# Virtual Machines

**Dick Maybach, Brookdale Computer User Group, NJ**

A virtual machine (VM) is a program on your PC (called a host in virtual-machine speak) that creates a box that appears to be a PC to an operating system (called a guest in virtual-machine speak) that resides in it. In Figure 1, the outer gray box is your PC hardware and software. The blue box within it is the program (called a hypervisor) that creates the software boxes that hold the guests, each shown as a light gray rectangle. Each guest has its own virtual hardware (in yellow) with which it communicates just as though it were a PC. Each guest also needs its own operating system (in orange) and applications (in green).

Figure 1. Virtual Machine Organization.

From the host's view, each guest is just an application, and each can operate independently of other applications on the PC, which means you can have two or more operating systems live at the same time. Moreover, you can copy and paste data and transfer files between them at any time.

It's not shown in Figure 1, but the hypervisor has an interface to configure and control the VMs; Figure 2 shows an example, in this case, VirtualBox. Note that there are five different VMs here, Windows 7, FreeDOS, Tails, Ubuntu 18.04, and Ubuntu 18.10, and all are powered off. The right portion of the screen summarizes the characteristics of the Windows 7 guest.

Figure 2. VirtualBox Administration Panel.

Figure 3 shows Windows 7, running as a guest under VirtualBox.

Figure 3. Windows 7 under VirtualBox.

As with most other applications, control follows the cursor. Place the cursor within the VM window and mouse clicks and keystrokes are sent to the VM. Move the cursor outside the VM window, and they affect something else.

Why bother with this? As you probably suspect, adding a software layer between the guest OS and the PC slows down the guest and complicates the host. Here are some possible uses.

- Run a different OS – my host runs on Linux, but there are a few applications available only on Windows, such as TurboTax and the software needed to update my GPS. If your host is Windows, putting Linux on a VM is far superior to running it from a live USB.
- Trial OS upgrades – I prefer to try a new version before I commit to it on the host, as often some

applications aren't compatible. In the case of Windows, free trial versions are often available before a new version is introduced, and a VM lets you play with these without risk.
- Test alternate configurations of your host OS – making such experiments on a VM can save you from serious, "It seemed like a good idea at the time," calamities. This is more difficult with a commercial OS but look for ways around this. I've made a duplicate installation of Windows on a VM, and while it complained, it did run long enough for me to complete my tests.
- Trial applications – testing applications on a VM means you don't have to uninstall them when they don't work out. This is made easier with the VM snapshot feature. Before you install, create a snapshot (equivalent to cloning the hard disk). You can then revert to the snapshot if you decide to discard the application.
- Test live USBs or DVDs – much software is available on live media. You download an ISO file, burn it onto a medium, and boot your PC from it. With a VM, you just designate the file as being installed on the guest's virtual DVD drive, which now boots from the virtual DVD instead of its virtual hard disk. As a result, you don't have to burn the file to a medium.

Before I used VMs, I relied on dual booting for the similar tasks. Here, the alternate OS has direct access to the PC, which means it's substantially faster than when running as a VM guest. However, this requires repartitioning the host's hard drive, where an error can be catastrophic, and moving data between the two hosts is awkward.

There are many VM systems available, with the four most popular being VMWare (for Windows, OS X, and Linux), VirtualBox (for Windows, OS X, and Linux)  Parallels Desktop (for OS X) and QEMU/KVM (for Linux). I've used VirtualBox for several years and have recently begun experimenting with QEMU/KVM, so everything I say about the other two is hearsay. From what I've read, it appears that VirtualBox is the easiest to configure and use, plus it's free for home users. VMWare, the oldest of the four, is possibly faster than VirtualBox, but configuring it requires some experience, and it's a commercial product. A free version is available, which is missing only a few unimportant features. QEMU/KVM is not for the faint-hearted, as it was designed by Red Hat for use in professionally staffed server farms. Also, although it appears to be quite fast it runs only on Linux. Making configuration changes such as, increasing the screen resolution and enabling host/guest file sharing involves working at the command line and being familiar with the Linux file organization and permissions. However, once set up, it's as easy to use as VirtualBox. GNOME Boxes (which I haven't used) allows Linux users to use KVM with a simple set-up process and user interface, although with more limited control of the virtual environment.

Figure 4 shows the QEMU configuration panel, which has essentially the same information as that of VirtualBox. After a VM has been installed, the day-by-day configuration is done from here.

Figure 4. QEMU/KVM Configuration Panel.

A VM operating under QEMU/KVM appears in Figure 5, also essentially the same as it would appear under VirtualBox.

Figure 5. QEMU/KVM in Operation.

Running VMs requires a host with adequate resources. It needs four Gbytes RAM (absolute minimum, eight is preferable) and 12 free Gbytes disk space for each guest (again more is better). VM features on CPU are desirable (VT-x on Intel and AMD-V on AMD processors). Check your VM documentation, as these features may be disabled in your BIOS.

Guest speed will be lower than those of a host, but for most applications this isn't important. Of course, you should run any resource-intensive tasks on your host. Graphics, in particular, will be slower and probably have fewer features than your host hardware. Resources used by a guest are no longer available to the host. For example, if your host has eight Gbytes of RAM and you allocate four to a guest, the host now can use only four.

A VM is completely defined by its file, which is essentially an image of its virtual hard disk. If you back up this file, you've backed up the VM. This isn't an unmixed blessing, as any time you boot the VM there will be changes in the file, which typically occupies at least several Gbytes of disk space. As a result, your backups will take longer and occupy more space.

Guests are reasonably well isolated from the host, except for any shared directories, making you fairly secure against the common risks, such as operator error and software bugs. However, some malware attacks can get through to the host, which means a VM is not a good vehicle for investigating malware.

Although VMs are valuable, they do introduce complexity and add risk. I used VirtualBox for several years but noticed beginning with Ubuntu version 18.04 that after a few minutes Ubuntu guests would drastically slow, to the point of being unusable. This can be remedied by turning off VirtualBox's 3D display acceleration, which is enabled by default. The developers have acknowledged the problem but say they don't have the resources to correct it. After the last VirtualBox update, it refused to run my Ubuntu 16.04 guest at all, which is why I ventured into QEMU/KVM land, and fortunately, the trip was successful. Using VMWare might also have solved the problem, but since my host is Linux, QEMU/KVM was a better approach.

Despite their disadvantages, VMs are a valuable resource for me, one I use almost every day. They allow me to run software experiments more easily and with less risk than do their alternatives. Without them, many of my articles would have been just cut-and-paste cribs from other folk's work.

*Reprinted with permission from the August 2019 edition of BCUG Bytes.*