

目 录

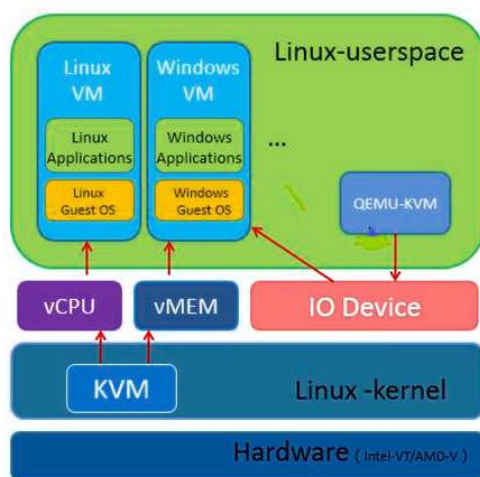
目 录.....	I
第 1 章 KVM 简介及环境.....	1
1.1 KVM 介绍.....	1
1.2 虚拟化概念.....	1
1.3 QEMU 与 KVM.....	2
1.4 Libvirt 与 KVM.....	2
第 2 章 KVM 安装.....	3
2.1 环境准备.....	3
2.1.1 硬件环境.....	3
2.1.2 系统环境.....	4
2.2 创建虚拟机.....	6
2.2.1 查看磁盘空间大小.....	6
2.2.2 上传镜像.....	6
2.2.3 创建磁盘.....	6
2.2.4 安装虚拟机.....	7
2.2.5 VNC 连接创建好的虚拟机并安装系统.....	8
2.2.6 KVM 桥接配置.....	9
第 3 章 KVM 的图形界面管理工具（virt-manager）.....	14

第 1 章 KVM 简介及环境

1.1 KVM 介绍

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。



因为对进程管理比较麻烦,RedHat 发布了一个开源项目 libvirt。libvirt 有命令行工具也有 API，可以通过图形化界面，完成对虚拟机的管理。大多数管理平台通过 libvirt 来完成对 KVM 虚拟机的管理；比如 Openstack、Cloudstack、OpenNebula 等。

1.2 虚拟化概念

1.软件模拟

优点：能够模拟任何硬件，包括不存在的

缺点：功能非常低效，一般用于研究，生产环境不同。

代表：QEM

2.虚拟化层翻译

2.1 软件全虚拟化----VMware

2.2 半虚拟化----改动虚拟机的内核（linux）xen（被淘汰）

2.3 硬件支持的全虚拟化----KVM

3.容器虚拟化 docker

4.虚拟化分类

1.硬件虚拟化 硬件虚拟化代表: KVM

2.软件虚拟化 软件虚拟化代表: Qemu

提示: 硬件虚拟化是需要 CPU 支持, 如果 CPU 不支持将无法创建 KVM 虚拟机。Qemu 和 KVM 的最大区别就是, 如果一台物理机内存直接 4G, 创建一个 vm 虚拟机分配内存分 4G, 在创建一个还可以分 4G。支持超配, 但是 qemu 不支持

1.3 QEMU 与 KVM

QUME 是一个开源项目, 实际就是一台硬件模拟器, 可以模拟许多硬件, 包括 X86 架构处理器、AMD64 架构处理器等。

QEMU 的优点是因为是纯软件模拟, 所以可以在支持的平台模拟支持的设备。缺点是因为纯软件模拟, 所以非常慢。

KVM 只是一个内核模块, 只能提供 CPU 和内存; 所以还需要 QEMU 模拟 IO 设备; 如磁盘、网卡等。

1.4 Libvirt 与 KVM

Libvirt 是一套开源的虚拟化管理工具, 主要由 3 部分组成。

- 一套 API 的 lib 库, 支持主流的编程语言, 包括 C、Python、Ruby 等
- Libvirt 服务
- 命令行工具 virsh

Libvirt 可以实现对虚拟机的管理, 比如虚拟机的创建、启动、关闭、暂停、恢复、迁移、销毁, 以及对虚拟网卡、硬盘、CPU、内存等多种设备的热添加、

第 2 章 KVM 安装

2.1 环境准备

2.1.1 硬件环境

- 首先 bios 需要开启虚拟化。



- 最后虚拟机开启虚拟化的配置。



虚拟化 Intel 使用的是 intel VT-X AMD 使用的是 AMD-V

创建虚拟机步骤

1.准备虚拟机硬盘

-
- 2.需要系统 iso 镜像
 - 3.需要安装一个 vnc 的客户端来连接

2.1.2 系统环境

```
[root@ CentOS7-200 ~]# cat /etc/redhat-release
CentOS Linux release 7.3.1611 (Core)
[root@ CentOS7-200 ~]# uname -r
3.10.0-514.el7.x86_64
[root@ CentOS7-200 ~]# getenforce
Disabled
[root@ CentOS7-200 ~]# systemctl stop firewalld.service
```

- 检查 CPU 是否支持虚拟化

vmx **##(for Intel CPU)**

svm **## (for AMD CPU)**

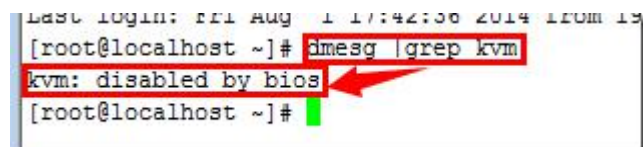
KVM 其实已经在 Centos7 内置到系统内核，无需安装。

```
[root@ CentOS7-200 ~]# egrep -o '(vmx|svm)' /proc/cpuinfo
vmx
[root@ CentOS7-200 ~]# grep -E '(vmx|svm)' /proc/cpuinfo
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts mmx fxsr sse sse2 ss syscall nx rdtscp lm constant_tscarch_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
```

- 检查 CPU 是否开启虚拟化

在 linux 平台下，我们可以通过 **dmesg |grep kvm** 命令来查看。

如果 CPU 没有开启虚拟化的话，显示如下：



```
Last login: Fri Aug 1 17:42:36 2014 from 19
[root@localhost ~]# dmesg |grep kvm
kvm: disabled by bios
[root@localhost ~]#
```

- 安装 kvm 用户态模块

```
[root@ CentOS7-200 ~]# yum list|grep kvm
```

```

qemu-kvm.x86_64                10:1.5.3-141.el7_4.6      @updates
qemu-kvm-common.x86_64         10:1.5.3-141.el7_4.6      @updates
qemu-kvm-tools.x86_64          10:1.5.3-141.el7_4.6      @updates
libvirt-daemon-kvm.x86_64      3.2.0-14.el7_4.9          updates
oci-kvm-hook.x86_64            0.3-1.el7                  epel
pcp-pmda-kvm.x86_64            3.11.8-7.el7               base
[root@ CentOS7-200 ~]# yum install qemu-kvm qemu-kvm-tools libvirt -y

```

libvirt 用来管理 kvm

kvm 属于内核态，不需要安装。但是需要一些类似于依赖的

● 启动 libvirt

```

systemctl start libvirtd.service
systemctl enable libvirtd.service

```

启动之后我们可以使用 ifconfig 进行查看，libvirtd 已经为我们安装了一个桥接网卡

```

[root@ CentOS7-200 ~]# ip a
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN qlen
1000
    link/ether 52:54:00:a5:70:e9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0

```

libvirtd 为我们启动了一个 dnsmasq，这个主要是用来 dhcp 连接的，这个工具会给我们的虚拟机分配 IP 地址

```

[root@ CentOS7-200 ~]# ps -ef|grep dns
nobody      8153      1  0 10:02 ?        00:00:00 /usr/sbin/dnsmasq
--conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro
--dhcp-script=/usr/libexec/libvirt_leaseshelper
root        8154    8153  0 10:02 ?        00:00:00 /usr/sbin/dnsmasq
--conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro
--dhcp-script=/usr/libexec/libvirt_leaseshelper

```

● 关闭 selinux 和防火墙

2.2 创建虚拟机

2.2.1 查看磁盘空间大小

最好是 20G 以上

```
[root@ CentOS7-200 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/cl-root 28G  14G   15G  50% /
devtmpfs        2.0G    0  2.0G   0% /dev
tmpfs           2.0G    0  2.0G   0% /dev/shm
tmpfs           2.0G  8.7M  2.0G   1% /run
tmpfs           2.0G    0  2.0G   0% /sys/fs/cgroup
/dev/sda1       1014M  121M   894M  12% /boot
tmpfs           394M    0   394M   0% /run/user/0
```

2.2.2 上传镜像

提示：如果使用 rz 上传镜像可能会出现错误，所以我们使用 dd 命令，复制系统的镜像。只需要挂载上光盘即可。

```
[root@ CentOS7-200 ~]# cd /opt/
[root@ CentOS7-200 opt]# dd if=/dev/cdrom of=/opt/CentOS-7.3.iso
```

2.2.3 创建磁盘

提示： qemu-img 软件包是我们安装 qemu-kvm-tools 依赖给安装上的

```
[root@ CentOS7-200 opt]# qemu-img create -f qcow2 /opt/CentOS-7.3-x86_64.qcow2 6G

Formatting '/opt/CentOS-7.3-x86_64.qcow2', fmt=qcow2 size=6442450944 encryption=off
cluster_size=65536 lazy_refcounts=off
[root@ CentOS7-200 opt]# ll
total 4277444
-rw-r--r-- 1 root root 4379901952 Apr 24 14:42 CentOS-7.3.iso
-rw-r--r-- 1 root root    197120 Apr 24 14:43 CentOS-7.3-x86_64.qcow2
```

-f 制定虚拟机格式

/opt/Centos 存放路径

6G 代表镜像大小

磁盘格式介绍

raw----裸磁盘不支持快照

qcow2----支持快照。Openstack 使用的方式推荐使用这个。注意：关闭虚拟机后操作。

区别：

全镜像格式（典型代表 raw），特点：设置多大就是多大，写入速度快，方便转换其他格式，性能最优，但是占用空间大。

稀疏格式（典型代表 qcow2），其特点：支持压缩、快照、镜像，更小的存储空间（即用多少占多少）

qcow2 数据的基本组成单元是 cluster

raw 性能比 qcow2 快

raw 创建多大磁盘，就占用多大空间直接分配，qcow2 动态的用多大占用多大空间。

2.2.4 安装虚拟机

```
[root@ CentOS7-200 opt]# yum install -y virt-install
[root@ CentOS7-200 opt]# virt-install --virt-type=kvm --name=c73 --vcpus=1 -r 1024
--cdrom=/opt/CentOS-7.3.iso --network network=default --graphics vnc,listen=0.0.0.0
--noautoconsole --os-type=linux --os-variant=rhel7 --disk
path=/opt/CentOS-7.3-x86_64.qcow2,size=6,format=qcow2
```

默认连接端口是从 5900 开始的

```
[root@ CentOS7-200 opt]# virsh list
Id      Name                                State
-----
 3      c73                                running

[root@ CentOS7-200 opt]# netstat -lntup|grep 5900
tcp      0      0 0.0.0.0:5900          0.0.0.0:*          LISTEN
8440/qemu-kvm
```

virt-install 常用参数

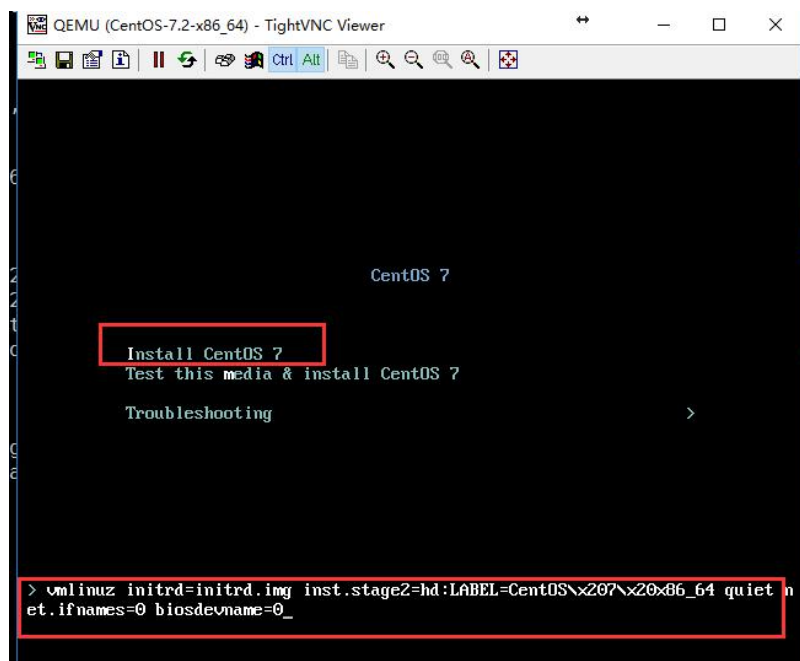
1. -n --name= 客户端虚拟机名称
2. -r --ram= 客户端虚拟机分配的内存
3. -u --uuid= 客户端 UUID 默认不写时，系统会自动生成
4. --vcpus= 客户端的 vcpu 个数
5. -v --hvm 全虚拟化
6. -p --paravirt 半虚拟化

7. `-l --location=localdir` 安装源，有本地、nfs、http、ftp 几种，多用于 ks 网络安装
8. `--vnc` 使用 vnc ， 另有 `--vncclient=` 监听的 IP `--vncport =` VNC 监听的端口
9. `-c --cdrom=` 光驱 安装途径
10. `--disk=` 使用不同选项作为磁盘使用安装介质
11. `-w NETWORK, --network=NETWORK` 连接客户机到主机网络
12. `-s --file-size=` 使用磁盘映像的大小 单位为 GB
13. `-f --file=` 作为磁盘映像使用的文件
14. `--cpuset=` 设置哪个物理 CPU 能够被虚拟机使用
15. `--os-type=OS_TYPE` 针对一类操作系统优化虚拟机配置（例如：‘linux’，‘windows’）
16. `--os-variant=OS_VARIANT` 针对特定操作系统变体（例如‘rhel6’，‘winxp’，‘win2k3’）进一步优化虚拟机配置
17. `--host-device=HOSTDEV` 附加一个物理主机设备到客户机。HOSTDEV 是随着 libvirt 使用的一个节点设备名（具体设备如‘virsh node-dev-list’的显示的结果）
18. `--accelerate KVM 或 KQEMU` 内核加速,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。
19. `-x EXTRA, --extra-args=EXTRA` 当执行从“`--location`”选项指定位置的客户机安装时，附加内核命令行参数到安装程序
20. `--nographics "virt-install"` 将默认使用 `--vnc` 选项，使用 `nographics` 指定没有控制台被分配给客户机

2.2.5 VNC 连接创建好的虚拟机并安装系统



因为 centos7 默认网卡发生改变，我们需要修改内核参数，使用 `eth0` 作为网卡



光标移动到 **Install CentOS** 上，按 **tab** 键 输入 **net.ifnames=0 biosdevname=0** 回车

注意：如果查看 5900 端口开启，但是 VNC 无法连接 KVM 虚拟机时，看下防火墙是否开启。创建的虚拟机用 VNC 连接时从默认端口 5900 开始,即虚拟机一:10.0.0.200:5900 虚拟机二:10.0.0.200:5901

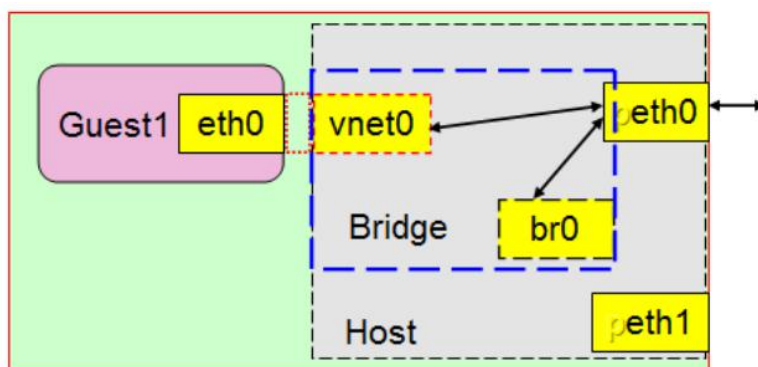
虚拟机安装完成后是关闭了，我们需要启动

```
[root@ CentOS7-200 opt]# virsh list --all
Id      Name                                State
-----
-       c73                                 shut off
[root@ CentOS7-200 opt]# virsh start c73
```

#c73 是虚拟机的名字，是我们创建的时候定义的

2.2.6 KVM 桥接配置

(建议先配置宿主机桥接网络→创建虚拟机)



在该模式下，宿主机虚拟出来一张虚拟网卡作为宿主机本身的通信网卡，而宿主机的物理网卡则成为桥设备（交换机），所以虚拟机相当于在宿主机所在局域网内的一个单独的主机，他的行为和宿主机是同等地位的，没有依存关系。

安装好虚拟化组件(RHEL6.0 之后，系统自带的均是 KVM，已经没有 XEN 虚拟化的支持了)，会自动生成一个 virbr0 这样的桥接设备

```
[root@ CentOS7-200 ~]# brctl show
bridge name      bridge id        STP enabled      interfaces
docker0          8000.0242e20b14dc  no
virbr0           8000.5254005f3794  yes              virbr0-nic
```

Bridge 设备其实就是网桥设备，也就相当于想在的二层交换机，用于连接同一网段内的所有机器，所以我们的目的就是将网络设备 eth0 配置成 br0，此时 br0 就成为了所谓的交换机设备，我们物理机的 eth0 也是连接在上面的。

1. 查看物理机网卡设备信息

```
[root@ CentOS7-200 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:04:53:f6 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.200/24 brd 10.0.0.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::3972:1692:8fab:33d4/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:04:53:00 brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.200/24 brd 172.16.1.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::c754:24e2:801b:163/64 scope link
        valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    link/ether 02:42:e2:0b:14:dc brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN qlen
```

```
1600
```

```
link/ether 52:54:00:5f:37:94 brd ff:ff:ff:ff:ff:ff
inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
    valid_lft forever preferred_lft forever
6: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN
qlen 1000
    link/ether 52:54:00:5f:37:94 brd ff:ff:ff:ff:ff:ff
```

2.配置桥接设备 br0

```
[root@ CentOS7-200 ~]# yum -y install bridge-utils
```

(1) 手动添加临时生效

```
[root@ CentOS7-200 ~]# brctl addbr br0
[root@ CentOS7-200 ~]# brctl show
bridge name      bridge id        STP enabled      interfaces
br0              8000.000000000000  no
docker0          8000.0242e20b14dc  no
virbr0           8000.5254005f3794  yes    virbr0-nic
```

```
[root@ CentOS7-200 ~]# brctl addif br0 eth0
```

执行此步后,会导致 xshell 与宿主机断开连接,以下操作在宿主机完成.

删除 eth0 上面的 ip 地址,将 br0 上面添加上固定 ip 地址:

```
[root@ CentOS7-200 ~]# ip addr del dev eth0 10.0.0.200/24 //删除 eth0 上的 IP 地址
[root@ CentOS7-200 ~]# ifconfig br0 10.0.0.200/24 up //配置 br0 的 IP 地址并启动设备
[root@ CentOS7-200 ~]# route add default gw 10.0.0.254 //重新加入默认网关
```

连接 xshell 查看是否生效

```
[root@ CentOS7-200 ~]# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	10.0.0.254	0.0.0.0	UG	0	0	0	br0
10.0.0.0	0.0.0.0	255.255.255.0	U	0	0	0	br0
172.16.1.0	0.0.0.0	255.255.255.0	U	100	0	0	eth1
172.17.0.0	0.0.0.0	255.255.0.0	U	0	0	0	docker0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

```
[root@ CentOS7-200 ~]# ip a
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
```

```
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master br0 state
UP qlen 1000
    link/ether 00:0c:29:04:53:f6 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::3972:1692:8fab:33d4/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:04:53:00 brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.200/24 brd 172.16.1.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::c754:24e2:801b:163/64 scope link
        valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    link/ether 02:42:e2:0b:14:dc brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN qlen
1000
    link/ether 52:54:00:5f:37:94 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid_lft forever preferred_lft forever
6: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN
qlen 1000
    link/ether 52:54:00:5f:37:94 brd ff:ff:ff:ff:ff:ff
8: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP qlen 1000
    link/ether 00:0c:29:04:53:f6 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.200/24 brd 10.0.0.255 scope global br0
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe04:53f6/64 scope link
        valid_lft forever preferred_lft forever
[root@ CentOS7-200 ~]# ping www.baidu.com
PING www.a.shifen.com (61.135.169.121) 56(84) bytes of data.
64 bytes from 61.135.169.121 (61.135.169.121): icmp_seq=1 ttl=128 time=4.95 ms
64 bytes from 61.135.169.121 (61.135.169.121): icmp_seq=2 ttl=128 time=4.19 ms
64 bytes from 61.135.169.121 (61.135.169.121): icmp_seq=3 ttl=128 time=6.30 ms
^C
--- www.a.shifen.com ping statistics ---
```

```
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 4.194/5.151/6.301/0.872 ms
```

此时宿主机的 ip:10.0.0.200 已经绑定到 br0 网卡;但是服务器重启后就不能生效。

(2) 通过配置文件配置桥接设备永久生效

为 KVM 宿主机创建虚拟网卡，并将物理网卡作为桥设备

```
[root@ CentOS7-200 ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0 .
[root@ CentOS7-200 opt]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
BRIDGE=br0
NM_CONTROLLED=no
[root@ CentOS7-200 opt]# cat /etc/sysconfig/network-scripts/ifcfg-br0
DEVICE=br0
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.0.0.200
NETMASK=255.255.255.0
GATEWAY=10.0.0.254
NM_CONTROLLED=no
[root@ CentOS7-200 opt]# systemctl restart network.service
```

通过 VNC 连接 KVM 虚拟机修改网卡配置文件

```
[root@ CentOS7-200 ~]# virsh list --all
Id      Name                                State
-----
-       c73                                 shut off
[root@ CentOS7-200 ~]# systemctl stop firewalld.service

[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
TYPE=Ethernet
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.0.0.100
PREFIX=24
```

```
GATEWAY=10.0.0.254
```

```
DNS1=223.5.5.5      此处配置后则不需要手动添加/etc/resolv.conf
```

```
DNS2=1.1.1.1
```

```
[root@localhost ~]# cat /etc/resolv.conf  #必须有否则 xshell 连不上
```

```
nameserver 223.5.5.5
```

```
[root@localhost ~]# ifup eth0
```

注意：此时宿主机还需要通过图形化工具设置网卡为桥接方式，否则无法 ping 通网关和外网。

第 3 章 KVM 的图形界面管理工具（virt-manager）

virt-manager 是用于管理 KVM 虚拟环境的主要工具，virt-manager 默认设置下需要使用 root 用户才能够使用该工具。当你想在 KVM hypervisor 服务器上托管虚拟机，由最终用户而非 root 用户访问这些虚拟机时并不总是很便利。

virt-manager 可以设置本机，同样也可以连接远程宿主机来管理。

利用 virt-manager、xmanager、xshell 启动界面来管理虚拟机,适合管理单机的 KVM.

1.首先查看本机 sshd 是否开启 X11 转发

```
[root@ CentOS7-200 ~]# grep X11Forwarding /etc/ssh/sshd_config --colour
```

```
X11Forwarding yes
```

```
# X11Forwarding no
```

2.安装 xorg-x11

```
yum install -y xorg-x11-font-utils.x86_64 xorg-x11-server-utils.x86_64
```

```
xorg-x11-utils.x86_64 xorg-x11-xauth.x86_64 xorg-x11-xinit.x86_64
```

```
xorg-x11-drv-ati-firmware
```

3. 安装 libvirt

libvirt 是管理虚拟机的 API 库，不仅支持 KVM 虚拟机，也可以管理 Xen 等方案下的虚拟机。

```
[root@ CentOS7-200 ~]# yum install virt-manager libvirt libvirt-Python python-virtinst
```

```
libvirt-client virt-viewer qemu-kvm mesa-libglapi -y
```

因为我的主机是服务器，没有图形化界面，想要用 virt-manager 图形化安装虚拟机，还需要安装 X-window。

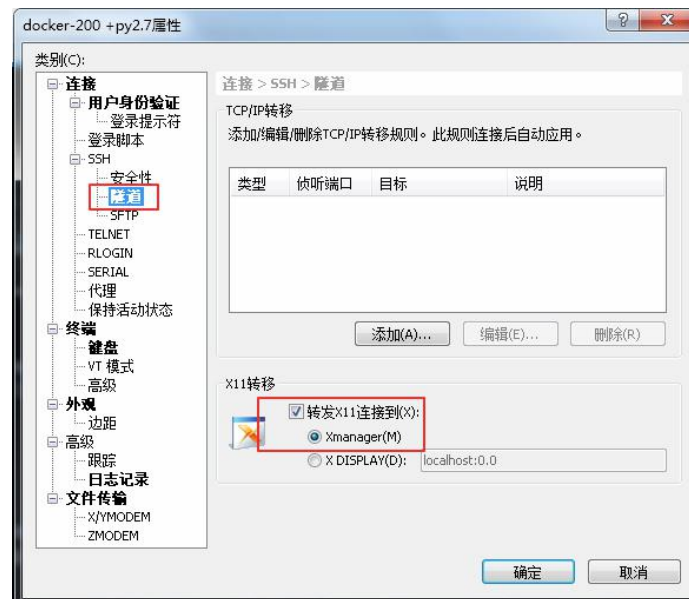
```
[root@ CentOS7-200 ~]# yum install libXdmcp libXmu libxkbfile xkeyboard-config
```

```
xorg-x11-xauth xorg-x11-xkb-utils -y
```

开启 libvirt 服务

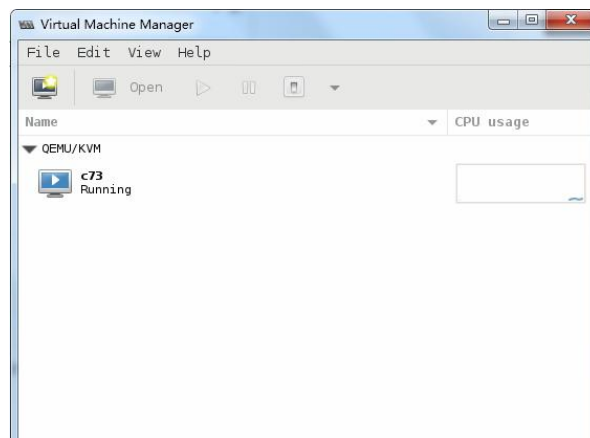
```
systemctl start libvirtd.service  
systemctl enable libvirtd.service
```

4. 安装好 xmanager 后, 打开 xshell, 在连接属性的 tunnelling 中, 勾选 Forwarding X11 connection to 选项, 可以正常打开 virt-manager 的图形界面。



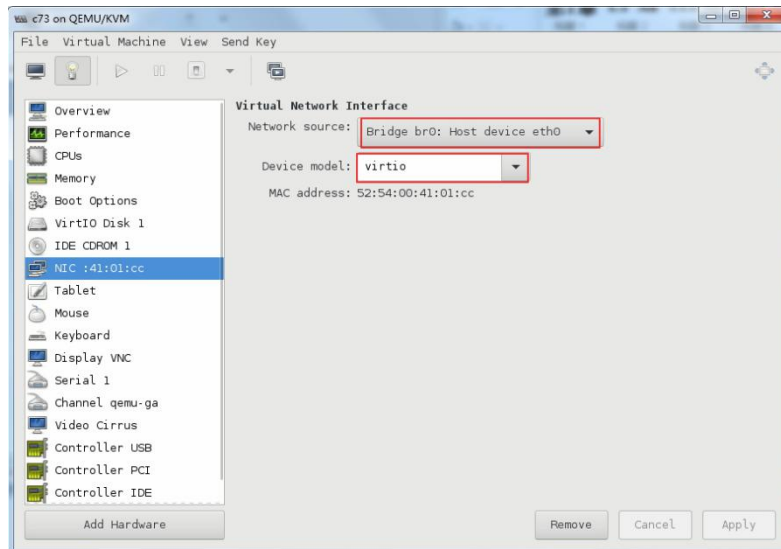
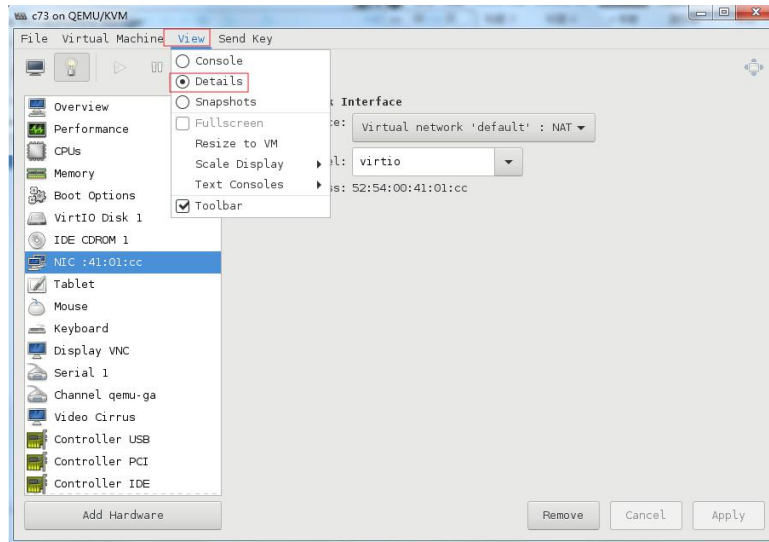
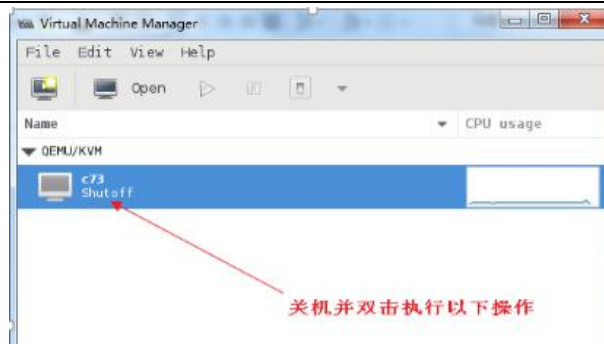
6. 启动 virt-manager

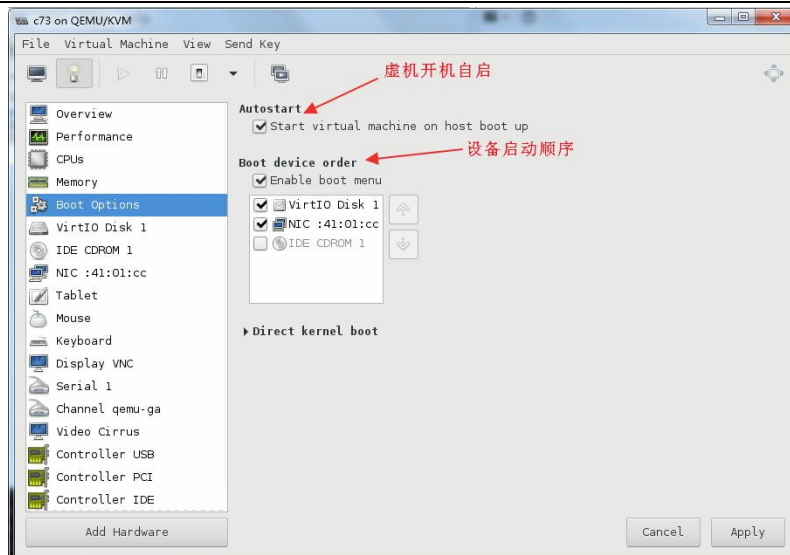
```
[root@ CentOS7-200 ~]# virt-manager
```



出现乱码, 请安装以下包

```
yum install dejavu-sans-mono-fonts -y
```



第 4 章 kvm 虚拟化管理平台 WebVirtMgr

当 KVM 宿主机越来越多，需要对宿主机的状态进行调控，决定采用 WebVirtMgr 作为 kvm 虚拟化的 web 管理工具，图形化的 WEB，让人能更方便的查看 kvm 宿主机的情况 and 操作

WebVirtMgr 是近两年来发展较快，比较活跃，非常清新的一个 KVM 管理平台，提供对宿主机和虚机的统一管理，它有别于 kvm 自带的图形管理工具（virtual machine manager），让 kvm 管理变得更为可视化，对中小型 kvm 应用场景带来了更多方便。

WebVirtMgr 采用几乎纯 Python 开发，其前端是基于 Python 的 Django，后端是基于 Libvirt 的 Python 接口，将日常 kvm 的管理操作变的更加的可视化。

WebVirtMgr 特点:

操作简单，易于使用

通过 libvirt 的 API 接口对 kvm 进行管理

提供对虚拟机生命周期管理

WebVirtMgr 功能

宿主机管理支持以下功能:

CPU 利用率

内存利用率

网络资源池管理

存储资源池管理

虚拟机镜像

虚拟机克隆

快照管理

日志管理

虚机迁移

虚拟机管理支持以下功能:

CPU 利用率

内存利用率

光盘管理

关/开/暂停虚拟机

安装虚拟机

VNC console 连接

创建快照

这里我将 webvirtmgr 服务器和 kvm 服务器放在同一台机器上部署的，即单机部署.
