

# Control Groups, Namespaces, Containers

## Explicación de práctica 4

Sistemas Operativos

Facultad de Informática  
Universidad Nacional de La Plata

2021



1 Linux cgroups y Namespaces

2 Linux Containers



1 Linux cgroups y Namespaces

2 Linux Containers



- Chroot es una forma de aislar aplicaciones del resto del sistema
- Introducido en la versión 7 de UNIX, 1979
- Cambia el directorio raíz aparente de un proceso. Afecta sólo a ese proceso y a sus procesos hijos
- Al entorno virtual creado por chroot a partir de la nueva raíz del sistema se le conoce como “jail chroot”
- No se puede acceder a archivos y comandos fuera de ese directorio
- “chroot /new-root-dir comando”
- chroot /usr/local/so ls /tmp



- En un sistema operativo se ejecutan varios procesos en forma concurrente
- En Linux, por defecto, todos los procesos reciben el mismo trato en lo que respecta a tiempo de CPU, memoria RAM, "I/O bandwidth".
- ¿Qué sucede si se tiene un proceso importante que requiere prioridad? O, ¿como limitar los recursos para un proceso o grupo de procesos?
- El kernel no puede determinar cuál proceso es importante y cuál no
- Existen algunas herramientas, como Nice, CPULimit o ulimit, que permiten controlar el uso de los recursos por un proceso, pero, ¿son suficientes?



- Control Groups, *cgroups*, son una característica del kernel de Linux que permite que los procesos sean organizados en grupos jerárquicos cuyo uso de varios tipos de recursos (CPU, memoria, I/O, etc.) pueda ser limitado y monitoreado.
- Desarrollo comenzado en Google por Paul Menage y Rohit Seth en el 2006 bajo el nombre de “process containers”
- Renombrado en 2007 como “Control Groups”. Disponible desde la versión del kernel 2.6.24
- Actualmente, Tejun Heo es el encargado del desarrollo y mantenimiento de CG.
- Versión 2 de CG con el Linux Kernel 4.5 de Marzo de 2016. Ambas versiones se habilitan por defecto
- La interface de cgroups del kernel es provista mediante un pseudo-filesystem llamado *cgroups*



- Permiten un control “fine-grained” en la asignación, priorización, denegación y monitoreo de los recursos del sistema
- cgroups provee lo siguiente:
  - **Resource Limiting:** grupos no pueden excederse en la utilización de un recurso (tiempo de CPU, cantidad de CPUs, cantidad de memoria, I/O, etc.)
  - **Prioritization:** un grupo puede obtener prioridad en el uso de los recursos (tiempo de CPU, I/O, etc.)
  - **Accounting:** permite medir el uso de determinados recursos por parte de un grupo (estadísticas, monitoreo, billing, etc.)
  - **Control:** permite freezear y reiniciar un grupo de procesos
- Procesos desconocen los límites aplicados por un “cgroup”



- Actualmente hay 12 subsistemas definidos

```
.config - Linux/x86 5.6.0 Kernel Configuration
+ General setup + Control Group support

Control Group support
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in
[ ] excluded <M> module <> module capable

-- Control Group support
[*] Memory controller
[*] Swap controller
[ ] Swap controller enabled by default
[*] IO controller
-- CPU controller --->
[*] PIDs controller
[*] RDMA controller
[*] Freezer controller
[ ] HugeTLB controller
[*] Cpuset controller
[*] Include legacy /proc/<pid>/cpuset file
[*] Device controller
[*] Simple CPU accounting controller
[*] Perf controller
[*] Support for eBPF programs attached to cgroups
[ ] Debug controller

<Select> < Exit > < Help > < Save > < Load >
```





- cgroups v1
  - Distintos controladores se han ido agregando en el tiempo para permitir la administración de distintos tipos de recursos
  - En cgroups v1, desarrollo de los controladores fue muy descoordinado
  - Administración de las distintas jerarquías se hizo cada vez más complejo
  - Diseño posterior a la implementación
- cgroups v2
  - cgroups v2 pensado como un reemplazo de cgroups v1
  - Controladores también agregados en el tiempo
- Ambos controladores pueden ser montados en el mismo sistema
- Una jerarquía de un controlador no puede estar en ambos cgroups simultáneamente



- **cgroup:** colección de procesos que están ligados a un conjunto de límites o parámetros definidos mediante el pseudo-filesystem cgroupfs.
- **Subsistema:** componente del kernel que modifica el comportamiento de los procesos en un cgroup. También llamado *resource controllers* o simplemente *controllers*.
- **Jerarquía:** forma en la que son organizados los cgroups para un subsistema.
- Cada subsistema representa un único recurso: tiempo de CPU, memoria, I/O, etc.
- Cada jerarquía es definida mediante la creación, eliminación y renombrado de subdirectorios dentro del pseudo-filesystem
- Cada proceso del sistema solo puede pertenecer a un cgroup dentro de una jerarquía (pero a varias jerarquías)

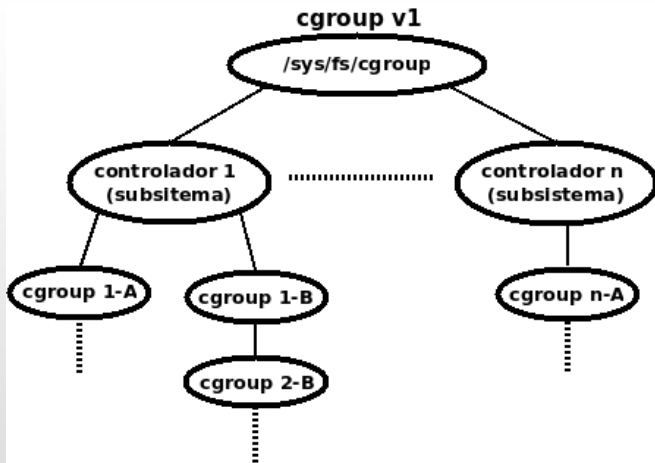


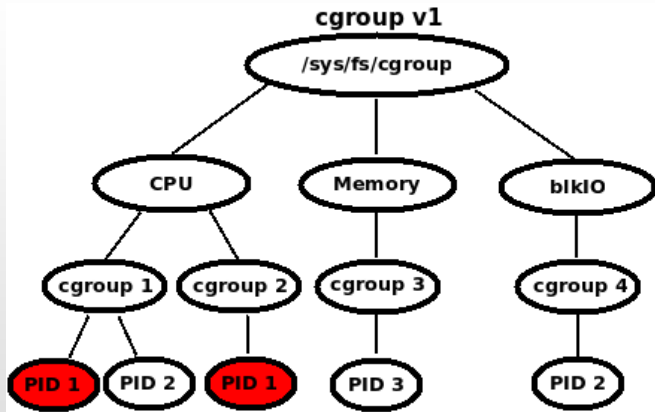
- Controladores pueden ser montados en pseudo-filesystems individuales o en un mismo pseudo-filesystem
  - Por cada jerarquía, la estructura de directorios refleja la jerarquía de los cgroups
  - Cada cgroup es representado por un directorio en una relación padre-hijo. Por ejemplo: `cpu/procesos/proceso1`
  - En cada nivel de la jerarquía se pueden definir atributos (por ej. límites). No pueden ser excedidos por los cgroups hijos.
  - Un proceso creado mediante un "fork" pertenece al mismo cgroup que el padre
  - Cada directorio contiene archivos que pueden ser escritos/leídos
  - Una vez definidos los grupos se le agregan los IDs de procesos
  - Posible asignar threads de un proceso a diferentes cgroups.
- Deshabilitado en la v2, restaurado luego, pero con limitaciones



- Un controlador v1 debe ser montado contra un filesystem de tipo cgroup. Usualmente es mediante un filesystem *tmpfs* montado en */sys/fs/cgroup*.
- *mount -t cgroup -o cpu none /sys/fs/cgroup/cpu* para montar un controlador en particular (CPU en este caso)
- *mount -t cgroup -o all cgroup /sys/fs/cgroup* para montar todo los controladores
- Un controlador puede ser desmontado si no está ocupado: no tiene cgroups hijos (*umount /sys/fs/cgroup/cpu*)
- Cada cgroup filesystem contiene un único cgroup raíz al cual pertenecen todos los procesos
- Un proceso creado mediante un "fork" pertenece al mismo cgroup que el padre
- *libcgroup*: tools para administrar los cgroups







# cgroup v1 - Características

```
root@so2020:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=100824k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
....
```



# cgroup v1 - Características

```
root@so2020:/sys/fs/cgroup# ls -l
total 0
dr-xr-xr-x 2 root root 0 abr 20 08:28 blkio
lrwxrwxrwx 1 root root 11 abr 20 08:28 cpu -> cpu,cpuacct
lrwxrwxrwx 1 root root 11 abr 20 08:28 cpuacct -> cpu,cpuacct
dr-xr-xr-x 2 root root 0 abr 20 08:28 cpu,cpuacct
dr-xr-xr-x 2 root root 0 abr 20 08:28 cpuset
dr-xr-xr-x 4 root root 0 abr 20 08:28 devices
dr-xr-xr-x 2 root root 0 abr 20 08:28 freezer
dr-xr-xr-x 4 root root 0 abr 20 08:28 memory
lrwxrwxrwx 1 root root 16 abr 20 08:28 net_cls -> net_cls,net_prio
dr-xr-xr-x 2 root root 0 abr 20 08:28 net_cls,net_prio
lrwxrwxrwx 1 root root 16 abr 20 08:28 net_prio -> net_cls,net_prio
dr-xr-xr-x 2 root root 0 abr 20 08:28 perf_event
dr-xr-xr-x 4 root root 0 abr 20 08:28 pids
dr-xr-xr-x 2 root root 0 abr 20 08:28 rdma
dr-xr-xr-x 5 root root 0 abr 20 08:28 systemd
dr-xr-xr-x 5 root root 0 abr 20 08:28 unified
```





- cgcreate, o mkdir dentro de la estructura, para crear un cgroup
- Systemd alternativa para administra los cgroups

```
so@so:/$sudo cgcreate -g cpuset:my_group
so@so:/$ ls -l /sys/fs/cgroup/cpuset/my_group/
total 0
-rw-r--r-- 1 root root 0 jun  5 23:18 cgroup.clone_children
-rw-r--r-- 1 root root 0 jun  5 23:18 cgroup.procs
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.cpu_exclusive
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.cpus
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.mem_exclusive
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.mem_hardwall
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.memory_migrate
-r--r--r-- 1 root root 0 jun  5 23:18 cpuset.memory_pressure
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.memory_spread_page
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.memory_spread_slab
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.mems
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.sched_load_balance
-rw-r--r-- 1 root root 0 jun  5 23:18 cpuset.sched_relax_domain_level
-rw-r--r-- 1 root root 0 jun  5 23:18 notify_on_release
-rw-r--r-- 1 root root 0 jun  5 23:18 tasks
```

- echo "0-2,4" > /sys/fs/cgroup/cpuset/my\_group/cpuset.cpus
- echo "PID" > /sys/fs/cgroup/cpuset/my\_group/cgroup.procs

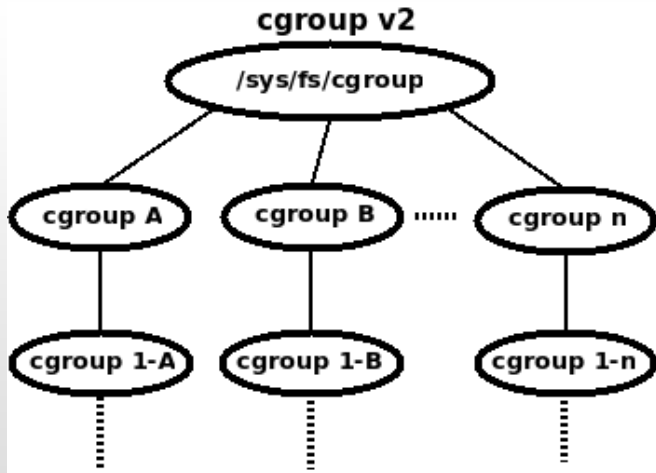


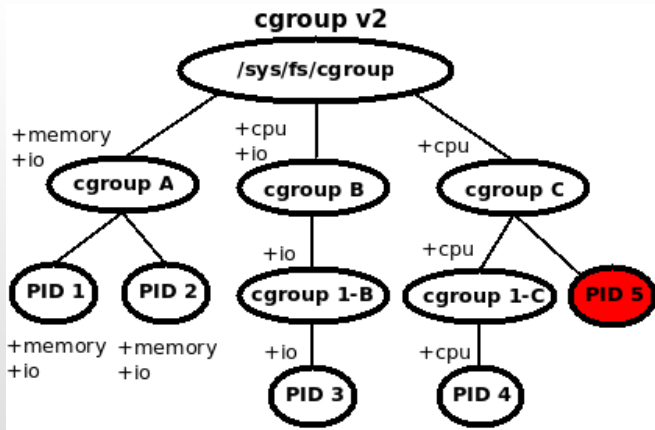
- Todos los controladores son montados en una jerarquía unificada
- No se permiten procesos internos, a excepción del cgroup root. Procesos deben asignarse a cgroups sin hijos (hojas)
- No es posible especificar un controlador en particular para montar
- `mount -t cgroup2 none /mnt/cgroup2`
- Todos los controladores son equivalente a los de cgroups v1, excepto `net_cls` y `net_prio`
- Cada cgroup en la jerarquía v2 contiene, entre otros, los siguiente archivos:
  - `cgroup.controllers`: archivo de solo lectura que indica los controladores disponibles en este cgroup
  - `cgroup.subtree_control`: lista de controladores habilitados en el cgroup.



- Conjunto de controladores activos es modificado escribiendo el nombre de los controladores en el archivo correspondiente. + para habilitar, - para deshabilitar
- `echo '+pids -memory' > cgroup.subtree_control`
- `cgroup.subtree_control` determina el conjunto de controladores que son empleados en los cgroups hijos.
- Procesos solo en las hojas. Por ej. si `/procesos/proc1` son cgroups entonces un proceso puede residir en `/procesos/proc1`, pero no en `/procesos`.
- Inicialmente, solo el cgroup root existe y todos los procesos pertenecen a él
- `mkdir CG_NAME` y `rmdir CG_NAME` para crear y eliminar cgroups.
- Cada cgroup tiene un archivo lectura/escritura `cgroup.procs`







```
root@so:/sys/fs/cgroup# ls -l
total 0
-r--r--r-- 1 root root 0 abr 20 08:16 cgroup.controllers
-rw-r--r-- 1 root root 0 abr 20 08:20 cgroup.max.depth
-rw-r--r-- 1 root root 0 abr 20 08:20 cgroup.max.descendants
-rw-r--r-- 1 root root 0 abr 20 08:16 cgroup.procs
-r--r--r-- 1 root root 0 abr 20 08:20 cgroup.stat
-rw-r--r-- 1 root root 0 abr 20 08:16 cgroup.subtree_control
-rw-r--r-- 1 root root 0 abr 20 08:20 cgroup.threads
-r--r--r-- 1 root root 0 abr 20 08:20 cpuset.cpus.effective
-r--r--r-- 1 root root 0 abr 20 08:20 cpuset.mems.effective
drwxr-xr-x 2 root root 0 abr 20 08:16 init.scope
drwxr-xr-x 19 root root 0 abr 20 08:16 system.slice
drwxr-xr-x 3 root root 0 abr 20 08:18 user.slice
```



## Namespace Isolation

Permite abstraer un recurso global del sistema para que los procesos dentro de ese “namespace” piensen que tienen su propia instancia aislada de ese recurso global

- Proceso, junto con sus hijos, tienen una vista diferente del sistema subyacente
- Modificaciones a un recurso quedan contenidas dentro del “namespace”
- Tres nuevas systems-calls:
  - **clone()**: crea un nuevo proceso y lo agrega al nuevo namespace especificado
  - **unshare()**: agrega el actual proceso a un nuevo namespace especificado
  - **setns()**: agrega el proceso actual a un namespace existente



- Entre los namespaces provistos por Linux:
  - **IPC:** Flag: CLONE\_NEWIPC. System V IPC, cola de mensaje POSIX
  - **Network:** Flag: CLONE\_NEWNET. Dispositivos de red, pilas, puertos, etc
  - **Mount:** Flag: CLONE\_NEWNS. Puntos de montaje
  - **PID:** Flag: CLONE\_NEWPID. IDs de procesos
  - **User:** Flag: CLONE\_NEWUSER. IDs de usuarios y grupos
  - **UTS:** Flag: CLONE\_NEWUTS. HostName y nombre de dominio
  - **Cgroup:** Flag: CLONE\_NEWCGROUP. Cgroup root directory
  - **Time:** Flag: CLONE\_NEWTIME. Distintos offsets al clock del sistema por namespace (Marzo 2020)
- Flag: usado para indicar el namespace en las system calls

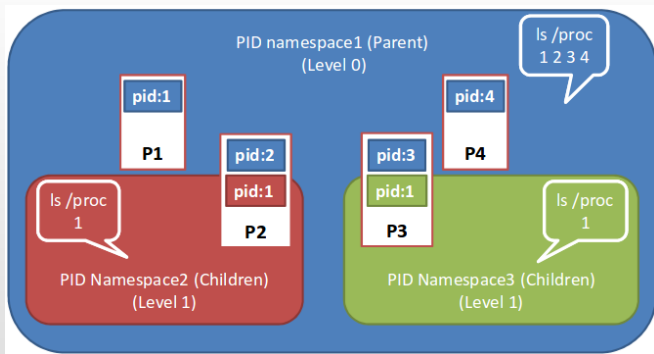




- **clone()** similar al fork pero mas general. Su funcionalidad puede ser controlada por flags pasados como argumentos
- Cada proceso tiene un subdirectorio que contiene los namespaces a los que está asociado: `/proc/[pid]/ns`
- **setns()** permite modificarlo: desasocia al proceso llamante de una instancia de un tipo de namespace y lo reasocia con otra instancia del mismo tipo de namespace
- **unshare()** es similar a clone, pero opera en al proceso llamante: crea el nuevo namespace y hace miembro de él al proceso llamador
- Un namespace es automáticamente eliminado cuando el último proceso en el namespace finaliza o lo abandona



- Posibilidad de tener múltiples árboles de procesos anidados y aislados

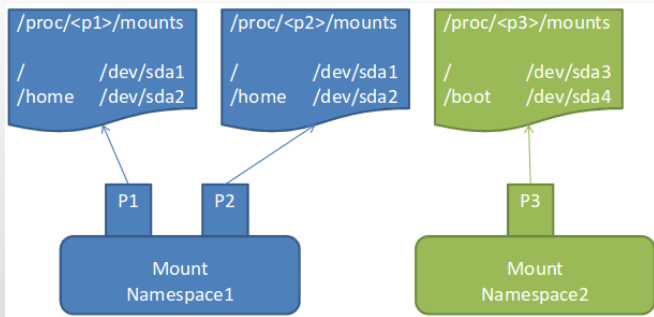


Fuente imagen: [http://events.linuxfoundation.org/sites/events/files/cojp13\\_feng.pdf](http://events.linuxfoundation.org/sites/events/files/cojp13_feng.pdf)



# Mount Namespace

- Permite aislar la tabla de montajes (montajes por namespace)
- Cada proceso, o conjunto de procesos, tiene una vista distinta de los puntos de montajes



Fuente imagen: [http://events.linuxfoundation.org/sites/events/files/cojp13\\_feng.pdf](http://events.linuxfoundation.org/sites/events/files/cojp13_feng.pdf)



1 Linux cgroups y Namespaces

2 Linux Containers



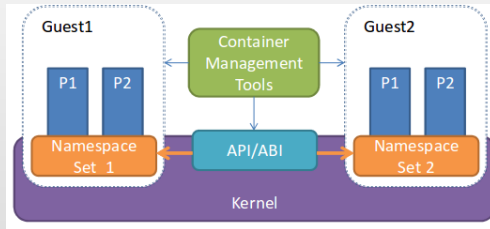
- Tecnología liviana de virtualización (lightweight virtualization) a nivel de sistema operativo que permite ejecutar múltiples sistemas aislados (conjuntos de procesos) en un único host
- Instancias ejecutan en el espacio del usuario. Comparten el mismo kernel (el del SO base)
- Dentro de cada instancia son como máquinas virtuales. Por fuera son procesos normales del SO
- Método de virtualización más eficiente. Mejor performance, booteo más rápido
- No es necesario un software de virtualización VMM o hypervisor
- LXC, Solaris Zones, BSD Jails, Docker, OpenVZ, etc.
- No es posible ejecutar instancias de SO con kernel diferente al SO base (por ej. Windows sobre Linux)



- Proyecto comenzado a mediados de 2008
- Utiliza las características de “cgroups” y “namespace isolation”
- Docker utilizaba LXC en su orígenes (luego pasó a sus propias librerías llamadas libcontainer)
- En 2015, Ubuntu lanzó LXD que permite desplegar y administrar containters LXC (usa un pequeño hypervisor)
- Actualmente el proyecto es mantenido por Canonical LTD.



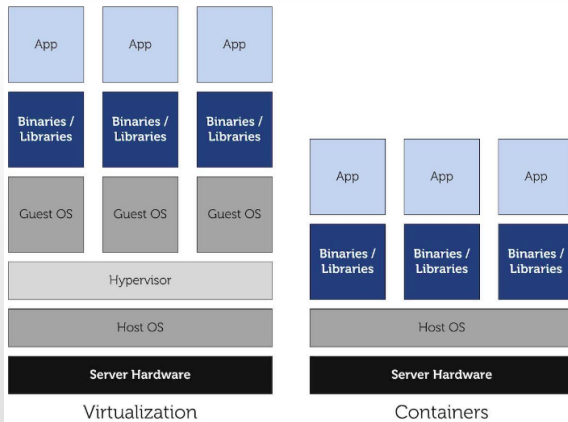
- Cada container:
  - Tiene su propio nombre (y dominio)
  - Tiene sus propias interfaces de red (junto con sus IPs)
  - Tiene sus propios filesystems
  - Tiene su propio espacios de nombre de procesos ID (PIDs) e IPC
  - Provee aislación (seguridad y uso de recursos)



Fuente imagen: [http://events.linuxfoundation.org/sites/events/files/cojp13\\_feng.pdf](http://events.linuxfoundation.org/sites/events/files/cojp13_feng.pdf)



# Linux Containers - Hypervisor vs. Container



Fuente Imagen:

<https://wiki.aalto.fi/download/attachments/109397667/Linux%20containers.pdf?version=2&modificationDate=1447254317>





- “apt-get install lxc”
- Paquetes adicionales: lxcctl, bridge-utils, libvirt-bin

```
so2018@debian:/$ lxc-checkconfig
Kernel configuration not found at /proc/config.gz; searching...
Kernel configuration found at /boot/config-3.16.0-4-amd64
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled
Multiple /dev/pts instances: enabled
--- Control groups ---
Cgroup: enabled
```



- Posibles templates para instalar en **lxc**

```
so2018@debian:/$ ls -alh /usr/share/lxc/templates/  
total 344K  
drwxr-xr-x 2 root root 4,0K mar 25 15:15 .  
drwxr-xr-x 7 root root 4,0K mar 25 15:15 ..  
-rwxr-xr-x 1 root root 11K nov 14 2015 lxc-alpine  
-rwxr-xr-x 1 root root 14K nov 14 2015 lxc-altlinux  
-rwxr-xr-x 1 root root 11K nov 14 2015 lxc-archlinux  
-rwxr-xr-x 1 root root 9,3K nov 14 2015 lxc-busybox  
-rwxr-xr-x 1 root root 29K nov 14 2015 lxc-centos  
-rwxr-xr-x 1 root root 10K nov 14 2015 lxc-cirros  
-rwxr-xr-x 1 root root 14K nov 14 2015 lxc-debian  
-rwxr-xr-x 1 root root 18K nov 14 2015 lxc-download  
-rwxr-xr-x 1 root root 47K nov 14 2015 lxc-fedora  
....
```



- *lxc-create* permite crear un contenedor:
  - *lxc-create -n "nombre contenedor" -t "template"*
  - Por ej.: *lxc-create -n lxcdebian -t debian*
- *lxc-start -n lxcdebian* para iniciar un container
- *lxc-info -n "nombre contenedor"* para obtener información sobre un container running/stopped
- *lxc-ls --active* para ver containers ejecutando (*--fancy* para ver todos los containers)
- *lxc-console -n "nombre contenedor"* para entrar a la consola de un container
- *lxc-stop -n "nombre contenedor"* para detener un contenedor



```
root@so2020:~# ps -ef
UID          PID    PPID  C  STIME TTY          TIME CMD
root           1        0  0  20:03 ?        00:00:01 /sbin/init
.....
root          265        1  0  20:03 ?        00:00:00 /lib/systemd/systemd-journald
root          284        1  0  20:03 ?        00:00:00 /lib/systemd/systemd-udevd
systemd+      300        1  0  20:03 ?        00:00:00 /lib/systemd/systemd-timesyncd
root          462        1  0  20:03 ?        00:00:00 /sbin/mdadm --monitor --scan
root          464        2  0  20:03 ?        00:00:00 [dm_bufio_cache]
root         1241       1144  0  20:08 ?        00:00:00 /usr/sbin/sshd -D
root         1310        1  0  20:12 ?        00:00:00 [lxc monitor] /var/lib/lxc sodebian
root         1316       1310  0  20:12 ?        00:00:00 /sbin/init
root         1364       1316  0  20:12 ?        00:00:00 /lib/systemd/systemd-journald
root         1416       1316  0  20:12 ?        00:00:00 /usr/sbin/sshd -D
root         1557        837  0  20:13 tty1      00:00:00 su -
root         1558       1557  0  20:13 tty1      00:00:00 -bash
root         1590       1558  0  20:16 tty1      00:00:00 /bin/bash
root         1738       1734  0  20:32 pts/3    00:00:00 su -
root         1739       1738  0  20:32 pts/3    00:00:00 -bash
.....
```



# Linux Containers - Namespaces

```
root@so2020:~# ls -l /proc/1/ns/
total 0
lrwxrwxrwx 1 root root 0 may 17 20:44 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 may 17 20:44 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 root root 0 may 17 20:44 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 root root 0 may 17 20:44 net -> 'net:[4026531992]'
lrwxrwxrwx 1 root root 0 may 17 20:44 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 may 17 20:44 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 may 17 20:44 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 may 17 20:44 uts -> 'uts:[4026531838]'
root@so2020:~# ls -l /proc/462/ns/
total 0
lrwxrwxrwx 1 root root 0 may 17 20:45 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 may 17 20:45 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 root root 0 may 17 20:45 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 root root 0 may 17 20:45 net -> 'net:[4026531992]'
lrwxrwxrwx 1 root root 0 may 17 20:45 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 may 17 20:45 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 may 17 20:45 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 may 17 20:45 uts -> 'uts:[4026531838]'
```

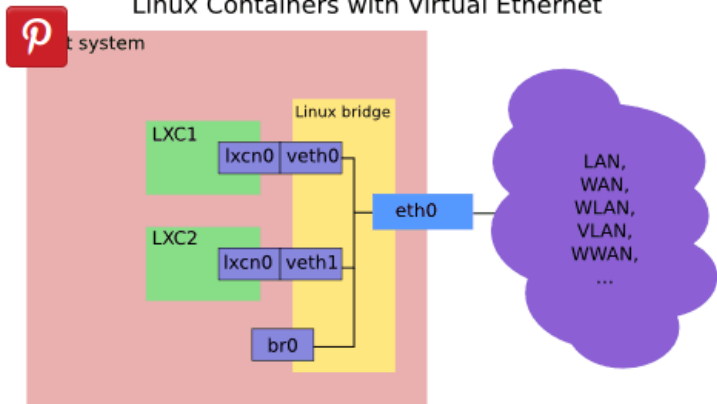


- Procesos del container ejecutan como procesos dentro del SO anfitrión

```
root@so2020:~\# lxc-info -n sodebian
Name:          sodebian
State:         RUNNING
PID:           1316
IP:            10.0.2.17
CPU use:       0.49 seconds
.....
root@so2020:~\# ls -l /proc/1316/ns/
total 0
lrwxrwxrwx 1 root root 0 may 17 16:08 cgroup -> 'cgroup:[4026532293]'
lrwxrwxrwx 1 root root 0 may 17 16:08 ipc -> 'ipc:[4026532237]'
lrwxrwxrwx 1 root root 0 may 17 16:08 mnt -> 'mnt:[4026532235]'
lrwxrwxrwx 1 root root 0 may 17 16:08 net -> 'net:[4026532240]'
lrwxrwxrwx 1 root root 0 may 17 16:08 pid -> 'pid:[4026532238]'
lrwxrwxrwx 1 root root 0 may 17 16:13 pid_for_children -> 'pid:[4026532238]'
lrwxrwxrwx 1 root root 0 may 17 16:13 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 may 17 16:08 uts -> 'uts:[4026532236]'
```



## Linux Containers with Virtual Ethernet



Copyright © 2010 Diego Elio Pettenò <flameeyes@gmail.com>  
Licensed under Creative Commons Attribution-ShareAlike License 3.0 Unported



- Cambiar root password:
  - chroot en el filesystem del contenedor
    - `chroot /var/lib/lxc/contenedor/rootfs`
  - Comando “passwd” e ingresar la nueva password de root
  - También es posible hacerlo modificando el archivo “shadow” del container
- Para montar otro file system agregar en el archivo de configuración del container (`/var/lib/lxc/contenedor/config`):
  - `lxc.mount.entry=/path/dir /var/lib/lxc/contenedor/rootfs/punto montanje none bind 0 0`
  - Reiniciar el contenedor







*cgroups v1 - Kernel Documentation*

<https://www.kernel.org/doc/Documentation/cgroup-v1/>



*cgroups v2 - Kernel Documentation*

<https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v2.html>



*cgroups v1/v2 - Linux Man Pages*

<http://man7.org/linux/man-pages/man7/cgroups.7.html>



*Namespaces - Linux Man Pages*

<http://man7.org/linux/man-pages/man7/namespaces.7.html>





## *Namespaces in Operation*

<https://lwn.net/Articles/531114/>



## *Linux Containers*

<https://linuxcontainers.org/>



## *Linux Containers*

<https://ubuntu.com/server/docs/containers-lxc>

