ABSOLUTELY NO WARRANTY | free software

Getting KVM to run on your machine

Prerequisites

You will need (see download section):

- kvm-release.tar.gz
- A VT capable Intel processor, or an SVM capable AMD processor
- **qemu prerequisites:**

  o *zlib* libraries and headers o *SDL* libraries and headers o *alsa* libraries and headers (optional alsa support: disabled by default but can be enabled with --enable-alsa) o *gnutls* libraries and headers (optional VNC TLS support: enabled by default but can be disabled with --disable-vnc-tls) o kernel headers (on Fedora, the kernel-devel package)

On a debian etch system you can install the prerequisites with:

```
apt-get install gcc libsdl1.2-dev zlib1g-dev libasound2-dev
linux-kernel-headers pkg-config libgnutls-dev libpci-dev
```

Note: When building from git, you also need gawk.

Please report problems (and successes) to the mailing list. [edit]
Unpacking and configuring kvm components

You may wish to take a look at the ["Kernel-optimizations"] page. There exists a [attachment:kvm-26-alt-grab.diff.gz patch] which will change the SDL keygrab combination from ctrl-alt to ctrl-alt-shift. It was written primarily to deal with the heavy use of ctrl-alt-delete in NT-based VMs.

If you are using a patched kernel (e.g. a recent -mm kernel or the kvm git tree), configure the kernel normally, boot into it, and:

```
tar xzf kvm-release.tar.gz
cd kvm-release
./configure --prefix=/usr/local/kvm --with-patched-kernel
make
```

```
sudo make install
sudo /sbin/modprobe kvm-intel
# or: sudo /sbin/modprobe kvm-amd
```

If you're not running a patched kernel:

```
tar xzf kvm-release.tar.gz
cd kvm-release
./configure --prefix=/usr/local/kvm
make
sudo make install
sudo /sbin/modprobe kvm-intel
# or: sudo /sbin/modprobe kvm-amd
```

Note: if sound doesn't play in the guest vm you can add --audio-drv-list="alsa oss" to ./configure as explained in http://www.linux-kvm.com/content/sound-problem-solved

Creating a disk image for the guest:

```
/usr/local/kvm/bin/qemu-img create -f qcow vdisk.img 10G
```

Installing a guest operating system:

```
 sudo /usr/local/kvm/bin/qemu-system-x86_64 -hda vdisk.img -cdrom
/path/to/boot-media.iso \
-boot d  -m 384
```

(kvm doesn't make a distinction between i386 and x86_64 so even in i386 you should use *qemu-system-x86_64*)BR

If you have less than 1GB of memory don't use the -m 384 flag (which allocates 384 MB of RAM for the guest). For computers with 512MB of RAM it's safe to use -m 192, or even -m 128 (the default)

Running the newly-installed guest:

```
sudo /usr/local/kvm/bin/qemu-system-x86_64 vdisk.img -m 384
```

or a slightly more complicated example, where it is assumed that bridged networking is available on tap0; see ["Kernel-optimizations"] for some setup hints:

```
/usr/local/kvm/bin/qemu-system-x86_64 -hda xp-curr.img -m 512
-soundhw es1370 -no-acpi -snapshot -localtime -boot c -usb -usbdevice
tablet -net nic,vlan=0,macaddr=00:00:10:52:37:48 -net
tap,vlan=0,ifname=tap0,script=no
```

(kvm doesn't make a distinction between i386 and x86_64 so even in i386 you should use *qemu-system-x86_64*)

If you're on Debian Etch, substitute *kvm* for *qemu-system-x86_64* (thanks to fromport, soren and mael_). See also the entries under the label "Ubuntu" on the HOWTO page. *qemu-system-x86_64*

If you're on Fedora/RHEL/CentOS (and installed a kvm package and not built kvm yourself from source) then substitute *qemu-kvm* for *qemu-system-x86_64*

choose the right kvm & kernel version

Three Components

To make it work, you need to get the right version for three components:

- Linux Module
- User Space Application
- Guest Virtio Driver

[edit] Linux Module

KVM come with a linux module to support full virtualization. The linux module is just three files: kvm.ko, kvm_intel.ko, kvm_amd.ko. You can install them just like you install drivers for your video card. The good news is, you might do not need it install it. The 2.6.20 version has already included those linux modules. Depends on your distribution configuration, it might not be installed, or as a module or built-in. Here is a table listing the relation:

- 2.6.20 kvm-12
- 2.6.21 kvm-17
- 2.6.22 kvm-22

If you are not sure your linux distribution contain it or not, use this command:

```
modprobe -l | grep kvm
```

The linux module can be built from source code. This is also the recommanded way to get the right version of linux module. Compiling from the source code, and than make install should make the linux module inserted into your /lib/modules/linux-*uname -r*. Simply make it in use by:

```
modprobe kvm
modprobe kvm_intel
or
modprobe kvm
modprobe kvm_amd
```

User space application

Compile from source code, you can get it. Otherwise, refer to the previous section.

Guest virtio driver

There was no special requirement for guest operating system if you are not using para-visualized disk ornetwork adapter. If you are using them, make sure you get virtio_pci.ko, virtio_rng.ko, virtio_blk.ko, virtio_net.ko. They are in 2.6.25 or later kernel. There is also a option to backport them.

Refer to Virtio for more information

How To Migrate From Vmware To KVM

The vmware system consists of two disks in raw format: the old boot disk and the second one. It is Windows 2000 Server guest OS.

- Create empty new boot image (use dd).
- Boot stand alone OS from any other disk with old boot disk and new boot image connected (that is at least 3 disks).
- Sign up, make partition and format the new disk (do not make volume).
- xcopy /e /c /r /h /k /o /x /y old_boot_disk:new_boot_disk:
- Copy boot.ini from new boot disk to the second one (that is not boot disk).
- Edit boot.ini on the second disk to boot from rdisk(1)
- Swap disks: new <-> second.
- Boot setup from CD and go on Repair Procedure (after licence agreement) for the new boot disk.
- fdisk new boot image and toggle Boot flag.

System Message: WARNING/2 (data/kvm-howto.txt, line 141)

Block quote ends without a blank line; unexpected unindent.

- Swap disks back: second <-> new.
- Boot from the new disk.
- Remove vmware tools and devices.

- Reboot and be happy.

That's all.

- To swap disks and edit boot.ini (items 5,6,7,10) should be excluded if the boot disk is named C:, that is right in most cases. Those items are for strange case, when boot disk is D: and second one is C:.

Setting guest network

Guest (VM) networking in kvm is the same as in qemu, so it is possible to refer to other documentations about networking for qemu. This page will try to explain how to configure the most frequent types of network needed. [edit] User Networking

Use case:

- You want a simple way for your virtual machine to access to the host, to the internet or to resources available on your local network.
- You don't need to access your guest from the network or from another guest.
- You are ready to take a huge performance hit.
- Warning: User networking does not support a number of networking features like ICMP. Certain applications (like ping) may not function properly.

Prerequisites:

- You need kvm up and running
- If you don't want to run as root, the user you want to use needs to have rw access to /dev/kvm
- If you want to be able to access the internet or a local network, your host system must be able to access the internet or the local network

Solution:

- simply run your guest with "-net nic -net user", e-g:

  System Message: WARNING/2 (data/kvm-howto.txt, line 175)

  Literal block expected; none found.

```
qemu-system-x86_64 -hda /path/to/hda.img -net nic -net user
```

Notes:

- The IP address can be automatically assigned to the guest thanks to the DHCP service integrated in QEMU
- If you run multiple guests on the host, you don't need to specify a different MAC address for each guest
- You can still access one specific port on the guest using the "-redir" option.This means e.g. if you want to transport a file with scp from host to guest, start the guest with "-net nic -net user -redir tcp:5555::22". Now you redirecting the host port 5555 to the guest port 22. After starting up the guest, you can transport a file with e.g. "scp -P 5555 file.txt root@host-ip:/tmp" from host to guest. Remember that you must use the ip address of your host to connect to the guest, because you are redirecting a local port to the guest.

private virtual bridge

Use case:

- You want to set up a private network between 2 or more virtual machines. This network won't be seen from the other virtual machines nor from the real network.

Prerequisites:

- You need kvm up and running
- If you don't want to run as root, the user you want to use needs to have rw access to /dev/kvm
- You need the following commands installed on your system, and if you don't want to run as root, the user you want to use needs to be able to sudo the following command:

  System Message: WARNING/2 (data/kvm-howto.txt, line 195)

  Literal block expected; none found.

/sbin/ip /usr/sbin/brctl /usr/sbin/tunctl

Solution:

- You need to create a bridge, e-g:

```
sudo /usr/sbin/brctl addbr br0
```

- You need a qemu-ifup script containing the following:

```
#!/bin/sh
```

```
set -x

switch=br0

if [ -n "$1" ];then
        /usr/bin/sudo /usr/sbin/tunctl -u `whoami` -t $1
        /usr/bin/sudo /sbin/ip link set $1 up
        sleep 0.5s
        /usr/bin/sudo /usr/sbin/brctl addif $switch $1
        exit 0
else
        echo "Error: no interface specified"
        exit 1
fi
```

- Generate a MAC address, either manually or using:

```
#!/bin/sh
# generate a random mac address for the qemu nic
# shell script borrowed from user pheldens @ qemu forum
echo $(echo -n DE:AD:BE:EF ; for i in `seq 1 2` ;
do echo -n `echo ":$RANDOM$RANDOM" | cut -n -c -3` ;done)
```

- Run each guest with the following, replacing $macaddress with the value from the previous step:

```
qemu-system-x86_64 -hda /path/to/hda.img -net
nic,macaddr=$macaddress -net tap
```

Notes:

- If you don't want to run as root, the qemu-ifup must be executable by the user you want to use
- You can either create a system-wide qemu-ifup in /etc/qemu-ifup or use another one. In the latter case, run:

```
qemu-system-x86_64 -hda /path/to/hda.img -net
nic,macaddr=$macaddress -net tap,script=/path/to/qemu-ifup
```

- Each guest on the private virtual network must have a different MAC address

public bridge

WARNING: The here shown method, will not work with most(all?) wireless drivers, as these do not support bridging.

Use case:

- You want to assign an IP address to your virtual machines and make them accessible from your local network
- You also want performance out of your virtual machine.

Prerequisites:

- You need kvm up and running
- If you don't want to run as root, the user you want to use needs to have rw access to /dev/kvm
- You need the following commands installed on your system, and if you don't want to run as root, the user you want to use needs to be able to sudo the following command:

  System Message: WARNING/2 (data/kvm-howto.txt, line 260)

  Literal block expected; none found.

  /sbin/ip /usr/sbin/brctl /usr/sbin/tunctl

- Your host system must be able to access the internet or the local network

Solution 1: using distro sysconfig script

- **Edit /etc/sysconfig/network-scripts/ifcfg-eth0**

  o comment out BOOTPROTO o Add BRIDGE=switch

- **Create /etc/sysconfig/network-scripts/ifcfg-br0**

  o The content should be:

  System Message: WARNING/2 (data/kvm-howto.txt, line 274)

  Literal block expected; none found.

  DEVICE=switch BOOTPROTO=dhcp ONBOOT=yes TYPE=Bridge

- /etc/init.d/network restart
- The bridge br0 should get the ip address (either static/dhcp) while the physical eth0 is left without ip address.

Solution 2: manual

- You need to create a bridge, e-g:

```
sudo /usr/sbin/brctl addbr br0
```

- Add one of your physical interface to the bridge, e-g for eth0:

```
sudo /usr/sbin/brctl  addif br0 eth0
```

- You need a qemu-ifup script containing the following:

```
#!/bin/sh
set -x

switch=br0

if [ -n "$1" ];then
        /usr/bin/sudo /usr/sbin/tunctl -u `whoami` -t $1
        /usr/bin/sudo /sbin/ip link set $1 up
        sleep 0.5s
        /usr/bin/sudo /usr/sbin/brctl addif $switch $1
        exit 0
else
        echo "Error: no interface specified"
        exit 1
fi
```

- Generate a MAC address, either manually or using:

```
#!/bin/sh
# generate a random mac address for the qemu nic
# shell script borrowed from user pheldens @ qemu forum
echo $(echo -n DE:AD:BE:EF ; for i in `seq 1 2` ;
do echo -n `echo ":$RANDOM$RANDOM" | cut -n -c -3` ;done)
```

- Run each guest with the following, replacing $macaddress with the value from the previous step:

```
qemu-system-x86_64 -hda /path/to/hda.img -net
nic,macaddr=$macaddress -net tap
```

Notes:

- If you don't want to run as root, the qemu-ifup must be executable by the user you want to use

- You can either create a system-wide qemu-ifup in /etc/qemu-ifup or use another one. In the latter case, run:

```
qemu-system-x86_64 -hda /path/to/hda.img -net
nic,macaddr=$macaddress -net tap,script=/path/to/qemu-ifup
```

- Each guest on the network must have a different MAC address:

```
iptables
```

you can also connect your guest vm to a tap in your host. then setting iptables rules in your host to become a router + firewall for your vm. vde

another option is using vde (virtual distributed ethernet). performance

Data on benchmarking results should go in here. There's now a page dedicated to ideas for improving Networking Performance.

set up a network console

To set up a network console, add the following command to /etc/rc.d /rc.local:

```
/sbin/modprobe netconsole netconsole=@/eth0,12345@10.0.0.1
/00:E0:81:2B:0C:C1
```

Where the mac address is the destination nc listener and 12345 is the port number for listenning.

To run the log client, issue the command:

```
nc -dul 12345
```

on the client machine, 12345 is the port number from above.

It also helps to disable sync logging by changing /etc/syslog.conf from:

```
*.info;mail.none;authpriv.none;cron.none                /var/log
/messages
```

to:

```
*.info;mail.none;authpriv.none;cron.none                -/var/log
/messages
```

and to add:

```
echo 9 > /proc/sysrq-trigger
```

to /etc/rc.d/rc/local

How to assign devices with VT-d in KVM

Assigning device to guest

- Modifying kernel config:
  - make menuconfig
  - set "Bus options (PCI etc.)" -> "Support for DMA Remapping Devices" to "*"
  - set "Bus options (PCI etc.)" -> "Enable DMA Remapping Devices" to "*"
  - set "Bus options (PCI etc.)" -> "Support for Interrupt Remapping" to "*"
  - set "Bus options (PCI etc.)" -> "PCI Stub driver" to "*"
  - exit/save
- build kernel:
  - make
  - make modules_install
  - make install
- reboot
- unbind device from host kernel driver (example PCI device 01:00.0)
  - lspci -n
  - locate the entry for device 01:00.0 and note down the vendor & device ID 8086:10b9

```
...
01:00.0 0200: 8086:10b9 (rev 06)
...
```

  - echo "8086 10b9" > /sys/bus/pci/drivers/pci-stub/new_id
  - echo 0000:01:00.0 > /sys/bus/pci/devices/0000:01:00.0/driver/unbir
  - echo 0000:01:00.0 > /sys/bus/pci/drivers/pci-stub/bind

- load KVM modules:
  - modprobe kvm
  - modprobe kvm-intel
- assign device:
  - /usr/local/bin/qemu-system-x86_64 -m 512 -boot c -net none -hda /root/ia32e_rhel5u1.img -pcidevice host=01:00.0

VT-d device hotplug

KVM also supports hotplug devices with VT-d to guest. In guest command interface (you can press Ctrl+Alt+2 to enter it), you can use following command to hot add/remove devices to/from guest:

- hot add:

  pci_add pci_addr=auto host host=01:00.0

- hot remove (e.g bdf is 00:06.0 in guest):

  pci_del pci_addr=6

Notes

- VT-d spec specifies that all conventional PCI devices behind a PCIe-to-PCI/PCI-X bridge or conventional PCI bridge can only be collectively assigned to the same guest. PCIe devices do not have this restriction.
- If the device doesn't support MSI, and it shares IRQ with other devices, then it cannot be assigned due to host irq sharing for assigned devices is not supported. You will get warning message when you assign it. Notice this also apply to the devices which only support MSI-X.

Changing disks in the cdrom drive

Qemu provides a way to change iso in the cdrom via the monitor interface.

QEMU 0.10.5 monitor - type 'help' for more information (qemu)

The commands you'll want to use, info block, eject, and change. First we need to determine which block device is the cdrom you are interested in. Issue the info block command and look for cdrom devices.

```
(qemu) info block
ide0-hd0: type=hd removable=0 file=/dev/null ro=0 drv=host_device
encrypted=0
ide1-cd0: type=cdrom removable=1 locked=0 [not inserted]
floppy0: type=floppy removable=1 locked=0 [not inserted]
sd0: type=floppy removable=1 locked=0 [not inserted]
```

ide1-cd0 is the only cdrom device in this example and there isn't any media inserted. To change the cdrom, we issue the change command supplying the device name (ide1-cd0) and the path to the new iso file.

```
(qemu) change ide1-cd0 /tmp/dsl-4.4.10.iso
(qemu) info block
ide0-hd0: type=hd removable=0 file=/dev/null ro=0 drv=host_device
encrypted=0
ide1-cd0: type=cdrom removable=1 locked=0 file=/tmp/dsl-4.4.10.iso
ro=0  drv=raw encrypted=0
floppy0: type=floppy removable=1 locked=0 [not inserted]
sd0: type=floppy removable=1 locked=0 [not inserted]
(qemu)
```

In case you already have a cdrom in the drive you need to eject the current iso first before issuing the change command. The eject command takes the name of the cdrom block device.

```
(qemu) eject ide1-cd0
(qemu) info block
ide0-hd0: type=hd removable=0 file=/dev/null ro=0 drv=host_device
encrypted=0
ide1-cd0: type=cdrom removable=1 locked=0 [not inserted]
floppy0: type=floppy removable=1 locked=0 [not inserted]
sd0: type=floppy removable=1 locked=0 [not inserted]
(qemu) change ide1-cd0 /tmp/fedora11.iso
(qemu) info block
ide0-hd0: type=hd removable=0 file=/dev/null ro=0 drv=host_device
encrypted=0
ide1-cd0: type=cdrom removable=1 locked=0 file=/tmp/fedora11.iso ro=0
drv=raw  encrypted=0
floppy0: type=floppy removable=1 locked=0 [not inserted]
sd0: type=floppy removable=1 locked=0 [not inserted]
```

use virtio

Using virtio_net For The Guest NIC

The following is done using Debian Lenny, with the 2.6.25 kernel installed from Sid.

- install the guest OS as per normal, using rtl8139 or e1000

for the guest NIC
- boot into the guest as per normal
- edit /etc/apt/sources.list to add a sid repo
- install the 2.6.25 kernel
- boot into the guest using the 2.6.25 kernel
- edit /etc/initramfs-tools/modules and add virtio, virtio_pci, virtio_ring, virtio_net, virtio_blk
- update the initramfs using update-initramfs -u
- shutdown the guest
- change the -net nic option to include model=virtio
- boot the guest

Throughput Tests Using iperf

To see what the throughput differences would be like, I ran a bunch of iperf tests from a Debian guest. Host Server Config

- Tyan h2000M motherboard
- 2x dual-core Opteron 2220 CPUs @ 2.8 GHz
- 8 GB DDR2-667 ECC SDRAM (2x 2 GB sticks per CPU)
- 3Ware 9650SE-16ML SATA-II RAID Controller
- 12x 500 GB SATA-II harddrives in a single RAID-6 array
- Intel PRO/1000-MT PCIe quad-port gigabit NIC (configured as 4-port bond0, used as the kvm bridge)
- Debian Lenny installed, updated June 04, 2008, with kvm-69 and kernel 2.6.25 (kernel from Sid)

Guest VM Config

- 1 CPU
- 2 GB RAM
- 100 GB virtual harddrive (via LVM)
- Debian Lenny with kernel 2.6.25 from Sid

System Message: WARNING/2 (data/kvm-howto.txt, line 524)

Bullet list ends without a blank line; unexpected unindent.

```
/usr/bin/kvm -name mail -daemonize -localtime -usb -usbdevice
tablet -smp 1 -m 1048 -vnc :04 -pidfile /var/run/kvm/mail.pid
-net nic,macaddr=00:16:3e:00:00:04,model=virtio -net
tap,ifname=tap04 -boot c -drive
index=0,media=disk,if=virtio,boot=on,file=/dev/mapper
/vol0-mail
```

iperf Server Config

```
iperf -s -w 65536 -p 12345 -I 5
```

iperf Client Configs

```
  * [1] iperf -c <server> -w 65536 -p 12345 -t 60
  * [2] iperf -c <server> -w 65536 -p 12345 -t 60 -d
  * [3] iperf -c <server> -w 65536 -p 12345 -t 60 -P 4
```

Results

These are averaged over 3 runs.

To the host (virtio)

- [1] 92 Mbps

To a server connected to a gigabit port on the same switch (virtio)

- [1] 834 Mbps
- [2] 519 Mbps out, 531 Mbps in
- [3] 906 Mbps combined

To a server connected to a gigabit port on the same switch (e1000)

- [1] 296 Mbps
- [2] 259 Mbps out, 62 Mbps in
- [3] 302 Mbps combined

Docutils System Messages

System Message: ERROR/3 (data/kvm-howto.txt, line 77); *backlink*

Unknown target name: "mael".

tags

2011 2012 apache backup bash chroot command consola debian file files fluxbox for from git how howto install iptables libre line linux mutt mysql para php python que red redes remote repository restructuredtext server software ssh text tutorial ubuntu unix using vim with

---

cat OSiUX.com/*.txt | frontweb