Exploring Ubuntu's virtualization tools

GET VIRTUAL

Marina Strizhak, 123rf.com

Why tie yourself to the hardware? We'll show you how to operate virtual environments from your Ubuntu desktop.

BY MARCEL GAGNÉ

After just loading your copy of the shiny new Linux release Ubuntu 9.04, things are going well. Now that you've read the other articles in this issue, you're starting to think you've found a pretty good home. But even when you love your home, once in a while you want a holiday – which means loading up another Linux distribution to play with.

Of course, it might not be all play. Sometimes you need to test another operating system, or you might need to run Windows as well as Ubuntu. Wouldn't it be nice if you could run all those things on your current computer, without having to reboot or reinstall? You can – with virtualization.

Virtualization software creates a virtual computer on your computer, on which you can install any other operating system, whether it be Ubuntu, RedHat, Mandriva, or yes, even Windows. From the perspective of the guest operating system, the virtual machine is a real machine with real hard drives, a real network card, real video hardware, and so on. When you boot that virtual machine, it will appear as though you are booting a real machine.

Virtualization Tools

Your Ubuntu system doesn't have any of this software installed by default, but getting it is easy. Furthermore, several virtualization packages are available. These include the command-line programs QEMU [1] and KVM [2], as well as one my favorites, the powerful graphical VirtualBox [3]. To install these programs, select them via Synaptic or the *apt-get* command:

```
apt-get install qemu kvm ➥
virtualbox-ose
```

Figure 1: QEMU virtual hardware boot.

Of the three, I prefer VirtualBox, but let me show you how easy it is to run a virtual machine using QEMU or KVM. KVM, the Kernel Virtual Machine, is a replacement for QEMU. It works by taking advantage of processor-based hardware virtualization technology. On the Intel platform, this is known as VT; on AMD processors, it is called Pacifica. What it means to you is that you can take advantage of KVM to run your virtual machines with substantially better performance than you can with straight software virtualization. The catch, as you might expect, is that not every machine out there supports hardware virtualization. To find out whether your system is a candidate, execute this little snippet of code from a terminal window or shell:

```
grep -E ➥
 '^flags.*(vmx|svm)' ➥
/proc/cpuinfo
```

If it returns the string *vmx* or *svm*, your processor is ready. If not, you have to stick with the software-based virtualization of QEMU. The KVM commands are pretty much interchangeable with those of QEMU, as are the install and operating system images. In fact, KVM is based on QEMU.

Next, I'll take a look at how QEMU works. For this first demonstration, I'll install a tiny distribution called Damn Small Linux (DSL) from an ISO image downloaded from the project website [4]. Because DSL is a cute little distro with minimal space requirements, I'm going to create a relatively small disk image (a virtual hard disk) for it to live on with the *qemu-img* command:

```
qemu-img create dsl.img 256M
```

The system returns with the following reply:

```
Formatting 'dsl.img', ➥
fmt=raw, size=262144 kB
```

The preceding command creates a raw-formatted disk image by default. A few different image formats are available, most notably qcow2, which is a portable image format that is useful if you want to install a non-Linux OS – like Windows XP, for example.

The next step is to install Linux into this disk image, which I do with the *qemu* command:

```
qemu -cdrom dsl.iso ➥
-hda dsl.img -m 256 -boot d
```

Several interesting things are happening here. For starters, the *-cdrom* parameter is the path to the CD-ROM image from which you are installing your distribution. If you were using a physical CD-ROM, that path would likely be something like */dev/cdrom*. The next parameter, *-hda*, defines the path to the disk image I just created; then, it's followed by the *-m* switch, which, in this case, allocates 256MB of RAM to the running session. Finally, *-boot* identifies the boot drive (the CD image), which is the so-called D drive. As soon as I press Enter, the virtual hardware boots (Figure 1).

The disk you created and the CD-ROM are represented as physical devices at boot time, as is the default video card and the BIOS. In fact, you can even interrupt the boot process and bring up a device boot menu. If you press Enter at this point, the system will go through its normal boot process, which, in the case of a Live CD, usually stops to let you decide what to do as far as booting or installing. DSL is built on Knoppix, so

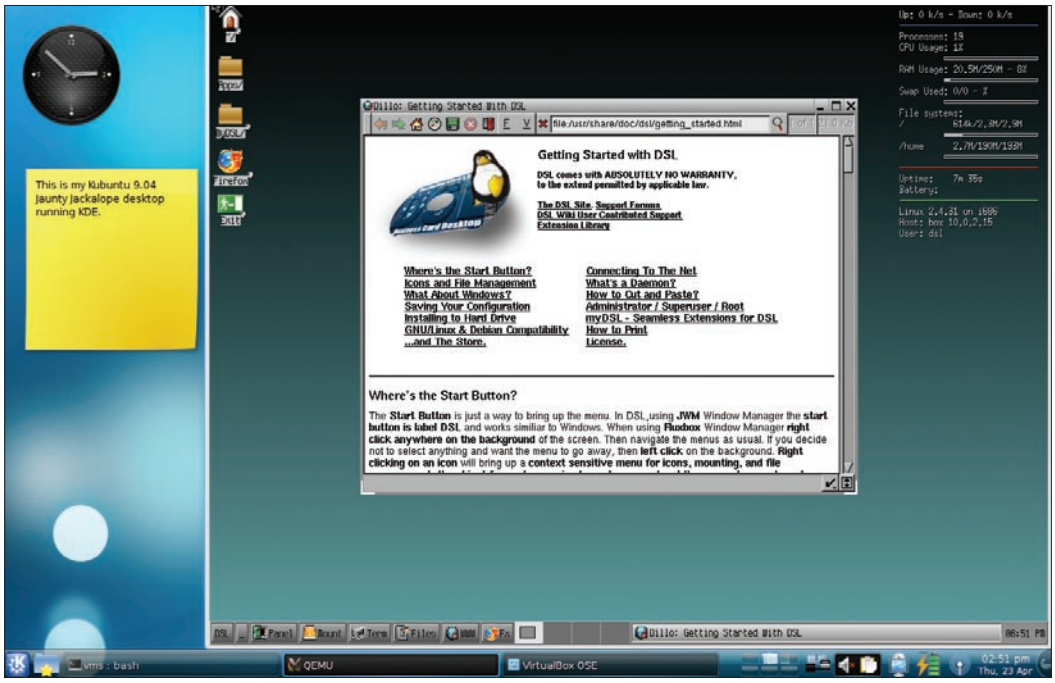

Figure 2: A Kubuntu Jaunty Jackalope desktop with an instance of Damn Small Linux booted and running on top.

users of the venerable Live CD will find some of this familiar. For everyone else, just press Enter.

Everything that happens from this step forward should happen as it would if you were running any other Linux distribution from a Live CD. The guest operating system will boot, or install, normally. Depending on the speed of your machine and the amount of memory you have at your disposal, the process could be peppy or quite slow. Remember that this virtual machine uses whatever portion of your host system's resources you alloted. That means less for your host system, both in terms of memory and processor resources. Also, if your processor allows KVM instead of QEMU, the performance will be substantially higher.

A few seconds (or minutes) later, your virtual machine is up and running (Figure 2).

KQEMU

Before, I suggested that running QEMU instead of KVM means you aren't privy to processor-level acceleration, which means you are emulating everything in memory without the benefit of kernel acceleration. When you are running a little tiny distribution like DSL, that's fine, but if you want to try a different Ubuntu, openSUSE, Fedora, or even Windows, you will feel the pain pretty quickly. In this case, the KQEMU hardware acceleration module comes into play. The catch is that it's not installed by default, and you can only install the source for it. Yes, that means you have to build it. Luckily, that's not complicated:

```
sudo apt-get install ➥
module-assistant
sudo module-asssitant ➥
prepare
sudo module-assistant -f ➥
get kqemu-source
```

The result is a kernel module that dramatically speeds up your QEMU sessions, giving you performance on par with KVM. Now, even with QEMU, you can run and install more demanding operating systems, like Ubuntu. For the pure fun of it, I loaded up Ubuntu Jaunty Jackalope (with the Gnome desktop) and ran a full implementation of it on my Kubuntu desktop (Figure 3).

Before I move on, I'll share a few other parameter with you. For starters, you might add *-no-acpi* to disable this in the virtual machine. Also, you can use the *-localtime* parameter to tell QEMU to run your session with the local machine time as opposed to the default UTC. Remember the kernel acceleration? Try *-kernel-kqemu*. In addition are network parameters, USB parameters, file-sharing parameters (SMB), devices parameters, and more. Because you are only seeing the basics, make sure you check out the man page for the *qemu* command. Also remember that my settings for this example imply a distro with a very small footprint.

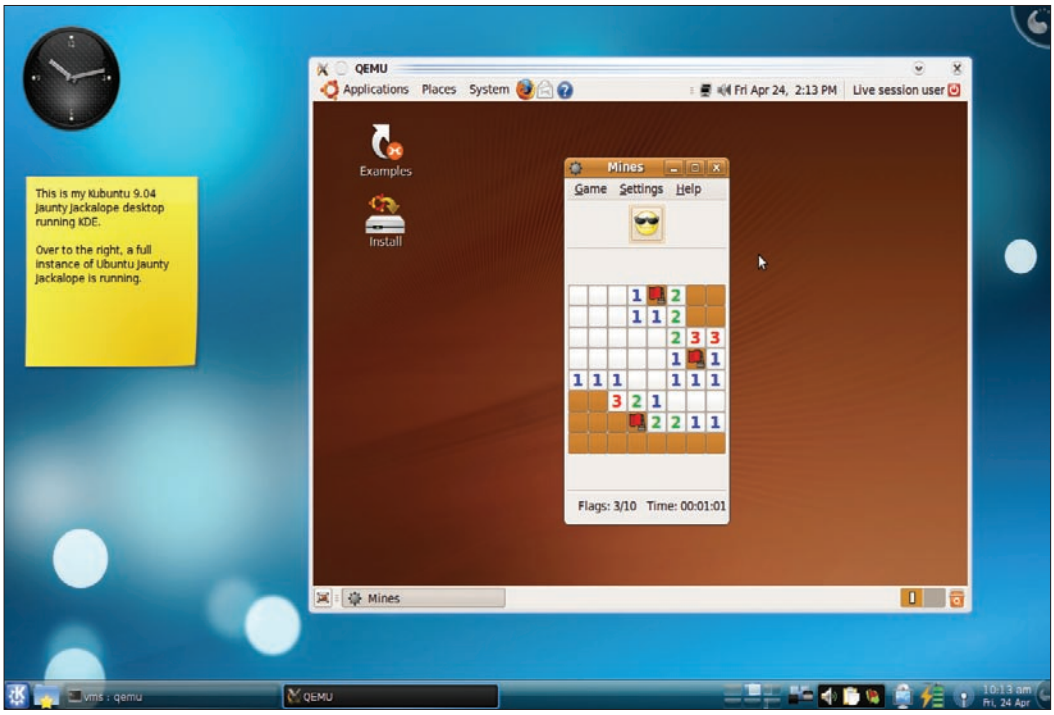

Figure 3: The best of both worlds: Ubuntu's Jaunty Jackalope 9.04 running in QEMU under Kubuntu.

Figure 4: Your first step is to tell VirtualBox what guest operating system will run on this VM.

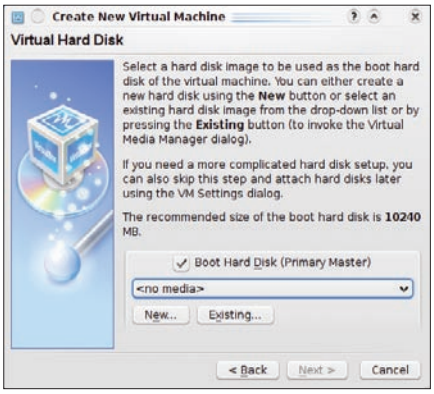

Figure 5: Unless you defined a previous virtual disk, you'll need to create one. Because it is possible to reuse disk images, previous images will be listed here.

VirtualBox

Another great piece of virtualization software is VirtualBox, an open source package freely distributed under the GPL and distributed by Sun Microsystems (who, as I write this, has been purchased by Oracle Corporation). VirtualBox is a program I use every day and one I recommend highly. With VirtualBox, I can load an operating system that you might still want access to: Windows XP. Of course, you do need a licensed copy of

Microsoft Windows to proceed.

When you start VirtualBox for the first time – assuming you already have Virtualbox installed via either Synaptic or apt-get – no machines will be running in it. Think of it as a blank slate or, better yet, a new computer with a blank hard drive waiting for your favorite distribution, or, in this case, Windows XP.

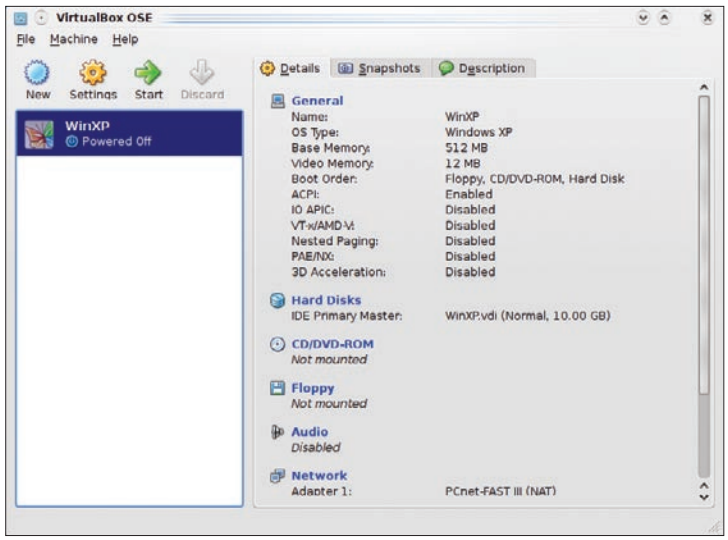

Figure 6: Your new machine appears, ready to be started.

Before you can install Windows, you need to create your virtual hardware. To do this, click *New* on the VirtualBox toolbar. A nice welcoming message explains that the wizard will guide you through the various steps. When you click *Next*, you will be asked to decide what kind of virtual machine you plan to create (Figure 4).

To start, give it a name. For example, I've chosen to call mine the very unoriginal *WinXP*, but you can call it whatever you like. Next, select the operating system (Microsoft Windows) from the drop-down list, then identify the version (Windows XP). The list of operating systems covers a lot of ground, including Windows, Linux, Solaris, BSD, and even IBM's old OS/2, and each of these operating systems have many versions. The Windows support goes right up to the new, not-yet-released Windows 7. When you click *Next*, you will be asked to select the amount of RAM you want to dedicate to the VM.

Resources

Deciding how much memory to allocate is a bit of a balancing act. On one hand, most modern operating systems have minimum requirements. On the other hand, you have to take into consideration how much memory your Ubuntu host system has. Either enter the amount into the text box or use the slider.

Next, your new machine needs a hard drive on which to load XP, so you must create a virtual hard disk (Figure 5). Because this is your first time out, the primary master (the main disk) will be listed as *< no media >*. Clicking *New* creates a disk image.

Clicking *Next* in the new welcome message for the virtual disk wizard leads you to the dialog from which you can choose one of two disk image types: a fixed-size storage medium or a dynamically expanding disk image. The footprint of a dynamically expanding image is minimal to start and expands as needed. In comparison, the fixed image takes up whatever space you give it right from the beginning and runs up against a wall when that space is used. Unless you have good reason to do otherwise, select *Dynamically expanding storage* and click *Next*.

Having made your decision on storage options, you can now define the size and location. The location, by default, is your machine name (my WinXP in Figure 4). In effect, this is a folder on your disk in which the machine resides. Next, you allocate the space either by typing it into the text window or with the slider. A size for this disk will be suggested according to the operating system you are loading. On the basis of your expected needs, accept the default or choose a size. After you click *Next*, you can review your selections. If you are happy with the choices you've made, click *Finish*.

Now you are back to the virtual hard disk selection screen you saw in Figure 5, but this time, your new disk image is selected for you. All you have to do is click *Next* to continue. This opportunity is the last you'll have to review everything you've done to date. VirtualBox will remind you of your chosen machine name, the memory allocated, the OS type, and the type and size of disk you've chosen. When you click *Finish*, your virtual machine is ready. In the main VirtualBox window (Figure 6), your new WinXP machine appears in the list of machines (on the left), ready to be loaded.

Remember, this is a clean computer, with nothing installed. To the right of the machine list are the parameters associated with the selected machine. Each of the blue labels can be clicked to config-

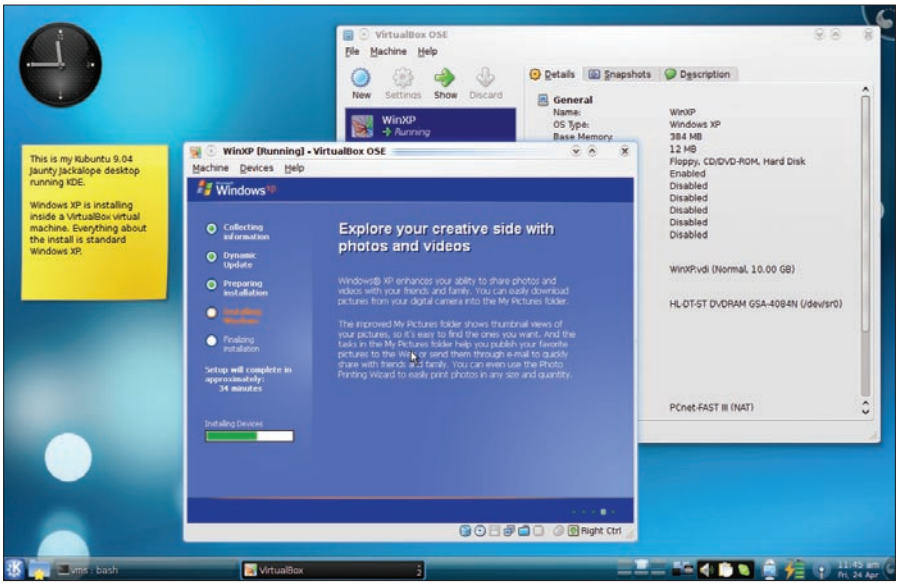

Figure 7: The new virtual machine is live, and Windows XP is installing normally. With VirtualBox running on my Ubuntu KDE desktop, the XP installation is in the foreground.

ure the resource listed. Notice that you can even go back and change the virtual machine specifications, such as memory. Other labels let you define what physical resources (on your host machine) the virtual machine has access to. For instance, to give your virtual machine access to the physical DVD-ROM drive in which the Windows XP install media is located, click on the *CD/DVD-ROM* label.

In the resulting dialog box, check the *Mount CD/DVD Drive* box. If you have more than one drive, select the one you want from the drop-down list. The radio button labeled *ISO Image File* is particularly interesting if you are loading another Linux distribution because you can install it directly from the hard drive, skipping the need for a physical disc (as is the case with Windows because free Live DVDs don't exist). When you click *OK*, you'll find yourself back at the main VirtualBox window with the *CD/DVD-ROM* label now indicating the host computer's physical drive. Now you are ready to install Windows XP.

Installing the Guest

First, make sure the virtual machine is selected, then click the *Start* button. VirtualBox will let you know that the autocapture keyboard feature has been activated. This includes your mouse as well. What this means is that the virtual machine has full access to your mouse and keyboard. To "uncapture" your mouse and keyboard, press the right Ctrl key on your keyboard. To acknowledge the message, click *OK*, and your new computer will boot from the Windows install disc. What follows is a standard Windows XP installation (Figure 7): You accept the license agreement, your virtual disk is formatted, and Windows does its thing.

Somewhere in the process, you'll need to enter your license code, answer some questions, and so on. Eventually, the installation will complete and your new Windows machine will be up and running. After you attend to standard Windows stuff, such as adjusting screen size and making other modifications to your running machine, you'll have a full Windows XP implementation running on top of your Ubuntu (or Kubuntu) desktop, with access to everything you normally use under Windows (Figure 8). Instead of emulating Windows, you are running a real, albeit virtual, PC loaded with Windows XP.

Although this might sound like a happy ending, in which the two seemingly irreconcilable operating systems are working together on the same hardware, VirtualBox has Guest Additions that can improve the relationship. Once you've had to hit the control key a few times to recapture your keyboard and mouse or you find that the default screen sizes provided by Windows just don't work well on your widescreen notebook, you'll be happy to learn that even these issues can be resolved. To install the Guest Additions, release your keyboard and mouse (right Ctrl key), then click the Devices menu on the virtual machine and select *Install Guest Additions* from the menu (Figure 9).

Figure 8: Windows XP running on my Kubuntu desktop – fully integrated and ready for action.

Figure 9: For nearly seamless integration into your Ubuntu desktop, install VirtualBox's Guest Additions into your running XP machine.

After you've followed the instructions for installing the guest additions and restarted your machine, your experience will be vastly different. Just by clicking on a window inside Windows, you can change the focus from your Ubuntu applications to your Windows desktop. Your mouse will be able to sail across your desktop from Windows to Ubuntu and back again. Furthermore, you will be able to resize Windows to fit whatever geometry makes sense to you and your Ubuntu desktop. Finally, you will be happy to know that these guest additions are not specifically a Windows thing and are available for other operating systems as well.

Conclusion

Happily ever after is where this story ends. By running Ubuntu, you have a superior operating system that is faster, more reliable, and more secure than Windows. But you can have your Ubuntu and your Windows too. Best of all, you aren't limited to just Windows. For example, you can try other Linux distributions, and even BSD or Solaris, from the comfort of your existing Ubuntu installation without reinstalling.

Yes, you can have it all – with a little virtualization. ■

INFO

[1] QEMU: *http://www.nongnu.org/qemu/*

[2] KVM: *http://www.linux-kvm.org/page/Main_Page*

[3] VirtualBox: *http://www.virtualbox.org/*

[4] DSL: *http://damnsmalllinux.org/*