

# Containerización con Docker

Coruña Abril-2016

Francisco Maseda Muiño



# Containerización: en cifras

## ▶ Google

- «Everything in Google runs in a container»
- Infraestructura de aproximadamente un millón de servidores físicos
- 2000 millones de containers creados a la semana

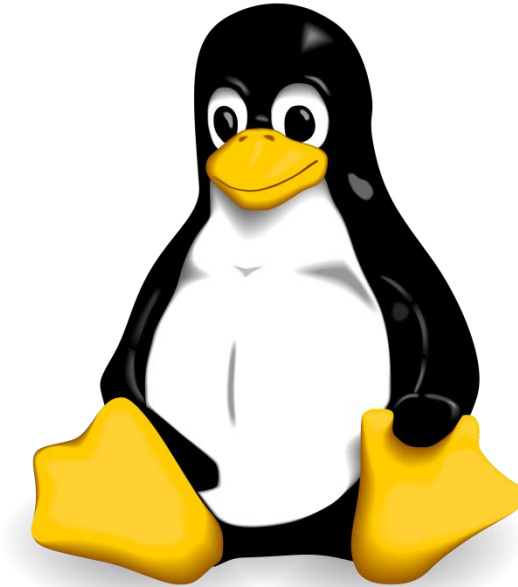
## ▶ Microsoft

- Anuncia su soporte oficial para Docker en Azure

## ▶ Iniciativa OpenContainer

- Google, Amazon AWS, Facebook, Dell, IBM, Intel, RedHat...

# Sistema Operativo



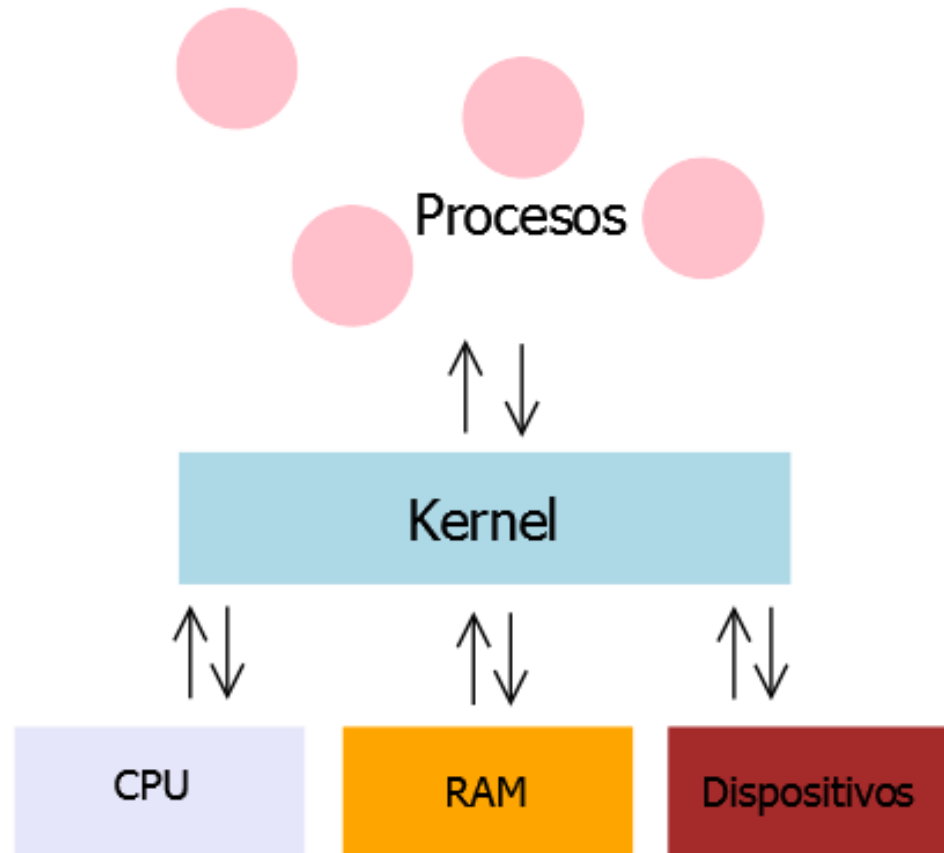
*“When you’re ready, the right operating system will appear in your life “*

Guy Kawasaki

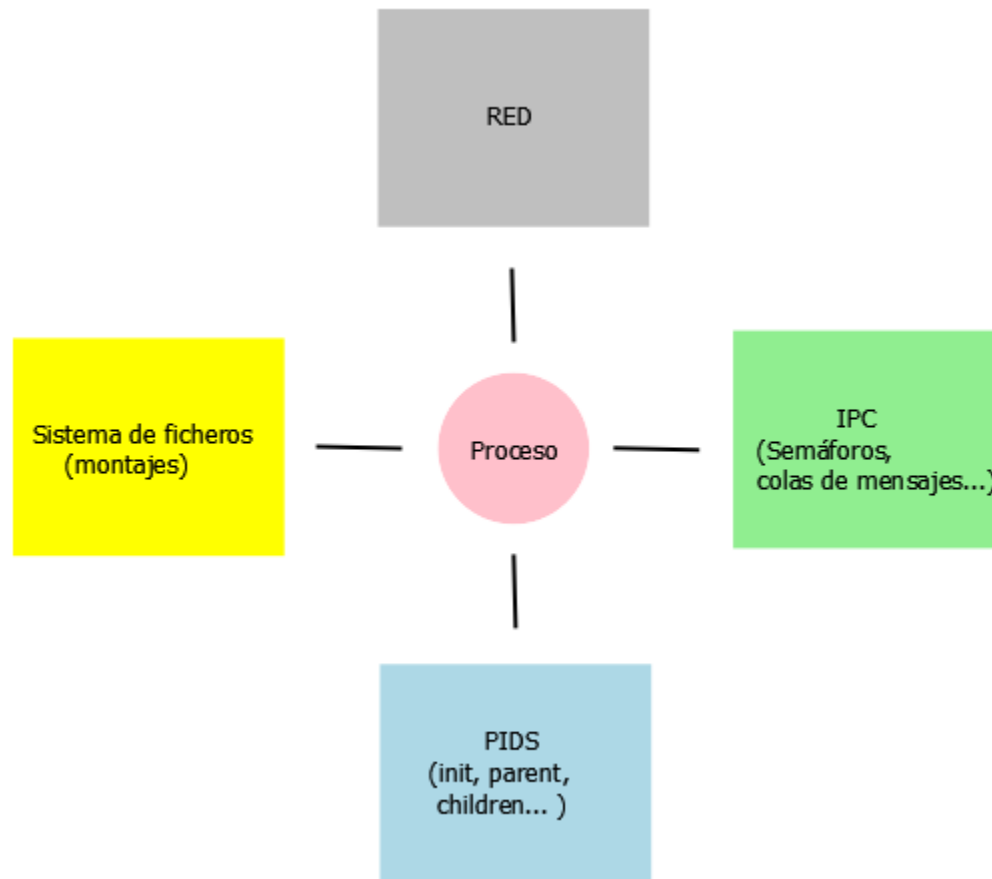
# SO: Una visión clásica

- ▶ Misiones fundamentales
  - Abstracción de recursos (hardware...)
  - Aislamiento de programas (procesos) y de sus accesos a los recursos del sistema

# SO: Una visión clásica

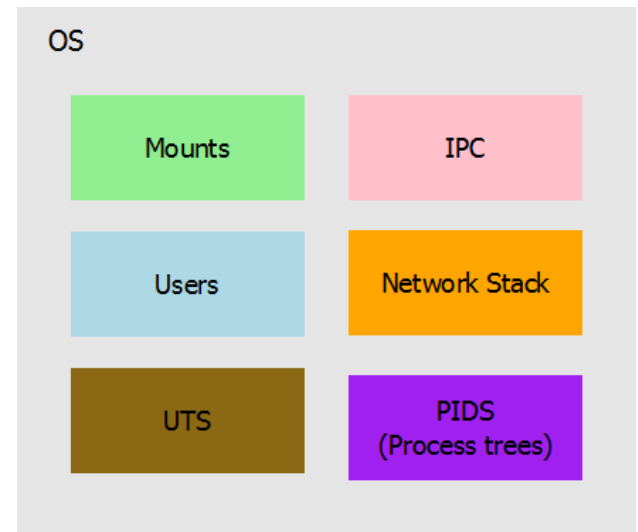


# SO: Una visión clásica



# SO: ¿Cumple su misión?

- ▶ Solo parcialmente:
  - ✓ Unidad de ejecución independiente: **Proceso**
  - ✓ Aislamiento de recursos computacionales:
    - ✓ RAM
    - ✓ Ventana de ejecución en CPU
- ❖ Pero:
  - ✗ Siguen siendo globales:
    - ✗ Usuarios
    - ✗ Sistema de ficheros
    - ✗ IPC...



# Virtualización



*"I once heard that Hypervisors are the living proof of Operating System's incompetence"*

Glauber Costa  
Parallels



# VM: Introducción

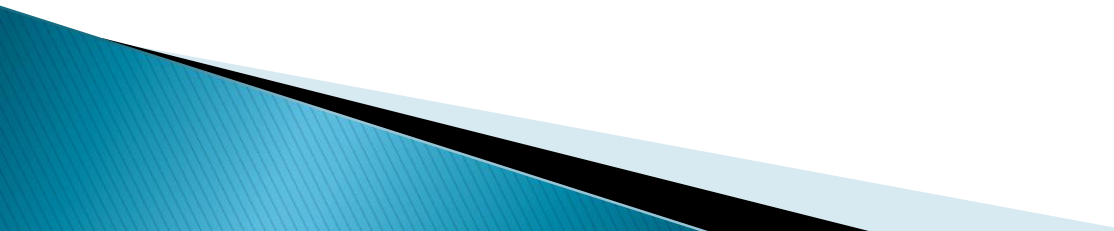
- ▶ Tecnologías que buscan superar las limitaciones:
  - de Hardware:
    - Mejor aprovechamiento de recursos
    - Abstracción y emulación de dispositivos
  - de Software
    - Soluciona el problema de falta de aislamiento del SO

*«Nuestros programas pueden crear servidores»*

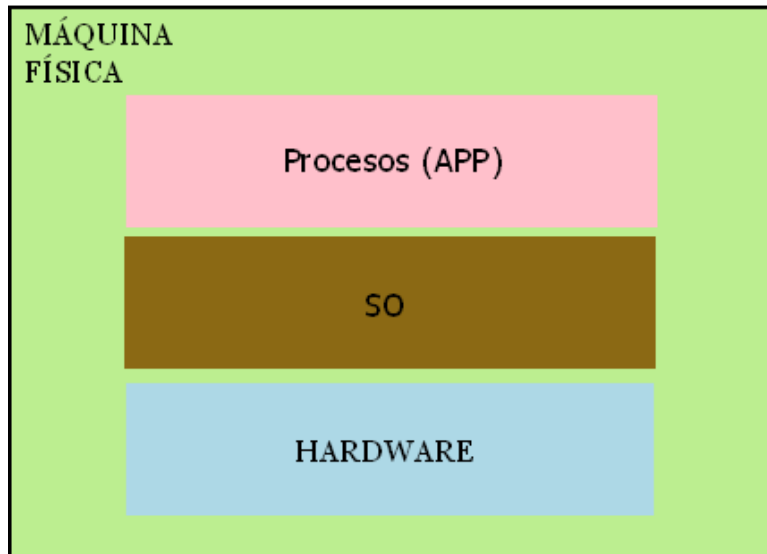
AWS 2007



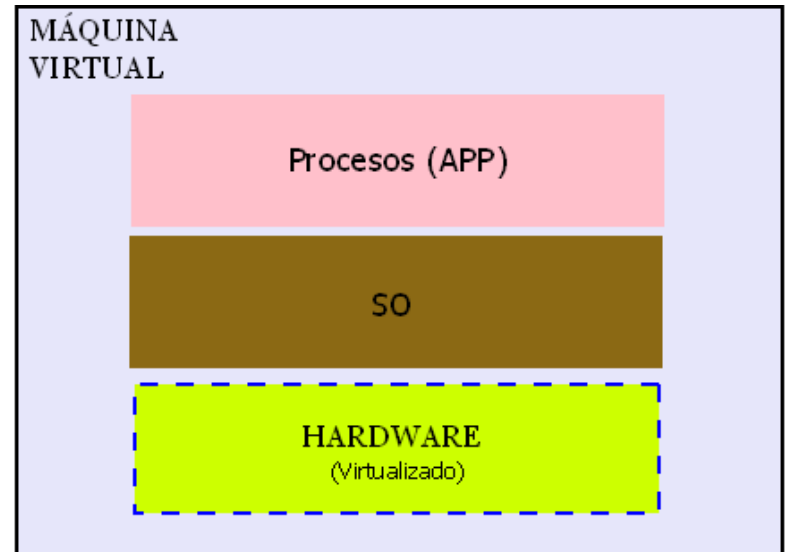
# VM: Concepto

- ▶ Software que emula el hardware de una máquina
  - ▶ Dentro de una máquina física corren varias máquinas virtuales
  - ▶ Diversas técnicas
    - Full virtualización
    - Para-virtualización
    - Virtualización asistida por hardware
- 

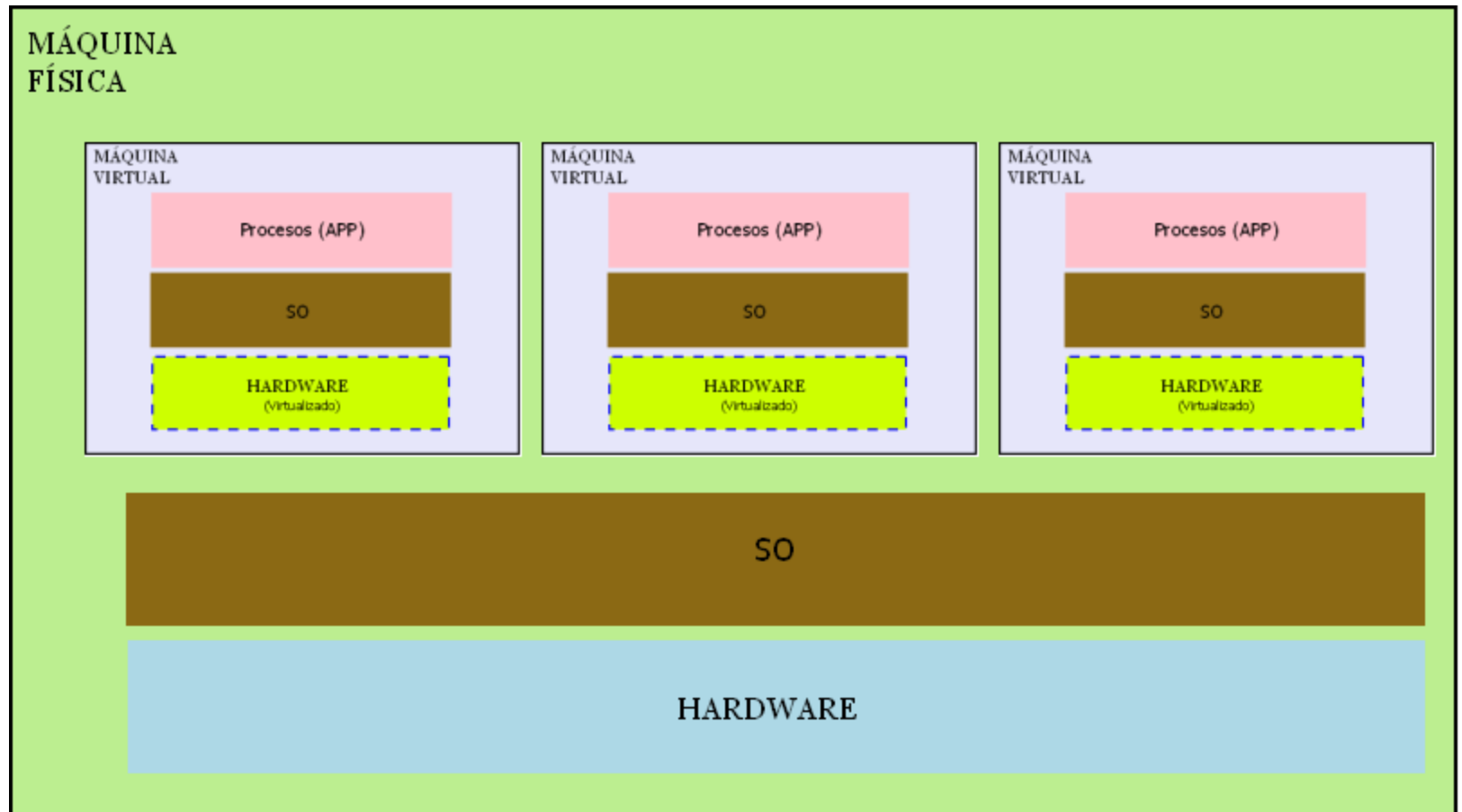
# VM: Concepto



Virtualización



# VM: Concepto



# VM: Problemas

- ▶ Coste en recursos
  - Cada VM necesita emular el hardware y el SO de una máquina física
- ▶ Tiempo de inicio/reinicio
  - Sigue siendo muy alto (media de 2' 30'')

# Containerización



*"Everything in Google runs in a container"*

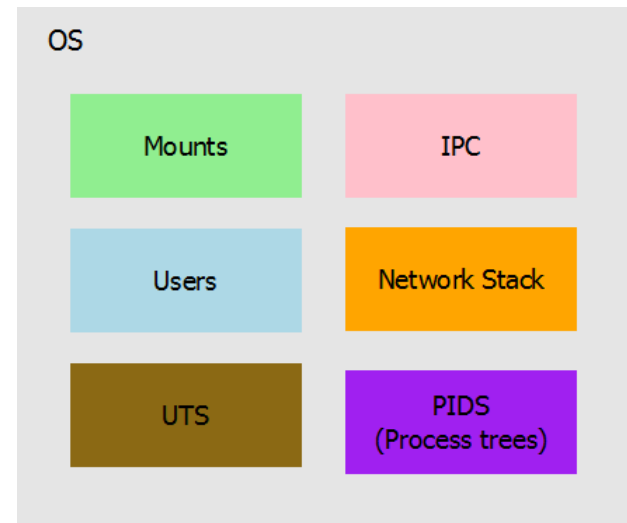
Joe Beda (Ingeniero jefe en Google)

# Containers: Concepto

- ▶ Técnica de virtualización a nivel de SO
  - Un proceso o conjunto de procesos tienen una vista «privada» de una serie de recursos tradicionalmente globales.
  - No se pretende emular una máquina entera sino el **contexto de ejecución** de un proceso: esto es, su **visión del sistema operativo**.

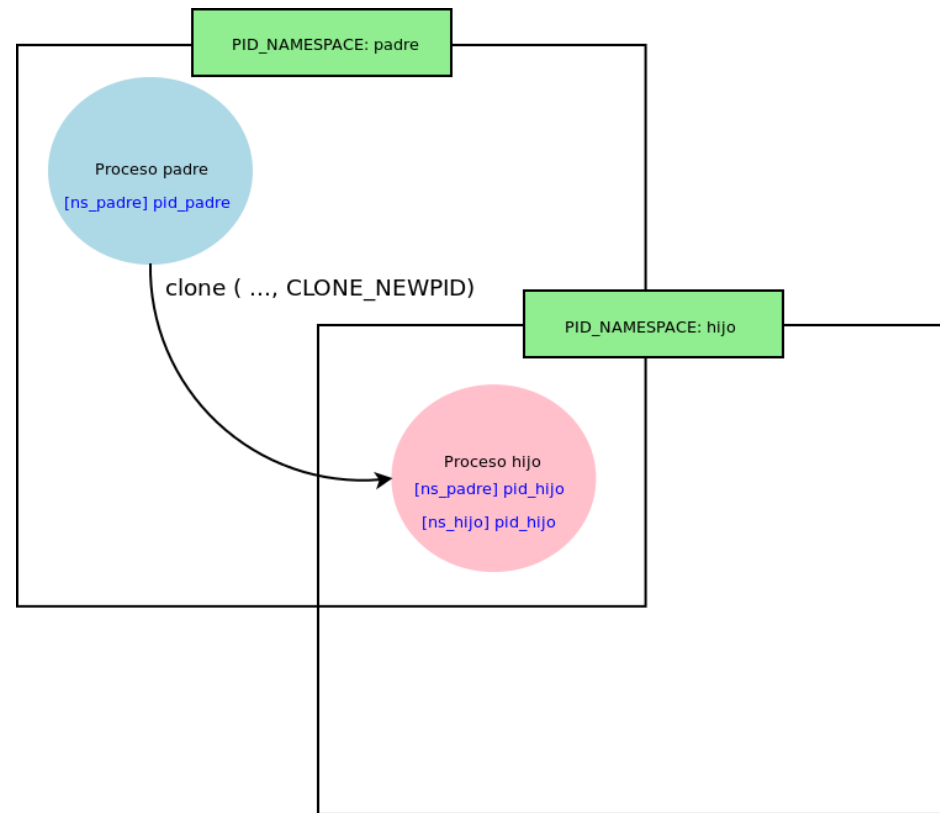
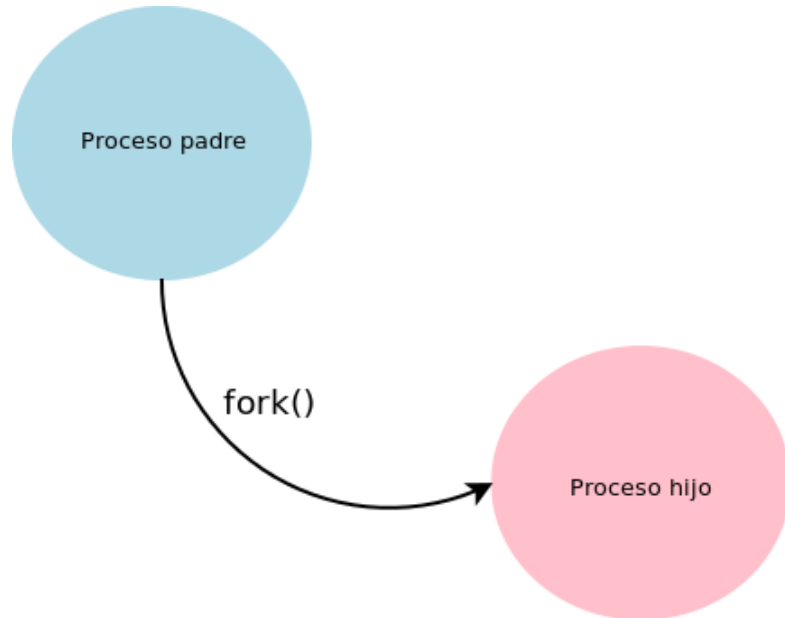
# Containers: Namespaces

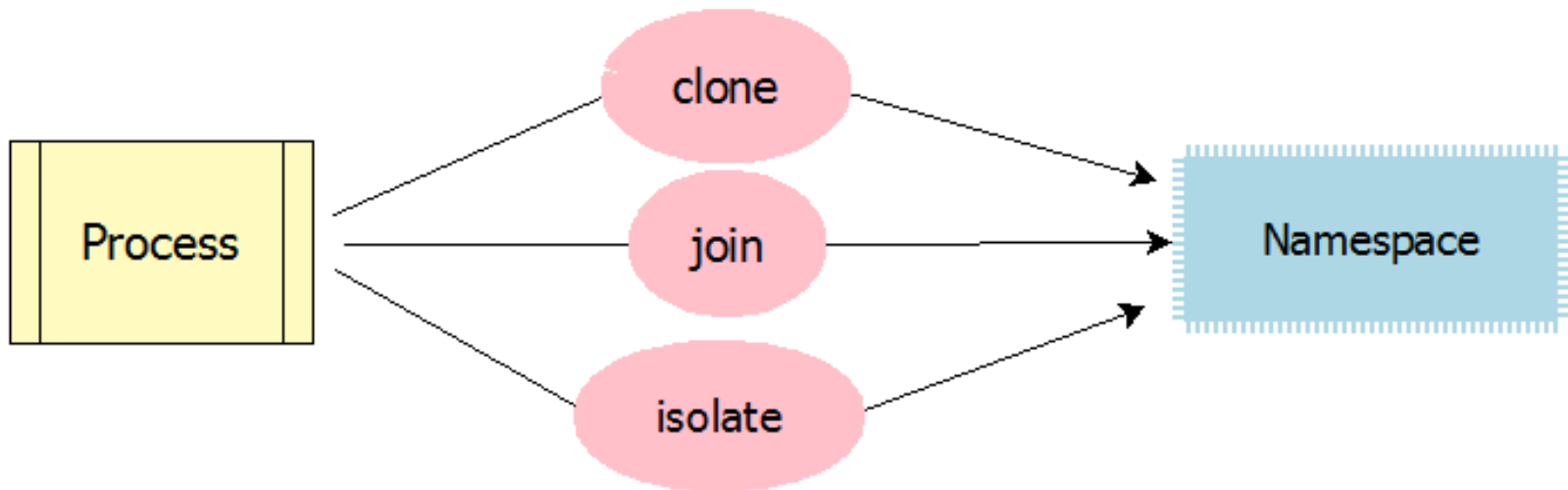
- ▶ La base tecnológica de los containers son los **namespaces**:
  - Se trata de una funcionalidad del Kernel que permite crear vistas privadas de determinados recursos a nivel de proceso





# Containers: Namespaces





## Clone

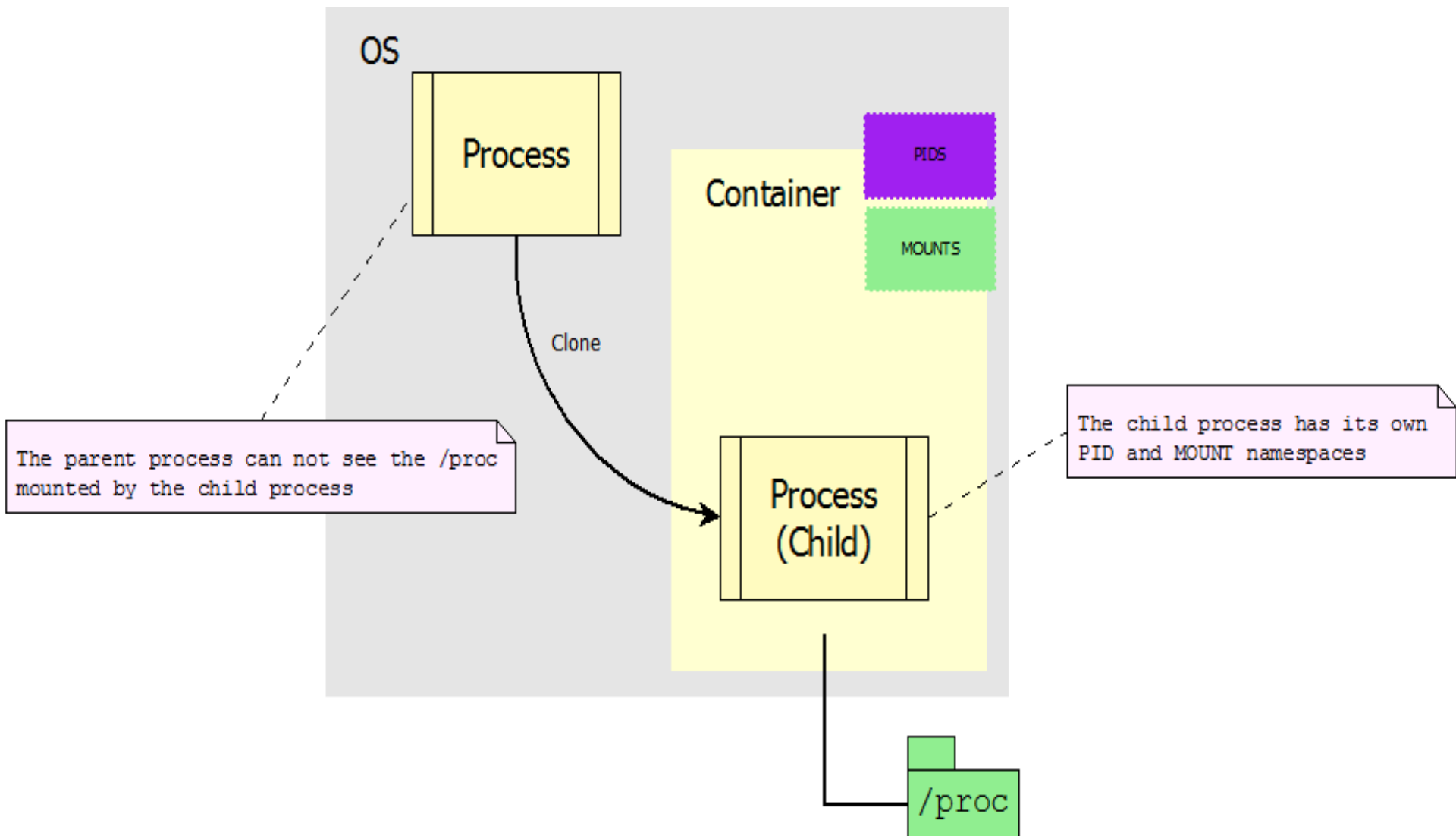
Alternativa a fork. Mediante la syscall **clone**, se pueden crear namespaces privados para el proceso y sus descendientes

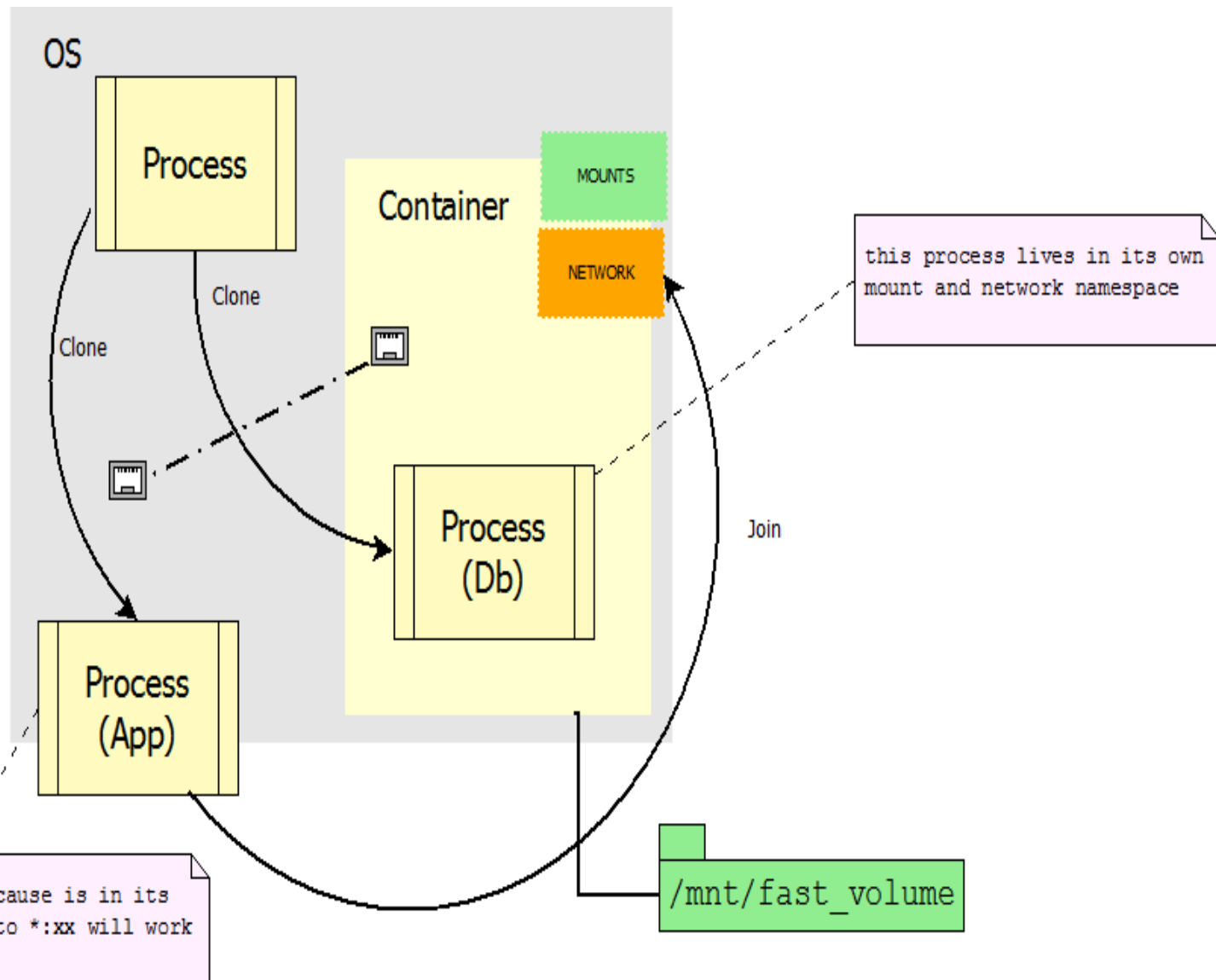
## Join

A través de **setns**, un proceso puede «entrar» en un namespace existente.

## Isolate

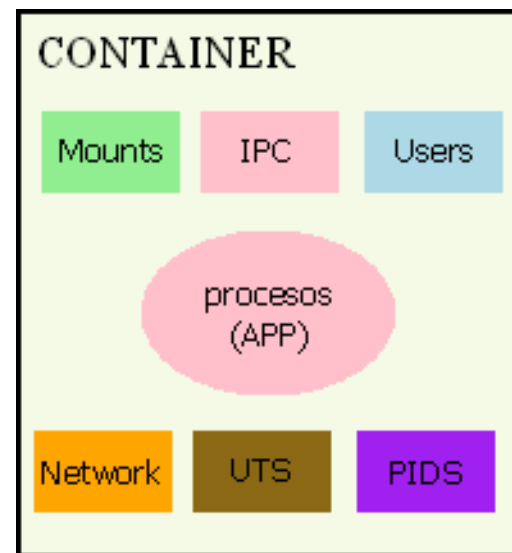
Con la syscall **unshare**, un proceso puede aislar namespaces que esté utilizando mediante su clonado.



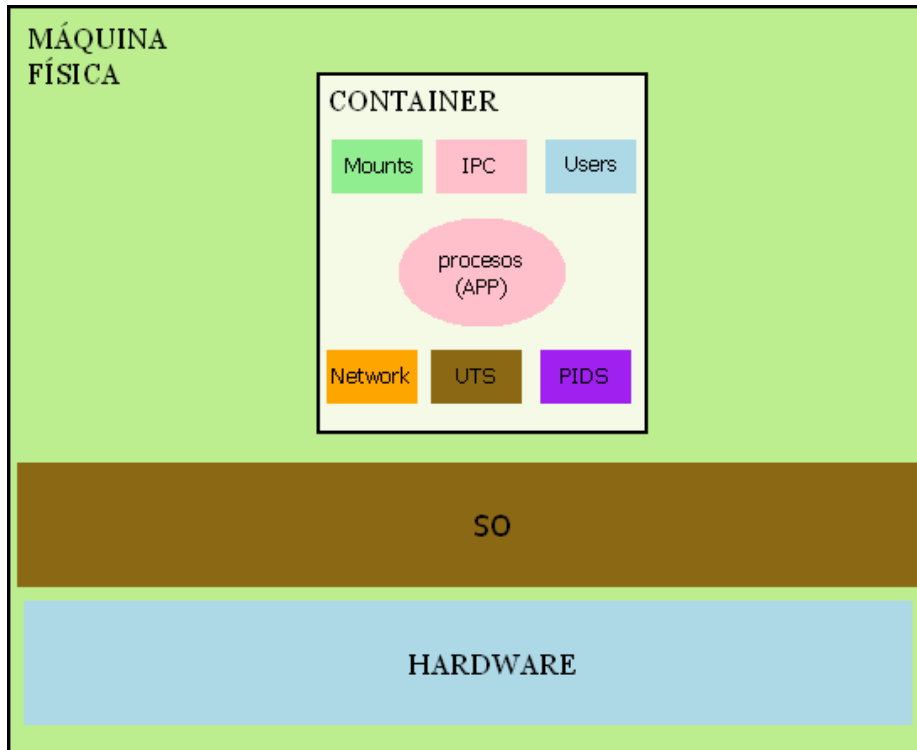


# Containers: uso real

- ▶ Si clonamos los 6 namespaces, el proceso «vive» a casi todos los efectos en su propia «máquina», está, por tanto, containerizado:
  - Puede tener su propia imagen de sistema
  - Crea sus propios usuarios
  - Ve su propio árbol de procesos
  - Tiene su propia stack de red
  - ...

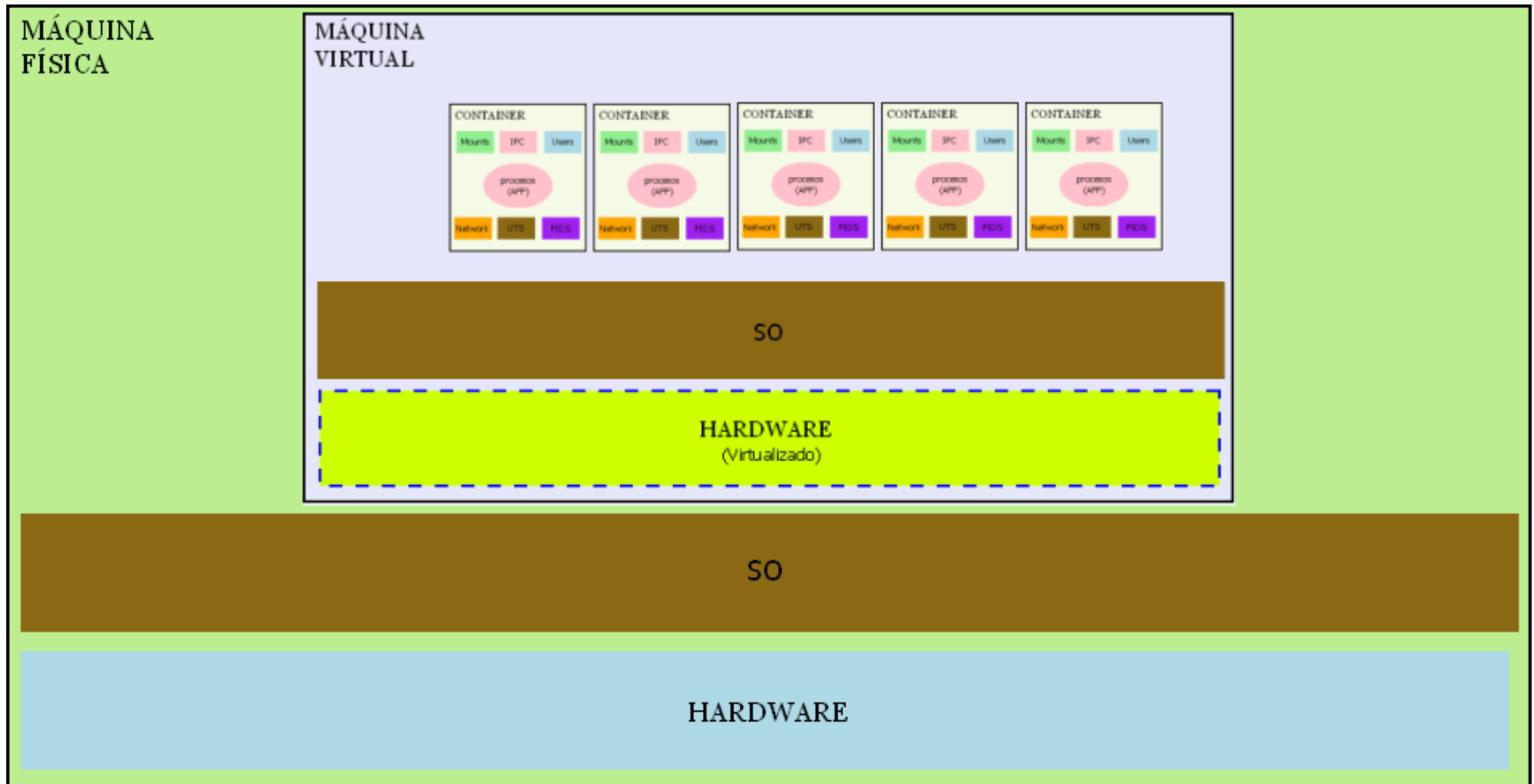


# Containers: uso real

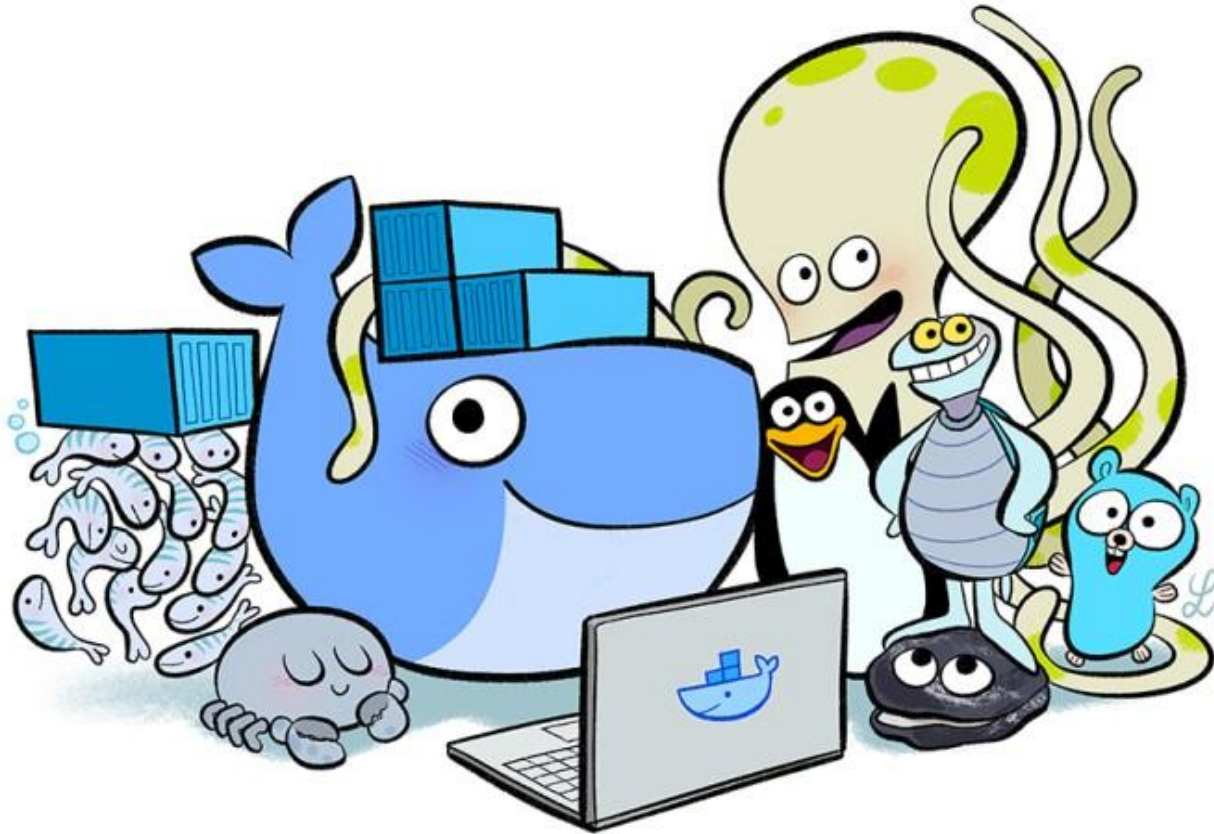


- Ventajas:
  - Ligereza:
    - ✓ No hay que emular hardware
    - ✓ No hay que correr un Kernel
  - Rapidez:
    - ✓ Tiempo de inicio/reinicio (< 1'')

# Containers: uso real



# Docker



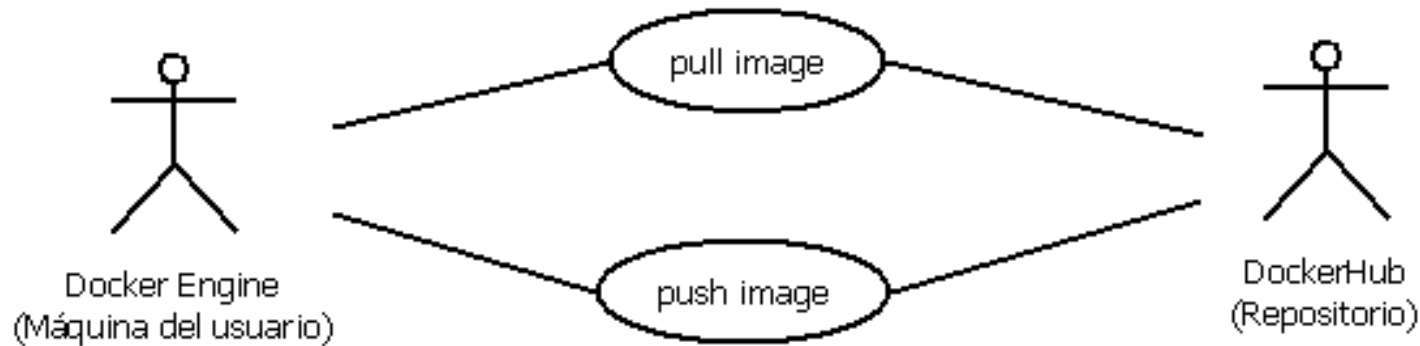


# Docker: Intro

- ▶ Conjunto de herramientas para crear, desplegar y gestionar containers
- ▶ Dos elementos principales:
  - **Imagen**: software (ficheros) que conforman el entorno de un container
  - **Container**: instancia de una imagen que corre uno o varios procesos

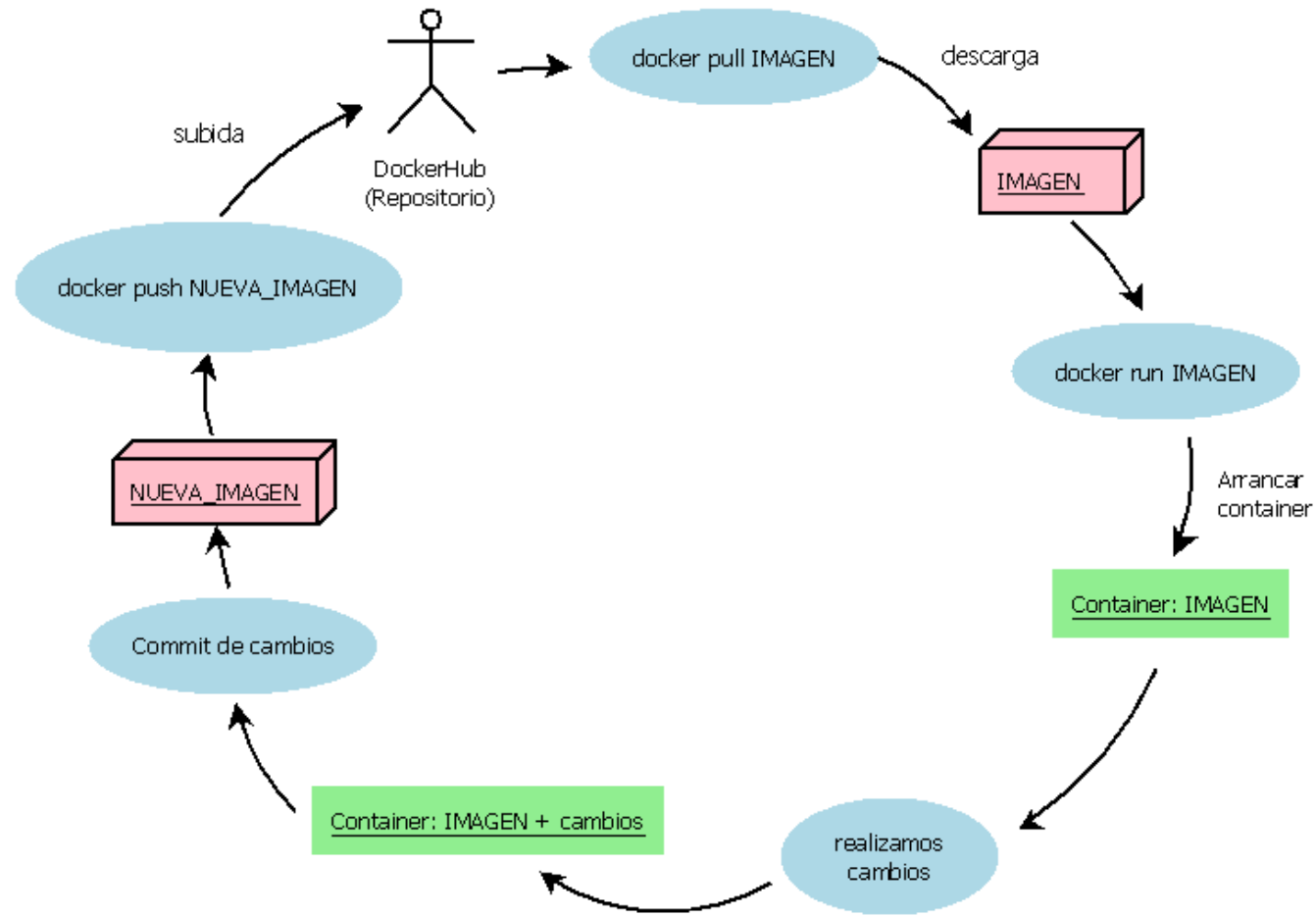
# Docker: imágenes

- ▶ Las imágenes tienen habilidades tipo «git»



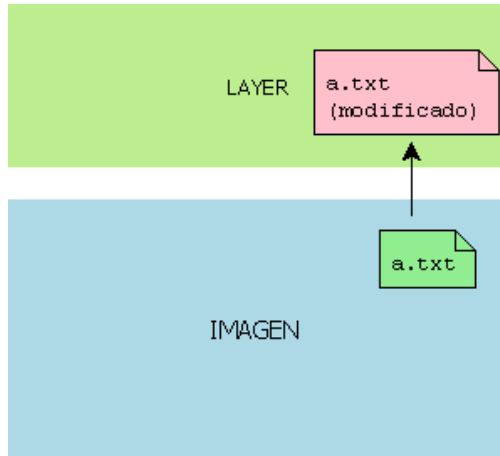
- ▶ Las imágenes se descargan a la máquina anfitrión (donde está el docker-engine) y se almacenan en el sistema de ficheros local

# Docker: Imágenes



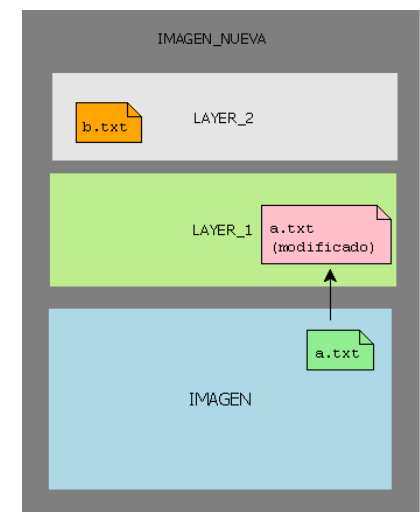
# Docker: Imágenes

Las imágenes son de sólo-lectura



Una modificación de un fichero de la imagen, supone su copia y «subida» a una nueva capa

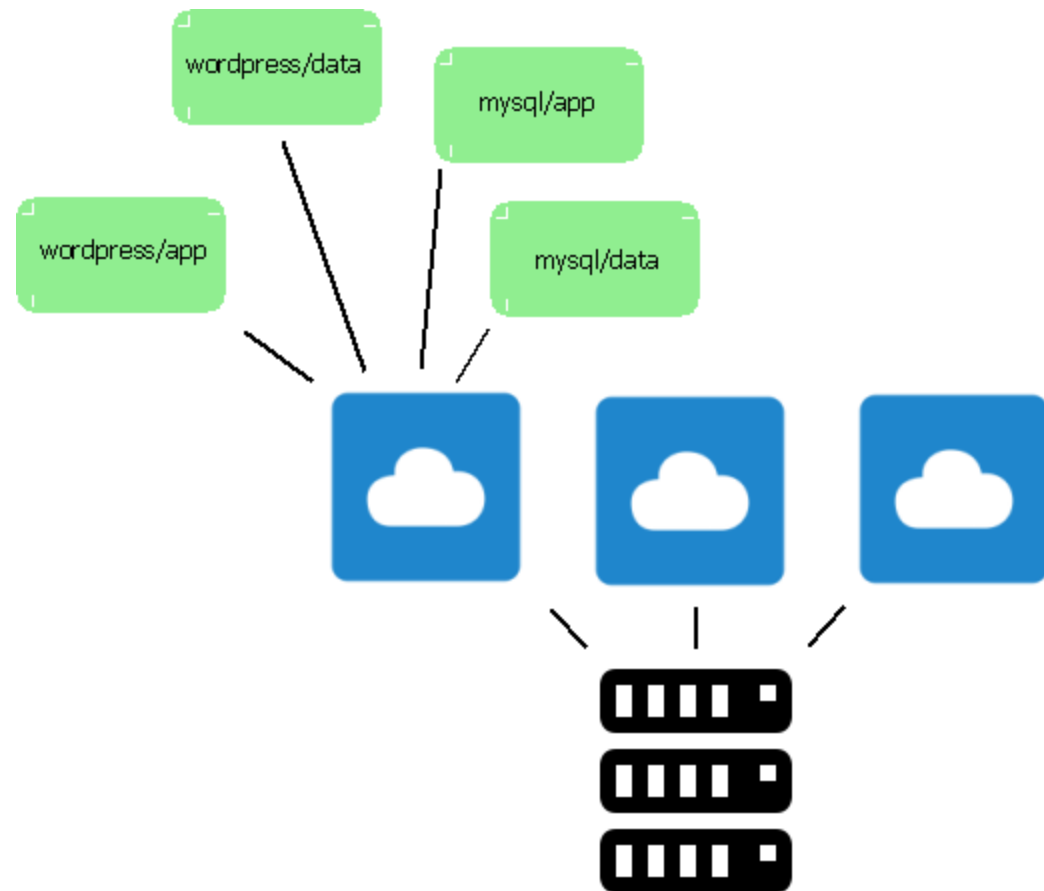
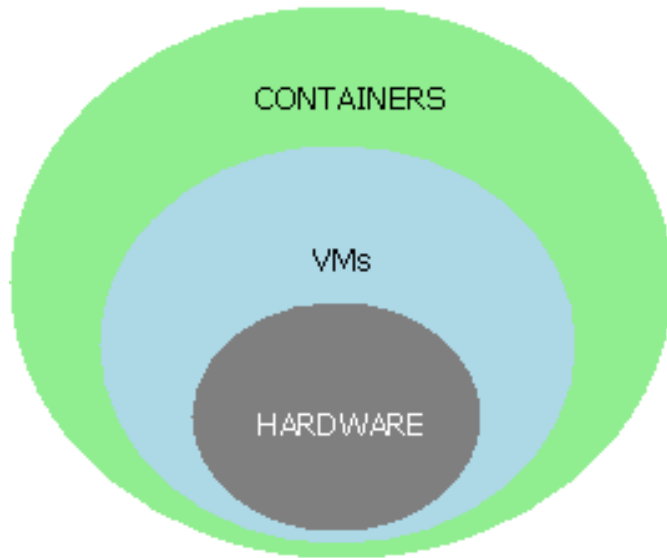
Una imagen es la suma de  $n$  capas, desde una imagen\_base. Cada cambio implica una Nueva capa



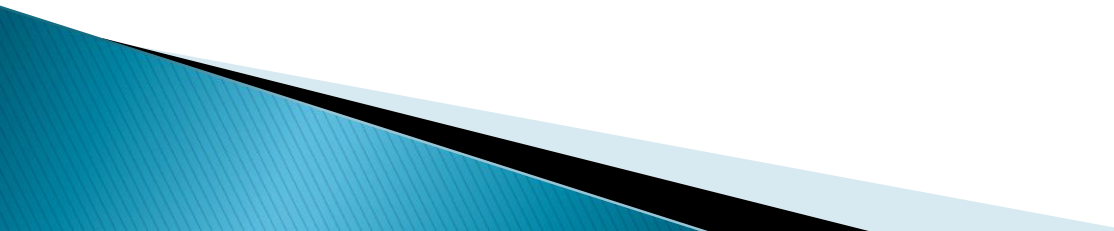
# Conclusiones

- ▶ La containerización no supone una disrupción con respecto a la virtualización, es una continuación
- ▶ Se trata de un nuevo **paradigma** de diseño y despliegue de aplicaciones:
  - Nos centramos en la producción de **imágenes** (base de los contenedores)
  - Nueva unidad de composición: el **container**

# Conclusiones



# Conclusiones

- ▶ ¿Cómo gestionamos un número grande de containers?
  - ▶ ¿Cómo relacionamos unos containers con otros?
  - ▶ ¿Cómo se puede automatizar la generación de imágenes para los containers?
- 

# Muchas gracias

