CREATING, EDITING AND MANAGING VMS

Creating VMs is simple enough with the virt-manager tool. However, importing an existing VM (from both VMWare as well as VirtualBox) and converting it to run on KVM was nightmarish. For some reason, when KVM is told to use an existing disk, it silently assumes that the disk it is being pointed at is a raw disk format, even if the disk file has a different extension. In order to get my KVM machine to boot, I had to export the VM definition as an XML file, change the disk type manually, and then reimport it.

KVM offers, by far, the most choice of all the three products reviewed here on virtual hardware selection. Several different types of hardware can be emulated, based on user selection. KVM also allows both NAT and bridged networking. However, the bridged network mode requires setting up a bridge on the host, manually, which can be tedious.

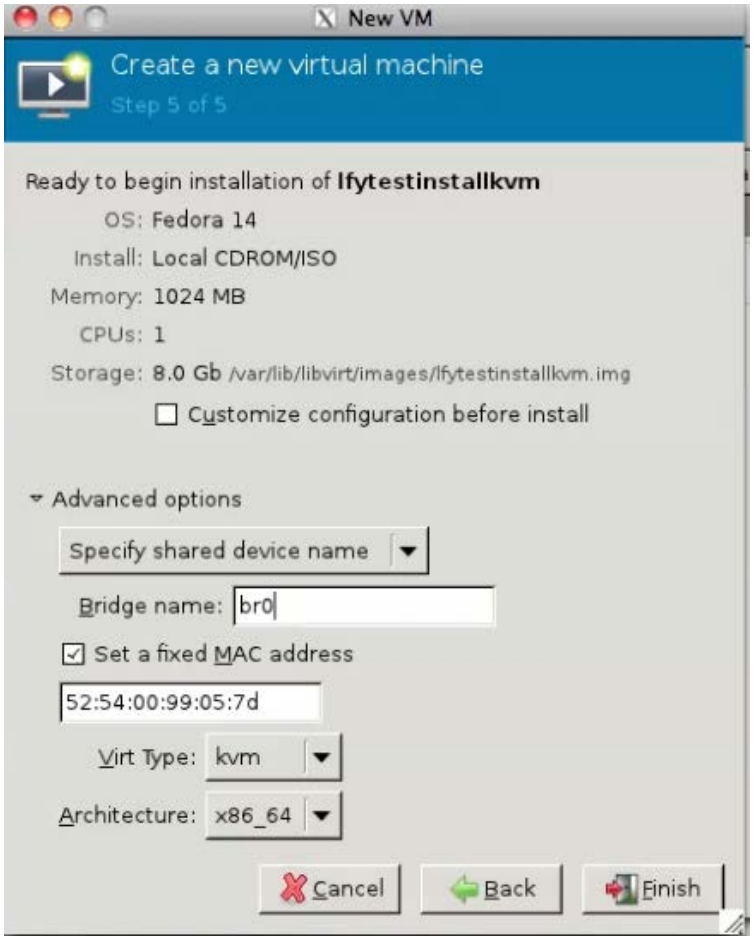

Figure 9: Choosing the bridged device in KVM

Device support

Since KVM is part of the Linux kernel, it supports most devices that Linux does. Moreover, since KVM has a large open source developer community behind it, support for new devices is usually quick to appear. However, KVM's device support is far more appealing from the perspective of server virtualisation than the desktop, at least for the average user.

Other features

KVM's feature set is mainly systems-oriented, without too many cute user-land baubles. Live snapshots of guest VMs are supported, from the command line. There is no concept of shared folders or clipboard. USB support is included out of the box. KVM also supports live migration of guests between non-homogeneous hosts.

Ease of use

KVM is the least user-friendly of the three products reviewed here. It has all the distinctive signs of a product by, and for, the elite geek or the expert systems architect.

Installation

Simply put, none required. KVM comes backed in with the Linux kernel. The only installation I required was that of virt-manager, a graphical tool to manage VMs, which was a simple installation from the distro's package repository. Unfortunately, I also had to install several X libraries on an otherwise headless system, and then access it through a remote X session. The alternative was to spend several hours mastering the command-line language of KVM and virsh, the command-line tool for managing VMs.

Administration

Even with virt-manager, which simplifies things drastically, KVM's administration requires familiarity with virtualisation. If you know exactly what you are looking to accomplish, KVM will probably have the means to accomplish it. However, if you are unsure of your need, it is very easy to get lost within KVM's myriad options. Also, the administration console is not very good at giving user feedback.

For example, changes to a VM that require a reboot can still be made while the machine is running, and will be applied the next time the machine reboots. However, after the change is queued up, there is no visible indication that a change is pending, until the reboot happens, and the change takes place. For an inexperienced user, this sort of behaviour can be quite confusing.

Look and feel

KVM is definitely not in the same league as VMWare Player or VirtualBox when it comes to look and feel. The VM display is over VNC, the quality of which is dependent on the network link between the client and the KVM server. On my installation, the VNC display refused to recognise several keys altogether, resulting in my having to use a different VNC client. Mouse-pointer and clipboard integration are also not as good as the other two products.

Performance

Performance is where KVM outshines both VMWare Player and VirtualBox (see Table 1 for a comparison). The secret behind KVM's success is the use of the libvirt libraries and the VirtIO network and hard disk interfaces. VirtIO is very similar to para-virtualisation, the only difference being that while para-virtualisation requires the entire guest OS to be modified (for example, Xen's requirement that guests run the Xen kernel), VirtIO works by installing components inside the guest. These components are included as the libvirt libraries in most of the new Linux distros. However, for Windows guests, this means that a VirtIO driver needs to be installed.

For disk I/O, KVM allows the virtual hard disk to be mounted in write-back or write-through mode. In write-back mode, the process of writing data to disk receives an acknowledgement of the write having occurred immediately after the I/O subsystem receives it, even if it has not been committed to disk. This means that processes run faster. However, data in transit (written but not committed) is stored in volatile memory, and so a crash or power outage will result in data loss.

For mission-critical systems that require high data integrity, the write-through mode is preferable, since it guarantees a commit to disk. However, the existence of the write-back mode means that KVM machines can be tweaked to perform disk I/O at super-fast speeds.

Licensing and support

KVM's licensing is hard to beat for attractiveness, since it's completely open source, with all the components released under the GPL or the LGPL. The KVM user community is somewhat biased towards developers and systems designers rather than end users, so finding solutions to problems requires some background and understanding of the platform. Essentially, you can find help, but you really do have to read the manual.

Score

On a scale of 5:

- Features: 2
- Ease of use: 1
- Performance: 5
- Licensing and support: 4

Table 1: I/O performance comparison

	VMWare Player	VirtualBox	KVM
Disk I/O (time to copy a 100 MB file using dd)	0.97 seconds, 105 MBps	0.63 seconds, 161 MBps	0.545 seconds, 188 MBps (write-back mode)
Network I/O speed (using iperf)	1.31 GBps	2.01 GBps / 3.3 GBps with virtio	4.61 GBps

Analysis

To sum up the comparison, look at Table 2.

Table 2: Performance summary

	Features	Ease of Use	Performance	Licensing and support	Total
VMWare Player	3	4	3	1	11
VirtualBox	4	4	4	4	16
KVM	2	1	5	4	12

From a look at the score board, VirtualBox clearly outshines the other two products by being better than average in all the categories, even if it doesn't top any of them, with KVM next, and VMWare Player last.

Let us now see what these scores mean, from the viewpoint of different types of users.

The enthusiast

From the enthusiast's standpoint, KVM would seem like the best choice, since it has the most configuration options. Enthusiasts will find plenty of new combinations of settings to experiment with. KVM's lack of end-user features and complexity of use also give it the flavour of being a tool for the elite, which the enthusiast is likely to find appealing.

Next in line is VirtualBox, which offers fewer options, but still enough to keep a geek interested. Moreover, getting bragging rights on VirtualBox doesn't require you to have a PhD. Simply reading the manual and following the forums will suffice.

VMWare Player is last in line for this category of users, since it offers very limited customisability.

The architect

For architects, the requirement determines the choice of component. For server virtualisation with an emphasis on performance and scale, KVM is the clear choice. For end users, VMWare Player is the best choice, since it can run a VM authored on Workstation in a manner that makes it extremely easy to use, particularly for a user who's not tech-savvy. However, for prototyping and getting off the ground quickly, VirtualBox' superior feature set makes it the tool of choice.

The executive

From the executive's standpoint, VirtualBox is the product with the likelihood of meeting the most requirements at the least cost. If budget was not a constraint, VMWare Player's paid version, VMWare Workstation, could give VirtualBox a run for its money on features. VirtualBox provides many more features in the free version than VMWare. The exception to this is if the VM is being authored elsewhere, and VMWare Player is being used only for access. In that scenario, VMWare Player is much easier to use than either of the other two.

KVM is not really a solution for the executive at all.

The follower

From the follower's viewpoint, the ideal and often only supportable option is to use VMWare Player to run existing VMs (created by Workstation). Next in line is VirtualBox. As with the executive, KVM is not an option for the follower.

Source : http://www.opensourceforu.com/2011/09/vmware-player-virtualbox-kvm-virtualization-comparative-review/