

Práctica 3 – Networking manual en LXC

1. Introducción y objetivos

El objetivo de esta práctica es probar la creación de contenedores e interfaces virtuales Ethernet asignados a contenedores que se encuentran todos ellos en el mismo Kernel Linux.

Esta actividad se ha probado con una instalación de Ubuntu Server 20.04 LTS en una VM de VirtualBox. Se puede emplear también una distribución Desktop u otras distribuciones donde se pueda instalar LXD. Se puede usar también un Linux instalado directamente sobre el hardware en lugar de en una VM. Puede ser recomendable una distribución con interfaz gráfico para poder tener múltiples terminales en diferentes contenedores con comodidad, aunque también se puede resolver en una instalación sin interfaz gráfico empleando un terminal multiplexer como por ejemplo `tmux` o `screen`. En las máquinas del laboratorio se debe llevar a cabo en una VM.

Para algunas partes de la práctica es necesario que el host disponga de un interfaz Ethernet conectado a una LAN.

2. Configuración de LXD

Configure un interfaz para la VM de forma que tenga acceso a Internet, por ejemplo, mediante un interfaz NAT. Compruebe que la VM tiene acceso a Internet.

Los siguientes comandos se recomienda correrlos como el usuario root.

Inicialice LXD (`lxd init`) indicando que no queremos crear un puente (lo crearemos después manualmente para tener mayor control sobre lo que se está haciendo)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]:
Name of the storage backend to use (zfs, ceph, btrfs, dir, lvm) [default=zfs]: dir
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 the LXD server to be available over the network? (yes/no) [default=no]: yes
Address to bind LXD to (not including port) [default=all]:
Port to bind LXD to [default=8443]:
Trust password for new clients:
Again:
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]:
```

Descargue un par de imágenes de contenedor para el resto de la actividad. Puede listar las imágenes disponibles en repositorios con:

```
# lxc image list ubuntu:
# lxc image list images:
```

En este ejemplo² descargamos la imagen de contenedor de Ubuntu 20.10 y de una minimalista Alpine 3.19:

```
# lxc image copy ubuntu:20.10 local: --copy-aliases
# lxc image copy images:alpine/3.19 local: --copy-aliases
```

Cuando termine puede comprobar que tiene las imágenes localmente haciendo:

```
# lxc image list
```

¹ Estas preguntas pueden variar con la versión concreta de LXD

² En caso de no encontrar esas versiones concretas descargue las disponibles actualmente

Usaremos un interfaz específico de la VM que esté puenteado con la NIC Ethernet real (para ciertas partes de la práctica **no** servirá que sea una NIC WiFi). Apague la VM y 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. La VM no necesitará más acceso a Internet pues ya ha descargado las imágenes de contenedor que se van a emplear.

Verifique que en la VM dispone de tcpdump y de brctl y si no es así instale los paquetes necesarios.

Puede emplear algún interfaz Host-only para acceder a la VM con mayor comodidad. También le puede servir para copiar trazas pcap fuera de la VM para abrirlas en Wireshark con comodidad.

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).

3. Primer contenedor

Arranque la VM. A partir de aquí llamaremos a esa VM “el hostVM”, ya que de cara a los contenedores que vamos a crear es el host y ellos son los “guests”.

Lance un contenedor empleando una de las imágenes que descargó antes:

```
# lxc launch local:alpine/3.19 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 normalmente lxc por nosotros. Con el comando anterior el contenedor tiene el nombre 'cont1'; si no indica un nombre será asignado al azar por lxc. Puede ver los contenedores creados con:

```
# lxc list
```

Ahí verá también los nombres de los contenedores.

Puede ver los interfaces que tiene el contenedor por ejemplo así:

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

Recuerde que si no pone la opción -a a ifconfig 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', o la Shell que esté disponible en esa imagen, que en caso de la Alpine será 'sh'). En cualquier caso, el contenedor debería tener solo el interfaz de loopback.

Puede ver los procesos corriendo en el contenedor:

```
# lxc exec cont1 -- ps faxw
```

Muestre el listado de procesos en el hostVM:

```
# ps faxu
```

Podrá encontrar en este último listado también los procesos que están en el contenedor y verá que tienen diferente PID que cuando los vio dentro del contenedor, lo cual indica que el contenedor tiene un PID namespace diferente. Compruébelo buscando en /proc las referencias a los namespaces de un proceso cualquiera en el contenedor y fuera de él.

Puede ver información sobre el contenedor con:

```
# lxc info cont1
```

Podrá encontrar ahí por ejemplo el PID del primer proceso creado en el contenedor.

Ahora creamos un par de interfaces virtuales o veths en el host:

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

Puede ver ambos con `ifconfig -a`

Asignaremos uno de los veth (`veth_cont`) 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 (visto desde el PID namespace actual), 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
```

Suponiendo que el proceso 4889 esté en el contenedor 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.

4. 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 hostVM 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 kernel³. Por supuesto tendrá que introducir rutas en el hostVM 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. Cree un puente en el hostVM⁴ y asigne `veth_host` a ese puente en lugar de emplearlo como un interfaz IP independiente (desconfigúrele la dirección IP asignándole la dirección 0). A continuación, cree un tercer contenedor que tenga un interfaz directamente en ese puente; para ello cree una nueva pareja de veths, con un extremo en el contenedor 3 y el otro en el hostVM y conectado al puente. Asigne dirección IP al extremo en el contenedor 3 correspondiente a la subred de esa LAN (la misma del `veth_cont`) 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.

Actualice el dibujo de la topología. Emplee `tcpdump` para ver el tráfico en diferentes interfaces virtuales.

Punto de control 1 (60%): Muestre el dibujo de la topología de contenedores y la comunicación entre ellos.

³ `sysctl net.ipv4.ip_forward=1`

⁴ `brctl` (si no lo encuentra es parte de las `bridge-utils`)

Los contenedores que ha creado hasta el momento están aislados del exterior de la VM que estábamos llamando “el hostVM”. 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. Supondremos que su máquina física (el auténtico host) está obteniendo dirección IP del router de la operadora mediante DHCP y que este router ofrece direcciones IP a cualquiera de la LAN. Si no hay servidor de DHCP o no puede emplearlo, como sucedería si está haciendo esta práctica con los PCs del laboratorio, haga una configuración manual estática cuando se le pida a en el siguiente párrafo obtenerla por DHCP.

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 el hostVM y asigne a él el interfaz de la VM (el que está puenteando VirtualBox al interfaz Ethernet físico). Cree un cuarto contenedor y un par de veths. Uno de los extremos asígnelo a este nuevo contenedor y el otro añádalo al nuevo puente. Ahora pruebe a obtener IP por DHCP *en el nuevo contenedor* corriendo en el mismo un cliente de DHCP (puede emplear `dhclient` en Ubuntu o `udhcpd` en Alpine). Como se ha explicado en el párrafo anterior, esto solo funcionará si en la LAN externa hay un servidor de DHCP que vaya a contestar a sus peticiones. Debería funcionar si está haciendo la práctica en casa conectando el host por cable a un router doméstico con servidor de DHCP configurado (y que asigne direcciones IP a cualquiera que las pida).

Si hace la práctica con un PC del laboratorio configure manualmente el interfaz IP de ese contenedor (el profesor en alguna práctica anterior es probable que ya le haya indicado direccionamiento que puede emplear de la subred del laboratorio). Puede ver el tráfico de su contenedor en el interfaz del host real empleando `tcpdump`/Wireshark en él sobre el interfaz físico y ver así los mensajes de DHCP (si lo está empleando). En caso de intentar usar un interfaz WiFi deberá investigar qué problemática plantea a este escenario.

Si todo ha ido bien, el contenedor 4 tendrá ahora una dirección IP asignada por el servidor de DHCP de su router de salida (o manualmente) 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/laboratorio, 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?

Punto de control 1 (30%): Muestre los dibujos de los diferentes escenarios y el funcionamiento de alguno de los últimos escenarios.

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

Punto de control 2 (10%): Muestre este escenario en funcionamiento.