# Libvirt, QEMU, Virt-Manager install guide

## Introduction

QEMU/KVM with Virt-Manager is a great alternative to Virtualbox. Virt-Manager depends on libvirt, so overall this alternative far exceeds Virtualbox, and it is not difficult to get this working. This is suppose to be a clear and reproducible path to get Virt-Manage with all those components installed on Slackware 15.0 and beyond. It is suppose to be newbie friendly “step-by-step”. QEMU-6.2.0 and libvirt-8.0.0 can both be built and installed from source on Slackware 15.0 out of the box without any issues, but might be missing some functions you would want. This guide could also be a reference to some of those optional dependencies.

## Requirements

Be sure to use **Slackware 15.0** (full install) and **Kernel-generic** to accurately be able to reproduce this build and as a reference point. It should work on others, but use kernel-generic to be sure. Your processor should be capable of virtualization, otherwise you can only use QEMU as a type 2 hypervisor (which is probably not what you want). You can check virtualization capability with:

```
lscpu | grep -i virtualization
```

It should give an answer like:

```
 Virtualization:                  VT-x
```

For amd it will give another answer like “AMD-v”.

## Preliminaries

If you try to just build QEMU or Libvirt on Slackware 15, you will find out this information: (not complete)
qemu says:

- Run-time dependency libnfs found: NO / libnfs support: NO
- Run-time dependency spice-protocol found: NO / spice protocol support: NO\
- Run-time dependency spice-server found: NO
- Run-time dependency virglrenderer found: NO / virgl support: NO
- vde support: NO
- usb net redir: NO
- slirp support: NO (since QEMU 7.2)

libvirt says:

- Library numa found: NO (numactl: NO)

- Run-time dependency yajl found: NO (yajl: NO)

Note. You can test this by going to the QEMU and libvirt directories and running ./configure and meson configure.

# Before getting started

**Please familiarize yourself with this article and this method to avoid making a mess:**
**https://docs.slackware.com/howtos:slackware_admin:building_a_package**

**A few things to consider before proceeding:**

- Audit support (general) - if you habitually use this and/or might need it, better to do it now before kvm/qemu/libvirt
- Gstreamer codecs (spice) - libav and gst-plugins-ugly support, better to deal with general codec stuff before qemu/spice etc, in particular those 2
  - https://gstreamer.freedesktop.org/src/gst-plugins-ugly
  - https://gstreamer.freedesktop.org/src/gst-libav/
- Compiling Qemu with only the machines you need (unless you need alot of exotic CPU emulation and/or cross-compilation)

# Methodology (read this carefully)

1. The norm for Slackware is to build packages as root on a disposable machine installation (for example a virtual machine, or a build computer)
2. This guide uses /opt/root/build directory and /opt/fakeroot as build destination (make DESTDIR=)
3. Double check this article to know what you are doing: Building a Slackware package
4. This guide assumes you use the same methods
5. This guide uses user for make and root for make install
6. Whenever it says “(as ROOT)” you must continue in the same folder you were in (su)
7. Whenever it says “(as USER)” or “(as ROOT)”, it is a note, there are also some other notes in (paranthesis) inside code brackets

It is possible to do the whole process (except installpkg, ps. not tested..) as user and install into a user “fakeroot”, but you MUST ensure correct owner, group, permissions after with chown/chgrp -R /dir/fakeroot after. Do so at your own risk if you know what you're doing. If you are on a non-disposable environment you probably should. But for ease and safety this guide does not.

Since there are quite alot of packages here, I can't be bothered to package them 1 and

1, so I rather package them together in “stages”, based on dependencies. It's much easier to do and deploy, but you can package them individually.

## Packages, dependencies, order

The below are just links to the project download pages and an overview of the build order. These versions have not been tested, but are all the dependencies of QEMU with libvirt and virt-manager. You just have to build them roughly in this order and you're good to go.

| Stage 1 | Stage 2 | Stage 3 | Stage 4 | Stage 5 | Stage 6 |
|---|---|---|---|---|---|
| libosinfo | dtc | libvirt | libvirt-glib | QEMU | virt-manager |
| osinfo-db-tools | libnfs | | libvirt-python | | |
| spice-protocol | osinfo-db | | gtk-vnc | | |
| | numactl | | spice-gtk | | |
| | spice-server | | libslirp | | |
| | usbredir | | | | |
| | vde-2 | | | | |
| | virglrenderer | | | | |
| | yajl | | | | |

## Building, packaging, installing

#### Prepare

```
cd /opt/root/build
mkdir stage1 stage2 stage3 stage4 stage5 stage6
```

To make it all reproducible, the packages in all stages are the exact versions used. Process 1 is both a step to do immediately and a reference point for later

### Stage 1

(hard dependencies of stage2)
https://gitlab.com/libosinfo/libosinfo/-/tags/v1.9.0 (dependency of virt-manager)
https://gitlab.com/libosinfo/osinfo-db-tools/-/tags/v1.9.0 (dependency of virt-manager and osinfo-db)
https://gitlab.com/spice/spice-protocol/-/tags/v0.14.3 (dependency of spice)
Hint. the download button is on the upper right side of the screen.

Download and put the packages in your stage1 build directory.

#### Prepare

```
(as USER)
cd /opt/root/build/stage1/
tar -xvf libosinfo-v1.9.0.tar.gz && tar -xvf osinfo-db-tools-v1.9.0.tar.gz
&& tar -xvf spice-protocol-v0.14.3.tar.gz
```

The below process “process 1” is used many times in this document. Instead of writing it again, just follow the steps in “process 1” again, but change the name of the relevant package

#### Process 1

### libosinfo-v1.11.0

```
(as USER)
cd libosinfo-v1.11.0
meson configure --prefix=/usr/local
meson build
ninja -C build
(as ROOT)
DESTDIR=/opt/fakeroot ninja -C build install
```

**osinfo-db-tools-v1.9.0** (follow process 1)
**spice-protocol-v0.14.3** (process 1)
Note. spice-protocol comes without install instructions, and acts strange, but you can still follow process 1, it will work correctly. (ignore the warnings)

(as ROOT still)

```
cd /opt/fakeroot
mkdir -p usr/lib64/girepository-1.0
mv usr/local/lib64/girepository-1.0/* usr/lib64/girepository-1.0/
rm -r usr/local/lib64/girepository-1.0
```

**process 2** (see link above “building_a_package”)

```
strip -s usr/local/bin/* usr/local/lib64/*     (you can skip this)
gzip -9 usr/local/share/man/man1/*             (you can skip this)
mkdir install
nano install/slack-desc
virt-stage1: virt-stage1 (stage1 dependencies for libvirt/qemu/virt-manager)
virt-stage1:
```

```
virt-stage1: libosinfo-v1.9.0 and osinfo-db-tools-v1.9.0, spice-
protocol-0.15.0
virt-stage1: dependencies for stage 2.
virt-stage1:
ctrl+x "save"
```

You should probably first check folder and file permissions (for the below step) by running: (hint. it should all be root root)

```
ls -lhaR
```

then

```
makepkg ../virt-stage1_zeebra_s15_64_v1.0.txz (you can name it something
else if you want, ending with .txz)
```

-You will be asked about script/symbolic links, answer “y” for yes
-You will be asked about setting safe permissions, answer “n” for no (if you checked and find no problem)

```
mkdir /opt/slackpacks
cp ../virt-stage1_zeebra_s15_64_v1.0.txz /opt/slackpacks
cd /opt/slackpacks
installpkg virt-stage1_zeebra_s15_64_v1.0.txz
```

if something went wrong, you can use:

```
removepkg virt-stage1_zeebra_s15_64_v1.0.txz
```

and try whatever necessary steps again.

#### cleanup

```
cd /opt/
mv fakeroot/ stage1-fakeroot
mkdir fakeroot
```

## Stage 2

(some are hard dependencies of stage/step 4 and 6)
https://github.com/dgibson/dtc/releases/tag/v1.6.1 (can be built after qemu according to xiling and can be dropped unless you find you need it after)
https://github.com/sahlberg/libnfs/releases/tag/libnfs-5.0.1 (optional dependency of qemu, may be optional dependency of libvirt)
https://gitlab.com/libosinfo/osinfo-db/-/tags/v20210202 (should be built/imported after osinfo-db-tools, libosinfo is c wrapper for those things)
https://github.com/numactl/numactl/releases/tag/v2.0.14 (optional dependency qemu AND libvirt and possibly some other packages)

https://www.spice-space.org/download/releases/spice-server/ (indirect link - spice is the default display mode of virt-manager for qemu and so a dependency of it)
https://www.spice-space.org/download/releases/spice-server/spice-0.15.0.tar.bz2 (direct link)
https://gitlab.com/spice/usbredir/-/tags/usbredir-0.9.0 (mentioned by several other packages as I remember, hard dependency of some, optional qemu)
https://github.com/virtualsquare/vde-2 (since vde2 release is ancient, I take master from today 06-feb-2022, I would say this is an important function in qemu, but optional )
https://github.com/freedesktop/virglrenderer/releases/tag/0.9.1 (I would say this is an important function in qemu, but optional)
https://github.com/lloyd/yajl/releases/tag/2.1.0 (it would be best to skip this, but for now we include it, libvirt optional dependency)

```
(as USER)
cd /opt/root/build/stage2
```

unpack all the files in stage2

```
tar -xvf dtc-1.6.1.tar.gz && tar -xvf libnfs-libnfs-5.0.1.tar.gz && tar -xvf
numactl-2.0.14.tar.gz && tar -xvf osinfo-db-v20210202.tar.gz && tar -xvf
spice-0.15.0.tar.bz2 && tar -xvf usbredir-usbredir-0.9.0.tar.gz && unzip
vde-2-master.zip && tar -xvf virglrenderer-0.9.1.tar.gz && tar -xvf
yajl-2.1.0.tar.gz
```

So, for the sake of THIS howto, the “GNU process” is as following:

#### GNU Process

```
(as USER)
cd packagedir
./configure --libdir=/usr/local/lib64
make
(as ROOT)
make DESTDIR=/opt/fakeroot install
```

So, every time it says the “GNU Process”, follow those steps. But there are exceptions, but those will be noted below the package. So you will need to do something additional, but it is noted how and where. You could also read README and INSTALL.. It is probably easier to do ALL the USER steps first (building), then do all the ROOT steps (DESTDIR fakeroot) after. If unsure you can do step-by-step.

alphabetical: (ls -la)
**dtc-1.6.1** (process 1)
**libnfs-5.0.1** (GNU process)
exception!! you need to run ./bootstrap before ./configure
**numactl-2.0.14** (GNU process)
exception!! you need to run ./autogen.sh before ./configure

**osinfo-db-v20210202**

```
(as USER)
cd osinfo-db-v20210202
make -j3 (there is alot to compile here, -j3 is 3 cores, use another number
for more/less)
(as ROOT)
mkdir /opt/fakeroot/run
ls (find the name of the osinfo-db-$VERSION.tar.xz database and use THAT in
YOUR case, in my example, I'm using the name it tells ME)
cp osinfo-db-20220206.tar.xz /opt/fakeroot/run
cd /opt/fakeroot
mkdir install
nano install/doinst.sh
( osinfo-db-import --local /run/osinfo-db-20220206.tar.xz )
( osinfo-db-validate )
ctrl+x (save)
```

(yes, the exact 2 lines above should be the content of the file, except the name variable)

```
cd /opt/root/build/stage2/
```

**spice-0.15.0** (GNU process)
Attention!! Spice also mentions 2 gstreamer codecs. You might want to install those and others before doing libvirt/qemu/virt-manager. You can also do that later and rebuild spice after that.
**usbredir-usbredir-0.9.0** (GNU process)
Exception!! You need to run ./autogen.sh before ./configure
**vde-2-master** (GNU process)
exception!! you need to run autoreconf –install before ./configure
**virglrenderer-0.9.1** (process 1)
**yajl-2.1.0**

```
(as USER)
cd yajl-2.1.0
./configure
make
(as ROOT)
make DESTDIR=/opt/fakeroot install
```

yajl doesn't obey –libdir unless done with cmake so do:

```
mv /opt/fakeroot/usr/local/lib/libyajl* /opt/fakeroot/usr/local/lib64/
rm -r /opt/fakeroot/usr/local/lib
```

Our stage2 package is now in /opt/fakeroot
Repeat

#### Process 2

```
cd /opt/fakeroot
strip -s usr/local/bin/* usr/local/lib64/*
gzip -9 usr/local/share/man/man1/* usr/local/share/man/man2/*
usr/local/share/man/man3/* usr/local/share/man/man8/*
nano install/slack-desc
virt-stage2: virt-stage2 (stage 2 dependencies of qemu/libvirt/virt-manager)
virt-stage2:
virt-stage2: This is a collection of tools and libraries required and/or
necessary
virt-stage2: for proper qmeu/libvirt/virt-manager implementation on
Slackware 15.
virt-stage2: dtc-1.6.1, libnfs-5.0.1, numactl-2.0.14, osinfo-db-v20210202,
virt-stage2: spice-0.15.0, usbredir-0.9.0, vde-2-master-06-feb-2022,
virt-stage2: virglrenderer-0.9.1, yajl-2.1.0.
virt-stage2: It's the 2nd stage of a series of dependencies.
virt-stage2:
ctrl+x (save)
ls -lhaR
makepkg ../virt-stage2_zeebra_s15_64_v1.0.txz
```

-You will be asked about symbolic links, answer “y” for yes
-You will be asked about setting safe permissions, answer “n” for no (if you checked and find no problem)

```
cp ../virt-stage2_zeebra_s15_64_v1.0.txz /opt/slackpacks
cd /opt/slackpacks
installpkg virt-stage2_zeebra_s15_64_v1.0.txz
```

( **process 2 ends here** )
You should NOT get an error like this: (Error when getting information for file “/usr/local/etc/osinfo”: No such file or directory), if you do, something went wrong with osinfo database loading, and you can load it manually (see doinst.sh) or fix your package.

the below simply means you extract qemu package to some directory, and run ./configure in the directory to check the output

after installing this package, we can go to qemu testdir and test with ./configure, we expect this:
libnfs support: Yes
virgl support: Yes
vde support: Yes
spice protocol support: Yes
spice server support: Yes
usb net redir: Yes

we can do the same for libvirt with “meson build” command and we expect this:
numactl: Yes
yajl: Yes

If that's not the case, something went wrong, you can clear /opt/fakeroot/usr and do DESTDIR install stuff again from each build folder or rebuild any problematic software and do DESTDIR install again for all. Then repeat process 2.

If everything went fine, do some cleaning:

```
mv /opt/fakeroot /opt/stage2-fakeroot
mkdir /opt/fakeroot
```

## Stage 3

https://libvirt.org/sources/
https://libvirt.org/sources/libvirt-8.0.0.tar.xz

**libvirt-8.0.0** (process 1)
Then..

```
(as ROOT)
cd /opt/fakeroot
mkdir -p /opt/fakeroot/usr/share/polkit-1/actions
mkdir /opt/fakeroot/usr/share/polkit-1/rules.d
mv /opt/fakeroot/usr/local/share/polkit-1/actions/org.libvirt*
/opt/fakeroot/usr/share/polkit-1/actions/
mv /opt/fakeroot/usr/local/share/polkit-1/rules.d/50-libvirt.rules
/opt/fakeroot/usr/share/polkit-1/rules.d/
rm -rf /opt/fakeroot/usr/local/share/polkit-1
```

**Then repeat process 2 to package and install this**. Adapt it to this package. You can use this: (for this package)

```
strip -s usr/local/bin/* usr/local/lib64/*
gzip -9 usr/local/share/man/man1/* usr/local/share/man/man7/*
usr/local/share/man/man8/*
```

When done with process 2, clean:

```
cd /opt
mv fakeroot/ libvirt-fakeroot
mkdir fakeroot
```

## Stage 4

(some are hard dependencies of stage/step 6)
https://gitlab.com/libvirt/libvirt-glib/-/tags/v4.0.0 (gobject, gconfig, language bindings, requirement of virt-manager)
https://github.com/libvirt/libvirt-python/releases/tag/v8.0.0 (language bindings, specifically mentioned as build time requirement by virt-manager)

https://download.gnome.org/sources/gtk-vnc/1.3/ (vnc widget“ not sure exactly what this does here, but core c library and python bindings for virt-manager)
https://www.spice-space.org/download/gtk/ (indirect link) (same purpose as the above)
https://www.spice-space.org/download/gtk/spice-gtk-0.39.tar.xz (direct link)

```
(as USER)
cd /opt/root/build/stage4
tar -xvf gtk-vnc-1.3.0.tar.xz && tar -xvf libvirt-glib-v4.0.0.tar.gz && tar
-xvf libvirt-python-8.0.0.tar.gz && tar -xvf spice-gtk-0.39.tar.xz
```

alphabetical (ls -la)
**gtk-vnc-1.3.0** (process 1)
**libvirt-glib-v4.0.0** (process 1)
**libvirt-python-8.0.0**

```
(as USER)
cd libvirt-python-8.0.0
python3 setup.py build
(I find python messy, so I will do install as user, if you don't, skip to OR
as ROOT)
mkdir ../fakeroot
python3 setup.py install --root=/opt/root/build/stage4/fakeroot/
(as ROOT)
cd /opt/root/build/stage4/
chown -R root fakeroot/
chgrp -R root fakeroot/
cd fakeroot/usr/
mv lib64/ /opt/fakeroot/usr/
cd /opt/root/build/stage4/
rm -rf fakeroot
```

```
(OR as ROOT)
(python3 setup.py install --root=/opt/fakeroot)
```

**spice-gtk-0.39** (process 1)
Attention: when running “meson build” expect due to stage2 to see this “usbredir: YES”

```
cd /opt/fakeroot
mkdir -p usr/lib64/girepository-1.0
mv usr/local/lib64/girepository-1.0/* usr/lib64/girepository-1.0/
rm -r usr/local/lib64/girepository-1.0
```

**Then repeat process 2 to package and install this.**

Then clean:

```
cd /opt
mv fakeroot/ stage4-fakeroot
mkdir fakeroot
```

## Stage 5

https://www.qemu.org/ (indirect link)
https://download.qemu.org/qemu-6.2.0.tar.xz (direct link)

QEMU steps will work regardless and doesn't depend on any of these stages. Just not with those wanted functions included in the stages.. QEMU takes a long time to compile unless you compile it with less machines. You should use all processor cores here, so use “make -j4” or however many you have instead of just “make”.

**qemu-8.0.0** (GNU Process)

**Then repeat process 2 to package and install this.**

makepkg will take alot of memory for this, so if you have 4gb of ram or less it could be an issue and you could get a compressor error with makepkg (you should close your web browser before doing this step). If you have an old machine with 2gb or less, it might not be possible to do this step unless you compile this package with only a few machines like x86, i386 and arm

Clean:

```
cd /opt
mv fakeroot qemu-fakeroot
mkdir fakeroot
```

If you want to build QEMU with less targets, at your own risk, you can use ./configure –target-list=target(s). Targets can be found from build directory with

```
ls configs/targets
```

example:”./configure –target-list=x86_64-softmmu,i386-softmmu,arm-softmmu“

## Stage 6

### Virt-Manager-3.2.0

```
as USER
cd Virt-Manager-3.2.0
python3 setup.py build
(I find python messy, so I will do install as user, if you don't, skip to OR
as ROOT)
mkdir ../fakeroot
python3 setup.py install --root=/opt/root/build/stage6/fakeroot/
as ROOT
```

```
mkdir /opt/fakeroot/usr
cd /opt/root/build/stage6/
chown -R root fakeroot/
chgrp -R root fakeroot/
cd fakeroot/usr/
mv bin/ share/ /opt/fakeroot/usr/
cd /opt/root/build/stage6/
rm -rf fakeroot
```

```
(OR as ROOT)
(python3 setup.py install --root=/opt/fakeroot)
```

```
cd /opt/fakeroot
mkdir install
nano install/doinst.sh
( rm /home/user/.config/dconf/user )
ctrl+x (save)
```

Note. You don't have to do the above step with doinst.sh, but it will make sure you will not encounter a crash inside virt-manager when you use the function “browse local”. You can do this step manually after instead if you have that error.

**Then repeat process 2 to package and install this.**

## Appendage

### Other packages to consider:

- acpica (acpi tables for qemu)
- ovmf (firmware UEFI stuff for qemu)
- snappy (particular compression algo for qemu)
- gst-plugin-ugly (codec for spice)
- gst-libav (gstreamer codec for spice)


These things have been tested on the build Slackware 15.0, and the resulting packages have also been tested on a clean Slackware 15.0 installation. Most functions have been tested, but not connecting to VM over network instead of local (vnc), and not import of images from other VM environments (I don't have) like virtualbox, vmware etc. Everything tested was found to work as expected. If you follow the instructions, it includes solutions to debugged things, everything should work as expected.

**If you want configure and set up libvirt/qemu/kvm and virt-manager on Slackware 15.0 use the info in this link:**
https://docs.slackware.com/howtos:emulators:libvirt_config_methods

**Cleaning**

Once you verify that everything is working, you might want to clean up and delete all the fakeroots (/opt/stage1-fakeroot etc) and sources (/opt/root/build/stage1 etc). You only need to keep the resulting slackpacks (/opt/slackpacks)

# Sources

* Originally written by zeebra

Thanks to truepatriot76 for already having tested a working build and providing a list of dependencies!
Discussions here:
https://www.linuxquestions.org/questions/slackware-14/virt-manager-anyone-got-a-clear-and-working-path-to-this-4175707508/

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