网站搜索

如何通过 libvirt 和 KVM 使用桥接网络

Libvirt 是一个免费的开源软件，它提供 API 来管理虚拟机的各个方面。在 Linux 上，它通常与 KVM 和 Qemu 结合使用。除此之外，libvirt 用于创建和管理虚拟网络。使用 libvirt 时创建的默认网络称为“默认”，并使用 NAT（网络地址转换）和数据包转发将模拟系统与“外部”世界（主机系统和互联网）连接起来。 ）。在本教程中，我们将了解如何使用**桥接网络**创建不同的设置。

在本教程中您将学习：

- 如何创建虚拟桥
- 如何将物理接口添加到网桥
- 如何使网桥配置持久化
- 如何修改固件规则以允许流量到达虚拟机
- 如何创建新的虚拟网络并在虚拟机中使用它

如何通过 libvirt 和 KVM 使用桥接网络

使用的软件要求和约定

“默认”网络

当使用 **libvirt** 且 **libvirtd** 守护进程运行时，会创建一个默认网络。我们可以使用 virsh 实用程序验证该网络是否存在，该实用程序在大多数 Linux 发行版上通常随 libvirt-client 软件包一起提供。要调用该实用程序以显示所有可用的虚拟网络，我们应该包含 `net-list` 子命令：

```
$ sudo virsh net-list --all
```

在上面的示例中，我们使用 `--all` 选项来确保结果中也包含**非活动**网络，该网络通常应与下面显示的网络相对应：

```
Name      State    Autostart   Persistent
----------------------------------------------
 default   active   yes         yes
```

要获取有关网络的详细信息并最终对其进行修改，我们可以使用 `edit` 子命令调用 virsh，并提供网络名称作为参数：

```
$ sudo virsh net-edit default
```

包含 **xml** 网络定义的临时文件将在我们最喜欢的文本编辑器中打开。在这种情况下，结果如下：

```
<network>
  <name>default</name>
  <uuid>168f6909-715c-4333-a34b-f74584d26328</uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:48:3f:0c'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>
```

正如我们所看到的，默认网络基于`virbr0`虚拟网桥的使用，并使用基于NAT的连接来将属于网络一部分的虚拟机连接到外面的世界。我们可以使用 ip 命令验证桥是否存在：

```
$ ip link show type bridge
```

在我们的例子中，上面的命令返回以下输出：

```
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default
qlen 1000
    link/ether 52:54:00:48:3f:0c brd ff:ff:ff:ff:ff:ff
```

要显示属于网桥的接口，我们可以使用 ip 命令并仅查询以 virbr0 网桥为主的接口：

```
$ ip link show master virbr0
```

运行命令的结果是：

```
6: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN mode DEFAULT group
default qlen 1000
    link/ether 52:54:00:48:3f:0c brd ff:ff:ff:ff:ff:ff
```

正如我们所看到的，当前只有一个接口连接到网桥，`virbr0-nic`。 `virbr0-nic` 接口是一个虚拟以太网接口：它是自动创建并添加到网桥的，其目的只是提供稳定的 **MAC** 地址 (52:54 :00:48:3f:0c 在本例中）用于桥接。

当我们创建和启动虚拟机时，其他虚拟接口将添加到桥中。为了本教程，我创建并启动了 Debian (Buster) 虚拟机；如果我们重新启动上面使用的命令来显示桥接从接口，我们可以看到添加了一个新接口，`vnet0`：

```
$ ip link show master virbr0
6: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN mode DEFAULT group
default qlen 1000
    link/ether 52:54:00:48:3f:0c brd ff:ff:ff:ff:ff:ff
7: vnet0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master virbr0 state UNKNOWN mode DEFAULT
group default qlen 1000
    link/ether fe:54:00:e2:fe:7b brd ff:ff:ff:ff:ff:ff
```

任何物理接口都不应添加到 `virbr0` 网桥，因为它使用 **NAT** 来提供连接。

对虚拟机使用桥接网络

默认网络提供了一种非常简单的方式来在创建虚拟机时实现连接：一切都已“准备就绪”并且开箱即用。然而，有时，我们希望实现**完全桥接**连接，其中访客设备连接到主机**LAN**，而不使用**NAT**，我们应创建一个新桥并共享主机物理以太网接口之一。让我们看看如何逐步执行此操作。

创建一座新桥梁

要创建新的网桥，我们仍然可以使用 ip 命令。假设我们想将这座桥命名为 `br0`；我们将运行以下命令：

```
$ sudo ip link add br0 type bridge
```

为了验证桥是否已创建，我们像以前一样执行以下操作：

```
$ sudo ip link show type bridge
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default
qlen 1000
    link/ether 52:54:00:48:3f:0c brd ff:ff:ff:ff:ff:ff
8: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 26:d2:80:7c:55:dd brd ff:ff:ff:ff:ff:ff
```

正如预期的那样，新桥 `br0` 已创建，并且现在包含在上述命令的输出中。现在新桥已创建，我们可以继续向其添加物理接口。

将物理以太网接口添加到网桥

在此步骤中，我们将向网桥添加一个主机物理接口。请注意，在这种情况下，您不能使用主以太网接口，因为一旦将其添加到网桥，您就会失去连接，因为它将失去其 IP 地址。在这种情况下，我们将使用一个附加接口，`enp0s29u1u1`：这是由连接到我的机器的以太网到 USB 适配器提供的接口。

首先我们确保接口状态为UP：

```
$ sudo ip link set enp0s29u1u1 up
```

要将接口添加到桥接，需要运行的命令如下：

```
$ sudo ip link set enp0s29u1u1 master br0
```

要验证接口是否已添加到网桥，请改为：

```
$ sudo ip link show master br0
3: enp0s29u1u1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UP mode DEFAULT
group default qlen 1000
    link/ether 18:a6:f7:0e:06:64 brd ff:ff:ff:ff:ff:ff
```

为网桥分配静态 IP 地址

此时我们可以为网桥分配一个静态IP地址。假设我们要使用`192.168.0.90/24`；我们会运行：

```
$ sudo ip address add dev br0 192.168.0.90/24
```

为了将地址添加到接口中，我们运行：

```
$ ip addr show br0
9: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 26:d2:80:7c:55:dd brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.90/24 scope global br0
       valid_lft forever preferred_lft forever
   [...]
```

使配置持久化

我们的桥接配置已准备就绪，但是，事实上，它无法在机器重新启动后继续存在。为了使我们的配置持久化，我们必须编辑一些配置文件，具体取决于我们使用的发行版。

Debian 及其衍生品

在 Debian 系列发行版上，我们必须确保安装了 `bridge-utils` 软件包：

```
$ sudo apt-get install bridge-utils
```

安装软件包后，我们应该修改 `/etc/network/interfaces` 文件的内容：

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# Specify that the physical interface that should be connected to the bridge
# should be configured manually, to avoid conflicts with NetworkManager
iface enp0s29u1u1 inet manual

# The br0 bridge settings
auto br0
iface br0 inet static
    bridge_ports enp0s29u1u1
        address 192.168.0.90
        broadcast 192.168.0.255
        netmask 255.255.255.0
        gateway 192.168.0.1
```

红帽发行版系列

在 Red Hat 系列发行版（包括 Fedora）上，我们必须操作 `/etc/sysconfig/network-scripts` 目录中的网络脚本。如果我们希望网桥不由 NetworkManager 管理，或者我们使用旧版本的 NetworkManager 无法管理网络交换机，我们需要安装 `network-scripts`包：

```
$ sudo dnf install network-scripts
```

安装软件包后，我们需要创建用于配置br0网桥的文件：`/etc/sysconfig/network-scripts/ifcfg-br0`。在文件中我们放置以下内容：

```
DEVICE=br0
TYPE=Bridge
BOOTPROTO=none
IPADDR=192.168.0.90
GATEWAY=192.168.0.1
NETMASK=255.255.255.0
ONBOOT=yes
DELAY=0
NM_CONTROLLED=0
```

然后，我们修改或创建用于配置将连接到网桥的物理接口的文件，在本例中为 `/etc/sysconfig/network-scripts/ifcfg-enp0s29u1u1`：

```
TYPE=ethernet
BOOTPROTO=none
NAME=enp0s29u1u1
DEVICE=enp0s29u1u1
ONBOOT=yes
BRIDGE=br0
DELAY=0
NM_CONTROLLED=0
```

准备好配置后，我们可以启动 `network` 服务，并在启动时启用它：

```
$ sudo systemctl enable --now network
```

禁用网桥的 netfilter

为了允许所有流量转发到网桥，从而转发到连接到它的虚拟机，我们需要禁用 netfilter。例如，为了使 DNS 解析能够在连接到网桥的来宾计算机上工作，这是必需的。为此，我们可以在 `/etc/sysctl.d` 目录中创建一个扩展名为 `.conf` 的文件，我们将其命名为 `99-netfilter-bridge.conf` 。在里面我们写入以下内容：

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

要加载文件中写入的设置，首先我们确保加载 `br_netfilter` 模块：

```
$ sudo modprobe br_netfilter
```

要在启动时自动加载模块，让我们创建 `/etc/modules-load.d/br_netfilter.conf` 文件：它应该只包含模块本身的名称：

```
br_netfilter
```

加载模块后，要加载我们存储在 99-netfilter-bridge.conf 文件中的设置，我们可以运行：

```
$ sudo sysctl -p /etc/sysctl.d/99-netfilter-bridge.conf
```

创建新的虚拟网络

此时，我们应该定义一个新的“网络”以供虚拟机使用。我们使用我们最喜欢的编辑器打开一个文件，并将以下内容粘贴到其中，然后将其保存为bridged-network.xml：

```
<network>
    <name>bridged-network</name>
    <forward mode="bridge" />
    <bridge name="br0" />
</network>
```

文件准备好后，我们将其位置作为参数传递给 `net-define virsh` 子命令：

```
$ sudo virsh net-define bridged-network.xml
```

要激活新网络并使其自动启动，我们应该运行：

```
$ sudo virsh net-start bridged-network
$ sudo virsh net-autostart bridged-network
```

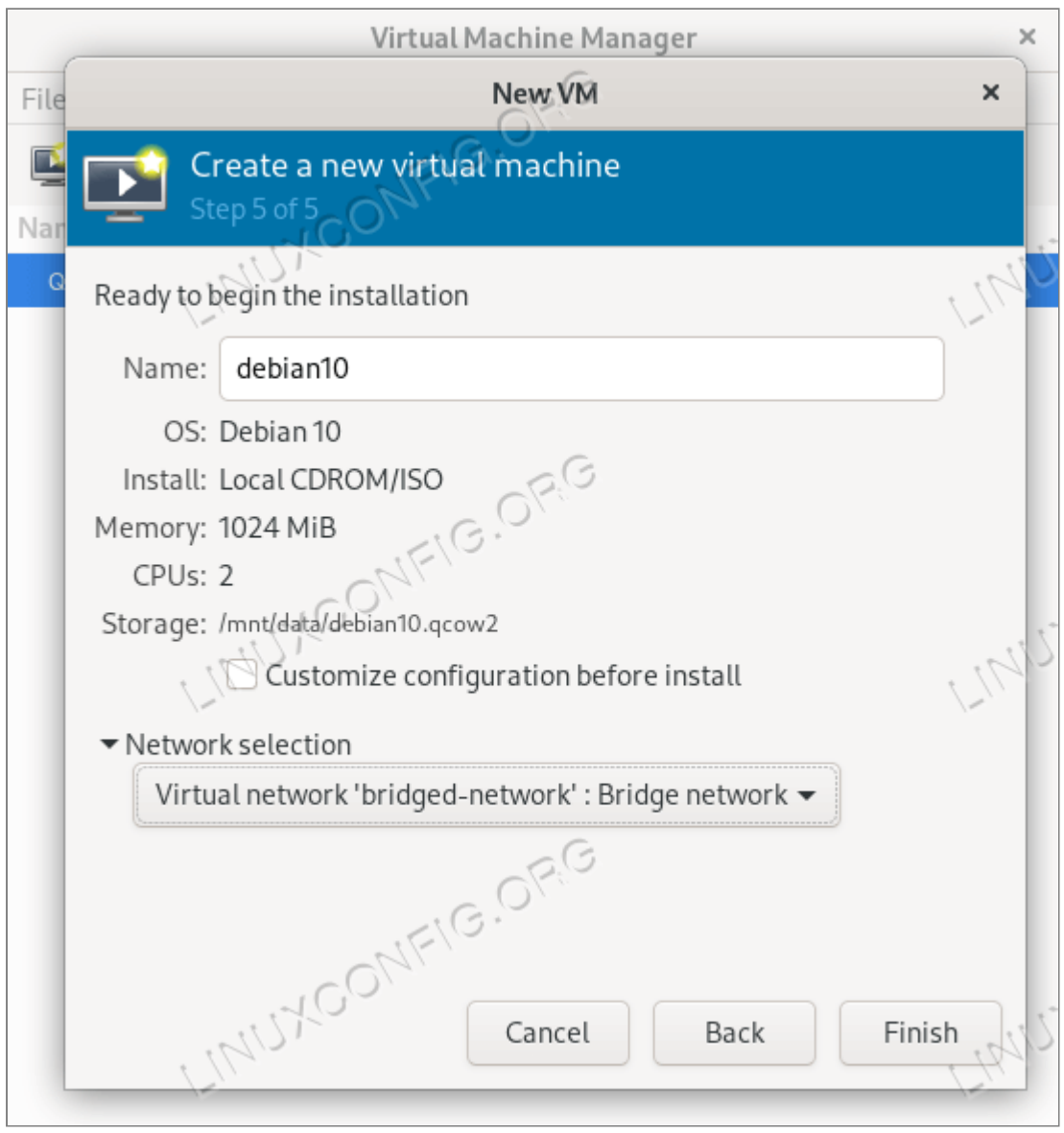
我们可以通过运行 virsh net-list 来验证网络是否已激活 再次命令：

```
$ sudo virsh net-list
Name              State    Autostart   Persistent
----------------------------------------------------
bridged-network   active   yes         yes
default           active   yes         yes
```

现在，我们可以在使用 `--network` 选项时按名称选择网络：

```
$ sudo virt-install \
  --vcpus=1 \
  --memory=1024 \
  --cdrom=debian-10.8.0-amd64-DVD-1.iso \
  --disk size=7 \
  --os-variant=debian10 \
  --network network=bridged-network
```

如果使用**virt-manager**图形界面，我们将能够在创建新虚拟机时选择网络：



结论

在本教程中，我们了解了如何在 Linux 上创建虚拟桥并将物理以太网接口连接到它，以便创建一个新的“网络”以在使用 libvirt 管理的虚拟机中使用。使用后者时，为了方便起见，提供了默认网络：它通过使用 NAT 提供连接。当使用桥接网络作为我们在本教程中配置的网络时，我们将提高性能并使虚拟机成为主机同一子网的一部分。