Libvirt basic configurations and methods

Introduction

This is actually a continuation of another article:
https://docs.slackware.com/howtos:emulators:libvirt_qemu_manage_install
And was intended just to give some basic steps to get libvirt working on Slackware 15 after installing qemu/libvirt/virt-manager. However, the topic is so generic that it's difficult to make it specific. And there are so many different settings that it's not possible to just say “do this”. So in the end this is just a few different ways to set up and use libvirt in Slackware 15, with an extra focus on libvirt with qemu+virt-manager. It's meant for libvirt/qemu/virt-manager, but it has some use outside that as well, and mostly just deals with libvirt settings (not qemu or virt-manager). I don't know how to use triple slash here, so I'll just use a name like qemu:session instead.

There are also notes in code brackets on this page in (paranthesis)

Summary

```
(as ROOT)
groupadd --system libvirt
groupadd --system kvm
usermod -aG libvirt,kvm username
chgrp kvm /dev/kvm
chmod g+rw /dev/kvm
```

Note. You need to log out of X and TTY and back in for group to take effect.
Go to the libvirt config file and uncomment the relevant options and set them as you prefer. Then start the libvirt daemons.

```
(as ROOT)
nano /usr/local/etc/libvirt/libvirtd.conf
unix_sock_group = "libvirt"
unix_sock_ro_perms = "0770"
unix_sock_rw_perms = "0770"
unix_sock_admin_perms = "0770"
auth_unix_ro = "polkit"
auth_unix_rw = "polkit
ctrl+x (save)
libvirtd -d
virtlogd -d
```

The above is an example, and fine if you are the only user of the computer and only member of the libvirt group. If not, you should use different settings.

Libvirt config, settings and method for Slackware

The above summary is more than enough (read. too much) to use libvirt/qemu/virt-manager actually. The above is enough to:

- run qemu:system
- run qemu:session
- start libvirt daemons as root
- start libvirt daemons as user
- to use qemu/kvm through virt-manager in all the above ways

Personally I only really care about running qemu:session through virt-manager and that's it. But that might not work for others due to networking needs (ex.qemu-vm-server). And if it's a multi user machine it's not a good idea to grant admin rights to everyone in the libvirt group. Some also might not want to give RW rights to kvm. There are many things to consider. Below are some.

libvirtd.conf

go to the libvirt config file and uncomment the relevant options and set them as you prefer.

```
nano /usr/local/etc/libvirt/libvirtd.conf
unix_sock_group = "libvirt"   (it makes sense to use libvirt group, if not
you also have to rewrite polkit rule to use virt-manager)
unix_sock_ro_perms = "0770"   (some suggest this could be 0777)
unix_sock_rw_perms = "0770"   (this is the most important, and decides who
can USE libvirt fully (socket setup etc))
unix_sock_admin_perms = "0770" (since bash doesn't care about polkit, this
setting decides, and allows group to start libvirt daemons, standard is
0700)
auth_unix_ro = "polkit"   (Polkit rule for virt-manager++)
auth_unix_rw = "polkit"   (Polkit rule for virt-manager++)
```

The above are internal libvirt settings, while polkit regulates who can use libvirt (sockets) through a GUI like virt-manager for example.
This is ok for a PC with one user where you are the only one in the libvirt group, but you might want to consider less and more strict settings and a different polkit policy. If you plan to also use LXC or other containers (with libvirt/virt-manager), you need to consider all this more carefully.

Polkit

```
ls /usr/share/polkit-1/rules.d/50-libvirt.rules
```

This is a standard policykit rule that comes with libvirt. What it says and does is allows users who are member of libvirt group to connect to libvirtd (through GUI, ex. virt-manager) to do so without typing a password. Obviously, different kind of polkit rules can be written instead, but I like that one as it is.

Libvirt daemons as root

```
ls -la /var/local/run/libvirt
srwxrwx---  1 root libvirt    0 Feb 18 18:55 libvirt-admin-sock=
srwxrwx---  1 root libvirt    0 Feb 18 18:55 libvirt-sock=
srwxrwx---  1 root libvirt    0 Feb 18 18:55 libvirt-sock-ro=
srwx------  1 root root       0 Feb 18 18:55 virtlogd-admin-sock=
srwx------  1 root root       0 Feb 18 18:55 virtlogd-sock=
```

The above is what we expect with a configuration file like the example. This is why you might want to consider other settings. If you start libvirtd -d and virtlogd -d as user it's different:

Libvirt daemons as user

Regular permissions (and internal libvirt.conf ones) allow you to run libvirt daemons properly as user: as USER (since libvirt is not in user PATH)

```
/usr/local/sbin/libvirtd -d
/usr/local/sbin/virtlogd -d
ls -la /var/run/user/$UID/libvirt/
srwx------  1 user user   0 Feb 18 19:12 libvirt-admin-sock
srwx------  1 user user   0 Feb 18 19:12 libvirt-sock
-rw-r--r--  1 user user   4 Feb 18 19:12 libvirtd.pid
srwx------  1 user user   0 Feb 18 19:13 virtlogd-admin-sock
srwx------  1 user user   0 Feb 18 19:13 virtlogd-sock
-rw-r--r--  1 user user   4 Feb 18 19:13 virtlogd.pid
```

note. can't open qemu:system only qemu:session
note. libvirt can't read/write anything not accessible to user usergroup

kvm

Running libvirt daemons as user and qemu:session depends on the step mentioned above involving kvm group and /dev/kvm g+rw. If you want to run things in this way or a similar way you have to write a udev rule to make /dev/kvm group and g+rw permanent.

The most common way to run the daemons with kvm/qemu and virt-manager is to run the daemons (on boot) as root and connect to them as user (through libvirt group) and set up a qemu:session. To do that you also need the abovementioned kvm group and settings.

/dev/dri

The same that applies to kvm also applies to video cards. To solve this, you ONLY need to add the “video” group to your user, if that's the solution you want and think it is a good idea.

```
usermod -aG video username
```

You need to log out of X/wayland and tty, and back in, with your user for this change to take effect.

/etc/rc.d

To start libvirtd and virtlogd on boot (as root), you can add them to /etc/rc.d/rc.local:

```
/usr/local/sbin/libvirtd -d
/usr/local/sbin/virtlogd -d
```

udev

To change /dev/kvm group and permissions on boot, add a file /etc/udev/rules.d/90-kvm-override.rules and add the following content:

```
# /etc/udev/rules.d/90-kvm-override.rules
#
# Udev rule to override kvm group and permissions:
# To allow kvm group read/write access to /dev/kvm
#

# Set kvm device writable by kvm group
KERNEL=="kvm", GROUP:="kvm", MODE:="0660"
```

Once you reboot you can verify the changes:

```
ls -la /dev/kvm
crw-rw---- root kvm
```

Appendage

This article is not really done yet

Sources

* Originally written by zeebra

howtos,, emulators,, qemu,, libvirt,, kvm,, virt-manager