# 一．虚拟机基础

## （一）虚拟化分类

### 1. Host是通过什么方式将硬件资源虚拟化，并提供给Guest使用的呢？

通过一个Hypervisor的程序，根据Hypervisor的实现方式和所处的位置，虚拟化又分为两种：1型虚拟化和2型虚拟化。

### 2. 1型虚拟化

Hypervisor直接安装在物理机上，多个虚拟机在Hypervisor上运行，这里的Hypervisor的实现方式一般是一个定制的Linux系统，如Xen何VMWare的ESXi都是这种类型。

### 3. 2型虚拟化

Host上安装常规的OS，如Linux，Windows等，Hypervisor作为OS上的一个程序运行，并对虚拟机进行管理。如KVM，VirtualBox和VMWare WorkStation就是这种类型。这种类型的虚拟化更加灵活，支持虚拟机的嵌套，即可以在KVM上再运行一个KVM。

## （二）KVM

### 1. KVM的概述

KVM（Kernal-Based Virtual Machine），即KVM是基于Linux内核实现的虚拟化基础设施，可以将Linux内核转化为一个Hypervisor。KVM在2007年被导入Linux2.6.20核心中，它能够在x86架构且硬件支持虚拟化技术（如Intel VT或AMD-T）的平台上运行。

### 2. KVM的虚拟化实现

KVM中有一个内核模块kvm.ko（kvm-intel.ko或kvm-AMD.ko），用来管理虚拟CPU和内存，而IO，存储和网络都交给Linux内核和Qemu来实现，QEMU软件（qemu-kvm）作为虚拟机上层控制和界面管理。

### 3. QEMU介绍

Qemu是一个广泛使用的开源计算机模拟器和虚拟机，它能够独立运行虚拟机，即根本不需要kvm，kqemu是该软件的加速软件。对于kvm来说，并不需要qemu来做虚拟化处理，只是需要qemu的上层管理界面进行虚拟机控开关至，虚拟机依旧是由kvm驱动的。

当qemu作为模拟器时，可以在一种架构（如x86 PC）下运行另一种架构（如ARM）下的操作系统和程序。通过使用动态转换，它可以获得非常好的性能。

当qemu作为虚拟机时，QEMU可以使用其他虚拟机管理程序（如 [Xen](https://wiki.archlinux.org/title/Xen_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)) 或 [KVM](https://wiki.archlinux.org/title/KVM_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87))）来使用CPU扩展（[HVM](https://en.wikipedia.org/wiki/Hardware-assisted_virtualization" \o "wikipedia:Hardware-assisted virtualization)）进行虚拟化，通过在主机CPU上直接执行客户机代码来获得接近于宿主机的性能。

### 4. Libvirt

#### （1）Libvirt概述

Libvirt是KVM的管理工具，当然除了KVM它还可以管理其他的Hypervisor，如Xen，VirtualBox等，OpenStack底层也使用了Libvirt。它实际上是一系列库函数提供给其他技术调用，来管理虚拟机的。它的设计是面向驱动的，对于任一种虚拟机技术都会开发该技术对应的驱动，这样不同的虚拟机技术就可以使用不同的驱动来调用libvirt的接口。

#### （2）Libvirt组成

Libvirt中包含三个组件：

后台daemon程序libvirtd：接受和处理API请求；

API库：让其他人可以开发基于Libvirt的高级工具，如virt-manager，这是个图形化的KVM管理工具；

virsh：KVM命令行工具；

## KVM虚拟化平台搭建

### 环境准备

CentOS7\_64位系统，配置图形界面。

### 查看硬件是否支持虚拟化

[root@localhost ~]# egrep ‘(vmx|svm)’ /proc/cpuinfo

结果是VMX表示Intel，SVM表示AMD。

### 安装KVM

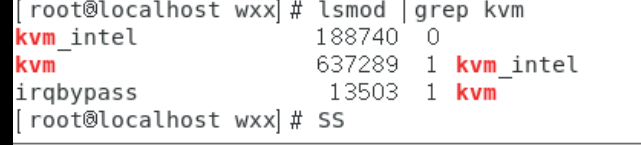
由于Linux内核已经将KVM收录了，在安装系统时已经加入了KVM，我们只需要在命令行模式下启用KVM即可：  
启用KVM模块:

[root@localhost ~]# modprobe kvm  
[root@localhost ~]# lsmod |grep kvm

modprobe可载入指定的个别模块，或是载入一组相依的模块。modprobe会根据depmod所产生的相依关系，决定要载入哪些模块。若在载入过程中发生错误，在modprobe会卸载整组的模块。

执行 lsmod 指令，会列出所有已载入系统的模块。Linux 操作系统的核心具有模块化的特性，因此在编译核心时，务须把全部的功能都放入核心。您可以将这些功能编译成一个个单独的模块，待需要时再分别载入。

可以看到是，KVM模块已经被载入Linux中。



### 4.安装kvm所依赖的组件

需要安装qemu-kvm和libvirt相关的组件，如下所示：

[root@localhost ~]# yum install -y virt-\* libvirt bridge-utils qemu-img qemu-kvm

安装完成后启动libvirtd后台服务：

[root@localhost ~]# service libvirtd start

查看libvirtd后台服务状态：

systemctl status libvirtd.service

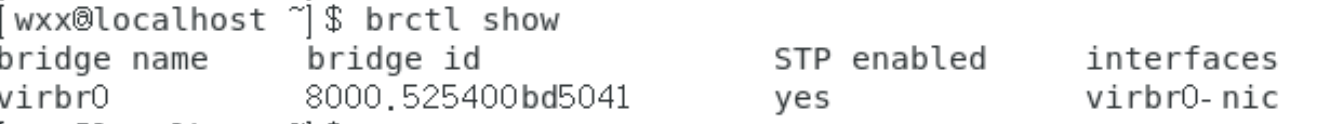
设置开机自启动libvirtd

systemctl enable libvirtd.service

### 5.创建物理桥接设备并关联网卡

在第4步中，我们已经创建了虚拟网桥virbr0，可以通过网桥管理命令查看：

brctl show



接下来，需要将NetworkManager服务关闭，开机启动也关闭：

[root@localhost ~]# chkconfig NetworkManager off  
[root@localhost ~]# systemctl stop NetworkManager

[root@localhost ~]# systemctl disable NetworkManager

接下来，需要创建物理桥接设备，使用virsh命令行工具创建物理网桥设备br0并关联网卡ens33到该设备中：

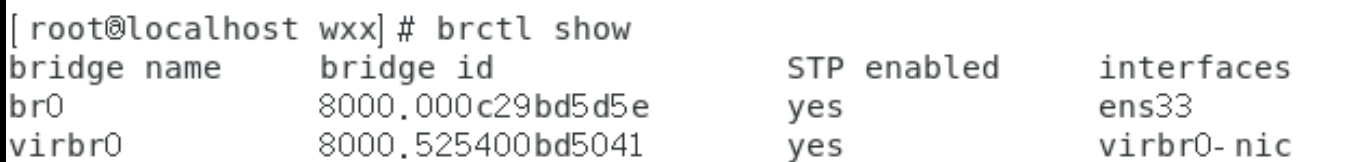
[root@localhost ~]# virsh iface-bridge ens33 br0

此时，ens33之前的IP已经被放置到br0网桥中了。

通过网桥管理来查看当前的网桥设备：

btctl show

如果列表中多了br0，则表示桥接成功。



### 6.测试使用KVM安装虚拟机

安装virt-manager图形化界面管理虚拟机

yum -y install virt-manager

启动virt-manager创建虚拟机：

virt-manager

从<http://download.cirros-cloud.net/>中下载一个很小的Linux镜像cirrors，上传到虚拟机中，使用virt-manager创建虚拟机，并分配内存和CPU，可以看到，cirrors已经成功在Centos7上生成了一个新的Linux虚拟机，如下所示：



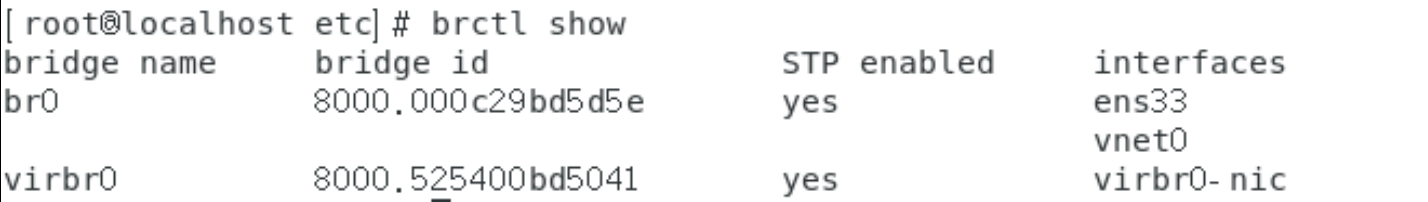
### 7.查看KVM虚拟机的状态

virsh list –all

### 8.登录虚机测试网络情况

登录cirros虚机，查看当前虚机的地址信息，然后尝试Ping宿主机以及百度，结果是成功的。说明当前的虚机已经和宿主机连通了，并且也能够访问外网了。返回到宿主机中，通过网桥管理器查看当前宿主机的网桥情况：

brvtl show



可以看到，br0是我们生成的物理网桥，它绑定了ens33宿主机的物理网卡，以及vnet0这个虚机的网卡，因此虚机和宿主机能够相互通信。

注意一点，虚拟中的虚拟网卡的名称是eth0，而不是宿主机中显示的vnet0，两个名称是不同的。

### 9.远程连接管理KVM虚拟机

由于KVM（准确来说是Libvirt）默认是不接受远程管理的，需要进行相关的配置才能够连接上KVM虚拟机。

## （四） KVM虚拟化原理

### 1. CPU虚拟化

KVM的虚拟化是需要CPU硬件的支持的，之前通过命令查找硬件的cpu信息中是否有vmx或svm，vmx是Intel的虚拟化，svm是AMD的虚拟化。

一个KVM虚拟机在宿主机中是一个qemu-kvm进程，与其他的Linux进程一样被调度，而虚机中的每一个虚机vCPU对应qemu-kvm进程的一个线程。

### 2.内存虚拟化

KVM通过内存虚拟机化共享物理系统内存，动态分配给虚机，为了在一台机器上运行多个虚机，KVM需要实现VA（虚拟内存）->PA（物理内存）->机器内存（MA）直接的地址转换。虚机的OS控制从VA到PA的转换，KVM负责PA到MA。

### 3.存储虚拟化

KVM的存储虚拟化通过存储池（Storage Pool）和卷（Volume）来管理的。存储池是宿主机上的一块空间，是多种类型的，卷是池中的一块空间，宿主机将卷分配给虚机，卷在虚机中就是一块硬盘。

文件目录是最常用的存储池类型，KVM将宿主机目录/var/lib/libvirt/images作为默认的存储池，而该存储池中的卷就是一个个的文件了，每一个文件对于虚拟来说就是一个磁盘。KVM所有可用的存储池都定义在宿主机的/etc/libvirt/storage目录下，每一个池子有一个XML文件，默认有一个default.xml文件，其中的默认目录就是/var/lib/libvirt/images。

使用文件作为卷的优点是：存储方便，移植性号，可复制，可远程访问。卷不一定都放在宿主机本地文件系统中，也可以存储在通过网络连接的远程文件系统，这样卷就能够在多个宿主机之间共享，便于虚机在不同宿主机之间进行热迁移。

KVM支持多种卷的文件格式，如：

raw是默认格式，原始磁盘镜像格式，移植性号，性能好，大小固定；

qcow2，cow表示copy on write写时复制，节省磁盘空间，支持多快照；

vmdk是VMWare的虚拟磁盘格式，即VMWare的虚机可以直接在KVM上运行；

### 4.网络虚拟化

网络虚拟化中最重要的是Linux Bridge和Vlan。

#### （1）Linux Bridge

宿主机中有一块连接外网的物理网卡eth0，如何让虚机VM1访问外网。选择的方案是给VM1分配一个虚拟网卡vnet0，通过Linux Bridge br0将eth0和vnet0连接起来，这样虚机和宿主机就通过Linux Bridege连接起来了。

Linux Bridge是Linux上用来做二层交换的设备，即二层交换机，多个网络设备可以连接到同一个Linux Bridge上，当某个设备收到数据包时，Linux Bridge会将数据转发给其他设备。

注意，在KVM中我们会看到一个virbr0，这是默认创建的一个Bridge，它的作用是为连接其上的虚拟网卡提供NAT访问外网的功能，virbo0默认分配了一个IP192.168.122.1，并通过dnsmasql为连接其上的其他虚拟网卡提供DHCP服务。注意，使用NAT服务的虚机可以访问外网，但是外网无法访问虚机，因为NAT会将虚机发出网络包的源地址转换为宿主机的IP地址，不像网桥，虚机通过自己的IP直接和外网通信。

#### （2）VLAN

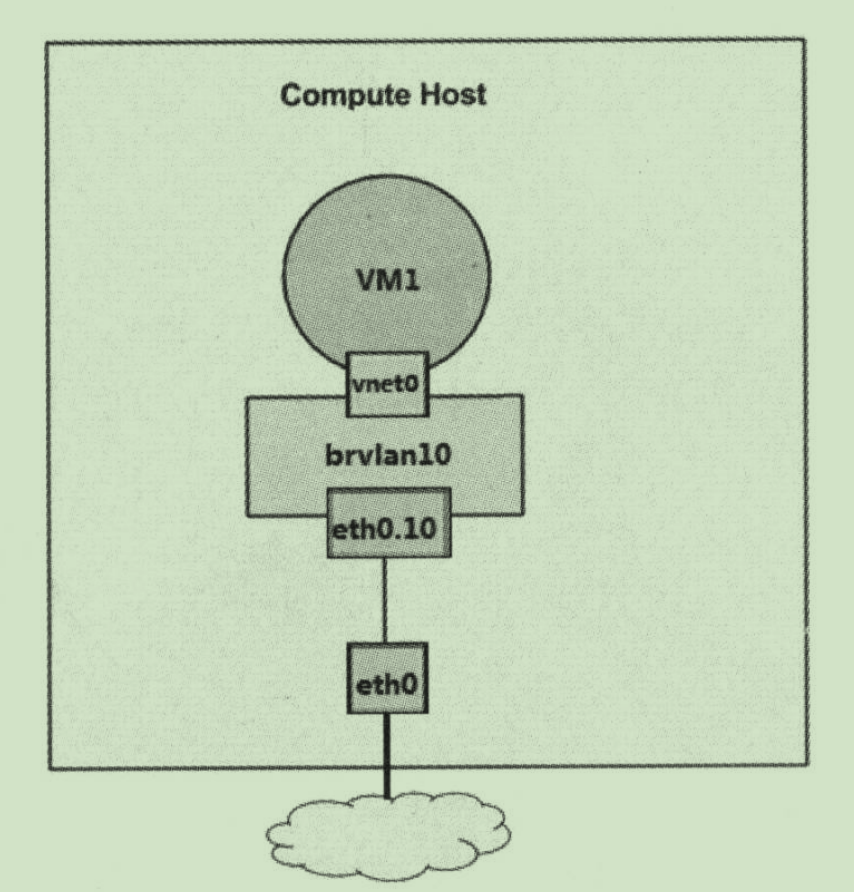
LAN表示本地局域网，通常使用集线器和交换机来连接LAN中的PC，一个LAN表示一个广播域，即LAN中的所有成员都会收到任一个成员发出的广播包。

VLAN表示虚拟的LAN，一个带有VLAN功能的交换机能够将自己的端口划分出多个LAN，即隔离出了多个广播域，每个域的二层广播包（arp）是无法跨越每个广播域的边界的，但是在三层上是可以通过路由器让不同的广播域相通的。

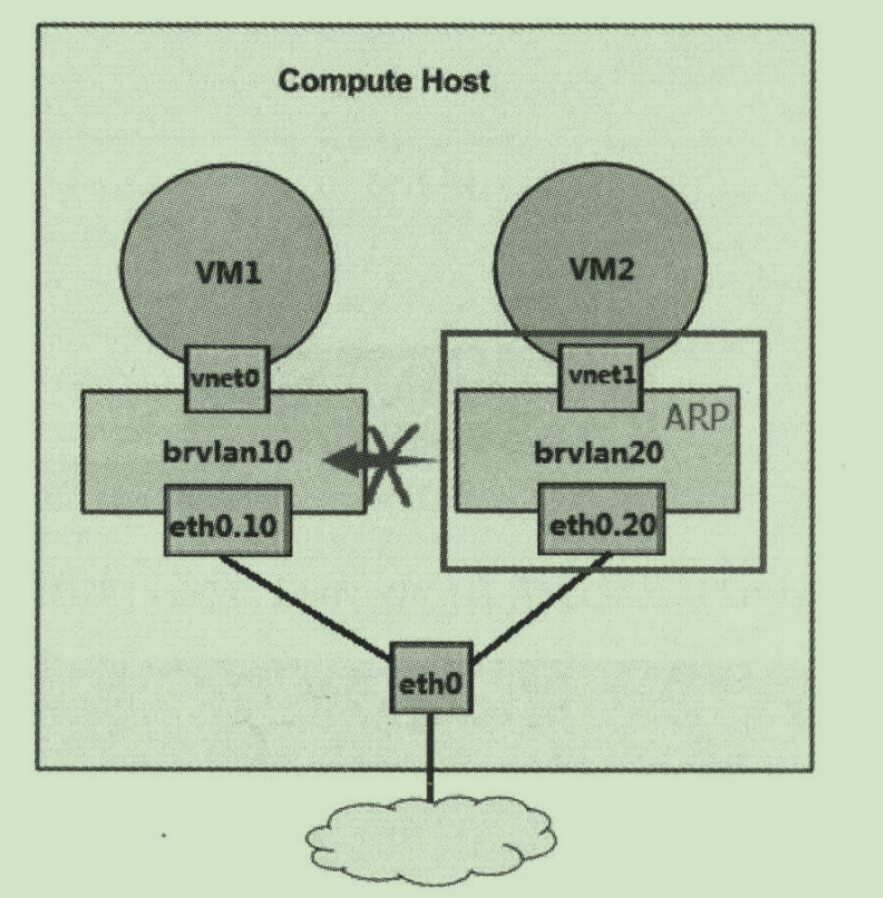
通常交换机的端口有两种配置模式：Access和Trunk。Access端口被打上了VLAN的标签，表明该端口属于哪个VLAN，不同的VLAN用VLAN ID来区分，Access端口都是和计算机的网卡直连的，这样从网卡出来的数据包流入Access端口后就能打上VLAN的标签。Trunk端口能够让不同的VLAN进行通信。

#### （3）KVM是如何实现VLAN的？

eth0是宿主机的物理网卡，宿主机通过软件实现一个虚拟交换机，在其上定义了一个VLAN10，通过VLAN设备eht0.10连接到虚拟网桥brvlan10上，同时连接的还有虚机VM1的虚拟网卡vnet0，这样vnet0，brvlan10，eth0.10相当于都接入了VLAN10的Access端口上，都属于VLAN10了，连接到虚拟网桥brvlan10上的设备都自动加入了VLAN10中，而eth0就是一个Trunk端口。



如果添加新的VLAN20网段到eth0上，那么两个VLAN区域的网络是隔离的，因为当VLAN10中的VM1向VLAN20中的VM2发送请求时，首先要获取VM2的MAC地址，因此在VLAN10中广播arp请求包，该请求包无法穿透VLAN10区域，因此获取不到VM2的MAC地址，也就无法与VM2通信。



#### （4）虚拟交换机的实现

交换机包含两种功能，一种是交换，另一种是隔离。Linux的VLAN设备的是隔离功能，将不同的VLAN网段隔离起来；Linux Bridge实现的是交换功能，将同一个VLAN的子设备挂载到一个Bridge上，设备之间就可以交换数据了。

因此，Linux Bridge + Vlan实现了虚拟交换机的功能。