LABORATORIO DE MVs

Sumario

Introducción

Vamos a utilizar la tecnología de virtualización Qemu/KVM para implantar un laboratorio de trabajo con Máquinas Virtuales. Para ello crearemos una infraestructura que luego podremos utilizar para modificar y adaptar el laboratorio a nuestros requisitos.

La práctica consiste en dos procedimientos

- Creación de bases para MV y clones enlazados
- Creación de Networks de tipo routed al que podremos conectar las MVs, de modo que habilitamos un mecanismo para separar las MVs en distintas redes, con posibilidad de interconexión

Creación de Network para Laboratorio

Crearemos un elemento de tipo NAT Network a la que se conectarán todas las MV que crearemos.

Vamos a tomar como referencia el archivo de definición de la Network default, para ello volcaremos a un archivo su contenido, lo editaremos y por último crearemos la nueva Network.

```
virsh net-dumpxml default > labnet.xml
```

Editamos el archivo resultante para que contenga los siguiente

```
<network>
  <name>labnet</name>
  <uuid>5ad8ead8-c62c-11e7-9ba3-c7038bd93603</uuid>
  <forward mode='route'/>
  <bridge name='virbr2' stp='on' delay='0'/>
  <mac address='de:ad:be:ef:e3:07'/>
  <domain name='labnet'/>
  <ip address='192.168.10.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.10.100' end='192.168.10.254'/>
    </dhcp>
  </ip>
</network>
```

Los datos correspondientes al **uuid**, que debe de ser único, lo generamos con

```
uuid
```

La dirección **MAC** del dispositivo vinculado a la Network, virbr2, lo generamos con el comando

```
printf 'DE:AD:BE:EF:%02X:%02X\n' $((RANDOM%256)) $((RANDOM%256))
```

Los otros elementos a definir serían* **name**: el nombre de la Network

- **forward**: indica el modo de reenvío, en este caso será una **red enrutada (routed mode)**, de modo que podremos comunicar nuestras MVs con otras conectadas a otra Network que opere en ese modo.
- **ip** address
- **rango DHCP** de direcciones asignables

Una vez editados los cambios en el archivo ejecutamos

```
virsh net-define labnet.xml
```

Por último activamos la Network y la definimos como de activación automática al iniciar el sistema

```
virsh net-start labnet
virsh net-autostart labnet
```

Creación de las MV base

Creación de MV base Debian 9

Usaremos la utilidad **virt-install**

```
virt-install --name lab-debian9-base --memory 512 --vcpus 1 --disk size=5 --cdrom /home/javier/Descargas/debian-9.2.1-amd64-DVD-1.is
```

Las opciones del comando son autoexplicativas, notar que se ha conectado el domain (MV) a la network ?labnet? que creamos en el apartado anterior.

Seguimos el proceso de instalación hasta el final, podemos ver el nuevo domain creado con el comando

```
virsh list

Id    Name                           State
----------------------------------------------------
 2     lab-debian9-base               running
```

Creación de MV base Windows 10

Seguiremos los pasos anteriores pero ahora para la instalación de un domain sobre Windows 10. Usaremos un comando diferente al anterior, esto se debe a que vamos a utilizar drivers paravirtualizados, que aumentarán el rendimiento de entrada/salida para dispositivos de bloques y tarjeta de red.

Podremos descargar la iso con los Driver paravirtualizados para Windows desde:

https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso

En primer lugar crearemos la imagen para el volumen del disco virtual

```
qemu-img create -f qcow2 lab-win10-base.qcow2 20G
```

Ahora lanzaremos el comando para la creación del domain, es decir, para crear la MV

```
qemu-system-x86_64 --enable-kvm -smp 1 -m 1512 -cpu host -vga qxl -drive file=/home/javier/Descargas/es_windows_10_multi-edition_vl_
```

Los parámetro del comando indican

- kvm está activado
- un procesador
- 1512 MB de RAM
- dispositivo gráfico vga qxl
- dispositivo cdrom conectado a la iso de instalación de windows
- dispositivo cdrom conectado a la iso que contiene los drivers paravirtualizados para windows
- dispositivo de tipo disco asociado a la imagen creada en el comando qemu-img anterior
- tarjeta de red paravirtualizada

Durante el proceso de instalación que desencadena el comando anterior veremos como, en el punto en el que se selecciona el disco para la instalación no aparece ningún dispositivo en la lista. Esto se debe a que hemos definido el dispositivo de almacenamiento en el comando anterior de tipo paravirtualizado, es decir ?virtio?. Por defecto el instalador no trae soporte para ese interfaz, por tanto deberemos cargar los correspondientes controladores de dispositivos desde el cdrom adicional que habíamos añadido en las opciones de qemu-system.

Llegados a esa pantalla seleccionamos la opción de añadir controladores de dispositivos adicionales, añadimos los siguientes desde unidad de cdrom a la que está conectado la iso de drivers paravirtualizados

- **Desde viostor**->amd64: Drivers paravirtualizados para el disco
- **Desde Net/KVM**->amd64: Drivers paravirtualizados para la tarjeta de red

Tras añadir estos dos controladores seguimos con la instalación de Windows 10 del modo habitual

Al finalizar la instalación cerramos la MV y ejecutamos:

```
virt-install --name lab-win10-base --memory 1512 --vcpus 1 --disk path=/var/lib/libvirt/images/lab-win10-base.qcow2,bus=virtio --net
```

El comando anterior crea un domain con virt-install, que usará la imagen creada con el comando qemu-system anterior como medio de almacenamiento de la MV, que usaba el driver virtio, así como una interfaz de red, también operando en virtio, conectada a la NAT Network labnet.

Creación de las MV de trabajo

Creación de imagen enlazada sobre la imagen base Debian 9

Vamos a crear una imagen enlazada a la imagen creada durante el proceso de instalación de la MV para debian9

```
qemu-img create -f qcow2 -b /var/lib/libvirt/images/lab-debian9-base.qcow2 /var/lib/libvirt/images/lab-debian9-mv1.qcow2
```

Este comando crea la nueva imagen, tomando como base la imagen de la máquina creada anteriormente, de este modo obtenemos un ?clon enlazado? a esa imagen. **lab-debian9-mv1.qcow2** almacenará solamente las diferencias respecto a la imagen de base **lab-debian9-base.qcow2**

Veamos los datos de la imagen

```
qemu-img info /var/lib/libvirt/images/lab-debian9-mv1.qcow2
```

Muestra

```
image: /var/lib/libvirt/images/lab-debian9-mv1.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 196K
cluster_size: 65536
backing file: /var/lib/libvirt/images/lab-debian9-base.qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
    refcount bits: 16
    corrupt: false
```

Creación de domain de trabajo para Debian9

Vamos a crear un domain, MV, de trabajo sobre la imagen que acabamos de crear. Para ello usaremos virt-clone

```
virt-clone --original=lab-debian9-base --name=lab-debian9-mv1 --preserve-data --file=/var/lib/libvirt/images/lab-debian9-mv1.qcow2
```

Creación de imagen enlazada sobre la imagen base Windows10

De modo análogo a como hemos procedido con la imagen para debian 9 generamos el archivo de imagen enlazado a la imagen de base para windows 10

```
qemu-img create -f qcow2 -b /var/lib/libvirt/images/lab-win10-base.qcow2 /var/lib/libvirt/images/lab-win10-mv1.qcow2
```

Veamos los datos de la imagen

```
qemu-img info /var/lib/libvirt/images/lab-win10-mv1.qcow2
```

Muestra

```
image: /var/lib/libvirt/images/lab-win10-mv1.qcow2
file format: qcow2
virtual size: 20G (21474836480 bytes)
disk size: 196K
cluster_size: 65536
backing file: /var/lib/libvirt/images/lab-win10-base.qcow2
Format specific information:
    compat: 1.1
    lazy refcounts: false
    refcount bits: 16
    corrupt: false
```

Creación de domain de trabajo para Windows10

Creamos el domain de trabajo para Windows 10 basado en la imagen que acabamos de crear

```
virt-clone --original=lab-win10-base --name=lab-win10-mv1 --preserve-data --file=/var/lib/libvirt/images/lab-win10-mv1.qcow2
```

Veamos un listado de los domains creados hasta el momento

```
virsh list --all
```

Mostraría

```
 Id    Name                           State
----------------------------------------------------
 -     lab-debian9-base               shut off
 -     lab-debian9-mv1                shut off
 -     lab-win10-base                 shut off
 -     lab-win10-mv1                  shut off
```

Creación de una segunda Network en modo Routed

Siguiendo un procedimiento similar al explicado en la sección ?Creación de Network para laboratorio?, vamos a crear una segunda Network de nombre **labnet2**, cuyo archivo de configuración labnet2.xml será el siguiente

```
<network>
  <name>labnet2</name>
  <uuid>85e79c24-b23f-4b4c-bc4f-35e07b887c37</uuid>
  <forward mode='route'/>
  <bridge name='virbr3' stp='on' delay='0'/>
  <mac address='52:54:00:20:39:60'/>
  <domain name='labnet2'/>
  <ip address='192.168.11.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.11.128' end='192.168.11.254'/>
    </dhcp>
```

```
  </ip>
</network>
```

Una vez editado el archivo creamos el objeto Network

```
virsh net-define labnet2.xml
```

Por último activamos la Network y la definimos como de activación automática al iniciar el sistema

```
virsh net-start labnet2
virsh net-autostart labnet2
```

Creación de domain en la nueva Network

Vamos a continuación a crear un nuevo domain basado en la imagen lab-debian9-base conectado a la Network labnet2.

Creamos en primer lugar la imagen para la MV

```
qemu-img create -f qcow2 -b /var/lib/libvirt/images/lab-debian9-base.qcow2 /var/lib/libvirt/images/lab-debian9-mv2.qcow2
```

A continuación creamos el domain con el comando virt-install

```
virt-install --name lab-debian9-mv2 --memory 512 --vcpus 1 --disk path=/var/lib/libvirt/images/lab-debian9-mv2.qcow2,bus=virtio --ne
```

Como puede verse en el comando anterior, creamos el domain con las características siguientes

- Nombre lab-debian9-mv2
- 512 MB de RAM
- 1 CPU virtual
- Como volumen de disco usaremos la imagen creada anteriormente. El dispositivo será gestionado con driver virtio
- Conectados el domain a la Network labnet2. El dispositivo será gestionado con driver virtio

Comunicación entre las Networks

Activación del reenvío IPv4 en el Host

Para que las 2 Networks, que operan en modo Routed, puedan enviar tráfico entre sí es preciso activar en el Host el reenvío IPv4 en el Kernel. Para ello editamos el archivo /**etc**/**sysctl.conf**, descomentando la línea

net.ipv4.ip_forward=1

A continuación cargamos en el kernel los parámetros del archivo

```
sysctl -p /etc/sysctl.conf
```

Inclusión de regla NAT en iptables para el enmascarado del tráfico saliente

Con la configuración de reenvío del Host activada podremos enviar paquetes entre los domains conectados a labnet y labnet2. Sin embargo para utilizar la interfaz conectada públicamente, a ?Internet?, del Host, es preciso definir reglas iptables de tipo NAT para la traducción automática de las direcciones IP.

En los siguientes comandos se supone que el dispositivo de red, en el Host, con salida pública es **wlp108s0**, esto puede cambiar en cada configuración, para consultar los dispositivos, ejecutamos en el Host

```
ip link show
```

Para la salida del tráfico desde la Network **labnet**

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -d 0/0 -o wlp108s0 -j MASQUERADE
```

Para la salida del tráfico desde la Network **labnet2**

```
iptables -t nat -A POSTROUTING -s 192.168.11.0/24 -d 0/0 -o wlp108s0 -j MASQUERADE
```

Pasos previos al test de conectividad

Tenemos por tanto el siguiente escenario

lab-debian9-mv1

- conectada a la Network labnet
- IP en el rango 192.168.10.0

lab-debian9-mv2

- conectada a Network labnet2
- IP en el rango 192.168.11.0

Determinamos el dispositivo Switch Virtual de labnet

```
virsh net-info labnet
```

Arroja en mi caso la salida:

```
Name:           labnet
UUID:           5ad8ead8-c62c-11e7-9ba3-c7038bd93603
Active:         yes
Persistent:     yes
Autostart:      yes
Bridge:         virbr2
```

Podemos ver que el Bridge asociado es virbr2, vamos a determinar la dirección IP del dispositivo en el Host

```
ip addr show virbr2

virbr2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether de:ad:be:ef:e3:07 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.1/24 brd 192.168.10.255 scope global virbr2
       valid_lft forever preferred_lft forever
```

En la salida anterior vemos que efectivamente para la Network **labnet** la subred asociada es **192.168.10.0**, y el dispositivo Switch Virtual asociado en el Host, **virbr2**, tiene asociada la IP **192.168.10.1**, es decir la IP más baja en ese rango.

Si repetimos los pasos anteriores para **labnet2** concluiremos que el Bridge asociado es **virbr3** y tiene asociado el rango **192.168.11.0**, con IP para el Bridge en el host **192.168.11.1**

A continuación iniciamos las MV

```
virsh start lab-debian9-mv1
virsh start lab-debian9-mv2
```

Determinamos las concesiones DHCP de ambas Network para conocer las direcciones IP de las MV iniciadas

```
virsh net-dhcp-leases labnet

 Expiry Time          MAC address        Protocol  IP address                Hostname        Client ID or DUID
-------------------------------------------------------------------------------------------------------------------
 2017-11-12 17:05:29  52:54:00:ff:a7:46  ipv4      192.168.10.217/24         debian          -

virsh net-dhcp-leases labnet2

 Expiry Time          MAC address        Protocol  IP address                Hostname        Client ID or DUID
-------------------------------------------------------------------------------------------------------------------
 2017-11-12 17:05:32  52:54:00:5a:42:f0  ipv4      192.168.11.171/24         debian          -
```

En conclusión

- **lab-debian9-mv1** tiene asociada la IP **192.168.10.217**
- **lab-debian9-mv2** tiene asociada la IP **192.168.11.171**

Test de conectividad

Desde lab-debian9-mv1

Efectuamos un test de conectividad a lab-debian9-mv2

```
ping -c2 192.168.11.171
```

La salida debería mostrar las respuestas desde el otro guest. Debería haber respuesta por parte del otro guest pues en un paso anterior configuramos el host para activar el reenvío, forward, de paquetes IPv4

Efectuamos un test de conectividad público

```
ping -c2 google.es
```

Debería haber respuesta por parte del host público debido a que anteriormente configuramos la NAT en el host para la subred asociada a labnet

Desde lab-debian9-mv2

Efectuamos un test de conectividad a lab-debian9-mv1

```
ping -c2 192.168.10.217
```

La salida debería mostrar las respuestas desde el otro guest. Debería haber respuesta por parte del otro guest pues en un paso anterior configuramos el host para activar el reenvío, forward, de paquetes IPv4

Efectuamos un test de conectividad público

```
ping -c2 google.es
```

Debería haber respuesta por parte del host público debido a que anteriormente configuramos la NAT en el host para la subred asociada a labnet2

Volver

JavierFP 16:47 12 nov 2017 (CET)