Control Groups, Namespaces, Containers

Explicación de práctica 4

Sistemas Operativos

Facultad de Informática
Universidad Nacional de La Plata

2018

Agenda

1. Linux cgroups y Namespaces
2. Linux Containers

Agenda

1. Linux cgroups y Namespaces
2. Linux Containers

¿Para qué utilizamos virtualización?

Service Isolation - chroot

- Chroot 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

- Debian posee un método simple que permite instalar un sistema base Debian en el subdirectorio de otro sistema instalado: “deboostrap”
- Puede ser instalada y ejecutada desde otro SO y otra arquitectura (cross-deboostraping)

```
debootstrap stable target/ http://httpredir.debian.org
mount --bind /dev/ target/dev/
mount --bind /proc/ target/proc/
mount --bind /sys/ target/sys/
chroot target
```

- En un sistema operativo se ejecutan varios procesos en forma concurrente
- En Linux, por defecto, todos los procesos reciben el mismo tiempo de CPU o "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*, es un mecanismo que permite organizar procesos en forma jerárquica y distribuir los recursos del sistema (CPU, memoria, I/O, etc.) a lo largo de esa jerarquía en un manera controlada y configurable
- 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
- Permite un control "fine-grained" en la alocación, priorización, denegación y monitoreo de los recursos del sistema

- Organizado jerárquicamente, como los procesos, y los procesos hijos heredan atributos
- Pueden existir muchas jerarquías diferentes de cgroups. cgroups2 una sola jerarquía
- Cada jerarquía se asocia a uno o más subsistemas
- Un subsistema representa un único recurso: tiempo de CPU, memoria, I/O, etc.
- Compuesto de dos partes:
  - **Core:** responsable de organizar jerarquicamente los procesos
  - **Controller:** responsable de distribuir los recursos del sistema a lo largo de la jerarquía.
- Cada proceso del sistema solo puede pertenecer a un cgroup dentro de una jerarquía

cgroup - Características

- Se implementa como un “pseudo filesystem”
- Un proceso, y todos sus threads, pertenecen a un solo “cgroup”
- Procesos desconocen los límites aplicados por un “cgroup”
- Una vez definidos los grupos se le agregan los IDs de procesos
- *libcgroup:* tools para administrar los cgroups

```
so2018@debian:/sys/fs/cgroup$ ls -l
total 0
dr-xr-xr-x 3 root root  0 jun  5 23:11 blkio
lrwxrwxrwx 1 root root 11 may 23 16:45 cpu -> cpu,cpuacct
lrwxrwxrwx 1 root root 11 may 23 16:45 cpuacct -> cpu,cpuacct
dr-xr-xr-x 6 root root  0 jun  5 23:11 cpu,cpuacct
dr-xr-xr-x 4 root root  0 jun  5 23:18 cpuset
dr-xr-xr-x 5 root root  0 jun  5 23:11 devices
dr-xr-xr-x 3 root root  0 jun  5 23:11 freezer
lrwxrwxrwx 1 root root 16 may 23 16:45 net_cls -> net_cls,net_prio
dr-xr-xr-x 3 root root  0 jun  5 23:11 net_cls,net_prio
lrwxrwxrwx 1 root root 16 may 23 16:45 net_prio -> net_cls,net_prio
dr-xr-xr-x 3 root root  0 jun  5 23:11 perf_event
dr-xr-xr-x 4 root root  0 jun  5 23:11 systemd
```

- cgcreate, o con mkdir dentro de la estructura, para crear un grupo

```
so2018@debian:/$sudo cgcreate -g cpuset:my_group
so2018@debian:/$ 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
```

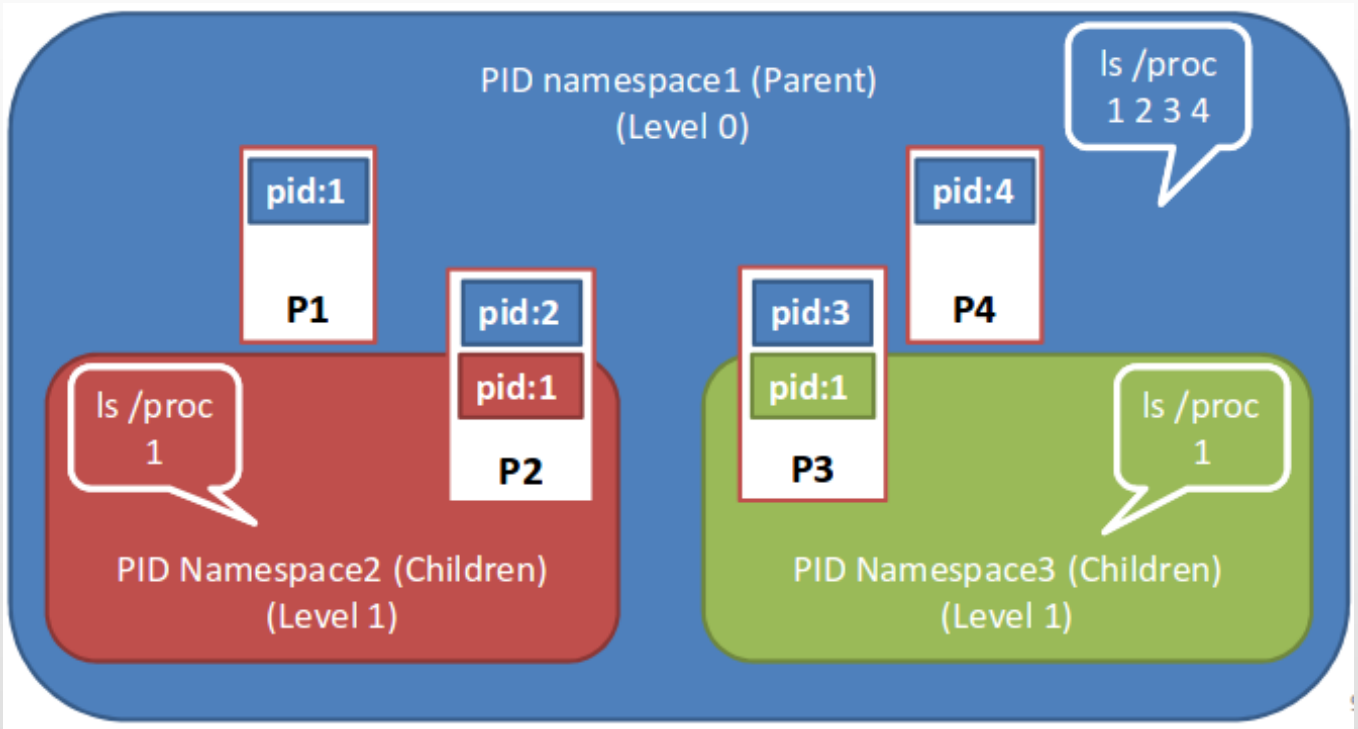
- echo "0-2,4" > /sys/fs/cgroup/cpuset/my_group/cpuset.cpus

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

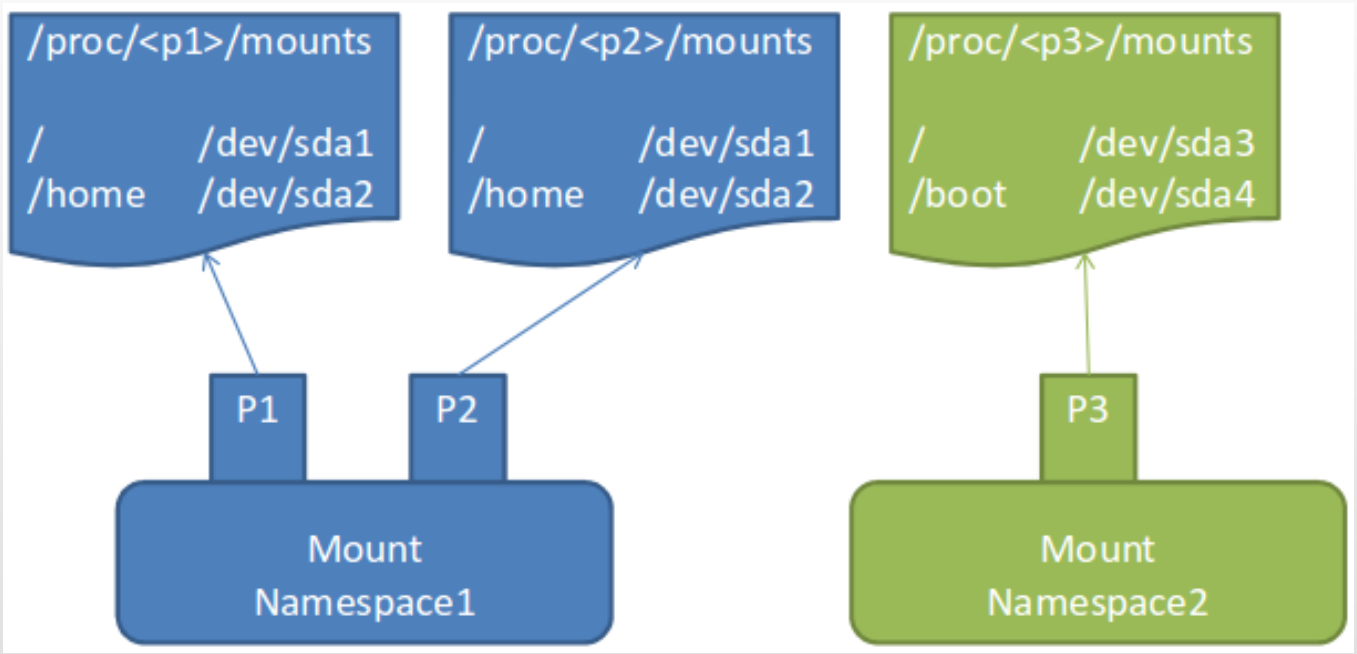
- Modificaciones a un recurso quedan contenidas dentro del "namespace"
- Entre los namespaces provistos por Linux:
  - IPC: System V IPC, cola de mensaje POSIX
  - Network: dispositivos de red, pilas, puertos, etc
  - Mount: puntos de montaje
  - PID: IDs de procesos
  - User: IDs de usuarios y grupos
  - UTS: HostName y nombre de dominio

- Posibilidad de tener múltiples árboles de procesos anidados y asilados



Fuente imagen: http://events.linuxfoundation.org/sites/events/files/cojp13_feng.pdf

- Permite aislar la tabla de montajes (montajes por namespace)
- Usado típicamente con "chroot"



Fuente imagen: http://events.linuxfoundation.org/sites/events/files/cojp13_feng.pdf

Agenda

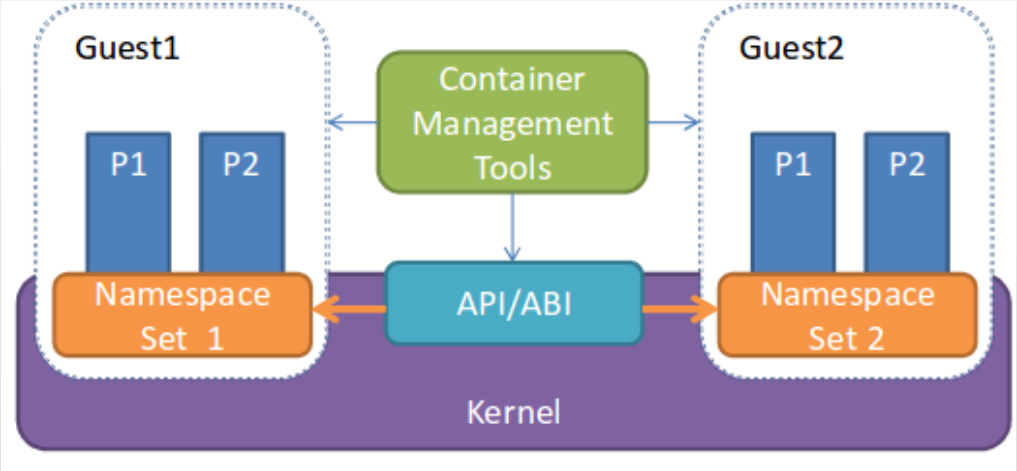
1. Linux cgroups y Namespaces
2. Linux Containers

Operating System Virtualization

- Tecnología liviana de virtualización (lightweight virtualization) a nivel de sistema operativo que permite ejecutar múltipes sistemas aislados (conjutos 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)

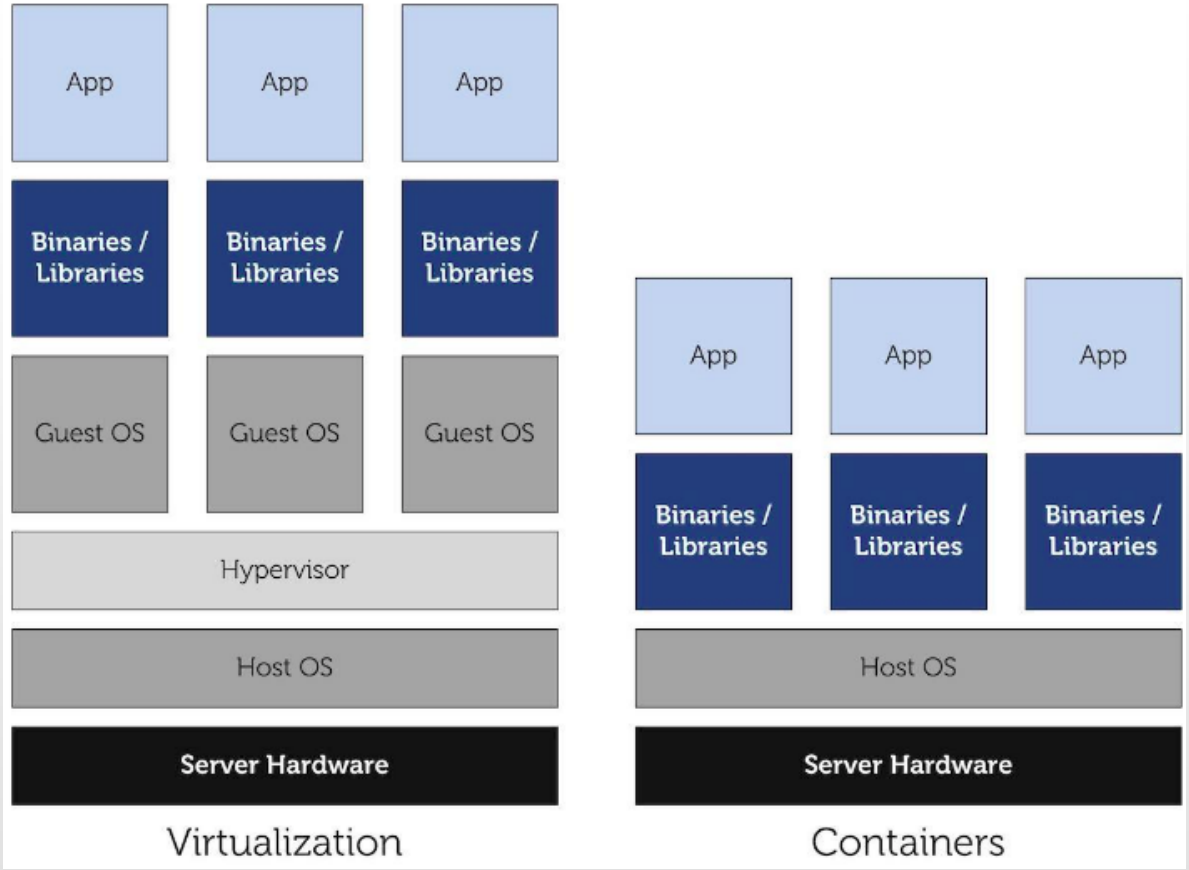
- Proyecto comenzado a mediados de 2008
- Utiliza las características de “cgroups” del kernel de Linux, “namespace isolation” y “chroot”
- Docker utilizaba LXC en su orígenes (luego pasó a sus propias librerías llamadas libcontainer)
- En 2015, Ubuntu lanzó LXD que permite desplegar y administrar containters 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

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: lxctl, 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
....
```

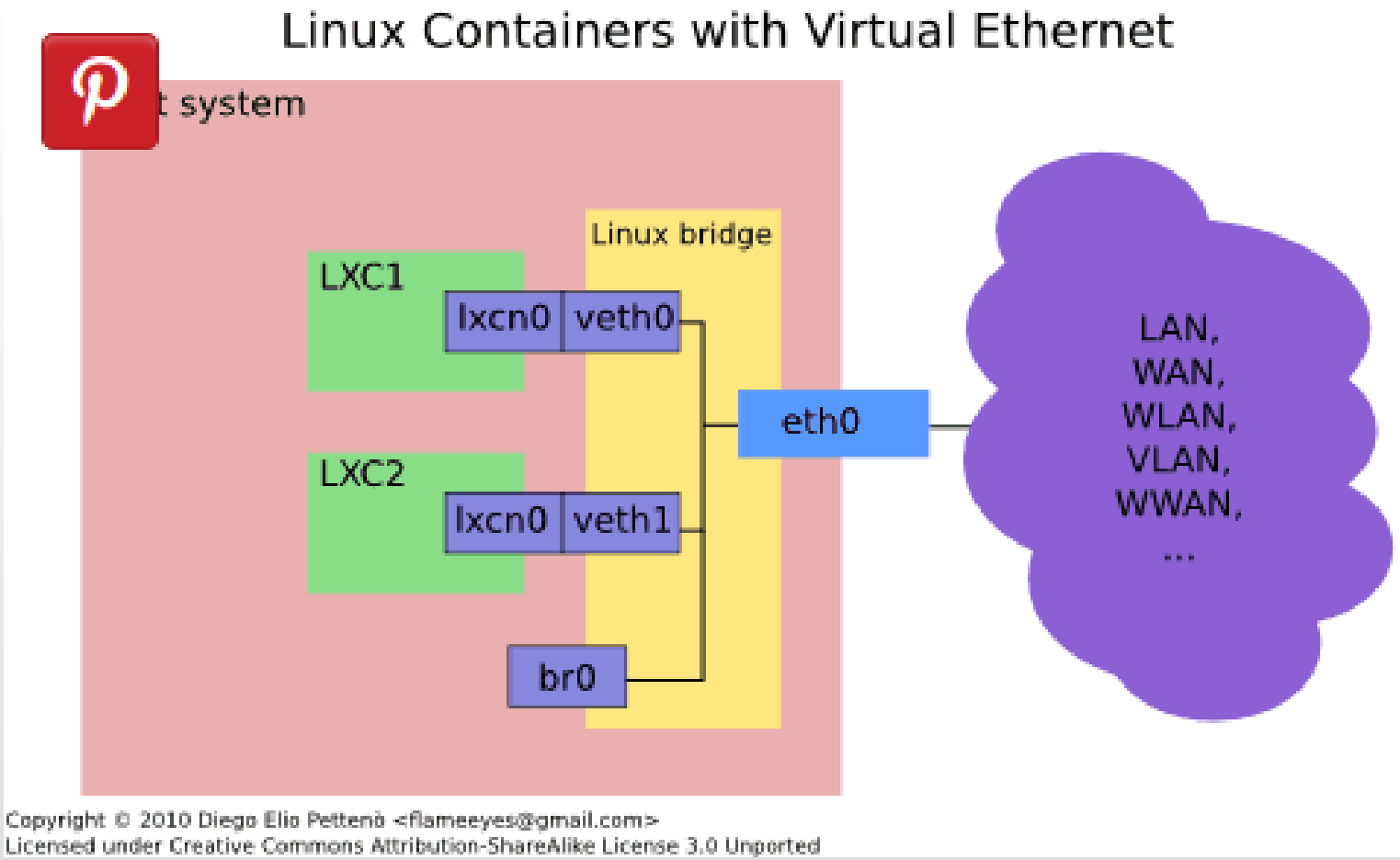
- *lxc-create* permite crear un contenedor:
  - lxc-create -n "nombre contenedor" - t "template"
  - Por ej.: lxc-create -n lxcdebian -t debian
- *lxc-start -n lxcdebian -d* para iniciar un container ("d" ejecuta como daemon)
- *lxc-info -n "nombre contenedor"* para obtener información sobre un container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

- Procesos del container ejectuan como procesos dentro del SO anfitrión

```
so2018@debian:/$ sudo lxc-info --name testlxc
Name:           testlxc
State:          RUNNING
PID:            10131
CPU use:        0.89 seconds
BlkIO use:      0 bytes
```

```
so2018@debian:/$ ps -ef | grep 10127
root     10127     1  0 12:58 ?        00:00:00 lxc-start -n testlxc -d

so2018@debian:/$ ps -ef | grep 10131
root     10131 10127  0 12:58 ?        00:00:00 /sbin/init
root     10151 10131  0 12:58 ?        00:00:00 /lib/systemd/systemd-journald
root     10213 10131  0 12:58 ?        00:00:00 /usr/sbin/sshd -D
root     10219 10131  0 12:58 pts/4    00:00:00 /sbin/agetty --noclear tty4 linux
root     10220 10131  0 12:58 pts/2    00:00:00 /sbin/agetty --noclear tty2 linux
root     10221 10131  0 12:58 pts/3    00:00:00 /sbin/agetty --noclear tty3 linux
root     10222 10131  0 12:58 pts/5    00:00:00 /sbin/agetty --noclear --keep-baud console
root     10728 10131  0 13:15 pts/1    00:00:00 /sbin/agetty --noclear tty1 linux
root     12723 10131  0 14:16 ?        00:00:00 /sbin/agetty --noclear tty6 linux
root     12724 10131  0 14:16 ?        00:00:00 /sbin/agetty --noclear tty5 linux
```



- Crear un container a partir de un archivo de configuración
- lxc-create -t debian -f file.conf -n “nombre contenedor”

```
so2018@debian:/$ more file.conf
lxc.utsname = so-lxc
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.hwaddr = 00:30:6E:08:EC:80
lxc.network.ipv4 = 192.168.1.10/24
lxc.network.name = eth0
```

- br0 indica el bridge al cual se une el container
- Debe estar creado antes de generar el container

```
so2018@debian:/$ apt-get install bridge-utils
so2018@debian:/$ brctl addbr br0
```

- 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

¿Preguntas?