Bridged network

A bridged network shares a real Ethernet device with virtual machines (VMs). Each VM can bind directly to any available IPv4 or IPv6 addresses on the LAN, just like a physical computer. Bridging offers the best performance and the least headache out of the libvirt network types.

Limitations

The libvirt server must be connected to the LAN via Ethernet. If it is connected wirelessly, a *Routed network* or *NAT-based network* are the only options.

Limitations for dedicated servers

A bridge is only possible when there are enough IP addresses to allocate one per VM. This is not a problem for IPv6, as hosting providers usually provide many free IPv6 addresses. However, extra IPv4 addresses are rarely free. If you only have one public IPv4 address (and need to serve clients over IPv4), either buy more IPv4 addresses or create a *NAT-based network*.

Hosting providers often only allow the MAC address of the server to bind to IP addresses on the LAN, which prevents bridging. Your provider may let you rent a private VLAN that allows VMs to bind directly to IP addresses, but if this is too costly then consider a *Routed network*.

Initial steps

In this example:

- The server has an Ethernet device called `eth0`.
- VMs share `eth0`, which is enslaved into a bridge called `br0`.
- The hosting provider has allocated two address blocks (see CIDR notation):
  - one public IPv4 address block (`203.0.113.160/29`)
  - one public IPv6 address block (`2001:db8::/64`)
- The server binds statically to `203.0.113.166` and `2001:db8::1`.
- VMs can bind to any available IPv4 and IPv6 addresses in the allocated blocks.

Identify the MAC address of the Ethernet device for later.

```
# ip address show dev eth0 | awk '$1=="link/ether" {print $2}'
19:7c:3b:92:ec:ee
```

For performance and security reasons, disable netfilter for bridges. Create `/etc/sysctl.d/bridge.conf` with these contents:

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

Create `/etc/udev/rules.d/99-bridge.rules` with the following contents. This udev rule applies the sysctl settings above when the bridge module is loaded. (If using Linux kernel 3.18 or later, change `KERNEL=="bridge"` to `KERNEL=="br_netfilter"`.)

```
ACTION=="add", SUBSYSTEM=="module", KERNEL=="bridge", RUN+="/sbin/sysctl -p /etc
```

Intranet or home lab

The example is for a dedicated server, but the instructions are equally useful for an Intranet or home lab. There are two small differences:

- Devices on the LAN are likely allocated private IP addresses (eg, `192.168.0.0/24`) by a local router. Simply replace the public addresses in the example with appropriate private addresses for your LAN.
- You may need to configure the router to always assign the same IP address to the MAC address of the server (and to any VMs that need a static address).

Configure the bridge

> **Note**
>
> NetworkManager has supported bridges since version 0.9.8, so there is usually no need to disable it. Even the ancient version of NetworkManager on Red Hat Enterprise Linux 6 is heavily patched and supports bridges.

Follow the instructions below for either Red Hat Enterprise Linux or Debian.

Red Hat Enterprise Linux (or Fedora)

Install the `bridge-utils` software.

```
# yum install bridge-utils
```

`/etc/sysconfig/ifcfg-eth0` should already exist. Save a copy of the original file somewhere safe, then replace the contents with the following:

```
DEVICE=eth0
NAME=eth0
# Use the MAC address identified above.
HWADDR=19:7c:3b:92:ec:ee
# Change to 'no' to disable NetworkManager for this interface.
NM_CONTROLLED=yes
ONBOOT=yes
TYPE=Ethernet
BRIDGE=br0
```

Create `/etc/sysconfig/ifcfg-br0` with these contents:

```
# If unsure what NETMASK, GATEWAY or IPV6_DEFAULTGW should be, check the
# original copy of ifcfg-eth0 or ask your hosting provider.

DEVICE=br0
NAME=br0
# Change to 'no' to disable NetworkManager for this interface.
NM_CONTROLLED=yes
```

```
ONBOOT=yes
TYPE=Bridge
# If you want to turn on Spanning Tree Protocol, ask your hosting
# provider first as it may conflict with their network.
STP=off
# If STP is off, set to 0. If STP is on, set to 2 (or greater).
DELAY=0

IPADDR=203.0.113.166
NETMASK=255.255.255.248
GATEWAY=203.0.113.161

IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6ADDR=2001:db8::1
# The gateway interface (ie, '%br0') must match the name of the bridge.
IPV6_DEFAULTGW=fe80::1%br0
```

If using NetworkManager:

```
# nmcli connection reload
# nmcli connection down eth0 && nmcli connection up eth0
```

If using NetworkManager on Red Hat Enterprise Linux 6:

```
# echo "NM_BOND_BRIDGE_VLAN_ENABLED=yes" >> /etc/sysconfig/network
# service NetworkManager restart
```

If not using NetworkManager:

```
# ifdown eth0 && ifup eth0 && ifup br0
```

Debian

Install the `bridge-utils` software.

```
# apt-get install bridge-utils
```

Add the following text to `/etc/network/interfaces`:

```
# If unsure what 'netmask' or 'gateway' should be, ask your hosting provider.

iface eth0 inet manual

auto br0
iface br0 inet static
```

```
    # Use the MAC address identified above.
    hwaddress ether 19:7c:3b:92:ec:ee
    address 203.0.113.166
    netmask 255.255.255.248
    gateway 203.0.113.161

    bridge_ports eth0
    # If you want to turn on Spanning Tree Protocol, ask your hosting
    # provider first as it may conflict with their network.
    bridge_stp off
    # If STP is off, set to 0. If STP is on, set to 2 (or greater).
    bridge_fd 0

iface br0 inet6 static
    address 2001:db8::1
    netmask 64
    gateway fe80::1
    autoconf 0
```

Bring the bridge up.

```
# ip address flush eth0 scope global && ifup br0
```

Check the bridge

Check that the bridge is up and the Ethernet device is listed as an interface. If the steps above did not work or you have a complicated network environment, you may need to reboot the server.

```
# brctl show br0
bridge name     bridge id               STP enabled     interfaces
br0             8000.197c3b92ecee       no              eth0
```

Configure virtual machines

New VM

```
# virt-install --network bridge=br0 ...
```

Optionally, pass `--network` more than once to create additional virtual Ethernet interfaces for the VM.

```
# virt-install --network bridge=br0 --network network=default ...
```

Existing VM

Open the XML configuration for the VM in a text editor.

```
# virsh edit name-of-vm
```

There should already be an `<interface>` section that configures a virtual Ethernet interface for the VM. Note down the MAC address.

```
<interface type="network">
   <source network="default"/>
   <mac address="52:54:00:4f:47:f2"/>
</interface>
```

To reconfigure the virtual Ethernet device, replace the `<interface>` section with the following contents. Use the MAC address noted above, otherwise the MAC address of the VM will change.

```
<interface type="bridge">
  <source bridge="br0"/>
  <mac address="52:54:00:4f:47:f2"/>
</interface>
```

To add an additional Ethernet interface, append a new `<interface>` section. libvirt generates a random MAC for the new interface if `<mac>` is omitted.

```
<interface type="bridge">
   <source bridge="br0"/>
</interface>
```

Reboot the VM to apply the changes. You may need to amend the VM's network initialization scripts to account for the network interface changes.

Comments

← Previous

Next →

Version 1.0.1 — Last updated on 2015-12-16.