Libvirt URI: qemu:///system VS qemu:///session

==========

qemu:///session 使用非特权的当前 user 的 uid/gid 连接到本地 libvirtd 实例
多用于桌面虚拟化, 每个 user 都有自己的 qemu:///session VMs
它有很多缺点，比如默认只能使用非常慢的 usermode networking, 若想使用桥，则需要 setuid 来帮助
host pci passthrough 也没有可能

qemu:///system 使用特权用户连接到本地 libvirtd 实例
可以访问所有资源，用于服务器虚拟化. 此 uri 也是 virt-manager 的默认选择

例 1:

```
MiWiFi-R1CL-srv:~ # id
uid=0(root) gid=0(root) groups=0(root)
MiWiFi-R1CL-srv:~ #
MiWiFi-R1CL-srv:~ # virsh uri
qemu:///system

MiWiFi-R1CL-srv:~ # virsh list --all
 Id    Name                           State
----------------------------------------------------
 -     sles12sp2                      shut off
 -     tumbleweed                     shut off

MiWiFi-R1CL-srv:~ # su - suse
suse:~$
suse:~$ id
uid=1000(suse) gid=100(users) groups=100(users)
suse:~$
suse:~$ virsh uri
qemu:///session

suse:~$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 -     sles11sp3                      shut off

suse:~$ su - novell
Password:
novell:~$
novell:~$ id
uid=1001(novell) gid=100(users) groups=100(users)
novell:~$
novell:~$ virsh uri
qemu:///session

novell:~$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 -     sles15rc1                      shut off

novell:~$
```

Libvirt URI aliases

==========

可通过/etc/libvirt/libvirt.conf 这个 client 端配置文件为特权用户自定义 uri 别名，以便那些调用 libvirt API 的管理工具去使用

例 2, 自定义两个别名，分别叫做 beijing 和 chongqing

```
c610:~ # egrep 'listen_tcp|auth_tcp' /etc/libvirt/libvirtd.conf
listen_tcp = 1
auth_tcp = "none"
c610:~ #
```

```
root:~# cat /etc/libvirt/libvirt.conf
uri_aliases = [
  "beijing=qemu+tcp://root@10.67.161.107/system",
  "chongqing=qemu+ssh://root@10.67.161.101/system",
]
root:~#
root:~# virsh -c beijing list --all
 Id    Name                           State
----------------------------------------------------
 -     sles12                         shut off
 -     sles12sp2                      shut off

root:~# virsh -c beijing nodeinfo
CPU model:           x86_64
CPU(s):              56
CPU frequency:       1350 MHz
CPU socket(s):       1
Core(s) per socket:  14
Thread(s) per core:  2
NUMA cell(s):        2
Memory size:         32682572 KiB

root:~# virsh -c chongqing list --all
Password:
 Id    Name                           State
----------------------------------------------------
 -     sles11sp4                      shut off
 -     sles12sp1                      shut off

root:~# virsh -c chongqing nodeinfo
Password:
CPU model:           x86_64
CPU(s):              12
CPU frequency:       2399 MHz
CPU socket(s):       1
Core(s) per socket:  6
Thread(s) per core:  2
NUMA cell(s):        1
Memory size:         32881800 KiB

root:~#
```

```
root:~# virsh
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh # uri
qemu:///system

virsh # connect beijing

virsh # uri
qemu+tcp://root@10.67.161.107/system

virsh # list --all
 Id    Name                           State
----------------------------------------------------
 -     sles12                         shut off
 -     sles12sp2                      shut off

virsh # nodeinfo
CPU model:           x86_64
CPU(s):              56
CPU frequency:       1200 MHz
CPU socket(s):       1
Core(s) per socket:  14
Thread(s) per core:  2
NUMA cell(s):        2
Memory size:         32682572 KiB

virsh # connect chongqing
Password:

virsh # uri
qemu+ssh://root@10.67.161.101/system

virsh # list --all
 Id    Name                           State
----------------------------------------------------
 -     sles11sp4                      shut off
 -     sles12sp1                      shut off

virsh # nodeinfo
CPU model:           x86_64
CPU(s):              12
CPU frequency:       2399 MHz
CPU socket(s):       1
Core(s) per socket:  6
Thread(s) per core:  2
NUMA cell(s):        1
Memory size:         32881800 KiB

virsh #
```

例 3: virt-install 使用 libvirt uri 别名远程安装 guest, 并传入串口配置的参数给 guest kernel, 使之输出消息至默认的 pty 后端的虚拟串口 S0, 用户可通过 virsh 的 console 子命令连接至 pty 获得交互环境

```
# virt-install \
--connect beijing \
--virt-type kvm \
--name sles12sp3 \
--memory 2048 \
--cpu kvm64,+x2apic \
--vcpu=2 \
--location http://147.2.207.1/install/SLP/SLE-12-SP3-Server-LATEST/x86_64/DVD1/ \
--disk path=/opt/vms/sles12sp3/disk0.qcow2,format=qcow2,bus=virtio \
--network model=virtio,bridge=br0 \
--graphics spice \
--extra-args "console=ttyS0,115200" \
--noautoconsole

# virsh -c beijing console sles12sp3
```

例 4: virt-install 使用 libvirt uri 别名远程安装 guest, 配置了一个 tcp 后端的虚拟串口(在所有网络地址的 tcp 5678 端口侦听), 并传入串口配置的参数给 guest kernel, 使之输出消息至该串口, 用户可通过 telnet 或 nc 等工具连接至该端口获得交互环境

```
# virt-install \
--connect beijing \
--virt-type kvm \
--name sles12sp3 \
--memory 2048 \
--cpu kvm64,+x2apic \
--vcpu=2 \
--location /opt/isos/SLE-12-SP3-Server-DVD-x86_64-GM-DVD1.iso \
--disk path=/opt/vms/sles12sp3/disk0.qcow2,format=qcow2,bus=virtio \
--network model=virtio,bridge=br0 \
--graphics spice \
--serial tcp,host=0.0.0.0:5678,mode=bind,protocol=telnet \
--extra-args "console=ttyS0,115200" \
--noautoconsole

# telnet 10.67.161.107 5678
# nc 10.67.161.107 5678
```

btw, 截止到目前，virt-install 远程安装虚拟机有个 bug,所以有些场景下会出问题，等我有时间就给它修了！

attach/detach virtual disks, nics and serial through virsh

==========

先说一下，virsh 的大多数子命令都支持四个选项：‘’--persistent’, ‘--config’, ‘--live’ 和 ‘--current’

--persistent　永久生效。

--config　改变会被保存至虚拟机配置文件，但在下一次虚拟机上电时(即 power on)才生效

--live　改变会立即生效，但不会被保存至虚拟机配置文件，也就是说虚拟机 power off 后，对设备的变更效果就消失。

--current　通俗点说，该选项就是墙头草的意思，若虚拟机的状态是 running，则其作用等效于—live；若虚拟机状态是 shut off，则其作用等效于--persistent

即:

对于 shut off 的虚拟机, --persistent, --config, --current 的效果是等价的.

对于 running 的虚拟机, --persistent 的效果等价于'--config' + '--live'.

对于 running 的虚拟机, --config 会把对设备的变更保存至虚拟机配置文件，但在下一次虚拟机上电时(即 power on)才生效.

对于 running 的虚拟机, --live 对设备的变更会立即生效但不保存. 一旦虚拟机 power off，就好似梦醒了，又回到了现实世界.

例 5: 通过 virsh attach-device/detach-device 从 xml 文件添加/删除网络或磁盘设备

绿色部分是可选项

对于虚拟网卡的 mac 项，未指定就会随机生成；对于 model 项，未指定就表示 hypervisor default，对于 qemu/kvm 的虚拟机来说，就是 rtl8139

```
# cat e1000_br0.xml
<interface type='bridge'>
  <mac address='52:54:00:35:e9:87'/>
  <source bridge='br0'/>
  <model type='e1000'/>
</interface>

# cat virtio_default.xml
<interface type='network'>
  <mac address='52:54:00:9f:bb:a0'/>
  <source network='default'/>
  <model type='virtio'/>
</interface>

# virsh attach-device sles12sp2 --file e1000_br0.xml --persistent
# virsh attach-device sles12sp2 --file virtio_default.xml --persistent

# virsh detach-device sles12sp2 --file virtio_default.xml --persistent


# cat virtio_blk_disk.xml
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2'/>
  <source file='/opt/vms/images/image0.qcow2'/>
  <target dev='vdb' bus='virtio'/>
</disk>
```

注: 当编写上述 xml 文件时，若不确定 target dev 应该用什么，可以先查看该虚拟机中已存在的磁盘设备名，比如

```
# virsh domblklist sles12sp2
Target     Source
------------------------------------------------
vda        /opt/vms/sles12sp2/disk0.qcow2
```

此输出表示该虚拟机中已经有一个 virtio blk 类型的虚拟磁盘了，’vda’已经被占用。当用户新添加一个 virtio blk 虚拟磁盘时，磁盘设备名应该用’vdb’.

```
# virsh attach-device sles12sp2 --file virtio_blk_disk.xml --persistent
# virsh detach-device sles12sp2 --file virtio_blk_disk.xml --persistent
```

添加 virtio scsi disk:

```
# cat virtio_scsi_controller0.xml
<controller type='scsi' index='0' model='virtio-scsi'/>

# cat virtio_scsi_0_disk.xml
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2'/>
  <source file='/opt/vms/images/image0.qcow2'/>
  <target dev='sda' bus='scsi'/>
  <address type='drive' controller='0'/>
</disk>
```

假设该虚拟机中没有任何虚拟 scsi HBA，那么用户新增 scsi 类型的虚拟磁盘时，最好先手动添加一个虚拟 scsi HBA。上面的'0'表示显式指定该 HBA 的索引值，在以后添加虚拟 scsi disks 时，此值标识着连接到哪个 HBA 上面. 若不指定，libvirt 会自动生成一个. 简单场景中是不会出问题的，但在多虚拟 scsi HBA 的场景中，可能出现用户的 scsi disks 被连接到非预期的 scsi HBA 上的情况。

```
# virsh attach-device sles12sp2 --file virtio_scsi_controller0.xml --persistent
# virsh attach-device sles12sp2 --file virtio_scsi_0_disk.xml --persistent
```

以 lun passthrough 的方式添加主机设备 sda 给虚拟机：

```
# cat virtio_scsi_lun_passthrough_host_sda.xml
<disk type='block' device='lun'>
  <driver name='qemu' type='raw'/>
  <source dev='/dev/sda'/>
  <target dev='sda' bus='scsi'/>
  <address type='drive' controller='0'/>
</disk>

# virsh attach-device sles12sp2 --file virtio_scsi_lun_passthrough_host_sda.xml --persistent
```

例 6: 通过 virsh attach-interface/detach-interface 添加/删除虚拟网络设备

```
# virsh attach-interface sles12sp3 --type bridge --source br0 --model virtio --mac 52:54:00:4b:73:5d  --persistent
# virsh attach-interface sles12sp3 --type network --source default --model virtio --mac 52:54:00:4b:73:5f --persistent

# virsh domiflist sles12sp3
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet0      bridge     br0        virtio      52:54:00:fb:0f:c8
vnet1      bridge     br0        virtio      52:54:00:4b:73:5d
vnet2      network    default    virtio      52:54:00:4b:73:5f

# virsh detach-interface sles12sp3 bridge --mac 52:54:00:4b:73:5d --persistent
# virsh detach-interface sles12sp3 network --mac 52:54:00:4b:73:5f --persistent
```

例 7: 通过 virsh attach-disk/detach-disk 添加/删除虚拟磁盘
老规矩，操作虚拟磁盘前还是先看下目标虚拟机的磁盘情况
virsh domblklist sles12sp3
Target Source
---
vda /opt/vms/sles12sp3/disk0.qcow2

virsh attach-disk sles12sp3 /opt/vms/images/image0.qcow2 vdb --driver qemu --subdriver qcow2 --targetbus virtio --persistent
virsh detach-disk sles12sp3 vdb --persistent

virsh attach-disk sles12sp3 /opt/vms/images/image1.raw vdb --driver qemu --subdriver raw --targetbus virtio --persistent

virsh attach-disk sles12sp3 /opt/isos/SLE12SP3-x86_64-DVD1.iso sda --targetbus scsi --type cdrom --persistent
virsh detach-disk sles12sp3 sda --persistent

virsh attach-disk sles12sp3 /opt/isos/SLE12SP3-x86_64-DVD1.iso hda --targetbus ide --type cdrom --persistent
virsh detach-disk sles12sp3 hda --persistent

virsh attach-disk sles12sp3 /dev/sr0 hda --type cdrom --persistent

添加一个连接到 virtio scsi HBA 0 上的虚拟磁盘,假设添加前虚拟机没有任何 scsi HBA 的存在，本例中我们先添加一个 virtio scsi HBA,索引号为 0，然后添加一个虚拟磁盘连接到这个索引号为 0 的 virtio scsi HBA 上面。
virsh attach-device sles12sp3 virtio_scsi_controller0.xml --persistent

virsh attach-disk sles12sp3 /opt/vms/images/image0.qcow2 sda --driver qemu --subdriver qcow2 --targetbus scsi --address scsi:0.0.0 --persistent
注: address format for scsi 是 scsi:controller.bus.unit

以 lun passthrough 的方式添加主机设备 sda 给虚拟机：
virsh attach-disk sles12sp3 --source /dev/sda --target sdb --sourcetype block --type lun --driver qemu --subdriver raw --targetbus scsi --address scsi:0.0.1 --persistent

例 8: 通过 virsh attach-device/detach-device 从 xml 文件添加/删除串口设备
若虚拟机当初 是通过 virt-install 或 virt-manager 安装，则已经包含了一个缺省的 pty 类型的串口 0.
用户可以使用 virsh console sles12sp3 来直接连接到该串口 0,如果虚拟机里面配置了输出消息至该串口，则用户就可以在 virsh console 中看到消息或与之交互。

下面演示的是 为用户再额外添加一个 file 类型的串口 **1**. 如此，虚拟机中凡是送往此串口的消息就会被保存在 host 上面的文件中，便于用户将其作为日志去分析。BTW, file 类型是不具体交互能力了，换句话说，用户只能读，不能写消息给虚拟机。若需要交互，则需要将类型变更为 pty, tcp, udp 或 unix socket.
cat serial1_file.xml
```
<serial type='file'>
  <source path='/tmp/sles12sp3_serial1.log'/>
  <target port='1'/>
</serial>
```

virsh attach-device sles12sp3 --file serial1_file.xml --persistent
virsh detach-device sles12sp3 --file serial1_file.xml --persistent

KVM-based Nested virtualization

==========

我们把第一级 guest 称为 L1 guest, 而在 L1 guest 中创建的 guest 称为 L2 guest.

例 9:

以 intel x64 处理器为例，想使用嵌套虚拟化，要求有 2 点

1. host 加载 kvm_intel 模块时，传入 nested=1 这个参数

若想每次开机自动为 kvm_intel 模块加载此参数，可以在 host 的 grub 中给 kernel 传入参数 kvm-intel.nested=1, 当然，通过/etc/modprobe.d/ 也可以做到.

若只是想临时使用，那么只需要重新加载一次 kvm_intel 模块：

先关闭所有虚拟机，然后

```
# modprobe -r kvm_intel
# modprobe kvm_intel nested=1
```

2. 通过 virsh edit sles12sp3 命令编辑 L1 guest, 将 cpu 部分变更为：

```
<cpu mode='host-passthrough'/>       //若使用这种 cpu model，该虚拟机就失去了被 live migration 的能力
```

或

加入 policy 为 require 的 vmx feature,像下面：

```
  <cpu mode='custom' match='exact' check='partial'>
    <model fallback='allow'>SandyBridge</model>
    <feature policy='require' name='vmx'/>
  </cpu>
```

如此, 启动 L1 guest 后就可以在其中看到处理器的 vmx 标志了，也就是说，现在用户可以开始在 L1 guest 中创建或操作基于 kvm 的 L2 guest 了

Send key through virsh

==========

有时用户想发送某些组合键到指定的 guest，但因故无法以常规方式发送时, 就可以借助 libvirt 的 sendkey 功能来发送

例 10:

比如某些会被 host 先响应的组合键如 ctrl-alt-f1…, 可以这样做：

```
# virsh qemu-monitor-command sles12sp3 --hmp "sendkey ctrl-alt-f1"
```

或

```
# virsh send-key sles12sp3 KEY_LEFTCTRL KEY_LEFTALT KEY_F7
```

比如当虚拟机挂掉，内核已无法响应普通按键的中断, 若用户之前配置了 sysrq,现在想通过它得到转储或其他信息，可以这样做：

```
# virsh qemu-monitor-command sles12sp3 --hmp "sendkey alt-sysrq-m"
```

或

```
# virsh send-key sles12sp3 KEY_LEFTALT KEY_SYSRQ KEY_M
```

Get guest network address info through libvirt on host

==========

先看右侧这张截图 - >
我们知道 libvirt 创建的虚拟网络'default'在 host 上的默认设备名就是 virbr0. 此虚拟机中有三个网卡。其中 vnet1 和 vnet2 最终都挂到了 virbr0 上面，但从 libvirt 角度看，vnet1 是通过'bridge'类型的挂接方式，而 vnet2 则是'network'类型的挂接方式。

```
c610:/opt/vms # virsh domiflist sles12sp3
setlocale: No such file or directory
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet0      bridge     br0        virtio      52:54:00:fb:0f:c8
vnet1      bridge     virbr0     virtio      52:54:00:4b:73:5f
vnet2      network    default    virtio      52:54:00:4b:73:56

c610:/opt/vms #
```

virsh 的子命令 domifaddr 用于输出 guest 中网络设备的地址信息
它有一个选项'--source', 值为'**lease**'或'**agent**'. 当用户仅输入 virsh domifaddr sles12sp3 时，默认就是--source lease.

'**lease**'仅在下列情况都满足时，才会返回 guest 中网络设备的地址信息
1. 网络设备的连接类型是'network'.
2. 该网络设备的'Source'上面有 libvirt 提供的 dhcp service.
这两条很容易满足，因为 libvirt 创建虚拟网络时，默认就会
调用 dnsmasq 在对应接口上提供 dhcp service，比如右侧截图 - >

```
c610:/opt/vms # virsh net-dumpxml default
setlocale: No such file or directory
<network connections='1'>
  <name>default</name>
  <uuid>b3f39768-54dd-43ce-bccf-d0908b0e6e84</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:8b:a5:62'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>

c610:/opt/vms #
```

'lease'获得地址信息的原理就是在 host 上面 libvirt 去绑定
在该接口上提供 dhcp 服务的租约文件中按 mac 查 ip.并返回

'**agent**'仅在下列情况都满足时，才会返回 guest 中网络设备的地址信息:
1. 虚拟机配置中包含一个**特殊名字**的 channel.(channel 说白了就是 virtio 类型的串口)
2. 虚拟机里面有 qemu guest agent 服务在 running.

```
c610:/opt/vms # virsh dumpxml sles12sp3 | sed -n '/<channel/,/<\/channel/p'
    <channel type='spicevmc'>
      <target type='virtio' name='com.redhat.spice.0' state='connected'/>
      <alias name='channel0'/>
      <address type='virtio-serial' controller='0' bus='0' port='2'/>
    </channel>
    <channel type='unix'>
      <source mode='bind' path='/var/lib/libvirt/qemu/channel/target/domain-27-sles12sp3/org.qemu.guest_agent.0'/>
      <target type='virtio' name='org.qemu.guest_agent.0' state='connected'/>
      <alias name='channel1'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
    </channel>
c610:/opt/vms #
```

例 11: 如果没有这个 channel，就新建一个：# cat channel_qemu_ga.xml

```
<channel type='unix'>
  <target type='virtio' name='org.qemu.guest_agent.0'/>
</channel>
```

virsh attach-device sles12sp3 --file channel_qemu_ga.xml --persistent

```
guest:~ # rpm -qa | grep qemu-guest-agent
qemu-guest-agent-2.9.1-6.9.2.x86_64
guest:~ # systemctl status qemu-ga.service
● qemu-ga.service - QEMU Guest Agent
   Loaded: loaded (/usr/lib/systemd/system/qemu-ga.service; s
   Active: active (running) since Wed 2018-03-21 18:22:32 CST
     Docs: http://wiki.qemu.org/Features/GuestAgent
 Main PID: 962 (qemu-ga)
    Tasks: 1 (limit: 512)
   CGroup: /system.slice/qemu-ga.service
           └─962 /usr/bin/qemu-ga

Mar 21 18:22:32 guest systemd[1]: Started QEMU Guest Agent.
guest:~ #
```

'agent'获得信息的原理就是 libvirt 发送 qemu 代理命令'guest-network-get-interfaces'，在虚拟机中运行着的 qemuguest agent daemon 就会从 virtio 串口收到此命令然后从此串口返回所有网络设备地址信息给 host

```
c610:~ # virsh domifaddr sles12sp3 --source lease
 Name       MAC address          Protocol     Address
-------------------------------------------------------------------------------
 vnet2      52:54:00:4b:73:56    ipv4         192.168.122.232/24

c610:~ # virsh domifaddr sles12sp3 --source agent
 Name       MAC address          Protocol     Address
-------------------------------------------------------------------------------
 lo         00:00:00:00:00:00    ipv4         127.0.0.1/8
 -          -                    ipv6         ::1/128
 eth0       52:54:00:fb:0f:c8    ipv4         10.67.162.172/21
 -          -                    ipv6         fe80::5054:ff:fefb:fc8/64
 eth1       52:54:00:4b:73:5f    ipv4         192.168.122.241/24
 -          -                    ipv6         fe80::5054:ff:fe4b:735f/64
 eth2       52:54:00:4b:73:56    ipv4         192.168.122.232/24
 -          -                    ipv6         fe80::5054:ff:fe4b:7356/64

c610:~ #
```

UEFI guest installation

==========

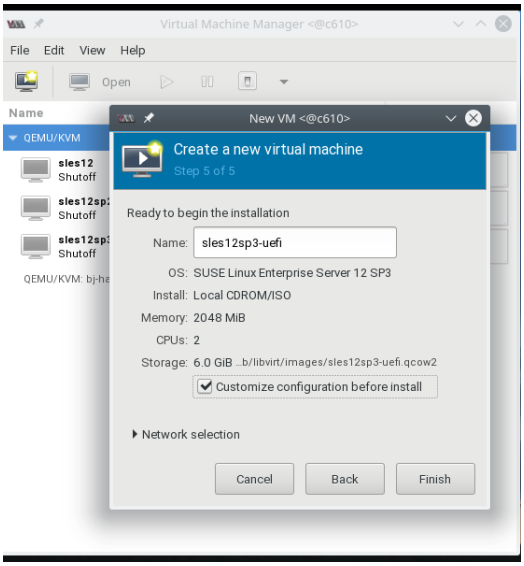
要求 host 上安装了用于虚拟化的 uefi 固件，也就是 qemu-ovmf 包

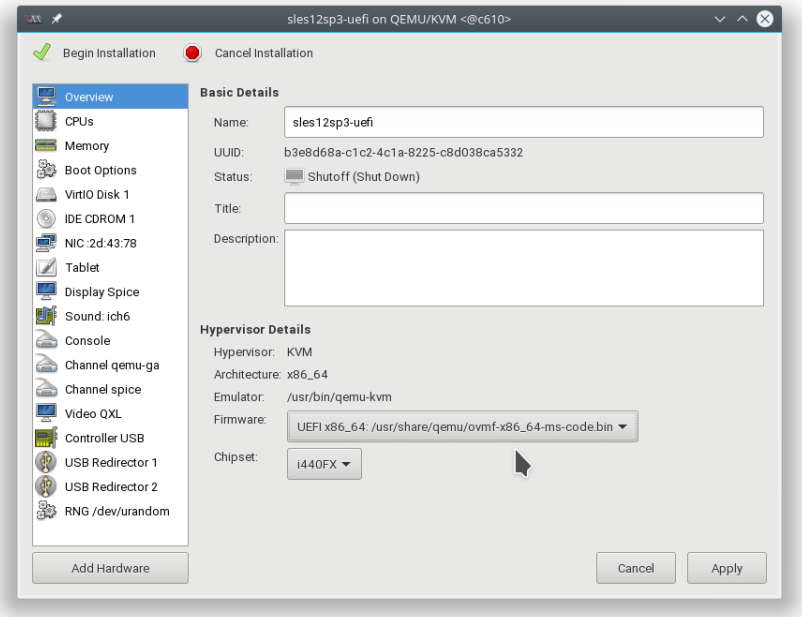
例 12:

对于 virt-install 命令行来安装基于 uefi 的虚拟机，有好几种语法，其中最便捷的是在'--boot'参数中加入'uefi'即可:

```
virt-install \
.....
--boot uefi \
.....
```

对于通过 virt-manager 的图形界面安装 uefi 虚拟机，则：





若在其他发行版上提示找不到 UEFI Firmware 时，可能需要用户手动编辑/etc/libvirt/qemu.conf 中的 nvram 段去显式指定 ovmf 的路径.

Easy qemu commandline passthrough with virt-xml and virt-install

==========

当用户在某些发行版上通过 libvirt 来管理 qemu/kvm 虚拟机时，若想使用一些 qemu 已经支持但该发行版的 libvirt 还暂不支持的虚拟设备、参数或环境变量时，就可以通过 libvirt 的 qemu commandline passthrough 来达到目的。

而 virt-manager 自带的 virt-install 和 virt-xml，从 1.4.1 开始,前者开始允许为新创建的虚拟机传入那些不支持的设备或参数,后者可以极方便地为已存在的虚拟机传入,无需用户像从前那样手动编辑虚拟机配置文件。

例 13:

```
c610:~ # virt-xml sles12sp3 --edit --confirm --qemu-commandline="-snapshot"
--- Original XML
+++ Altered XML
@@ -1,4 +1,4 @@
-<domain type="kvm">
+<domain xmlns:qemu="http://libvirt.org/schemas/domain/qemu/1.0" type="kvm">
   <name>sles12sp3</name>
   <uuid>5b415246-1b4f-482b-9a2a-23a3268d734b</uuid>
   <memory unit="KiB">4194304</memory>
@@ -108,4 +108,7 @@
       <address type="pci" domain="0x0000" bus="0x00" slot="0x09" function="0x0"/>
     </rng>
   </devices>
+  <qemu:commandline>
+    <qemu:arg value="-snapshot"/>
+  </qemu:commandline>
 </domain>

Define 'sles12sp3' with the changed XML? (y/n): y
Domain 'sles12sp3' defined successfully.
c610:~ #
```

上图中为 libvirt 管理的 sles12sp3 虚拟机配置中加入了一个 qemu 支持，但 libvirt 不支持的选项：'-snapshot'
该选项的含义是 用户登陆到虚拟机后，在里面无论做过任何更改，只要虚拟机 Power off，所有更改全部消失。
它与 readonly 是有本质区别的，readonly 是根本不允许写操作，而-snapshot 是虚拟机关机后所有变更立即失效。

注意，千万不能把-snapshot 通过 virt-install 传给虚拟机，否则你的虚拟机就白安装了 :-)

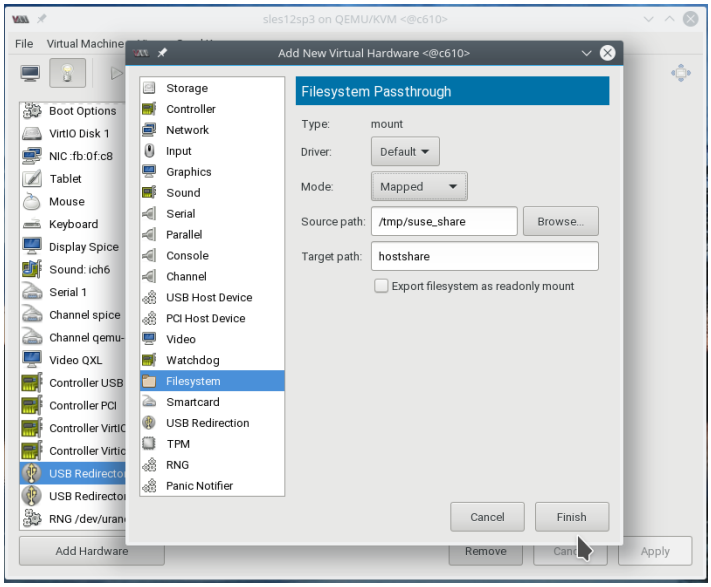
```
# virt-install \
--name vgpu_pretest \
--cpu core2duo \
--vcpus=2 \
--memory 2048 \
--os-variant=sles12sp3 \
--network model=virtio,bridge=br0 \
--pxe \
--disk pool=default,format=qcow2,size=8,bus=scsi \
--controller scsi,model=virtio-scsi \
--qemu-commandline="-display gtk,gl=on"
```

virtio 9p

======

KVM 的 VirtFS 使用半虚的文件系统驱动，避免了虚拟机应用程序->文件系统操作->块设备操作->再到 host 文件系统操作.VirtFS 使用 Plan-9 network protocol 在 guest 与 host 间进行通信。
VirtFS 可用于多 guest 间，或 host-guest 间共享文件。

作为 VirtFS server 的 qemu 通过两种虚拟设备配合来提供 VirtFS
-fsdev: 定义所导出的文件系统属性，如 type 和 security model.
-device virtio-9p: 在 guest 和 host 间传递协议消息和数据



在 guest 中挂载 host 导出的文件系统路径时，
若报告不支持，则需检查 guest kernel config.

```
CONFIG_NET_9P=y
CONFIG_NET_9P_VIRTIO=y
CONFIG_NET_9P_DEBUG=y (Optional)
CONFIG_9P_FS=y
CONFIG_9P_FS_POSIX_ACL=y
CONFIG_PCI=y
CONFIG_VIRTIO_PCI=y
```

例 14:

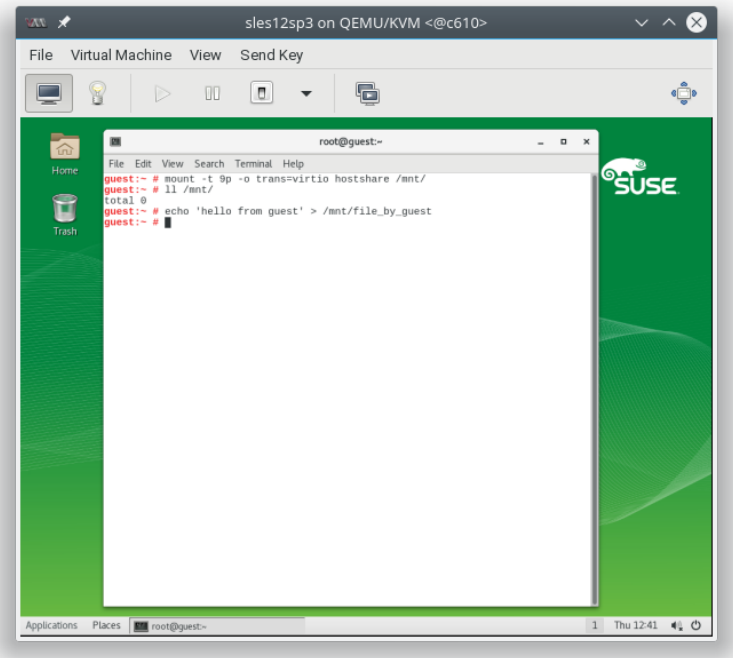
```
c610:/ # stat /tmp/suse_share/
  File: '/tmp/suse_share/'
  Size: 4096            Blocks: 8  ...
Device: 82bh/2091d      Inode: ...
Access: (0777/drwxrwxrwx)  Uid: (    0/    root)   Gid: (  100/   users)
...
```

若虚拟机配置中还未包含 virtio9p 文件系统，则要么参考上面的截图通过 virt-manager 添加，要么如下用命令行添加

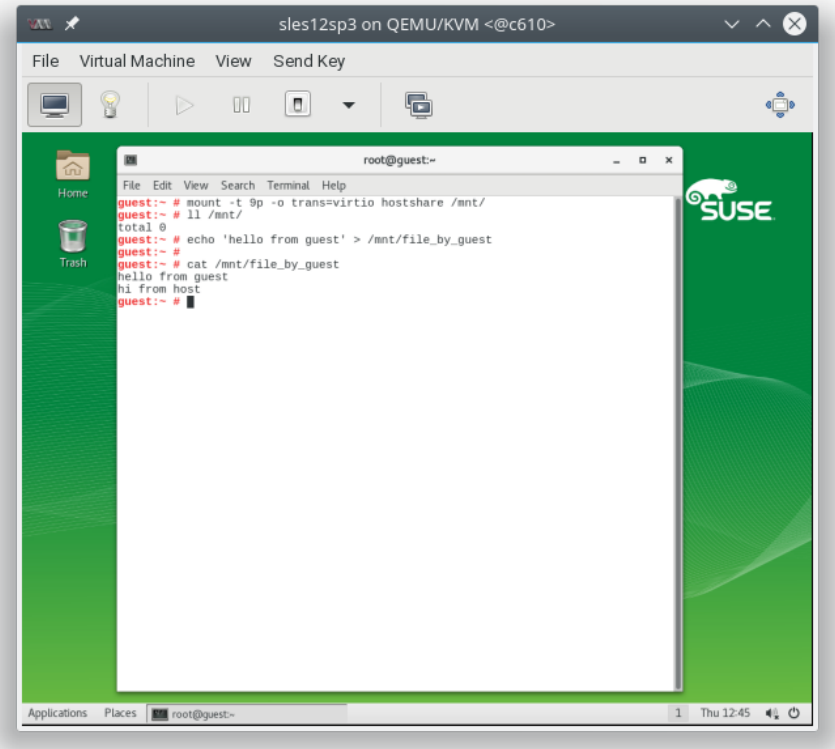
```
c610:/ # ll /tmp/suse_share/
total 0

c610:/ # cat /opt/vms/virtio9p.xml
<filesystem type='mount' accessmode='mapped'>
  <source dir='/tmp/suse_share'/>
  <target dir='hostshare'/>
  <alias name='fs0'/>
</filesystem>
```

'**mapped**',是 security_model 的可选值之一，取值范围是'mapped','passthrough','none'，但目前我还未完全弄清适用场景，留作日后分解吧
'**/tmp/suse_share**'就是在 host 上要被导出的文件系统路径
'**hostshare**'就是在 guest 中 mount 导出的文件系统时命令行里要使用的 mount tag

```
c610:/ # virsh attach-device sles12sp3 --file /opt/vms/virtio9p.xml -persistent

c610:/ # virsh dumpxml sles12sp3 | sed -n '/<filesystem/,/<\/filesystem/p'
    <filesystem type='mount' accessmode='mapped'>
      <source dir='/tmp/suse_share'/>
      <target dir='hostshare'/>
      <alias name='fs0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x0b' function='0x0'/>
    </filesystem>


c610:/ # virsh start sles12sp3
```




```
c610:/opt/vms # ll /tmp/suse_share/
total 8
-rw------- 1 qemu qemu 17 Mar 22 12:40 file_by_guest

c610:/opt/vms # cat /tmp/suse_share/file_by_guest
hello from guest
c610:/opt/vms #
c610:/opt/vms # echo 'hi from host' >> /tmp/suse_share/file_by_guest
```



Host USB device redirection

==========

'主机 usb 设备重定向'这个 feature 的常用场景是：
用户的业务布置在云端虚拟机，需要使用物理 usb key，最典型的就是网上银行的应用，会调 usb key 去做认证。
比如用户从本地 virt-manager 连接到远程的虚拟机，但其中的应用需要 usb key 去做认证，此时用户可以把手中的 usb key 插入到本地计算机，然后使用这个功能把设备重定向到远程虚拟机，就好像是把 usb key 插在了远程主机上并 assign 给虚拟机去使用的效果。

要求：远程虚拟机必须包含 USB Redirector 设备且 Display 设备必须是 Spice,而不可以是 VNC。
(virt-manager 默认创建的 SLES 虚拟机，这些条件都能满足)

如果远程的虚拟机是处于 running 状态，且**焦点**就在该虚拟机，这时只要用户在自己的主机上插入 usb 设备，则该设备就会自动被重定向到该虚拟机，直接可以使用了。
其他情况需用户手动执行重定向,并选择欲定向到远程虚拟机的本地 usb 设备，下面例子中，插入一个本地 u 盘做为粟子 :-)
例 15: