# KVM虚拟化部署文档V1.0

修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 修订版本 | 描述 | 作者 |
| 2016/06/06 | V1.0 | 创建 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **KVM介绍**

（kvm是kernel virtual module的简写）kvm是红帽公司5.4版本后所推出的最新虚拟化技术产品。它和vpc，virtual  box都属于完全虚拟化产品，而Xen是属于半虚拟化产品。虽说kvm是完全虚拟化，但在关键的硬盘和网卡上支持半虚拟化virtio，在很大程度上提高了性能，kvm是轻量级虚拟化的代表，也需要硬件vt技术的支持，另外从操作界面上看，xen和kvm都是用virt-manager工具，差别不是很大，但不可以在同一台机器上同时安装且运行Xen和KVM；另外kvm在只能运行在红帽企业版64位系统上,同时需要关闭selinux。KVM是指基于 Linux 内核的虚拟机(Kernel-based Virtual Machine)。增加 KVM 到 Linux 内核是Linux 发展的一个重要里程碑，这也是第一个整合到 Linux 主线内核的虚拟化技术。

KVM所使用的方法是通过简单地加载内核模块而将 Linux 内核转换为一个系统管理程序。这个内核模块导出了一个名为 /dev/kvm 的设备，它可以启用内核的客户模式(除了传统的内核模式和用户模式)。有了 /dev/kvm 设备，VM 使自己的地址空间独立于内核或运行着的任何其他 VM 的地址空间。设备树(/dev)中的设备对于所有用户空间进程来说都是通用的。但是每个打开 /dev/kvm 的进程看到的是不同的映射(为了支持 VM 间的隔离)。KVM安装 KVM 之后，您可以在用户空间启动客户操作系统。每个客户操作系统都是主机操作系统(或系统管理程序)的一个单个进程。

1.1 KVM虚拟机的管理工具

准确来说，KVM仅是Linux 内核中的一个模块。管理和创建完的 KVM 虚拟机，需要更多的辅助工具。QEMU-KVM：在 Linux 系统中，首先我们可以用 modprobe 命令加载 KVM 模块，如果用 RPM 安装 KVM 软件包，系统会在启动时自动加载模块。加载了模块后，才能进一步通过其他工具创建虚拟机。但仅有 KVM 模块是远远不够的，因为用户无法直接控制内核模块去做事情，还必须有一个用户空间的工具。关于用户空间的工具，KVM 的开发者选择了已经成型的开源虚拟化软件 QEMU。QEMU 是一个强大的虚拟化软件，它可以虚拟不同的 CPU 构架。比如说在 x86 的 CPU 上虚拟一个Power的CPU，并利用它编译出可运行在 Power 上的程序。KVM使用了QEMU的基于 x86 的部分，并稍加改造，形成可控制 KVM 内核模块的用户空间工具QEMU-KVM。所以 Linux 发行版中分为 内核部分的 KVM 内核模块和 QEMU-KVM 工具。这就是 KVM 和 QEMU 的关系。 Libvirt、virsh、virt-manager：尽管 QEMU-KVM 工具可以创建和管理 KVM 虚拟机，RedHat 为 KVM 开发了更多的辅助工具，比如 libvirt、libguestfs 等。原因是 QEMU 工具效率不高，不易于使用。Libvirt 是一套提供了多种语言接口的 API，为各种虚拟化工具提供一套方便、可靠的编程接口，不仅支持 KVM，而且支持 Xen 等其他虚拟机。使用 libvirt，你只需要通过 libvirt 提供的函数连接到 KVM 或 Xen 宿主机，便可以用同样的命令控制不同的虚拟机了。Libvirt 不仅提供了 API，还自带一套基于文本的管理虚拟机的命令 virsh，你可以通过使用 virsh 命令来使用 libvirt 的全部功能。但最终用户更渴望的是图形用户界面，这就是 virt-manager。他是一套用 python 编写的虚拟机管理图形界面，用户可以通过它直观地操作不同的虚拟机。Virt-manager 就是利用 libvirt 的 API 实现的。

1. **KVM服务器部署**

**2.1 打开CPU虚拟化功能**

**BIOS开启Virtualization Technology**

# egrep ‘vmx|svm’ /proc/cpuinfo

**#查看CPU是否支持虚拟化，VMX表示是Intel处理器，SVM表示是AMD处理器**

**2.2 最小化安装系统**

安装如下虚拟化包，主要有：

@virtualization

@Base

@Core

@additional-devel

@base

@large-systems

@storage-client-iscsi

@system-manage-snmp

@virtualization

@virtualization-client

@virtualization-platform

@virtualization-toolsk

**2.3 安装KVM相关RPM包文件**

#yum install –y kvm python-virtinst libvirt bridge-utils virt-manager qemu-kvm-tools virt-viewer virt-viewer virt-v2v libguestfs-tools

#rpm –qa|grep –E ‘qemu|libvirt|virt’

**#查看装了哪些包**

**#libvirt-client 是libvirt的客户端，最重要的功能之一就是在宿主机关机时可以通知虚拟机也关机，使虚拟机系统正常关机，而不是被强制关机，导致数据丢失**

**#gpxe-rom-qemu 是虚拟机iPXE的启动固件，支持虚拟机从网络启动**

**#libvirt-python 是libvirt为Python提供的API**

**#python-virtinst 是一套python的虚拟机安装工具**

**#qemu-kvm 是KVM在用户空间运行的程序**

**#virt-manager 是基于libvirt的图形化虚拟机管理软件**

**#libvirt 是用于管理虚拟机，它提供了一套虚拟机操作API，可以使用virsh等命令**

**#virt-viewer 用于虚拟机的控制台的console**

**#virt-top 用于类似于top命令，查看虚拟机的资源使用情况**

**#virt-what 在虚拟机内部执行，查看虚拟机运行的虚拟化平台**

**#qemu-img 用于操作虚拟机硬盘镜像的创建、查看和格式化**

**#kvm软件包包含有kvm内核模块，它可以默认Linux内核中提供kvm管理程序**

**#bridge-utils 安装网络支持，设置桥接**

**2.4 宿主机系统配置**

关闭防火墙，关闭selinux

**#由于防火墙和selinux会阻止libvirt为虚拟机创建网络设备**

**2.5 确认KVM模块是否加载**

#lsmod |grep kvm

kvm\_intel 53484 0

kvm 316506 1 kvm\_intel

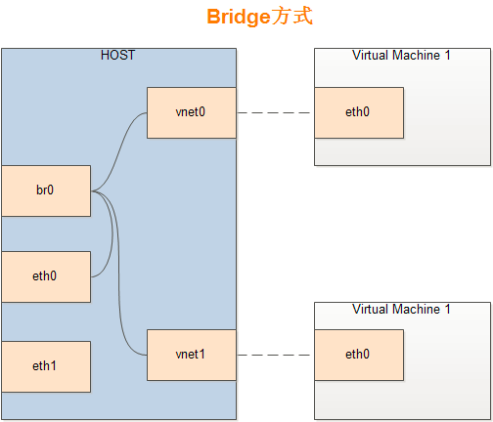
**如果没有正确加载，手动加载模块**

#modprobe kvm-intel

**reboot重新验证一下**

**2.6 Bridge桥接原理**

Bridge方式即虚拟网桥的网络连接方式，是客户机和子网里面的机器能够互相通信。可以使虚拟机成为网络中具有独立IP的主机。桥接网络（也叫物理设备共享）被用作把一个物理设备复制到一台虚拟机。网桥多用作高级设置，特别是主机多个网络接口的情况。

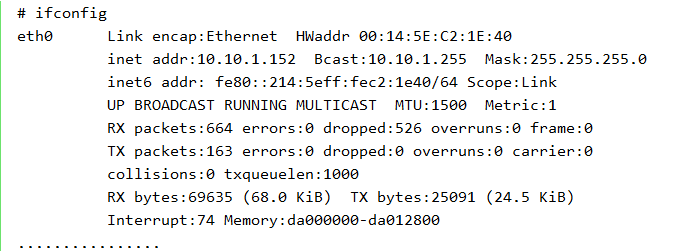


如上图，网桥的基本原理就是创建一个桥接接口br0，在物理网卡和虚拟网络接口之间传递数据。

**2.7 host机器配置桥接网络**

2.7.1查看网络是否工作

确保你想使用bridge网桥的网卡能够为kvm虚拟机模块提供网络连接能力并且网卡是否可以工作.这个网卡应该要能够给guest的kvm虚拟机提供相同的网络支持.接下来例子说明了网卡的配置,并且可以被外部网络访问.这个例子中,使用**eth0**网卡.



2.7.2备份network-scripts网络脚本文件

备份网络配置到不同的目录下,将来可以用来恢复网络配置,. 根据下面的命令将网络脚本ifcfg-eth0 网卡到 /tmp目录:

|  |
| --- |
| # cp /etc/sysconfig/network-scripts/ifcfg-eth0 /tmp |

**注意:不要把文件copy到/etc/sysconfig/network-scripts/目录或者其子目录**

2.7.3为桥接新建配置文件

为Linux的bridge创建一个新的配置文件为/etc/sysconfig/network-scripts/ifcfg-br0,这里br0是bridge网桥的名字,同eth0类似.使用以下命令

#cp ifcfg-eth0 ifcfg-br0

2.7.4编辑bridge网桥配置文件

编辑配置文件,让数据包通过bridge(网桥)进行上网.你的网络配置很可能是通过静态ip(BOOTPROTO=static)上网或者是从DHCP服务器获得ip(BOOTPROTO=dhcp)上网的.

如果你的网络是通过静态ip地址来配置的,那么你的配置很可能是和下面一样的:

DEVICE=eth0

BOOTPROTO=static

HWADDR=00:14:5E:C2:1E:40

IPADDR=10.10.1.152

NETMASK=255.255.255.0

ONBOOT=yes

下面的表格显示了eth0和br0的静态ip配置内容,可以根据下面进行编辑

|  |  |
| --- | --- |
| static ip配置 | |
| **/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 |

下面表格显示的eth0和br0的DHCP配置情况.

|  |  |
| --- | --- |
| DHCP配置 | |
| **/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=dhcp ONBOOT=yes |

2.7.5 重启网络服务

重启网络以验证网络配置是否工作，如果网络配置不正确,网络连接将会断开,你的电脑将失去访问能力.这样的话,检查以下配置文件,然后使用以下的命令重启以下网络服务:

# service network restart

2.7.6 系统内核参数配置

禁用网络过滤器：

向文件/etc/sysctl.conf添加以下代码:

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

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

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

重新加载kernel参数：

# sysctl -p

验证配置是否成功

通过运行ifconfig来验证以下配置是否成功.如果出现两个项,即网桥的br0和现在在工作的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

注意:你可能发现eth0的网络信息里面没有ipv4的地址了.或者你的GUI界面的网络显示你是断网的. 这不用担心.因为eth0的网络能力已经通过br0进行工作.你的host网络还是能正常工作的。

也可以使用以下的命令来查看网桥是否工作。

# brctl show

bridge name     bridge id               STP enabled     interfaces

virbr0          8000.000000000000       yes

br0             8000.000e0cb30550       no              eth0

如果是类似的输出,那么你的Linux的Bridge网桥就配置的差不多了。

2.7.7 开启相应的服务

#/etc/init.d/messagebus start

**#开启Linux ICP服务，这是 Linux 的 IPC（Interprocess Communication，进程间通讯）服务。确切地说，它与 DBUS 交互，是重要的系统服务。强烈建议开启它。**

#/etc/init.d/libvirtd restart

**#启动virbr0**

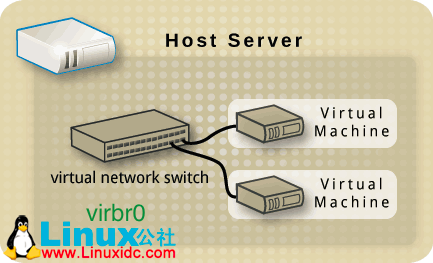
2.7.8 可关闭virbr0

安装完KVM后会发现网络接口中多了一个叫做virbr0的虚拟网络接口。

#ifconfig

...  
virbr0    Link encap:Ethernet  HWaddr d2:91:97:b8:3d:fc   
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0  
          UP BROADCAST MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
...

这是由于安装和启用了 libvirt 服务后生成的，libvirt 在服务器（host）上生成一个 virtual network switch (virbr0)，host 上所有的虚拟机（guests）通过这个 virbr0 连起来。默认情况下 virbr0 使用的是 NAT 模式（采用 IP Masquerade），所以这种情况下 guest 通过 host 才能访问外部。



大多数时候我们虚拟机使用的是 bridge（网桥）直接连到局域网里，所以这个 virbr0 不是必须的（注：不要搞混淆了，bridge 和这里的 virbr0 bridge 是互不相干的）。如何关掉这个 virbr0 呢？先 net-destroy 然后 net-undefine，最后别忘了重启 libvirtd 让设置生效：

# virsh net-list  
Name                State      Autostart  
-----------------------------------------  
default              active    yes

# virsh net-destroy default  
Network default destroyed

# virsh net-undefine default  
Network default has been undefined  
# service libvirtd restart  
Stopping libvirtd daemon:                                  [  OK  ]  
Starting libvirtd daemon:                                  [  OK  ]

1. **virt-install 命令部署虚拟机**

virt-install 是一个在命令行创建KVM虚拟机的工具，使用virt-install 配合一些配置参数，最终可以生成一个完整的.XML虚拟机配置文件。

**3.1 安装并配置VNC服务**

#yum install –y tigervnc

#yum install -y tigervnc-server

#vim /etc/sysconfig/vncserver

VNCSERVERS=”1:root”

**#配置宿主机VNC显示器为1，端口是5901**

#vncpasswd

**#设置VNC密码**

**3.2 新建虚拟机磁盘**

KVM虚拟机的磁盘镜像从存储方式上看，可以分为两种，第一种方式为存储于文件系统上，第二种方式为直接使用裸设备。裸设备的使用方式可以是直接使用裸盘，也可以使用LVM的方式。经常使用的是RAW和qcow2。

3.2.1 创建RAW格式镜像

#qemu-img create test 50G

3.2.2 创建qcow2格式镜像

#qemu-img create test.qcow2 -f qcow2 50G

3.2.3 镜像信息查看

[root@localhost network-scripts]# qemu-img info /data/kvm/rhel01.img

image: /data/kvm/rhel01.img

file format: qcow2

virtual size: 20G (21474836480 bytes)

disk size: 5.5G

cluster\_size: 65536

**3.3 虚拟机创建**

#virt-install --name=rhel01 --ram=1024 --vcpu=2 --os-type=linux --hvm --cdrom=/tmp/rhel-server-6.5-x86\_64-dvd.iso -disk path=/data/kvm/rhel01.qcow2, format=qcow2,size=20,bus=virtio --bridge=br0 --vnc --vnclisten=0.0.0.0 --vncport=5910 --accelerate

**#--ram 配置虚拟机内存，单位是MB**

**#--name 配置虚拟机的名称**

**#--vcpu 配置CPU个数**

**#--os-type 指定操作系统类型，如linux 、windows**

**#--hvm 配置使用全虚拟化**

**#--cdrom 使用cdrom安装系统，指定ISO位置**

**#--disk path 设置虚拟机磁盘文件的路径**

**#format 磁盘文件的格式**

**#size 磁盘文件的大小**

**#bus=virtio 指定磁盘总线**

**#--bridge 配置桥接的网卡**

**#--vnc 打开VNC支持**

**#--vcport 指定VNC端口**

**#--vnclisten 指定监听的地址**

**#--acclerate 当安装QEMU客户机时，如果支持可用KVM或KQEMU内核加速能力。除非一个客户机操作系统不兼容加速，这个选项是推荐最好加上。如果KVM和KQEMU都支持，KVM加速器优先使用。**

注意安装windows2012系统时，需要额外安装virtio 驱动，可以参考如下

#virt-install --name=windows01 --ram=1024 --vcpu=2 --os-type=windows --hvm --cdrom=/tmp/cn\_windows\_server\_2012\_x64\_dvd\_915588.iso --disk path=/data/kvm/windows01.qcow2, format=qcow2,size=50,bus=virtio --bridge=br0 --vnc --vnclisten=0.0.0.0 --vncport=5920 --accelerate --os-variant=win2k8 --disk path=/tmp/virtio-win-0.1.117\_amd64.vfd,device=floppy



1. **管理虚拟机**

**4.1 查看虚拟机列表和状态**

[root@localhost tmp]# virsh list --all

Id Name State

----------------------------------------------------

- rhel01 shut off

- rhel02 shut off

**4.2 启动虚拟机**

[root@localhost tmp]# virsh start rhel01

Domain rhel01 started

**4.3 关闭虚拟机**

正常关闭虚拟机

#virsh shutdown rhel01

强制关闭虚拟机

#virsh destroy rhel01

**4.4 删除虚拟机**

#virsh undefine rhel01

#rm –rf /data/kvm/rhel01.img

**然后再删除磁盘文件**

**4.5 编辑虚拟机**

#virsh edit rhel01

**注意直接vi修改XML文件并不生效**

**4.6 查看虚拟机启动的VNC端口**

[root@localhost tmp]# virsh vncdisplay rhel01

:101

**显示为101，这个不是我们说的传统意义上的端口，实际端口应该是5900+101=6001**

**4.7 查看日志信息**

4.7.1 查看linvirtd的运行日志

在运行libvirtd的时候,我们需要获得lbivirtd的运行信息.所以我们需要找到他的日志文件.一般情况下,它是在/var/log/libvirt/libvirtd.log路径下。可能在这个目录下没有发现这个的日志文件.那么就要配置一些libvitd的参数了。

编辑文件/etc/libvirt/libvirtd.conf 将日志级别设置为 1（调试），#指定日志输出文件名称

log\_level = 1

log\_outputs="1:file:/var/log/libvirt/libvirtd.log"

注意 :libvirtd. 日志文件可能会飞速增长。 用户应配置logrotate ，否则您的 /var 文件系统最后会装满内容。以上的日志的输出级别为debug级别.这个级别的日志是最多的,一般只在开发以及测试的时候使用.而在生产运行环境中,日志的级别为info,warn,error,fatal. 以下前面的数字为他们的level。

* 1 debug
* 2 info
* 3 warn
* 4 error
* 5 fatal

重启libvirtd 。虚拟机Guest操作系统正在运行时可重启 libvirtd。

# /etc/init.d/libvirtd restart

如果在目录下还是没发现日志文件,那么你可能需要使用一下命令来运行libvirtd。

#libvirtd --daemon --listen --config /etc/libvirt/libvirtd.conf

4.7.2 查看virsh管理qemu虚拟机的信息

在使用virsh管理qemu的虚拟机时候,也是有日志的输出的，具体的log在目录/var/log/livirt/qemu/ 下面,具体的log文件名与虚拟机名字关联。