## **kvm虚拟机热迁移**

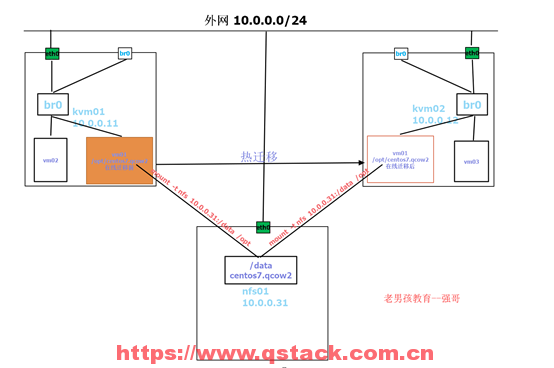
热迁移描述：

相比KVM虚拟机冷迁移中需要拷贝虚拟机虚拟磁盘文件，kvm虚拟机热迁移无需拷贝虚拟磁盘文件，但是需要迁移到的宿主机之间需要有相同的目录结构虚拟机磁盘文件，也就是共享存储，本文这部分内容通过nfs来实现，当然也可以采用Glusterfs集群文件系统来实现.

热迁移流程：

在kvm01上挂起虚拟机vm01，发送vm的虚拟机配置文件和运行时内存中的数据到kvm02, 接受完毕，kvm02恢复vm01,热迁移完成。

架构图如下：



环境要求：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 主机名 | ip | 内存 | 网络 | 软件需求 | 虚拟化 |
| kvm01 | 10.0.0.11 | 2G | 创建br0桥接网卡 | kvm和nfs | 开启虚拟化 |
| kvm02 | 10.0.0.12 | 2G | 创建br0桥接网卡 | kvm和nfs | 开启虚拟化 |
| nfs01 | 10.0.0.31 | 1G | 无 | nfs | 无 |

注意：需要互相做好host解析

操作步骤：

一：在kvm01和kvm02上安装kvm和nfs,配置桥接网卡

yum install libvirt\* virt-\* qemu-kvm\* nfs-utils openssh-askpass -y

systemctl start libvirtd.service

virsh iface-bridge eth0 br0

二：在nfs01上安装配置nfs

yum install nfs-utils -y

mkdir /data

vim /etc/exports

/data 10.0.0.0/24(rw,async,no\_root\_squash,no\_all\_squash)

systemctl restart rpcbind

systemctl restart nfs

三：kvm01和kvm02挂载共享目录/opt

mount -t nfs 10.0.0.31:/data /opt

四：安装一台基于桥接模式的虚拟机

virt-install --virt-type kvm --os-type=linux --os-variant rhel7 --name web04 --memory 512,maxmemory=2048 --vcpus 1 --disk /data/web04.qcow2 --boot hd --network bridge=br0 --graphics vnc,listen=0.0.0.0 --noautoconsole

热迁移的命令：

virsh migrate --live --verbose oldboy qemu+ssh://10.0.0.12/system --unsafe

将宿主机10.0.0.11上的kvm虚拟机oldboy迁移到10.0.0.12

五：在kvm01上安装图形界面、vnc服务端和virt-manager

yum groups install "GNOME Desktop" -y

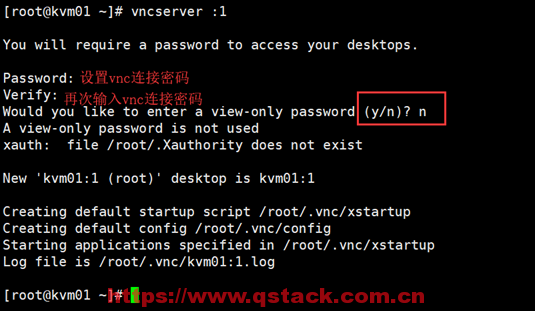
yum install tigervnc-server.x86\_64 -y

yum install virt-manager -y

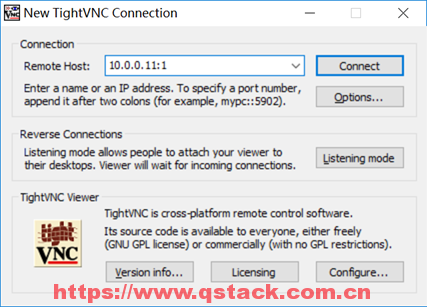
六：启动vnc服务端

vncserver :1 启动5901端口的vnc服务端

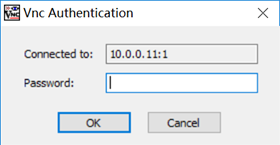
vncserver -kill :1 关闭5901端口的vnc服务端

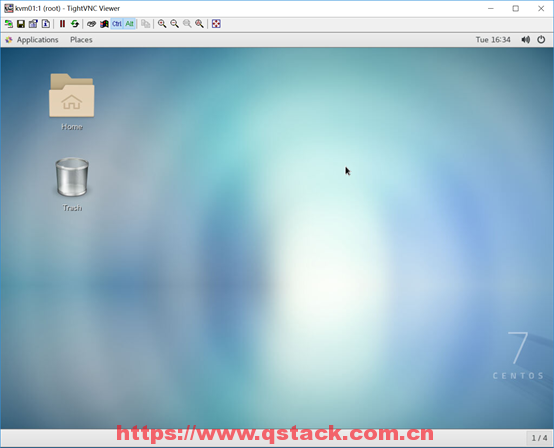


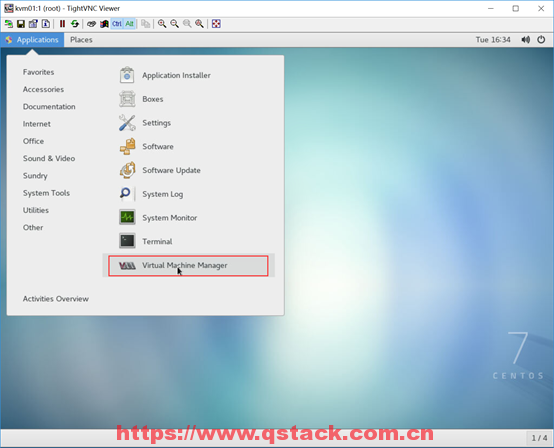
七：使用vnc连接宿主机，使用virt-manager进行迁移

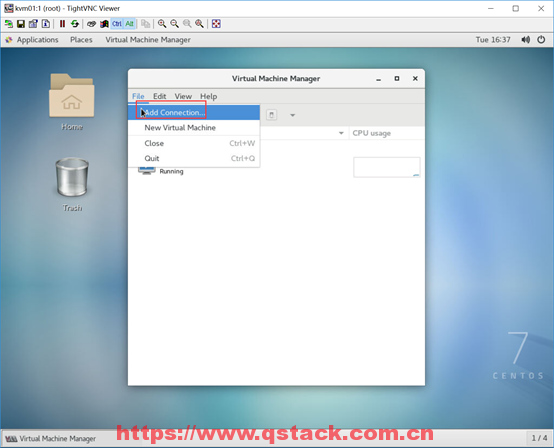


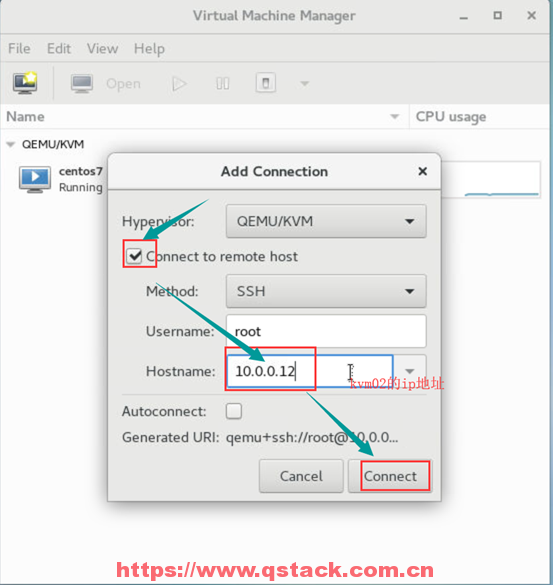
这时候会提醒输入密码，就是之前第6步的时候设置的vnc连接密码

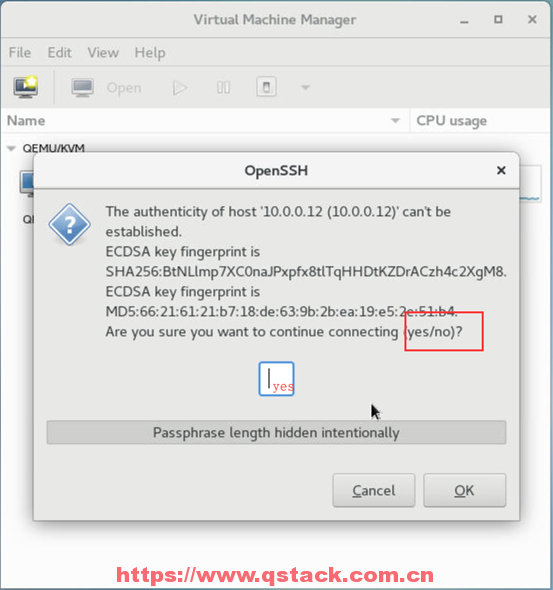


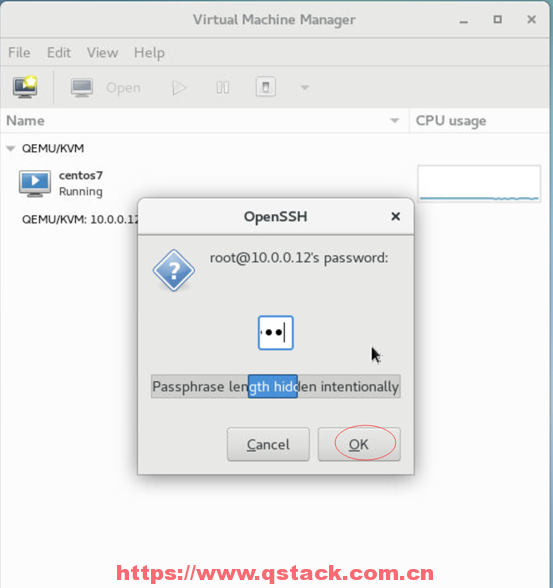


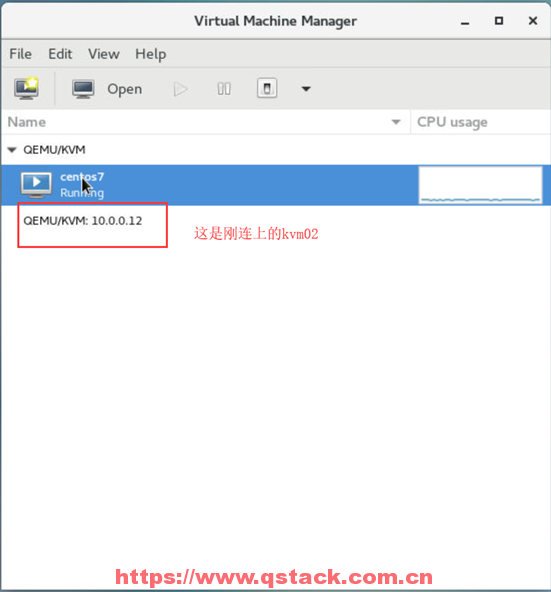


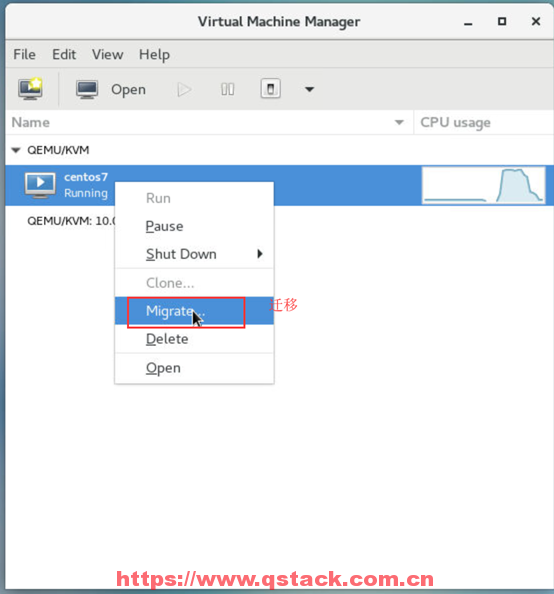


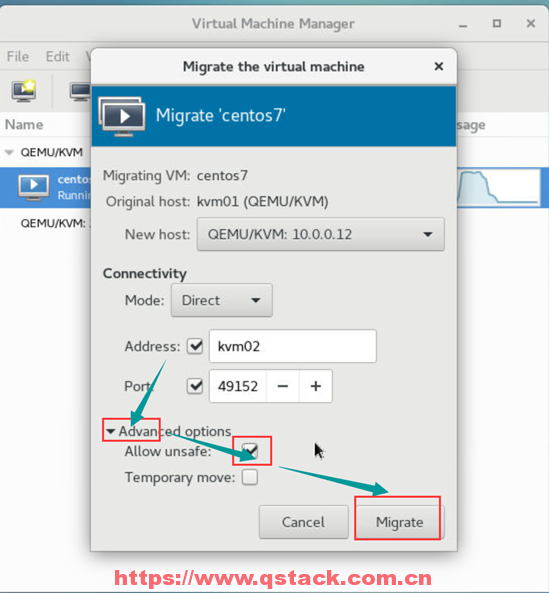


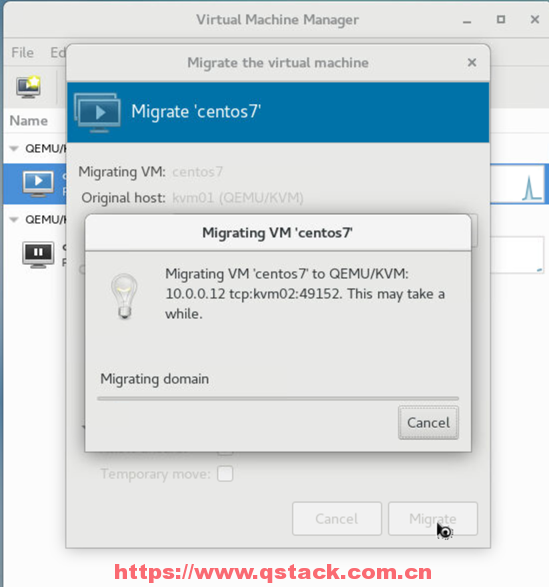




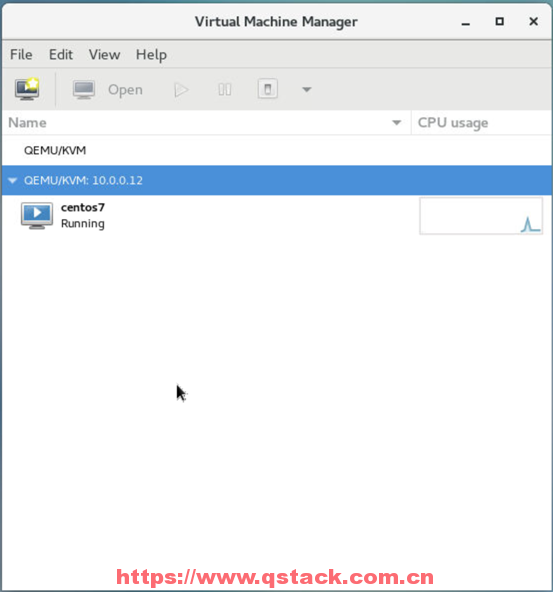




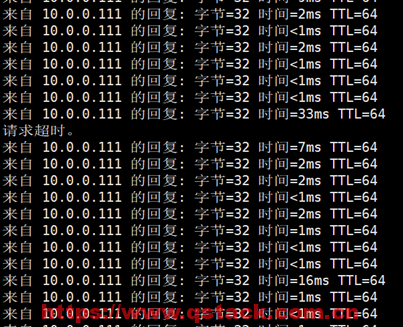




迁移成功！



在迁移的过程中，使用ping虚拟机的ip，发现只丢了一个包



至此热迁移完成！

## **全自动链接克隆KVM虚拟机**

virt-clone这个命令是基于全克隆的，也就是拷贝虚拟磁盘文件和虚拟配置文件来实现的完整克隆，速度慢，占用空间多

kvm软件包中并没有实现全自动链接克隆的命令或工具，只能手动实现，于是我决定写一个脚本来搞定kvm虚拟机的自动链接克隆

想自动，先手动

手动克隆：

第一步：创建基于链接克隆的虚拟磁盘文件

第二步：修改xml配置文件

1）修改name虚拟机名称

2）删除uuid，导入之后会自动生成新的uuid

3）修改虚拟磁盘文件存储路径

4）删除mac地址，导入之后会自动生成新的mac地址

5）删除channel部分中带有source路径的行，如<source mode='bind' path='/var/lib/libvirt/qemu/channel/target/domain-node1/org.qemu.guest\_agent.0'/>

防止出现以下报错：

qemu-kvm: -chardev socket,id=charchannel0,path=/var/lib/libvirt/qemu/channel/target/domain-vm1/org.qemu.guest\_agent.0,server,nowait: Failed to bind socket: Permission denied

2016-10-20T02:20:11.713371Z qemu-kvm: -chardev socket,id=charchannel0,path=/var/lib/libvirt/qemu/channel/target/domain-vm1/org.qemu.guest\_agent.0,server,nowait: chardev: opening backend "socket" failed

第三步：导入要被克隆的虚拟机

第四步：测试启动

下面献上比较简陋的自动化脚本，看不下去的可自行完善。

#!/bin/bash

old\_vm=$1

new\_vm=$2

new\_xml="/tmp/${new\_vm}.xml"

virsh dumpxml $old\_vm >$new\_xml

old\_disk=`grep qcow2 $new\_xml|awk -F "'" '/source file/{print $2}'`

tmp\_dir=`dirname $old\_disk`

new\_disk=${tmp\_dir}/${new\_vm}.qcow2

#1.创建基于链接克隆的虚拟磁盘文件

qemu-img create -f qcow2 -b $old\_disk $new\_disk

#2.修改xml配置文件

sed -i '/uuid/d' $new\_xml

sed -i '/mac address/d' $new\_xml

sed -i '2s#'$old\_vm'#'$new\_vm'#' $new\_xml

sed -i 's#'$old\_disk'#'$new\_disk'#g' $new\_xml

sed -i '/\/var\/lib\/libvirt\/qemu\/channel\/target/d' $new\_xml

#3.导入要被克隆的虚拟机

virsh define $new\_xml

#4.测试启动

virsh start $new\_vm

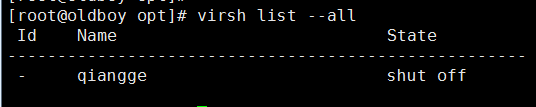
## **Centos7虚拟机根分区扩展**

2018-01-05 分类：[虚拟化](https://oldqiang.com/archives/category/kvm) 阅读(1506) 评论(0)

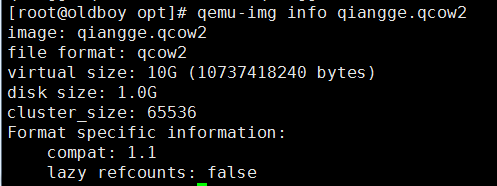
线上的kvm虚拟机，原来只规划了8G，后来发现硬盘动不动就被日志塞满了，需要进行扩容。

扩容步骤如下：

### 1、先把kvm虚拟机关机

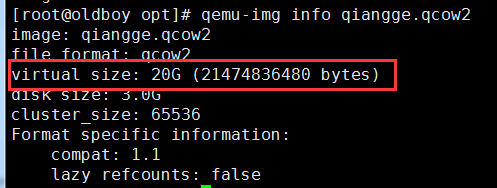


### 2、在宿主机上进行kvm虚拟机的磁盘扩容

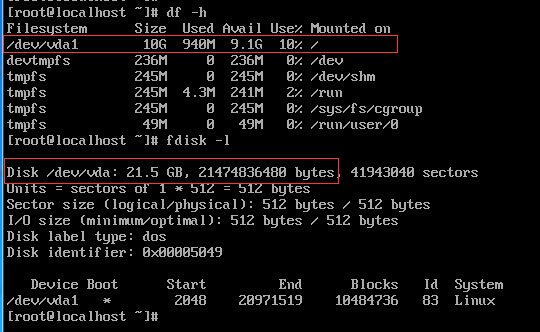


qemu-img resize qiangge.qcow2 +10G

调整之后

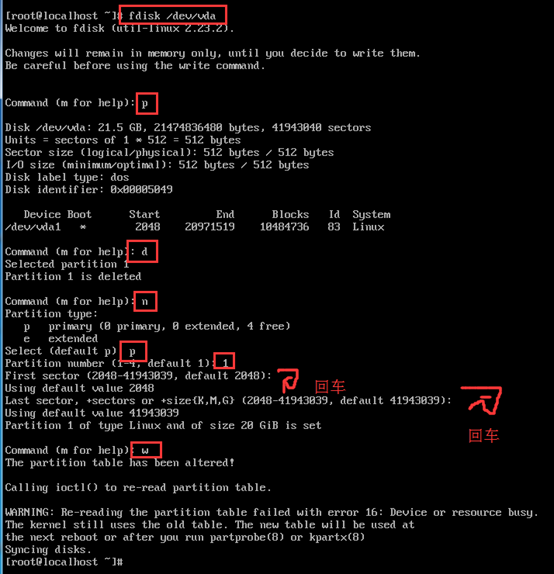


接下来进入到虚拟机发现没有生效！！！



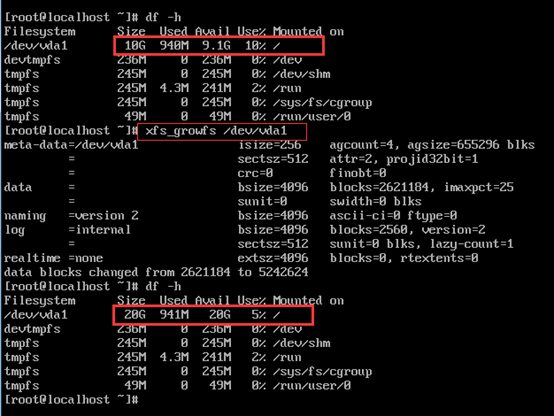
### 3、在虚拟机中调整分区

由于分区表没有更新所以没有生效，接下来我们还需要调整分区表（核心是调整根分区的起始扇区的位置）



### 4、重启虚拟机，完成根分区扩容

重启reboot之后执行xfs\_growfs /dev/vda1，再次查看，分区扩容成功！



本文适合xfs的文件系统，如果是ext4的文件系统，请将xfs\_growfs /dev/vda1的命令替换为resize2fs /dev/vda1。

## **构建docker基本镜像**

2017-11-03 分类：[虚拟化](https://oldqiang.com/archives/category/kvm) 阅读(1738) 评论(0)

###### 准备：

创建一个目录oldboy-hello

###### 编写Dockerfile

内容如下，只有三行

FROM scratch

ADD hello /

CMD ["/hello"]

当前目录下的文件

[root@linux-node1 oldboy-hello]# ls

Dockerfile hello

###### 构建docker镜像

[root@linux-node1 oldboy-hello]# docker build -t oldboyedu/hello .

Sending build context to Docker daemon 4.608kB

Step 1/3 : FROM scratch

--->

Step 2/3 : ADD hello /

---> 0018e580ccaf

Removing intermediate container 64b74f2071e1

Step 3/3 : CMD /hello

---> Running in 6438314dc49d

---> 4a047322f97d

Removing intermediate container 6438314dc49d

Successfully built 4a047322f97d

Successfully tagged oldboyedu/hello:latest

###### 启动容器

[root@linux-node1 oldboy-hello]# docker run --rm -it oldboyedu/hello

Welcome oldboy education!

###### 补充

hello的是使用C语言编译之后的二进制程序

[root@linux-node1 opt]# cat hello.c

//#include <unistd.h>

#include <sys/syscall.h>

#ifndef DOCKER\_IMAGE

#define DOCKER\_IMAGE "hello-world"

#endif

#ifndef DOCKER\_GREETING

#define DOCKER\_GREETING "Welcome oldboy education!"

#endif

const char message[] =

DOCKER\_GREETING "\n";

void \_start() {

//write(1, message, sizeof(message) - 1);

syscall(SYS\_write, 1, message, sizeof(message) - 1);

//\_exit(0);

syscall(SYS\_exit, 0);

}

编译命令gcc -o hello -static -nostartfiles hello.c

测试执行

[root@linux-node1 oldboy-hello]# ./hello

Welcome oldboy education!

如果编译出错

[root@linux-node1 ~]# gcc -o hello -static -nostartfiles hello.c

/usr/bin/ld: cannot find -lc

collect2: error: ld returned 1 exit status

安装glibc-static

yum install glibc-static gcc -y

## **KVM和Docker的对比**

2017-11-02 分类：[虚拟化](https://oldqiang.com/archives/category/kvm) 阅读(2633) 评论(0)

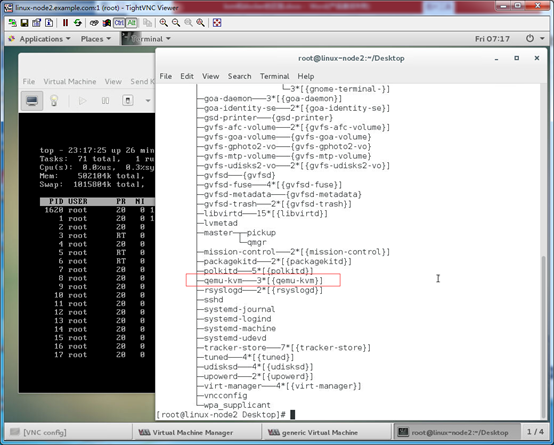
###### 虚拟化技术对比：

KVM：全虚拟化，需要模拟各种硬件

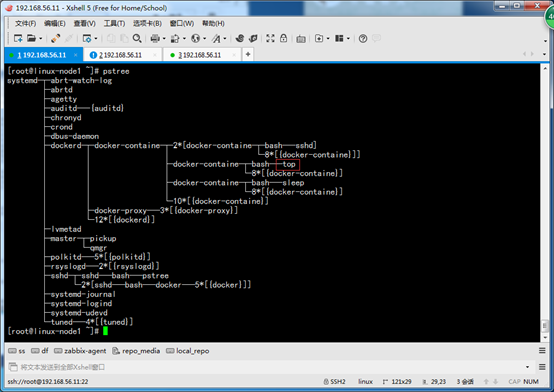
docker：严格来说不算是虚拟化技术，只是进程隔离和资源限制

###### 实例启动进程对比：

在kvm虚拟机中执行top命令，看宿主机进程树，根本看不到top的进程



在docker容器中，同样执行top命令，在宿主机进程数中直接可以看到top进程



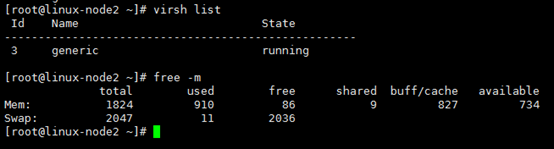
###### 占用内容资源对比：

KVM虚拟机一般会独占一段内存，即使闲置，其他虚拟机也无法使用。而容器可以只有一个内存上限，没有下限。如果它只使用1MB内存，那么它只占用宿主机1MB内存。宿主机可以将富余内存作为他用。

docker：启动两个实例，才占用一点

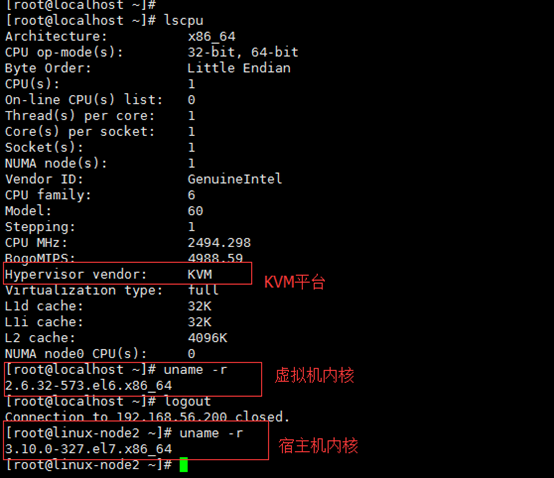


kvm：启动一个512M内存的实例，就占了这么多了



###### 实例的内核对比

kvm的虚拟机内核无需与宿主机一致



docker：实例内核和宿主机内核一致



###### 操作系统支持度对比：

kvm：支持多种操作系统，除了linux，还支持windows，uninx、solaris等

docker：只支持linux

###### 启动一个实例需要的时间对比

lxc容器技术将操作系统抽象到了一个新的高度。直接从init启动，省去了硬件自检、grub引导、加载内核、加载驱动等传统启动项目，因此启动飞速。

docker：秒级

kvm：分钟级别

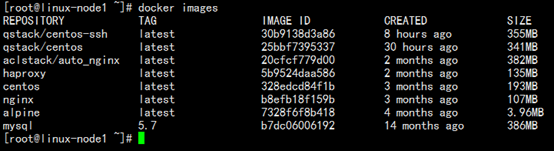
###### 镜像模板占用空间对比

kvm：占用空间大

[root@linux-node2 ~]# du -smh /var/lib/libvirt/images/generic.qcow2

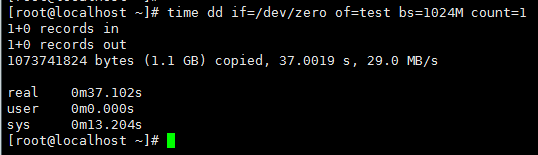
1.9G    /var/lib/libvirt/images/generic.qcow2

docker：占用空间小

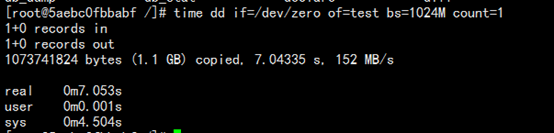


###### 磁盘读写性能对比：

kvm

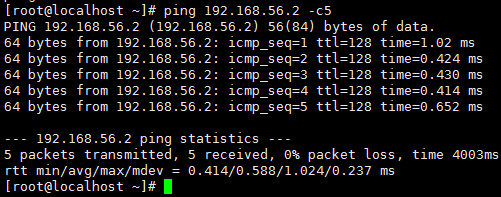


docker

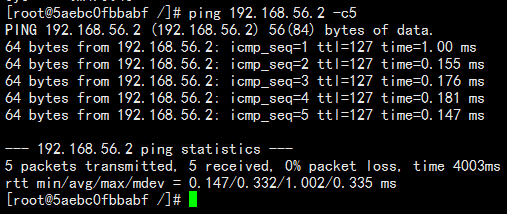


###### 网络性能测试对比

kvm



docker



综合来说：docker比kvm更省资源，可以提供非常接近宿主机的性能，而kvm资源隔离比docker更高，kvm支持的操作系统类型更多

## **在centos7上kvm网卡桥接**

系统环境准备

[root@linux-node1 ~]# cat /etc/redhat-release

CentOS Linux release 7.2.1511 (Core)

[root@linux-node1 ~]# sestatus

SELinux status: disabled

[root@linux-node1 ~]# systemctl stop firewalld.service

最小化安装系统时勾选3个包组"Compatibility libraries" "Base" "Development tools"

host解析

[root@linux-node1 ~]# cat /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.56.11 linux-node1

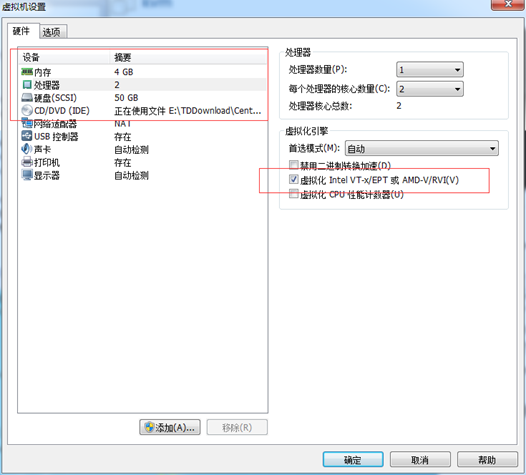
安装常用软件

yum install -y net-tools vim lrzsz tree screen lsof tcpdump wget

安装epel源

rpm -ivh http://mirrors.aliyun.com/epel/epel-release-latest-7.noarch.rpm

硬件环境，建议内存多给点，一定要开启虚拟化，使用vmware workstation12版本，如下图



安装

[root@linux-node1 ~]# yum install -y qemu-kvm qemu-kvm-tools libvirt virt-install

启动

[root@linux-node1 ~]# systemctl start libvirtd.service

注意：如果前面没做host解析，启动的时候，可能会报错

设置开机启动

[root@linux-node1 ~]# systemctl enable libvirtd.service

[root@linux-node1 ~]# cd /etc/sysconfig/network-scripts/

[root@linux-node1 network-scripts]# cp ifcfg-eth0 ifcfg-br0

[root@linux-node1 network-scripts]# vim ifcfg-eth0

[root@linux-node1 network-scripts]# vim ifcfg-br0

修改结果如下：

[root@linux-node1 network-scripts]# cat ifcfg-eth0

TYPE=Ethernet

BOOTPROTO=none

DEVICE=eth0

ONBOOT=yes

BRIDGE=br0

[root@linux-node1 network-scripts]# cat ifcfg-br0

TYPE=Bridge

BOOTPROTO=static

DEVICE=br0

ONBOOT=yes

IPADDR=192.168.56.11

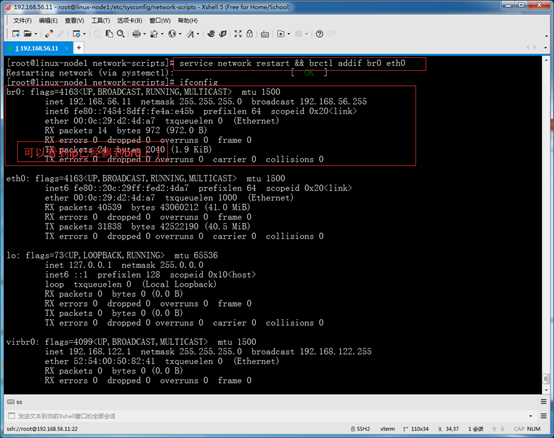
NETMASK=255.255.255.0

GATEWAY=192.168.56.2

DNS1=192.168.56.2

DNS2=223.5.5.5

[root@linux-node1 network-scripts]# service network restart



[root@linux-node1 network-scripts]# cd /opt/

上传镜像

[root@linux-node1 opt]# ls

CentOS-6.7-x86\_64-bin-DVD1.iso

我这里上传的是centos6.7

创建虚拟磁盘

[root@linux-node1 opt]# qemu-img create -f raw /opt/vm1.raw 20G

创建虚拟机

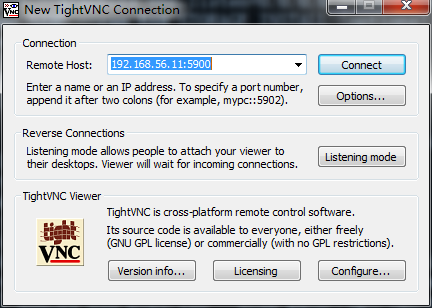
virt-install --virt-type kvm --name vm1 --ram 1024 \

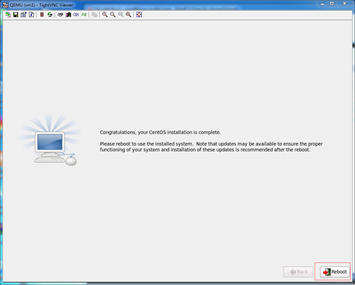
--cdrom=/opt/CentOS-6.7-x86\_64-bin-DVD1.iso --disk path=/opt/vm1.raw \

--network network=default --graphics vnc,listen=0.0.0.0 --noautoconsole



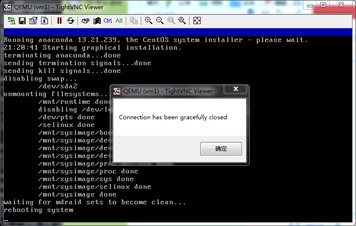
打开vnc



后面用图形化安装centos6.7的系统，就不啰嗦了

点击reboot之后

vnc会断掉



回到宿主机

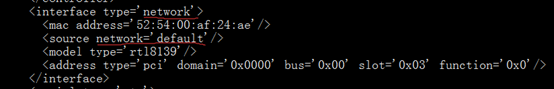
[root@linux-node1 opt]# virsh list --all

查看我们刚创建好的虚拟机

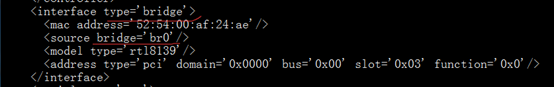
编译虚拟机

[root@linux-node1 opt]# virsh edit vm1

将下图的type='network'和network='default'分别修改为type='bridge'和bridge='br0'



修改之后的效果

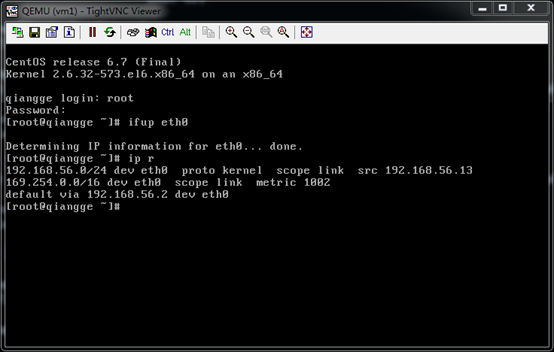


启动虚拟机

[root@linux-node1 opt]# virsh start vm1

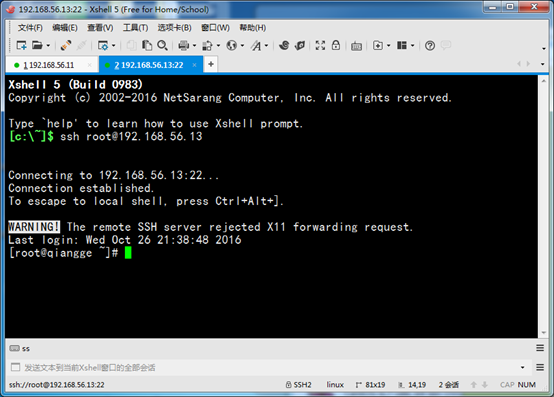
Domain vm1 started

继续使用vnc连接



在这里我的dhcp是打开的，所以，虚拟机能直接获取到ip

在使用xshell ssh连接刚才看到的虚拟机的ip



如果没有dhcp的话，也可以手动配置ifcfg-eth0后重启network都可以的

到这里kvm桥接就完成了！！！