KVM虚拟化应用

**一：虚拟化之KVM(上)**

2017-07-28 11:47:22

标签：[kvm](http://blog.51cto.com/tag-kvm.html)

**一：虚拟化之KVM(上)**

阅读(750)

一：传统的物理机部署方案、虚拟化与云计算定义：

服务器选型及采购—IDC选择及上架-系统选择及安装–应用规划及部署–域名选择及注册–DNS映射–外网访问：

工信部备案-公安部备案–ICP备案(电子商务网站–>ICP证经营性ICP备案)，游戏公司文化部(文网文)备案等，在备案没有成功之前网站是不能上线访问的，论坛BBS有BBS公告备案是备案的前置审批，没有的话备案不成功，还要在公安局系统备案，另外域名接入到一个地方比如机房还要做接入备案，备案在个人名下的备案不能直接转公司，个人注销后网站属于未备案状态随时可能会被封，域名备案一般可以找代理，因为过程比较复杂：

传统数据中心面临的问题：

服务器资源利用率低下，CPU、内存等不能共享

资源分配不合理

初始化成本高

自动化能力差

集群环境需要大量的服务器主机

1.1:云计算的定义

云计算是一种交付模式

云计算必须通过网络使用

支持弹性横向扩展、按需付费和快速扩展

无需关心基础设置，服务器硬件、存储、机房安全、硬件监控、底层数据安全等无需考虑

1.2：云计算的概念与分类：

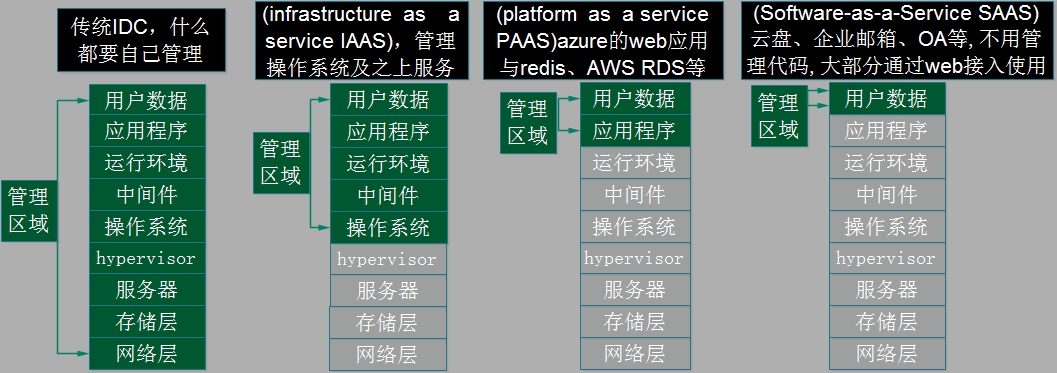
云计算是概念最早是由Google 前首席执行官埃里克施密特（Eric Schmidt）在2006 年8 月9 日的搜索引擎大会上首次提出的一种构想，而“云计算”就是这种构想的代名词，云计算以虚拟化为基础，以网络为中心，为用户提供安全、快速、便捷的数据存储和网络计算服务，包括所需要的硬件、平台、软件及服务等资源，而提供资源的网络就被称为“云”。

公有云：向aws、阿里云以及azure都属于公有云，每个人都可以付费使用，不需要自己关心底层硬件，但是数据安全需要考利。

私有云：在自己公司内部或IDC自建openstack等环境

混合云：临时需求使用公有云，但是主要业务放在自己的私有云，即自己的私有云的部分业务和公有云有交接，这部分称为混合云

1.3:云计算的分层：



1.4：虚拟化的分类：

服务器虚拟化-虚拟服务器

桌面虚拟化-瘦客户机连接win7 实现普通办公

存储虚拟化-SAN(基于磁盘)/NAS(NFS/Samba)/GlusterFS

应用虚拟化-将办公软件虚拟化，最典型的就是office

网络虚拟化-SDN

1.5：KVM

kmv是一种技术，云计算是一种模式，虚拟化是利用相应的技术方法在一台物理机器上将其按照不同的需求划分成多个相同或者不同的虚拟操作系统，并且各个虚拟系统可以同时运行，互不干扰，其中任何一个虚拟系统的损坏都不影响同一物理服务器上的其他虚拟系统。

1.6：虚拟化的前生与今世：

1964年，IBM推出了专为 System/360 Mainframe 量身订造的操作系统 CP-40，首次实现了虚拟内存和虚拟机。1967 年，第一个管理程序(hypervisor)诞生，5年之后，IBM 发布用于创建灵活大型主机的虚拟机(VM)技术，该技术可根据动态的需求快速而有效地使用各种资源。从此，虚拟化这一词汇正式被引入了IT的现实世界。20世纪 90 年代 Windows 的广泛使用以及 Linux 作为服务器系统的出现奠定了 x86 服务器的行业标准地位。1998年VMware公司在美国成立， 1999年VMware发布了它的第一款产品VMware Workstation、 2001年发布VMware GSX Server和VMware ESX Server宣布进入服务器虚拟化市场， 2003年VMware推出了VMware Virtual Center， 2004年推出了64位支持版本，同年被EMC收购，2013年收入52.1亿美元。 2007年8月21日，思杰宣布5亿美元收购XenSource公司，并推出服务器虚拟化XenServer、桌面虚拟化XenDesktop和应用虚拟化XenApp，2013年收入29亿美元。2008年3月13日微软在北京发布Windows Server 2008，内置虚拟化技术hyper-v。2008年9月，红帽以1.07亿美元的价格收购KVM的母公司Qumranet，并推出企业级虚拟化解决方案RHEV，目前最新版本3.3，2013年收入超过13亿美元

1.7：目前全球主要的虚拟化厂商：



1.8：各厂商对比：

1、Vmware是目前市场上服务器虚拟化占有率最高的厂商，原因是其起步早、并且优先于其他厂商推出X86虚拟化产品、产品功能相对完善从而有限占据市场，但是其代码不公开，并且严重依赖Windows服务器系统，也就是厂商绑定，一旦用了Vmware就得购买微软的系统，购买了微软的系统就处处受微软的限制，另外Vmware的授权费用较高（每颗CPU授权78000元，两颗起售）。2、微软针对WindowsServer2008标准版、企业版与数据中心版，各推出内建Hyper-V的虚拟化环境，需要购买Win Server 2008、Hyper-Vmanager与CPU授权等，其总体花费较高，而且稳定性不高，目前使用市场较小。3、思杰看到Vmwae在虚拟化领域的成果之后在2007年收购XenSource公司（Xen虚拟化），随后推出针对服务器虚拟化的XenServer、桌面虚拟化XenDesktop以及针对应用虚拟化的XenApp，但是思杰的虚拟化方案成本比较高，配置和管理非常复杂，Xen Desktop也严重依赖微软的WinServer系统，也就是要先购买微软的服务器授权，然后在WinServer上安装Xendesktop，因此产生了较高的费用而且稳定性也不是很好。4、 红帽的虚拟化基于KVM，而KVM（kernel-based Virtual Machne )是红帽在2008年收购而来，其代码完全开源，工作原理是将模块kvm.ko加载到kernel里运行，从而获得最高效的虚拟能力，红帽成立于1995年是全球最大的Linux开源技术厂家, KVM 是基于虚拟化扩展（Intel VT 或AMD-V）的X86 硬件，是Linux 完全原生的全虚拟化解决方案，主要是通过准虚拟网络驱动程序的形式用于 Linux 和 Windows 客户机系统的，KVM支持广泛的客户机操作系统，比如 Linux、BSD、Solaris、Windows XP、windows 7、windows 8、ubuntu等大多数操作系统，目前KVM已经是各大linux发行版本所支持的虚拟化产品。

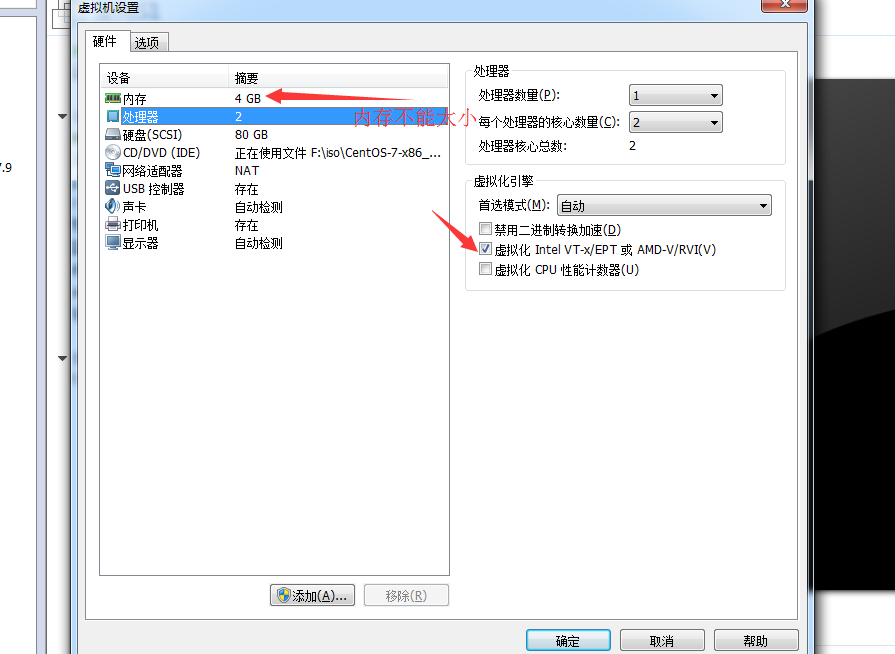
二：虚拟化技术之KVM：

Kernel-based Virtual Machine的简称，是一个开源的系统虚拟化模块，自Linux 2.6.20之后集成在Linux的各个主要发行版本中。它使用Linux自身的调度器进行管理，所以相对于Xen，其核心源码很少。KVM目前已成为学术界的主流VMM之一。  
KVM的虚拟化需要硬件支持（如Intel VT技术或者AMD V技术)。是基于硬件的完全虚拟化。而Xen早期则是基于软件模拟的Para-Virtualization，新版本则是基于硬件支持的完全虚拟化。但Xen本身有自己的进程调度器，存储管理模块等，所以代码较为庞大。广为流传的商业系统虚拟化软件VMware ESX系列是基于软件模拟的Full-Virtualization。

IBM,文档：http://www.ibm.com/developerworks/cn/linux/l-using-kvm/

lxd容器虚拟化: https://linuxcontainers.org/lxd/introduction/

qume是软件虚拟化，即使用软件完全模拟出一个具备CPU/内存/和硬盘的等设备的服务器，KVM是全虚拟化，Xen是半虚拟化，KVM在部分设备也支持半虚拟化

2.1:环境准备：  


2:2.1：确认CPU已经支持虚拟化，intel的CPU虚拟化技术交vmx，AMD的CPU交svm：

[root@localhost ~]# grep -E "vmx|svm" /proc/cpuinfoflags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts mmx fxsr sse sse2 ss ht syscall nx rdtscp lm constant\_tsc arch\_perfmon pebs bts nopl xtopology tsc\_reliable nonstop\_tsc aperfmperf pni pclmulqdq vmx ssse3 cx16 pcid sse4\_1 sse4\_2 x2apic popcnt tsc\_deadline\_timer aes xsave avx f16c rdrand hypervisor lahf\_lm ida arat epb pln pts dtherm tpr\_shadow vnmi ept vpid fsgsbase tsc\_adjust smep

2.2.2：安装kvm平台及工具包：

[root@localhost ~]# yum install qemu-kvm qemu-kvm-tools libvirt virt-manager virt-install

2.2.3：启动libvirtsd并设置为开机启动，librirtd会创建一个桥接的网卡virbr0而且IP地址是192168.122.1：

[root@localhost ~]# systemctl start libvirtd[root@localhost ~]# systemctl enable libvirtd[root@localhost ~]# ifconfig virbr0virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255

ether 52:54:00:cb:ca:b0 txqueuelen 0 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

2.2.4：为虚拟机分配IP地址的服务：

[root@localhost ~]# ps -aux | grep dnsnobody 2877 0.0 0.0 15552 888 ? S 13:06 0:00 /sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/libexec/libvirt\_leaseshelper

root 2878 0.0 0.0 15524 180 ? S 13:06 0:00 /sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/libexec/libvirt\_leaseshelper

root 4546 0.0 0.0 112664 980 pts/0 S+ 13:29 0:00 grep --color=auto dns

2.2.5：创建磁盘：

[root@llocalhost ~]# qemu-img create -f raw /opt/CentOS-7-x86\_64.raw 10G #创建一个格式为raw大小为10G的裸磁盘Formatting '/opt/CentOS-7-x86\_64.raw', fmt=raw size=10737418240

2.2.6：开始安装：

[root@linux-node1 ~]# virt-install --virt-type kvm --name CentOS-7-x86\_64 --ram 1024 --cdrom=/opt/CentOS-7-x86\_64-DVD-1511.iso --disk path=/opt/CentOS-7-x86\_64.raw --network network=default --graphics vnc,listen=0.0.0.0 --noautoconsole开始安装......创建域...... | 0 B 00:00:01 域安装仍在进行。您可以重新连接到控制台以便完成安装进程。#virsh-install命令帮助：[root@linux-node1 ~]# virt-install --helpusage: virt-install --name NAME --ram RAM STORAGE INSTALL [options]使用指定安装介质新建虚拟机。optional arguments: -h, --help show this help message and exit

--version show program's version number and exit

--connect URI 使用 libvirt URI 连接到 hypervisor

通用选项:

-n NAME, --name NAME 客户端事件名称

--memory MEMORY 配置虚拟机内存分配。例如：

--memory 1024 (in MiB)

--memory 512,maxmemory=1024

--vcpus VCPUS 为虚拟机配置的 vcpus 数。例如：

--vcpus 5

--vcpus 5,maxcpus=10,cpuset=1-4,6,8

--vcpus sockets=2,cores=4,threads=2,

--cpu CPU CPU 型号及功能。例如：

--cpu coreduo,+x2apic

--cpu host

--metadata METADATA 配置虚拟机元数据。例如：

--metadata name=foo,title="My pretty title",uuid=...

--metadata description="My nice long description"

安装方法选项:

--cdrom CDROM 光驱安装介质

-l LOCATION, --location LOCATION

安装源(例如：nfs:host:/path、http://host/path

ftp://host/path)

--pxe 使用 PXE 协议从网络引导

--import 在磁盘映像中构建虚拟机

--livecd 将光驱介质视为 Live CD

-x EXTRA\_ARGS, --extra-args EXTRA\_ARGS

附加到使用 --location 引导的内核的参数

--initrd-inject INITRD\_INJECT

使用 --location 为 initrd 的 root

添加给定文件

--os-variant DISTRO\_VARIANT

在其中安装 OS 变体的虚拟机，比如

'fedora18'、'rhel6'、'winxp' 等等。

--boot BOOT 配置虚拟机引导设置。例如：

--boot hd,cdrom,menu=on

--boot init=/sbin/init (for containers)

--idmap IDMAP 为 LXC 容器启用用户名称空间。例如：

--idmap uid\_start=0,uid\_target=1000,uid\_count=10

设备选项:

--disk DISK 使用不同选项指定存储。例如：

--disk size=10 (new 10GiB image in default location)

--disk /my/existing/disk,cache=none

--disk device=cdrom,bus=scsi

--disk=?

-w NETWORK, --network NETWORK

配置虚拟机网络接口。例如：

--network bridge=mybr0

--network network=my\_libvirt\_virtual\_net

--network network=mynet,model=virtio,mac=00:11...

--network none

--network help

--graphics GRAPHICS 配置虚拟机显示设置。例如：

--graphics vnc

--graphics spice,port=5901,tlsport=5902

--graphics none

--graphics vnc,password=foobar,port=5910,keymap=ja

--controller CONTROLLER

配置虚拟机控制程序设备。例如：

--controller type=usb,model=ich9-ehci1

--input INPUT 配置虚拟机输入设备。例如：

--input tablet

--input keyboard,bus=usb

--serial SERIAL 配置虚拟机串口设备

--parallel PARALLEL 配置虚拟机并口设备

--channel CHANNEL 配置虚拟机沟通频道

--console CONSOLE 配置虚拟机与主机之间的文本控制台连接

--hostdev HOSTDEV 将物理 USB/PCI/etc

主机设备配置为与虚拟机共享

--filesystem FILESYSTEM

将主机目录传递给虚拟机。例如：

--filesystem /my/source/dir,/dir/in/guest

--filesystem template\_name,/,type=template

--sound [SOUND] 配置虚拟机声音设备模拟

--watchdog WATCHDOG 配置虚拟机 watchdog 设备

--video VIDEO 配置虚拟机视频硬件。

--smartcard SMARTCARD

配置虚拟机智能卡设备。例如：

--smartcard mode=passthrough

--redirdev REDIRDEV 配置虚拟机重定向设备。例如：

--redirdev usb,type=tcp,server=192.168.1.1:4000

--memballoon MEMBALLOON

配置虚拟机 memballoon 设备。例如：

--memballoon model=virtio

--tpm TPM 配置虚拟机 TPM 设备。例如：

--tpm /dev/tpm

--rng RNG 配置虚拟机 RNG 设备。例如：

--rng /dev/random

--panic PANIC 配置虚拟机 panic 设备。例如：

--panic default

虚拟机配置选项:

--security SECURITY 设定域安全驱动器配置。

--numatune NUMATUNE 为域进程调整 NUMA 策略。

--memtune MEMTUNE 为域进程调整内粗策略。

--blkiotune BLKIOTUNE

为域进程调整 blkio 策略。

--memorybacking MEMORYBACKING

为域进程设置内存后备策略。例如：

--memorybacking hugepages=on

--features FEATURES 设置域 <features> XML。例如：

--features acpi=off

--features apic=on,eoi=on

--clock CLOCK 设置域 <clock> XML。例如：

--clock offset=localtime,rtc\_tickpolicy=catchup

--pm PM 配置 VM 电源管理功能

--events EVENTS 配置 VM 生命周期管理策略

--resource RESOURCE 配置 VM 资源分区（cgroups）

虚拟化平台选项:

-v, --hvm 客户端应该是一个全虚拟客户端

-p, --paravirt 这个客户端一个是一个半虚拟客户端

--container 这台虚拟机需要一个容器客户端

--virt-type HV\_TYPE 要使用的管理程序名称(kvm、qemu、xen

等等)

--arch ARCH 模拟的 CPU 构架

--machine MACHINE 要模拟的机器类型

其它选项:

--autostart 引导主机时自动启动域。

--wait WAIT 等待安装完成的分钟数。

--noautoconsole 不要自动尝试连接到客户端控制台

--noreboot 完成安装后不要引导虚拟机。

--print-xml [XMLONLY]

输出所生成域 XML，而不是创建虚拟机。

--dry-run 完成安装步骤，但不要创建设备或者定义

虚拟机。

--check CHECK 启用或禁用验证检查。例如：

--check path\_in\_use=off

--check all=off

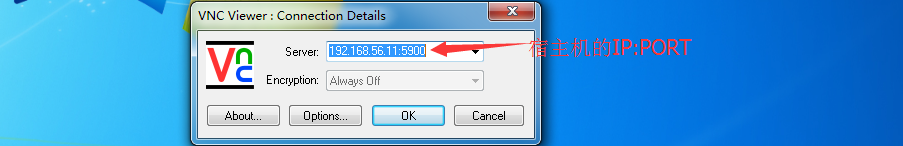
-q, --quiet 禁止无错误输出

-d, --debug 输入故障排除信息

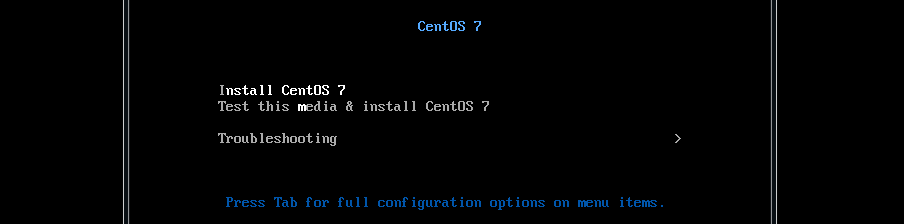
使用 '--option=?' 或者 '--option help' 查看可用子选项

有关示例及完整选项语法，请查看 man page。

2.2.7：通过vnc客户端连接到安装界面,kvm虚拟机的默认端口从5900开始



2.2.8：选择安装centos安装界面：



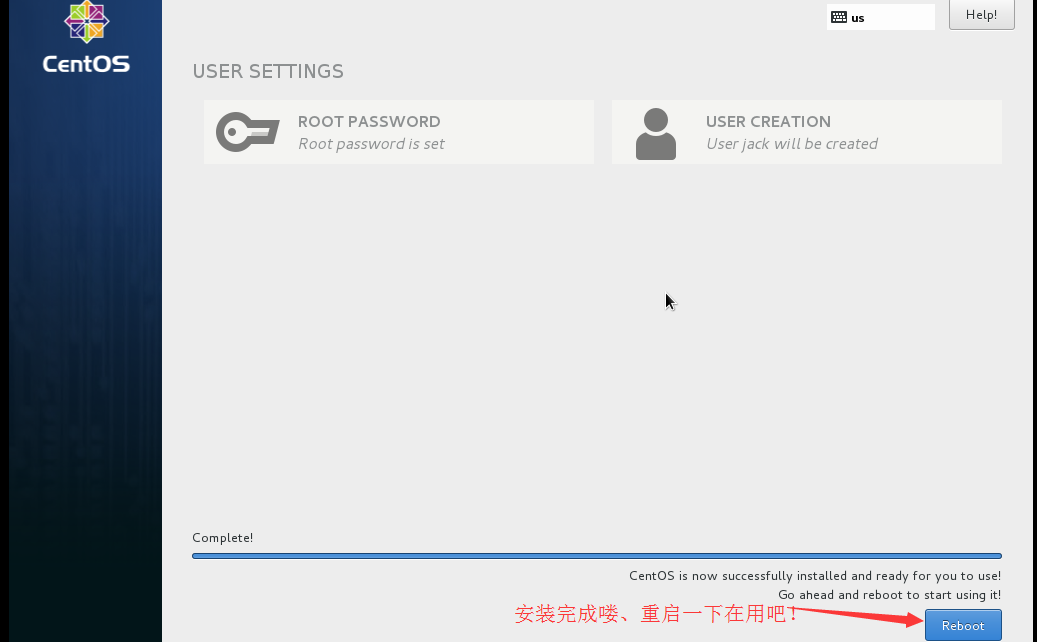
2.2.9：开始正常安装旅程，以下为给内核传递net.ifnames=0以及biosdevname=0使网卡的命名为eth\*,否则为em\*：



2.2.10：安装过程中。。。。。。：



2.2.11：安装完成：



2.3:查看正在运行的虚拟机：

[root@linux-node1 ~]# virsh -help #获取命令帮助[root@linux-node1 ~]# virsh list --all #当前正在运行的所有虚拟机

Id 名称 状态----------------------------------------------------

2 CentOS-7-x86\_64 running - CentOS-x86\_64 关闭

- CentOS\_x86\_64 关闭[root@linux-node1 ~]# virsh list --help #命令帮助

OPTIONS --inactive 列出不活跃的域

--all 不活跃和活跃的域列表

--transient 列出临时域

--persistent 列出持久域

--with-snapshot 列出现有快照的域

--without-snapshot 列出没有快照的域

--state-running 运行状态的域列表

--state-paused 列出暂停状态的域

--state-shutoff 列出关闭状态的域

--state-other 列出其他状态的域

--autostart 列出启用 autostart 的域

--no-autostart 列出禁用 autostart 的域

--with-managed-save 列出有管理的保存状态的域

--without-managed-save 列出没有管理的保存状态的域

--uuid 只列出 uuid --name 只列出域名

--table 列出表格（默认）

--managed-save 标记有管理的保存状态的域

--title show domain title[root@linux-node1 ~]# virsh list --inactive #列出关闭的虚拟机

Id 名称 状态----------------------------------------------------

- CentOS-x86\_64 关闭

- CentOS\_x86\_64 关闭

三：虚拟机管理工具libvirt：

是一个免费开源的主流的虚拟化管理工具，有两个组成进程，一个后台主进程，另外提供API，openstack就是通过libvirt的API管理虚拟机，virt-install就是通过libvirt完成的，服务名称叫做libvirtd，即使此服务down了虚拟机不受影响，但是无法管理，openstack挂了虚拟机不受影响

3.1：配置文件路径：

[root@linux-node1 ~]# cd /etc/libvirt/qemu/

[root@linux-node1 qemu]# ls #保存了虚拟机的配置xml配置文件，不能使用vim直接更改虚拟机配置文件，需要使用virsh edit xxx编辑  
CentOS-7-x86\_64.xml CentOS\_x86\_64.xml CentOS-x86\_64.xml networks

3.2：一个完整的虚拟机配置文件：

[root@linux-node1 qemu]# cat CentOS-7-x86\_64.xml<!--

WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE

OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:

virsh edit CentOS-7-x86\_64

or other application using the libvirt API.

--><domain type='kvm'> #虚拟化类型kvm <name>CentOS-7-x86\_64</name> #虚拟机名称 <uuid>2d92275e-4dab-4f10-adea-c874ea49c9a6</uuid> #唯一标示UUID <memory unit='KiB'>1048576</memory> #内存 <currentMemory unit='KiB'>1048576</currentMemory> #当前内存 <vcpu placement='static'>1</vcpu> #1个cpu并且是静态分配 <os>

<type arch='x86\_64' machine='pc-i440fx-rhel7.0.0'>hvm</type> #hvm标示是硬件虚拟化 <boot dev='hd'/> #从硬盘启动 </os>

<features>

<acpi/>

<apic/>

</features>

<cpu mode='custom' match='exact'>

<model fallback='allow'>IvyBridge</model>

</cpu>

<clock offset='utc'>

<timer name='rtc' tickpolicy='catchup'/>

<timer name='pit' tickpolicy='delay'/>

<timer name='hpet' present='no'/>

</clock>

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

<pm>

<suspend-to-mem enabled='no'/>

<suspend-to-disk enabled='no'/>

</pm>

<devices>

<emulator>/usr/libexec/qemu-kvm</emulator>

<disk type='file' device='disk'> #磁盘信息 <driver name='qemu' type='raw'/>

<source file='/opt/CentOS-7-x86\_64.raw'/>

<target dev='vda' bus='virtio'/> #虚拟磁盘，半虚拟化 <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>

</disk>

<disk type='block' device='cdrom'>

<driver name='qemu' type='raw'/>

<target dev='hda' bus='ide'/>

<readonly/>

<address type='drive' controller='0' bus='0' target='0' unit='0'/>

</disk>

<controller type='usb' index='0' model='ich9-ehci1'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x7'/>

</controller>

<controller type='usb' index='0' model='ich9-uhci1'> #usb信息 <master startport='0'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0' multifunction='on'/>

</controller>

<controller type='usb' index='0' model='ich9-uhci2'>

<master startport='2'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x1'/>

</controller>

<controller type='usb' index='0' model='ich9-uhci3'>

<master startport='4'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x2'/>

</controller>

<controller type='pci' index='0' model='pci-root'/>

<controller type='ide' index='0'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>

</controller>

<controller type='virtio-serial' index='0'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>

</controller>

<interface type='network'> #网络信息 <mac address='52:54:00:60:84:d0'/>

<source network='default'/>

<model type='virtio'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>

</interface>

<serial type='pty'>

<target port='0'/>

</serial>

<console type='pty'>

<target type='serial' port='0'/>

</console>

<channel type='unix'>

<source mode='bind' path='/var/lib/libvirt/qemu/channel/target/domain-CentOS-7-x86\_64/org.qemu.guest\_agent.0'/>

<target type='virtio' name='org.qemu.guest\_agent.0'/>

<address type='virtio-serial' controller='0' bus='0' port='1'/>

</channel>

<input type='tablet' bus='usb'/>

<input type='mouse' bus='ps2'/> #鼠标和键盘 <input type='keyboard' bus='ps2'/>

<graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'> #vnc接口，从5900开始，可以自定义 <listen type='address' address='0.0.0.0'/>

</graphics>

<video>

<model type='cirrus' vram='16384' heads='1'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>

</video>

<memballoon model='virtio'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>

</memballoon>

</devices></domain>

3.3:dumpxml命令，导出虚拟机的xml文件：

3.3.1：命令帮助：

[root@linux-node1 qemu]# virsh dumpxml --help

NAME

dumpxml - XML 中的域信息

SYNOPSIS

dumpxml <domain> [--inactive] [--security-info] [--update-cpu] [--migratable]

DESCRIPTION

把域信息作为一个 XML 输出到 stdout。

OPTIONS

[--domain] <string> 域名，id 或 uuid

--inactive 显示不活跃定义的 XML

--security-info 包括 XML 转储中与安全性相关的信息

--update-cpu 根据主机 CPU 更新虚拟机 CPU

--migratable 为迁移提供 XML 可用性

3.3.2：导出虚拟机xml配置文件，xml是虚拟机的静态描述文件，如果虚拟机磁盘未被删除则可以恢复，如果虚拟机磁盘被删除则即使有描述文件也难以恢复：

[root@linux-node1 qemu]# virsh dumpxml CentOS-7-x86\_64 > Centos-7-x86\_64.bak.xm

3.4:undefine：彻底摧毁一个虚拟机

3.4.1：命令帮助信息

[root@linux-node1 ~]# virsh undefine --help

NAME

undefine - 取消定义一个域

SYNOPSIS

undefine <domain> [--managed-save] [--storage <string>] [--remove-all-storage] [--wipe-storage] [--snapshots-metadata] [--nvram]

DESCRIPTION 取消定义一个域或者将持久转换为临时。

OPTIONS [--domain] <string> 域名或 uuid --managed-save 删除域管理的状态文件

--storage <string> 删除关联的存储卷（用逗号分开的目标或者源路径列表）（查看 domblklist）

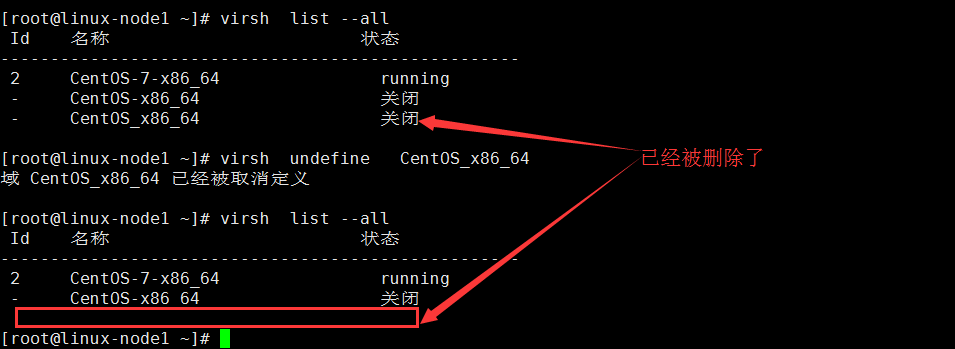
--remove-all-storage 删除关联的存储卷（小心使用）

--wipe-storage 擦除删除卷中的数据

--snapshots-metadata 如果不活跃则删除所有域快照元数据

--nvram remove nvram file, if inactive

3.4.2：删除一个虚拟机：



3.4.2：一个完整的备份删除和恢复的过程：

[root@linux-node1 ~]# virsh list --all #当前所有的虚拟机有两个

Id 名称 状态----------------------------------------------------

2 CentOS-7-x86\_64 running - CentOS-x86\_64 关闭[root@linux-node1 ~]# virsh dumpxml CentOS-x86\_64 > CentOS-x86\_64.xml.bak #备份[root@linux-node1 ~]# virsh undefine CentOS-x86\_64 #摧毁一个虚拟机域 CentOS-x86\_64 已经被取消定义

[root@linux-node1 ~]# virsh list --all #当前所有的虚拟机还有一个了

Id 名称 状态----------------------------------------------------

2 CentOS-7-x86\_64 running[root@linux-node1 ~]# virsh define /root/CentOS-x86\_64.xml.bak #从备份的xml文件中恢复虚拟机定义域 CentOS-x86\_64（从 /root/CentOS-x86\_64.xml.bak）

[root@linux-node1 ~]# virsh list --all #当前有俩虚拟机了，欧耶！

Id 名称 状态----------------------------------------------------

2 CentOS-7-x86\_64 running - CentOS-x86\_64 关闭

四：CPU管理：

4.1：命令帮助，在安装的时候指定CPU的特性：

[root@linux-node1 ~]# virt-install --help | grep cpu

--vcpus VCPUS 为虚拟机配置的 vcpus 数。例如：

--vcpus 5 #指定CPU的数量

--vcpus 5,maxcpus=10,cpuset=1-4,6,8 #默认是5最大是10，set为设值的某一个物理CPU核心上，相当于Nginx的CPU绑定

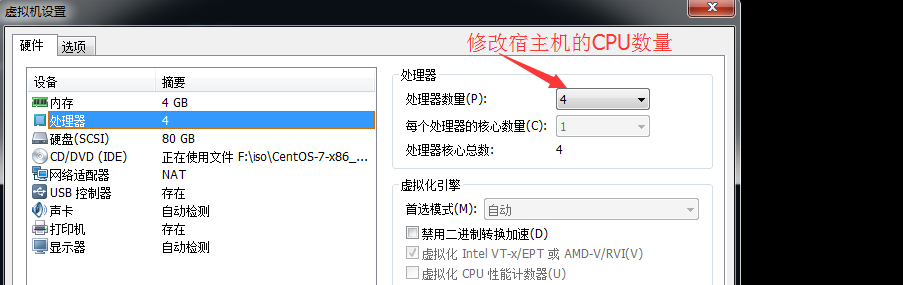
--vcpus sockets=2,cores=4,threads=2,

--cpu CPU CPU 型号及功能。例如：

--cpu coreduo,+x2apic --cpu host

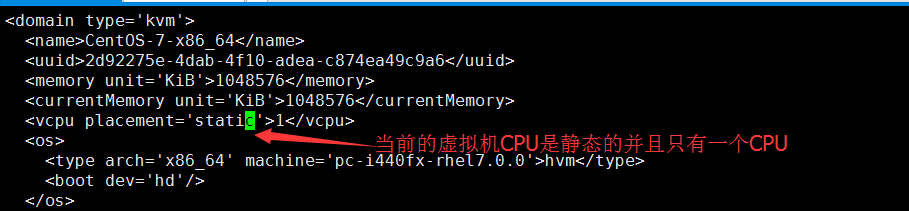
4.2：修改已经存在虚拟机的CPU：

4.2.1:修改宿主机的CPU数量并重启宿主机；

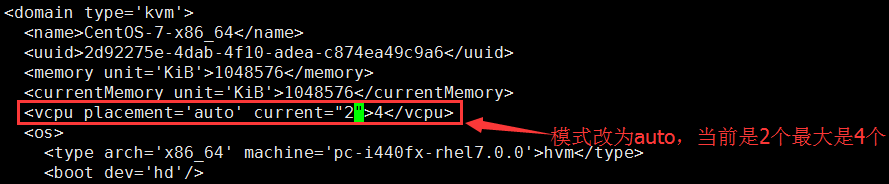


4.2.2：修改虚拟机的xml描述文件：

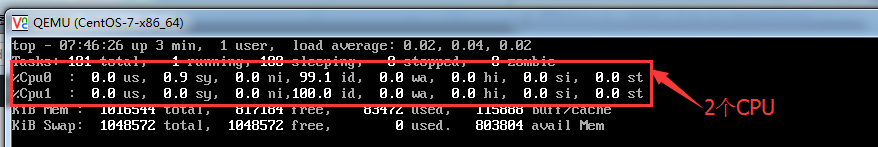
[root@linux-node1 ~]# virsh edit CentOS-7-x86\_64 #必须通过vim编辑配置文件



4.2.3：修改CPU配置如下：

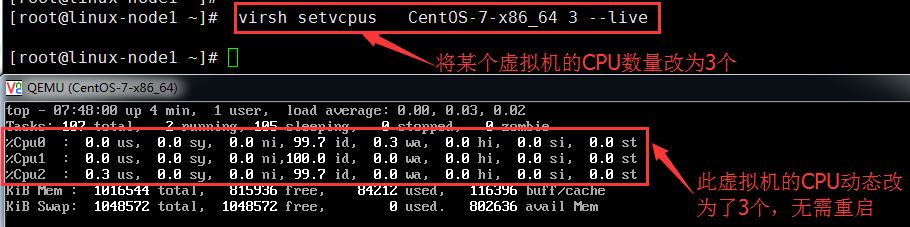


4.2.4：重启虚拟机验证CPU数量：



4.2.5：动态修改CPU数量：

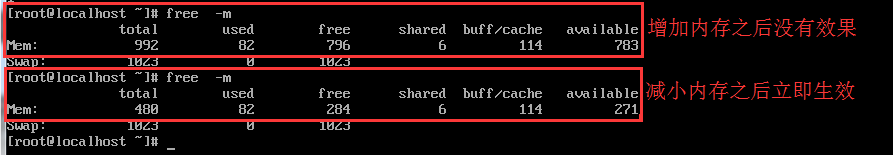
[root@linux-node1 ~]# virsh setvcpus CentOS-7-x86\_64 2 –live #需要宿主机有足够的CPU支撑，即修改的数量不能大于宿主机的CPU数量：



# [root@linux-node1 ~]# cat /sys/devices/system/cpu/cpu3/online #查看新添加的CPU是否在线，即是否处于工作状态  
1  
4.2.6：减小内存，如果创建虚拟机的时候内存是固定的，那么只能减少内存而不能动态添加内存：

[root@linux-node1 ~]# virsh qemu-monitor-command CentOS-7-x86\_64 --hmp --cmd info balloon #查看当前内存balloon: actual=1024[root@linux-node1 ~]# virsh qemu-monitor-command CentOS-7-x86\_64 --hmp --cmd balloon 2048 #尝试增加内存[root@linux-node1 ~]# virsh qemu-monitor-command CentOS-7-x86\_64 --hmp --cmd info balloon #增加之后还是之前的大小balloon: actual=1024[root@linux-node1 ~]# virsh qemu-monitor-command CentOS-7-x86\_64 --hmp --cmd balloon 512 #尝试减少内存[root@linux-node1 ~]# virsh qemu-monitor-command CentOS-7-x86\_64 --hmp --cmd info balloon #减少之后的大小balloon: actual=512

4.2.7：增加和减小内存之后的效果：



**二：虚拟化之KVM(下)**

2017-07-28 11:49:47

标签：[kvm](http://blog.51cto.com/tag-kvm.html)

**二：虚拟化之KVM(下)**

阅读(636)

一：虚拟机磁盘的格式：

1.1：根据存储数据方式的不同可以分为两种格式，一种是稀疏模式。一种是全镜像模式，全镜像模式无法做快照，IO层面是有qemu模拟的，CPU和内存是有KVM实现的，以下是KVM的功能：

KVM 所支持的功能包括：支持CPU 和 memory 超分（Overcommit）支持半虚拟化I/O （virtio）支持热插拔 （cpu，块设备、网络设备等）支持对称多处理（Symmetric Multi-Processing，缩写为 SMP ）支持实时迁移（Live Migration）支持 PCI 设备直接分配和 单根I/O 虚拟化 （SR-IOV）支持 内核同页合并 （KSM ）支持 NUMA （Non-Uniform Memory Access，非一致存储访问结构 ）

1.2：KVM工具集合：

libvirt：操作和管理KVM虚机的虚拟化 API，使用 C 语言编写，可以由 Python,Ruby, Perl, PHP, Java 等语言调用。可以操作包括 KVM，vmware，XEN，Hyper-v, LXC 等 Hypervisor。Virsh：基于 libvirt 的 命令行工具 （CLI）Virt-Manager：基于 libvirt 的 GUI 工具virt-v2v：虚机格式迁移工具virt-\* 工具：包括 Virt-install （创建KVM虚机的命令行工具）， Virt-viewer （连接到虚机屏幕的工具），Virt-clone（虚机克隆工具），virt-top 等sVirt：安全工具

1.3：磁盘格式：

1.3.1：raw：指定多大就创建多大，直接占用指定大小的空间：  
老牌的格式了，用一个字来说就是裸，也就是赤裸裸，你随便dd一个file就模拟了一个raw格式的镜像。由于裸的彻底，性能上来说的话还是不错的。目前来看，KVM和XEN默认的格式好像还是这个格式。因为其原始，有很多原生的特性，例如直接挂载也是一件简单的事情。裸的好处还有就是简单，支持转换成其它格式的虚拟机镜像对裸露的它来说还是很简单的（如果  
其它格式需要转换，有时候还是需要它做为中间格式），空间使用来看，这个很像磁盘，使用多少就是多少（du -h看到的大小就是使用大小），但如果你要把整块磁盘都拿走的话得全盘拿了（copy镜像的时候），会比较消耗网络带宽和I/O。接下来还有个有趣的问题，如果那天你的硬盘  
用着用着不够用了，你咋办，在买一块盘。但raw格式的就比较犀利了，可以在原来的盘上追加空间：  
dd if=/dev/zero of=zeros.raw bs=1024k count=4096（先创建4G的空间）  
cat foresight.img zeros.raw > new-foresight.img（追加到原有的镜像之后）  
当然，好东西是吹出来的，谁用谁知道，还是有挺多问题的。由于原生的裸格式，不支持snapshot也是很正常的。传说有朋友用版本管理软件对raw格式的文件做版本管理从而达到snapshot的能力，估计可行，但没试过，这里也不妄加评论。但如果你使用LVM的裸设  
备，那就另当别论。说到LVM还是十分的犀利的，当年用LVM做虚拟机的镜像，那性能杠杠的。而且现在好多兄弟用虚拟化都采用LVM来做的。在LVM上做了很多的优化，国外听说也有朋友在LVM增量备份方面做了很多的工作。目前来LVM的snapshot、性能、可扩展性方面都还是有相当的  
效果的。目前来看的话，备份的话也问题不大。就是在虚拟机迁移方面还是有很大的限制。但目前虚拟化的现状来看，真正需要热迁移的情况目前需求还不是是否的强烈。虽然使用LVM做虚拟机镜像的相关公开资料比较少，但目前来看牺牲一点灵活性，换取性能和便于管理还是不错的选择。对于LVM相关的特性及使用可以参考如下链接：  
<http://www.ibm.com/developerworks/linux/library/l-lvm2/index.html>

1.3.2：cow:

曾经qemu的写时拷贝的镜像格式，目前由于历史遗留原因不支持窗口模式。从某种意义上来说是个弃婴，还没得它成熟就死在腹中，后来被qcow格式所取代。

1.3.3:qcow:

一代的qemu的cow格式，刚刚出现的时候有比较好的特性，但其性能和raw格式对比还是有很大的差距，目前已经被新版本的qcow2取代。其性能可以查看如下链接：  
<http://www.linux-kvm.org/page/Qcow2>

1.3.4:qcow2，是openstack默认也是比较推荐的格式，将差异保存在一个文件，文件比较小而且做快照也比较小，空间是动态增长的：  
现在比较主流的一种虚拟化镜像格式，经过一代的优化，目前qcow2的性能上接近raw裸格式的性能，这个也算是redhat的官方渠道了,对于qcow2的格式，几点还是比较突出的，qcow2的snapshot，可以在镜像上做N多个快照：更小的存储空间，即使是不支持holes的文件系统也可以（这下du -h和ls -lh看到的就一样了）,支持多个snapshot，对历史snapshot进行管理,支持zlib的磁盘压缩,支持AES的加密

1.3.5:vmdk   
VMware的格式，这个格式说的蛋疼一点就有点牛X，原本VMware就是做虚拟化起家，自己做了一个集群的VMDK的pool，做了自己的虚拟机镜像格式。又拉着一些公司搞了一个OVF的统一封包。从性能和功能上来说，vmdk应该算最出色的，由于vmdk结合了VMware的很多能力，目前来看，KVM和XEN使用这种格式的情况不是太多。但就VMware的Esxi来看，它的稳定性和各方面的能力还是可圈可点。

1.3.6:vdi  
VirtualBox 1.1 compatible image format, for exchanging images with VirtualBox.SUN收购了VirtualBox，Oracle又收购了SUN，这么说呢，vdi也算虚拟化这方面的一朵奇葩，可惜的是入主的两家公司。SUN太专注于技术（可以说是IT技术最前端也不为过），Oracle又是开源杀手（mysql的没落）。单纯从能力上来说vdi在VirtualBox上的表现还是不错的。也是不错的workstation级别的产品。

1.4:磁盘格式的转换：

1.4.1：raw转换为qcow2：

此步骤使用qemu-img工具实现，如果机器上没有，可以通过rpm或yum进行安装，包名为qemu-img。

qemu-img是专门虚拟磁盘映像文件的qemu命令行工具。

具体命令如下：

qemu-img convert -f raw centos.img -O qcow2 centos.qcow2 参数说明：convert 将磁盘文件转换为指定格式的文件

-f 指定需要转换文件的文件格式

-O 指定要转换的目标格式

转换完成后，将新生产一个目标映像文件，原文件仍保存。

1.4.2：qcow2转换为raw：

# qemu-img convert -O qcow2 my.raw myqow.qcow

1.4.3：VMDK转换为qcow2:  
# qemu-img convert -f vmdk -O qcow2 xxx.vmdk xxx.img

<http://www.ibm.com/developerworks/cn/linux/l-cn-mgrtvm3/index.html>

1.5：raw实际空间的大小：

[root@linux-node1 opt]# qemu-img convert -O qcow2 CentOS-7-x86\_64.raw myqcow.qcow[root@linux-node1 opt]# ll -h总用量 7.4G-rw-r--r-- 1 qemu qemu 4.1G 9月 3 13:23 CentOS-7-x86\_64-DVD-1511.iso-rw-r--r-- 1 root root 10G 9月 10 11:09 CentOS-7\_x86\_64.raw-rw-r--r-- 1 qemu qemu 10G 9月 22 21:16 CentOS-7-x86\_64.raw #raw格式直接占用实际分配的空间大小-rw-r--r-- 1 root root 1.1G 9月 22 21:17 myqcow.qcow #转化怒为qcow2格式之后在占用的是实际空间大小

1.6：查看磁盘详细信息：

[root@linux-node1 opt]# qemu-img info CentOS-7-x86\_64.raw #查看raw格式的磁盘信息image: CentOS-7-x86\_64.raw

file format: raw

virtual size: 10G (10737418240 bytes)disk size: 1.1G[root@linux-node1 opt]# qemu-img info myqcow.qcow #查看qcow格式的磁盘信息image: myqcow.qcow

file format: qcow2

virtual size: 10G (10737418240 bytes)disk size: 1.1Gcluster\_size: 65536Format specific information:

compat: 1.1

lazy refcounts: false

二：关于网络:

2.1：默认是使用的nat网络，会称为虚拟机的瓶颈，因此需要创建一个桥接网卡并把虚拟机桥接至网卡，然后将虚拟机桥接一下即可使用物理网络：

2.1.1:通过命令创建桥接网卡：

[root@linux-node1 opt]# brctl addbr br0

[root@linux-node1 opt]# brctl addif br0 eth0 #会断网，这是将br0和eth0进行关联

ip addr del dev eth0 192.168.56.11/24 #删除网卡的IP地址信息

ifconfig br0 192.168.56.11/24 up #为br0配置IP并启动网卡，此时可以远程连接

[root@linux-node1 ~]# route add default gw 192.168.56.2  
[root@linux-node1 ~]# ping www.baidu.com  
PING www.a.shifen.com (112.80.248.73) 56(84) bytes of data.  
64 bytes from 112.80.248.73: icmp\_seq=1 ttl=128 time=70.0 ms  
64 bytes from 112.80.248.73: icmp\_seq=2 ttl=128 time=74.7 ms

2.1.2：通过配置文件文件创建桥接网卡：

# cd /etc/sysconfig/network-scripts/

# cp ifcfg-eth0 ifcfg-br0

# vim ifcfg-eth0 #去掉eth0网卡配置文件中的UUID/MAC/IP等信息BOOTPROTO=static

DEVICE=eth0

ONBOOT=yes

BRIDGE=br0 #添加一个桥接到br0的配置NM\_CONTROLLED=no

br0的配置信息如下：

# vim ifcfg-br0 #里面也不要有UUID和MAC等信息，IP地址配置成eth0使用的地址即可，当然也可以是其他可以使用的IPTYPE=Bridge #类型为桥接BOOTPROTO=static

DEFROUTE=yes

PEERDNS=yes

PEERROUTES=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_PEERDNS=yes

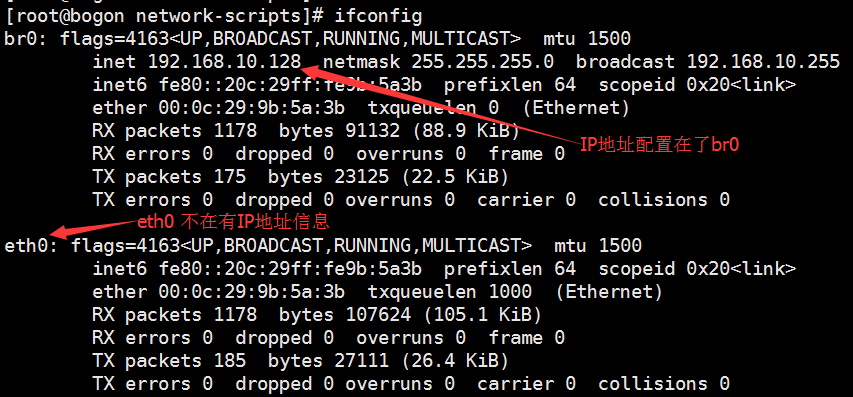
IPV6\_PEERROUTES=yes

IPV6\_FAILURE\_FATAL=no

NAME=br0 #网卡名称DEVICE=br0 #设备名称ONBOOT=yes

IPADDR=192.168.10.128NETMASK=255.255.255.0GATEWAY=192.168.10.2DNS1=192.168.10.2

#最后重启服务器，验证重启服务器之后br0网卡生效，然后在服务器上创建的虚拟机就可以通过桥接网卡的方式使用和物理机同网段的IP地址连接外网了：



2.2:修改虚拟机配置文件：

[root@linux-node1 ~]# virsh edit CentOS-7-x86\_64

<interface type='bridge'> #类型改为bridge

<mac address='52:54:00:60:84:d0'/>

<source bridge='br0'/> #注意是bridgr='br0'

<model type='virtio'/> #类型为通过virtio实现的

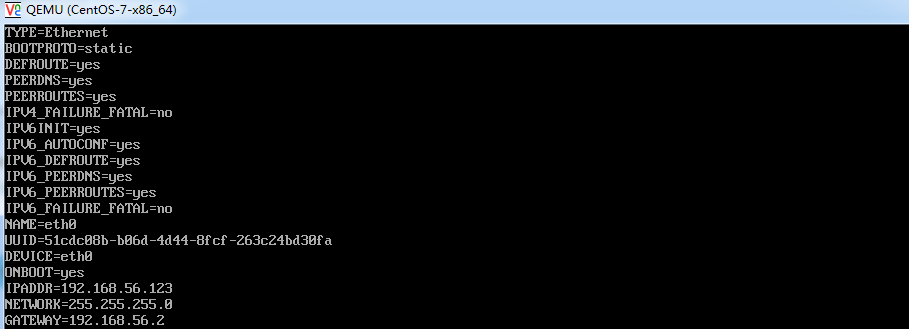
<address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>

</interface>

2.3:重启虚拟机：

[root@linux-node1 ~]# virsh shutdown CentOS-7-x86\_64域 CentOS-7-x86\_64 被关闭[root@linux-node1 ~]# virsh start CentOS-7-x86\_64域 CentOS-7-x86\_64 已开始

2.3：编辑虚拟机网卡，配置IP和网关：



2.4：将上一步保存退出后重启网络访问并测试连接及确认可以通过外网安装服务：



三：宿主机的优化：

全虚拟化和半虚拟化，内核态Ring 0可以直接访问周边的硬件设备，CPU、内存、网卡和硬盘等，应用运行在Ring 3是用户态，用户态不能直接操作硬件，如果要操作硬件会产生系统调用，客户端的操作系统不能工作在Ring 0，因此Intel 提供vt-x提供加速上下文切换，KVM在系统里面是一个qemu进程，进程要受到CPU的调度，在多核CPU进程可能会被调度到任意CPU核心，CPU的一级缓存是使用静态内存做的，二级缓存是使用高速的动态内存，缓存是将经常访问的数据缓存下来以加速访问速度，将进程绑定在一个CPU可以提高缓存的命中率：

3.1：CPU绑定优化:

[root@linux-node1 ~]# taskset --help用法：taskset [选项] [掩码 | cpu列表] [pid|命令 [参数...]]选项：

-a, --all-tasks 在给定 pid 的所有任务(线程)上操作

-p, --pid 在存在的给定 pid 上操作

-c, --cpu-list 以列表格式显示和指定 CPU -h, --help 显示此帮助

-V, --version 输出版本信息默认行为是运行一条新命令：

taskset 03 sshd -b 1024您可以获取现有任务的掩码：

taskset -p 700或设置掩码：

taskset -p 03 700使用逗号分隔的列表格式而不是掩码：

taskset -pc 0,3,7-11 700列表格式中的范围可以带一个跨度参数：

例如 0-31:2 与掩码 0x55555555 等效[root@linux-node1 ~]# taskset -cp 2 42340 #将进程号为42340的进程指定运行在第二核心CPUpid 42340's current affinity list: 0-3

pid 42340's new affinity list: 2

3.2：内存优化：

3.2.1：内存EPT技术：

系统将内存识别为虚拟内存，包含物理内存和交换分区，KVM是一个进程，EPT是Intel用于加快内存映射的技术

3.2.2：大页内存，加快内存寻址：

[root@linux-node1 ~]# cat /proc/meminfo MemTotal: 3866944 kBMemFree: 166796 kBMemAvailable: 1775636 kBBuffers: 1632 kBCached: 1691036 kBSwapCached: 52 kBActive: 1682932 kBInactive: 1594028 kBActive(anon): 1090476 kBInactive(anon): 558692 kBActive(file): 592456 kBInactive(file): 1035336 kBUnevictable: 0 kBMlocked: 0 kBSwapTotal: 2097148 kBSwapFree: 2096024 kBDirty: 36 kBWriteback: 0 kBAnonPages: 1584492 kBMapped: 78708 kBShmem: 64876 kBSlab: 283776 kBSReclaimable: 234440 kBSUnreclaim: 49336 kBKernelStack: 10528 kBPageTables: 10088 kB

NFS\_Unstable: 0 kBBounce: 0 kBWritebackTmp: 0 kBCommitLimit: 4030620 kBCommitted\_AS: 2855828 kBVmallocTotal: 34359738367 kBVmallocUsed: 193548 kBVmallocChunk: 34359483232 kBHardwareCorrupted: 0 kBAnonHugePages: 1181696 kBHugePages\_Total: 0HugePages\_Free: 0HugePages\_Rsvd: 0HugePages\_Surp: 0Hugepagesize: 2048 kB #Centos 7默认已开启DirectMap4k: 100160 kBDirectMap2M: 4093952 kB

3.2.3:开启内存合并：

[root@linux-node1 ~]# cat /sys/kernel/mm/transparent\_hugepage/enabled #将连续的没有使用的内存合并为2M一个表，减少内存碎片[always] madvise never

3.2.4：IO优化

使用virtIO半虚拟化的IO技术，让磁盘知道其运行在虚拟机环境当中

3.2.5：磁盘的调度算法：顺序读写远大于随机读写，系统的IO调度器是

[root@linux-node1 ~]# cat /sys/block/sda/queue/scheduler #Centos 7默认只有3个，Centos 6有4个  
noop：不进行调度，用于SSD

[deadline]：最后期限算法，防止写操作因为不能读取而被饿死的情况

cfq：完全公平，公平分配IO访问，Centos 6的默认算法

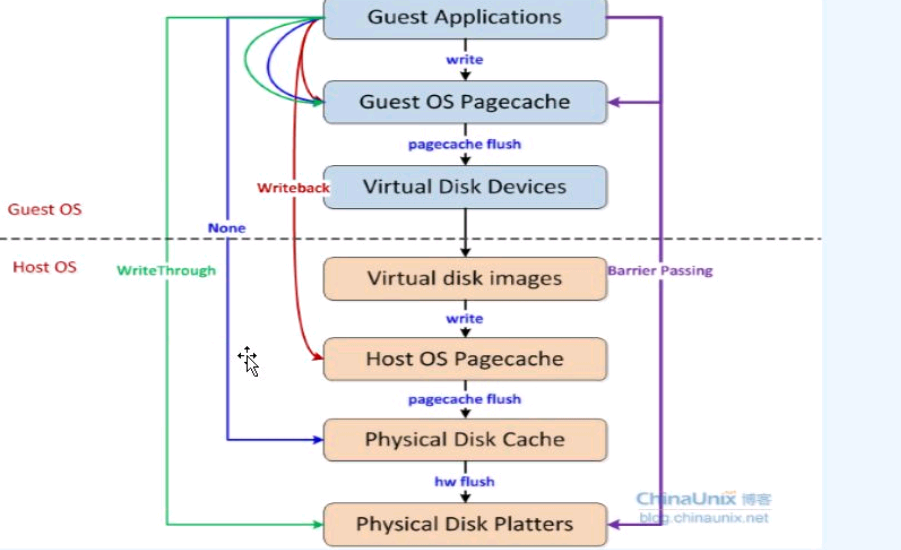
3.2.6：写入磁盘的几种方式：

writeback：同时使用了虚拟机和物理机的pagecache，会被同时写入到虚拟机和物理机的pagecache在写入到物理磁盘缓存，这种方式写入速度快但是假如突然断电会导致出现数据丢失而导致数据一致性出现问题，本方式性能最好但是不安全。

None：将数据直接写入到物理磁盘缓存在写入磁盘，速度次于writeback。

writethrough：直接写入到物理磁盘，突然断电的话数据丢失最少，但是速度比较慢，因为绕过了上面的两层pagecache即没有使用缓存。

web站点适用于使用writeback，即读多写少的情况，对于需要提供数据安全的场景必须数据库等推荐使用writethrough，KVM默认就是使用的writethrough：



http://2.gravatar.com/avatar/5010366747b67897d72f8a85c6698843?s=49&d=mm&r=g作者

**三、SHELL脚本自动部署KVM虚拟化**

原贴：http://zhangyc.blog.51cto.com/955813/1338414

随着IT产业的不断发展，IT技术的不断革新，近几年虚拟化、云计算技术火热，那我们今天来研究一下虚拟化技术，到底什么是虚拟化技术呢？我们为什么要使用虚拟化呢？

虚拟化是指计算元件在虚拟的基础上而不是真实的基础上运行，是一个为了简化管理，优化资源的解决方案。

[虚拟化技术](http://baike.baidu.com/view/13605.htm)可以扩大硬件的容量，简化[软件](http://baike.baidu.com/view/37.htm)的重新配置过程。CPU的[虚拟化技术](http://baike.baidu.com/view/13605.htm)可以单CPU模拟多CPU并行，允许一个平台同时运行多个操作系统，并且[应用程序](http://baike.baidu.com/view/330120.htm)都可以在相互独立的空间内运行而互不影响，从而显著提高计算机的工作效率。

目前主流的虚拟化技术主要有：KVM、Xen、VMware、VirtualBox，虚拟化技术也越来越广泛的应用在企业中，今天我们来研究和复习一下KVM虚拟化集群的搭建。

1、系统版本：

CentOS 6.4 x86\_64

2、处理器需求：

需要一台可以运行最新linux内核的Intel处理器（含VT虚拟化技术）或AMD处理器（含SVM安全虚拟机技术的AMD处理器，也叫AMD-V）

[root@kvm~]# cat /etc/redhat-release

CentOSrelease 6.4 (Final)

3、CPU 支持需求：

[root@localhost~]# egrep 'vmx|svm' /proc/cpuinfo

如果输出的结果包含VMX，它是Intel处理器虚拟机技术标志：如果包含SVM，它是AMD处理器虚拟机技术的标志，如果你什么都没有得到，那应你的系统并没有支持虚拟化的处理，不能使用KVM，另处linux发行版本必须在64bit环境中才能使用KVM。

4、KVM正式安装：

这里直接使用脚本，比较方便快捷，了解更多可以参官方网站。（仅供参考，后期不断优化）

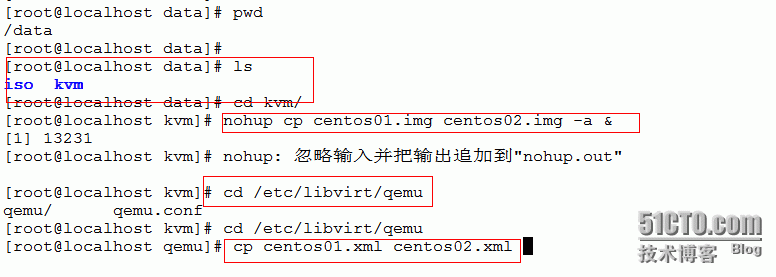
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | #!/bin/sh  #Auto Make KVM Virtualization  #Author wugk 2013-12-06  #Defined Path  cat<<EOF  ++++++++++++++++Welcome To Use Auto Install KVM Scripts ++++++++++++++++++  +++++++++++++++++++++++++This KVM Install Virtual ++++++++++++++++++++++++  +++++++++++++++++++++++++2013-12-06 Author wugk ++++++++++++++++++++++++++  EOF  KVM\_SOFT=(  kvm python-virtinst libvirt bridge-utils virt-manager qemu-kvm-tools virt-viewer virt-v2v libguestfs-tools  )  NETWORK=(  HWADDR=`ifconfigeth0 |egrep"HWaddr|Bcast"|tr"\n"" "|awk'{print $5,$7,$NF}'|sed-e's/addr://g'-e 's/Mask://g'|awk'{print $1}'`  IPADDR=`ifconfigeth0 |egrep"HWaddr|Bcast"|tr"\n"" "|awk'{print $5,$7,$NF}'|sed-e's/addr://g'-e 's/Mask://g'|awk'{print $2}'`  NETMASK=`ifconfigeth0 |egrep"HWaddr|Bcast"|tr"\n"" "|awk'{print $5,$7,$NF}'|sed-e's/addr://g'-e 's/Mask://g'|awk'{print $3}'`  GATEWAY=`route -n|grep"UG"|awk'{print $2}'`  )  #Check whether the system supports virtualization  egrep'vmx|svm'/proc/cpuinfo>>/dev/null  if  [ "$?"-eq"0"];then  echo'Congratulations, your system success supports virtualization !'  else  echo-e 'OH,your system does not support virtualization !\nPlease modify the BIOS virtualization options (Virtualization Technology)'  exit0  fi  if  [ -e /usr/bin/virsh];then  echo"Virtualization is already installed ,Please exit ....";exit0  fi  yum -y install${KVM\_SOFT[@]}  /sbin/modprobekvm  lsmod | grepkvm >>/dev/null  if  [ "$?"-eq"0"];then  echo'KVM installation is successful !'  else  echo'KVM installation is falis,Please check ......'  exit1  fi  cd/etc/sysconfig/network-scripts/  mkdir-p /data/backup/`date+%Y%m%d-%H:%M:%S`  yes|cpifcfg-eth\* /data/backup/`date+%Y%m%d-%H:%M:%S`/  if  [ -e /etc/sysconfig/network-scripts/ifcfg-br0];then  echo"The ifcfg-br0 already exist ,Please wait exit ......"  exit2  else  cat>ifcfg-eth0 <<EOF  DEVICE=eth0  BOOTPROTO=none  ${NETWORK[0]}  NM\_CONTROLLED=no  ONBOOT=yes  TYPE=Ethernet  BRIDGE="br0"  ${NETWORK[1]}  ${NETWORK[2]}  ${NETWORK[3]}  USERCTL=no  EOF  cat>ifcfg-br0 <<EOF  DEVICE="br0"  BOOTPROTO=none  ${NETWORK[0]}  IPV6INIT=no  NM\_CONTROLLED=no  ONBOOT=yes  TYPE="Bridge"  ${NETWORK[1]}  ${NETWORK[2]}  ${NETWORK[3]}  USERCTL=no  EOF  fi  echo'Your can restart Ethernet Service: /etc/init.d/network restart !'  echo'---------------------------------------------------------'  sleep1  echo'Your can restart KVM Service : /etc/init.d/libvirtd restart !'  echo  echo-e "You can create a KVM virtual machine: \nvirt-install --name=centos01 --ram 512 --vcpus=1 --disk path=/data/kvm/centos01.img,size=7,bus=virtio --accelerate --cdrom /data/iso/centos58.iso --vnc --vncport=5910 --vnclisten=0.0.0.0 --network bridge=br0,model=virtio --noautoconsole" |

自此KVM简单安装到此结束，安装完毕，怎样来创建一个虚拟机并且安装呢，首先我们把需要安装的系统ISO镜像上传到/data/iso目录，然后新建/data/kvm虚拟机安装目录，然后启动如下脚本：

|  |  |
| --- | --- |
| 1 | virt-install --name=centos01 --ram 512--vcpus=1--disk path=/data/kvm/centos01.img,size=7,bus=virtio --accelerate --cdrom /data/iso/centos58.iso --vnc --vncport=5910--vnclisten=0.0.0.0--network bridge=br0,model=virtio |

然后在客户端（PC机）使用VNC客户端连接KVM服务端5910端口，进行安装系统即可。

当安装完一台后，如何快速启动第二台呢，这里只需要复制我们刚安装的第一台的两个文件（/data/kvm/centos01.img /etc/libvirt/qemu/centos01.xml），然后做相应修改即可。

[](http://img1.51cto.com/attachment/201312/204720916.png)

这样就配置完毕，怎样重启两台虚拟机呢？命令如下：

（参数主要有reboot、start、shutdown）

|  |  |
| --- | --- |
| 1  2  3  4 | virsh reboot centos01 （重启）  virsh reboot centos02 （重启）  virsh start centos01 （启动）  virsh start centos02 （启动） |

KVM简单操作就先介绍到这里，更多功能和维护在后期会更新。

本文参考如下文章，非常感谢：

<http://koumm.blog.51cto.com/703525/1290191>

<http://g.51cto.com/netengineer/216414>

KVM详解

**kvm虚拟化学习笔记(一)之kvm虚拟化环境安装**

2013-09-04 21:53:31

标签：[虚拟化](http://blog.51cto.com/tag-%E8%99%9A%E6%8B%9F%E5%8C%96.html) [kvm](http://blog.51cto.com/tag-kvm.html) [kvm虚拟化](http://blog.51cto.com/tag-kvm%E8%99%9A%E6%8B%9F%E5%8C%96.html)

原创作品，允许转载，转载时请务必以超链接形式标明文章 [原始出处](http://koumm.blog.51cto.com/703525/1288795) 、作者信息和本声明。否则将追究法律责任。<http://koumm.blog.51cto.com/703525/1288795>

平时一直玩RHEL/CentOS/OEL系列的操作，玩虚拟化也是采这一类系统,kvm在RHEL6系列操作系统支持比较好，本文采用采用OEL6.3操作系统，网上所有文章都说KVM比xen简单，我怎么感觉kvm比较复杂，可能是它的工具太多了吧，所以决定开始记录KVM的学习过程。

本文出自：[http://koumm.blog.51cto.com](http://koumm.blog.51cto.com/)

kvm虚拟化学习笔记(一)之kvm虚拟化环境安装  
<http://koumm.blog.51cto.com/703525/1288795>  
kvm虚拟化学习笔记(二)之linuxkvm虚拟机安装  
<http://koumm.blog.51cto.com/703525/1289627>  
kvm虚拟化学习笔记(三)之windowskvm虚拟机安装  
<http://koumm.blog.51cto.com/703525/1290191>  
kvm虚拟化学习笔记(四)之kvm虚拟机日常管理与配置  
<http://koumm.blog.51cto.com/703525/1290269>  
kvm虚拟化学习笔记(五)之windows虚拟机性能调整  
<http://koumm.blog.51cto.com/703525/1290682>  
kvm虚拟化学习笔记(六)之kvm虚拟机控制台登录配置  
<http://koumm.blog.51cto.com/703525/1290996>  
kvm虚拟化学习笔记(七)之kvm虚拟机克隆  
<http://koumm.blog.51cto.com/703525/1291793>  
kvm虚拟化学习笔记(八)之kvm虚拟机vnc配置  
<http://koumm.blog.51cto.com/703525/1291803>  
kvm虚拟化学习笔记(九)之kvm虚拟机时间配置  
<http://koumm.blog.51cto.com/703525/1291862>  
kvm虚拟化学习笔记(十)之kvm虚拟机快照备份  
<http://koumm.blog.51cto.com/703525/1291893>  
kvm虚拟化学习笔记(十一)之kvm虚拟机扩展磁盘空间  
<http://koumm.blog.51cto.com/703525/1292146>

kvm虚拟化学习笔记(十二)之kvmlinux虚拟机在线扩展磁盘  
<http://koumm.blog.51cto.com/703525/1295296>

kvm虚拟化学习笔记(十三)之kvm虚拟机磁盘文件读取小结  
<http://koumm.blog.51cto.com/703525/1298845>

kvm虚拟化学习笔记(十四)之kvm虚拟机静态迁移  
<http://koumm.blog.51cto.com/703525/1298852>  
kvm虚拟化学习笔记(十五)之kvm虚拟机动态迁移  
<http://koumm.blog.51cto.com/703525/1300783>

kvm虚拟化学习笔记(十六)之kvm虚拟化存储池配置  
<http://koumm.blog.51cto.com/703525/1304196>  
kvm虚拟化学习笔记(十七)之KVM到KVM之v2v迁移

<http://koumm.blog.51cto.com/703525/1304271>

kvm虚拟化学习笔记(十八)之ESXi到KVM之v2v迁移  
<http://koumm.blog.51cto.com/703525/1304461>

kvm虚拟化学习笔记(十九)之convirt集中管理平台搭建  
<http://koumm.blog.51cto.com/703525/1305553>

kvm虚拟化学习笔记(二十)之convirt安装linux系统

<http://koumm.blog.51cto.com/703525/1306526>

kvm虚拟化学习笔记(二十一)之KVM性能优化学习笔记  
<http://koumm.blog.51cto.com/703525/1606422>

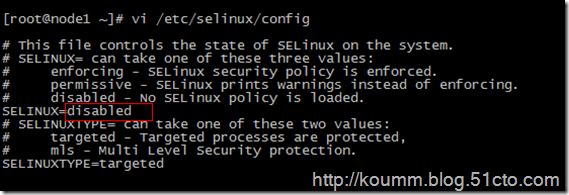
### 1.操作系统安装

本文采用OEL6.3X64操作系统，也可以采用RHEL/CentOS6.x。

(1)修改内核模式为兼容内核启动

[](http://img1.51cto.com/attachment/201309/4/703525_1378302770PTzi.png)

(2)关闭selinux，重启后生效

[](http://img1.51cto.com/attachment/201309/4/703525_13783027749qCX.png)

(3)关闭防火墙

|  |  |
| --- | --- |
| 1  2 | chkconfig ip6tables off  chkconfig iptables off |

(4)重启

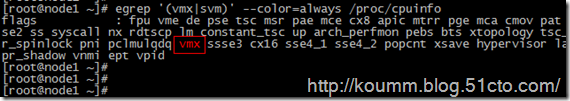
|  |  |
| --- | --- |
| 1 | # reboot |

### 2.虚拟化环境配置

(1)查看是否支持虚拟机

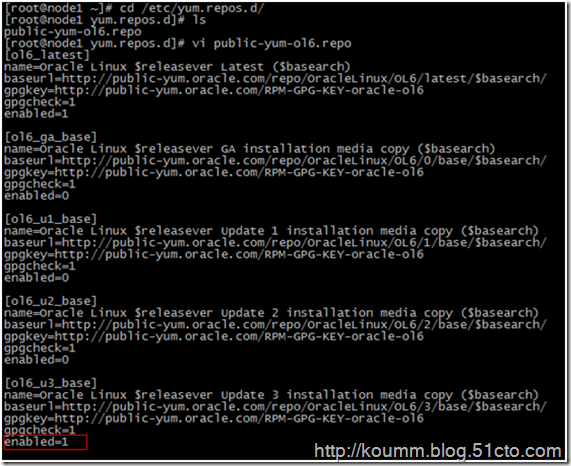
说明1:半虚拟化是不能运行与安装KVM虚拟机的。

|  |  |
| --- | --- |
| 1 | #egrep '(vmx|svm)' --color=always /proc/cpuinfo |

[](http://img1.51cto.com/attachment/201309/4/703525_1378302777Kj9n.png)

(2)配置yum环境

OEL6.3已经配置好源，只需要开启即可。

[](http://img1.51cto.com/attachment/201309/4/703525_1378302785X2rP.png)

(3)安装kvm软件包

|  |  |
| --- | --- |
| 1 | #yum -y install kvm python-virtinst libvirt tunctl bridge-utils virt-manager qemu-kvm-tools virt-viewer virt-v2v |

过程略。

补充安装kvm虚拟化一些管理工具包

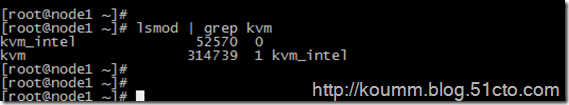
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | | #yum -y install libguestfs-tools | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | Loaded plugins: refresh-packagekit, security  Setting up Install Process  Resolving Dependencies  --> Running transaction check  ---> Package libguestfs-tools.x86\_64 1:1.16.34-2.el6 will be installed  --> Processing Dependency: libguestfs-tools-c = 1:1.16.34-2.el6 for package: 1:libguestfs-tools-1.16.34-2.el6.x86\_64  --> Processing Dependency: perl(XML::Writer) for package: 1:libguestfs-tools-1.16.34-2.el6.x86\_64  --> Processing Dependency: /usr/bin/guestmount for package: 1:libguestfs-tools-1.16.34-2.el6.x86\_64  --> Running transaction check  ---> Package libguestfs-tools-c.x86\_64 1:1.16.34-2.el6 will be installed  --> Processing Dependency: libconfig.so.8()(64bit) for package: 1:libguestfs-tools-c-1.16.34-2.el6.x86\_64  ---> Package perl-XML-Writer.noarch 0:0.606-6.el6 will be installed  --> Running transaction check  ---> Package libconfig.x86\_64 0:1.3.2-1.1.el6 will be installed  --> Finished Dependency Resolution  Dependencies Resolved  =====================================================================================================================  Package Arch Version Repository Size  =====================================================================================================================  Installing:  libguestfs-tools x86\_64 1:1.16.34-2.el6 ol6\_latest 94 k  Installing for dependencies:  libconfig x86\_64 1.3.2-1.1.el6 ol6\_latest 50 k  libguestfs-tools-c x86\_64 1:1.16.34-2.el6 ol6\_latest 850 k  perl-XML-Writer noarch 0.606-6.el6 ol6\_latest 26 k  Transaction Summary  =====================================================================================================================  Install 4 Package(s)  Total download size: 1.0 M  Installed size: 3.6 M  Is this ok [y/N]: y  Downloading Packages:  (1/4): libconfig-1.3.2-1.1.el6.x86\_64.rpm | 50 kB 00:00  (2/4): libguestfs-tools-1.16.34-2.el6.x86\_64.rpm | 94 kB 00:00  (3/4): libguestfs-tools-c-1.16.34-2.el6.x86\_64.rpm | 850 kB 00:04  (4/4): perl-XML-Writer-0.606-6.el6.noarch.rpm | 26 kB 00:00  ---------------------------------------------------------------------------------------------------------------------  Total 115 kB/s | 1.0 MB 00:08  Running rpm\_check\_debug  Running Transaction Test  Transaction Test Succeeded  Running Transaction  Installing : libconfig-1.3.2-1.1.el6.x86\_64 1/4  Installing : 1:libguestfs-tools-c-1.16.34-2.el6.x86\_64 2/4  Installing : perl-XML-Writer-0.606-6.el6.noarch 3/4  Installing : 1:libguestfs-tools-1.16.34-2.el6.x86\_64 4/4  Verifying : perl-XML-Writer-0.606-6.el6.noarch 1/4  Verifying : libconfig-1.3.2-1.1.el6.x86\_64 2/4  Verifying : 1:libguestfs-tools-c-1.16.34-2.el6.x86\_64 3/4  Verifying : 1:libguestfs-tools-1.16.34-2.el6.x86\_64 4/4  Installed:  libguestfs-tools.x86\_64 1:1.16.34-2.el6  Dependency Installed:  libconfig.x86\_64 0:1.3.2-1.1.el6 libguestfs-tools-c.x86\_64 1:1.16.34-2.el6 perl-XML-Writer.noarch 0:0.606-6.el6  Complete! | |

### 3.查看虚拟化环境

(1)查看虚拟机环境

[](http://img1.51cto.com/attachment/201309/4/703525_1378302792Fk2r.png)

(2)查看kvm模块支持

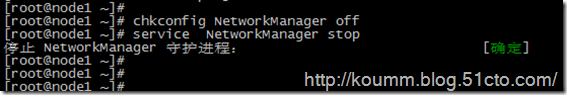
[](http://img1.51cto.com/attachment/201309/4/703525_1378302794hQkj.png)

(3)查看虚拟工具版本

[](http://img1.51cto.com/attachment/201309/4/703525_1378302795niBs.png)

### 4.手动配置虚拟网桥

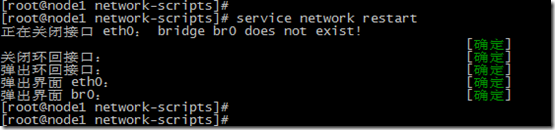
(1)关闭networkmanager服务

[](http://img1.51cto.com/attachment/201309/4/703525_1378302797TFIP.png)

(2)创建br0网桥

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | [root@node1~]#cd /etc/sysconfig/network-scripts/  [root@node1network-scripts]#cp ifcfg-eth0 ifcfg-br0  [root@node1network-scripts]#  [root@node1network-scripts]# | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | [root@node1network-scripts]#vi ifcfg-eth0  DEVICE="eth0"  BOOTPROTO=static  NM\_CONTROLLED="no"  ONBOOT=yes  TYPE="Ethernet"  BRIDGE="br0"  UUID="99dde4d3-94f1-4293-b8a8-b10255d5b8be"  HWADDR=00:0C:29:47:41:8C  IPADDR=192.168.233.130  PREFIX=24  GATEWAY=192.168.233.2  DNS1=192.168.233.2  DEFROUTE=yes  IPV4\_FAILURE\_FATAL=yes  IPV6INIT=no  NAME="Systemeth0"  [root@node1network-scripts]#  [root@node1network-scripts]#vi ifcfg-br0  DEVICE="br0"  BOOTPROTO=static  ONBOOT=yes  TYPE="Bridge"  HWADDR=00:0C:29:47:41:8C  IPADDR=192.168.233.130  PREFIX=24  GATEWAY=192.168.233.2  DNS1=192.168.233.2 |

(3)关闭了networkmanager服务之后，才能通过servicenetworkrestart管理网络。

[](http://img1.51cto.com/attachment/201309/4/703525_1378302799jYDo.png)

(4)查看网桥br0.

[](http://img1.51cto.com/attachment/201309/4/703525_1378302802g9Ee.png)

(4)查看网桥

[](http://img1.51cto.com/attachment/201309/4/703525_13783028100eEX.png)

到此kvm虚拟化环境安装完毕。

**kvm虚拟化学习笔记(二)之linux kvm虚拟机安装**

### 1.上传ISO文件，这里采用OEL5.8x64iso

[](http://img1.51cto.com/attachment/201309/5/703525_1378390788dcbq.png)

### 2.开始安装OEL5.8

**(1)raw格式磁盘**

|  |  |
| --- | --- |
| 1 | virt-install --name=oeltest01 --ram 512 --vcpus=1 --disk path=/data/test02.img,size=7,bus=virtio --accelerate --cdrom/data/iso/oel58x64.iso --vnc --vncport=5910 --vnclisten=0.0.0.0 --network bridge=br0,model=virtio --noautoconsole |

如果采用qcow2磁盘格式，需要事先创建qcow2格式虚拟磁盘

**(2)qcow2格式**(空间动态增长)**2013.9.10日更新**

|  |  |
| --- | --- |
| 1  2 | qemu-img create -f qcow2 test02.img 7G  virt-install --name=oeltest02 --os-variant=RHEL5.8 --ram 512 --vcpus=1 --disk path=/data/test02.img,format=qcow2,size=7,bus=virtio --accelerate --cdrom /data/iso/oel58x64.iso --vnc --vncport=5910 --vnclisten=0.0.0.0 --network bridge=br0,model=virtio --noautoconsole |

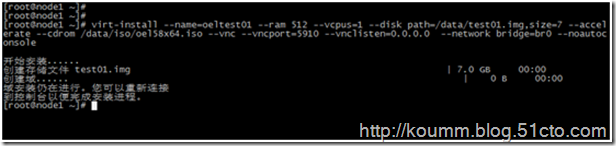
说明：在linux系统安装开始就要注意添加提高性能的一些参数，后面就不需要做一些调整了。

[kvm虚拟化学习笔记(五)之windows虚拟机性能调整](http://koumm.blog.51cto.com/703525/1290682)，这篇文章介绍windows配置virtio的驱动还有些麻烦，最好是在开始安装时注意加入这些参数。

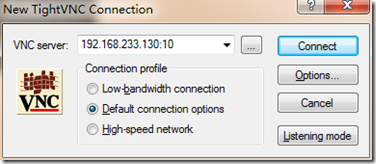
参数说明:

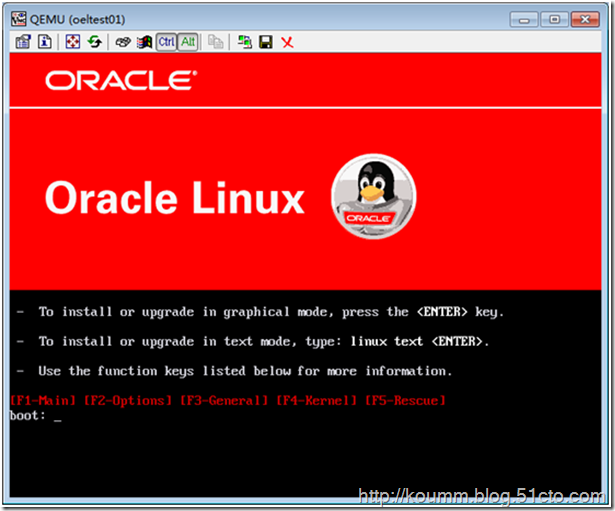
--name指定虚拟机名称  
--ram分配内存大小。  
--vcpus分配CPU核心数，最大与实体机CPU核心数相同  
--disk指定虚拟机镜像，size指定分配大小单位为G。  
--network网络类型，此处用的是默认，一般用的应该是bridge桥接。  
--accelerate加速  
--cdrom指定安装镜像iso  
--vnc启用VNC远程管理，一般安装系统都要启用。  
--vncport指定VNC监控端口，默认端口为5900，端口不能重复。  
--vnclisten指定VNC绑定IP，默认绑定127.0.0.1，这里改为0.0.0.0。

--os-type=linux,windows  
--os-variant=  
win7:MicrosoftWindows7  
vista:MicrosoftWindowsVista  
winxp64:MicrosoftWindowsXP(x86\_64)  
winxp:MicrosoftWindowsXP  
win2k8:MicrosoftWindowsServer2008  
win2k3:MicrosoftWindowsServer2003  
freebsd8:FreeBSD8.x  
generic:Generic  
debiansqueeze:DebianSqueeze  
debianlenny:DebianLenny  
fedora16:Fedora16  
fedora15:Fedora15  
fedora14:Fedora14  
mes5.1:MandrivaEnterpriseServer5.1andlater  
mandriva2010:MandrivaLinux2010andlater  
rhel6:RedHatEnterpriseLinux6  
rhel5.4:RedHatEnterpriseLinux5.4orlater  
rhel4:RedHatEnterpriseLinux4  
sles11:SuseLinuxEnterpriseServer11  
sles10:SuseLinuxEnterpriseServer  
ubuntuoneiric:Ubuntu11.10(OneiricOcelot)  
ubuntunatty:Ubuntu11.04(NattyNarwhal)  
ubuntumaverick:Ubuntu10.10(MaverickMeerkat)  
ubuntulucid:Ubuntu10.04(LucidLynx)  
ubuntuhardy:Ubuntu8.04LTS(HardyHeron)

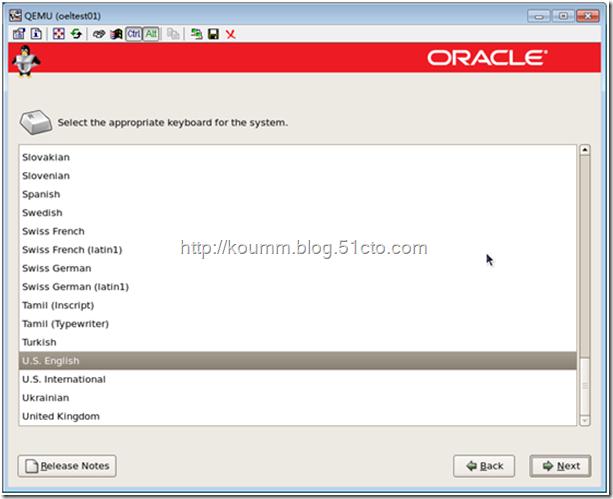
[[](http://img1.51cto.com/attachment/201309/5/703525_1378390790wTwd.png)](http://img1.51cto.com/attachment/201309/5/703525_1378390790wTwd.png)

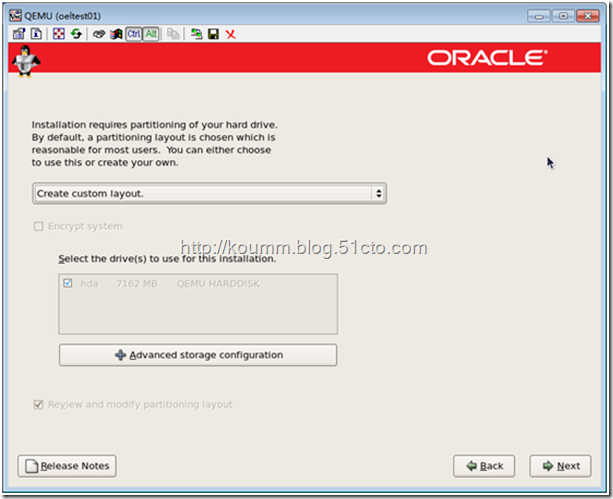
### 3.开始通过VNC进行连接，进行安装过程。

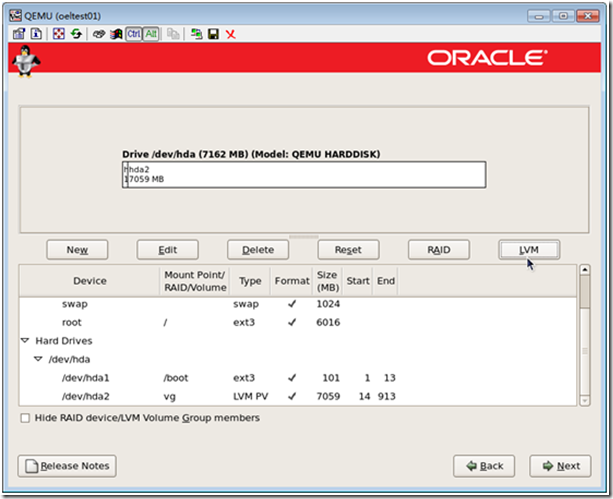
[](http://img1.51cto.com/attachment/201309/5/703525_1378390791ARdL.png)

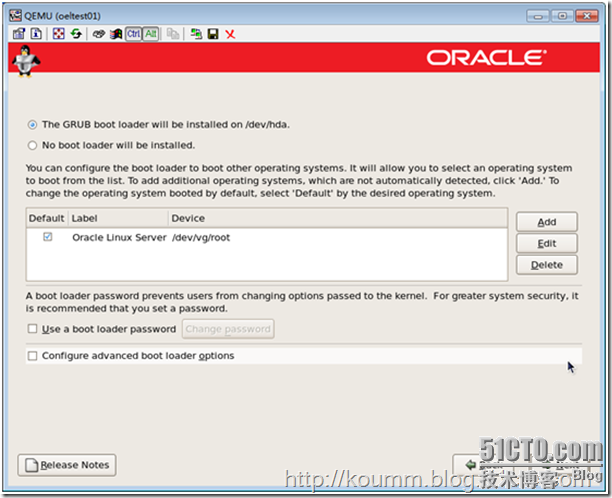
[](http://img1.51cto.com/attachment/201309/5/703525_1378390794ewBZ.png)

[image](http://img1.51cto.com/attachment/201309/5/703525_1378390798FQNg.png)

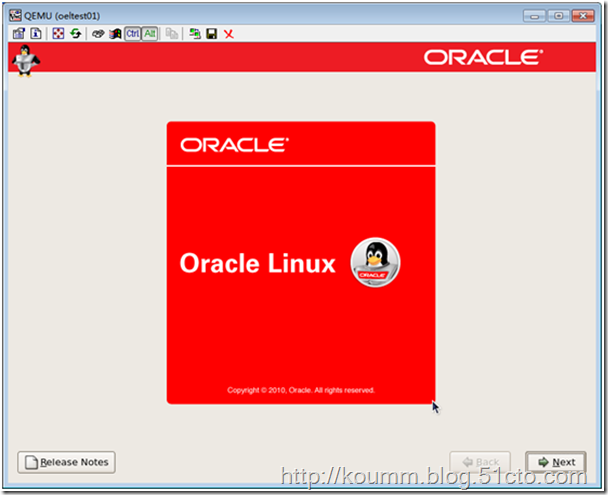
[](http://img1.51cto.com/attachment/201309/5/703525_13783908013km1.png)

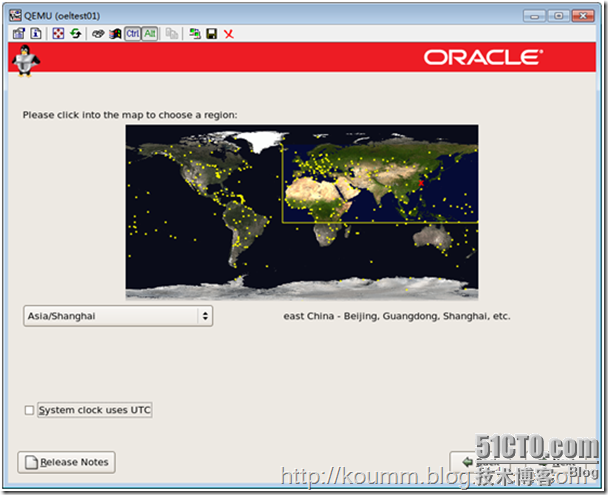
[](http://img1.51cto.com/attachment/201309/5/703525_137839080467Tp.png)

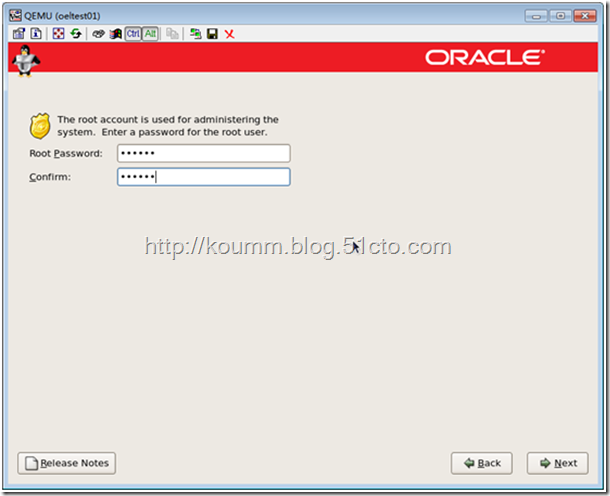
[](http://img1.51cto.com/attachment/201309/5/703525_1378390808wKSh.png)

[](http://img1.51cto.com/attachment/201309/5/703525_1378390815W4mc.png)

[](http://img1.51cto.com/attachment/201309/5/703525_1378390819JE05.png)

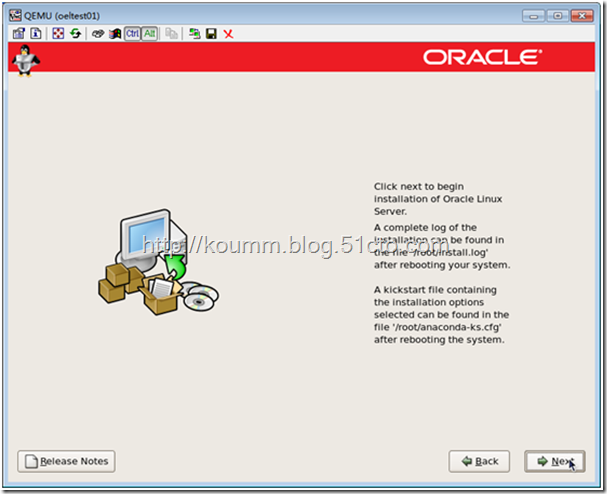
[](http://img1.51cto.com/attachment/201309/5/703525_1378390824Nb7L.png)

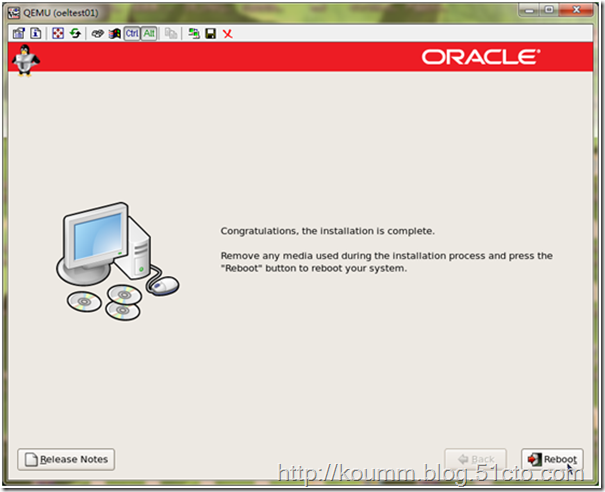
[](http://img1.51cto.com/attachment/201309/5/703525_1378390829K34D.png)

[](http://img1.51cto.com/attachment/201309/5/703525_1378390835LHWu.png)

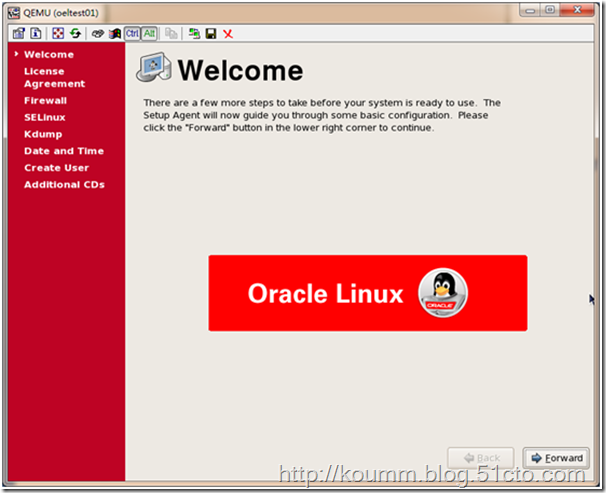
[image](http://img1.51cto.com/attachment/201309/5/703525_1378390838fTJg.png)

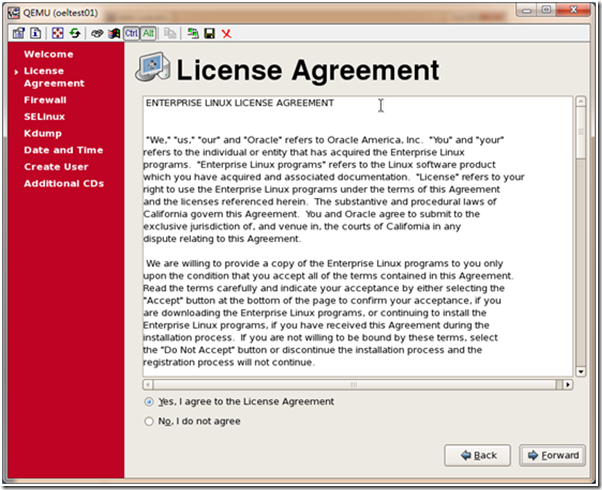
[image](http://img1.51cto.com/attachment/201309/5/703525_1378390843YJxh.png)

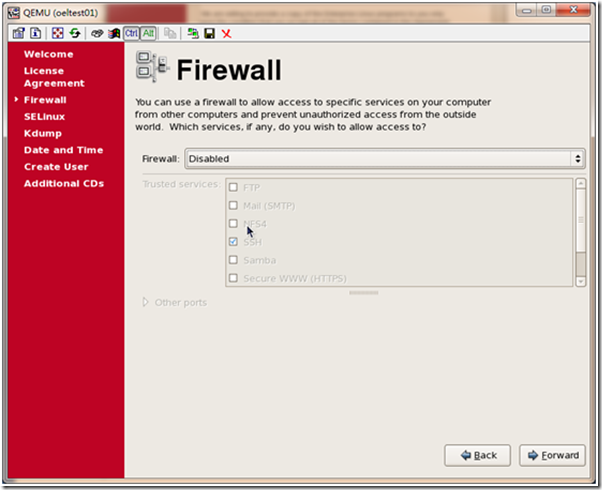
[](http://img1.51cto.com/attachment/201309/5/703525_1378390850Rfvr.png)

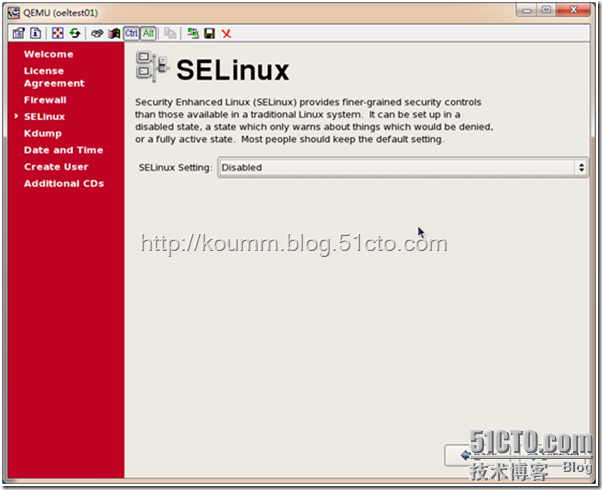
[](http://img1.51cto.com/attachment/201309/5/703525_1378390856OQo6.png)

重启后自动关闭了。

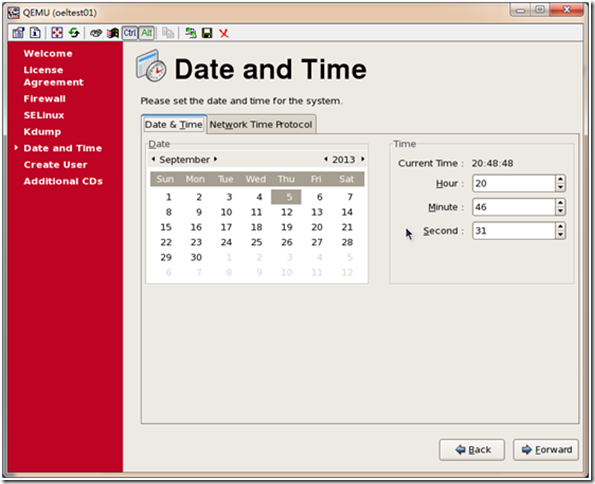
[](http://img1.51cto.com/attachment/201309/5/703525_1378390862YzEu.png)

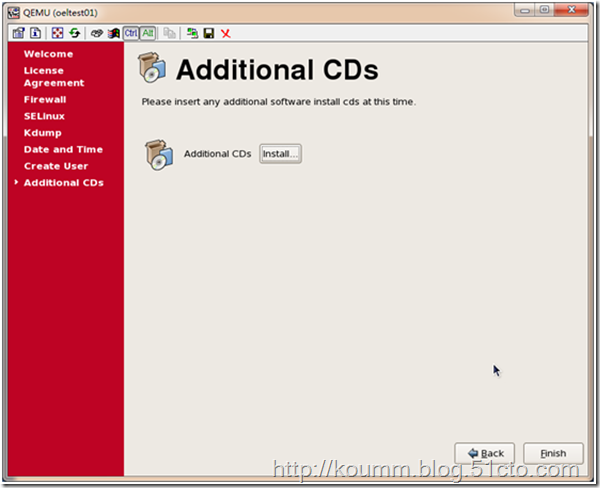
[](http://img1.51cto.com/attachment/201309/5/703525_1378390868wWZk.png)

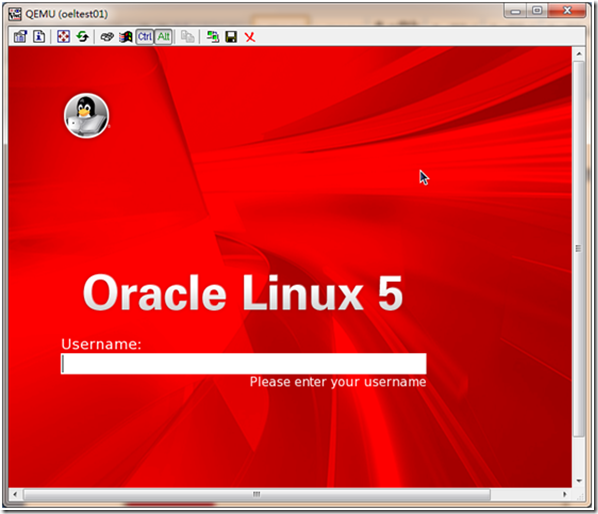
[](http://img1.51cto.com/attachment/201309/5/703525_1378390876lAvN.png)

[](http://img1.51cto.com/attachment/201309/5/703525_1378390881BNTd.png)

[](http://img1.51cto.com/attachment/201309/5/703525_13783908884BXX.png)

[](http://img1.51cto.com/attachment/201309/5/703525_13783908935mLO.png)

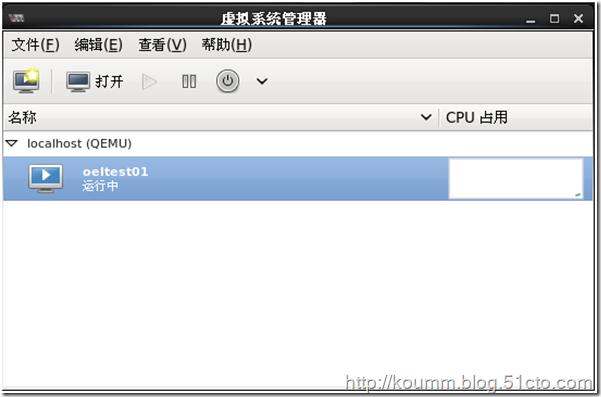
[](http://img1.51cto.com/attachment/201309/5/703525_1378390903xLLp.png)

[](http://img1.51cto.com/attachment/201309/5/703525_1378390910YH95.png)

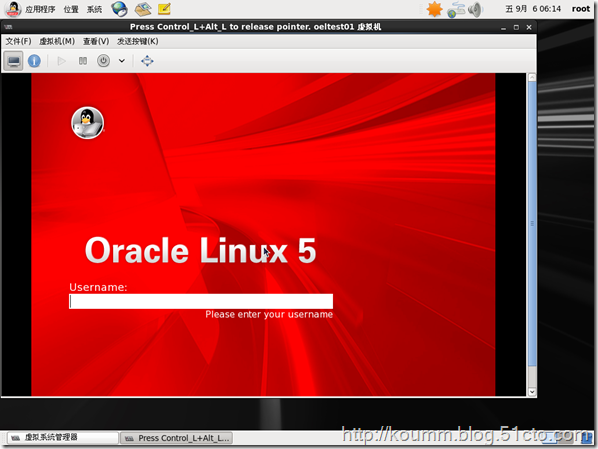
操作系统安装完毕。

### 4.控制台方式登录

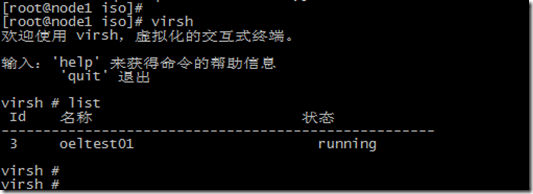
[](http://img1.51cto.com/attachment/201309/5/703525_1378390921QY3q.png)

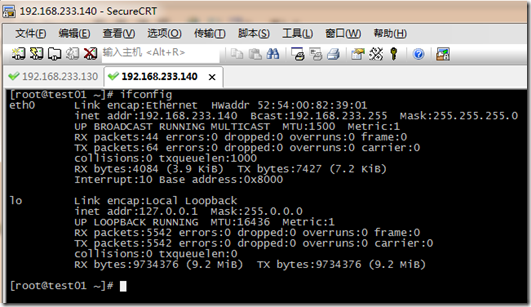
[](http://img1.51cto.com/attachment/201309/5/703525_1378390928nk9I.png)

[](http://img1.51cto.com/attachment/201309/5/703525_1378390931jfzI.png)

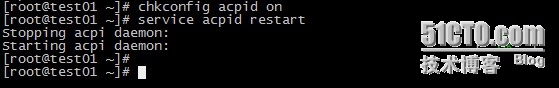
[](http://img1.51cto.com/attachment/201309/5/703525_1378390935JjhC.png)

### 5.登录操作系统查看

[](http://img1.51cto.com/attachment/201309/5/703525_1378390941fa4S.png)

[](http://img1.51cto.com/attachment/201309/5/703525_1378390947ZXRZ.png)

开启电源模式服务

[](http://img1.51cto.com/attachment/201309/214418108.jpg)

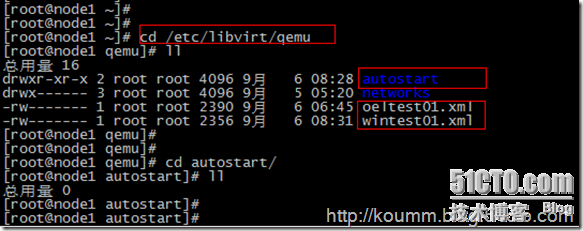
至此OEL5.8X64KVM虚拟机安装完毕。

**kvm虚拟化学习笔记(四)之kvm虚拟机日常管理与配置**

KVM虚拟机的管理主要是通过virsh命令对虚拟机进行管理。

### 1. 查看KVM虚拟机配置文件及运行状态

**(1) KVM虚拟机默认配置文件位置: /etc/libvirt/qemu/**

autostart目录是配置kvm虚拟机开机自启动目录。   
[](http://img1.51cto.com/attachment/201309/6/703525_1378480473E6C5.png)

**(2) virsh命令帮助**

# virsh -help

或直接virsh命令和，再执行子命令。如下所示。

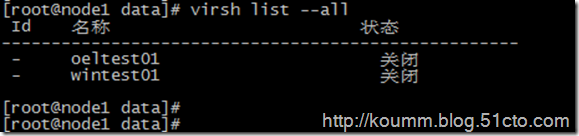
[root@node1 ~]# virsh   
欢迎使用 virsh，虚拟化的交互式终端。

输入：'help' 来获得命令的帮助信息   
'quit' 退出

virsh # help   
……

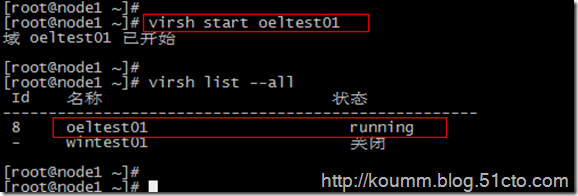
**(3) 查看kvm虚拟机状态**

# virsh list --all

[](http://img1.51cto.com/attachment/201309/6/703525_1378480475NhF9.png)

### 2. KVM虚拟机开机

# virsh start oeltest01

[](http://img1.51cto.com/attachment/201309/6/703525_1378480476mX7K.png)

### 3. KVM虚拟机关机或断电

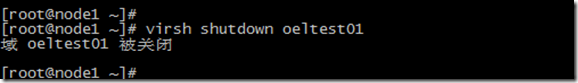
(1) 关机

默认情况下virsh工具不能对linux虚拟机进行关机操作，linux操作系统需要开启与启动acpid服务。在安装KVM linux虚拟机必须配置此服务。

# chkconfig acpid on   
# service acpid restart

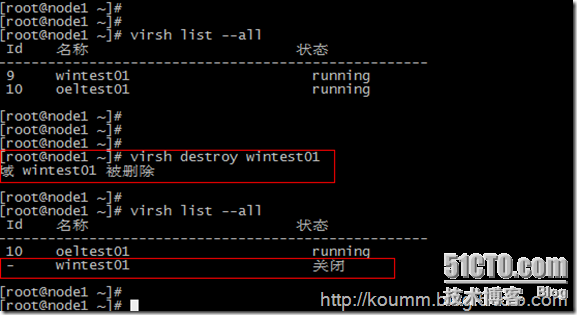
virsh关机

# virsh shutdown oeltest01

[](http://img1.51cto.com/attachment/201309/6/703525_1378480477Gp0U.png)

(2) 强制关闭电源

# virsh destroy wintest01

[](http://img1.51cto.com/attachment/201309/6/703525_13784804782tFD.png)

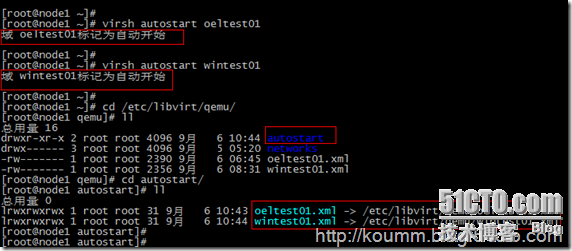
### 4. 通过配置文件启动虚拟机

# virsh create /etc/libvirt/qemu/wintest01.xml

[](http://img1.51cto.com/attachment/201309/6/703525_13784804806HvH.png)

### 5. 配置开机自启动虚拟机

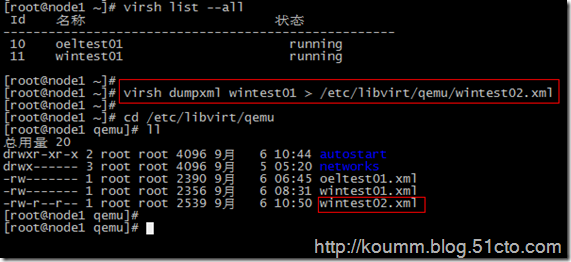
# virsh autostart oeltest01

[](http://img1.51cto.com/attachment/201309/6/703525_1378480484fiVa.png)

autostart目录是kvm虚拟机开机自启动目录，可以看到该目录中有KVM配置文件链接。

### 6. 导出KVM虚拟机配置文件

# virsh dumpxml wintest01 > /etc/libvirt/qemu/wintest02.xml

[](http://img1.51cto.com/attachment/201309/6/703525_1378480487jzlb.png)

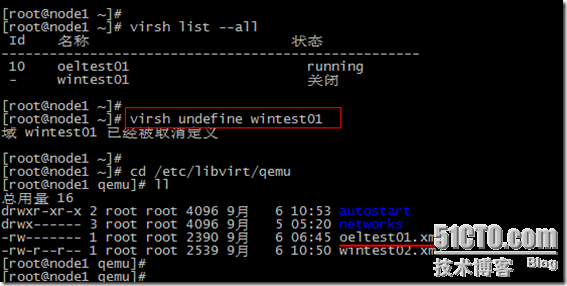
KVM虚拟机配置文件可以通过这种方式进行备份。

### 7. 添加与删除KVM虚拟机

(1) 删除kvm虚拟机

# virsh undefine wintest01

说明：该命令只是删除wintest01的配置文件，并不删除虚拟磁盘文件。如下图所示。

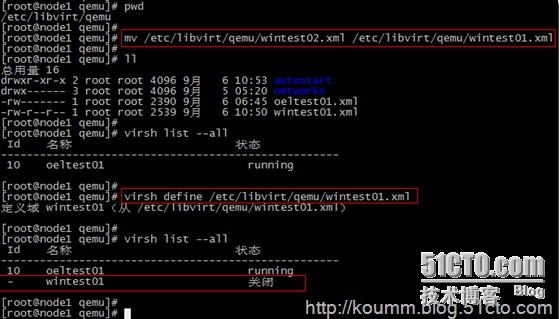
[](http://img1.51cto.com/attachment/201309/6/703525_1378480492b6Ch.png)

(2) 重新定义虚拟机配置文件

通过导出备份的配置文件恢复原KVM虚拟机的定义，并重新定义虚拟机。

# mv /etc/libvirt/qemu/wintest02.xml /etc/libvirt/qemu/wintest01.xml

# virsh define /etc/libvirt/qemu/wintest01.xml

[](http://img1.51cto.com/attachment/201309/231742747.jpg)

### 8. 编辑KVM虚拟机配置文件

# virsh edit wintest01

virsh edit将调用vi命令编辑/etc/libvirt/qemu/wintest01.xml配置文件。也可以直接通过vi命令进行编辑，修改，保存。

**可以但不建议直接通过vi编辑。**

[](http://img1.51cto.com/attachment/201309/6/703525_1378480501gyhF.png)

[root@node1 qemu]# vi /etc/libvirt/qemu/wintest01.xml

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | <!--  WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY TO BE  OVERWRITTEN AND LOST. Changes to this xml configuration should be made using:  virsh edit wintest01  or other application using the libvirt API.  -->  <domain type='kvm'>  <name>wintest01</name>  <uuid>fe31ea48-7d6a-f3cb-cede-2f9bd9dec2bd</uuid>  <memory unit='KiB'>524288</memory>  <currentMemory unit='KiB'>524288</currentMemory>  <vcpu placement='static'>2</vcpu>  <os>  <type arch='x86\_64' machine='rhel6.4.0'>hvm</type>  <boot dev='hd'/>  </os>  <features>  <acpi/>  <apic/>  <pae/>  </features>  <clock offset='utc'/>  <on\_poweroff>destroy</on\_poweroff>  <on\_reboot>restart</on\_reboot>  <on\_crash>restart</on\_crash>  <devices>  <emulator>/usr/libexec/qemu-kvm</emulator>  <disk type='file' device='disk'>  <driver name='qemu' type='raw' cache='none'/>  <source file='/data/wintest01.img'/>  <target dev='hda' bus='ide'/>  <address type='drive' controller='0' bus='0' target='0' unit='0'/>  </disk>  <disk type='block' device='cdrom'>  <driver name='qemu' type='raw'/>  <target dev='hdc' bus='ide'/>  <readonly/>  <address type='drive' controller='0' bus='1' target='0' unit='0'/>  </disk>  <controller type='usb' index='0'>  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/>  </controller>  <controller type='ide' index='0'>  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>  </controller>  <interface type='bridge'>  <mac address='52:54:00:2b:2f:fe'/>  <source bridge='br0'/>  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>  </interface>  <serial type='pty'>  <target port='0'/>  </serial>  <console type='pty'>  <target type='serial' port='0'/>  </console>  <input type='mouse' bus='ps2'/>  <graphics type='vnc' port='5911' autoport='no' listen='0.0.0.0'>  <listen type='address' address='0.0.0.0'/>  </graphics>  <video>  <model type='cirrus' vram='9216' heads='1'/>  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>  </video>  <memballoon model='virtio'>  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>  </memballoon>  </devices>  <seclabel type='none'/>  </domain> |

### 9. virsh console 控制台管理linux虚拟机

**配置virsh console见下文**

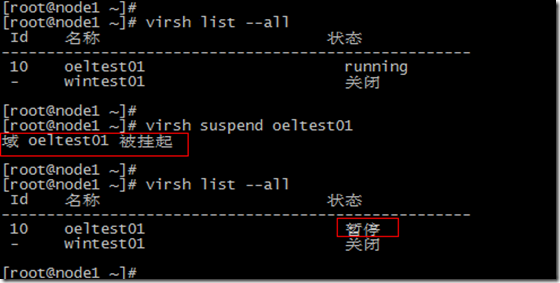
[kvm虚拟化学习笔记(六)之kvm虚拟机控制台登录配置](http://koumm.blog.51cto.com/703525/1290996)

# virsh console oeltest01

### 10. 其它virsh命令

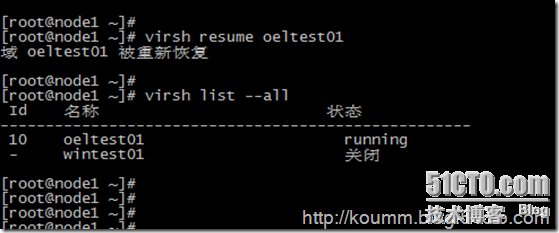
**(1) 挂起服务器**

# virsh suspend oeltest01

[](http://img1.51cto.com/attachment/201309/6/703525_1378480506AjsY.png)

**(2) 恢复服务器**

# virsh resume oeltest01

[](http://img1.51cto.com/attachment/201309/6/703525_1378480508peWv.png)

virsh命令丰富。可以执行各种维护任务，本文只是从维护与管理的角度例举了常用的命令，为该命令的使用提供一个思路。

**kvm克隆**

virt-clone -o f92 -n f7 -f /data/kvm/f7\_centos7.img  
说明：-o 源 -n 目的

**kvm 迁移记录**

1. 在原机器上关闭要迁移的kvm

virsh shutdown 虚拟机名字

1. 把镜像文件和默认的xml配置文件传输到目标机器上

默认的域路劲和xml配置文件路劲 /etc/libvirt/qemu/

1. 在目标机器上配置虚拟桥接网卡br0

cp -pr 目标机器em1配置文件为br0 修改模式为桥接

1. virsh define 命令定义一个域

virsh define /etc/libvirt/qemu/CentOS-193.xml

1. 启动虚拟机

virsh start CentOS-193

**RedHat 7 KVM虚拟机在两台宿主机之间在线迁移**

原创作品，允许转载，转载时请务必以超链接形式标明文章 [原始出处](http://jiangjianlong.blog.51cto.com/3735273/1793913) 、作者信息和本声明。否则将追究法律责任。<http://jiangjianlong.blog.51cto.com/3735273/1793913>

本文主要通过两台RedHat 7 KVM宿主机和NFS共享，将位于共享存储的虚拟机在两台宿主机之间进行在线迁移。

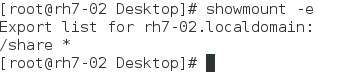
本文原始出处：江健龙的技术博客<http://jiangjianlong.blog.51cto.com/3735273/1793913>

**环境介绍：**

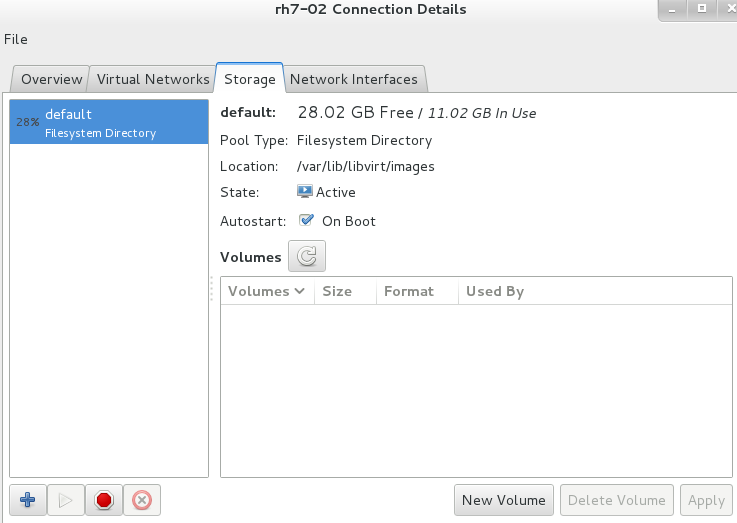
[](http://s4.51cto.com/wyfs02/M01/83/6C/wKioL1dzP5nBZTyDAAAg7-I6d2M497.png)

**配置过程：**

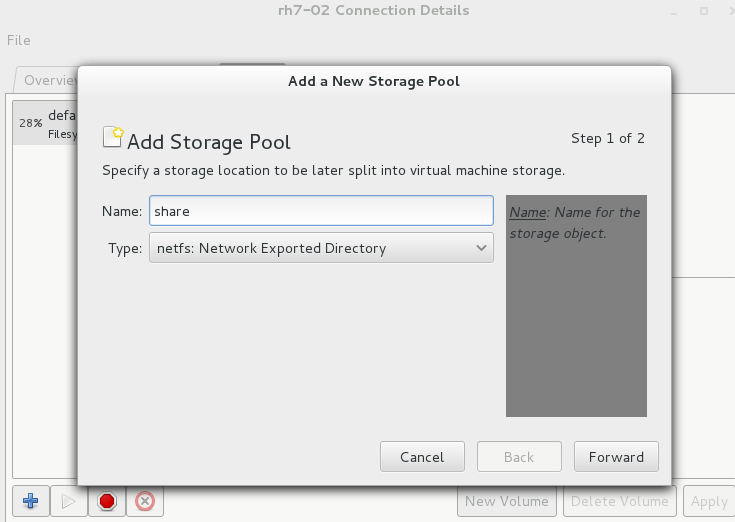
1、在rh7-02上配置NFS服务，共享/share目录出来

[](http://s3.51cto.com/wyfs02/M01/83/53/wKiom1dwwInRNS2lAAAcQx8n3Fk720.png)

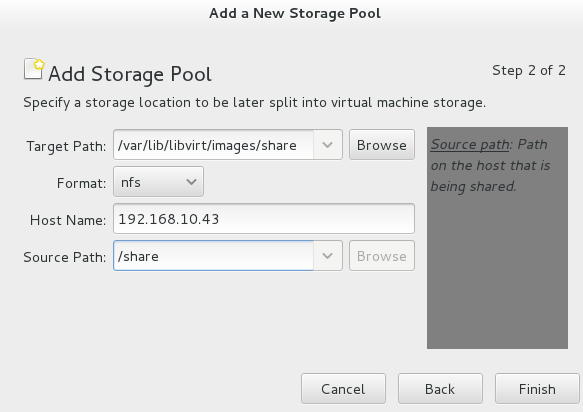
2、在两台宿主机的存储池都添加该NFS共享目录，下面以rh7-02添加存储池为例，点击左下角的+号添加一个存储池

[](http://s4.51cto.com/wyfs02/M02/83/52/wKioL1dwwIrx-O9-AACdZHOyF3k415.png)

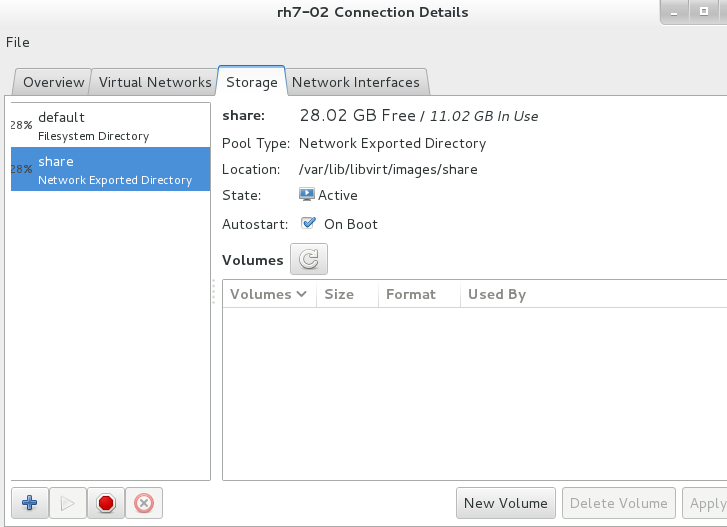
3、输入存储池的名称为share，选择类型为netfs：Network Exported Directory

[](http://s4.51cto.com/wyfs02/M02/83/53/wKiom1dwwIqTViWUAAChjHbyPHo963.png)

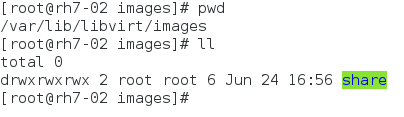
4、填写存储池的相关信息，其中Target Path路径必须两台宿主机都相同，格式必须选择nfs，然后输入NFS服务器的主机名，这里直接填写IP地址，Source Path是指NFS服务器提供的共享目录

[](http://s1.51cto.com/wyfs02/M00/83/52/wKioL1dwwIqQHeM4AAB-viq3hrg389.png)

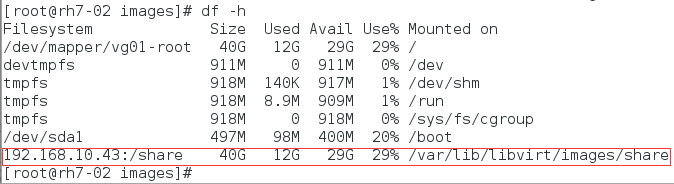
5、完成存储池的添加

[](http://s1.51cto.com/wyfs02/M00/83/53/wKiom1dwwIvzCXIiAACn66CSN-Q523.png)

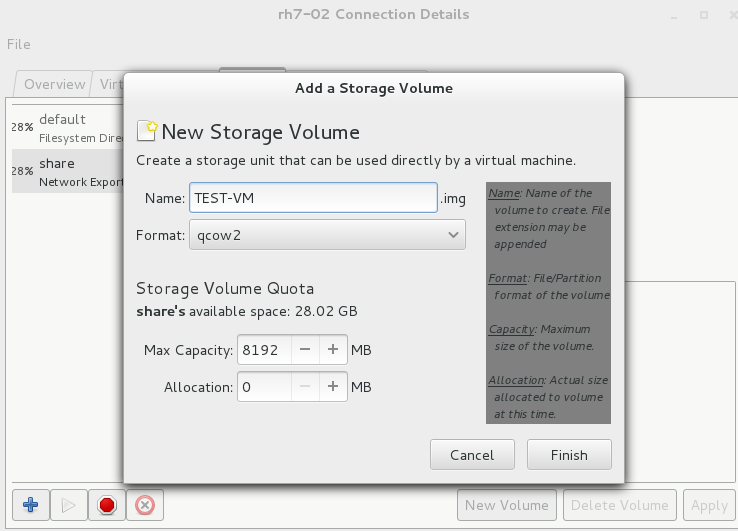
6、创建完存储池后，自动在Target Path路径创建了share子目录

[](http://s5.51cto.com/wyfs02/M01/83/52/wKioL1dwwIvj3j2HAAApIffg2qI095.png)

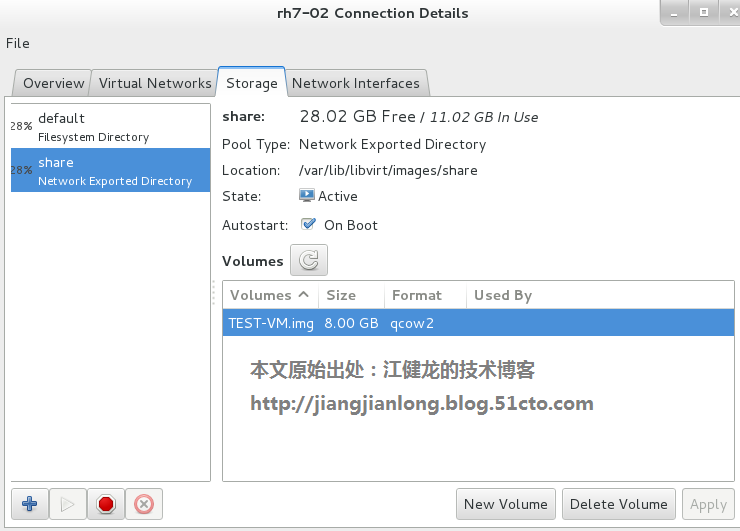
7、创建完存储池后，自动挂载了NFS共享目录到Target Path

[](http://s1.51cto.com/wyfs02/M01/83/53/wKiom1dwwIvyD8b6AABuhaQOJ2U856.png)

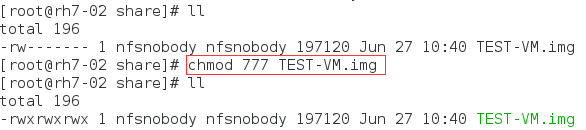
8、创建新的存储卷，点击new volume，输入存储卷的参数，如名称为TEST-VM.img、格式为qcow2、大小为8G的存储卷

[](http://s2.51cto.com/wyfs02/M02/83/52/wKioL1dwwIzBm4h_AADvgLQgJ_4277.png)

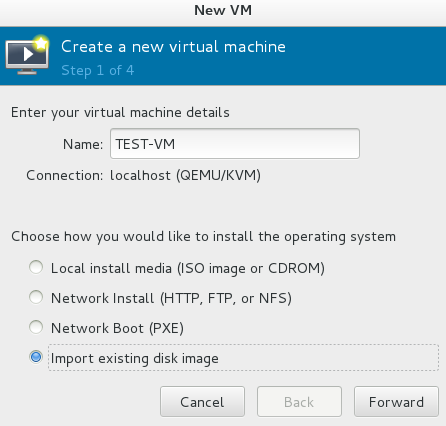
9、完成存储卷的创建，该卷将可以作为虚拟机的硬盘使用

[](http://s5.51cto.com/wyfs02/M01/84/4C/wKiom1eMTLXDhQxJAADQkYhZ6u0952.png)

10、该存储卷创建好后权限很低，无法被虚拟机使用，需要改下权限，直接授予777权限

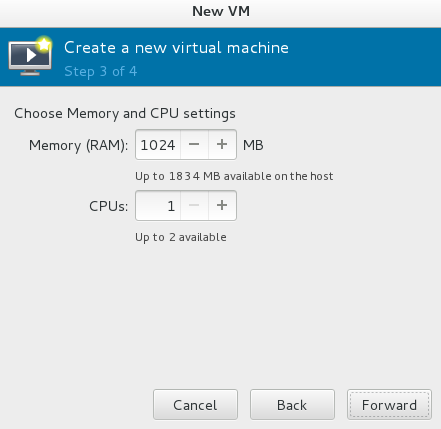
[](http://s3.51cto.com/wyfs02/M01/83/53/wKiom1dwwKbRal60AABDVR1jUo4860.png)

11、创建虚拟机，名称为TEST-VM，选择Import existing disk image

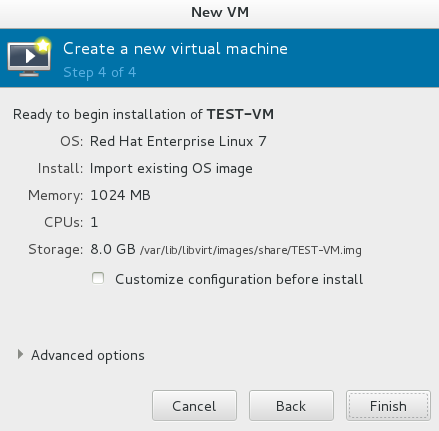
[](http://s2.51cto.com/wyfs02/M00/83/52/wKioL1dwwI2DwB3MAAB_3uFjOKI396.png)

12、导入我们上文创建的存储卷

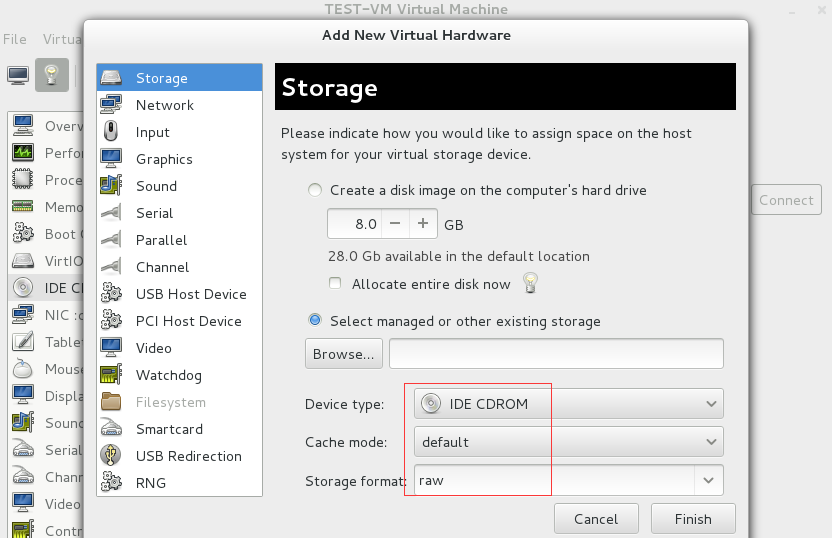
13、为虚拟机选择CPU、内存的配置

[](http://s5.51cto.com/wyfs02/M00/83/53/wKiom1dwwI2ilxeTAABTsr2GzZY123.png)

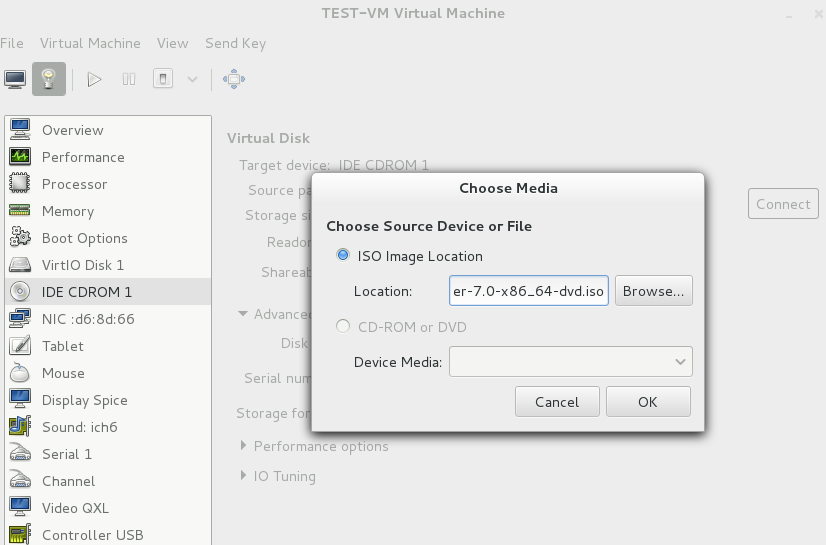
14、完成虚拟机的创建

[](http://s5.51cto.com/wyfs02/M01/83/52/wKioL1dwwI2xkCC4AABzN55R78Y377.png)

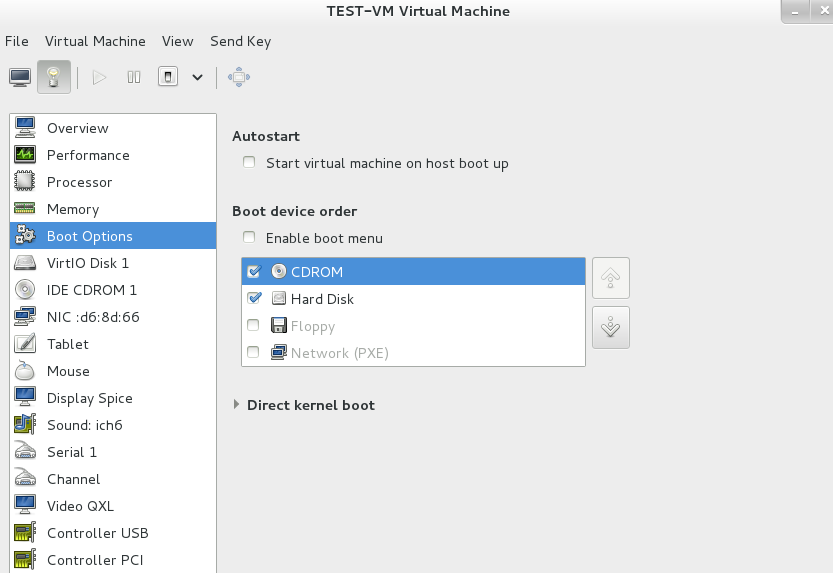
15、调整虚拟机硬件，添加新的硬件，选择存储下的IDE CDROM，用于加载镜像安装操作系统



16、在IDE CDROM1界面下，点击Connect，选择一个redhat 7的安装镜像进行连接

[](http://s3.51cto.com/wyfs02/M02/83/52/wKioL1dwwKezFdM3AAEoQPLUgDY443.png)

17、在boot options界面下，勾选CDROM作为启动首选项，以便开机从光驱引导

[](http://s2.51cto.com/wyfs02/M02/83/53/wKiom1dwwKijukyJAAEIZHGBFto519.png)

18、打开虚拟机电源，安装操作系统，安装完操作系统后，记得要把boot options的CDROM的勾去掉或者将CDROM的优先级降级，确保从Hard Disk引导

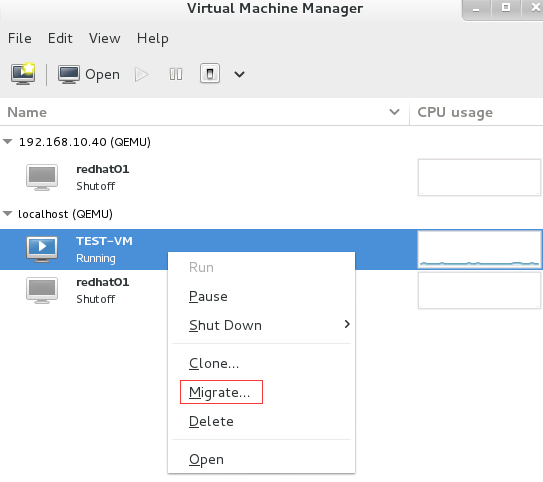
[](http://s2.51cto.com/wyfs02/M00/83/52/wKioL1dwwKiw2V9FAABEpjJuqTI822.png)

[](http://s1.51cto.com/wyfs02/M00/83/53/wKiom1dwwKnDUMahAAGPLAuxH70111.png)

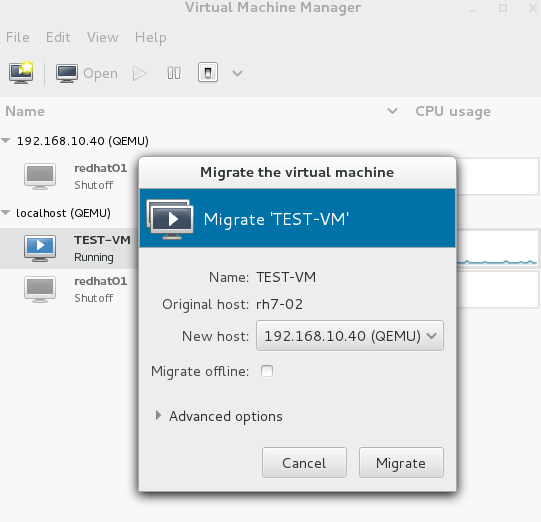
19、另一台宿主机rh7-01的配置：与rh7-02一样，配置NFS存储池。

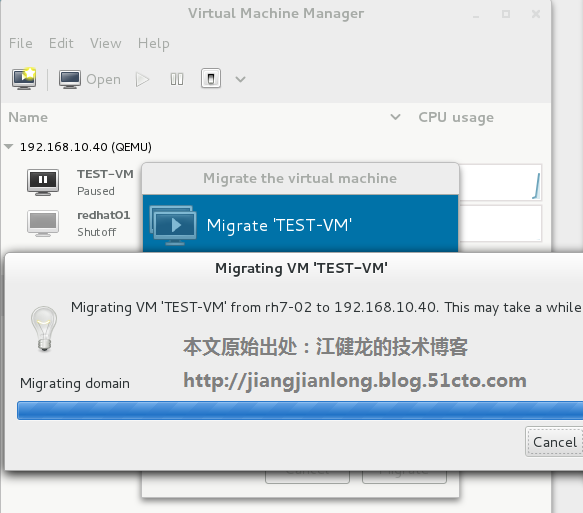
20、域名解析：在两台宿主机的/etc/hosts文件中，分别添加对方的IP和主机名进行hosts解析

21、在rh7-02的virtual machine manager中，连接192.168.10.40（即rh7-01），然后右击localhost下的TEST-VM，选择Migrate

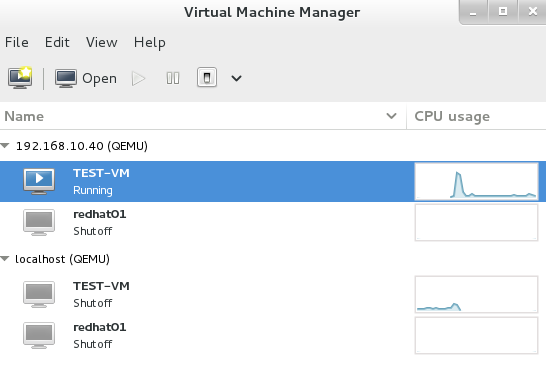
[](http://s1.51cto.com/wyfs02/M00/83/52/wKioL1dwwKuyVEsBAABy_A3skqo224.png)

22、在迁移对话框中，选择New host为192.168.10.40（即rh7-01），点击Migrate即可

[](http://s4.51cto.com/wyfs02/M01/83/52/wKioL1dwwLCShrJWAACtQCU56Iw948.png)

[](http://s2.51cto.com/wyfs02/M01/84/4C/wKioL1eMTIiDh9C0AAC9j7Dignw323.png)

23、只需数秒便已将虚拟机TEST-VM在线迁移到另一台宿主机192.168.10.40（即rh7-01），而rh7-02上原来的TEST-VM已处于关机状态，通过在迁移前对虚拟机进行长ping，丢包情况为丢失0~1个包

[](http://s4.51cto.com/wyfs02/M02/83/52/wKioL1dwwLOhvvtIAABnmqlYxS0029.png)

24、但是，如果反过来将虚拟机TEST-VM从192.168.10.40（即rh7-01）在线迁移回localhost(即rh7-02)，却会丢失二三十个数据包，原因尚未查清楚。

KVM日常管理命令

**一 KVM对用户提供的命令主要在/usr/bin/目录下**

**[root@centoskvm ~]# ls /usr/bin/vir\***

**/usr/bin/virsh /usr/bin/virt-manager**

**/usr/bin/virt-clone /usr/bin/virt-pki-validate**

**/usr/bin/virt-convert /usr/bin/virt-top**

**/usr/bin/virt-host-validate /usr/bin/virt-viewer**

**/usr/bin/virt-image /usr/bin/virt-who**

**/usr/bin/virt-install /usr/bin/virt-xml-validate**

**[root@centoskvm ~]# ls /usr/bin/qe\***

**/usr/bin/qemu-img /usr/bin/qemu-io /usr/bin/qemu-kvm**

**常用的命令有:virsh,virsh-install,virsh-clone,virsh-convert,virsh-image,qemu-img和qemu-kvm**

**二 virsh常用的管理命令**

**使用xml配置文件开启虚拟机**

**[root@centoskvm kvm]# virsh create 1003vm.xml**

**Domain 1003vm created from 1003vm.xml**

**列出虚拟机：**

**[root@centoskvm kvm]# virsh list**

**Id Name State**

**----------------------------------------------------**

**13 1003vm running**

**Id和Name可以作为控制虚拟机的主键**

**列出域中全部的虚拟机：**

**[root@centoskvm kvm]# virsh list --all**

**Id Name State**

**----------------------------------------------------**

**13 1003vm running**

**- 1001vm shut off**

**关闭虚拟机：**

**[root@centoskvm kvm]# virsh shutdown 13**

**或者**

**[root@centoskvm kvm]# virsh shutdown 1003vm**

**重启虚拟机：**

**[root@centoskvm kvm]#virsh reboot 13**

**或者**

**[root@centoskvm kvm]# virsh reboot 1003vm**

**强制关闭虚拟机：（不建议使用，这样容易破坏虚拟机的文件系统）**

**[root@centoskvm kvm]#virsh destroy 13**

**或者**

**[root@centoskvm kvm]# virsh destroy 1003vm**

**添加虚拟机：**

**[root@centoskvm kvm]#virsh define /home/kvm/1002vm.xml #根据主机配置文档添加虚拟机**

**Domain 1002vm defined from /home/kvm/1002vm.xml**

**[root@centoskvm kvm]# virsh list --all**

**Id Name State**

**----------------------------------------------------**

**13 1003vm running**

**- 1001vm shut off**

**- 1002vm shut off #1002vm已被添加到域中**

**移除虚拟机：**

**[root@centoskvm kvm]#virsh undefine 1002vm**

**Domain 1002vm has been undefined**

**[root@centoskvm kvm]# virsh list --all**

**Id Name State**

**----------------------------------------------------**

**13 1003vm running**

**- 1001vm shut off**

**#1002vm已被从域中移除**

**导出虚拟机的配置信息xml文件**

**[root@centoskvm kvm]# virsh dumpxml 1003vm > ~/1003vm.xml**

**[root@centoskvm kvm]# cat ~/1003vm.xml**

**<domaintype='kvm' id='13'>**

**<name>1003vm</name>**

**<uuid>323286df-2fa9-9864-4754-b51df76bd336</uuid>**

**<memory unit='KiB'>2097152</memory>**

**<currentMemory unit='KiB'>2097152</currentMemory>**

**<vcpu placement='static'>2</vcpu>**

**<os>**

**<type arch='x86\_64' machine='rhel6.2.0'>hvm</type>**

**<boot dev='hd'/>**

**</os>**

**<features>**

**<acpi/>**

**<apic/>**

**<pae/>**

**</features>**

**<clock offset='utc'/>**

**<on\_poweroff>destroy</on\_poweroff>**

**<on\_reboot>restart</on\_reboot>**

**<on\_crash>restart</on\_crash>**

**<devices>**

**<emulator>/usr/libexec/qemu-kvm</emulator>**

**<disk type='file' device='disk'>**

**<driver name='qemu' type='raw' cache='none'/>**

**<source file='/home/kvm/1003vm.img'/>**

**<target dev='vda' bus='virtio'/>**

**<shareable/>**

**<alias name='virtio-disk0'/>**

**<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>**

**</disk>**

**<controller type='usb' index='0'>**

**<alias name='usb0'/>**

**<address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/>**

**</controller>**

**<interface type='bridge'>**

**<mac address='52:54:00:88:f8:03'/>**

**<source bridge='br0'/>**

**<target dev='vnet0'/>**

**<model type='virtio'/>**

**<alias name='net0'/>**

**<address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>**

**</interface>**

**<serial type='pty'>**

**<source path='/dev/pts/1'/>**

**<target port='0'/>**

**<alias name='serial0'/>**

**</serial>**

**<console type='pty' tty='/dev/pts/1'>**

**<source path='/dev/pts/1'/>**

**<target type='serial' port='0'/>**

**<alias name='serial0'/>**

**</console>**

**<input type='tablet' bus='usb'>**

**<alias name='input0'/>**

**</input>**

**<input type='mouse' bus='ps2'/>**

**<graphics type='vnc' port='5900' listen='127.0.0.1'>**

**<listen type='address' address='127.0.0.1'/>**

**</graphics>**

**<video>**

**<model type='cirrus' vram='9216' heads='1'/>**

**<alias name='video0'/>**

**<address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>**

**</video>**

**<memballoon model='virtio'>**

**<alias name='balloon0'/>**

**<address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>**

**</memballoon>**

**</devices>**

**<seclabel type='dynamic' model='selinux' relabel='yes'>**

**<label>system\_u:system\_r:svirt\_t:s0:c97,c672</label>**

**<imagelabel>system\_u:object\_r:svirt\_image\_t:s0:c97,c672</imagelabel>**

**</seclabel>**

**</domain>**

**也可以使用xml配置来启动虚拟机：**

**[root@centoskvm kvm]# virsh shutdown 1003vm**

**[root@centoskvm kvm]# virsh list --all**

**Id Name State**

**----------------------------------------------------**

**- 1003vm**  **shut off**

**- 1001vm shut off**

**[root@centoskvm kvm]# virsh create ~/1003vm.xml**

**[root@centoskvm kvm]# virsh list --all**

**Id Name State**

**----------------------------------------------------**

**14 1003vm running**

**- 1001vm shut off**

**三 使用命令接口创建虚拟机**

**创建镜像文件：**

**[root@centoskvm kvm]# qemu-img create -f qcow2 /home/kvm/1001vm.img 16G**

**查看img文件信息：**

[root@centoskvm img]# qemu-img info 1001vm.img

image: 1001vm.img

file format: qcow2

virtual size: 16G (17179869184 bytes)

disk size: 5.9G

cluster\_size: 65536

**其中参数 -f 表示镜像文件的格式[qcow2]，具体介绍如下：**

**[root@centoskvm img]# qemu-img create --help**

create: invalid option -- '-'

qemu-img version 0.12.1, Copyright (c) 2004-2008 Fabrice Bellard

usage: qemu-img command [command options]

QEMU disk image utility

Command syntax:

check [-f fmt] filename

create [-f fmt] [-o options] filename [size]

commit [-f fmt] [-t cache] filename

convert [-c] [-p] [-f fmt] [-t cache] [-O output\_fmt] [-o options] [-S sparse\_size] filename [filename2 [...]] output\_filename

info [-f fmt] filename

snapshot [-l | -a snapshot | -c snapshot | -d snapshot] filename

rebase [-f fmt] [-t cache] [-p] [-u] -b backing\_file [-F backing\_fmt] filename

resize filename [+ | -]size

Command parameters:

'filename' is a disk image filename

'fmt' is the disk image format. It is guessed automatically in most cases

'cache' is the cache mode used to write the output disk image, the valid

options are: 'none', 'writeback' (default), 'writethrough' and 'unsafe'

'size' is the disk image size in bytes. Optional suffixes

'k' or 'K' (kilobyte, 1024), 'M' (megabyte, 1024k), 'G' (gigabyte, 1024M)

and T (terabyte, 1024G) are supported. 'b' is ignored.

'output\_filename' is the destination disk image filename

'output\_fmt' is the destination format

'options' is a comma separated list of format specific options in a

name=value format. Use -o ? for an overview of the options supported by the

used format

'-c' indicates that target image must be compressed (qcow format only)

'-u' enables unsafe rebasing. It is assumed that old and new backing file

match exactly. The image doesn't need a working backing file before

rebasing in this case (useful for renaming the backing file)

'-h' with or without a command shows this help and lists the supported formats

'-p' show progress of command (only certain commands)

'-S' indicates the consecutive number of bytes that must contain only zeros

for qemu-img to create a sparse image during conversion

Parameters to snapshot subcommand:

'snapshot' is the name of the snapshot to create, apply or delete

'-a' applies a snapshot (revert disk to saved state)

'-c' creates a snapshot

'-d' deletes a snapshot

'-l' lists all snapshots in the given image

Supported formats: raw cow qcow vdi vmdk cloop dmg bochs vpc vvfat qcow2 qed parallels nbd blkdebug host\_cdrom host\_floppy host\_device file

常用的格式为raw和qcow2，推荐使用qcow2格式。

raw格式：可以简单、容易地导出到其它模拟器中，但是立即分配占用空间大。

qcow2格式：是qcow格式的升级版本，是目前最万能的格式。使用它可获得较小映像，也是虚拟池一直在使用的镜像格式，支持镜像快照，方便的恢复管理。

安装虚拟机客户操作系统：

[root@centoskvm img]#virt-install --name 1001vm --ram=1024 --arch=x86\_64 --vcpus=2 --check-cpu --os-type=linux --os-variant='rhel6' -c /home/mysoft/os/CentOS-6.2-x86\_64-bin-DVD1.iso --disk path=/home/img/1001vm.img,device=disk,bus=virtio,format=qcow2 --bridge=br0 --noautoconsole --vnc --vncport=5902 --vnclisten=0.0.0.0

[root@centoskvm img]# virt-install --help

Usage: virt-install --name NAME --ram RAM STORAGE INSTALL [options]

Options:

--version show program's version number and exit

-h, --help show this help message and exit

--connect=URI Connect to hypervisor with libvirt URI

General Options:

-n NAME, --name=NAME

Name of the guest instance

-r MEMORY, --ram=MEMORY

Memory to allocate for guest instance in megabytes

--vcpus=VCPUS Number of vcpus to configure for your guest. Ex:

--vcpus 5

--vcpus 5,maxcpus=10

--vcpus sockets=2,cores=4,threads=2

--cpuset=CPUSET Set which physical CPUs domain can use.

--cpu=CPU CPU model and features. Ex: --cpu coreduo,+x2apic

--description=DESCRIPTION

Human readable description of the VM to store in the

generated XML.

--security=SECURITY

Set domain security driver configuration.

--numatune=NUMATUNE

Tune NUMA policy for the domain process.

Installation Method Options:

-c CDROM, --cdrom=CDROM

CD-ROM installation media

-l LOCATION, --location=LOCATION

Installation source (eg, nfs:host:/path,

http://host/path, ftp://host/path)

--pxe Boot from the network using the PXE protocol

--import Build guest around an existing disk image

--init=INIT Path to init binary for container guest. Ex:

--init /path/to/app (to contain an application)

--init /sbin/init (for a full OS container)

--livecd Treat the CD-ROM media as a Live CD

-x EXTRA, --extra-args=EXTRA

Additional arguments to pass to the install kernel

booted from --location

--initrd-inject=INITRD\_INJECTIONS

Add given file to root of initrd from --location

--os-type=DISTRO\_TYPE

The OS type being installed, e.g. 'linux', 'unix',

'windows'

--os-variant=DISTRO\_VARIANT

The OS variant being installed guests, e.g. 'fedora6',

'rhel5', 'solaris10', 'win2k'

--boot=BOOTOPTS Optionally configure post-install boot order, menu,

permanent kernel boot, etc.

Storage Configuration:

--disk=DISKOPTS Specify storage with various options. Ex.

--disk path=/my/existing/disk

--disk path=/my/new/disk,size=5 (in gigabytes)

--disk vol=poolname:volname,device=cdrom,bus=scsi,...

--nodisks Don't set up any disks for the guest.

--filesystem=FILESYSTEMS

Pass host directory to the guest. Ex:

--filesystem /my/source/dir,/dir/in/guest

--filesystem template\_name,/,type=template

Networking Configuration:

-w NETWORK, --network=NETWORK

Configure a guest network interface. Ex:

--network bridge=mybr0

--network network=my\_libvirt\_virtual\_net

--network network=mynet,model=virtio,mac=00:11...

--nonetworks Don't create network interfaces for the guest.

Graphics Configuration:

--graphics=GRAPHICS

Configure guest display settings. Ex:

--graphics vnc

--graphics spice,port=5901,tlsport=5902

--graphics none

--graphics vnc,password=foobar,port=5910,keymap=ja

--noautoconsole Don't automatically try to connect to the guest

console

Device Options:

--serial=SERIALS Configure a guest serial device

--parallel=PARALLELS

Configure a guest parallel device

--channel=CHANNELS Configure a guest communication channel

--console=CONSOLES Configure a text console connection between the guest

and host

--host-device=HOSTDEVS

Configure physical host devices attached to the guest

--soundhw=SOUNDHW Configure guest sound device emulation

--watchdog=WATCHDOG

Configure a guest watchdog device

--video=VIDEO Configure guest video hardware.

--smartcard=SMARTCARD

Configure a guest smartcard device. Ex:

--smartcard mode=passthrough

Virtualization Platform Options:

-v, --hvm This guest should be a fully virtualized guest

-p, --paravirt This guest should be a paravirtualized guest

--container This guest should be a container guest

--virt-type=HV\_TYPE

Hypervisor name to use (kvm, qemu, xen, ...)

--arch=ARCH The CPU architecture to simulate

--machine=MACHINE The machine type to emulate

--noapic Disables APIC for fully virtualized guest (overrides

value in os-type/os-variant db)

--noacpi Disables ACPI for fully virtualized guest (overrides

value in os-type/os-variant db)

-u UUID, --uuid=UUID

UUID for the guest.

Miscellaneous Options:

--autostart Have domain autostart on host boot up.

--print-xml Print the generated domain XML rather than define the

guest.

--print-step=XMLSTEP

Print XML of a specific install step (1, 2, 3, all)

rather than define the guest.

--noreboot Don't boot guest after completing install.

--wait=WAIT Time to wait (in minutes)

--dry-run Run through install process, but do not create devices

or define the guest.

--force Forces 'yes' for any applicable prompts, terminates

for all others

-q, --quiet Suppress non-error output

--prompt Request user input for ambiguous situations or

required options.

-d, --debug Print debugging information

**四 使用console登录虚拟机**

**[root@centoskvm img]#virsh list**

Id Name State

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

13 1003vm running

[root@centoskvm img]# virshconsole 13

**若出现无法从console登录的情况，会卡在“**

Escape character is ^]”处无法继续。请从桌面以root登录做如下操作：

**1、添加ttyS0的安全许可，允许root登录:**

**[root@1001vm img]# echo "ttyS0" >> /etc/securetty**

**[root@1001vm ~]# cat /etc/securetty**

**console**

**vc/1**

**vc/2**

**。。。。。。**

**vc/10**

**vc/11**

**tty1**

**tty2**

**。。。。。。**

**tty11**

**ttyS0**

**2、在/etc/grub.conf文件中为内核添加参数:console=ttyS0**

**[root@1001vm ~]# cat /etc/grub.conf**

**# grub.conf generated by anaconda**

**#**

**# Note that you do not have to rerun grub after making changes to this file**

**# NOTICE: You have a /boot partition. This means that**

**# all kernel and initrd paths are relative to /boot/, eg.**

**# root (hd0,0)**

**# kernel /vmlinuz-version ro root=/dev/mapper/vg\_1001vm-lv\_root**

**# initrd /initrd-[generic-]version.img**

**#boot=/dev/vda**

**default=0**

**timeout=5**

**splashimage=(hd0,0)/grub/splash.xpm.gz**

**hiddenmenu**

**title CentOS (2.6.32-220.el6.x86\_64)**

**root (hd0,0)**

**kernel /vmlinuz-2.6.32-220.el6.x86\_64 ro root=/dev/mapper/vg\_1001vm-lv\_root rd\_NO\_LUKS rd\_LVM\_LV=vg\_1001vm/lv\_root LANG=en\_US.UTF-8 rd\_NO\_MD quiet SYSFONT=latarcyrheb-sun16 rhgb crashkernel=auto rd\_LVM\_LV=vg\_1001vm/lv\_swap KEYBOARDTYPE=pc KEYTABLE=us rd\_NO\_DM console=ttyS0**

**initrd /initramfs-2.6.32-220.el6.x86\_64.img**

**3、在/etc/inittab中添加agetty:**

**S0:12345:respawn:/sbin/agetty ttyS0 115200**

**[root@1001vm ~]# cat /etc/inittab**

**# inittab is only used by upstart for the default runlevel.**

**#。。。。。。。。。。。。。**

**#。。。。。。。。。。。。。**

**id:5:initdefault:**

**S0:12345:respawn:/sbin/agetty ttyS0 115200**

**4、重启客户虚拟机:**

**[root@1001vm img]# reboot**

**此时，再次尝试console命令可以正常使用:**

**[root@centoskvm img]# virsh console 1001vm**

**Connected to domain 1001vm**

**Escape character is ^]**

**[ OK ]**

**Setting hostname 1001vm: [ OK ]**

**Setting up Logical Volume Management: 2 logical volume(s) in volume group "vg\_1001vm" now active**

**[ OK ]**

**。。。。。。。。。。。。。。。。。。。。。。。。。。。。**

**。。。。。。。。。。。。。。。。。。。。。。。。。。。。**

**Starting atd: [ OK ]**

**Starting up CIM server: [ OK ]**

**CentOS release 6.2 (Final)**

**Kernel 2.6.32-220.el6.x86\_64 on an x86\_64**

**1001vm login: root**

**Password:**

**[root@1001vm ~]#**