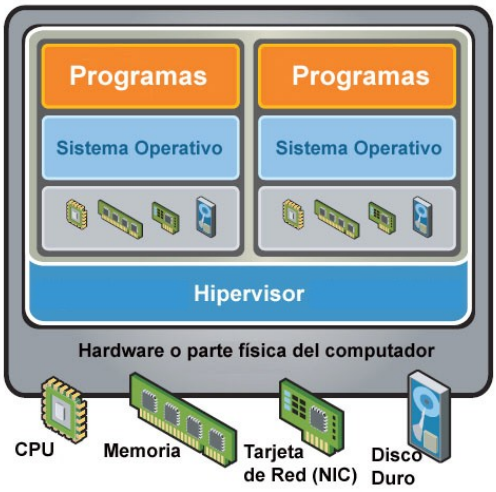
1. Virtualización

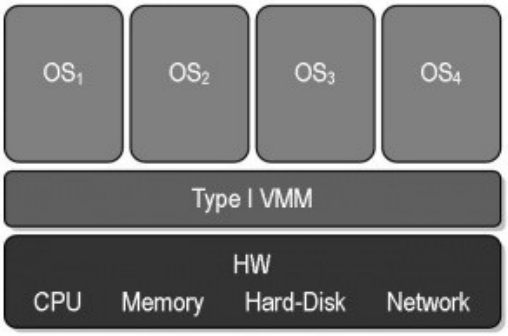


La virtualización permite crear mediante software una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red.

Una máquina virtual (MV) es un ordenador virtual creado dentro de un ordenador físico y que tiene su disco duro (un fichero que se guarda en el disco duro real), su memoria RAM (que toma de la memoria física real), sus dispositivos de E/S (dispositivos virtuales que se enlazan con los reales), su BIOS (que nada tiene que ver con la BIOS del sistema real), su sistema operativo (que nada tiene que ver con el de la máquina física y que se instala como en un ordenador cualquiera).

El software de virtualización implementa lo que se llama un **Hipervisor o VMM** (*Virtual Machine Monitor*) que consiste en una capa de abstracción entre el hardware de la máquina física (**host o anfitrión**) y la máquina virtual (**MV, invitado o guest**) formada por hardware y software virtualizado, haciendo el papel de intermediario entre lo real y lo virtualizado. Esta capa de software, hypervisor o VMM, maneja, gestiona y arbitra los cuatro recursos principales de un ordenador (CPU, memoria, almacenamiento y conexiones de red) y así podrá repartir dinámicamente dichos recursos entre todas las MVs creadas en el ordenador anfitrión. Existen dos tipos de hypervisores: Tipo I y Tipo II.

Hypervisor de Tipo I

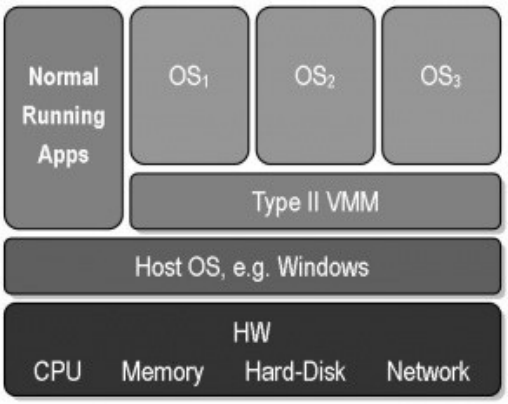


En este caso se instala un S.O. especialmente diseñado para la virtualización. Esta plataforma está en contacto directo con el hardware del ordenador físico actuando de intermediario entre éste y las MVs. Por tanto, en el ordenador físico no se instala un S.O. de los conocidos (Windows, GNU/Linux, OSX) y después se instala un programa para virtualizar, sino que el S.O. que se instala en el ordenador físico es ya la herramienta para virtualizar.

Este tipo de virtualización es la más óptima y está pensada para la virtualización de servidores, pero como contrapartida, el equipo físico únicamente se podrá usar para virtualizar y no para instalar programas.

Las plataformas de Tipo I más usadas son Vmware vSphere Hypervisor (ESXi), Citrix Xen Server, Microsoft Hper-V y Proxmox (KVM).

Hypervisor de Tipo II



El anfitrión o host es un ordenador con un S.O. convencional (GNU/Linux, Windows, MAC OS) sobre el que se instala un programa de virtualización. Para iniciar las Mvs es necesario

iniciar el programa de virtualización, como quien abre cualquier aplicación, y después iniciar la máquina que se desee.

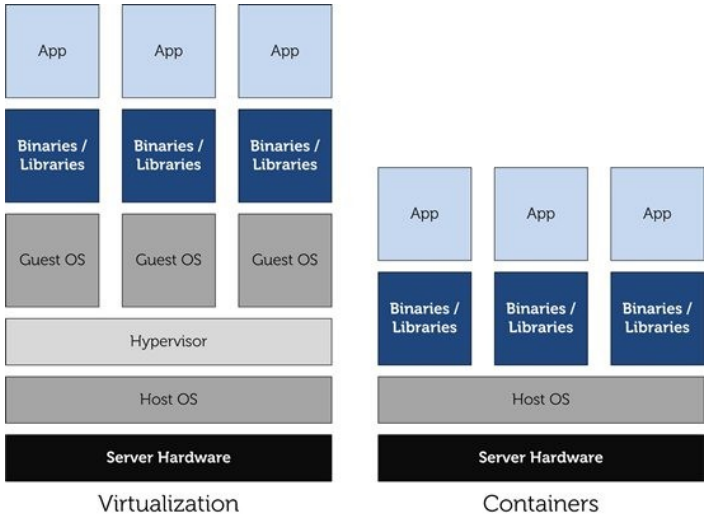
Esta forma de virtualizar es menos eficiente que el Tipo I, pero se puede seguir usando el equipo físico con las demás aplicaciones (editor de textos, navegador, etc.).

Las plataformas de Tipo II más usadas son Vmware WorkStation, Virtualbox, qemu-KVM.

2. Virtualización con contenedores

LXC (Linux Containers) es una tecnología de virtualización a nivel de sistema operativo para Linux, permitiendo que un servidor físico ejecute múltiples instancias de sistemas operativos Linux aislados, cada uno con su propio sistema de ficheros, espacio de procesos y redes. Puede verse la máquina virtual (contenedor) como si una parte del sistema operativo corriese aislada del resto, como una jaula o chroot.

Cuando se habla de LXC aparece muchas veces como “lightweight virtualization” o virtualización ligera. Al trabajar con un contenedor trabajamos con un sistema aislado, pero no estamos a instalar un sistema operativo separado para el contenedor ni precisamos de un hypervisor para correrlo. En la imagen puede compararse la virtualización tradicional con los contenedores.



Un contenedor LXC está aislado y corre sus propias aplicaciones/procesos; sin embargo comparte con el 'equipo real' ciertas cosas, como el kernel y ciertos binarios/librerías. Esto se traduce en una menor carga y consumo de recursos y en un mayor rendimiento. El principal objetivo de los contenedores LXC es crear entornos Linux lo más parecido posible a una instalación Linux estándar sin la necesidad de correr un kernel separado o parcheado como ocurre con otros sistemas de virtualización como Xen o OpenVZ.

Por la forma de trabajar de LXC, una misma máquina física permite correr muchos más contenedores que máquinas virtuales tradicionales:

	Virtualization (i.e. kvm, xen)	**LXC Containers**
Footprint	Requires a hypervisor and a full operating system image.	Does not require a hypervisor or a separate operating system image.
OS supported	Any OS supported by the hypervisor	Most Linux distros, uses same kernel as host
Typical server deployment	10 – 100 VMs	100 - 1000 containers
Boot time	Less than a minute	Seconds
Physical resources use (i.e. memory, CPU)	Each VM has resource reserved for its own use	Shared by all containers

Fig. Comparativa entre máquinas virtuais e contenedores

En la OpenStack Developer Summit de 2015 en Vancouver se presentó un estudio comparando LXC con KVM donde se decía que en un mismo servidor físico, los contenedores permitían 14,5 veces más sistemas virtuales que la solución de virtualización KVM (536 frente a 37), arrancaban un 94% más rápido (1,5 segundos frente a 25) y con un 57% menos de latencia. Canonical, la empresa tras Ubuntu, usa contenedores desde hace tiempo para pruebas de desarrollo y lanzó a finales de 2015 LXD, que trata de proporcionar una mejor interfaz para trabajar con contenedores LXC.

Si comparamos LXC con la tecnología de contenedores Docker, la primera pretende proporcionar un sistema linux completo aislado pero corriendo en el equipo anfitrión y la segunda está orientada a aislar procesos (p.e. un servidor apache sería un contenedor y MySQL sería otro).

3. Instalación e inicialización

Storage Backends

LXD puede almacenar los contenedores e imágenes en directorios (*storage* de tipo dir) y de esta forma todo lo relativo a un contenedor está asociado a una carpeta. Se trata de la solución más sencilla; sin embargo, comparada con las otras alternativas es lenta y poco eficiente a la hora de crear clones e instantáneas al tener que copiar todo el sistema de ficheros del contenedor. LXD puede trabajar con sistemas de ficheros como ZFS, btrfs, LVM y CEPH aprovechando, cuando sea posible, sus características avanzadas para obtener un mejor rendimiento.

En la siguiente table puede verse una comparativa de características:

Funcionalidad	**Directorio**	**Btrfs**	**LVM**	**ZFS**	**CEPH**
Almacenamiento optimizado de imágenes	No	Si	Si	Si	Si
Creación optimizada de contenedores	No	Si	Si	Si	Si
Creación optimizada de instantáneas	No	Si	Si	Si	Si
Copy on write	No	Si	Si	Si	Si
Cuotas de almacenamiento	No	Si	Si	Si	No
Clonación instantánea	No	Si	Si	Si	Si
Restauración desde instantáneas anteriores (no la última)	Si	Si	Si	No	Si

Tabla. Características de los Storage Backends.

Los desarrolladores de LXD señalan a ZFS y Btrfs como mejores opciones; y si el sistema tiene soporte para ZFS, esta sería la mejor opción. Otra recomendación de los desarrolladores es la de dedicar un disco entero o una partición para el *storage pool*. LXD permite crear '*loop based storage*' (archivo de un tamaño determinado que funciona como un disco con el sistema de ficheros deseado) pero no la recomiendan para sistemas en producción.

Instalando LXD en Ubuntu 20.04

Instalamos LXD vía paquete snap:

```
$ sudo snap install lxd
```

Inicializando LXD para trabajar en un segundo disco duro/partición

Para hacer la configuración inicial de LXD ejecutamos el comando `lxd init` e indicamos que:

- La instalación no forma parte de un cluster LXD.
- Queremos crear un nuevo *storage backend* (donde se almacenarán los contenedores e las imágenes) de tipo ZFS usando un dispositivo de bloque ya existente (el segundo disco duro o partición).
- No queremos conectarnos a un servidor MAAS[1].
- Queremos crear una nueva interfaz de red lxdbr0, que actuará como un switch-router-nat que dará salida al exterior a los contenedores (semejante al modo red NAT de Virtualbox).
- No queremos que el servicio LXD sea accesible por red desde otros equipos.
- Queremos que las imágenes (plantillas a partir de las que se crean los contenedores) se actualice automáticamente.

Aceptamos las opciones por defecto salvo en la configuración del *storage backend* donde indicamos la ruta a segundo disco duro (o partición) reservado para los contenedores.

```
$ sudo lxd init
[sudo] password for magasix:
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (btrfs, ceph, dir, lvm, zfs) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]: yes
Path to the existing block device: /dev/sdb
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
$
```

Fijarse que al tener instalado el soporte para ZFS el tipo por defecto para el *storage backend* es ZFS y que creamos un nuevo pool ZFS empleando un disco duro (*block device*) llamado /dev/sdb. Indicamos en lxd init que se cree un nuevo espacio de almacenamiento ZFS en el disco duro; por supuesto, los pools ZFS pueden crearse con otras herramientas y posteriormente configurar lxd para emplearlos.

Para terminar, los usuarios que van a trabajar con LXD tienen que pertenecer al grupo lxd por lo que se procede a añadir nuestro usuario a ese grupo:

```
$ sudo usermod -a -G lxd <nome_do_usuario>
```

1 https://maas.io/

Como la información sobre los grupos a los que pertenece un usuario se lee en el momento de iniciar la sesión, puede ser necesario salir y volver a entrar.

Inicializando LXD para trabajar en un Loop based storage

Para hacer la configuración inicial de LXD ejecutamos el comando lxd init e indicamos que:

- La instalación no forma parte de un cluster LXD.
- Queremos crear un nuevo *storage backend* (donde se almacenarán los contenedores y las imágenes) de tipo ZFS sin usar un dispositivo de bloque ya existente (un segundo disco duro o partición). Fijarse que al tener instalado el soporte para ZFS el tipo por defecto para el *storage backend* es ZFS.
- Indicamos el tamaño para el *loop device* que será el *storage backend*.
- No queremos conectarnos a un servidor MAAS.
- Queremos crear una nueva interfaz de red lxdbr0, que actuará como un switch-router-nat que dará salida al exterior a los contenedores (semejante al modo red NAT de Virtualbox).
- No queremos que el servicio LXD sea accesible por red desde otros equipos.
- Queremos que las imágenes (plantillas a partir de las que se crean los contenedores) se actualice automáticamente.

Aceptamos las opciones por defecto salvo en la configuración del *storage backend* donde indicaremos un tamaño adecuado para el *loop device*, acorde al tamaño del disco duro del equipo. En mi caso, el nuevo *storage backend* de tipo ZFS de tipo *loop based storage* es de 26 GB:

```
$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (btrfs, ceph, dir, lvm, zfs) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]:
Size in GB of the new loop device (1GB minimum) [default=15GB]: 26
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
$
```

Para terminar, los usuarios que van a trabajar con LXD tienen que pertenecer al grupo lxd por lo que se procede a añadir nuestro usuario a ese grupo:

```
$ sudo usermod -a -G lxd <nome_do_usuario>
```

Como la información sobre los grupos a los que pertenece un usuario se lee en el momento de iniciar la sesión, puede ser necesario salir y volver a entrar.

4. Operaciones básicas con LXD

Storage backends

Podemos conocer información sobre el pool ZFS creado durante la instalación. Si trabajamos con un disco duro/partición reservada para ZFS tendremos:

```
$ lxc storage list
+---------+-------------+--------+---------+---------+
|  NAME   | DESCRIPTION | DRIVER | SOURCE  | USED BY |
+---------+-------------+--------+---------+---------+
| default |             | zfs    | default | 1       |
+---------+-------------+--------+---------+---------+
```

Si trabajamos con un *loop based storage* puede verse que es un archivo y no un disco duro/partición:

```
$ lxc storage list
+---------+-------------+--------+-----------------------------------------+---------+
|  NAME   | DESCRIPTION | DRIVER |                 SOURCE                  | USED BY |
+---------+-------------+--------+-----------------------------------------+---------+
| default |             | zfs    | /var/snap/lxd/common/lxd/disks/default.img | 1       |
+---------+-------------+--------+-----------------------------------------+---------+
```

Para ver información sobre el estado del pool ZFS:

```
$ sudo zpool list
NAME      SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
default    26G   840M  25,2G         -     1%     3%  1.00x  ONLINE  -
$ sudo zpool status
  pool: default
 state: ONLINE
  scan: none requested
config:

    NAME                                        STATE     READ WRITE CKSUM
    default                                     ONLINE       0     0     0
      /var/snap/lxd/common/lxd/disks/default.img  ONLINE       0     0     0

errors: No known data errors
$ sudo zfs list
NAME                    USED  AVAIL  REFER  MOUNTPOINT
default                 729K  25,2G    24K  none
default/containers       24K  25,2G    24K  none
default/custom           24K  25,2G    24K  none
default/deleted          48K  25,2G    24K  none
```

```
default/deleted/images    24K  25,2G    24K  none
default/images            24K  25,2G    24K  none
default/snapshots         24K  25,2G    24K  none
$
```

Repositorios de imágenes para contenedores LXD

Los contenedores se lanzan a partir de imágenes alojadas en repositorios locales o remotos; es decir, que cuando creamos un contenedor se usa una de esas imágenes para construir una nueva 'máquina virtual' corriendo la distribución escogida. LXD permite gestionar los contenedores, las imágenes y los repositorios. Para ver los repositorios disponibles ejecutamos el comando:

$ lxc remote list

```
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
|      NAME       |                  URL                    |   PROTOCOL    |  AUTH TYPE  | PUBLIC | STATIC | GLOBAL |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| images          | https://images.linuxcontainers.org      | simplestreams | none        | YES    | NO     | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| local (current) | unix://                                 | lxd           | file access | NO     | YES    | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| ubuntu          | https://cloud-images.ubuntu.com/releases | simplestreams | none        | YES    | YES    | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| ubuntu-daily    | https://cloud-images.ubuntu.com/daily   | simplestreams | none        | YES    | YES    | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
```

Tenemos tres repositorios remotos y uno local (en la propia máquina donde está corriendo lxd). Si queremos ver las imágenes disponibles en un repositorio ejecutamos el comando: `lxc image list <repositorio>`: Por ejemplo, para ver las imágenes disponibles en el repositorio local:

$ lxc image list local:

```
+-------+-------------+--------+-------------+-----+--------+-------------+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPCIÓN | ARQ | TAMAÑO | UPLOAD DATE |
+-------+-------------+--------+-------------+-----+--------+-------------+
```

Podemos apreciar que aún no tenemos ninguna imagen en el repositorio local; sin embargo, en el repositorio oficial de Ubuntu si hay una gran cantidad de ellas. Para poder verlas con calma añadimos al comando un | less :

$ lxc image list ubuntu: | less

```
+-----------------+-------------+--------+-----------------------------------------------+-------+-------
| f (11 more)         | 37aad8824a20 | yes | ubuntu 20.04 LTS amd64 (release) (20211108)  | x86_64 | VIRTUAL-MACHINE |
522.63MB | Nov 8, 2021 at 12:00am (UTC)  |
+-----------------+-------------+--------+-----------------------------------------------+-------------
| f (11 more)      | bd2ffb937c95 | yes    | ubuntu 20.04 LTS amd64 (release) (20211108)   | x86_64  | CONTAINER       |
370.58MB | Nov 8, 2021 at 12:00am (UTC)  |
+------------------+-------------+--------+-----------------------------------------------+-------------
```

Fijándonos en la línea resaltada podemos comprobar que se trata de una imagen para crear contenedores Ubuntu 20.04 LTS (Focal Fossa) de 64 bits. De estar interesados en otras distribuciones podemos recurrir al repositorio *images* donde hay imágenes de Debian, Alpine, Fedora, Ubuntu, ...

Creando nuestro primer contenedor

Una vez visto que tenemos a nuestra disposición las imágenes en los repositorios remotos procederemos a crear un contenedor. Nuestro contenedor se llamará focal00 y será un Ubuntu 20.04 LTS de 64 bits creado a partir de la imagen del repositorio oficial de Ubuntu:

```
$ lxc launch ubuntu:f focal00
Creating focal00
Starting focal00
```

Lo que sucedió fue lo siguiente:

- Accedemos al repositorio ubuntu (https://cloud-images.ubuntu.com/releases) a buscar una imagen llamada f (fijarse en la columna ALIAS del listado de imágenes del repositorio Ubuntu). En vez de usar el ALIAS también podemos usar el FINGERPRINT para indicar la imagen en la que estamos interesados: `lxc launch ubuntu:bd2ffb937c95 focal00`
- Se procede a descargarla al equipo local.
- Después se usa para crear un contenedor llamado focal00.

Una vez terminado el proceso podemos ver que el contenedor fue creado y está operativo, recibiendo la configuración de red desde el 'switch-router-nat' lxdbr0 creado en el lxd init:

```
$ lxc ls
+------------+---------+----------------------+----------------------------------------------+-----------+-----------+
|    NAME    |  STATE  |         IPV4         |                    IPV6                      |   TYPE    | SNAPSHOTS |
+------------+---------+----------------------+----------------------------------------------+-----------+-----------+
| focal00    | RUNNING | 10.122.236.78 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe8a:7c7f (eth0) | CONTAINER | 0         |
+------------+---------+----------------------+----------------------------------------------+-----------+-----------+
```

En este momento tenemos funcionando un contenedor que a efectos prácticos es equivalente a una máquina virtual Ubuntu Server 20.04 de 64 bits con su propio sistema de ficheros, configuraciones, aplicaciones, ... Es muy interesante comparar como ve el sistema operativo anfitrión una máquina virtual tradicional y un contenedor. En la siguiente captura, puede verse una máquina virtual de Virtualbox que es vista por el sistema como un proceso (dentro de la máquina virtual habrá sus correspondientes procesos, pero no son visibles para el equipo anfitrión):

```
$ ps auxf
...
manuel    6734  0.3  0.8 1092720 71072 ?       Sl   17:56   0:37 /usr/lib/virtualbox/VirtualBox
manuel    6753  0.1  0.1 169120 12208 ?        S    17:56   0:19 /usr/lib/virtualbox/VBoxXPCOMIPCD
manuel    6758  0.3  0.3 1161020 30160 ?       Sl   17:56   0:39 /usr/lib/virtualbox/VBoxSVC --auto-shutdown
manuel    8839 34.3  4.8 2549708 389648 ?      Sl   20:52   0:11  \_ /usr/lib/virtualbox/VirtualBox --
comment u16-LXD --startvm aa3f0392-32fe-40fa-8447-2f8ebe8de6d4 --no-startvm-errormsgbox
...
```

En cambio, un contenedor es visto como un conjunto de procesos que se están ejecutando por el propio equipo. Los procesos del contenedor se ejecutan directamente en el hardware del anfitrión, de forma aislada pero sin capa de virtualización como en Virtualbox; por lo que, la velocidad es mayor y hay un mejor aprovechamiento de los recursos.

```
$ ps auxf
...
```

```
magasix   19001   0.0   0.6 502664   6560 ?          Sl    19:00    0:00 /usr/lib/x86_64-linux-gnu/indicator-application-service
root       21616  0.0  0.0 2162176 18472 ?       Ss   10:48   0:00 [lxc monitor] /var/snap/lxd/common/lxd/containers focal00
1000000    21642  0.2  0.0 170344  9212 ?        Ss   10:48   0:00  \_ /sbin/init
1000000    22498  0.0  0.0  35088  9040 ?        Ss   10:48   0:00      \_ /lib/systemd/systemd-journald
1000000    22549  0.0  0.0  11884  3388 ?        Ss   10:48   0:00      \_ /lib/systemd/systemd-udevd
1000000    22573  0.1  0.0   3868  1368 ?        Ss   10:48   0:00      \_ snapfuse /var/lib/snapd/snaps/core20_1169.snap /snap/core20/1169 -o ro,nodev,allow_other,suid
1000000    22575  0.7  0.0   3728  1460 ?        Ss   10:48   0:03      \_ snapfuse /var/lib/snapd/snaps/snapd_13640.snap /snap/snapd/13640 -o ro,nodev,allow_other,suid
1000000    22576  0.0  0.0   3816  1220 ?        Ss   10:48   0:00      \_ snapfuse /var/lib/snapd/snaps/lxd_21835.snap /snap/lxd/21835 -o ro,nodev,allow_other,suid
1000100    22668  0.0  0.0  26604  5356 ?        Ss   10:48   0:00      \_ /lib/systemd/systemd-networkd
1000101    22670  0.0  0.0  23960  9560 ?        Ss   10:48   0:00      \_ /lib/systemd/systemd-resolved
1000000    22716  0.0  0.0 241032  5720 ?        Ssl  10:48   0:00      \_ /usr/lib/accountsservice/accounts-daemon
1000000    22720  0.0  0.0   8536  1704 ?        Ss   10:48   0:00      \_ /usr/sbin/cron -f
1000103    22722  0.0  0.0   7440  2956 ?        Ss   10:48   0:00      \_ /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
1000000    22728  0.0  0.0  29272 14624 ?        Ss   10:48   0:00      \_ /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
1000104    22729  0.0  0.0 154836  3092 ?        Ssl  10:48   0:00      \_ /usr/sbin/rsyslogd -n -iNONE
1000000    22730  0.5  0.0 4398100 33400 ?       Ssl  10:48   0:02      \_ /usr/lib/snapd/snapd
1000000    22732  0.0  0.0  16656  5068 ?        Ss   10:48   0:00      \_ /lib/systemd/systemd-logind
1000000    22734  0.0  0.0 394568  8552 ?        Ssl  10:48   0:00      \_ /usr/lib/udisks2/udisksd
1000001    22735  0.0  0.0   3792  1516 ?        Ss   10:48   0:00      \_ /usr/sbin/atd -f
1000000    22754  0.0  0.0  12176  4380 ?        Ss   10:48   0:00      \_ sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
1000000    22758  0.0  0.0   7352  1412 pts/0    Ss+  10:48   0:00      \_ /sbin/agetty -o -p -- \u --noclear --keep-baud console 115200,38400,9600 linux
1000000    22762  0.0  0.0 236420  5932 ?        Ssl  10:48   0:00      \_ /usr/lib/policykit-1/polkitd --no-debug
1000000    22777  0.0  0.0 108176 16304 ?        Ssl  10:48   0:00      \_ /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signalupgrades/unattended-upgrade-shutdown --wait-for-signal
...
```

Creando nuestro segundo contenedor

En el punto anterior descargamos la imagen plantilla de Ubuntu 20.04 de 64 bits para crear el contenedor focal00. Como esta descarga ralentiza el proceso de creación de contenedores, automáticamente se procedió a guardar una copia de esta imagen en el repositorio local. La próxima vez que creemos contenedores Ubuntu 20.04, ya no descargaremos la imagen desde Internet y se empleará la copia local, acelerando todo el proceso.

$ lxc image list local:

```
+----------+--------------+--------+-------------------------------------------+--------------+-----------+----------+------------+
|  ALIAS   | FINGERPRINT  | PUBLIC |                DESCRIPTION                | ARCHITECTURE |   TYPE    |   SIZE   | UPLOAD DATE |
+----------+--------------+--------+-------------------------------------------+--------------+-----------+----------+------------+
|          | bd2ffb937c95 | no     | ubuntu 20.04 LTS amd64 (release) (20211108) | x86_64       | CONTAINER | 370.58MB | Nov 11, 2021 |
+----------+--------------+--------+-------------------------------------------+--------------+-----------+----------+------------+
```

$ lxc launch bd2ffb937c95 focal01

```
Creating focal01
Starting focal01
```

$ lxc ls

```
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
|    NAME    |  STATE  |         IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
```

```
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
| focal00    | RUNNING | 10.122.236.78 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe8a:7c7f (eth0) | CONTAINER | 0         |
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
| focal01    | RUNNING | 10.122.236.112 (eth0)| fd42:9345:33a5:8cc9:216:3eff:fe98:126b (eth0) | CONTAINER | 0         |
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
```

Poniendo un alias a las imágenes

Es posible asignar un nombre (*alias*) a las imágenes en vez de trabajar con el *fingerprint* a la hora de crear contenedores. Podemos consultar la ayuda del comando lxc image escribiendo:

$ lxc image

Vamos a crear un alias para a imagen que tenemos y le llamaremos trusty64 para después usarlo para crear un nuevo contenedor:

$ lxc image list local:

```
+---------+-------------+--------+----------------------------------------+-------------+-----------+---------+------------+
|  ALIAS  | FINGERPRINT | PUBLIC |               DESCRIPTION              | ARCHITECTURE |   TYPE    |   SIZE  | UPLOAD DATE
+---------+-------------+--------+----------------------------------------+-------------+-----------+---------+------------+
|         | bd2ffb937c95 | no     | ubuntu 20.04 LTS amd64 (release) (20211108) | x86_64      | CONTAINER | 370.58MB | Nov 11, 2021
+---------+-------------+--------+----------------------------------------+-------------+-----------+---------+------------+
```

$ lxc image alias create focal64 bd2ffb937c95

$ lxc image list local:

```
+---------+-------------+--------+----------------------------------------+--------+---------+----------------------------+
+---------+-------------+--------+----------------------------------------+-------------+-----------+---------+------------+
|  ALIAS  | FINGERPRINT | PUBLIC |               DESCRIPTION              | ARCHITECTURE |   TYPE    |   SIZE  | UPLOAD DATE
+---------+-------------+--------+----------------------------------------+-------------+-----------+---------+------------+
|  focal64 | bd2ffb937c95 | no     | ubuntu 20.04 LTS amd64 (release) (20211108) | x86_64      | CONTAINER | 370.58MB | Nov 11, 202)
+---------+-------------+--------+----------------------------------------+-------------+-----------+---------+------------+
```

$ lxc launch focal64 focal02

```
Creating focal02
Starting focal02
```

$ lxc ls

```
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
|    NAME    |  STATE  |         IPV4        |                     IPV6                     |   TYPE    | SNAPSHOTS |
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
| focal00    | RUNNING | 10.122.236.78 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe8a:7c7f (eth0) | CONTAINER | 0         |
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
| focal01    | RUNNING | 10.122.236.112 (eth0)| fd42:9345:33a5:8cc9:216:3eff:fe98:126b (eth0) | CONTAINER | 0         |
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
| focal02    | RUNNING | 10.122.236.50 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe04:c3f3 (eth0) | CONTAINER | 0         |
+------------+---------+---------------------+----------------------------------------------+-----------+-----------+
```

Trabajando con contenedores de diferentes versiones/distros

Es posible crear y trabajar con contenedores de diferentes versiones/distribuciones. A modo de ejemplo, vemos como crear un contenedor a partir de la imagen de Ubuntu 16.04 (ubuntu:x) y como crear un alias para la imagen descargada:

$ lxc launch ubuntu:x xenial00

```
Creando xenial00
Iniciando xenial00
```

$ lxc ls

```
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
|     NAME   |  STATE  |         IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| focal00    | RUNNING | 10.122.236.78 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe8a:7c7f (eth0) | CONTAINER | 0         |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| focal01    | RUNNING | 10.122.236.112 (eth0)| fd42:9345:33a5:8cc9:216:3eff:fe98:126b (eth0) | CONTAINER | 0         |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| focal02    | RUNNING | 10.122.236.50 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe04:c3f3 (eth0) | CONTAINER | 0         |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
```

$ lxc image list local:

```
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
|  ALIAS   | FINGERPRINT | PUBLIC |                 DESCRIPCIÓN                  |  ARQ   |  TAMAÑO  |          UPLOAD DATE         |
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
| trusty64 | 5b8c11faa8e2 | no     | ubuntu 14.04 LTS amd64 (release) (20180913) | x86_64 | 121.77MB | Oct 6, 2018 at 5:36pm (UTC) |
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
|          | c966933fdfd3 | no     | ubuntu 1604 LT.S amd64 (release) (20181004) | x86_64 | 157.99MB | Oct 6, 2018 at 6:09pm (UTC) |
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
```

$ lxc image alias create xenial64 c966933fdfd3

$ lxc image list local:

```
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
|  ALIAS   | FINGERPRINT | PUBLIC |                 DESCRIPCIÓN                  |  ARQ   |  TAMAÑO  |          UPLOAD DATE         |
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
| trusty64 | 5b8c11faa8e2 | no     | ubuntu 14.04 LTS amd64 (release) (20180913) | x86_64 | 121.77MB | Oct 6, 2018 at 5:36pm (UTC) |
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
| xenial64 | c966933fdfd3 | no     | ubuntu 16.04 LTS amd64 (release) (20181004) | x86_64 | 157.99MB | Oct 6, 2018 at 6:09pm (UTC) |
+----------+-------------+--------+----------------------------------------------+--------+----------+------------------------------+
```

También podemos crear contenedores corriendo otras distribuciones, por ejemplo Alpine Linux:

$ lxc launch images:alpine/3.8 alpine00

```
Creating alpine00
Starting alpine00
```

$ lxc ls

```
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
|     NAME   |  STATE  |         IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| focal00    | RUNNING | 10.122.236.78 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe8a:7c7f (eth0) | CONTAINER | 0         |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| focal01    | RUNNING | 10.122.236.112 (eth0)| fd42:9345:33a5:8cc9:216:3eff:fe98:126b (eth0) | CONTAINER | 0         |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| focal02    | RUNNING | 10.122.236.50 (eth0) | fd42:9345:33a5:8cc9:216:3eff:fe04:c3f3 (eth0) | CONTAINER | 0         |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| xenial00   | RUNNING | 10.122.236.3 (eth0)  | fd42:c0b9:6787:9563:216:3eff:fe12:acbf (eth0) | CONTAINER |           |
+------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
```

Arrancar y parar los contenedores

Los contenedores se pueden parar usando el comando `lxc stop <contenedor>`

$ lxc stop focal00

Para arrancar contenedores se usa el comando `lxc start <contenedor>`

```
$ lxc start focal00
```

Si en vez de un único contenedor indicamos varios, es posible arrancar/parar varios contenedores a la vez:

```
$ lxc stop focal00 focal01 focal02 xenial00
$ lxc start focal00 focal01 focal02 xenial00
```

Borrar contenedores

Para eliminar un contenedor hay que pararlo primero y después proceder a su borrado con `lxc delete` o forzar el borrado sin apagarlo con `lxc delete -f <contenedor>`

```
$ lxc stop focal01
$ lxc delete focal01
```

Ejemplo de como forzar el borrado de varios a la vez:

```
$ lxc delete -f focal02 xenial00
```

Usando los contenedores

Tenemos que ver un contenedor como una máquina virtual donde poder ejecutar comandos, correr servicios y aplicaciones, ... De querer ejecutar un comando en los contenedores podemos usar `lxc exec <contenedor> -- <comando>`

```
$ lxc exec focall00 -- date
Thu Nov 11 10:08:25 UTC 2021
$ lxc exec focal00 -- free -h
               total        used        free      shared  buff/cache   available
Mem:            62Gi       141Mi        62Gi       0.0Ki       120Mi        62Gi
Swap:             0B          0B          0B
```

Si lo que se quiere es trabajar directamente en el contenedor podemos lanzar el comando `lxc exec` con la opción `bash` para abrir una terminal. De esta manera, accederemos a una línea de comandos dentro del contenedor donde poder trabajar, hasta que salgamos escribiendo `exit` (volviendo al equipo anfitrión).

```
$ lxc exec focal00 bash
root@focal00:~# cat /etc/issue
Ubuntu 20.04.3 LTS \n \l

root@focal00:~# cat /etc/netplan/50-cloud-init.yaml
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        eth0:
            dhcp4: true
```

```
root@focal00:~# exit
exit
$
```

Las imágenes que se bajaron del repositorio oficial de Ubuntu vienen con un usuario administrador llamado ubuntu al que le podemos poner contraseña:

```
root@focal00:~# passwd ubuntu
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@focal00:~#
```

Si queremos acceder al contenedor como usuario ubuntu podemos ejecutar:

```
$ lxc exec focal00 -- su - ubuntu
ubuntu@focal00:~$
```

Estas imágenes ya vienen con el servidor ssh instalado pero impidiendo el acceso mediante contraseñas. Para activar el acceso por ssh usando contrasiñas, simplemente hay que poner a `Yes` el parámetro `PasswordAuthentication` y reiniciar el servicio ssh de contenedor

```
root@focal00:~# nano /etc/ssh/sshd_config
...
PasswordAuthentication yes
...
root@focal00:~# systemctl restart sshd.service
```

A partir de este momento podremos acceder por ssh al contenedor:

```
$ ssh ubuntu@10.122.236.78
The authenticity of host '10.122.236.78 (10.122.236.78)' can't be established.
ECDSA key fingerprint is SHA256:tOjTSZMA+dowudwDW4QXOmMMb8mnzGiD5ghgDcOBOz4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.122.236.78' (ECDSA) to the list of known hosts.
ubuntu@10.122.236.78's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-90-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Nov 11 10:11:54 UTC 2021

  System load:           0.52
  Usage of /home:        unknown
  Memory usage:          0%
  Swap usage:            0%
  Temperature:           31.0 C
  Processes:             28
```

```
  Users logged in:       0
  IPv4 address for eth0: 10.122.236.78
  IPv6 address for eth0: fd42:9345:33a5:8cc9:216:3eff:fe8a:7c7f

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@focal00:~$
```

Instantáneas y clonación

Después de estar trabajando con un contenedor puede surgir la necesidad de hacer una instantánea para poder recuperar ese estado en un futuro. En lxd usando ZFS las instantáneas son inmediatas y se hace con el comando `lxc snapshot`:

```
$ lxc snapshot focal00 snap_11_11_21
$ lxc info focal00
Name: focal00
Status: STOPPED
Type: container
Architecture: x86_64
Created: 2021/11/11 10:48 CET
Last Used: 2021/11/11 10:48 CET

Snapshots:
+--------------+---------------------+-----------+----------+
|     NAME     |       TAKEN AT      | EXPIRES AT | STATEFUL |
+--------------+---------------------+-----------+----------+
| snap_11_11_21 | 2021/11/11 11:13 CET |           | NO       |
+--------------+---------------------+-----------+----------+
$
```

Como puede observarse, el comando `lxc info` proporciona información sobre el contenedor incluyendo información sobre las instantáneas. De querer recuperar el estado de la máquina en el momento de la instantánea tenemos que ejecutar el comando: `lxc restore <contenedor> <instantánea>`.

Una limitación que tiene el sistema ZFS (que no de LXD) es que únicamente permite recuperar la última instantánea con `lxc restore`. La solución a este problema es hacer un clon a partir de la instantánea con el comando `lxc copy` que permite hacer clonaciones:

```
$ lxc copy xenial00 clon00
$ lxc ls
+--------------+---------+------+------+-----------+-----------+
|     NAME     |  STATE  | IPV4 | IPV6 |   TYPE    | SNAPSHOTS |
+--------------+---------+------+------+-----------+-----------+
| clon00       | STOPPED |      |      | CONTAINER | 1         |
+--------------+---------+------+------+-----------+-----------+
| focal00      | STOPPED |      |      | CONTAINER | 1         |
+--------------+---------+------+------+-----------+-----------+
```

De querer hacer una copia a partir de una instantánea sería: `lxc copy <contenedor>/<instantánea> <nome do clon>`

```
$ lxc copy focal00/snap_11_11_21 clon01
$ lxc ls
+--------------+---------+------+------+-----------+-----------+
|     NAME     |  STATE  | IPV4 | IPV6 |   TYPE    | SNAPSHOTS |
+--------------+---------+------+------+-----------+-----------+
| clon00       | STOPPED |      |      | CONTAINER | 1         |
+--------------+---------+------+------+-----------+-----------+
| clon01       | STOPPED |      |      | CONTAINER | 0         |
+--------------+---------+------+------+-----------+-----------+
| focal00      | STOPPED |      |      | CONTAINER | 1         |
+--------------+---------+------+------+-----------+-----------+
```

5. Profiles

Perfil por defecto

Los *profiles* o perfiles permiten de forma 'centralizada' definir la configuración y los dispositivos de un contenedor para así poder aplicarlo a todos los contenedores que lo precisen. Por ejemplo, podemos crear un perfil donde está impuesto un uso máximo de 256 MB de memoria y 1 cpu. Este perfil podemos aplicarlo a todos aquellos contenedores a los que queramos aplicar esta restricción. Del mismo modo, podemos tener un perfil donde indicamos que habrá dos interfaces de red (p.e. una con conexión a la red local y otra conectada a una red interna/privada) y aplicarlo a todos aquellos contenedores que lo precisen. **Podemos crear varios perfiles y aplicarlos a un contenedor; de esta forma, cuando se cree la configuración final del contenedor los perfiles se aplicarán en el orden en que fueron indicados (sobrescribiendo la configuración anterior en caso de haber coincidencias)**.

Podemos ver los perfiles disponibles ejecutando el comando `lxc profile list` y también modificar el comando `lxc ls` para ver directamente que perfiles están usando los contenedores (ejecutar `lxc ls -h` para ver todas las posibilidades):

```
$ lxc profile list
+--------------+---------------------------------+---------+
|     NAME     |           DESCRIPTION           | USED BY |
+--------------+---------------------------------+---------+
```

```
+---------------+---------------------------------+---------+
| default       | Default LXD profile             | 2       |
+---------------+---------------------------------+---------+
```

$ lxc ls -c n,4,s,P,S,l,m

```
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
|     NAME     |         IPV4        |  STATE  | PROFILES | SNAPSHOTS |     LAST USED AT    | MEMORY USAGE |
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
| alpine00     | 10.122.236.30 (eth0) | RUNNING | default  | 0         | 2021/11/11 10:22 UTC | 13.12MB      |
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
| focal00      | 10.122.236.78 (eth0) | RUNNING | default  | 1         | 2021/11/11 10:21 UTC | 247.22MB     |
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
```

Para ver las características del perfil y los contenedores que lo utilizan ejecutamos el comando `lxc profile show <nombre_perfil>`:

```
$ lxc profile show default
config: {}
description: Default LXD profile
devices:
  eth0:
    name: eth0
    network: lxdbr0
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: default
used_by:
- /1.0/instances/focal00
- /1.0/instances/alpine00
```

Al instalar lxd se crea un perfil llamado *default* que se corresponde con un contenedor con un disco duro y una interfaz de red conectada al dispositivo *lxdbr0*.

Control de recursos

LXD permite controlar/limitar los recursos a usar por parte de un contenedor, entre los que destacan:

- Espacio de disco y límites de velocidad lectura/escritura.
- Número y tiempo de CPUs.
- Memoria RAM,
- Ancho de banda en las conexiones de red.

Podemos aplicar estas restricciones de dos formas:

- Asignando el límite directamente en la configuración del contenedor usando los comandos `lxc config set` y `lxc config device set`.

- Creando un perfil con los límites y asignarlo al contenedor.

Limitando la CPU

Ejecutando el comando `lscpu` en el equipo anfitrión podemos ver los datos de la CPU:

```
$ lscpu
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de bytes:        Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Hilo(s) de procesamiento por núcleo:1
Núcleo(s) por «socket»:4
Socket(s):             1
Modo(s) NUMA:          1
ID de fabricante:      GenuineIntel
Familia de CPU:        6
Modelo:                58
Model name:            Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz
Revisión:             9
CPU MHz:               3403.336
BogoMIPS:              6806.67
Fabricante del hipervisor:KVM
Tipo de virtualización:lleno
Caché L1d:            32K
Caché L1i:            32K
Caché L2:             256K
Caché L3:             6144K
NUMA node0 CPU(s):     0-3
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 x2apic popcnt aes xsave avx rdrand hypervisor lahf_lm
```

Como puede verse el equipo posee 4 CPUs y el mismo resultado aparece al ejecutar `lscpu` en el contenedor (además se ves que usa todas ellas):

```
root@focal00:~# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
```

```
Model:                58
Model name:           Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz
Stepping:             9
CPU MHz:              3403.338
BogoMIPS:             6806.67
Hypervisor vendor:    KVM
Virtualization type:  full
L1d cache:            32K
L1i cache:            32K
L2 cache:             256K
L3 cache:             6144K
NUMA node0 CPU(s):    0-3
...
```

```
root@focal00:~# cat /proc/cpuinfo | grep ^proces
processor    : 0
processor    : 1
processor    : 2
processor    : 3
root@web01:~#
```

Vamos a crear un nuevo perfil donde se limitará el número de CPUs a usar a una única CPU y el tiempo a un 30% del total:

```
$ lxc profile create limites
$ lxc profile edit limites
config:
  limits.cpu: "1"
  limits.cpu.allowance: 30%
description: Perfil con limites
devices: {}
name: limites
used_by: []
$ lxc profile add focal00 limites
Profile limites added to focal00
$ lxc ls -c n,4,s,P,S,l,m
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
|     NAME     |         IPV4        |  STATE  | PROFILES | SNAPSHOTS |     LAST USED AT    | MEMORY USAGE |
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
| alpine00     | 10.122.236.30 (eth0) | RUNNING | default  | 0         | 2021/11/11 10:22 UTC | 13.18MB      |
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
| focal00      | 10.122.236.78 (eth0) | RUNNING | default  | 1         | 2021/11/11 10:26 UTC | 168.64MB     |
|              |                     |         | limites  |           |                     |              |
+--------------+---------------------+---------+----------+-----------+---------------------+--------------+
```

En el contenedor ahora vemos que se usa una única CPU:

```
root@focal00:~# cat /proc/cpuinfo | grep ^proces
processor    : 0
root@focal00:~#
```

Limitando la memoria RAM

El comando `free` permite ver que la memoria RAM del equipo anfitrión de pruebas es de aproximadamente 2GB. Como curiosidad, esta captura se hizo inmediatamente después de arrancar la máquina y con 6 contenedores Ubuntu Server corriendo, y por lo que se ve hay 999MB de RAM disponibles.

```
$ free -h
              total        used        free      shared  buff/cache   available
Memoria:        2,0G        601M        326M         56M        1,0G        999M
Swap:           2,9G          0B        2,9G
```

No contenedor sucede o mesmo:

```
root@focal00:~# free -h
              total        used        free      shared  buff/cache   available
Mem:            2.0G         17M        1.9G          0B         63M        1.9G
Swap:           2.9G          0B        2.9G
```

Añadiendo en el perfil el parámetro `limits.memory` podemos controlar cuanta RAM puede usar el contenedor:

```
$ lxc profile edit limites
config:
  limits.cpu: "1"
  limits.cpu.allowance: 30%
  limits.memory: 400MB
description: Perfil con limites
devices: {}
name: limites
used_by:
- /1.0/containers/web01
```

Una vez guardado el perfil, los límites se aplican inmediatamente (incluso a los contenedores en ejecución, siempre que sea posible) y puede comprobarse que la RAM que ve el contenedor cambió:

```
root@web01:~# free -h
              total        used        free      shared  buff/cache   available
Mem:          381Mi        75Mi       221Mi       0.0Ki        84Mi       305Mi
Swap:            0B          0B          0B
```

El contenedor no podrá usar más RAM de la indicada (*hard limit*); sin embargo, con el parámetro `limits.memory.enforce: soft` puede configurarse un *soft limit*, permitiendo al contenedor usar más RAM de la indicada siempre y cuando esté disponible en el equipo anfitrión.

Limitando el tamaño del disco duro

El tamaño del *pool* ZFS de mi equipo es de unos 25 GB:

```
$ sudo zpool list
NAME      SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
default  24,9G   547M  24,3G         -     2%     2%  1.00x  ONLINE  -
```

Corresponde con el tamaño visto en el contenedor:

```
root@web01:~# df -h
Filesystem                Size  Used Avail Use% Mounted on
default/containers/web01   24G  308M   24G   2% /
none                      492K     0  492K   0% /dev
udev                      979M     0  979M   0% /dev/tty
tmpfs                     100K     0  100K   0% /dev/lxd
tmpfs                     100K     0  100K   0% /dev/.lxd-mounts
tmpfs                    1000M     0 1000M   0% /dev/shm
tmpfs                    1000M  8.2M  992M   1% /run
tmpfs                     5.0M     0  5.0M   0% /run/lock
tmpfs                    1000M     0 1000M   0% /sys/fs/cgroup
```

Añadiendo las siguientes líneas en el perfil limites podemos asignar un tamaño máximo de disco en el contenedor de unos 12 GB:

```
$ lxc profile edit limites
config:
  limits.cpu: "1"
  limits.cpu.allowance: 30%
  limits.memory: 400MB
  description: Perfil con limites
devices:
  root:
    path: /
    pool: default
    type: disk
    size: 12GB
name: limites
used_by:
- /1.0/containers/web01
```

Ahora en el contenedor se ve el límite de 12GB:

```
root@web01:~# df -h
Filesystem                  Size  Used Avail Use% Mounted on
default/containers/focal00   12G  569M   12G   5% /
none                        492K  4.0K  488K   1% /dev
udev                         32G     0   32G   0% /dev/tty
tmpfs                       100K     0  100K   0% /dev/lxd
tmpfs                       100K     0  100K   0% /dev/.lxd-mounts
tmpfs                        32G     0   32G   0% /dev/shm
tmpfs                       6.3G  188K  6.3G   1% /run
tmpfs                       5.0M     0  5.0M   0% /run/lock
tmpfs                        32G     0   32G   0% /sys/fs/cgroup
```

Si repasamos lo hecho para limitar el espacio de disco, vemos que añadimos en el perfil limites un dispositivo de disco duro de 12 GB de tamaño. Si recordamos, en el perfil *default* ya había un dispositivo de red y otro de un disco duro sin límite de tamaño; por lo tanto, cuando a un

contenedor se le aplican varios perfiles, la configuración se sobreescribe en base al orden de aplicación de los perfiles, quedando un contenedor con un dispositivo de red y un disco duro de tamaño 12GB.

Para asignar un perfil en el momento de la creación del contenedor ejecutamos el comando:

```
$ lxc launch ubuntu:f focal02 -p default -p limites
```

Para asignar un perfil a un contenedor ya creado:

```
$ lxc profile assign focal02 limites
```

Hay que tener en cuenta que no es lo mismo añadir (*add*) que asignar (*assign*) un perfil.

Limitando el tamaño del disco duro

Los límites también se pueden aplicar directamente en la configuración del contenedor usando los comandos `lxc config set` y `lxc config device set`:

```
manuel@x99:~$ lxc exec remoto -- nproc
48
manuel@x99:~$ lxc exec remoto -- free -h
              total        used        free      shared  buff/cache   available
Mem:           62Gi       204Mi        62Gi       0.0Ki       165Mi        62Gi
Swap:            0B          0B          0B
manuel@x99:~$ lxc exec remoto -- df -h
Filesystem                 Size  Used Avail Use% Mounted on
default/containers/remoto  210G  645M  209G   1% /
none                       492K  4.0K  488K   1% /dev
udev                        32G     0   32G   0% /dev/tty
tmpfs                      100K     0  100K   0% /dev/lxd
tmpfs                      100K     0  100K   0% /dev/.lxd-mounts
tmpfs                       32G     0   32G   0% /dev/shm
tmpfs                      6.3G  216K  6.3G   1% /run
tmpfs                      5.0M     0  5.0M   0% /run/lock
tmpfs                       32G     0   32G   0% /sys/fs/cgroup
manuel@x99:~$ lxc config set remoto limits.cpu=2 limits.memory=512MB
manuel@x99:~$ lxc config device override remoto root size=10GB
Device root overridden for remoto
manuel@x99:~$ lxc exec remoto -- nproc
2
manuel@x99:~$ lxc exec remoto -- free -h
              total        used        free      shared  buff/cache   available
Mem:          488Mi        93Mi       394Mi       0.0Ki          0B       394Mi
Swap:            0B          0B          0B
manuel@x99:~$ lxc exec remoto -- df -h
Filesystem                 Size  Used Avail Use% Mounted on
default/containers/remoto  9.9G  645M  9.2G   7% /
none                       492K  4.0K  488K   1% /dev
udev                        32G     0   32G   0% /dev/tty
tmpfs                      100K     0  100K   0% /dev/lxd
```

```
tmpfs                    100K     0  100K   0% /dev/.lxd-mounts
tmpfs                     32G     0   32G   0% /dev/shm
tmpfs                    6.3G  216K  6.3G   1% /run
tmpfs                    5.0M     0  5.0M   0% /run/lock
tmpfs                     32G     0   32G   0% /sys/fs/cgroup
```

6. Redes

Interfaz lxdbr0

Al trabajar con contenedores LXD podemos crear configuraciones de red similares a las que ofrecen soluciones de virtualización como Virtualbox. La gestión de las redes se puede hacer a través de la configuración de red del sistema operativo o directamente a través de LXD.

En el momento de hacer `lxd init` creamos una nueva interfaz de red lxdbr0 que proporciona funcionalidades similares a las del modo red NAT en Virtualbox:

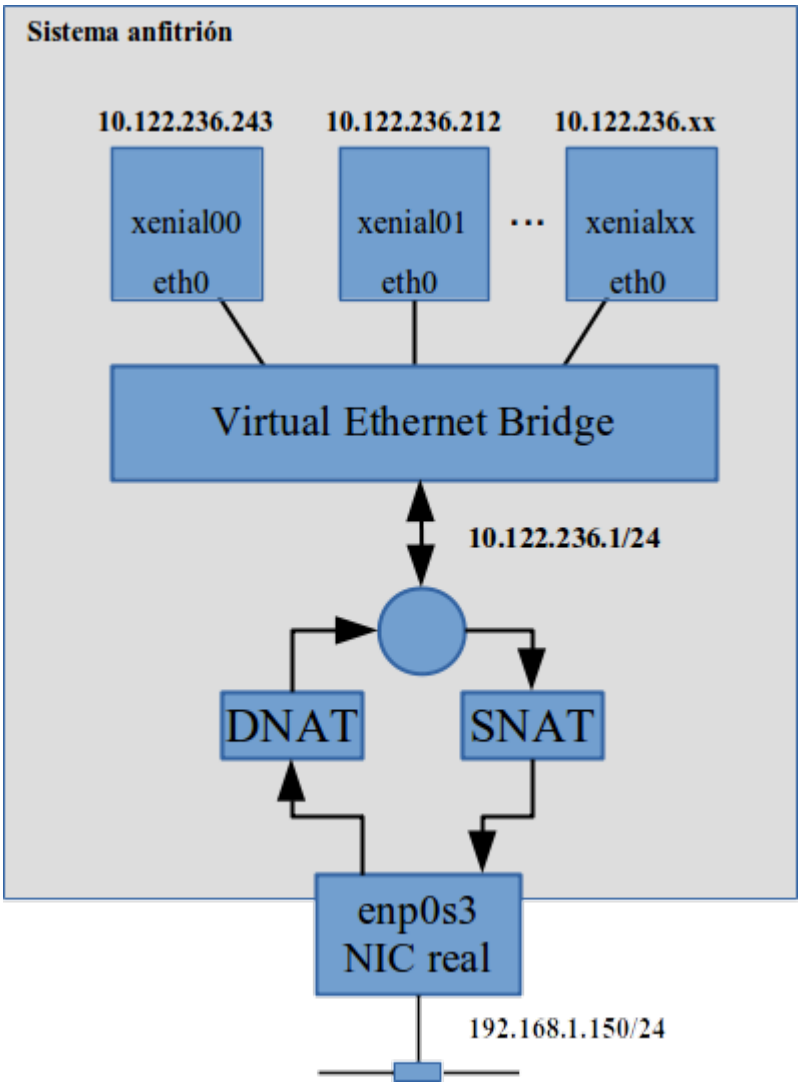
- Los contenedores reciben por dhcp una configuración válida para una red privada 10.122.236.0/24 (esta es la red creada en mi equipo y seguramente sea diferente en el vuestro).
- Los contenedores conectados a ella pueden comunicarse entre si. Es decir, funciona como se fuese un conmutador/switch para ellos.
- Los contenedores conectados a ella pueden salir al exterior. Es decir, funciona como si fuese un router-nat, traduciendo las direcciones IP privadas por la IP real del equipo anfitrión. Automáticamente se crearon reglas iptables de filtrado y NAT en el equipo anfitrión durante la inicialización de lxd.
- El equipo anfitrión puede comunicarse con los contenedores a través las direcciones de la red interna (en mi ejemplo, la red 10.122.236.0/24) y viceversa.
- Desde fuera del equipo anfitrión no se pueden iniciar comunicaciones directas hacia los servicios corriendo en los contenedores; por lo tanto, habría que crear reglas de filtrado y NAT para permitirlo o crear a través de LXD un dispositivo tipo proxy asociado al contenedor.

En mi equipo de pruebas hay dos contenedores usando el perfil por defecto que emplea lxdbr0 como tarjeta de red:

```
$ lxc ls -c n,4,s,P,S,l
+--------------+----------------------+---------+----------+-----------+----------------------+
|     NAME     |         IPV4         |  STATE  | PROFILES | SNAPSHOTS |     LAST USED AT     |
+--------------+----------------------+---------+----------+-----------+----------------------+
| alpine00     | 10.122.236.30 (eth0) | RUNNING | default  | 0         | 2021/11/11 10:22 UTC |
+--------------+----------------------+---------+----------+-----------+----------------------+
| focal00      | 10.122.236.78 (eth0) | RUNNING | default  | 1         | 2021/11/11 10:26 UTC |
|              |                      |         | limites  |           |                      |
+--------------+----------------------+---------+----------+-----------+----------------------+
$ ip addr
...
7: lxdbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
```

```
    link/ether 00:16:3e:ab:ff:54 brd ff:ff:ff:ff:ff:ff
    inet 10.122.236.1/24 scope global lxdbr0
...
```

Gráficamente la red lxdbr0 podría entenderse como:



Redes en LXD

Para ver las redes o dispositivos de red a los que conectar los contenedores ejecutamos el comando:

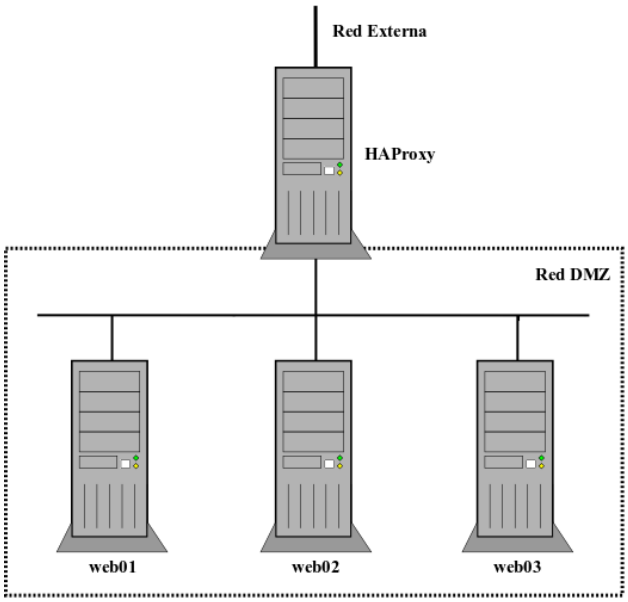
```
$ lxc network list
```

```
+--------+----------+---------+-----------------+---------------------------+------------+---------+
|  NAME  |   TYPE   | MANAGED |       IPV4      |           IPV6            | DESCRIPTION | USED BY |
+--------+----------+---------+-----------------+---------------------------+------------+---------+
| enp6s0 | physical | NO      |                 |                           |            | 1       |
+--------+----------+---------+-----------------+---------------------------+------------+---------+
| lxdbr0 | bridge   | YES     | 10.122.236.1/24 | fd42:9345:33a5:8cc9::1/64 |            | 4       |
+--------+----------+---------+-----------------+---------------------------+------------+---------+
```

En mi caso tengo a mi disposición dos interfaces:

- `enp6s0`: podría usarse para trabajar en modo macvlan o physical.

- `lxdbr0`: además de interfaz de red tenemos que verla como una red tipo red NAT (fijarse en que aparece como *bridge* en el tipo de red).



Comenzamos creando una nueva interfaz/red para simular una red externa con direccionamiento IP público usando el comando `lxc network create`:

```
$ lxc network create wan ipv4.address=192.0.2.1/24 ipv6.address=none ipv4.dhcp=true ipv4.dhcp.ranges="192.0.2.100-192.0.2.150" ipv4.firewall=true ipv4.nat=true
Network wan created
$ lxc network list
+--------+----------+---------+-----------------+---------------------------+------------+---------+
|  NAME  |   TYPE   | MANAGED |       IPV4      |           IPV6            | DESCRIPTION | USED BY |
+--------+----------+---------+-----------------+---------------------------+------------+---------+
| enp6s0 | physical | NO      |                 |                           |            | 1       |
+--------+----------+---------+-----------------+---------------------------+------------+---------+
| lxdbr0 | bridge   | YES     | 10.122.236.1/24 | fd42:9345:33a5:8cc9::1/64 |            | 4       |
+--------+----------+---------+-----------------+---------------------------+------------+---------+
| wan    | bridge   | YES     | 192.0.2.1/24    | none                      |            | 0       |
+--------+----------+---------+-----------------+---------------------------+------------+---------+
$ lxc network show wan
config:
  ipv4.address: 192.0.2.1/24
  ipv4.dhcp: "true"
  ipv4.dhcp.ranges: 192.0.2.100-192.0.2.150
  ipv4.firewall: "true"
  ipv4.nat: "true"
  ipv6.address: none
description: ""
name: wan
type: bridge
used_by: []
managed: true
```

```
status: Created
locations:
- none
```

La red tiene las siguientes características:

- Nombre wan y de tipo bridge (todos los contenedores situados en ella podrán comunicarse entre si y con el anfitrión).
- Asociada a una nueva interfaz en el equipo anfitrión llamada wan y con la IP 192.0.2.1 con máscara /24.
- Sin IPv6 asignada.
- Con un servidor dhcp y dns corriendo para dar ese servicio a los contenedores conectados a ella que así lo necesite. Las IPs reservadas para otorgar por dhcp van de la 192.0.2.100 a la 192.0.2.150.
- Se crearon las reglas iptables/netfilter de filtrado y NAT necesarias para funcionar el servicio dhcp/dns y permitir la salida al exterior de los contenedores conectados a esta red.

Al hacer un `ip addr` aparecerán las nuevas interfaces asociadas a las redes creadas (si tenemos arrancados contenedores aparecerán otras interfaces de nombre `vethXXXX` que se corresponden con dispositivos de red efímeros asociados a los contenedores y que permiten sus comunicaciones):

```
$ ifconfig -a
...
8: wan: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 00:16:3e:4a:33:cb brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 scope global wan
       valid_lft forever preferred_lft forever
...
```

Otros comandos interesantes para trabajar con redes en LXD son:

- `lxc network create <nombre_de_la_red>` --> crea una nueva red encargándose LXD de asignarle una IP al azar y las características/funcionalidades por defecto.
- `lxc network edit <nombre_de_la_red>` --> se abrirá el editor `vi` permitiendo editar las características de la red.

Redes y profiles

Para que los contenedores usen la red *wan* tenemos que crear un perfil que asocie la interfaz de red del contenedor con la red wan:

```
$ lxc profile copy default wan
$ lxc profile edit wan
config: {}
description: WAN
devices:
  eth0:
    name: eth0
```

```
    network: wan
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: wan
used_by: []
```

Como puede verse modificamos el parámetro `network` que sirve para indicar cual de las interfaces de red del anfitrión va estar ligada a la interfaz virtual del contenedor. En este perfil queremos que la tarjeta de red del contenedor esté conectada a la *wan* y no a la *lxdbr0*.

En caso de precisar un contenedor conectado a dos redes, repetimos el proceso creando un nuevo perfil que en este caso tendrá dos tarjetas de red, una conectada a una red (p.e. la red *wan*) y la otra a otra red (p.e. *lxdbr0*):

```
$ lxc profile copy default 2NICs
$ lxc profile edit 2NICs
config: {}
description: perfil 2 NICs
devices:
  eth0:
    name: eth0
    network: wan
    type: nic
  eth1:
    name: eth1
    network: lxdbr0
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: 2NICs
used_by: []
```

Los contenedores que usen este perfil tendrán dos tarjetas de red (*eth0* e *eth1*), la primera conectada a la red externa (`network: wan`) y la segunda conectada a la red interna (`network: lxdbr0`). Sólo faltaría asignarle el perfil al contenedor:

```
$ lxc launch focal02 firewall -p 2NICs
```

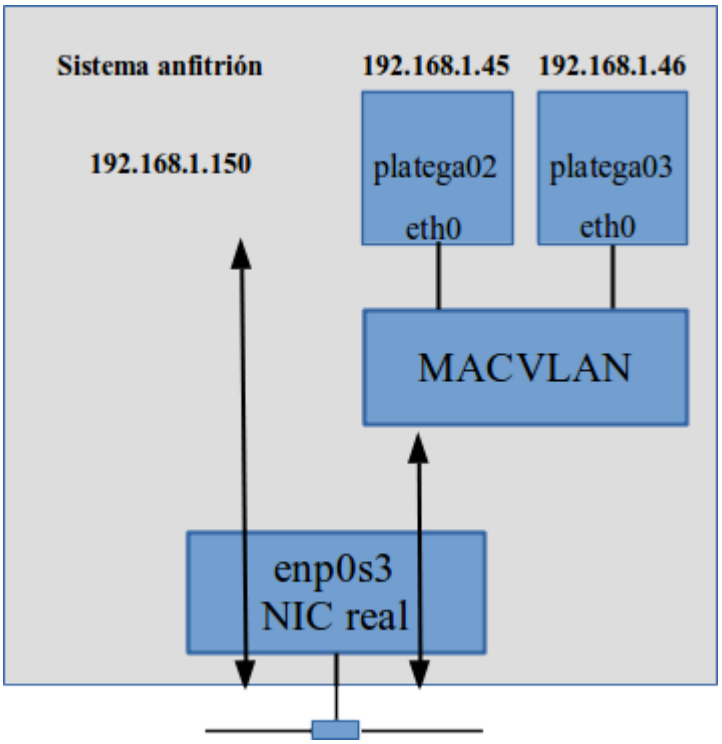
Modo MACVLAN

De este modo de red hay que destacar los siguientes puntos:

- Es el modo más sencillo de conectar los contenedores directamente a la red física y no obliga a tocar la configuración de red del host anfitrión.
- Normalmente ofrece un mejor rendimiento que el modo *bridge* que se verá más adelante.

- Permite la comunicación entre los contenedores.
- NO permite la comunicación entre los contenedores y el host anfitrión. Por tanto, si necesitamos que haya comunicaciones entre contenedores y el anfitrión, este modo no nos vale.
- No funciona si el equipo anfitrión de los contenedores está virtualizado en Virtualbox.

En la siguiente imagen puede verse una interpretación de las conexiones en este modo de red:



En primer lugar, se crea un perfil a partir del perfil por defecto y modificamos el *nictype* a *macvlan* y el *parent* lo asignamos a una de las tarjetas de red reales del anfitrión:

```
$ lxc profile copy default macvlan
$ lxc profile edit macvlan
config: {}
description: Perfil macvlan
devices:
  eth0:
    name: eth0
    nictype: macvlan
    parent: enp6s0
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: macvlan
used_by: []
$ lxc profile list
+---------+---------+
```

```
| NOMBRE  | USED BY |
+---------+---------+
| default | 2       |
+---------+---------+
| macvlan | 0       |
+---------+---------+
```

A continuación creamos dos nuevos contenedores aplicándoles el perfil macvlan.

```
$ lxc launch ubuntu:f mac00 -p macvlan
Creating mac00
Starting mac00
$ lxc launch ubuntu:f mac01 -p macvlan
Creating mac01
Starting mac01
$
```

Podemos comprobar que reciben la configuración de red dende el servidor dhcp de la red local real 192.168.1.0/24 (y no del dhcp asociado a *lxdbr0*):

```
$ lxc ls -c n,s,4,6,P,S
+--------------+---------------------+---------+----------+-----------+----------------------+
|     NAME     |         IPV4        |  STATE  | PROFILES | SNAPSHOTS |     LAST USED AT     |
+--------------+---------------------+---------+----------+-----------+----------------------+
| alpine00     | 10.122.236.30 (eth0) | RUNNING | default  | 0         | 2021/11/11 10:22 UTC |
+--------------+---------------------+---------+----------+-----------+----------------------+
| focal00      | 10.122.236.78 (eth0) | RUNNING | default  | 1         | 2021/11/11 10:26 UTC |
|              |                     |         | limites  |           |                      |
+--------------+---------------------+---------+----------+-----------+----------------------+
| focal02      | 10.122.236.51 (eth0) | RUNNING | default  | 0         | 2021/11/11 10:33 UTC |
|              |                     |         | limites  |           |                      |
+--------------+---------------------+---------+----------+-----------+----------------------+
| mac00        | 192.168.1.35 (eth0)  | RUNNING | macvlan  | 0         | 2021/11/11 11:25 UTC |
+--------------+---------------------+---------+----------+-----------+----------------------+
| mac01        | 192.168.1.38 (eth0)  | RUNNING | macvlan  | 0         | 2021/11/11 11:26 UTC |
+--------------+---------------------+---------+----------+-----------+----------------------+
```

Haciendo algunas pruebas de conectividad podemos verificar lo indicado sobre las comunicaciones con el exterior y entre los contenedores: si se permiten.

```
$ lxc exec mac00 -- host www.xunta.gal 8.8.8.8
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:
$ lxc exec mac00 -- ping -c 2 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.587 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.470 ms

--- 192.168.1.1 ping statistics ---
```

```
2 packets transmitted, 2 received, 0% packet loss, time 1033ms
rtt min/avg/max/mdev = 0.470/0.528/0.587/0.058 ms
$ lxc exec mac00 -- ping -c 2 192.168.1.38
PING 192.168.1.38 (192.168.1.38) 56(84) bytes of data.
64 bytes from 192.168.1.38: icmp_seq=1 ttl=64 time=0.106 ms
64 bytes from 192.168.1.38: icmp_seq=2 ttl=64 time=0.055 ms

--- 192.168.1.38 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1030ms
rtt min/avg/max/mdev = 0.055/0.080/0.106/0.025 ms
```

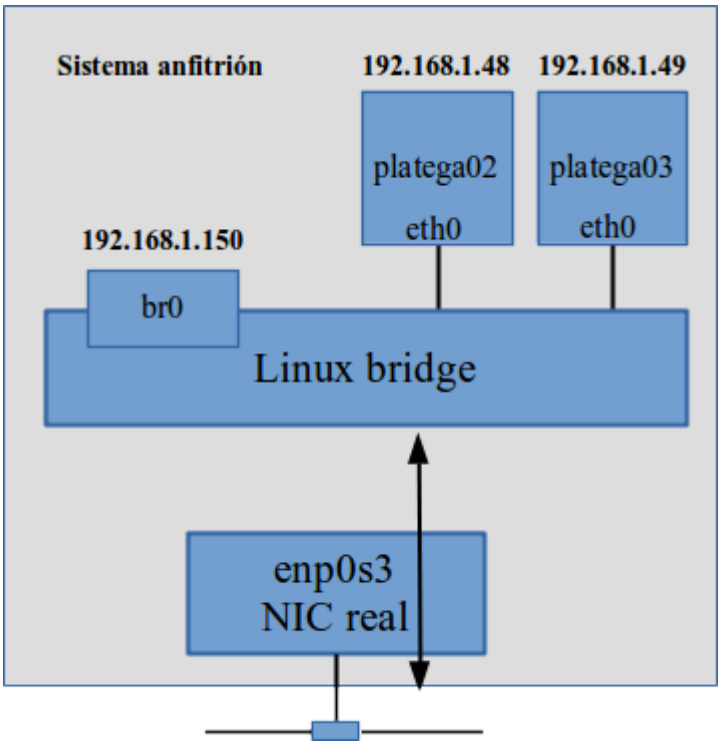
Las comunicaciones con el equipo anfitrión no se permiten:

```
$ lxc exec mac00 -- ping -c 2 192.168.1.150
PING 192.168.1.150 (192.168.1.150) 56(84) bytes of data.
From 192.168.1.45 icmp_seq=1 Destination Host Unreachable
--- 192.168.1.150 ping statistics ---
2 packets transmitted, 0 received, +1 errors, 100% packet loss, time 1018ms
pipe 2
```

A partir de ahora, el contenedor en modo macvlan debe tener una configuración válida para la red de la organización. Si arrancamos el contenedor, este trataría de obtener una configuración de red por dhcp en base a su archivo de configuración (`/etc/network/interfaces` o el ubicado en `/etc/netplan/`. Si queremos asignarle una configuración estática, simplemente habría que indicar en el archivo de configuración la nueva configuración.

Modo Bridge

El modo Bridge permite trabajar con los contenedores de forma similar al modo puente (bridge) de Virtualbox, permitiendo la comunicación de los contenedores con la red externa y con el equipo anfitrión (y viceversa). La idea es crear un switch virtual donde estarán conectados los contenedores, el equipo anfitrión y la red local. También podemos tener el modo rede interna de Virtualbox creando switches 'aislados' de forma que los contenedores conectados a ellos, únicamente podrán comunicarse entre si (con el resto del mundo podrían empleando otro contenedor a modo de router). En la siguiente imagen puede verse una interpretación gráfica del modo *bridge*:



Configuración de red del equipo anfitrión

Para poder crear un bridge en Linux Ubuntu hay que instalar el paquete `bridge-utils` y cambiar la configuración de red del equipo anfitrión para crear el puente:

```
$ sudo apt-get update
$ sudo apt-get install bridge-utils
```

```
$ sudo nano /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
#auto enp0s3
#iface enp0s3 inet static
auto br0
iface br0 inet static
        address 192.168.1.150
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 8.8.8.8 8.8.4.4
        bridge_ports enp0s3
        bridge_stp off
        bridge_fd 0
        bridge_maxwait 0
```

En Ubuntu 18.04 y posteriores hay que usar netplan para configurar la red:

```
$ sudo nano /etc/netplan/01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: false
      dhcp6: false
  bridges:
    br0:
      interfaces: [enp0s3]
      addresses: [192.168.1.150/24]
      gateway4: 192.168.1.1
      mtu: 1500
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
      parameters:
        stp: true
        forward-delay: 4
      dhcp4: no
      dhcp6: no
$ sudo netplan apply
```

Activamos *br0* reiniciando el equipo y verificamos que está activa. Fijarse que en la salida de `ifconfig` aparecen varias interfaces:

- *br0*: el nuevo bridge con una configuración IP completa.

- *enp0s3*: la tarjeta de red que usará el bridge para conectarse físicamente a la red.
- *lo*: lazo cerrado (loopback)
- *lxdbr0*: el bridge creado durante la inicialización de LXD para conectar los contenedores.

```
$ ifconfig -a
br0       Link encap:Ethernet  direcciónHW 08:00:27:48:e3:f7
          Direc. inet:192.168.1.150  Difus.:192.168.1.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe48:e3f7/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:964 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:647 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:915829 (915.8 KB)  TX bytes:66968 (66.9 KB)

enp0s3    Link encap:Ethernet  direcciónHW 08:00:27:48:e3:f7
          Dirección inet6: fe80::a00:27ff:fe48:e3f7/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:1262 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:658 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:949139 (949.1 KB)  TX bytes:69498 (69.4 KB)

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:82 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:82 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1
          Bytes RX:6112 (6.1 KB)  TX bytes:6112 (6.1 KB)

lxdbr0    Link encap:Ethernet  direcciónHW fe:38:c1:39:dc:63
          Direc. inet:10.119.191.1  Difus.:0.0.0.0  Másc:255.255.255.0
          Dirección inet6: fd42:380b:4b9f:3572::1/64 Alcance:Global
          Dirección inet6: fe80::1499:aff:fe7d:7134/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:208 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:187 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:16598 (16.5 KB)  TX bytes:21776 (21.7 KB)
```

Si tenemos arrancados contenedores aparecerán otras interfaces de nombre `vethXXXX` que se corresponden con dispositivos de red efímeros asociados a los contenedores y que permiten sus comunicaciones. Podemos hacer un listado de las interfaces de red que puede usar LXD ejecutando:

```
$ lxc network list
+--------+----------+---------+------------+---------+
| NOMBRE |   TIPO   | MANAGED | DESCRIPCIÓN | USED BY |
+--------+----------+---------+------------+---------+
| br0    | bridge   | NO      |            | 0       |
+--------+----------+---------+------------+---------+
| enp0s3 | physical | NO      |            | 0       |
+--------+----------+---------+------------+---------+
| lxdbr0 | bridge   | SÍ      |            | 4       |
+--------+----------+---------+------------+---------+
```

Igual que se hizo en el modo macvlan, comenzamos creando un nuevo perfil a partir del *default* y modificamos el `parent` a *br0* para indicar que la tarjeta *eth0* del contenedor va a conectarse al switch *br0*:

```
$ lxc profile copy default ponte-br0
$ lxc profile edit ponte-br0
config:
  environment.http_proxy: ""
  user.network_mode: ""
description: Perfil bridge
devices:
  eth0:
    nictype: bridged
    parent: br0
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: ponte-br0
used_by: []
$ lxc profile list
+-----------+---------+
|  NOMBRE   | USED BY |
+-----------+---------+
| default   | 2       |
+-----------+---------+
| macvlan   | 2       |
+-----------+---------+
| ponte-br0 | 0       |
+-----------+---------+
```

Si queremos crear un nuevo contenedor que use este perfil tenemos que ejecutar:

```
$ lxc launch trusty64 platega04 --profile ponte-br0
Creando platega04
Iniciando platega04
```

Se no queremos crear un nuevo contenedor y si aplicarle el nuevo perfil a contenedores ya creados, tenemos que usar `lxc profile assign`:

```
$ lxc profile assign platega02 ponte-br0
Profiles ponte-br0 applied to platega02
```

Podemos comprobar que reciben la configuración de red desde el servidor dhcp de la red local (y no por el dhcp asociado a *lxdbr0*):

```
$ lxc ls -c n,s,4,6,P,S
+-----------+---------+----------------------+-----------------------------------------------+-----------+-----------+
|  NOMBRE   | ESTADO  |         IPV4         |                     IPV6                      | PERFILES  | SNAPSHOTS |
+-----------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| platega00 | RUNNING | 10.119.191.239 (eth0) | fd42:380b:4b9f:3572:216:3eff:feaf:a7c7 (eth0) | default   | 1         |
+-----------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| platega01 | RUNNING | 10.119.191.183 (eth0) | fd42:380b:4b9f:3572:216:3eff:fe1d:1b21 (eth0) | default   | 0         |
+-----------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| platega02 | RUNNING | 192.168.1.48 (eth0)   |                                               | ponte-br0 | 0         |
+-----------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| platega04 | RUNNING | 192.168.1.49 (eth0)   |                                               | ponte-br0 | 0         |
+-----------+---------+----------------------+-----------------------------------------------+-----------+-----------+
```

Podemos comprobar que funcionan las comunicaciones en todos las direcciones:

Entre contenedores:

```
$ lxc exec platega02 -- ping -c 2 192.168.1.49
PING 192.168.1.49 (192.168.1.49) 56(84) bytes of data.
64 bytes from 192.168.1.49: icmp_seq=1 ttl=64 time=0.244 ms
64 bytes from 192.168.1.49: icmp_seq=2 ttl=64 time=0.090 ms
--- 192.168.1.49 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.090/0.167/0.244/0.077 ms
```

Entre contenedor y equipo anfitrión (y viceversa):

```
$ lxc exec platega02 -- ping -c 2 192.168.1.150
PING 192.168.1.150 (192.168.1.150) 56(84) bytes of data.
64 bytes from 192.168.1.150: icmp_seq=1 ttl=64 time=0.203 ms
64 bytes from 192.168.1.150: icmp_seq=2 ttl=64 time=0.077 ms
--- 192.168.1.150 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.077/0.140/0.203/0.063 ms
$ ping -c 2 192.168.1.48
PING 192.168.1.48 (192.168.1.48) 56(84) bytes of data.
64 bytes from 192.168.1.48: icmp_seq=1 ttl=64 time=0.066 ms
64 bytes from 192.168.1.48: icmp_seq=2 ttl=64 time=0.084 ms
--- 192.168.1.48 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.066/0.075/0.084/0.009 ms
```

Entre contenedor y el exterior:

```
$ lxc exec platega02 -- host www.edu.xunta.es 8.8.8.8
```

```
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:
www.edu.xunta.es has address 85.91.64.102
```

Dende equipos externos y el contenedor:

```
$ ping -c 2 192.168.1.48
PING 192.168.1.48 (192.168.1.48) 56(84) bytes of data.
64 bytes from 192.168.1.48: icmp_seq=1 ttl=64 time=0.559 ms
64 bytes from 192.168.1.48: icmp_seq=2 ttl=64 time=0.386 ms
--- 192.168.1.48 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1027ms
rtt min/avg/max/mdev = 0.386/0.472/0.559/0.089 ms
```

Igual que lo indicado en el modo macvlan, a partir de ahora el contenedor debe tener una configuración válida para la red de la organización. En este momento si arrancamos el contenedor, este trataría de obtener una configuración de red por dhcp en base a su archivo de configuración de red. Si queremos asignarle una configuración estática, simplemente habría que indicarla en el archivo de configuración.

Acceso a un servicio corriendo en un contenedor mediante un dispositivo proxy

En los modos macvlan y bridge los servicios corriendo en un contenedor son directamente accesibles desde la red exterior pero con redes tipo `lxdbr0` eso no es así. Para hacerlo posible hay dos formas:

- Mediante reglas NAT de netfilter/iptables.
- Mediante un dispositivo de tipo proxy (a partir da versión 3.0 de lxd) que hace el trabajo de las reglas NAT.

A modo de ejemplo, haremos que un servidor web corriendo en xenial00 sea accesible desde cualquier ordenador a través de la IP del equipo real; para eso, en el equipo real creamos un dispositivo de tipo proxy que vincule el puerto tcp/80 del equipo real con el puerto tcp/80 del contenedor donde corre el servidor web:

```
$ lxc config device add focal00 apache proxy listen=tcp:0.0.0.0:80 connect=tcp:127.0.0.1:80
Device apache added to xenial00
magasix@lxd:~$ lxc config show focal00
architecture: x86_64
config:
  image.architecture: amd64
  image.description: ubuntu 20.04 LTS amd64 (release) (20211108)
  image.label: release
  image.os: ubuntu
  image.release: focal
  image.serial: "20211108"
  image.type: squashfs
  image.version: "20.04"
```

```
  volatile.base_image: bd2ffb937c95633a28091e6efc42d6c7b1474ad8eea80d6ed8df800e44c6bfdd
  volatile.eth0.host_name: vethfd7d3234
  volatile.eth0.hwaddr: 00:16:3e:8a:7c:7f
  volatile.idmap.base: "0"
...
devices:
  apache:
    connect: tcp:127.0.0.1:80
    listen: tcp:0.0.0.0:80
    type: proxy
ephemeral: false
profiles:
- default
- limites
stateful: false
description:
```

Atacando la IP del equipo real puerto tcp/80 podremos acceder al servidor Apache corriendo en el contenedor.

7. Proyectos

La funcionalidad de los proyectos o *projects* está pensada para agrupar uno o más contenedores/perfiles y poder trabajar más cómodamente. Usaremos los projects para agrupar contenedores y perfiles de forma que al ejecutar comandos lxc dentro de un project únicamente se aplicarán a los contenedores/perfiles del project.

Hay un perfil *default* donde se crean los contenedores/perfiles por defecto:

```
$ lxc project list
+------------------+--------+----------+-----------------+----------+--------------------+---------+
|       NAME       | IMAGES | PROFILES | STORAGE VOLUMES | NETWORKS |     DESCRIPTION    | USED BY |
+------------------+--------+----------+-----------------+----------+--------------------+---------+
| default (current) | YES    | YES      | YES             | YES      | Default LXD project | 34      |
+------------------+--------+----------+-----------------+----------+--------------------+---------+
$ lxc project show default
config:
  features.images: "true"
  features.networks: "true"
  features.profiles: "true"
  features.storage.volumes: "true"
description: Default LXD project
name: default
used_by:
- /1.0/images/712a5836865525eecd8b429afa7430e8a2480aeff68804c59a0524dc8b7683ab
...
- /1.0/instances/alpine00
- /1.0/instances/focal00
```

```
…
- /1.0/networks/lxdbr0
- /1.0/networks/wan
...
- /1.0/profiles/default
- /1.0/profiles/macvlan
...
```

Podemos crear un *project* ejecutando el comando:

```
$ lxc project create DMZS -c features.images=false -c features.profiles=true
Project DMZS created
$ lxc project list
+------------------+--------+----------+-----------------+----------+--------------------+---------+
|       NAME       | IMAGES | PROFILES | STORAGE VOLUMES | NETWORKS |     DESCRIPTION    | USED BY |
+------------------+--------+----------+-----------------+----------+--------------------+---------+
| DMZS             | NO     | YES      | YES             | NO       |                    | 1       |
+------------------+--------+----------+-----------------+----------+--------------------+---------+
| default (current) | YES    | YES      | YES             | YES      | Default LXD project | 34      |
+------------------+--------+----------+-----------------+----------+--------------------+---------+
$ lxc project show DMZS
description: ""
config:
  features.images: "false"
  features.profiles: "true"
name: DMZS
used_by:
- /1.0/profiles/default?project=DMZS
```

A la hora de crear un *project*:

- `features.images=false` → controla si se usan las imágenes del project por defecto (*false*) o unas propias del project DMZS (*true*).
- `features.profiles=true` → controla si se usan los perfiles del project por defecto (*false*) o unos propios del project (*true*).

A modo de ejemplo, trabajaremos en el project DMZS y crearemos una red de nombre dmzint, un perfil que la usa y dos contenedores que usen ese perfil:

```
$ lxc network create dmzint ipv4.address=172.30.0.1/16 ipv6.address=none ipv4.dhcp=true ipv4.dhcp.ranges="172.30.0.100-172.30.0.200" ipv4.firewall=true ipv4.nat=true
$ lxc network list
+--------+----------+---------+-------------+---------+
|  NAME  |   TYPE   | MANAGED | DESCRIPTION | USED BY |
+--------+----------+---------+-------------+---------+
| dmzint | bridge   | YES     |             | 0       |
+--------+----------+---------+-------------+---------+
| enp0s3 | physical | NO      |             | 1       |
+--------+----------+---------+-------------+---------+
```

```
| lxdbr0 | bridge   | YES     |             | 4       |
+--------+----------+---------+-------------+---------+
```

Observar que las redes son globales, no asociadas a ningún project en particular. Cambiamos al project DMZS para trabajar dentro de él:

$ lxc project switch DMZS

$ lxc project list

```
+----------------+--------+----------+-----------------+----------+--------------------+---------+
|      NAME      | IMAGES | PROFILES | STORAGE VOLUMES | NETWORKS |     DESCRIPTION    | USED BY |
+----------------+--------+----------+-----------------+----------+--------------------+---------+
| DMZS (current) | NO     | YES      | YES             | NO       |                    | 1       |
+----------------+--------+----------+-----------------+----------+--------------------+---------+
| default        | YES    | YES      | YES             | YES      | Default LXD project | 34      |
+----------------+--------+----------+-----------------+----------+--------------------+---------+
```

$ lxc profile edit dmzint

```
config: {}
description: Perfil DMZ Interna
devices:
  eth0:
    name: eth0
    neteork: dmzint
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: dmzint
used_by: []
```

$ lxc launch ubuntu:f int00 -p dmzint

```
Creating int00
Starting int00
```

$ lxc launch ubuntu:f int01 -p dmzint

```
Creating int01
Starting int01
```

$ lxc project show DMZS

```
description: ""
config:
  features.images: "false"
  features.profiles: "true"
name: DMZS
used_by:
- /1.0/containers/int00?project=DMZS
- /1.0/containers/int01?project=DMZS
- /1.0/profiles/default?project=DMZS
- /1.0/profiles/dmzint?project=DMZS
```

```
$ lxc ls -c n,4,s,P,S
+-------+---------------------+---------+----------+-----------+
| NAME  |        IPV4         |  STATE  | PROFILES | SNAPSHOTS |
+-------+---------------------+---------+----------+-----------+
| int00 | 172.30.0.125 (eth0) | RUNNING | dmzint   | 0         |
+-------+---------------------+---------+----------+-----------+
| int01 | 172.30.0.132 (eth0) | RUNNING | dmzint   | 0         |
+-------+---------------------+---------+----------+-----------+
```