KVM and libvirt

With the combination of KVM and libvirt, you have an easy way of creating and managing virtual machines.

According to the official homepage, libvirt is: A toolkit to interact with the virtualization capabilities of recent versions of Linux (and other OSes). It provides management of virtual machines, virtual networks and storage; both local and remote. Since libvirt acts as an intermediate between a hypervisor and client applications, you must have a supported hypervisor installed. Examples are: KVM/QEMU, Virtualbox, Xen and VMware ESX.

Quote: KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.

Installation

Libvirt can be installed using a slackbuild script from slackbuilds.org. It provides a daemon that interacts between applications and virtual machines. It also provides a command-line shell, *virsh*, that can be used to manage virtual machines and to configure the libvirt environment. Virsh can also be used in shell scripts to start and stop virtual machines.

The slackware kernel has the KVM module enabled. The libvirt startup script will check the CPU and modprobe the correct driver. User-space tools are supplied with QEMU, which is available from slackbuilds.org. Previously, a modified QEMU, *qemu-kvm*, was used. Since version 1.3 however, QEMU incorporates those changes and qemu-kvm is depricated.

A graphical desktop management tool, *virt-manager*, is also available on slackbuilds.org. This provides an overview of all virtual machines and has a nice wizard to create new virtual machines in an easy way.

Configuration

Automatic startup

If you want to have the libvirt daemon started automatically, add the following section to `/etc/rc.d/rc.local`:

```
# start libvirt
if [ -x /etc/rc.d/rc.libvirt ]; then
    /etc/rc.d/rc.libvirt start
fi
```

Make sure `/etc/rc.d/rc.libvirt` is executable.

Managing storage pools

Storage in libvirt is handled in terms of *storage pools* and *storage volumes*. A *pool* is a generic container for various storage objects. It can be a local directory, physical partition, or a network share. A storage *volume* is the virtual representation of a disk for a guest system. On the guest, this volume is seen as a local disk. An iso image of an installation cd or dvd is also considered a volume.

When libvirt is installed, a default storage pool (called *default*) is created with local directory `/var/lib/libvirt/images`. Any newly created volumes are created in this directory.

Create a new directory-based storage pool using virsh

Virsh commands can be passed as parameters to *virsh* on the command line, or you can start an interactive virsh shell by calling `virsh` without parameters :

```
# virsh
Welcome to virsh, the virtualization interactive terminal.

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

virsh #
```

To create a new directory-based storage pool, first make sure the target directory exists. Then use the `pool-define-as` command. The basic syntax for this command is : `pool-define-as <pool-name> dir - - - - <directory-name>`. For example, to create pool *disks* for directory `/srv/virtualmachines/disks`, use the following command:

```
# virsh pool-define-as disks dir - - - - /srv/virtualmachines/disks
Pool disks defined
```

For more complex examples of this command, check the man-page for virsh.

Check that the pool exists with the `pool-list` command. The `-all` option shows both active and inactive pools :

```
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
disks                inactive   no
```

Now, build the actual pool with the `pool-build` command :

```
# virsh pool-build disks
Pool disks built
```

When the pool is built, it can be started with the `pool-start` command :

```
# virsh pool-start disks
Pool disks started
```

Now the new pool can be used. At this point, the pool must always be started manually. In order for libvirt to start the pool when the daemon is started, you must check the autostart flag with the `pool-autostart` command:

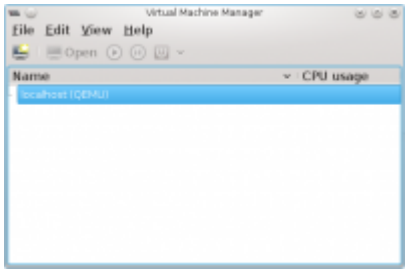
```
# virsh pool-autostart disks
Pool disks marked as autostarted
```

Display information about the pool with the `pool-info` command :

```
# virsh pool-info disks
Name:           disks
UUID:           4ae08c3d-4622-9f2a-cfa9-9dea4d1eb465
State:          running
Persistent:     yes
Autostart:      yes
Capacity:       697.92 GiB
Allocation:     250.89 GiB
Available:      447.04 GiB
```

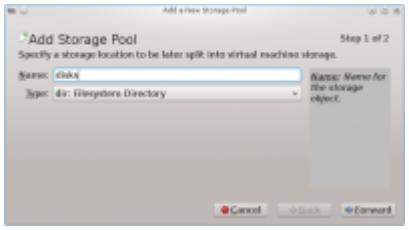
Create a new directory-based storage pool using virt-manager

First, make sure the target directory exists. Then start *virt-manager*.



Select the host machine (default is *localhost*). Select *Edit, Connection Details* from the menu, or right-click the machine and select *Details*, or double-click the machine. The *Connection Details* window appears. Select the *Storage* tab.

Press the + button on the bottom left. The *Add Storage Pool* window appears.

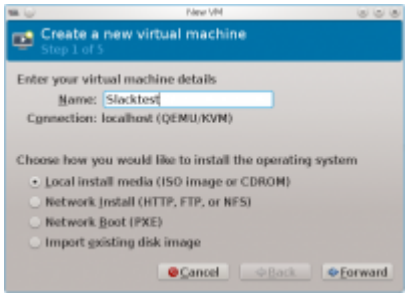


Enter the name of the new pool. The default type is *dir*, which is the correct type. Press Forward and enter the system directory in the *Target Path* entry field. Press Finish to create the pool.

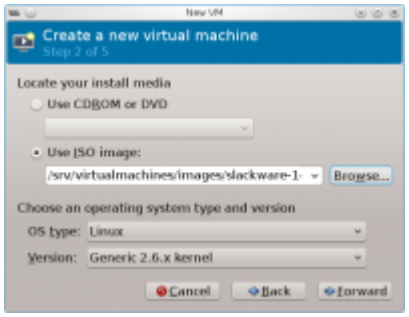
Creating a new virtual machine

Creating a new virtual machine using virt-manager

Select the host on which you want to create the new virtual machine. This will be *localhost* by default. Either right-click the host and select *New*, or click the Create button to start the Creation wizard.



Step 1: Name the new machine and select the method of OS installation and press Forward.

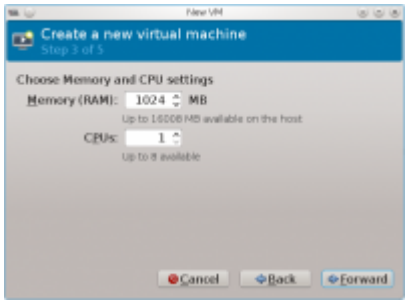


Step 2: Depending on the method of installation, you are asked here to enter more details, such as the name of the dvd iso image used.

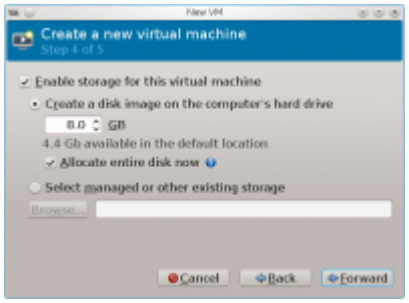
Also choose the OS type and version. When choosing *Linux*, choose *Show all OS options* at the Version prompt to get more options. When Choosing *Generic 2.6.x kernel*, the wizard will assume an IDE type hard disk. You can change this at step 5.

If you choose *Generic 2.6.25 or later kernel with virtio*, the new machine will use the virtio kernel driver to emulate a hard disk controller. The Slackware installation image will boot, but you need to create an initrd with the virtio kernel drivers included, otherwise your new machine won't boot.

Step 3: Select the amount of RAM and CPUs.



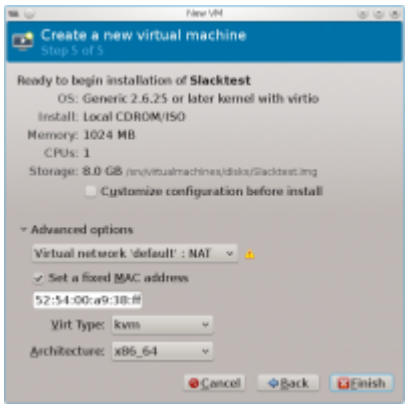
Step 4: Set up storage for the new machine. This option is checked by default. If you deselect this option, you will have a diskless machine that can still boot live distros.



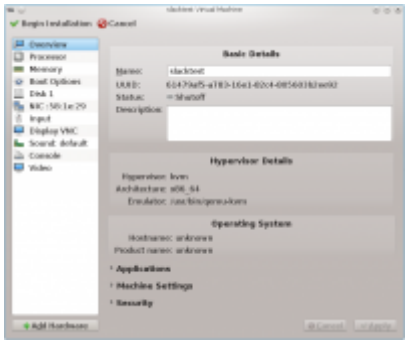
You can let the wizard create a new disk image on the fly. Enter the size and a new disk image (type *raw*) will be created in the default storage pool. If you select *Allocate entire disk now*, the full size of the new disk is allocated in advance. Otherwise, a smaller image is created that will grow when data is written to the disk.

Alternatively, you can select another disk image (volume) from a storage pool. You can also use this option to create a new storage volume. This method will give you more flexibility, because you have more options on the type of disk you can create. Lastly, you can use this option to browse the local filesystem and select a disk image you created earlier.

Step 5: This will give an overview of the new machine



If you check *Customize configuration before install*, you will be presented with the hardware configuration page of virt-manager. Here you can make changes to the machine before installing.



The default network option on step 5 is *Virtual network* '*default*' *: NAT*. This virtual network will use the *virbr0* bridge device to connect to the local host using NAT. Libvirt will start dnsmasq to provide DHCP for this network.

Finally, you can change the type of virtual machine (KVM or QEMU) and the architecture (x86_64, i686, arm, etc.). Press Finish to create the virtual machine. It will be created and booted.

Networking

A virtual machine can be set up in one of two ways: connected to a virtual network or connected to a shared physical device (bridged).

Virtual networks

Virtual networks are implemented by libvirt in the form of virtual switches. A virtual machine will have it's network card plugged into this virtual switch. The virtual switch appears on the host as a network interface.

The virtual network can operate in 3 different configurations:

1. NAT: the virtual switch is connected to the host LAN in NAT mode. The host can connect to the VM's but other machines on the host network cannot. This is the default configuration.
2. Routed: the virtual switch is connected to the host LAN without NAT.
3. Isolated: the virtual switch is not connected to the host. Virtual machines can see each other and the host, but network traffic does not pass outside the host.

Dnsmasq and iptables

DHCP and DNS on the virtual networks is handled by Dnsmasq. For every virtual network that is started, a separate instance of Dnsmasq is started by libvirt.

When starting a virtual network, libvirt will also add iptables rules to handle routing and NAT between the host and the virtual network. It will also enable ip_forward.

Default network

After installation, one network (appropriately called *default*) is already created. It is configured as

type NAT and this is the default network that is linked to new virtual machines. The network is setup to start automatically when libvirt is started, so it is immediately available when libvirt is running.

The network is visible on the host as bridge virbr0.

Do not add physical interfaces to this bridge. It is handled by libvirt.

Creation and maintenance

Virtual networks are visible on the *Virtual Networks* tab of the *Host Connection Details* window in Virtual Machine Manager. The + key can be used to add a new virtual network. Enter the name, IPv4 network address, DHCP range and the type of network.

Once the virtual network is created, it can be used in the virtual machine maintenance screens.

Shared folders using VirtFS

It is possible to share folders between the guest and host system using VirtFS (Plan 9 folder sharing over Virtio). This is available from kernel version 2.6.36 onwards. VirtFS is a passthrough filesystem, which means that a directory on the host can be directly accessed from the guest through the virtualization layer.

Prepare host kernel

Make sure the following options are set in the host kernel:

```
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
```

On the guest system, modules `9p`, `9pnet` and `9pnet_virtio` are needed. These should be available in the standard Slackware kernel.

Guest configuration

In virt-manager, edit the guest virtual machine and add new hardware. Select *Filesystem* and fill the required fields as shown below.

- Mode. Select one of the following:
  - Passthrough: the host share is accessed with the permissions of the guest user.
  - Mapped: the host share is accessed with the permissions of the hypervisor (QEMU process).
  - Squash: Similar to *passthrough* except that failure of privileged operations like *chown* are ignored. This makes a passthrough-like mode usable for people who run the hypervisor as non-root.
- Driver: use *Path*.
- Write Policy (only available for `path` driver): use *Default*.
- Source path = directory on the host which is shared.
- Target path = mount tag that is made available on the guest system. This doesn't have to be an existing path.

Option *Export filesystem as readonly mount* does what it suggests.

Mounting the share

To mount the filesystem with tag `hostshare` on the guest at `/mnt/share`, logon to the guest and use:

```
# mount -t 9p -o trans=virtio,version=9p2000.L hostshare /mnt/share
```

Now the /mnt/share folder is available and changes are visible on the host and the guest. When permission errors occur, try sharing the host directory with a different mode.

Remote access

Work in progress

Advanced topics

Mount qcow image using nbd

Raw disk images can be mounted outside the virtual machine using a loopback device. To mount other image types like qcow, the `qemu-nbd` command can be used, which comes with qemu-kvm. It relies on the nbd (network block device) kernel module.

Start by loading the kernel module. The only parameter is the maximum partitions to be accessed. If this parameter is omitted, the default value is 0, which means no partitions will be mapped.

```
# modprobe nbd max_part=8
```

This will create various new devices `/dev/nbdxx`. Now the disk image can be connected to one of them:

```
# qemu-nbd -c /dev/ndb0 slackware.img
```

This will create additional devices `/dev/nbd0pxx` for the partitions on the disk. Partitions are numbered sequentially starting with 1. You can use the nbd0 device to access the whole disk, or the nbd0pxx devices to access the partitions:

```
# fdisk /dev/nbd0
# mount /dev/nbd0p1 /mnt/hd
```

Make sure the virtual machine is not running when you mount the disk image. Mounting the disk of a running machine will damage it.

To remove the connection:

```
# qemu-nbd -d /dev/nbd0
```

Setup PXE boot in libvirt

To enable PXE booting for the guest machines, a PXE boot server and a TFTP server are needed. Libvirt can be configured to handle both internally. These configuration options are not available in virt-manager, so virsh must be used to set this up.

1. Create a directory `/tftpboot` and fill with the required files for the tftp boot service. See the article PXE: Installing Slackware over the network by AlienBOB for more details.
2. Stop the default network and edit the network definition:

```
# virsh net-destroy default
# virsh net-edit default
```

3. This will open the network configuration in a vi session. Add the *tftp* and *bootp* parameters in the *ip* section and save the file:

```
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <tftp root='/tftpboot' />
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254' />
      <bootp file='pxelinux.0' />
    </dhcp>
  </ip>
```

4. Now restart the network:

```
# virsh net-start default
```

Now the libvirt DHCP server will allow guests to PXE boot.

Troubleshooting

Remove password prompt in virt-manager

When you start virt-manager as a regular user, you may still be asked for the root password, even when you have setup the correct unix socket permissions (notification: “system policy prevents management of local virtualized systems”). This is because older versions of libvirt were using PolicyKit by default. Disable the use of PolicyKit by editing `/etc/libvirt/libvirtd.conf`. Uncomment the following options and change them to `none` :

```
auth_unix_ro = "none"
auth_unix_rw = "none"
```

Improve mouse movement

In graphics mode, the mouse movement can be erratic and difficult to change in the settings of your VM. To solve this, add a virtual tablet.

In *virt-manager*, open the VM by double-clicking on the machine and select the hardware info screen (the blue `i` button). Now, press +Add Hardware and select *Input*. Select Type *EvTouch USB Graphics Tablet* and press Finish. The next time the VM is started, the mouse movement is synchronized with your desktop cursor.

Change screen resolution to higher than 1024x768

The default emulated video card is of type Cirrus. This has a maximum resolution of 1024×768. The *vga* type can achieve a higher resolution, but for that to work, the X configuration in the guest OS needs to be changed as well.

To change this, open the VM and select the hardware info screen. Select the Video card and change the type to *vga*.

Start the VM. If a file `/etc/X11/xorg.conf` exists, change that. Otherwise create a new text file in `/etc/X11/xorg.conf.d` with extension `conf`, for example `/etc/X11/xorg.conf.d/monitor.conf`.

Add or change the following sections:

```
Section "Monitor"
        Identifier "Monitor0"
        HorizSync   30 - 80
        VertRefresh 40 - 90
EndSection

Section "Device"
        Identifier "Card0"
        Driver     "vesa"
```

```
Endsection

Section "Screen"
        Identifier "Screen0"
        Device     "Card0"
        Monitor    "Monitor0"
        SubSection "Display"
                Viewport 0 0
                Modes "1440x900"
        EndSubSection
EndSection
```

You can change the screen resolution (the *Modes* line) depending on your needs.

Resources

- Official pages for libvirt, virt-manager, QEMU, KVM.
- Red Hat Virtualization Administration Guide.

Sources

- This page originally written by Frank Donkers

howtos, kvm, libvirt, virtualization, virt-manager, qemu, author fdonkers