Published on *Linux.com* | *The source for Linux information* (https://www.linux.com)

Containers vs. Hypervisors: Choosing the Best Virtualization Technology [1]

Choosing a virtualization solution isn't always easy. The good news is you have many choices to pick from. The bad news is, well, pretty much the same thing. You'll find tons of options for Linux, most of which break down to hypervisor or container-based virtualization. Not sure which is which? We'll break it down.

Unfortunately, the various vendors that offer virtualization (say that five times fast) don't always agree on terminology. You'll also find several different types of hypervisor virtualization, depending on the solution and hardware that are being used. For simplicity's sake, I'm going to refer a bit broadly to the different types of virtualization. For most types of virtualization, I'll refer to hypervisor virtualization. For virtualization at the operating system level, I'll refer to container-based virtualization. Note also that I'm mainly looking at server virtualization on commodity x86/x86-64 hardware here. The options are even more diverse if you're working with platforms like IBM's System z mainframes.

Which solution is right for you? That depends a lot on your workload, hardware and environment. In some cases, you could go either way, and in other cases there's a clear winner between hypervisor and container-based virtualization. In some cases, it may even make sense to deploy both, though that can become more complex in terms of finding management tools that work well with all the options.

The Skinny on Hypervisor Virtualization

When we're talking about hypervisor virtualization on Linux, we're talking about running a full operating system on top of a host operating system, usually a mixture of operating systems. For example, running Red Hat Enterprise Linux and Microsoft Server on top of a Red Hat Enterprise Linux (RHEL) host using VMware or Xen. Hypervisor virtualization is more or less operating system agnostic when it comes to guest OSes, though in practice it's not guaranteed that every x86/x86-64 guest is going to run flawlessly on top of any given hypervisor-based solution.

Initially, virtualization options on Linux x86 were limited to full virtualization, where the virtualization software had to completely isolate the guest operating system and emulate the hardware entirely. The x86 platform was not originally designed with virtualization in mind, though the concept of virtualization had been around since the 70s. One of the first virtualization solutions on the scene for Linux was VMware's Workstation in 1998 or so. Back then, it was a novel idea that you could run a guest OS on top of Linux. A lot of folks used hypervisor virtualization as a way to run a couple of Windows apps on top of Linux in order to move away from Windows on the desktop -- or just to consolidate their OSes onto one workstation or laptop so they could avoid dual-booting or maintaining two machines. Performance took a hit, but overall it was usable for desktop machines.

Fast forward a bit more than a decade, and we have plenty of choice for running multiple OSes on the desktop and server. In fact, hypervisor virtualization is actually baked into the Linux kernel now in the form of the Linux Kernel Virtual Machine [2], which is a way to turn the Linux kernel into a hypervisor by adding a kernel module.

Some solutions offer what's called *paravirtualization*, which requires the guest operating systems to be modified. Paravirtualization allows the guest OS to interact more directly with the host system's hardware, and it provides a performance benefit for the guest OS. The downside is that the guest OS has to be modified to be aware that it's being virtualized. This limits one of the advantages of hypervisor virtualization, which is that you can typically run almost any operating system that's designed for that hardware architecture.

For example, using Linux as a guest or host OS for Xen is not a problem if the kernel supports it. It's not a default feature in the kernel, but many distros add the appropriate patches for Xen to work as a host (dom0) or guest (domU). You're a bit more limited, though, in running older releases of Microsoft Windows Server on top of Xen because they're not Xen-aware.

As mentioned, the x86 platform wasn't originally designed to handle virtualization. That's changed in the past few years, with Intel and AMD offering virtualization extensions to their x86-64 offerings. For Intel, this means the Intel Virtualization Technology [3] (VT) offerings and AMD's AMD-V [4] technology. If you're shopping hardware, be sure to check that your systems will have Intel VT or AMD-V, as not all new processors have virtualization extensions.

The primary advantage of hypervisor-based solutions is that they allow you to run a fuller range of operating systems. Solutions like KVM, VMware Server or ESX, Parallels Server Bare Metal and Xen (to name only a few) allow you to run just about any x86 operating system as a guest on a wide variety of host OSes.

This is very effective for server consolidation when you're looking at consolidating existing workloads into a virtualized environment. So, if you have 20 servers running Microsoft Windows Server 2000 and 50 servers running SUSE Linux Enterprise Server 9, you'd likely want to look at consolidating on top of one of the hypervisor solutions.

Performance, however, is likely to take at least a slight hit when running on a hypervisor. You're introducing an extra layer of abstraction between the operating system and hardware with hypervisor virtualization, and you'll see some performance impact as a result. You also have to have a complete OS stack for each guest when using hypervisor virtualization, from the kernel to libraries, applications, and so on. So you'll have additional storage overhead and memory use from running OSes entirely separate.

That being said, performance is less of a factor today than a few years ago. The various hypervisor solutions have been pretty well-optimized for heavy loads, and continue to improve at a good clip.

All About Containers

The other contender is container-based virtualization. Container-based virtualization is also called operating system virtualization, and sometimes it's not really talked about as virtualization at all. One of the first container technologies on x86 was actually on FreeBSD, in the form of FreeBSD Jails [5].

Instead of trying to run an entire guest OS, container virtualization isolates the guests, but doesn't try to virtualize the hardware. Instead, you have containers (hence the name) for each virtual environment. With container-based technologies, you'll need a patched kernel and user tools to run the virtual environments. The kernel provides process isolation and performs resource management. This means that even though all the virtual machines are running under the same kernel, they effectively have their own filesystem, processes, memory, devices, etc.

The net effect is very similar to hypervisor virtualization, and there's a good chance that users of the guest systems will never know the difference between using a system that's running on bare metal, under a

hypervisor, or in a container. However, the actual experience for admins and planning for deployment varies.

With container-based virtualization, installing a guest OS is not as straightforward as hypervisor solutions. That is, you can't just pop in a DVD or CD to whip up a new guest machine. You'll need to create a container template, if you're using something like OpenVZ or Parallels Virtuozzo Containers. Usually you'll not need to create these on your own, though as OpenVZ provides quite a few templates and Parallels provides supported templates for its users.

You're also usually limited to a single operating system with containers. You can't run Linux and Windows together, for example. In some configurations, you can run Linux and Solaris/OpenSolaris using Solaris Zones, but for Linux virtualization, you're limited to running Linux guests on a Linux host. You're not limited to a specific distribution, though. You can run Debian and RHEL templates, for instance, on top of a CentOS host running OpenVZ or Virtuozzo.

If this sounds limiting, remember there are also advantages to container-based virtualization in terms of performance and scalability. Hypervisor virtualization usually has limits in terms of how many CPUs and how much memory a guest can address, whereas the container-based solutions should be able to address as many CPUs and as much RAM as the host kernel.

It mostly breaks down to the type of workload you have. If you're going to be deploying dozens or hundreds of Linux guests, then a container-based solution works very well and might be a better option over hypervisor virtualization. Containers are especially popular in hosting environments or any scenario where there's a need to consolidate a large number of Linux instances.

Summary

The right type of virtualization depends heavily on the operating systems that you'll be deploying as well as the types of workloads that you need to run. To some extent it's also dependent on hardware, though this is mostly a major factor when working with older (by server-room standards) machines that don't support hardware-assisted virtualization. If you're not working with hardware that supports virtualization extensions, that will limit your options somewhat.

You also have the option of deploying workloads "in the cloud." Next week we'll cover cloud computing options and some other virtualization technologies for Linux, and follow up the week after with a strategy guide for organizations looking to deploy workloads on Linux in the short to medium term.

News Category:
System Administration [6]

Source URL: https://www.linux.com/news/containers-vs-hypervisors-choosing-best-virtualization-technology

Links:
[1] https://www.linux.com/news/containers-vs-hypervisors-choosing-best-virtualization-technology
[2] http://www.ibm.com/developerworks/linux/library/l-linux-kvm/
[3] http://www.intel.com/technology/virtualization/index.htm
[4] http://sites.amd.com/us/business/it-solutions/usage-models/virtualization/Pages/amd-v.aspx
[5] http://www.freebsd.org/doc/en/books/handbook/jails.html
[6] https://www.linux.com/news/category/system-administration