### KVM虚拟化技术

**前言：**

1.蓝色字体表示命令行命令，正式执行时不要复制前面的#号，#号只是提示应该使用root权限操作

2.绿色字体表示注释，有时注释太多就不用绿色表示了

3.注意：本文档的所有操作请先在测环境进行实践，请不要直接在真实的服务器中操作！

**版权声明**：

本文档以开源的形式发布，所有条款如下：

 （1）无担保：作者不保证文档内容的准确无误，亦不承担由于使用此文档所导致的任何后果

 （2）自由使用：任何人可以出于任何目的而自由地 阅读/链接/打印/转载/引用/分发/再创作 此文档，无需任何附加条件

若您 阅读/链接/打印/转载/引用/分发/再创作 本文档，则说明接受以上2个条款。

作者：李茂福

更新日期：2025-08-01

**★kvm与qemu虚拟化概念**

**半虚拟化**：虚拟机操作系统内核需要经过修改，与宿主机系统内核共享底层硬件。优点是半虚拟化的虚拟机系统内核能直接管理底层硬件，性能较好，缺点为虚拟机系统内核需要修改（使操作系统知道自己运行在虚拟环境下），安装时不太便捷，兼容性较差。

**全虚拟化**：（Full Virtuallization）虚拟机与底层硬件之间有一个虚拟化逻辑层Hypervisor，由Hypervisor来完全模拟底层硬件，上层虚拟机完全感知不到运行在虚拟硬件上。优点是虚拟机系统内核不需要修改，便于部署，兼容性高。缺点是虚拟机系统内核要通过宿主机的Hypervisor来管理底层硬件，有转换性能开销。

**KVM**（Kernel-based Virtual Machine）是基于硬件虚拟化扩展（Intel VT / AMD-V）的全虚拟化解决方案，KVM也是一个开源的项目，核心组件kvm内核模块包含在linux内核中（2.6.20版本开始），用户空间组件包含在QEMU中（QEMU 1.3版本开始）

KVM虚拟机表现为一个正常的进程，像其他进程一样可以被kill掉。KVM的api是通过/dev/kvm设备访问，/dev/kvm是一个标准字符设备，可使用open,close,ioctl接口操作，在KVM的实现中，所有对KVM的操作都是通过ioctl接口操作的。

**Libvirt**是一个函数库，包含实现linux虚拟化功能的api，提供了管理虚拟机的通用接口。包含三部分：Libvirt API，Libvirtd进程，virsh工具集。

virsh工具集是给用户层使用的，Libvirt API是给程序调用的，Libvirtd是主要进程，实现各抽象设备（最终底层还是调的QEMU或XEN等虚拟化技术接口）

**以上各组件关系图如下：**



**★qemu-kvm创建虚拟机**

宿主机要求开启Intel VT-x 或AMD-V虚拟化技术



# cat /proc/cpuinfo | grep -e vmx -e svm #查看cpu是否支持虚拟化



# lsmod | grep kvm #查看已加载的内核模块



**#使用命令安装以下KVM核心组件**

# yum install qemu-kvm -y #安装QEMU（用户空间），安装GNOME桌面后默认已安装此组件

# ln -s /usr/libexec/qemu-kvm /usr/bin/qemu-kvm #创建链接，才可使用qemu-kvm命令

# qemu-img create -f qcow2 /data/vm\_01.qcow2 10G #创建虚拟磁盘

# qemu-img info /data/vm\_01.qcow2

**#启动虚拟机**

|  |  |
| --- | --- |
| qemu-kvm -name vm\_01 \ | 指定虚拟机名称 |
| -m 1024 -smp 1 \ | 指定内存1024MB，cpu 1核 |
| -hda /data/vm\_01.qcow2 \ | 指定虚拟机硬盘文件 |
| -cdrom /data/CentOS-7-x86\_64-DVD-2009.iso \ | 指定挂载的光盘文件 |
| -enable-kvm \ | 使用KVM进行加速 |
| -boot order=cd,once=d \ | 启动顺序为cd，一次性启动dc表示hda硬盘，d表示cdrom光盘 |
| -vnc 127.0.0.1:1 | 监听5900+1号端口，即5901可指定0.0.0.0:port，监听所有ip |

#默认vnc监听本地::1:5900端口，只能从本地连接



# yum install tigervnc -y #安装vnc客户端

# vncviewer #打开客户端，输入 :5900 连接虚拟机console界面





连接到虚拟机console界面后，就可操作虚拟机了，如安装系统，安装完成后，如果需要关闭虚拟机，可在虚拟机里init 0，也可在宿主机上按下Ctrl + C即可强行关闭

需要重启启动虚拟机，则再次执行启动命令即可（这次可不指定光盘-cdrom）

# qemu-kvm -name vm\_01 \

-m 1024 -smp 1 \

-hda /data/vm\_01.qcow2 \

-enable-kvm \

-boot c #从硬盘启动

**★虚拟机网络配置**

默认创建的虚拟机没有连通宿主机的网络，可以自行创建网桥，再将虚拟机的网卡桥接到宿主机的目标网桥上即可

# yum install bridge-utils -y #安装bridge工具

# qemu-kvm -name vm\_01 \

-m 1024 -smp 1 \

-hda /data/vm\_01.qcow2 \

-enable-kvm \

-net nic,model=e1000,macaddr=86:4c:5d:00:01:02 \

-net tap,ifname=tap1,script=/data/qemu-ifup.sh,downscript=no \

-boot c

#macaddr指定的是虚拟机里虚拟网卡的mac地址，不是宿主机上的tap的mac地址

# qemu-kvm -net nic,model=? #查看系统支持的网卡设备类型，未指定model时默认设置为e1000

qemu: Supported NIC models: ne2k\_pci,i82551,i82557b,i82559er,rtl8139,e1000,pcnet,virtio

#script默认找/etc/qemu-ifup和/etc/qemu-ifdown

# cat /data/qemu-ifup.sh

#!/usr/bin/bash

bridge=virbr0 #要桥接的网桥

if [ -n "$1" ]; then

 ip link set $1 up

 sleep 1

 brctl addif ${bridge} $1

 exit 0

else

 echo "Error: no interface specified"

 exit 1

fi

# chmod +x /data/qemu-ifup.sh #需要添加可执行权限

**★qemu-img磁盘镜像操作**

# qemu-img create -f qcow2 /data/vm\_01.qcow2 10G #创建虚拟磁盘

# qemu-img info /data/vm\_01.qcow2 #查看磁盘信息



#磁盘镜像格式转换，-f指定原镜像文件格式，-O指定转换后的格式，生成新的文件，原磁盘镜像文件还在（以下示例中，原文件为vm\_01.qcow2，转换后新生成的文件为vm\_01.vmdk）

# qemu-img convert -f qcow2 -O vmdk vm\_01.qcow2 vm\_01.vmdk

支持的格式有：

|  |  |
| --- | --- |
| 格式 | 虚拟化平台 |
| qcow2 | KVM,XEN |
| raw | raw |
| vdi | VirtualBox |
| vhd | Hyper-V |
| vmdk | VMware |

# qemu-img check vm\_01.qcow2 #对磁盘镜像文件进行一致性检查，仅支持qcow2,vdi



# qemu-img resize vm\_01.qcow2 +5GB #调整磁盘大小，+-表示在原有大小上增加或减少

# qemu-img resize vm\_01.qcow2 18GB #直接指定调整后的大小

# qemu-img snapshot -l vm\_01.qcow2 #列出镜像文件中的所有快照

# qemu-img snapshot -c snap\_0314 vm\_01.qcow2 #创建快照

# qemu-img snapshot -a snap\_0314 vm\_01.qcow2 #使用快照（恢复到快照）

# qemu-img snapshot -d snap\_0314 vm\_01.qcow2 #删除快照

**★libvirt安装及virsh命令**

在Libvirt工具集中包含了Libvirt API，Libvirtd进程及virsh工具

# yum install qemu-kvm -y #安装qemu用户空间组件

# yum install libvirt -y #安装libvirt

# systemctl enable libvirtd

# systemctl start libvirtd

virsh命令支持直接带参数使用，以及支持交互式登录使用

# virsh -c qemu:///system #交互式登录到本地的qemu:///system进程



#默认libvirtd只监听本地的Unix Domain Socket，不监听基于网络的tcp/ssh socket，若开启了基于网络的socket，则virsh命令可通过网络连接到本服务器的libvritd守护进程进行操作

virsh命令不带-c参数时，默认连接到本地的qemu:///system

virsh命令最终也是调用qemu-kvm命令来完成实际的工作

**★virt-install创建虚拟机**

# qemu-img create -f qcow2 /data/vm\_01.qcow2 10G #创建虚拟磁盘

# qemu-img info /data/vm\_01.img

**#创建br0网桥**

# yum install bridge-utils -y #安装bridge工具

# nmcli conn add type bridge con-name br0 ifname br0 #创建bridge

# nmcli conn del ens33 #删除成员口的配置

# nmcli conn add type bridge-slave con-name ens33 ifname ens33 master br0

#添加成员口到bridge里，master后为 ifname

# nmcli conn up ens33 #先up成员口，再up网桥

# nmcli conn up br0

# nmcli conn modify br0 bridge.stp no

|  |  |
| --- | --- |
| virt-install --name=vm\_01 \ | 创建虚拟机 |
| --ram 1024 --vcpus=1 \ | 指定内存1024MB，cpu 1核 |
| --disk path=/root/vm\_01.img,size=10 \ |  |
| --network bridge=br0 \ |  |
| --accelerate \ |  |
| --cdrom /root/CentOS-7-x86\_64-Minimal-2009.iso \ |  |
| --graphics vnc,port=5921 \ | 端口号必须大于5900，真实监听的端口 |
| --check disk\_size=off \ | #不检查磁盘大小是否超出宿主机剩余磁盘大小 |
| --wait 0 | 不等待客户系统安装结束，单位分钟，默认是等待-1（永久） |

# virt-clone --connect=qemu:#/system -o vm\_01 -n vm\_02 -f /root/vm\_02.img

#克隆vm\_01

**★virsh命令行操作虚拟机**

# yum install libvirt-daemon libvirt-client libvirt-daemon-driver-qemu -y #安装libvirt

# yum install virt-manager -y #图形管理工具，可选

# systemctl enable libvirtd

# systemctl start libvirtd

# virt-manager #打开图形管理界面，前提是安装了virt-manager图形管理工具，没有安装则使用命令操作

# virsh list --all #查看kvm虚拟机状态



# virsh start vm\_test1 #启动虚拟机

# virt-viewer vm\_test1 #如果有图形界面的话，可以进入虚拟机的界面

# virsh shutdown vm\_test1 #关闭虚拟机

# virsh reboot vm\_test1 #重启虚拟机

# virsh suspend vm\_test1 #暂停虚拟机

# virsh resume vm\_test1 #恢复虚拟机

# virsh autostart vm\_test1 #配置开机自启动虚拟机

# virsh undefine vm\_01 #删除虚拟机，仅删除vm\_01的配置文件，并不删除虚拟磁盘文件

**#虚拟机配置文件默认路径**

/etc/libvirt/qemu/

# virsh dumpxml vm\_01 > /etc/libvirt/qemu/vm\_01.xml #导出KVM虚拟机配置文件

# virsh create /etc/libvirt/qemu/vm\_01.xml #通过配置文件启动虚拟机

#通过导出备份的配置文件恢复原KVM虚拟机的定义

# virsh define /etc/libvirt/qemu/vm\_01.xml

# virsh edit vm\_01 #编辑KVM虚拟机配置文件；virsh edit将调用vi命令编辑/etc/libvirt/qemu/vm\_01.xml配置文件，也可以直接通过vi命令进行编辑，编辑后要重启虚拟机，（重启操作为：先关机后开机）

# virsh console vm\_01 #控制台登录虚拟机

未完待续