Libvirt+xen

**修改配置文件**

sudo gedit /etc/xen/xend-config.sxp

将/etc/xen/xend-config.sxp文件中的“#(xend-unix-server no)”注释去掉 即把“#”去掉，并把“no”改成“yes”，保存即可。 然后重启xend服务

sudo gedit ~/.bashrc ,在打开的.bashrc文件中添加下面这句话

export VIRSH\_DEFAULT\_CONNECT\_URI="xen:///"

**通过libvirt查看xen的版本信息，确认libvirt和xen都已经安装成功**

# virsh version

Compiled against library: libvir 0.9.8

Using library: libvir 0.9.8

Using API: Xen 0.9.8

Running hypervisor: Xen 4.1

<http://libvirt.org/drvxen.html>

# Xen hypervisor driver

* [Project Links](http://libvirt.org/drvxen.html#project)
* [Deployment pre-requisites](http://libvirt.org/drvxen.html#prereq)
* [Connections to Xen driver](http://libvirt.org/drvxen.html#uri)
* [Import and export of libvirt domain XML configs](http://libvirt.org/drvxen.html#imex)
  + [Converting from XM config files to domain XML](http://libvirt.org/drvxen.html#xmlimport)
  + [Converting from domain XML to XM config files](http://libvirt.org/drvxen.html#xmlexport)
* [Example domain XML config](http://libvirt.org/drvxen.html#xmlconfig)

The libvirt Xen driver provides the ability to manage virtual machines on any Xen release from 3.0.1 onwards.

## Project Links [¶](http://libvirt.org/drvxen.html#project)

* The [Xen](http://www.cl.cam.ac.uk/Research/SRG/netos/xen/index.html) hypervisor on Linux and Solaris hosts

## Deployment pre-requisites [¶](http://libvirt.org/drvxen.html#prereq)

The libvirt Xen driver uses a combination of channels to manage Xen virtual machines.

* **XenD**: Access to the Xen daemon is a mandatory requirement for the libvirt Xen driver. It requires that the UNIX socket interface be enabled in the /etc/xen/xend-config.sxp configuration file. Specifically the config settings (xend-unix-server yes). This path is usually restricted to only allow the root user access. As an alternative, the HTTP interface can be used, however, this has significant security implications.
* **XenStoreD**: Access to the Xenstore daemon enables more efficient codepaths for looking up domain information which lowers the CPU overhead of management.
* **Hypercalls**: The ability to make direct hypercalls allows the most efficient codepaths in the driver to be used for monitoring domain status.
* **XM config**: When using Xen releases prior to 3.0.4, there is no inactive domain management in XenD. For such releases, libvirt will automatically process XM configuration files kept in the /etc/xen directory. It is important not to place any other non-config files in this directory.
* **libxl**: Starting with Xen 4.2, the legacy XenD/xm toolstack is deprecated in favor of libxl, also commonly called libxenlight. libvirt supports this new Xen toolstack via the libxl driver. If XenD is enabled, the legacy xen driver consisting of the above mentioned channels will be used. If XenD is disabled, the libxl driver will be used.

## Connections to Xen driver [¶](http://libvirt.org/drvxen.html#uri)

The libvirt Xen driver is a single-instance privileged driver, with a driver name of 'xen'. Some example connection URIs for the libvirt driver are:

xen:/// (local access, direct)

xen+unix:/// (local access, via daemon)

xen://example.com/ (remote access, TLS/x509)

xen+tcp://example.com/ (remote access, SASl/Kerberos)

xen+ssh://root@example.com/ (remote access, SSH tunnelled)

## Import and export of libvirt domain XML configs

The Xen driver currently supports two native config formats. The first known as xen-xm is the format used by the XM tool for files in /etc/xen. The second known as xen-sxpr, is the format used for interacting with the XenD's legacy HTTP RPC service.

### Converting from XM config files to domain XML [¶](http://libvirt.org/drvxen.html#xmlimport)

The virsh domxml-from-native provides a way to convert an existing set of XM config files into a guest description using libvirt Domain XML that can then be used by libvirt.

$ virsh -c xen:/// domxml-from-native xen-xm rhel5.cfg

<domain type='xen'>

<name>rhel5pv</name>

<uuid>8f07fe28-753f-2729-d76d-bdbd892f949a</uuid>

<memory>2560000</memory>

<currentMemory>307200</currentMemory>

<vcpu>4</vcpu>

<bootloader>/usr/bin/pygrub</bootloader>

<os>

<type arch='x86\_64' machine='xenpv'>linux</type>

</os>

<clock offset='utc'/>

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

<devices>

<disk type='file' device='disk'>

<driver name='tap' type='aio'/>

<source file='/var/lib/xen/images/rhel5pv.img'/>

<target dev='xvda' bus='xen'/>

</disk>

<disk type='file' device='disk'>

<driver name='tap' type='qcow'/>

<source file='/root/qcow1-xen.img'/>

<target dev='xvdd' bus='xen'/>

</disk>

<interface type='bridge'>

<mac address='00:16:3e:60:36:ba'/>

<source bridge='xenbr0'/>

</interface>

<console type='pty'>

<target port='0'/>

</console>

<input type='mouse' bus='xen'/>

<graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'/>

</devices>

</domain>

### Converting from domain XML to XM config files [¶](http://libvirt.org/drvxen.html#xmlexport)

The virsh domxml-to-native provides a way to convert a guest description using libvirt Domain XML, into the XM config file format.

$ virsh -c xen:/// domxml-to-native xen-xm rhel5pv.xml

name = "rhel5pv"

uuid = "8f07fe28-753f-2729-d76d-bdbd892f949a"

maxmem = 2500

memory = 300

vcpus = 4

bootloader = "/usr/bin/pygrub"

kernel = "/var/lib/xen/boot\_kernel.0YK-cS"

ramdisk = "/var/lib/xen/boot\_ramdisk.vWgrxK"

extra = "ro root=/dev/VolGroup00/LogVol00 rhgb quiet"

on\_poweroff = "destroy"

on\_reboot = "restart"

on\_crash = "restart"

sdl = 0

vnc = 1

vncunused = 1

vnclisten = "0.0.0.0"

disk = [ "tap:aio:/var/lib/xen/images/rhel5pv.img,xvda,w", "tap:qcow:/root/qcow1-xen.img,xvdd,w" ]

vif = [ "mac=00:16:3e:60:36:ba,bridge=virbr0,script=vif-bridge,vifname=vif5.0" ]

## Example domain XML config [¶](http://libvirt.org/drvxen.html#xmlconfig)

Below are some example XML configurations for Xen guest domains. For full details of the available options, consult the [domain XML format](http://libvirt.org/formatdomain.html) guide.

### Paravirtualized guest bootloader

Using a bootloader allows a paravirtualized guest to be booted using a kernel stored inside its virtual disk image

<domain type='xen' >

<name>fc8</name>

<bootloader>/usr/bin/pygrub</bootloader>

<os>

<type>linux</type>

</os>

<memory>131072</memory>

<vcpu>1</vcpu>

<devices>

<disk type='file'>

<source file='/var/lib/xen/images/fc4.img'/>

<target dev='sda1'/>

</disk>

<interface type='bridge'>

<source bridge='xenbr0'/>

<mac address='aa:00:00:00:00:11'/>

<script path='/etc/xen/scripts/vif-bridge'/>

</interface>

<console tty='/dev/pts/5'/>

</devices>

</domain>

### Paravirtualized guest direct kernel boot

For installation of paravirtualized guests it is typical to boot the domain using a kernel and initrd stored in the host OS

<domain type='xen' >

<name>fc8</name>

<os>

<type>linux</type>

<kernel>/var/lib/xen/install/vmlinuz-fedora8-x86\_64</kernel>

<initrd>/var/lib/xen/install/initrd-vmlinuz-fedora8-x86\_64</initrd>

<cmdline> kickstart=http://example.com/myguest.ks </cmdline>

</os>

<memory>131072</memory>

<vcpu>1</vcpu>

<devices>

<disk type='file'>

<source file='/var/lib/xen/images/fc4.img'/>

<target dev='sda1'/>

</disk>

<interface type='bridge'>

<source bridge='xenbr0'/>

<mac address='aa:00:00:00:00:11'/>

<script path='/etc/xen/scripts/vif-bridge'/>

</interface>

<graphics type='vnc' port='-1'/>

<console tty='/dev/pts/5'/>

</devices>

</domain>

### Fullyvirtualized guest BIOS boot

Fullyvirtualized guests use the emulated BIOS to boot off the primary harddisk, CDROM or Network PXE ROM.

<domain type='xen' id='3'>

<name>fv0</name>

<uuid>4dea22b31d52d8f32516782e98ab3fa0</uuid>

<os>

<type>hvm</type>

<loader>/usr/lib/xen/boot/hvmloader</loader>

<boot dev='hd'/>

</os>

<memory>524288</memory>

<vcpu>1</vcpu>

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

<features>

<pae/>

<acpi/>

<apic/>

</features>

<clock sync="localtime"/>

<devices>

<emulator>/usr/lib/xen/bin/qemu-dm</emulator>

<interface type='bridge'>

<source bridge='xenbr0'/>

<mac address='00:16:3e:5d:c7:9e'/>

<script path='vif-bridge'/>

</interface>

<disk type='file'>

<source file='/var/lib/xen/images/fv0'/>

<target dev='hda'/>

</disk>

<disk type='file' device='cdrom'>

<source file='/var/lib/xen/images/fc5-x86\_64-boot.iso'/>

<target dev='hdc'/>

<readonly/>

</disk>

<disk type='file' device='floppy'>

<source file='/root/fd.img'/>

<target dev='fda'/>

</disk>

<graphics type='vnc' port='5904'/>

</devices>

</domain>

### Fullyvirtualized guest direct kernel boot

With Xen 3.2.0 or later it is possible to bypass the BIOS and directly boot a Linux kernel and initrd as a fullyvirtualized domain. This allows for complete automation of OS installation, for example using the Anaconda kickstart support.

<domain type='xen' id='3'>

<name>fv0</name>

<uuid>4dea22b31d52d8f32516782e98ab3fa0</uuid>

<os>

<type>hvm</type>

<loader>/usr/lib/xen/boot/hvmloader</loader>

<kernel>/var/lib/xen/install/vmlinuz-fedora8-x86\_64</kernel>

<initrd>/var/lib/xen/install/initrd-vmlinuz-fedora8-x86\_64</initrd>

<cmdline> kickstart=http://example.com/myguest.ks </cmdline>

</os>

<memory>524288</memory>

<vcpu>1</vcpu>

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

<features>

<pae/>

<acpi/>

<apic/>

</features>

<clock sync="localtime"/>

<devices>

<emulator>/usr/lib/xen/bin/qemu-dm</emulator>

<interface type='bridge'>

<source bridge='xenbr0'/>

<mac address='00:16:3e:5d:c7:9e'/>

<script path='vif-bridge'/>

</interface>

<disk type='file'>

<source file='/var/lib/xen/images/fv0'/>

<target dev='hda'/>

</disk>

<disk type='file' device='cdrom'>

<source file='/var/lib/xen/images/fc5-x86\_64-boot.iso'/>

<target dev='hdc'/>

<readonly/>

</disk>

<disk type='file' device='floppy'>

<source file='/root/fd.img'/>

<target dev='fda'/>

</disk>

<graphics type='vnc' port='5904'/>

</devices>

</domain>