KVM: Setting up a network bridge in the host

If you are planning to only access the guest from the KVM host and to access the outside network from the guests, the default networking set up will be sufficient for you, and you can skip this step and go to the next section. If the KVM guests need full network access (to and from an external host), one of the options is to set up a Linux bridge in the host. Note that the Linux bridge configuration does not work in a wireless host environment.

About this task

You should set up this Linux bridge before the guest OS installation so that the bridge is available for selection during the guest operating system installation. Bridged networking allows you to link two Ethernet network segments using packet forwarding technology. More information about Linux bridges can be found at http://en.wikipedia.org/wiki/Network_bridge.

Note: Be careful when you configure the bridge. If you are accessing the host machine using the same network card you are configuring for the bridge, any discrepancy might cause you to lose your network connection.

Before setting up this bridge, make sure that the network card that you want to use for the bridge is providing the network connection you want for your KVM modules and is working. This card should be setup to provide the same networking capability you want your guest KVM to have.

In the following example, **eth0** is the network card used. This card has already been configured for external access.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:14:5E:C2:1E:40
          inet addr:10.10.1.152  Bcast:10.10.1.255  Mask:255.255.255.0
          inet6 addr: fe80::214:5eff:fec2:1e40/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:664 errors:0 dropped:526 overruns:0 frame:0
          TX packets:163 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:69635 (68.0 KiB)  TX bytes:25091 (24.5 KiB)
          Interrupt:74 Memory:da000000-da012800
................
```

If your network card is not set up yet, create its network script and save it in the **/etc/sysconfig/network-scripts/** directory with help from this site: http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Deployment_Guide-en-US/s2-networkscripts-interfaces-eth0.html or the libvirt wiki page about writing network scripts at http://wiki.libvirt.org/page/Networking.

Follow these steps to create a public bridge in the host system.

Procedure

1. Back up the corresponding network script file at a different location. Note that this is important because you will need to refer back to the file and also for network recovery. Issue the following command to back up the network script for **ifcfg-eth0** to the /root directory:

   ```
   # cp /etc/sysconfig/network-scripts/ifcfg-eth0 /root/.
   ```

   Note: Do not copy this file to the same network script directory or any of its subdirectories.
2. Create another copy of the network script for defining a Linux® bridge associated with the network card to a new file called **/etc/sysconfig/network-scripts/ifcfg-br0**, where *br0* is the name of the bridge. The complete content of the Linux bridge's configuration file will be based on what is already in the working script of your network card.

   ```
   # cd /etc/sysconfig/network-scripts/
   ```

```
# cp ifcfg-eth0 ifcfg-br0
```

Note: The name of the bridge is arbitrary and you can name it differently as long as you use the same name in the next step inside the network card's script file.

3. Edit both script files to ensure that the network environment remains the same except that now the packets will go through the bridge. Your network card most likely is configured with a static IP address (`BOOTPROTO=static`) or is configured to get an IP address from a DHCP server (`BOOTPROTO=dhcp`).
   - If your network card is configured with static IP address, your original network script file should look similar to this example:

     ```
     DEVICE=eth0
     BOOTPROTO=static
     HWADDR=00:14:5E:C2:1E:40
     IPADDR=10.10.1.152
     NETMASK=255.255.255.0
     ONBOOT=yes
     ```

     The following table shows the contents of the two network configuration scripts after editing was completed. Edit your scripts accordingly.

     Table 1. Bridging network files comparison

/etc/sysconfig/network-scripts/ifcfg-eth0	**etc/sysconfig/network-scripts/ifcfg-br0**

`DEVICE=eth0`
`TYPE=Ethernet`
`HWADDR=00:14:5E:C2:1E:40`
`ONBOOT=yes`
`NM_CONTROLLED=no`
`BRIDGE=br0`
	`DEVICE=br0`
	`TYPE=Bridge`
	`NM_CONTROLLED=no`
	`BOOTPROTO=static`
	`IPADDR=10.10.1.152`
	`NETMASK=255.255.255.0`
	`ONBOOT=yes`

     In the left column is the network script file for network card (**eth0**). Note that all the information directly about this network card stays the same, such as the `DEVICE` (device name), `TYPE` (device type), `HWADDR` (hardware address), and `ONBOOT` (whether the device will be activated on boot). The `NM_CONTROLLED=no` option was added because both device should not be controlled by the Network Manager for bridge to work. `BRIDGE=br0` is also added to associate this card with the bridge.

     In the right column is the network script for the bridge (**br0**). Note that all information directly related to this bridge is there, such as `DEVICE` (device name), `TYPE` (device type, `Bridge` is case-sensitive and must be added exactly as represented here with an upper case `'B'` and lower case `'ridge'`), and the `NM_CONTROLLED=no` option to disable Network Manager control to this device. The rest are retained from the network card configuration file (`BOOTPROTO`, `IPADDR`, `NETMASK`, and `ONBOOT`). Note that there should not be a hardware address in this file. These values set up the bridge to behave like the network card: the **ifcfg-br0** file acting as an extension of the **ifcfg-eth0** file where the `BRIDGE=br0` is pointing to the **ifcfg-br0** file.

   - If your network card is configured with dynamic IP address, your original network script file should look similar to this example:

     ```
     DEVICE=eth0
     BOOTPROTO=dhcp
     HWADDR=00:14:5E:C2:1E:40
     ONBOOT=yes
     ```

     The following table shows the contents of the two network configuration scripts after editing was completed. Edit your scripts accordingly.

     Table 2. Bridging network files comparison

/etc/sysconfig/network-scripts/ifcfg-eth0	**/etc/sysconfig/network-scripts/ifcfg-br0**
`DEVICE=eth0`	`DEVICE=br0`
`TYPE=Ethernet`	`TYPE=Bridge`
`HWADDR=00:14:5E:C2:1E:40`	`NM_CONTROLLED=no`
`ONBOOT=yes`	`BOOTPROTO=dhcp`
`NM_CONTROLLED=no`	`ONBOOT=yes`
`BRIDGE=br0`	

In the left column is the network script file for network card (**eth0**), which looks exactly like the one in the static IP address scenario. Note that all the information directly about this network card stays the same, such as the `DEVICE` (device name), `TYPE` (device type, `Bridge` is case-sensitive and must be added exactly as represented here with an upper case `'B'` and lower case `'ridge'`.), `HWADDR` (hardware address), and `ONBOOT` (whether the device will be activated on boot). The `NM_CONTROLLED=no` option was added because both device should not be controlled by the Network Manager for the bridge to work. `BRIDGE=br0` is also added to associate this card with the bridge.

In the right column is the network script for the bridge (**br0**). Note that similar all information directly related to this bridge is there, such as `DEVICE` (device name), `TYPE` (device type), and the `NM_CONTROLLED=no` option to disable Network Manager control to this device. The rest are retained from the network card configuration file (`BOOTPROTO` and `ONBOOT`). Note that there should not be a hardware address in this file. These values will set up the bridge to behave like the network card: the **ifcfg-br0** file acting as an extension of the **ifcfg-eth0** file where the `BRIDGE=br0` is pointing to the **ifcfg-br0** file.

4. Restart the network to verify that the configuration works. Note that if you configured the network incorrectly, the network connection may drop and you may lose access to your machine. Check the scripts once again carefully, and then restart the network by running the following command:

   ```
   # service network restart
   ```

5. Disable Netfilter processing in the bridged traffic. Append the following lines to the **/etc/sysctl.conf** file:

   net.bridge.bridge-nf-call-ip6tables = 0
   net.bridge.bridge-nf-call-iptables = 0
   net.bridge.bridge-nf-call-arptables = 0

   Note: For more information about why disabling Netfilter processing is a good security measure, see the "Network isolation options" section of the *Securing KVM guests and the host system* blueprint at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/kvmsec/kvmsecstart.htm.

6. Reload the kernel parameters with the **sysctl** command:

   ```
   # sysctl -p
   net.ipv4.ip_forward = 0
   ...
   net.bridge.bridge-nf-call-ip6tables = 0
   net.bridge.bridge-nf-call-iptables = 0
   net.bridge.bridge-nf-call-arptables = 0
   ```

7. Check that your network still has the same behavior as it did before you made the bridging changes. However, now **ifconfig** has a different output. The following example shows the first two entries of **ifconfig** in the test environment. Note that the bridge **br0** now acts for **eth0**:

   ```
   br0       Link encap:Ethernet  HWaddr 00:14:5E:C2:1E:40
             inet addr:10.10.1.152  Bcast:10.10.1.255  Mask:255.255.255.0
             inet6 addr: fe80::214:5eff:fec2:1e40/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:125 errors:0 dropped:0 overruns:0 frame:0
             TX packets:81 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:0
             RX bytes:16078 (15.7 KiB)  TX bytes:18542 (18.1 KiB)
   eth0      Link encap:Ethernet  HWaddr 00:14:5E:C2:1E:40
   ```

```
          inet6 addr: fe80::214:5eff:fec2:1e40/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:206 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27308 (26.6 KiB)  TX bytes:13881 (13.5 KiB)
          Interrupt:74 Memory:da000000-da012800
```

You can also see this bridge by running the following command:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
br0             8000.000e0cb30550       no              eth0
```

Results

Your Linux bridge should be up and running.