TODAY'S TOP STORIES

What are containers and why do you need them?

Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another. Here's what you need to know about this popular technology.

By Paul Rubens
CIO |
JUN 27, 2017 3:00 AM PT

Table of Contents

Docker exploded onto the scene in 2013, and it's been causing excitement in IT circles ever since.

The application container technology provided by Docker promises to change the way that IT operations are carried out just as virtualization technology did a few years previously.

Here are answers to 13 of the most common questions related to this technology.

[Beware the 9 warning signs of bad IT architecture and see why these 10 old-school IT principles still rule. | Sign up for CIO newsletters.]

What are containers and why do you need them?

Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another. This could be from a developer's laptop to a test environment, from a staging environment into production, and perhaps from a physical machine in a data center to a virtual machine in a private or public cloud.

Problems arise when the supporting software environment is not identical, says Docker creator Solomon Hykes. "You're going to test using Python 2.7, and then it's going to run on Python 3 in production and something weird will happen. Or you'll rely on the behavior of a certain version of an SSL library and another one will be installed. You'll run your tests on Debian and production is on Red Hat and all sorts of weird things happen."

And it's not just different software that can cause problems, he added. "The network topology might be different, or the security policies and storage might be different but the software has to run on it."

How do containers solve this problem?

Put simply, a container consists of an entire runtime environment: an application, plus all its dependencies, libraries and other binaries, and configuration files needed to run it, bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away.

What's the difference between containers and virtualization?

With virtualization technology, the package that can be passed around is a virtual machine, and it includes an entire operating system as well as the application. A physical server running three virtual machines would have a hypervisor and three separate operating systems running on top of it.

By contrast a server running three containerized applications with Docker runs a single operating system, and each container shares the operating system kernel with the other containers. Shared parts of the operating system are read only, while each container has its own mount (i.e., a way to access the container) for writing. That means the containers are much more lightweight and use far fewer resources than virtual machines.

What other benefits do containers offer?

A container may be only tens of megabytes in size, whereas a virtual machine with its own entire operating system may be several gigabytes in size. Because of this, a single server can host far more containers than virtual machines.

Another major benefit is that virtual machines may take several minutes to boot up their operating systems and begin running the applications they host, while containerized applications can be started almost instantly. That means containers can be instantiated in a "just in time" fashion when they are needed and can disappear when they are no longer required, freeing up resources on their hosts.

A third benefit is that containerization allows for greater modularity. Rather than run an entire complex application inside a single container, the application can be split in to modules (such as the database, the application front end, and so on). This is the so-called microservices approach. Applications built in this way are easier to manage because each module is relatively simple, and changes can be made to modules without having to rebuild the entire application. Because containers are so lightweight, individual modules (or microservices) can be instantiated only when they are needed and are available almost immediately.

What's the difference between Docker and containers?

Docker has become synonymous with container technology because it has been the most successful at popularizing it. But container technology is not new; it has been built into Linux in the form of LXC for over 10 years, and similar operating system level virtualization has also been offered by FreeBSD jails, AIX Workload Partitions and Solaris Containers.

Is there a standard container format?

Back in 2015, a company called CoreOS produced its own App Container Image (ACI) specification that was different from Docker's container specification, and at the time there was a risk that the newly-popular container movement would fragment with rival Linux container formats.

But later in the same year an initiative called the Open Container Project was announced, and later renamed as the Open Container Initiative (OCI). Run under the auspices of the Linux Foundation, the purpose of the OCI is to develop industry standards for a container format and container runtime software for all platforms. The starting point of the OCP standards was Docker technology, and Docker donated about 5 percent of its codebase to the project to get it off the ground.

The project's sponsors include AWS, Google, IBM, HP, Microsoft, VMware, Red Hat, Oracle, Twitter, and HP as well as Docker and CoreOS

Why are all these companies involved in the Open Container Initiative?

The idea of the OCI is to ensure that the fundamental building blocks of container technology (such as the container format) are standardized so that everyone can take advantage of them.

That means that rather than spending resources developing competing container technologies, organizations can focus on developing the additional software needed to support the use of standardized containers in an enterprise or cloud environment. The type of software needed includes container orchestration and management systems and container security systems.

Are there any free open source container management systems?

Yes. Probably the best known and most widely used free and open source container management systems is Kubernetes, which is a software project that originated at Google. Kubernetes provides mechanisms for deploying, maintaining and scaling containerized applications

What commercial container management solutions exist today?

Docker Enterprise Edition is perhaps the best known commercial container management solution. It provides an integrated, tested and certified platform for apps running on enterprise Linux or Windows operating systems and cloud providers.

But there are many others, and several notable ones have a layer of proprietary software built around Kubernetes at the core. Examples of this type of management software product include:

- CoreOS's Tectonic pre-packages all of the open source components required to build a Google-style infrastructure and adds additional commercial features, such as a management console, corporate SSO integration, and Quay, an enterprise-ready container registry.
- Red Hat's Open Shift Container Platform is an on-premises private platform as a service product, built around a core of application containers powered by Docker, with orchestration and management provided by Kubernetes, on a foundation of Red Hat Enterprise Linux.
- Rancher Labs' Rancher is a commercial open source solution designed makes it easy to deploy and manage containers in production on any infrastructure.

How secure are containers?

Many people believe that containers are less secure than virtual machines because if there's a vulnerability in the container host kernel, it could provide a way into the containers that are sharing it. That's also true with a hypervisor, but since a hypervisor provides far less functionality than a Linux kernel (which typically implements file systems, networking, application process controls and so on) it presents a much smaller attack surface.

But in the last couple of years a great deal of effort has been devoted to developing software to enhance the security of containers.

For example, Docker (and other container systems) now include a signing infrastructure allowing administrators to sign container images to prevent untrusted containers from being deployed.

However, it is not necessarily the case that a trusted, signed container is secure to run, because vulnerabilities may be discovered in some of the software in the container after it has been signed. For that reason, Docker and others offer container security scanning solutions that can notify administrators if any container images have vulnerabilities that could be exploited.

More specialized container security software has also been developed. For example, Twistlock offers software that profiles a container's expected behavior and "whitelists" processes, networking activities (such as source and destination IP addresses and ports) and even certain storage practices so that any malicious or unexpected behavior can be flagged.

Another specialist container security company called Polyverse takes a different approach. It takes advantage of the fact that containers can be started in a fraction of a second to relaunch containerized applications in a known good state every few seconds to minimize the time that a hacker has to exploit an application running in a container.

Which Linux distributions are suitable for use as a container host?

Most Linux distributions are unnecessarily feature-heavy if their intended use is simply to act as a container host to run containers. For that reason, a number of Linux distributions have been designed specifically for running containers.

Some examples include:

- Container Linux (formerly CoreOS Linux) — one of the first lightweight container operating systems built for containers
- RancherOS — a simplified Linux distribution built from containers, specifically for running containers.
- Photon OS — a minimal Linux container host, optimized to run on VMware platforms.
- Project Atomic Host — Red Hat's lightweight container OS has versions that are based on CentOS and Fedora, and there is also a downstream enterprise version in Red Hat Enterprise Linux.
- Ubuntu Core — the smallest Ubuntu version, Ubuntu Core is designed as a host operating system for IoT devices and large-scale cloud container deployments

What if you are a Windows shop?

In addition to running on any Linux distribution running version 3.10 (or later) of the Linux kernel, Docker also runs on Windows.

That's because in 2016 Microsoft introduced the ability to run Windows containers in Windows Server 2016 and Windows 10. These are Docker containers designed for Windows, and they can be managed from any Docker client or from Microsoft's PowerShell.

(Microsoft also introduced Hyper-V containers, which are Windows containers running in a Hyper-V virtual machine for added isolation.)

Windows containers can be deployed on a standard install of Windows Server 2016, the streamlined Server Core install, or the Nano Server install option which is specifically designed for running applications inside containers or virtual machines.

In addition to Linux and Windows, Docker also runs on popular cloud platforms including Amazon EC2, Google Compute Engine, Microsoft Azure and Rackspace.

Will containers eventually replace full-blown server virtualization?

That's unlikely in the foreseeable future for a number of important reasons.

First, there is still a widely held view that virtual machines offer better security than containers because of the increased level of isolation that they provide.

Second, the management tools that are available to orchestrate large numbers of containers are also not yet as comprehensive as software for managing virtualized infrastructure, such as VMware's vCenter or Microsoft's System Center. Companies that have made significant investments in this type of software are unlikely to want to abandon their virtualized infrastructure without very good reason.

Perhaps more importantly, virtualization and containers are also coming to be seen as complementary technologies rather than competing ones. That's because containers can be run in lightweight virtual machines to increase isolation and therefore security, and because hardware virtualization makes it easier to manage the hardware infrastructure (networks, servers and storage) that are needed to support containers.

VMware encourages customers who have invested in its virtual machine management infrastructure to run containers on its Photon OS container Linux distro inside lightweight virtual machines that can then be managed from vCenter. This is VMware's "container in a VM" strategy.

But VMware has also introduced what it calls vSphere Integrated Containers (VICs). These containers can be deployed directly to a standalone ESXi host or deployed to vCenter Server as if they were virtual machines. This is VMware's "container as a VM" strategy.

Both approaches have their benefits, but what's important is that rather than replacing virtual machines, it can often be useful to be able to use containers within a virtualized infrastructure.

Next read this:

- *20 ways to kill your IT career (without knowing it)*
- *16 time-saving tips for IT leaders*
- *CIO resumes: 6 best practices and 4 strong examples*
- *Why IT-business alignment still fails*
- *Top 12 ITSM tools for 2018*
- *6 most-dreaded IT projects*
- *CIO playbook: 10 tips for leading IT in the digital era*
- *15 ways to advance your IT career*
- *7 habits of highly effective digital transformations*

Paul Rubens is a technology journalist based in England.

Follow