



Mondragon
Unibertsitatea

Faculty of
Engineering



Contenedores de aplicación

¿ Son realmente seguros ?

1

Introducción





¿Son realmente seguros?

- ¿Son realmente seguros?
 - Pues como casi todo... depende...



¿Son realmente seguros?

- ¿Son realmente seguros?
 - ¿Cuáles son algunos de los vectores de ataque?
 1. Sistema Operativo del *host*
 - “Los contenedores son tan seguros como el sistema operativo del host”
 2. Código de aplicación
 - Aplicación vulnerable
 3. Imagen de contenedor mal configurado
 - Usuario por defecto *root*
 4. *host* donde se crea la imagen del contenedor
 - Alterar la imagen final atacando al *host*
 5. Repositorio de imágenes
 - Reemplazar una imagen por otra



¿Son realmente seguros?

- ¿Son realmente seguros?
 - ¿Cuáles son algunos de los vectores de ataque?
 6. Contenedores mal configurados
 - Contenedor con demasiados privilegios
 7. *host* vulnerable
 - Ejecutar los contenedores en máquinas ejecutando código vulnerable
 8. Secretos expuestos
 9. Red insegura
 10. Escape de contenedores
 11. ...
 - Tal y como se puede observar, hay que tener muchas cosas en cuenta

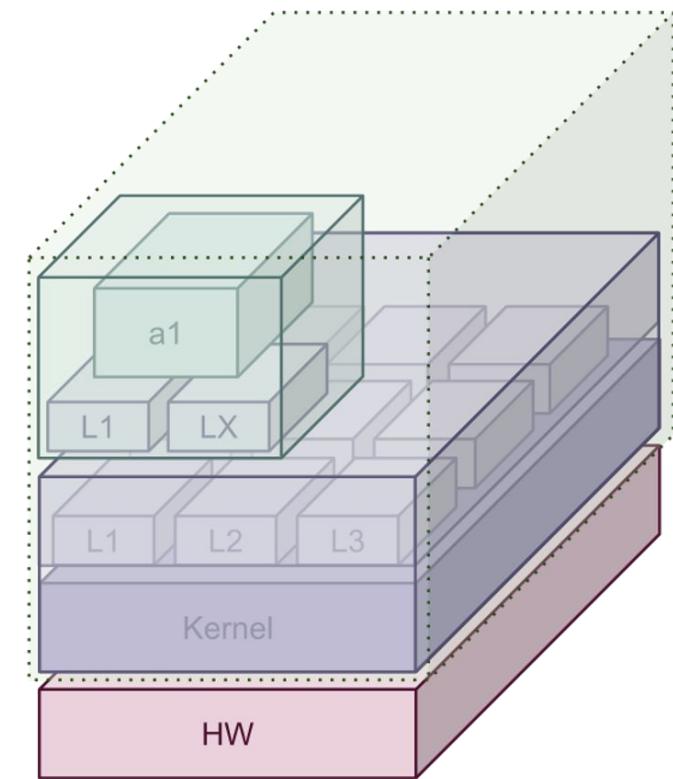
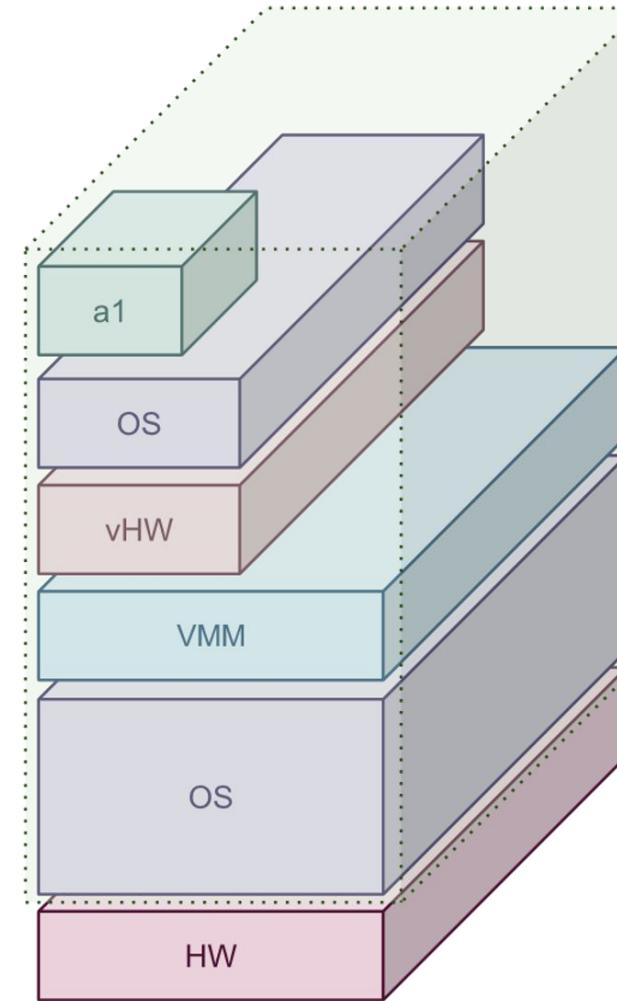
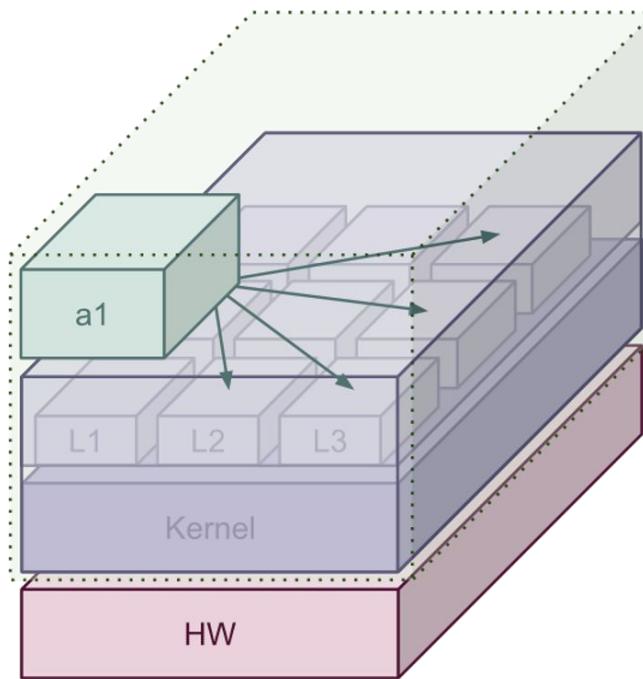




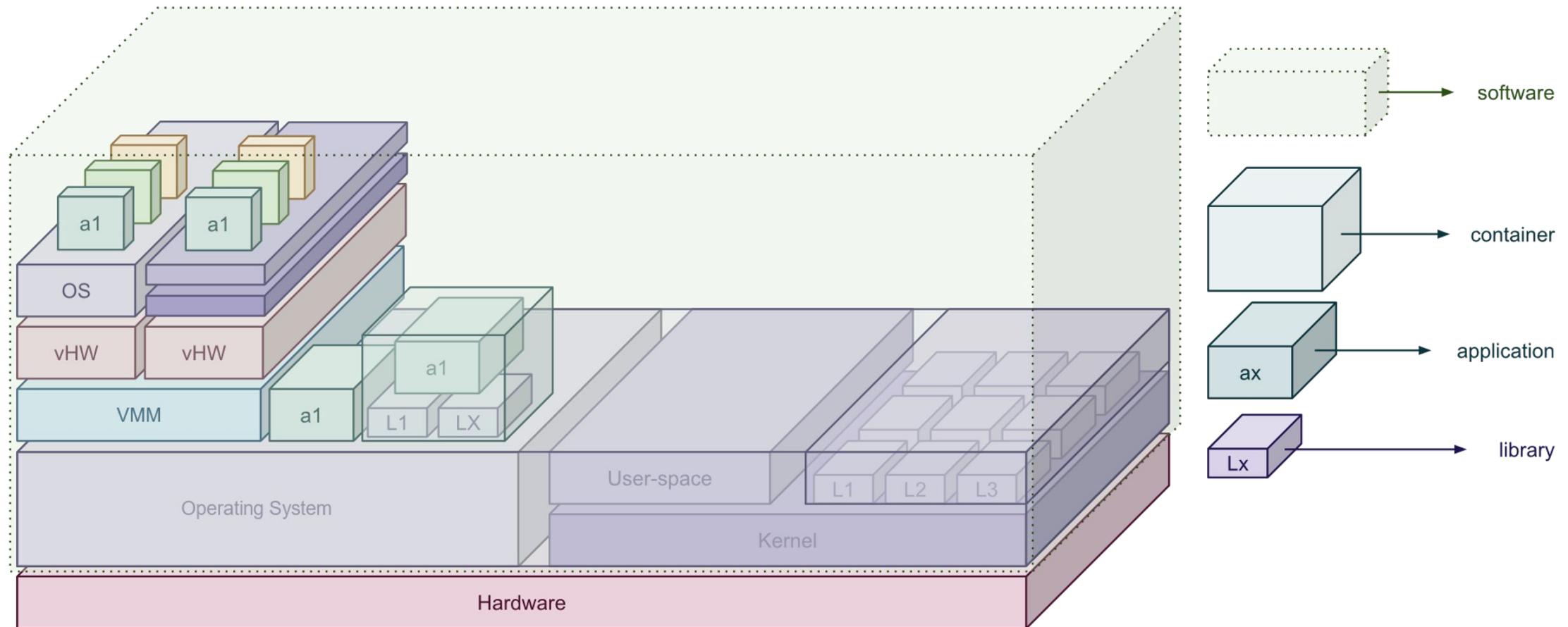
Contenedores vs Máquinas Virtuales

- A menudo los contenedores se confunden con máquinas virtuales
 - Incluso se les denomina "máquinas virtuales ligeras"
 - Pero... ¿Son realmente máquinas virtuales? → pues... **NO**
- ¿En qué se diferencian?

Contenedores vs Máquinas Virtuales

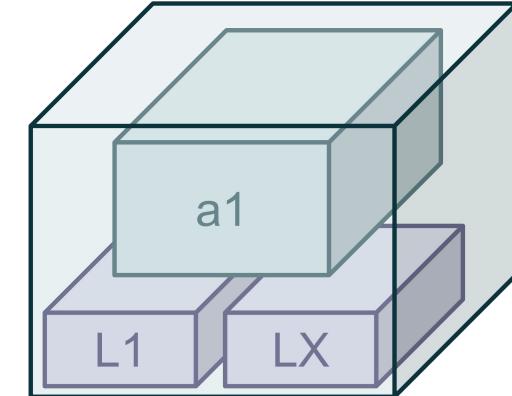


Contenedores vs Máquinas Virtuales



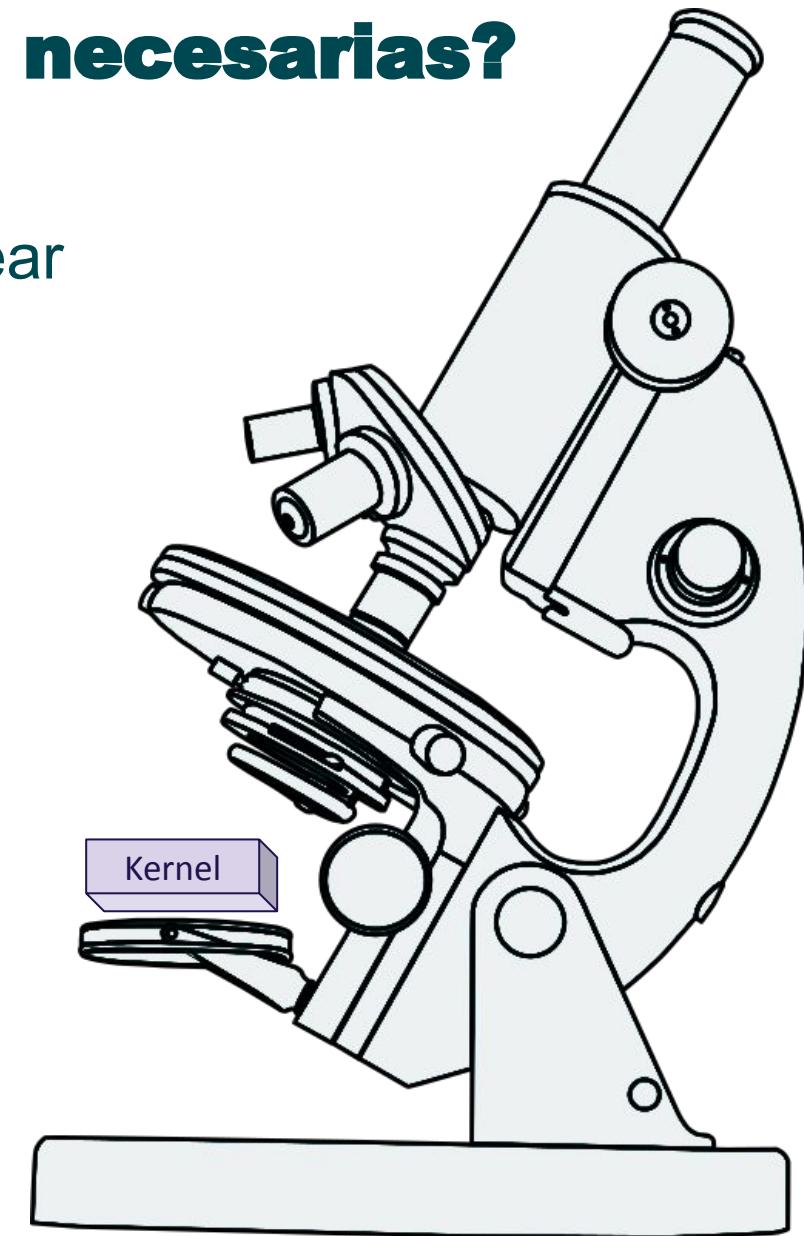
¿Qué es un contenedor?

- Un contenedor es un grupo de procesos
 - Aislados (no necesariamente) del resto de los procesos del sistema en términos de:
 - Usuarios
 - Red
 - Sistema de archivos
 - Identificador de procesos
 - Límites de CPU y memoria



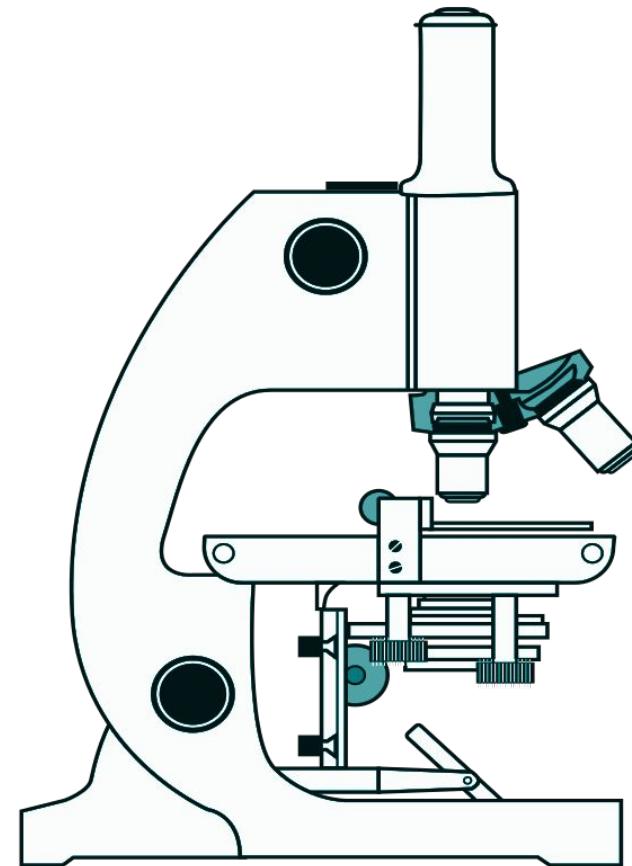
¿Cuáles son las características necesarias?

- “Características” del Kernel que permiten crear contenedores
 - cgroups
 - namespaces
 - capabilities
 - seccomp-bpf
 - chroot
 - Sistemas de archivos copy-on-write



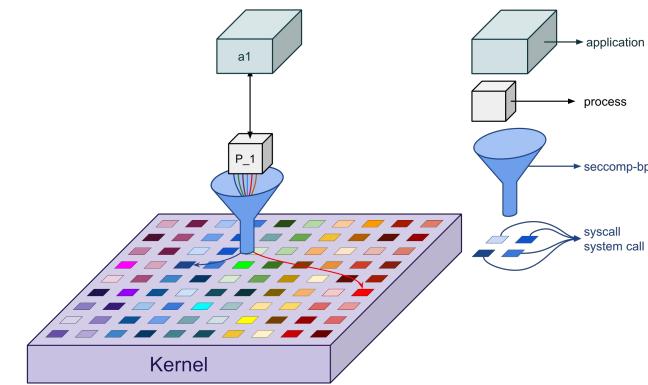
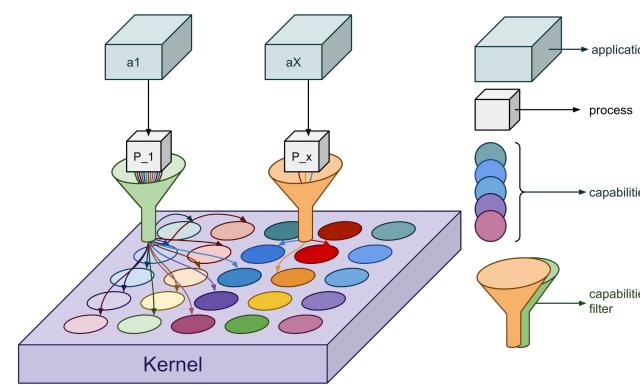
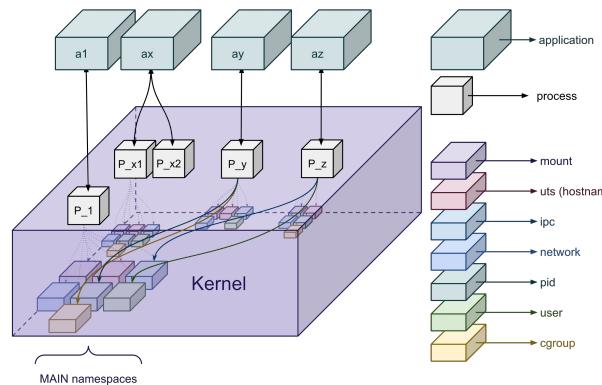
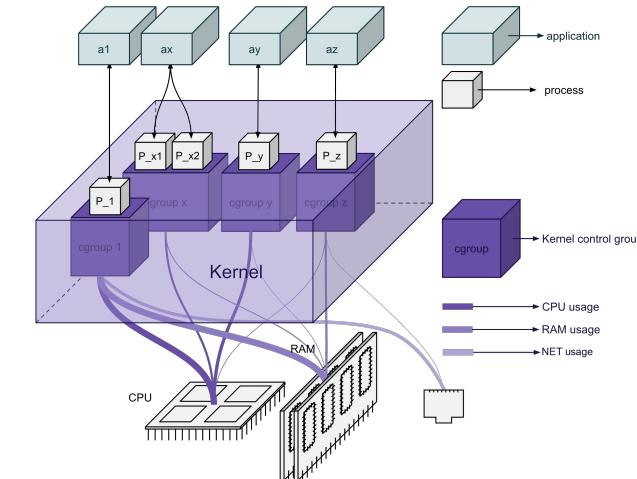
2

Características del Kernel



¿Cuáles son las características necesarias?

- Una mirada más cercana a las siguientes características
 - cgroups
 - namespaces
 - capabilities
 - seccomp-bpf
 - chroot
 - Sistemas de archivos copy-on-write





cgroups

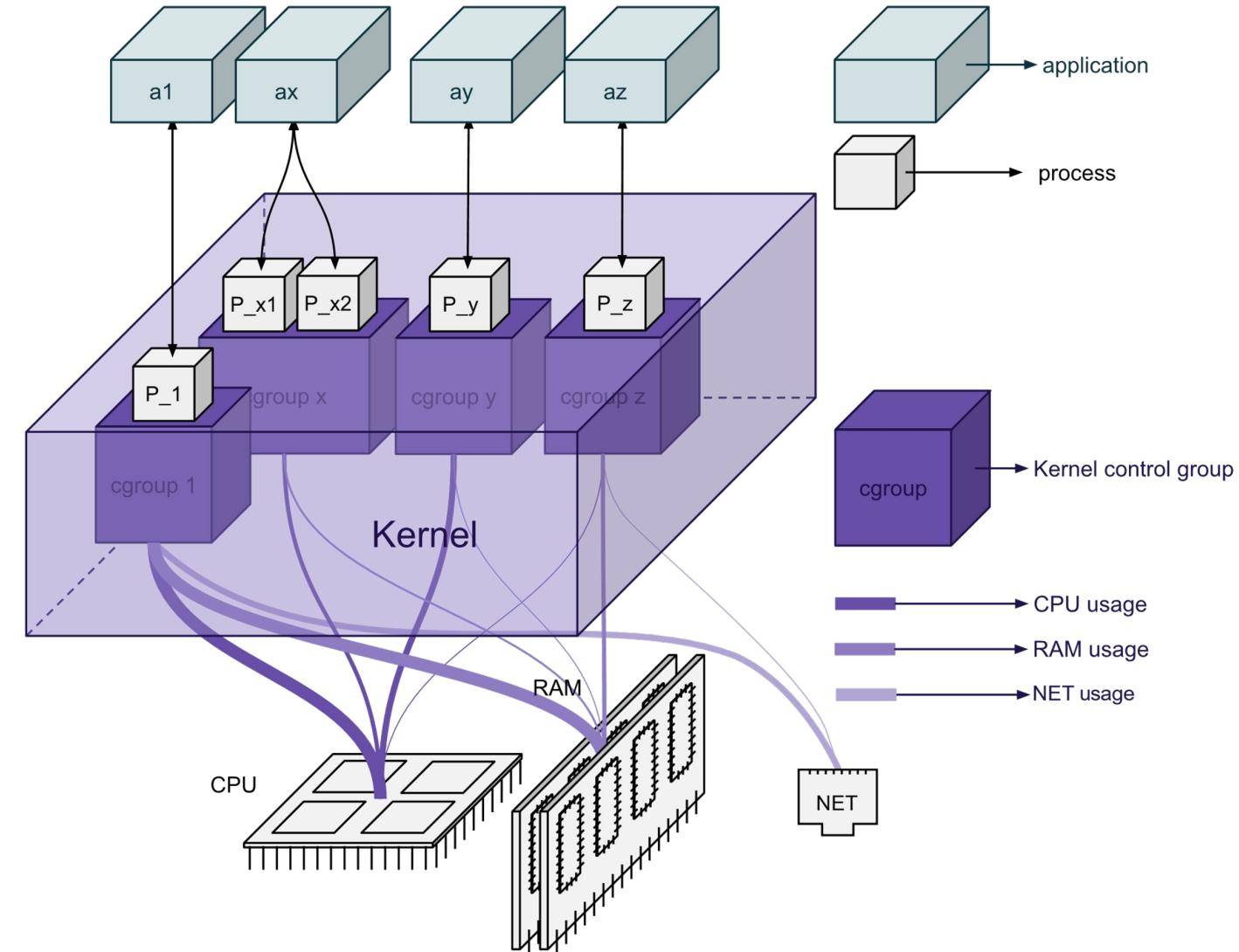
- ¿Qué significa *cgroups*? *control groups* o grupos de control
 - ¿Qué es o cuales son los grupos que controlan los *cgroups*?

gestionan, limitan y monitorizan grupos de procesos

- Se usan para limitar
 - CPU
 - Memoria
 - I/O
 - PID



cgroups





cgroups

- *cgroups*
 - Sistema jerárquico
 - Los límites impuestos por el *cgroup* padre lo hereda el *cgroup* hijo
 - Existen DOS versiones
 - No todos los controladores de la versión v1 están soportados en la v2



cgroups

- DEMO
 - Stress test
 - El *stress test* intenta consumir el 100% de la CPU
 - Fork bomb
 - El *fork bomb* intenta crear el máximo número de procesos hijo



namespaces

- ¿Qué son los *namespaces*?

particiones de los recursos del Kernel por aislamiento



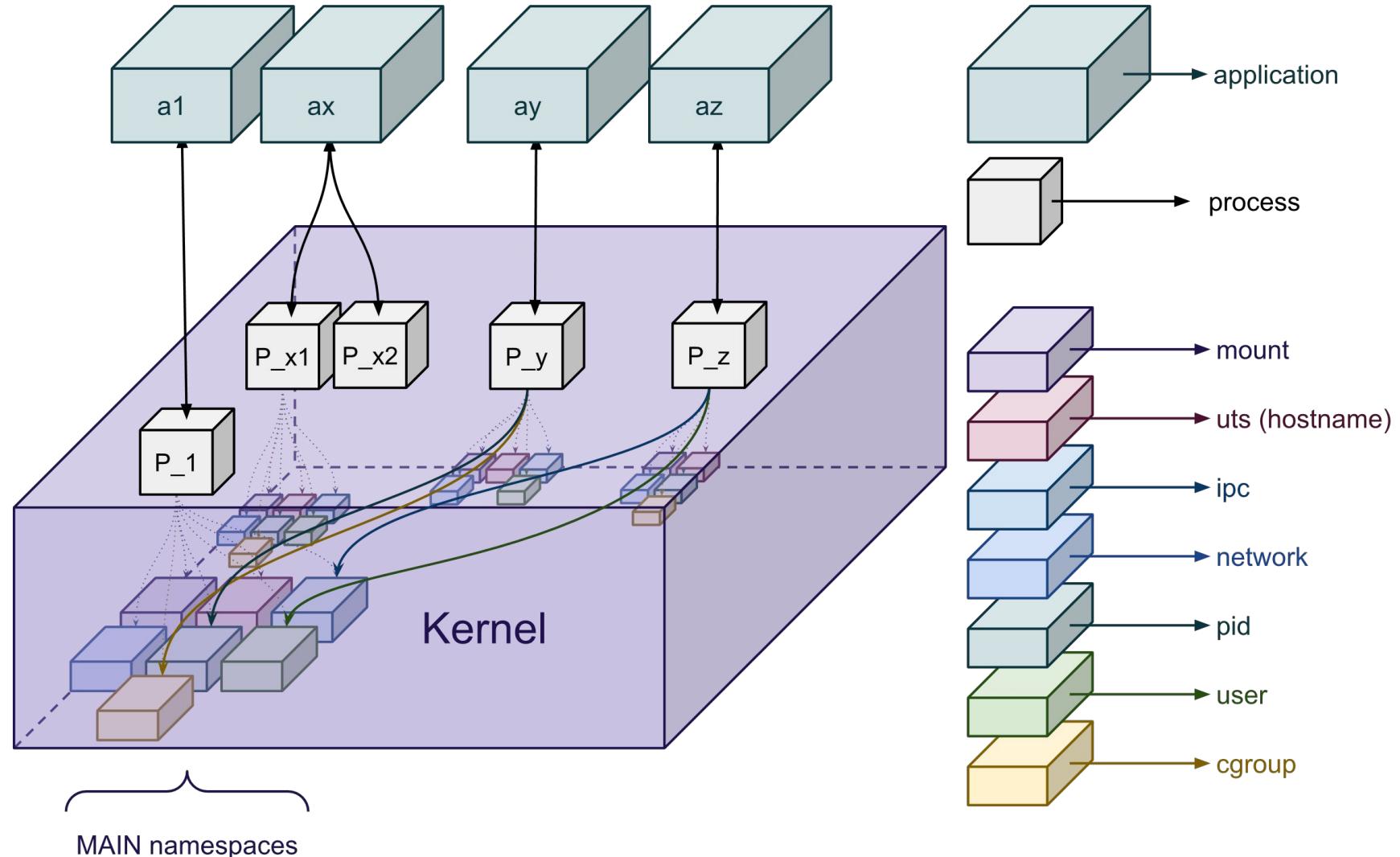
namespaces

- ¿Qué son los *namespaces*?
 - Los *namespaces* partitionan de los recursos del Kernel por aislamiento
 - Actualmente existen 8

namespace	Nivel de aislamiento
Mount	Puntos de montaje
UTS	Hostname y nombre de dominio NIS
IPC	IPC de System V y colas de mensajes POSIX
PID	Identificación de procesos
Network	Dispositivos de red, pilas, puertos, ...
User	Identificación de usuarios y grupos
Control groups	Directorio raíz del grupo de control
Time	Dos relojes de sistema <i>monotonic</i> y <i>boottime</i>

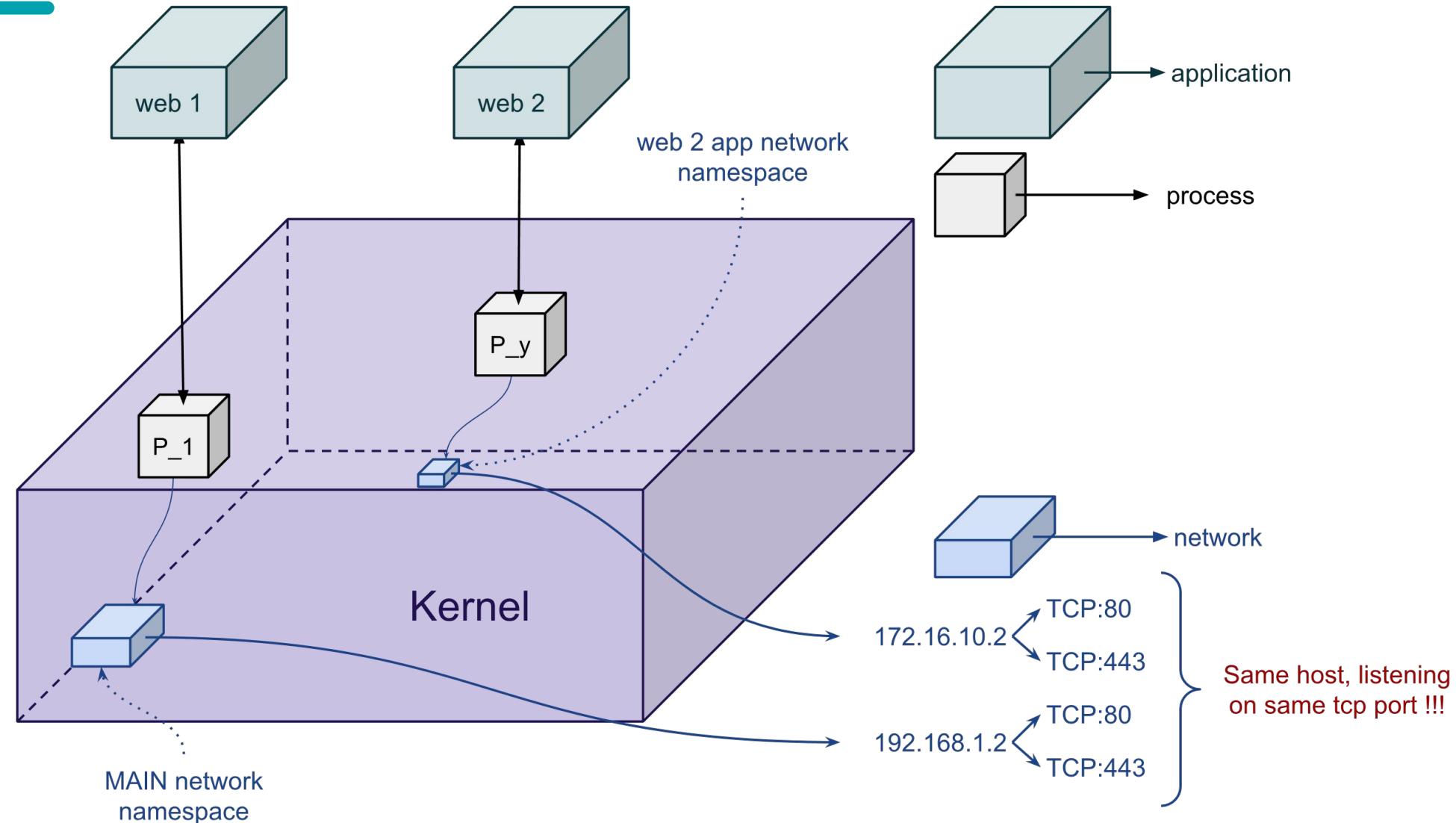


namespaces



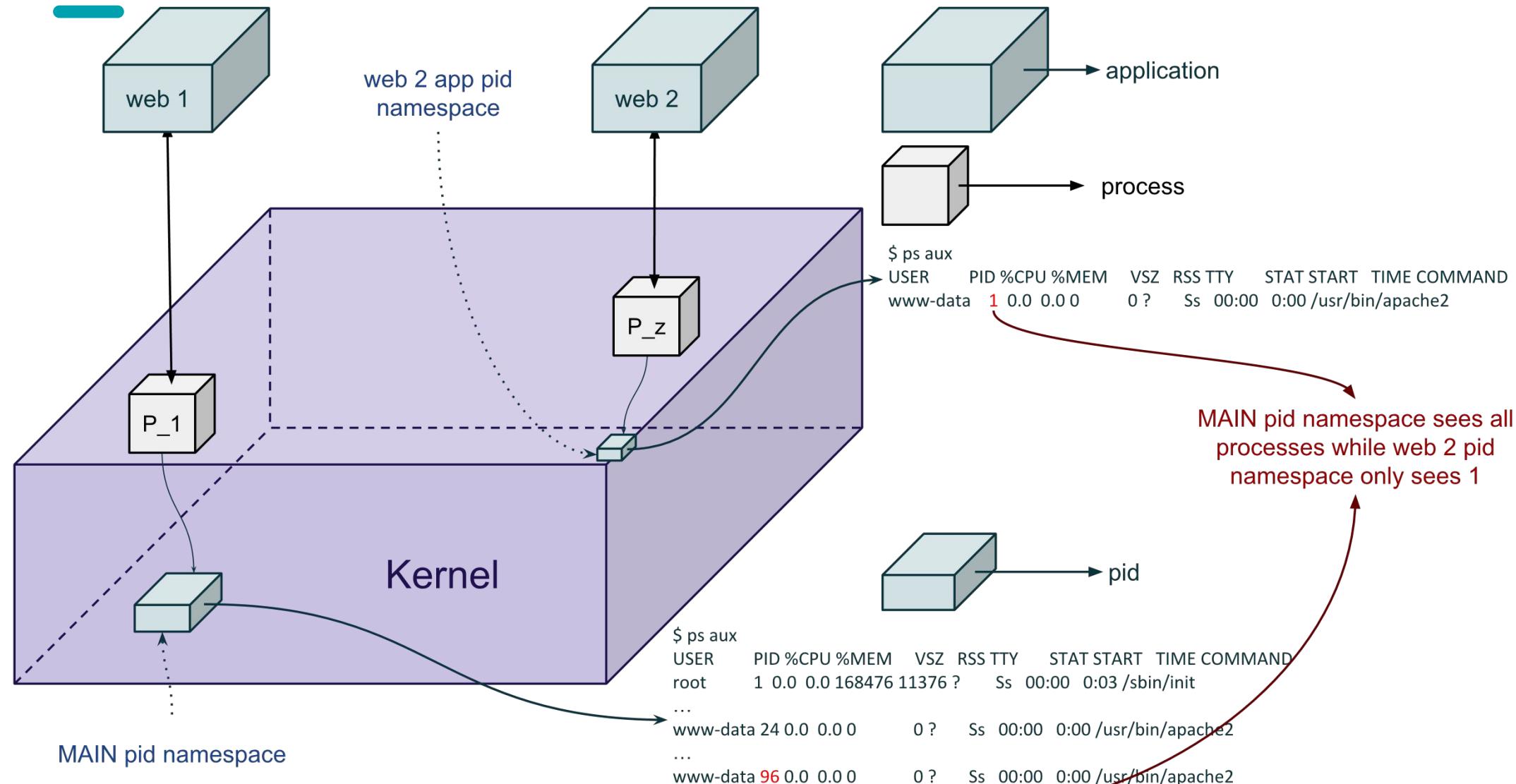


namespaces - Network





namespaces - PID





namespaces

- DEMO
 - mount
 - El **namespace mount** oculta los nuevos puntos de montaje al **namespace principal**
 - pid
 - El **namespace PID** permite hacer creer al proceso que tiene el identificador 1
 - uts
 - El **namespace UTS** permite renombrar el host sin afectar al **hostname principal**
 - network
 - El **namespace network** proporciona un **stack** de red sin ninguna interfaz
 - user
 - Mediante el **namespace user** podemos hacer creer al proceso que lo ejecuta **root**



capabilities

- ¿Qué son las *capabilities*?

divisiones de los privilegios del usuario *root*



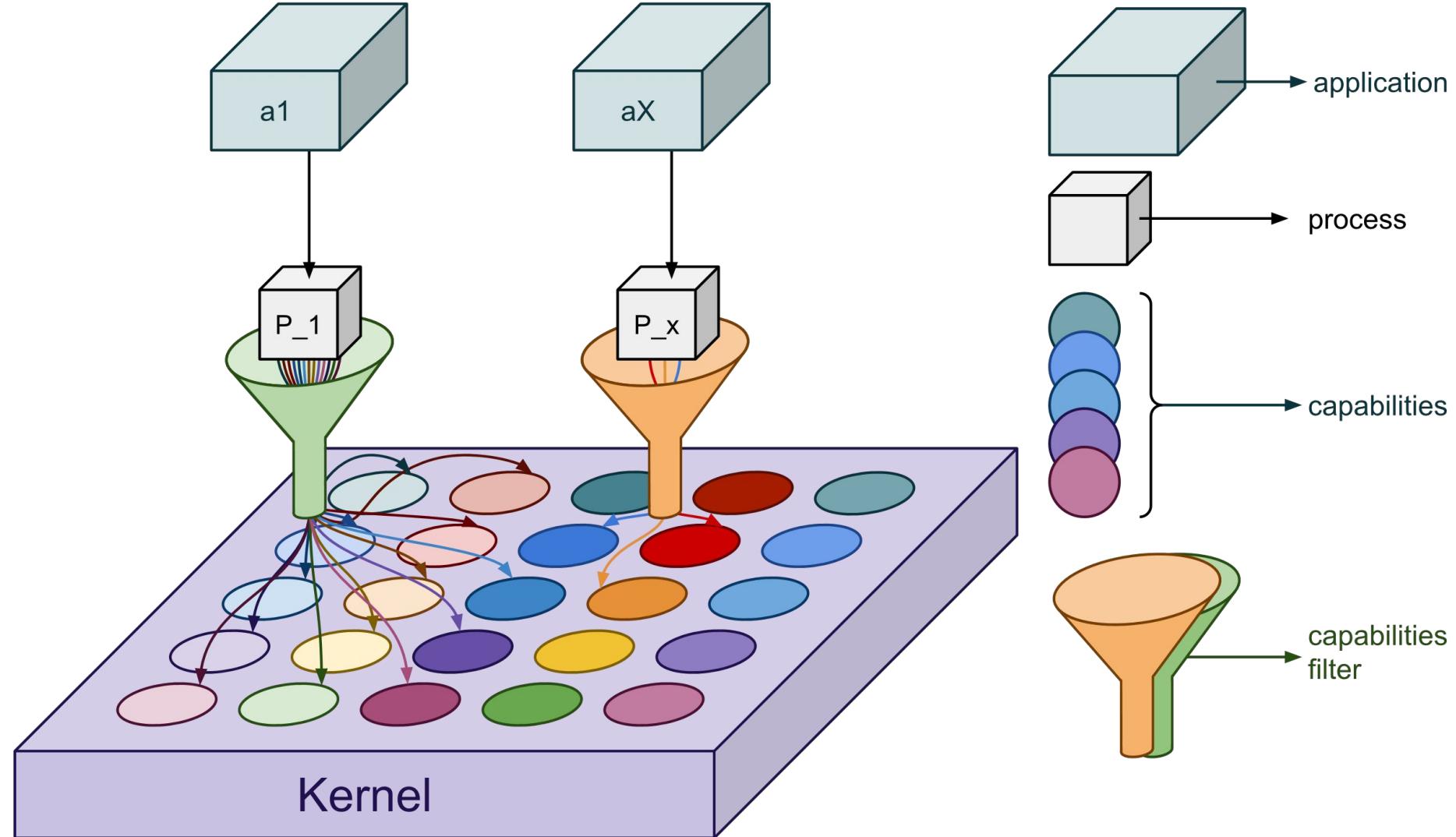
capabilities

- ¿Qué son las *capabilities*?
 - Las *capabilities* son divisiones de los privilegios del usuario root
 - Así, un proceso puede realizar acciones privilegiadas sin tener que ser *root*
 - Se pueden asignar a
 - un *thread*, o
 - un archivo para realizar ciertas acciones
- ¿Qué *capabilities* existen?

CAP_CHOWN	CAP_SETPCAP	CAP_SYS_RAWIO	CAP_SYS_TTY_CONFIG
CAP_DAC_OVERRIDE	CAP_LINUX_IMMUTABLE	CAP_SYS_CHROOT	CAP_MKNOD
CAP_DAC_READ_SEARCH	CAP_NET_BIND_SERVICE	CAP_SYS_PTRACE	CAPLEASE
H	E	CAP_SYS_PACCT	
CAP_FOWNER	CAP_NET_BROADCAST	CAP_SYS_ADMIN	
CAP_FSETID	CAP_NET_ADMIN	CAP_SYS_BOOT	
CAP_FS_MASK	CAP_NET_RAW	CAP_SYS_NICE	
CAP_KILL	CAP_IPC_LOCK	CAP_SYS_RESOURCE	
CAP_SETGID	CAP_IPC_OWNER	CAP_SYS_TIME	
CAP_SETUID	CAP_SYS_MODULE		



capabilities





capabilities

- DEMO
 - La aplicación *ping* tiene por defecto la *capability CAP_NET_RAW*



seccomp-bpf

- Antes de ver lo que es *seccomp-bpf*, ¿Qué es un *syscall*?

la interfaz fundamental entre una aplicación en el
espacio de usuario y el Kernel



seccomp-bpf

- ¿Qué es un *syscall*?
 - Un *syscall* es la interfaz fundamental entre una aplicación en el espacio de usuario y el Kernel
 - Cada vez que una aplicación quiere comunicarse con el Kernel, utiliza un *syscall*
 - El número de *syscalls* disponibles varía en función de la arquitectura
 - En la actualidad el Kernel de Linux tiene alrededor de 300



seccomp-bpf

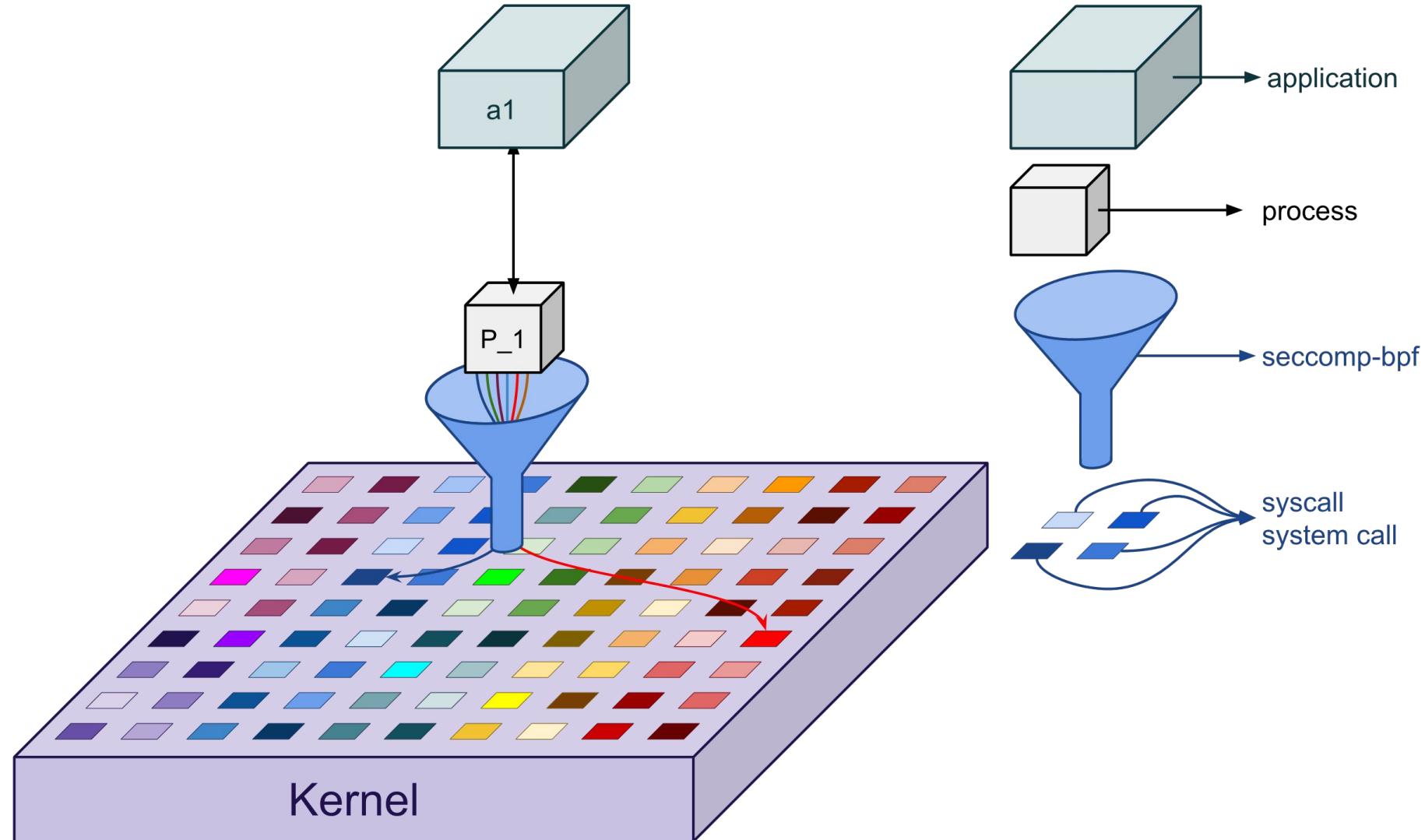
- ¿Qué significa seccomp-bpf?

secure computing berkeley packet filter

- ¿Qué hace seccomp-bpf?
 - Filtra los *syscall* que una aplicación es capaz de utilizar
 - Idealmente, cada aplicación debería de disponer de su perfil *seccomp-bpf* para permitir sólo aquellos *syscalls* necesarios para su correcto funcionamiento
 - Docker ejecuta los contenedores con un perfil por defecto el cual bloquea más de 40 *syscalls*



seccomp-bpf

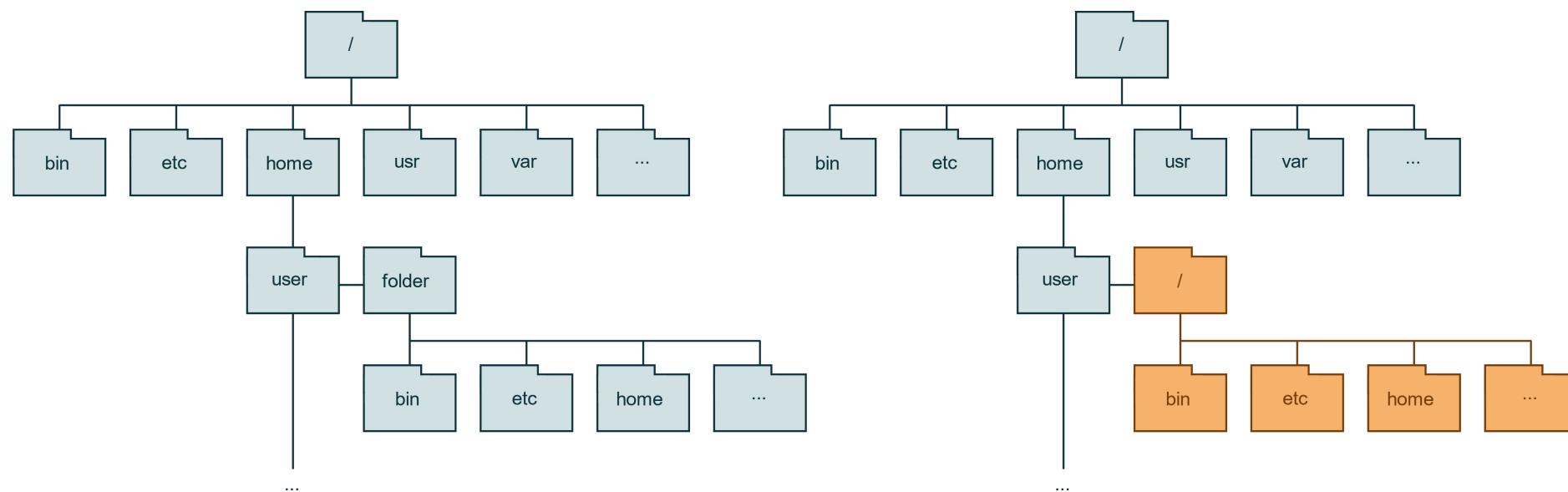




chroot

- ¿Qué significa *chroot*?

change root directory

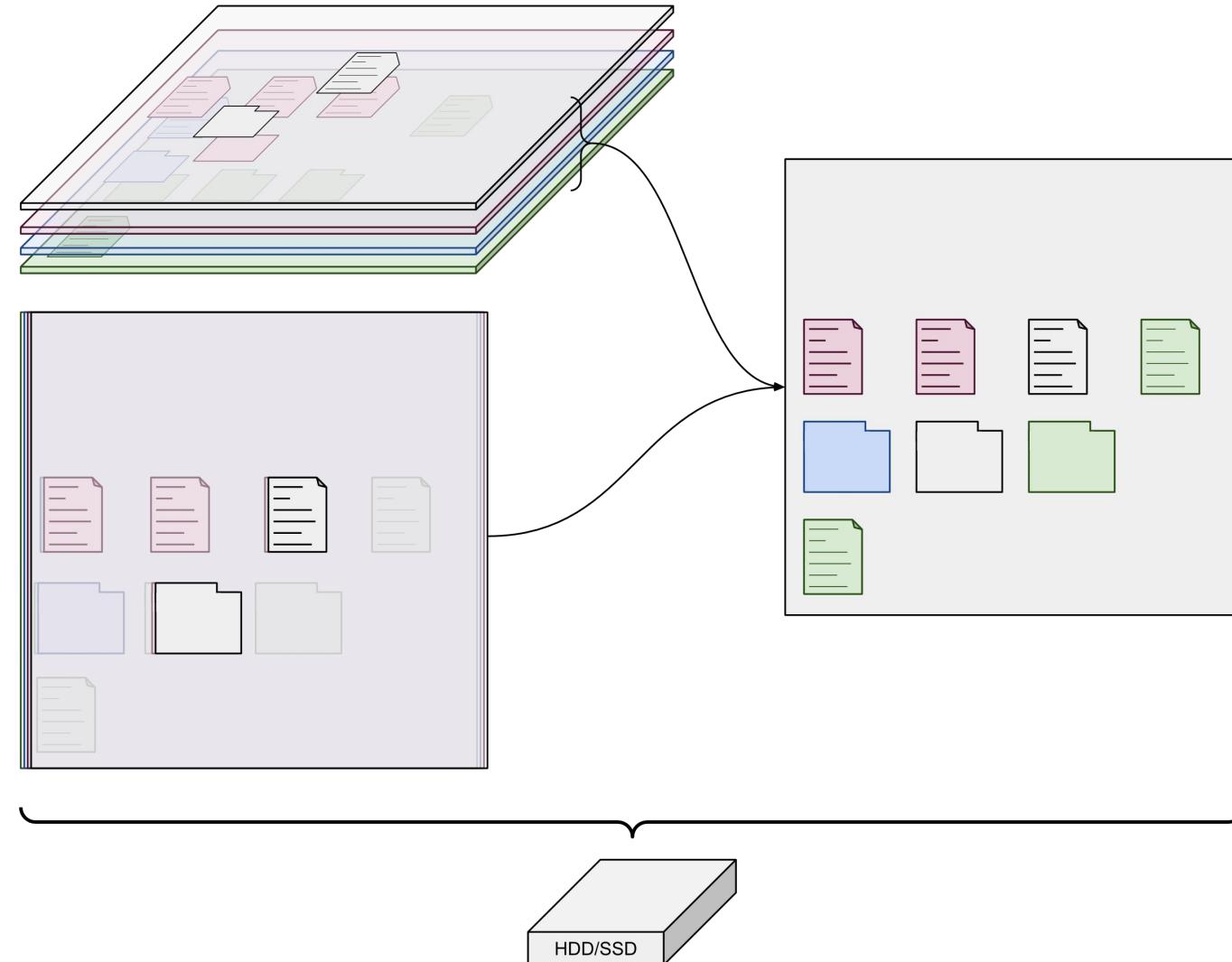




Sistemas de archivos copy-on-write

- Los sistemas de archivos *copy-on-write* (COW) se utilizan como mecanismo para los *snapshots*
 - Normalmente, en los *snapshots* solo se almacenan los datos modificados
 - *Overlay filesystem* o *overlayfs* es un tipo de sistema de archivos COW
 - Permite leer y escribir un árbol de directorios sobrepuerto sobre otro árbol de directorios de solo lectura
 - Todas las modificaciones se almacenan en el árbol de directorios superior
 - Se añadió en la versión 3.18 del Kernel de Linux

Sistemas de archivos copy-on-write





COW

- DEMO
 - *Overlayfs* nos permite crear y modificar directorios y archivos sin alterar los que ya existían



Conclusión

- He empezado con...
 - ¿Son realmente seguros?
 - Pues como casi todo... depende...
- Los mecanismos que hemos visto son la base de los contenedores
 - Mecanismos en los cuales se fundamentan soluciones como docker
 - Cómo se utilizan estos mecanismos es la base de la seguridad o la *inseguridad* de los contenedores :)



Mondragon
Unibertsitatea

Faculty of
Engineering



**Eskerrik asko
Thank you**

Iñaki Garitano

igaritano@mondragon.edu

Data Analysis and Cybersecurity
Electronics and Computing Department
Mondragon University - Faculty of Engineering
Goiuru, 2; 20500 Arrasate - Mondragón (Gipuzkoa), Spain
Tel. : +(34) 647503682 / +(34) 943794700 + Ext. 8119