虚拟化 --- KVM （ kernel-based VM ）

KVM 虚拟化需要处理器对虚拟化技术的支持，当我们需要进行虚拟机嵌套虚拟机时，我们需要让虚拟机中处理器对 VT 功能的支持达到透传的效果

虚拟化 --- KVM （ kernel-based VM ）

nested 虚拟机嵌套（ kvm on kvm ）
nested 技术，简单的说，就是在虚拟机上跑虚拟机。
KVM 虚拟机嵌套和 VMWare 原理不同， VMWare 第一层是用的硬件虚拟化技术，第二层就是完全软件模拟出来的，所以 VMWare 只能做两层嵌套。 KVM 是将物理 CPU 的特性全部传给虚拟机，所有理论上可以嵌套 N 多层

虚拟化 --- KVM （ kernel-based VM ）

1. 查看一层客户端是否支持 VT

#grep vmx /proc/cpuinfo

查询未果，证明一层 KVM 的虚拟机，并未将宿主机处理器的 VT 功能成功透传。所以，没有对 VT 功能的支持，我们不能实现在该层虚拟机中嵌套 KVM 虚拟机

虚拟化 --- KVM （ kernel-based VM ）

1. 查看一层客户端是否支持 VT

#grep vmx /proc/cpuinfo

查询未果，证明一层 KVM 的虚拟机，并未将宿主机处理器的 VT 功能成功透传。所以，没有对 VT 功能的支持，我们不能实现在该层虚拟机中嵌套 KVM 虚拟机

虚拟化 --- KVM （ kernel-based VM ）

为嵌套虚拟机做准备 --- CPU 虚拟化透传

1.# vim /etc/modprobe.d/kvm-nested.conf

在文件中添加下面语句

options kvm_intel nested=1

在宿主机启用 kvm_intel 模块的嵌套虚拟化功能，并且使透传永久有效

虚拟化 --- KVM （ kernel-based VM ）

为嵌套虚拟机做准备 --- CPU 虚拟化透传

2. 重新加载 kvm 模块

```
# modprobe   -r   kvm_intel
# modprobe   kvm_intel
```

虚拟化 --- KVM （ kernel-based VM ）

为嵌套虚拟机做准备 --- CPU 虚拟化透传

3. 验证是否加载成功

#cat /sys/module/kvm_intel/parameters/nested

Y ---“Y” 表示 cpu 虚拟化透传功能开启

虚拟化 --- KVM （ kernel-based VM ）

为嵌套虚拟机做准备 --- CPU 虚拟化透传

4.# virsh edit centos7 （ Virtual Machine Name ）

编辑需要做虚拟化透传的虚拟机的配置文件

<cpu mode='host-passthrough'>

host-passthrough 直接将物理 CPU 暴露给虚拟机使用，在虚拟机上完全可以看到的就是物理 CPU 的型号

虚拟化 --- KVM （ kernel-based VM ）

HOST 技术适用的情况：
1 CPU 压力非常大；
2 需要将物理 CPU 的一些特性传给虚拟机使用；
3 需要在虚拟机里面看到和物理 CPU 一模一样的 CPU 品牌型号，这个在一些公有云很有意义；
注意： HOST 方式虚拟机不能迁移到不同型号的 CPU 上；

虚拟化 --- KVM （ kernel-based VM ）

命令行下管理虚拟机

virsh 既有命令行模式，也有交互模式，在命令行直接输入 virsh 就进入交互模式， virsh 后面跟命令参数，则是命令行模式

语法结构：

virsh <command> <domain-id> [OPTIONS]

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下管理虚拟机

#virsh
Welcome to virsh, the virtualization interactive terminal.
Type: 'help' for help with commands
'quit' to quit
切换到 virsh 命令行维护模式下

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下管理虚拟机

#help list --- 列出 list 命令下的参数

--inactive list inactive domains

--all list inactive & active domains

--with-snapshot list domains with existing snapshot

--without-snapshot list domains without a snapshot

--state-running list domains in running state

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下管理虚拟机

#help list --- 列出 list 命令下的参数

--state-paused list domains in paused state
--state-shutoff list domains in shutoff state
--state-other list domains in other states
--autostart list domains with autostart enabled
--no-autostart list domains with autostart disabled

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下管理虚拟机

#help list --- 列出 list 命令下的参数

--uuid list uuid's only

--name list domain names only

--table list table (default)

--managed-save mark inactive domains with managed save state

--title show short domain description

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下管理虚拟机

#help list --- 列出 list 命令下的参数

--inactive list inactive domains

--all list inactive & active domains

--transient list transient domains

--persistent list persistent domains

--with-snapshot list domains with existing snapshot

--without-snapshot list domains without a snapshot

--state-running list domains in running state

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下简单管理虚拟机

virsh # list
列出当前宿主机上处于运行状态的虚拟机

virsh#list --all
列出当前宿主机上所有的虚拟机

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下简单管理虚拟机

virsh # start liu1 （ name ）
开启某一台虚拟机
virsh # shutdown liu1 （ name ）
正常关闭某一台虚拟机
virsh # destory liu1 （ name ）
强制关闭某一台虚拟机

虚拟化 --- KVM （ kernel-based VM ）

基础操作 --- 命令行下简单管理虚拟机

virsh #virsh autostart liu1 （ name ）
开机自启动虚拟机 libvirtd
virsh #virsh autostart --disable liu1 （ name ）
关闭开机自启动

虚拟化 --- KVM （ kernel-based VM ）

1. 虚拟机配置文件的位置 /etc/libvirt/qemu 下
2. 配置文件的名字必须为 *.xml （以 .xml 结尾）
3. 我们可以利用某个文件作为模板，改动后我们又可以利用该模板生成一个新的满足我们需求的虚拟机

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<domain type='kvm'>
  <name>s1</name>
  <uuid>4f8702f3-e710-cda4-41ea-
835ee7a7ee38</uuid>
  <memory unit='KiB'>2560000</memory>
  <currentMemory
unit='KiB'>2560000</currentMemory>
  <vcpu placement='static' current='2'>3</vcpu>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/kvm-spice</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw'/>
      <source file='/var/lib/libvirt/images/s1.img'/>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000'
bus='0x00' slot='0x05' function='0x0'/
>
    </disk>
    <disk type='file' device='cdrom'>
      <driver name='qemu' type='raw'/>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<source file='/home/liulg/ 学习 /ISO 镜像 /C7_64_everything.iso'/>
      <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>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<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>
    <interface type='network'>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<mac address='52:54:00:9e:23:2d'/>
      <source network='default'/>
      <model type='rtl8139'/>
      <address type='pci' domain='0x0000'
bus='0x00' slot='0x03' function='0x0'/
>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
</interface>
    <serial type='pty'>
      <target port='0'/>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <input type='tablet' bus='usb'/>
    <input type='mouse' bus='ps2'/>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<input type='keyboard' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes'/>
    <sound model='ich6'>
      <address type='pci' domain='0x0000'
bus='0x00' slot='0x04' function='0x0'/
>
    </sound>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
```

虚拟化 --- KVM （ kernel-based VM ）

配置文件解读：

```
<address type='pci' domain='0x0000' bus='0x00'
slot='0x02' function='0x0'/
>
    </video>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000'
bus='0x00' slot='0x06' function='0x0'/
>
    </memballoon>
  </devices>
</domain>
```

虚拟化 --- KVM （ kernel-based VM ）

至此，我们在解读完配置文件之后，只需简单复制 / 修改即可生成一个新的虚拟机出来

```
#cd /etc/libvirt/qemu
#cp s1.xml     liu.xml
#vim   liu.xml
```

```
#virsh  define   liu.xml  ---  加载配置文件给 qemu
#virsh  start  liu
#virsh   list  --all
#virsh   destory   liu
```

虚拟化 --- KVM （ kernel-based VM ）

命令行下管理虚拟机的其他命令

#dominfo　liu（ name ）

查看虚拟机 -liu 的详细信息

#domstat　liu（ name ）

查看虚拟机 -liu 的状态

#domid　liu（ name ）

查看虚拟机的 ID 号

#dumpxml　liu　（ name ）

查看虚拟机的配置文件的信息（开启后可能与安装定义时不同，因为随着启动会分配一些端口 /IP.. ）

虚拟化 --- KVM （ kernel-based VM ）

命令行下管理虚拟机的其他命令

#edit liu（ name ）

编辑某个虚拟机的配置文件

#setmem liu（ name ） 512000

修改虚拟机当前的内存大小，修改之后可通过 dumpxml liu 来查看修改之后的数值

#setmaxmem liu（ name ） 600000

#setvcpus liu 4

修改内存的最大使用值和 cpu 的个数

虚拟化 --- KVM （ kernel-based VM ）

命令行下创建 && 管理资源池

#pool-list
列出存储池
#pool-define-as pooliu dir - - - - “/pool”
将本地 /pool 目录指定为新存储池资源（非激活）
#pool-build poolliu
构造存储池

虚拟化 --- KVM （ kernel-based VM ）

命令行下创建 && 管理资源池
#pool-start poolliu
激活存储池
#pool-autostart
存储池随 libvirtd 服务开机自启动
#pool-info poolliu
查看某个存储池的相关信息

虚拟化 --- KVM （ kernel-based VM ）

命令行下创建 && 管理磁盘映像
qemu-img create -f raw /img/s1.img 10G
创建一个大小为 10G 的，格式为 raw 的磁盘映像
qemu-info /img/s1.img
查看映像文件的详细信息

警告： 不要用 qemu-img 命令来修改被运行中的虚拟机或任何其它进程所正在使用的映像，那样映像会被破坏。

虚拟化 --- KVM （ kernel-based VM ）

命令行下创建 && 管理虚拟机
Virt-install
命令行下创建虚拟机的命令
不过，在它后面需要跟上很多的 参数

虚拟化 --- KVM （ kernel-based VM ）

--name: 虚拟机的名字。
--disk Location: 磁盘映像的位置。
--graphics：怎样连接 VM ，通常是 SPICE 。
--vcpu：虚拟 CPU 的数量。
--ram：以兆字节计算的已分配内存大小。
--location：指定安装源路径。

--network：指定虚拟网络，通常是 virbr0 或者自己设定的 br0

虚拟化 --- KVM （ kernel-based VM ）

命令行下安装虚拟机

virt-install --name=C7llg --disk path=/pool/C7llg.img --ram=1024 --vcpus=1 --graphics spice --location=/home/liulg/ 学习 /ISO 镜像 /C7_64_everything.iso --network bridge=virbr0

以此安装操作之后，会在桌面显示一个 virt-viewer ，进入到安装步骤

虚拟化 --- KVM （ kernel-based VM ）

除了这些简单命令，实际在 virsh 形式下还有更多的命令

虚拟化 --- KVM （ kernel-based VM ）

help 打印帮助
attach-device 从一个 XML 文件附加装置
attach-disk 附加磁盘设备
attach-interface 获得网络界面
autostart 自动开始一个域
capabilities 性能
cd change the current directory
connect 连接（重新连接）到 hypervisor
console 连接到客户会话
cpu-baseline compute baseline CPU

虚拟化 --- KVM （ kernel-based VM ）

cpu-compare compare host CPU with a CPU described by an XML file

create 从一个 XML 文件创建一个域

start 开始一个（以前定义的）非活跃的域

destroy 删除一个域

detach-device 从一个 XML 文件分离设备

detach-disk 分离磁盘设备

detach-interface 分离网络界面

define 从一个 XML 文件定义（但不开始）一个域

虚拟化 --- KVM （ kernel-based VM ）

domid 把一个域名或 UUID 转换为域 id
domuuid 把一个域名或 id 转换为域 UUID
dominfo 域信息
domjobinfo domain job information
domjobabort abort active domain job
domname 将域 id 或 UUID 转换为域名
domstate 域状态
domblkstat 获得域设备块状态

虚拟化 --- KVM （ kernel-based VM ）

domifstat 获得域网络接口状态

dommemstat get memory statistics for a domain

domblkinfo domain block device size information

domxml-from-native Convert native config to domain XML

domxml-to-native Convert domain XML to native config

dumpxml XML 中的域信息

edit 编辑某个域的 XML 配置

虚拟化 --- KVM （ kernel-based VM ）

domifstat 获得域网络接口状态

dommemstat get memory statistics for a domain

domblkinfo domain block device size information

domxml-from-native Convert native config to domain XML

domxml-to-native Convert domain XML to native config

dumpxml XML 中的域信息

edit 编辑某个域的 XML 配置

虚拟化 --- KVM （ kernel-based VM ）

find-storage-pool-sources 发现潜在存储池源

find-storage-pool-sources-as 找到潜在存储池源

freecell NUMA 可用内存

hostname 打印管理程序主机名

list 列出域

migrate 将域迁移到另一个主机中

migrate-setmaxdowntime set maximum tolerable downtime

net-autostart 自动开始网络

虚拟化 --- KVM （ kernel-based VM ）

pool-destroy 销毁池
pool-delete 删除池
pool-dumpxml XML 中的池信息
pool-edit 为存储池编辑 XML 配置
pool-info 存储池信息
pool-list 列出池
pool-name 将池 UUID 转换为池名称
pool-refresh 刷新池
pool-start 启动一个（以前定义的）非活跃的池

虚拟化 --- KVM （ kernel-based VM ）

quit 退出这个非交互式终端
exit 退出这个非交互式终端
reboot 重新启动一个域
restore 从一个存在一个文件中的状态恢复一个域
resume 重新恢复一个域
save 把一个域的状态保存到一个文件
schedinfo 显示 / 设置日程安排变量
dump 把一个域的内核 dump 到一个文件中以方便分析

虚拟化 --- KVM （ kernel-based VM ）

虚拟机管理命令 --- virt tools
使用之前我们需要先安装软件
#yum -y install libguestfs-tools virt-top

安装完成之后让我们与他们愉快的玩耍一下吧

虚拟化 --- KVM （ kernel-based VM ）

#virt-top
#virt-what
#virt-ls
#virt-cat
#virt-df
#virt-copy-out
#virt-copy-in