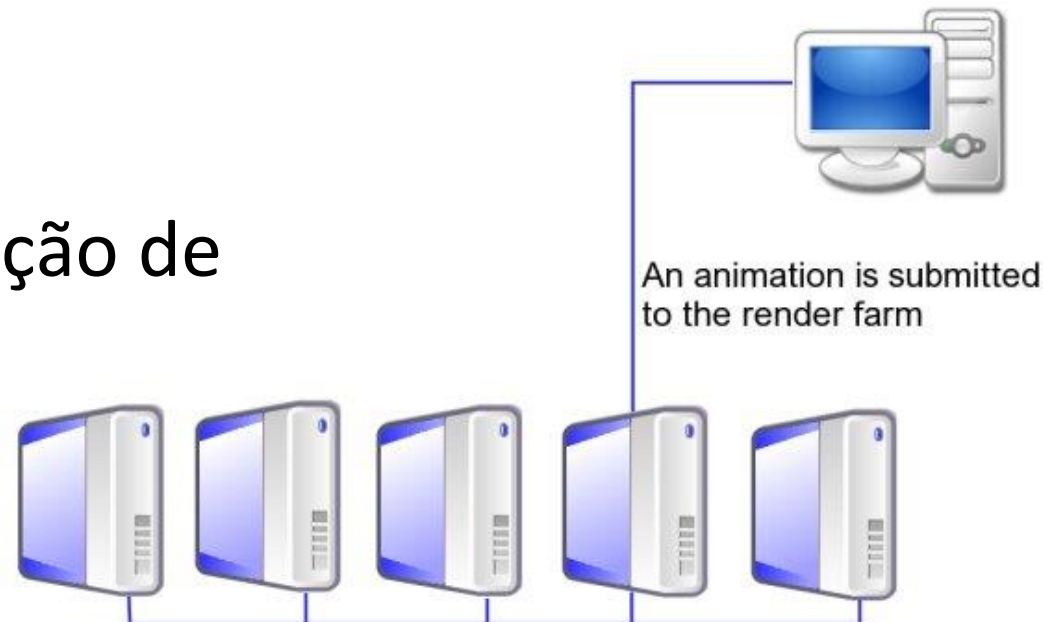
- ✓ O SO não permite o uso interativo do sistema
 - Utilizadores recebem “lote de horas de computação”
- ✓ Exemplo: 1000 horas

SO não interativo (2)

- ✓ As tarefas devem ser autónomas
 - Não podem parar a meio a pedir indicações aos utilizadores
 - Tarefas em lote (“batch processing”)
 - Existe um computador de acesso
 - *Frontend machine*

- ✓ Exemplos

- Computadores para a criação de imagens de síntese (“rendering farms”)



<http://ainkaboot.co.uk/cluster-architecture.php>



IPL

escola superior de tecnologia e gestão
instituto politécnico de leiria

Rendering farm da Disney

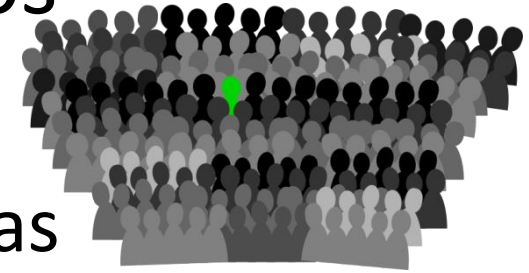
- ✓ Recursos computacionais empregues na renderização do filme “frozen” (2013)
 - 30,000 Core Renderfarm
 - 60M horas renderização
 - 1.5MW centro de dados
 - 6PB armazenamento
 - 1000 computadores linux
 - 800 Macs
- ✓ Fonte:
https://www.usenix.org/sites/default/files/conference/protected-files/lisa13_geibel_johnson_slides.pdf



<http://bit.ly/2sCnblj>



✓ SO permite o uso simultâneo de vários utilizadores

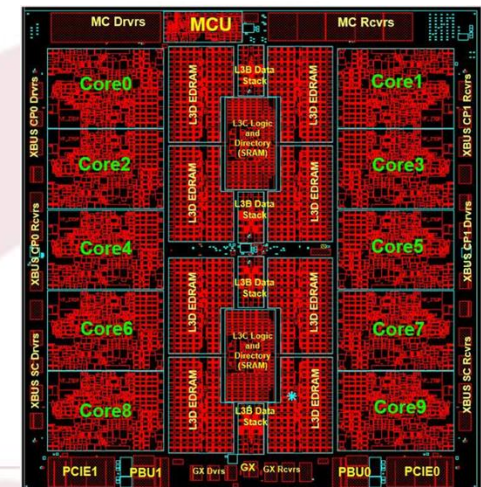


- Utilizadores ligados em sessões separadas
- O tempo de CPU é partilhado pelos utilizadores
 - *Timesharing*
- Tarefas interativas deve ser tratadas rapidamente
 - Curto tempo de resposta
- Empregues inicialmente em sistemas de grande porte
 - IBM OS 360, multics (anos 1960)
- Atualmente
 - servidores de ficheiros, de bases de dados, servidores web



SO multiutilizador/timesharing (#2)

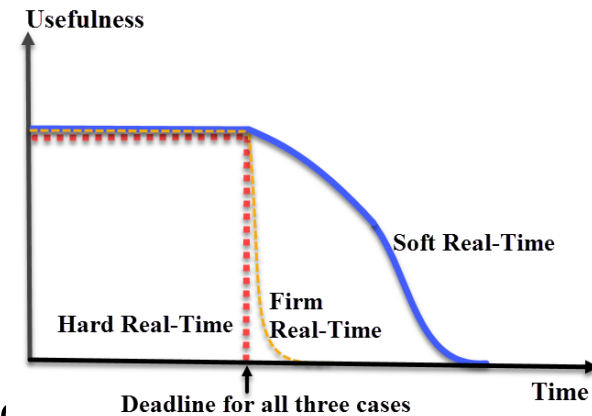
- ✓ Exemplo de servidor *timesharing*
 - IBM mainframe Z-serie
- ✓ z14 suporta
 - Suporta até 12 x 10⁹ transações cifradas por dia
 - Suporta até 32 TiB de memória RAM
 - Importante manter os dados cifrados para proteção de dados (RGPD)
- ✓ 23 dos 25 maiores vendedores usam *mainframe*
- ✓ 92 dos 100 maiores bancos usam *mainframe*



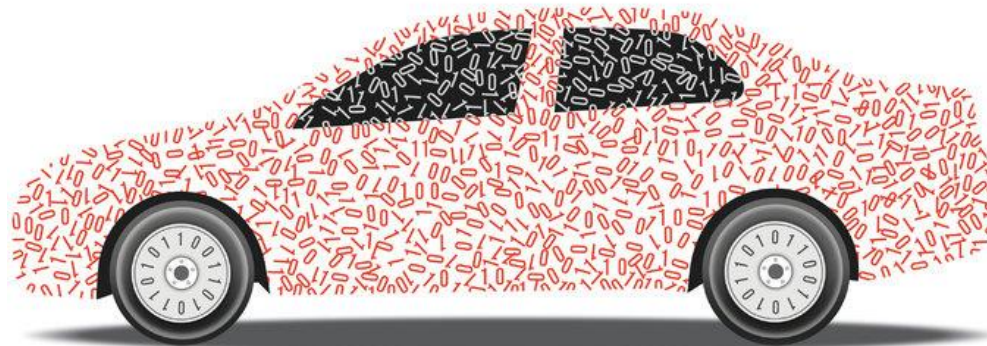
- ✓ *“A real-time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation, but also upon the time at which the result is produced. If the timing constraints are not met, system failure is said to have occurred.”*
 - Fonte: Real-time Computing FAQ
- ✓ Conceito de “latência”
 - Na computação, designa-se por latência o tempo que medeia entre um evento e a resposta a esse mesmo evento

✓ Sistemas de tempo real:

- **hard realtime** – garantem a execução das tarefas críticas dentro dos limites de tempo estabelecidos
 - **Exemplos:** sistema ABS/ESP de uma viatura; sistema de segurança de central nuclear
- **firm realtime** – falhar um *deadline* é tolerado se o número de falhas for muito diminuto. A utilidade de um resultado é nulo após que tenha expirado o *deadline*.
 - **Exemplo:** *robot de construção automóvel que falha um ponto de soldadura. Se falhar vários pontos de soldadura consecutivamente pode criar problema de qualidade*
- **soft realtime** – apenas é dada prioridade às tarefas críticas, não sendo garantida a sua execução dentro de limites de tempo estritos. Um resultado produzido após expirado o *deadline* ainda pode ter utilidade.
 - Muitas vezes, o “soft” realtime é especificado através de probabilidades. O sistema garante com probabilidade **P** que a resposta a determinado evento ocorrerá num prazo **T**
 - **Exemplos:** visionamento em *streaming* de um vídeo; resposta do OS a evento no ecrã *touch* do telemóvel



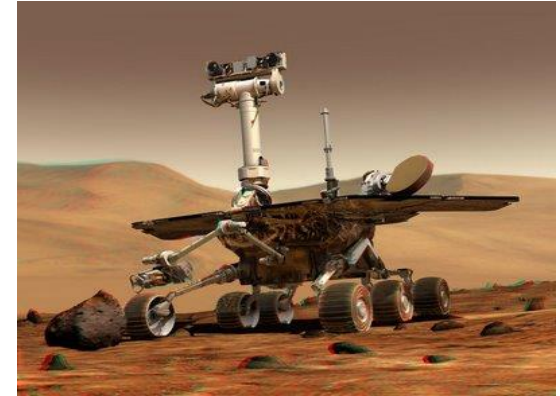
- ✓ Veículos do século XXI
 - Computadores sobre rodas...
- ✓ Sugestão de leitura
 - “Complex Car Software Becomes the Weak Spot Under the Hood”, NYTimes, September, 26th 2015
 - Link: <http://nyti.ms/1iBBEKa>



SO de tempo real (3)

✓ Áreas de aplicação

- Sistemas de suporte de vida (“e-care”)
- Sistemas de controlo industrial
- Sistemas de comunicações
- Sistemas de controlo de tráfego aéreo, aeronaves
- Sistemas de controlo de motores
- Detecção de fraude financeira (clonagem de cartões, etc.)
- Robótica, exploração espacial
- Etc.



- ✓ Definição de máquina virtual
 - Implementação via software de uma máquina (por exemplo, um computador) que executa programas / processos
- ✓ Existem dois tipos de máquinas virtuais
 - Máquina virtual *software*
 - Máquina virtual *sistema*

Próximo slide: máquina virtual software >>

✓ Máquina virtual “software”

- Máquina abstrata que implementa uma determinada “Instruction Set Architecture” (arquitetura virtual)
- A máquina virtual disponibiliza alguns serviços
 - Gestão de memória, gestão de fluxos de execução (threads), ...
- Exemplos
 - bytecode da JVM (Java Virtual Machine)
 - CLI (Common Language Interface) do Microsoft .NET
 - O código fonte dos programas é compilado para a arquitetura virtual
 - *bytecode* no java
 - CLR no .NET



✓ (+) Portabilidade

- Código pode ser executado em qualquer máquina física/SO, desde que exista uma implementação da máquina virtual para o SO em causa

✓ (+) Programação

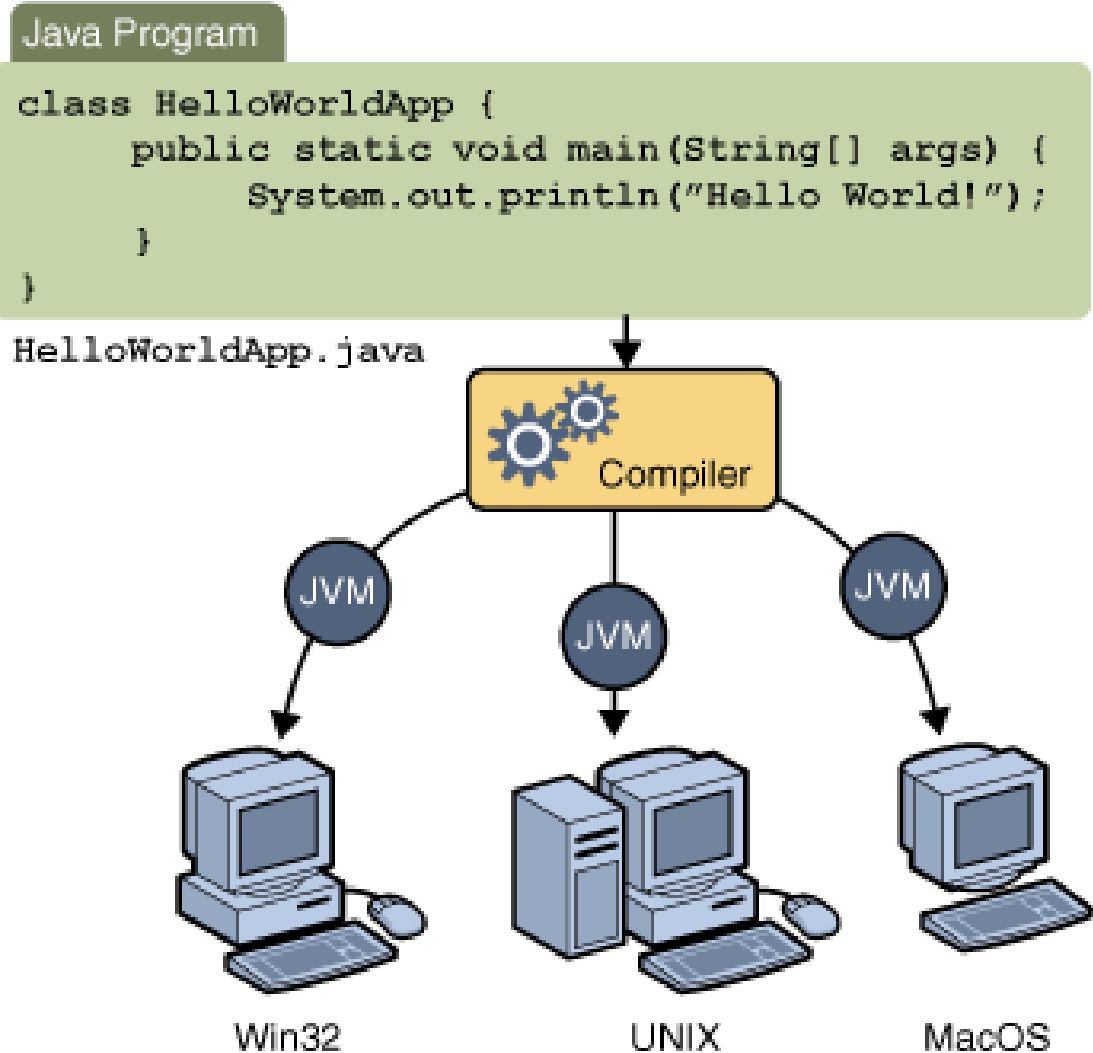
- As máquinas virtuais software disponibilizam várias funcionalidades que facilitam a programação
 - Gestão de memória dinâmica, mecanismos de *threading* e sincronização, ...

✓ (-) Lentidão

- Código é interpretado, o que torna a sua execução mais demorada

Exemplo: *java* e *bytecode* (1)

1. Escrita do código java
(.java)
2. Compilação: o
compilador cria uma
representação em
bytecode
3. O bytecode é executado
na Java Virtual Machine
(JVM)



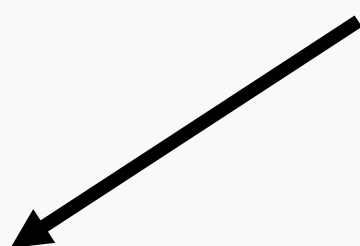
Fonte: <http://bit.ly/9QKMtO>



Exemplo: *java* e *bytecode* (2)

Code:

```
0:   iconst 2
1:   istore 1
2:   iload 1
3:   sipush 1000
6:   if icmpge      44
9:   iconst 2
10:  istore 2
11:  iload 2
12:  iload 1
13:  if icmpge      31
16:  iload 1
17:  iload 2
18:  irem           # remainder
19:  ifne          25
22:  goto          38
25:  iinc          2, 1
28:  goto          11
31:  getstatic     #84; //Field java/lang/System.out:Ljava/io/PrintStream;
34:  iload 1
35:  invokevirtual #85; //Method java/io/PrintStream.println:(I)V
38:  iinc          1, 1
41:  goto          2
44:  return
```



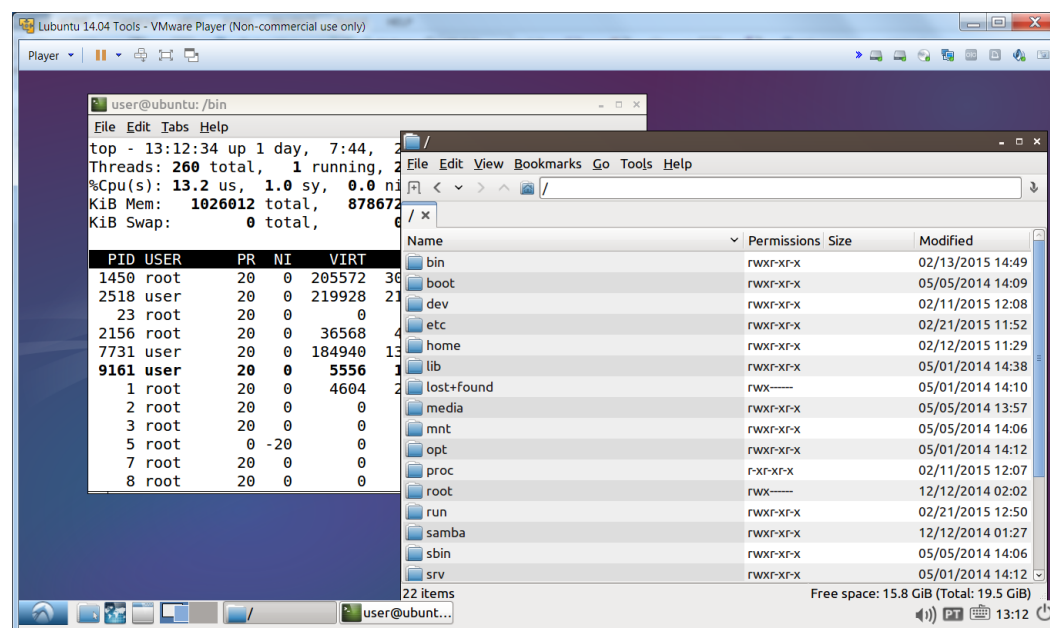
outer:

```
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

Fonte: wikipedia

Máquina virtual sistema (1)

- Software que cria a ilusão de um sistema físico (PC) dentro de um outro SO
 - (e.g. VmWare, Xen, Parallels, Virtual PC, Virtualbox, Virtual Iron, QEMU)
- Uma máquina virtual sistema implementa uma máquina...dentro de outra máquina





✓ Uma máquina virtual sistema deve providenciar as seguintes funcionalidades:

➤ Similitude

- O comportamento da máquina virtual sistema deve ser indistinto do comportamento de uma máquina física

➤ Eficiência

- A máquina virtual deve providenciar um desempenho similar ao das máquinas físicas

➤ Gestão de recursos

- A máquina virtual deve ter pleno controlo sobre os recursos que implementa (e.g. disco virtuais, rede virtual, etc.)
- A máquina virtual deve isolar as aplicações do sistema operativo hospedeiro

Fonte: Diogo Ferreira (U. Coimbra)



Máquina virtual sistema (2)

- ✓ Quando se fala no contexto de máquinas virtuais sistema, fala-se em...
 - SO Hospedeiro
 - SO que efetivamente executa o software de virtualização
 - Aulas práticas: SO Windows dos laboratórios
 - SO Hóspede
 - SO que corre na máquina virtual
 - Ubuntu no caso das aulas práticas
- ✓ Designação anglo-saxonica
 - Host OS (hospedeiro) and guest OS (hóspede)

Máquina virtual sistema (3)

- ✓ As máquinas virtuais ganharam muita relevância
 - Possibilitar a agregação de serviços, i.e., vários serviços executados em máquinas físicas separadas passam a ser executados em máquinas virtuais
 - Várias máquinas virtuais pode ser executadas na mesma máquina física
 - Menos máquinas, menos custos (hardware, energia e recursos humanos)
- ✓ Máquinas virtuais também são ótimas para testes
 - Vários SO podem correr numa mesma máquina

✓ (1) Virtualização completa

- Completa simulação do hardware físico
- O SO hóspede pensa que está num sistema físico (não existem diferenças), a execução é transparente
- Não é necessário modificar o SO hóspede

✓ Exemplo

- Vmware, QEMU, VirtualPC, VirtualBox

vmware®



QEMU
open source processor emulator

Próximo: paravirtualização >>



✓ (2) Paravirtualização

- Recorre a uma camada intermédia entre o SO hóspede e o hardware
- O SO hóspede precisa de ser modificado
- Apenas viável para SO de código aberto (Linux, etc.)
- Não é viável para SO da família Windows (devido a motivos de licenciamentos, tecnologicamente a paravirtualização é viável)

✓ Exemplo

- Xen

Próximo: virtualização assistida por *hardware*>>

- ✓ (3) Virtualização assistida por hardware
 - Tanto a INTEL como a AMD providenciam suporte para virtualização na plataforma x86
 - AMD implementa o AMD-V (*pacífica*)
 - INTEL implementa o VT-x

Software *containers* (#1)

- As máquinas virtuais sistemas necessitam de uma quantidade apreciável de recursos

- Software

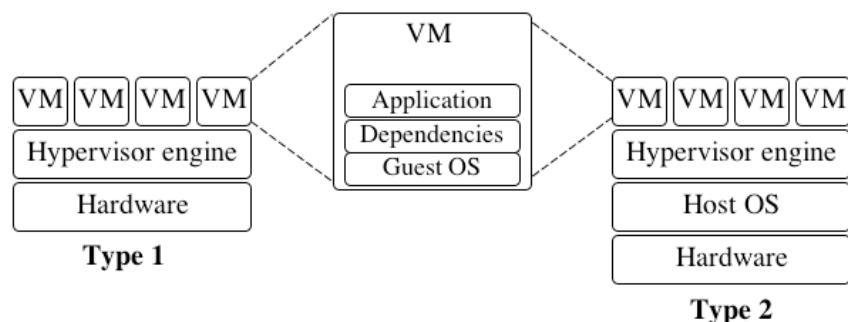


- Hipervisor para funcionamento



- SO hospedeiro

- Espaço disco + memória RAM



- Exemplo

- VM empregue em SO

- Vários GiB de disco

- Espaço em disco para instalação do Ubuntu

- 1 GiB de RAM

- Vantagens máquinas virtuais Sistema

- Isolamento do software que corre na máquina virtual

- Não deve interferir com supervisor/SO hospedeiro

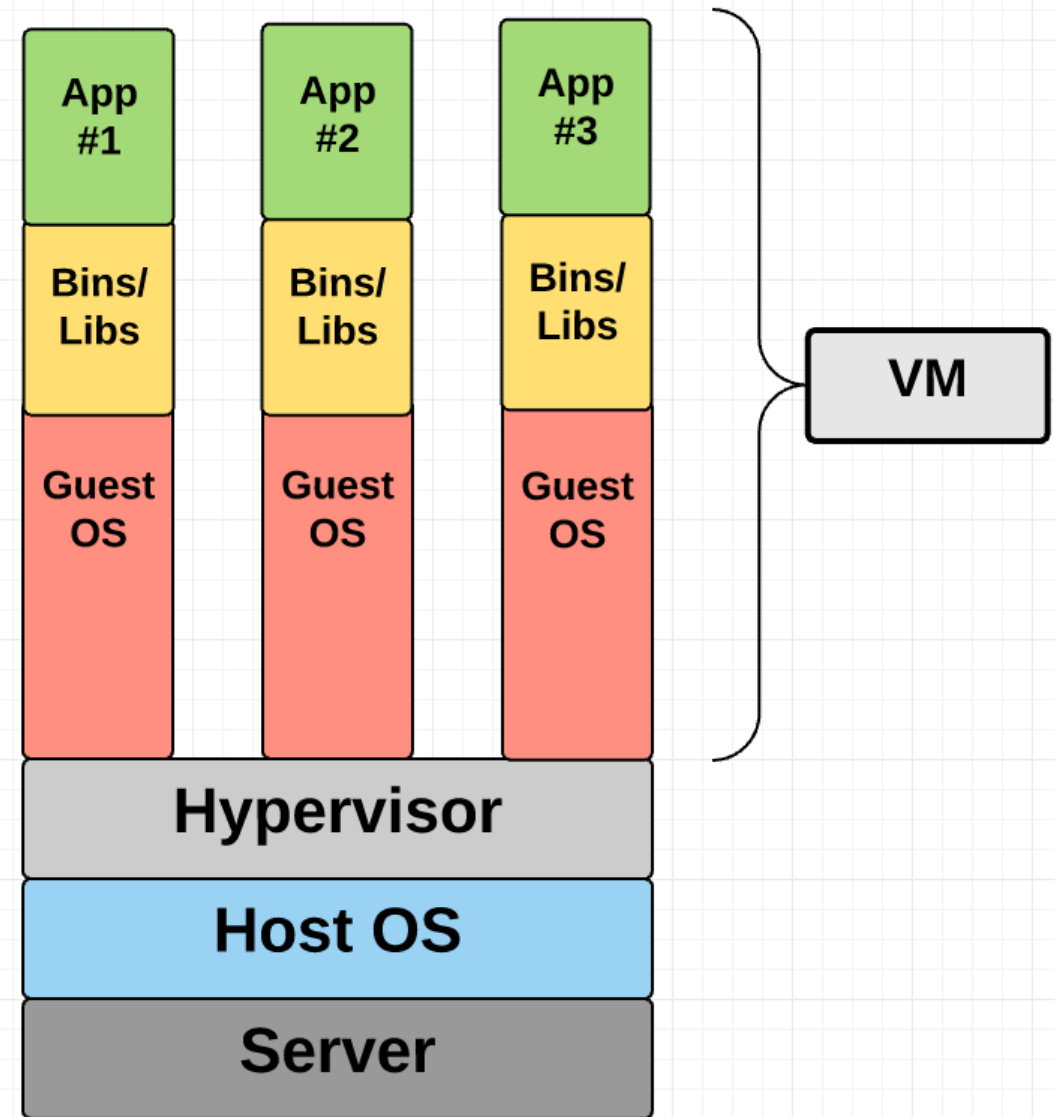
- Solução alternativa mais sóbria em recursos: *containers*

Software *containers* (#2)

- Os *containers* (contentores) têm objetivos similares às VM
 - Isolamento de aplicações e das respetivas dependências num item auto-contido que pode executar em qualquer sistema
 - Tanto as VMs como os *containers* eliminam a necessidade de uma máquina dedicada
- Principal diferença entre VM sistema e *containers*
 - Container: foco está na aplicação
 - VM: foco está em emular máquina física
- VM sistema
 - emula computador real
 - Corre por cima de hipervisor
 - Hipervisor pode correr por cima do hardware (*bare-metal*) ou por cima de SO hospedeiro

Software *containers* (#3)

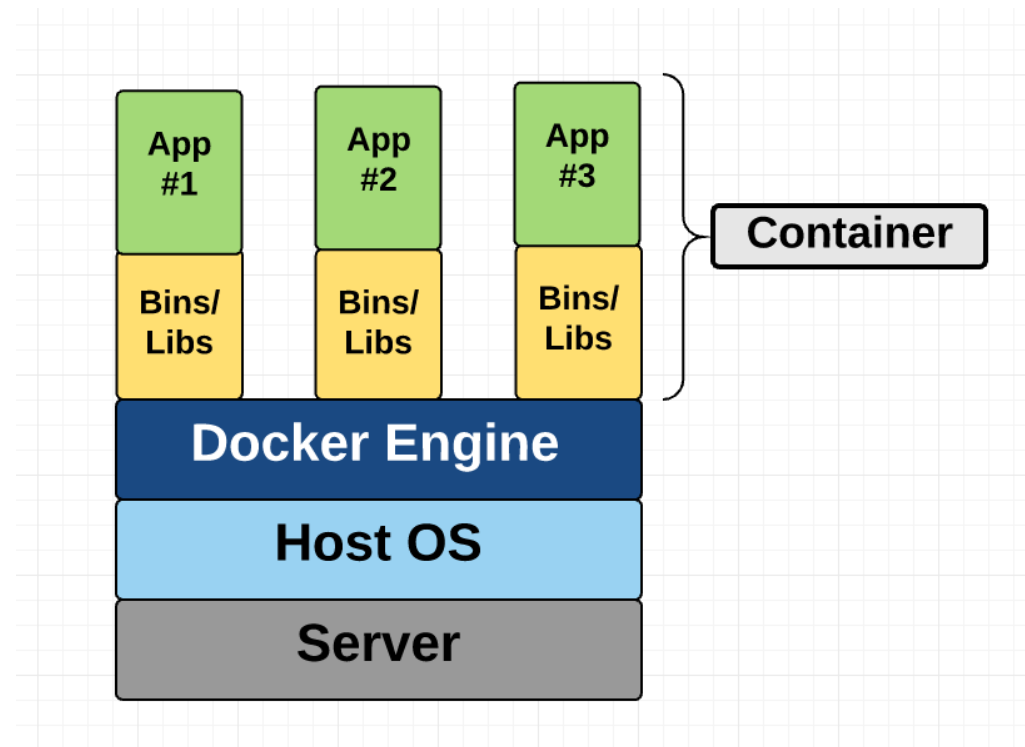
- Máquina virtual sistema
 - Emula computador real
 - Corre por cima de hipervisor
 - Hipervisor pode correr
 - diretamente por cima do hardware (*bare-metal hypervisor*)
 - ou por cima de SO hospedeiro (*guest hypervisor*)



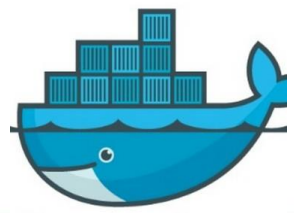
<https://bit.ly/2tYeZyX>

Software *containers* (#4)

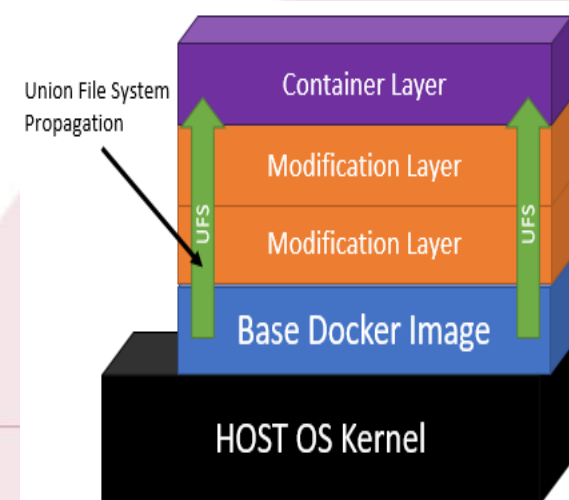
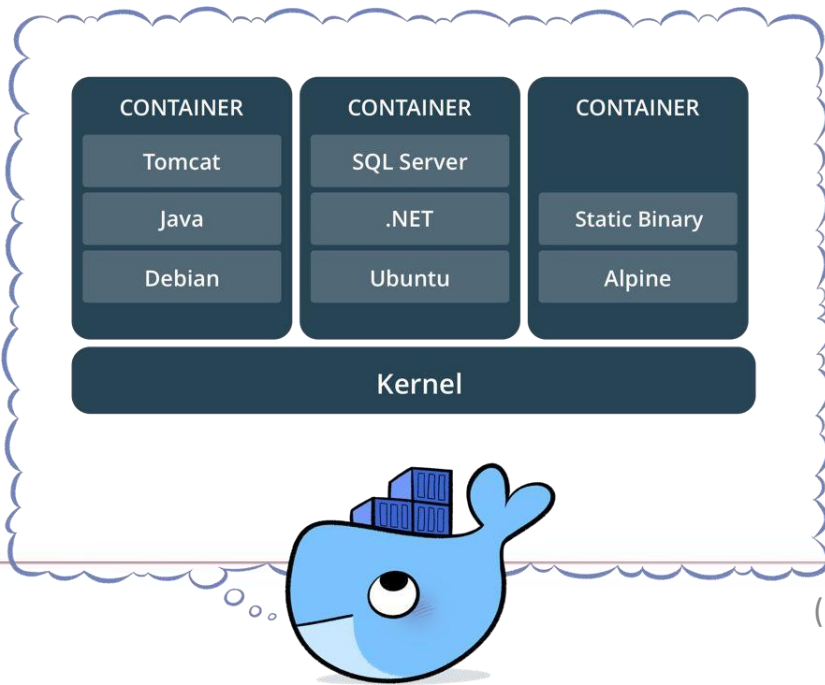
- Container
 - Abstração do espaço do utilizador
 - Os *containers* que executam num dado sistema **partilham** o SO hospedeiro
 - Poupança de recursos, nomeadamente de espaço em disco
 - Isolamento de aplicações
 - Para o utilizador, os containers aparentam-se com VM
 - Podem executar comandos como root, tem um espaço privado para execução, interface de rede própria, endereço IP

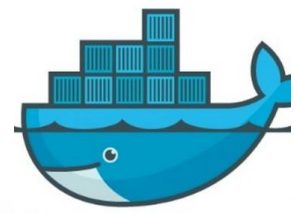


<https://bit.ly/2tYeZyX>

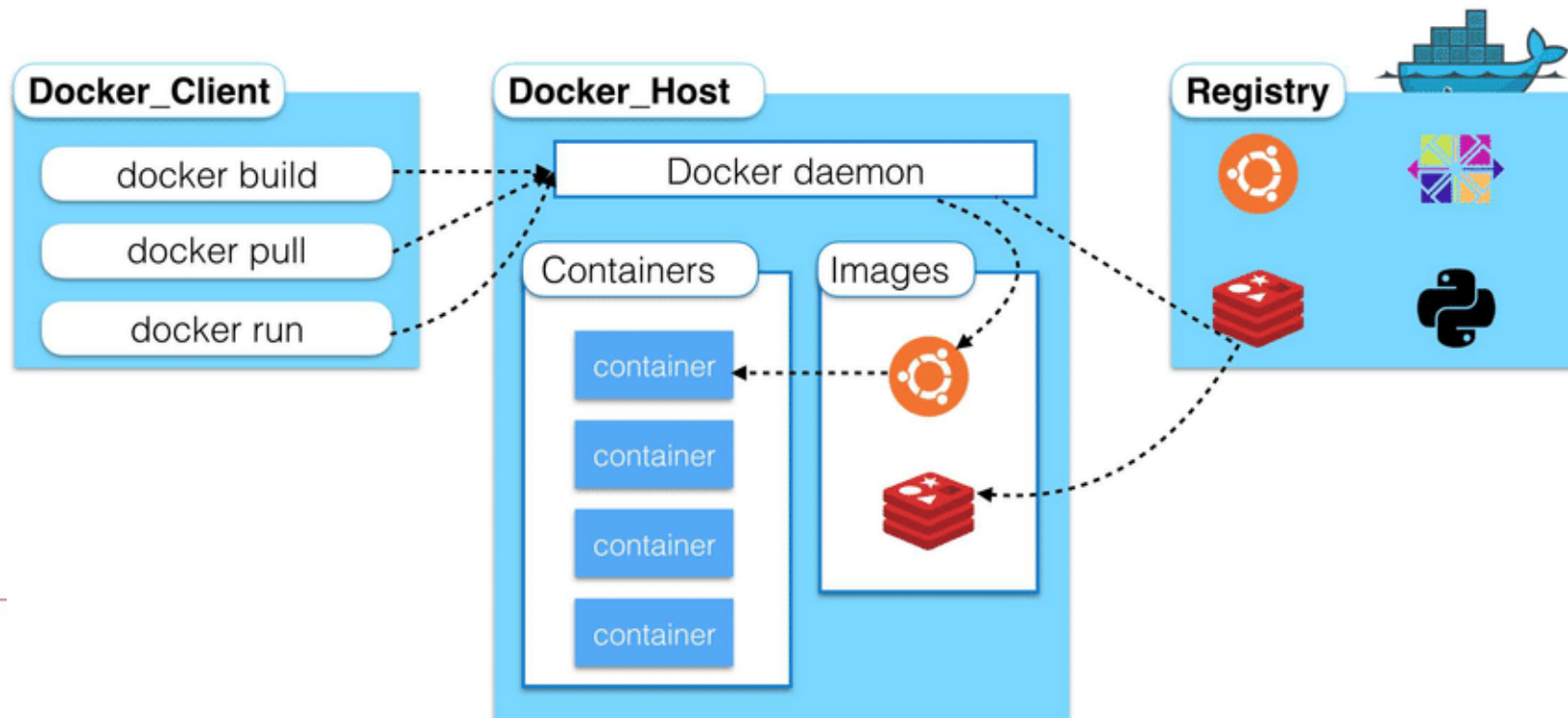


- Docker
 - Projeto código aberto
 - Originalmente para sistemas operativos Linux
 - Recorre a serviços do SO para criar ambiente de contentores por cima do SO
- Vantagens do docker
 - Facilidade de uso
 - Conjunto de comandos
 - Velocidade
 - Uso de pouco recursos
 - Existência de repositório de contentores
 - Docker hub que contém imagens docker
 - Modularidade e escalabilidade





- Arquitetura docker
 - Docker_client
 - Interage com docker_daemon
 - Docker_host
 - Armazena imagens
 - Executar contentores
- (continua)
 - Registry
 - Serviço de repositório de imagens
 - Existem repositórios públicos (<https://hub.docker.com/>)
 - Também são suportados repositórios privados
 - Aplicações da empresa, etc.



Docker (#3)

- Diferença entre...
 - Imagem
 - e
 - Contentor
- Imagem
 - Ficheiro ou conjunto de ficheiros que quando executados no ambiente docker dão origem a um contentor
 - Representa uma aplicação
 - E.g.: servidor de bases de dados + servidor HTTP
- (continua)
 - Contentor
 - Imagem instanciada que se encontra em execução
 - A partir de uma mesma imagem é possível criar N contentores...

Docker – exemplo (#1)

- Não é recomendado instalar no Windows 10
 - Requer Hyper-V que é incompatível com virtualbox e vmware...
- Pode ser instalado...na máquina virtual da UC
 - Recomenda-se que usem uma VM só para esse efeito
- Instalar no lubuntu

```
sudo apt-get install docker
```
- Formato dos comandos

```
sudo docker COMANDO
```
- Exemplos

```
sudo docker run hello-world
```

```
user@ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d8aec4eeb95f: Pull complete
Digest: sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb77324b0fe535
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

```
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. (i386)
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/
```

Docker – exemplo (#2)

✓ Exemplo

– Uso da imagem Alpine

- Alpine: micro-distribuição de Linux

```
sudo docker run --interactive --tty alpine
```

```
user@ubuntu:~$ sudo docker run --interactive --tty alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
bcb4f889a459: Pull complete
e4099db9edcf: Pull complete
Digest: sha256:b3dbf31b77fd99d9c08f780ce6f5282aba076d70a513a8be859d8d3a4d0c92b8
Status: Downloaded newer image for alpine:latest
/ # ps aux
PID   USER     TIME   COMMAND
    1  root      0:00   /bin/sh
     6  root      0:00   ps aux
/ # whoami
root
/ # █
```

Docker – alguns comandos

- Executar como root
 - docker images
 - Lista imagens que existem localmente
 - docker ps
 - Lista contentores que estão a executar na máquina local
 - docker pause IDcontentor
 - IDcontentor= ac259e20f9a5
 - docker unpause IDcontentor
 - docker stop IDcontentor
 - docker restart IDcontentor
 - docker attach IDContentor
- (continua)
 - docker rm IDcontentor
 - Termina contentor
 - docker rmi imagem
 - Remove cópia local de *imagem*
 - docker run -p 8080:80 container
 - porto local:porto contentor
 - docker --help

1 - 19 of 19 available images.

X x86-64

X Programming Languages

X Official Image



golang

Updated 23 minutes ago

Go (golang) is a general purpose, higher-level, imperative programming language.

Container

Linux

Windows

IBM Z

PowerPC 64 LE

386

ARM 64

ARM

x86-64



python

Updated 23 minutes ago

Python is an interpreted, interactive, object-oriented, open-source programming language.

Container

Windows

Linux

IBM Z

x86-64

PowerPC 64 LE

386

ARM 64

ARM

- ✓ Capítulo 2 de “Operating Systems – Internals and Design Principles”, William Stallings, Prentice Hall, 9ª edição, 2017
- ✓ Docker documentation,
<https://docs.docker.com/>

