Networking manual con contenedores

1- Instalación

Esta actividad se ha probado con una instalación de Ubuntu Desktop 18.04.4 en una VM de VirtualBox. Se recomienda una distribución Desktop para poder tener múltiples ventanas con shells en los diferentes contenedores.
Configure un interfaz para la VM de forma que tenga acceso a Internet, por ejemplo mediante un interfaz NAT. Comprueben que la VM tiene acceso a Internet.
Instale lxd, lxd-client, net-tools y bridge-utils (con apt).
Inicialice LXD (lxd init) indicando que no queremos crear un puente (lo haremos después manualmente)[1]:

```
# 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]:"
"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]: no"
"Would you like to configure LXD to use an existing bridge or host interface? (yes/no) [default=no]: no"
"Would you like LXD to be available over the network (yes/no) [default=no]: no"
"Would you like to stale cached images to be updates automatically? (yes/no) [default=yes] no"
"Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]: no"
```

Ahora descargaremos[2] una imagen de un contenedor para utilizar en el resto de la actividad, por ejemplo una imagen de contenedor de la Ubuntu 18.04:

```
# lxc image copy ubuntu:18.04 local:
```

Cuando termine puede comprobar que la localmente tiene haciendo:

```
# lxc image list
```

Apague la VM. Reconfigure el interfaz Ethernet de la misma en VirtualBox para que esté puenteado al interfaz físico del host. En la parte avanzada de configuración del interfaz, respecto al modo promiscuo (Promiscuous Mode) indique “Allow All” (o el equivalente). Esto permitirá que se le pase tráfico a la VM aunque no vaya destinado a su dirección MAC. En la primera parte de esta actividad no necesitaremos el interfaz Ethernet de la VM pero sí al final.

A partir de aquí intente ir dibujando lo que está creando, tanto la parte de elementos de red como elementos terminales (PCs hechos con contenedores).

[1] Estas preguntas pueden variar con la versión concreta de LXD
[2] Algunos de estos commandos puede requerir permisos de super-usuario

2- Primer contenedor

Arranque la VM. A partir de aquí llamaremos a esa VM “el host”, ya que de cara a los contenedores que vamos a crear es el host y ellos son los “guest”. Es la única VM VirtualBox que vamos a lanzar en toda la actividad.
Lance un contenedor empleando la imagen que se descargó antes:

```
# lxc launch ubuntu:18.04 cont1
```

Si lxc dice algo sobre que ese contenedor no tiene ninguna red, ignórelo. Vamos a configurarle la red manualmente, paso a paso, para entender lo que está haciendo luego lxc por nosotros. Puede ver los interfaces que tiene el contenedor por ejemplo así:

```
# lxc exec cont1 -- ifconfig -a
```

Recuerde que si no pone -a se muestran solo los interfaces que estén ‘Up’. También puede crear una shell en el contenedor para hacer el comando (ponga ‘bash’ en vez de ‘ifconfig -a’). En cualquier caso, el contenedor debería tener solo el interfaz de loopback.

Muestre el listado de procesos en el host (la VM):

```
# ps faxu
```

Busque un proceso lxc que entre sus argumentos tiene el nombre del contenedor que ha creado. Debajo de él, con una estructura en árbol, tendrá el primer proceso que se creó dentro del contenedor (/sbin/init). Ese proceso y todos los descendientes suyos están en el contenedor.
Estar en el contenedor se puede estar para diferentes aspectos, como puede ser para solo ver los procesos del contenedor, la pila de red del contenedor, la lista de usuarios del contenedor, etc. Se puede por ejemplo crear un contenedor que no tenga una pila de red independiente sino que emplee la del host, o que emplee la lista de usuarios del host. Los contenedores que crea lxd parecen tener todos los elementos independientes. Eso lo puede ver mirando los *namespaces* de uno cualquiera de esos procesos. En muchas instalaciones de Linux tiene un directorio /proc donde se pueden inspeccionar variables del kernel. En concreto dentro de él hay un directorio para cada proceso que está corriendo, con nombre el PID (Process IDentifier) del proceso. Dentro del directorio de un proceso hay otro directorio que se llama ‘ns’ y dentro de él hay una referencia a cada namespace.
Por ejemplo, en la figura 1 puede ver que el proceso init del contenedor cont1 es el 4137 y que si listamos los ficheros del directorio /proc/4137/ns nos encontramos con las referencias a los diferentes namespaces (procesos, IPCs, red, control groups, etc).

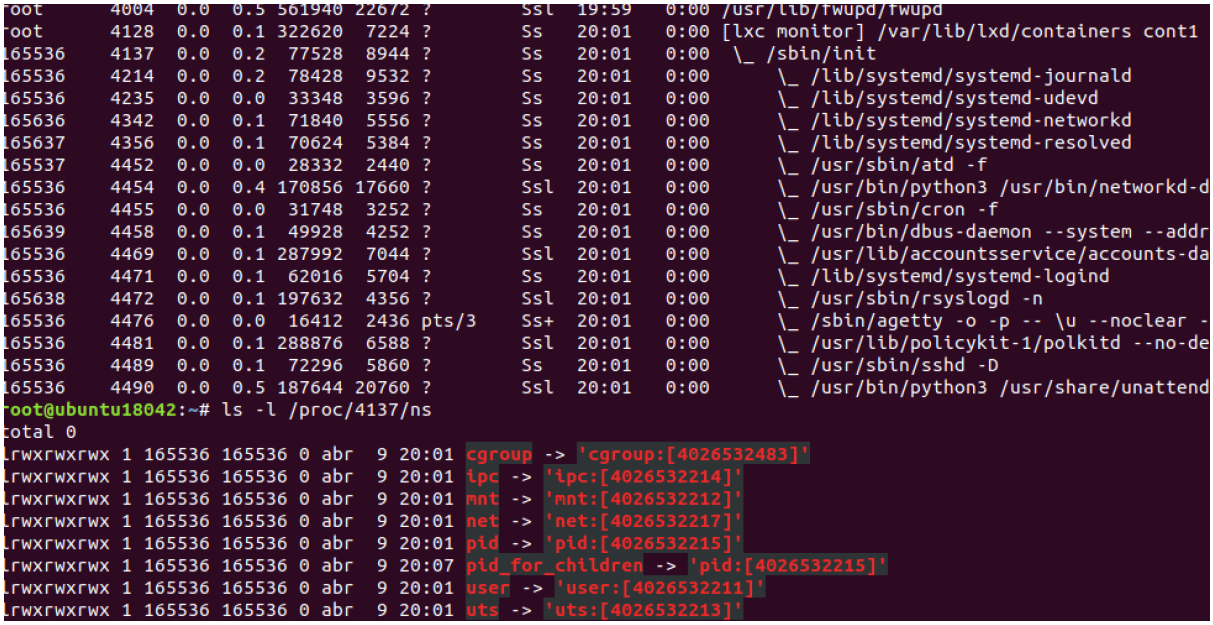

Figura 1 – Lista de procesos y contenido de /proc

Cualquier otro proceso del contenedor tiene las mismas referencias pero un proceso de fuera del contenedor emplea otros namespaces (vea la figura 2).

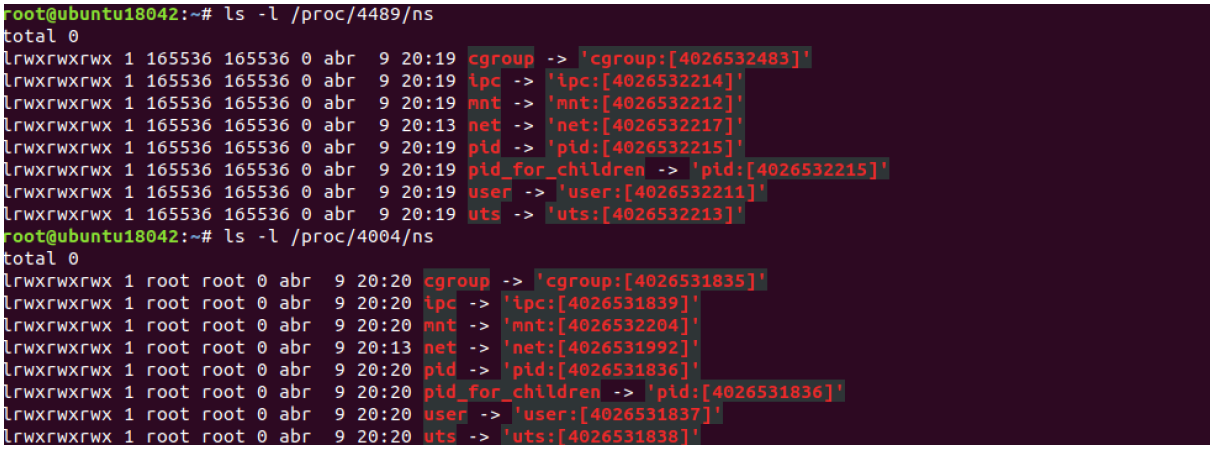

Figura 2 – Contenido de varios subdirectorios de /proc

Investigue si quiere un poco más el contenido del directorio en /proc de un proceso.
Ahora creamos un par de interfaces virtuales o veths:

```
# ip link add 'veth_host' type veth peer name 'veth_cont'
```

Puede ver ambos con ifconfig -a .
Asignaremos uno de los veth al namespace de red del contenedor que hemos creado antes. Para ello hacemos:

```
# ip link set dev veth_cont netns <PID>
```

Donde <PID> lo debemos sustituir por el PID de un proceso que esté en el contenedor, para que el comando pueda obtener de él el namespace de red al que queremos mover el interfaz.

Se puede poner nombre a cada namespace pero lxd no lo ha hecho al crearlos, así que ésta (con el PID) es la forma más rápida de hacer referencia a uno.
En el ejemplo anterior podríamos hacer:

```
# ip link set dev veth_cont netns 4489
```

Así estamos pasando el interfaz veth_cont al contenedor, dejando el otro extremo en el namespace de red del host. Si ahora vuelve a hacer ifconfig -a en el contenedor verá que tiene ese interfaz.

En este punto puede dar dirección IP a los dos extremos veth en la misma subred y con eso tendrá comunicación por esa red entre host y contenedor.

3- Contenedores interconectados

Cree un segundo contenedor y un nuevo par de veth que llamaremos veth2_cont1 y veth2_cont2. Asigne veth2_cont1 al contenedor que creó en el apartado anterior y veth2_cont2 al segundo contenedor que ha creado. Con esos nuevos interfaces puede conseguir comunicación entre los contenedores. El primer contenedor tiene ahora 2 interfaces, uno que lleva al host y el segundo que lleva al segundo contenedor. Configure el primer contenedor para que encamine el tráfico IP entre el host y el segundo contenedor. Recuerde que el contenedor intermedio debe reenviar paquetes IP, así que su pila de red debe tener activo el reenvío de paquetes en el kernel[3]. Por supuesto tendrá que introducir rutas en el host y en el contenedor 2.

En este punto el contenedor 2 tiene un enlace directo al contenedor 1 y éste tiene otro enlace directo al host. Pruebe ahora a crear un puente en el host y asignar veth_host a ese puente en lugar de emplearlo como un interfaz IP independiente (desconfigúrele la dirección IP). A continuación cree un tercer contenedor que tenga un interfaz directamente en ese puente (creando un nuevo interfaz virtual, con un extremo en él y el otro en el host conectado al puente) y pruebe la comunicación entre contenedor 1 y contenedor 3 en la misma subred IP. Consiga también que contenedor 3 se comunique con contenedor 2 enrutado a través de contenedor 1.

Se detalla todo esto a continuación.

Cree un bridge en la VM:

```
$ brctl addbr mibr0
```

Cree el nuevo contenedor (contenedor 3).
Cree un nuevo par de veth, un extremo asígnelo al contenedor 3 y el otro extremo añádalo al bridge como ya se ha hecho anteriormente.
Retire la dirección IP de veth_host que configuró en el apartado anterior y añada el interfaz a ese puente. Configure el interfaz de contenedor 3 en la misma subred que veth_cont y pruebe la comunicación puenteada entre ellos.

---

[3] `sysctl net.ipv4.ip_forward=1`

Recuerde levantar los interfaces. Cuando le asigna dirección IP a un interfaz, la propia operación los levanta, pero el incluirlos en un puente no, así que tiene que hacer 'ifconfig up' a los interfaces que han quedado en el host conectados al puente, así como tiene que hacerle 'up' al propio puente (ifconfig mibr0 up).
Actualice el dibujo de la topología.

Los contenedores que ha creado hasta el momento están aislados del exterior de la VM que estábamos llamando "el host". Si está haciendo esta actividad en su ordenador en su casa es muy probable que su LAN del hogar disponga de un router de acceso con servidor de DHCP. Si está en la universidad con un interfaz WiFi está en una situación similar. Supondremos que su máquina física está obteniendo dirección IP del router de la operadora mediante DHCP y que este último ofrece direcciones IP a cualquiera de la LAN. Si no hay servidor de DHCP o no puede emplearlo, como sucedería en el laboratorio, haga una configuración manual estática.
Habíamos dejado el interfaz de la VM configurado en VirtualBox para que estuviera puenteado con la NIC de su máquina física. Cree un segundo puente en la VM (brctl addbr) y asigne a él el interfaz de la VM (que está puenteado al físico). Cree un cuarto contenedor y un par de veths. Uno de los extremos asígnelo al contenedor y el otro añádalo al nuevo puente. Ahora pruebe a obtener IP por DHCP *en el nuevo contenedor* corriendo en el mismo:

```
$ dhclient -v <interfaz>
```

Donde <interfaz> es el nombre del interfaz que tiene el contenedor y que le lleva al puente (que lleva al interfaz físico).
Si todo ha ido bien entonces el contenedor 4 tendrá ahora una dirección IP asignada por el servidor de DHCP de su router de salida y con esa configuración tendrá salida a Internet (también le habrá dado la configuración de ruta por defecto y servidor de DNS).
Añada estos nuevos elementos al dibujo.

Ahora podríamos por ejemplo dar acceso al resto de contenedores a su LAN del hogar, simplemente para que puedan acceder y ser accedidos por otras máquinas de su LAN (por ejemplo porque quiera usted correr algún tipo de servidor en un contenedor). Por ejemplo podríamos crear un enlace entre contenedor 4 y contenedor 2 y ajustar las tablas de rutas. ¿Qué problemas podrían surgir y cómo se podrían resolver? ¿Y si queremos que esos contenedores tengan también acceso a Internet?

Vuelva al guión de la práctica 1 e intente crear la topología que se presentaba para el primer punto de control pero ahora con contenedores.

Todo lo que hemos descrito aquí tiene un doble nivel de virtualización por partir de un Linux corriendo en una VM VirtualBox. Si su PC físico tuviera Linux como sistema operativo de host podríamos haber creado los contenedores directamente en él, pero el doble nivel de virtualización también es una actividad interesante para forzarse a tener claros los conceptos.