Red Hat

Whitepaper

Demystifying application containers and virtual machines

Containers quick reference guide

Container

Technology that allows computer software to be packaged and isolated in a standardized, lightweight, and portable format.

Container runtime

Software that runs and manages the components of a container running on a host (e.g., Podman, CRI-O, Docker).

Open Container Initiative (OCI)

An open governance structure for creating open industry standards around container formats and runtimes.

Namespaces

A Linux concept for labeling processes, network interfaces, files, and other system components. Labeling is used to isolate containers from each other and the host system.

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com

Application containers have quickly become the foundation of modern software design and digital infrastructures. Their characteristics enable an essential transformation toward data-driven warfare and the development of highly agile and flexible warfighting capabilities that can be delivered at the speed of need.

While containers are not new, they may be unfamiliar to many IT professionals, and there is often confusion about what containers are and how they differ from virtual machines. Fortunately, the key to understanding these two related but fundamentally different concepts is in their names.

Virtual machines

A virtual machine (VM) is a software-defined computer with every component explicitly defined and represented such as memory, CPU, disk, and audio and video devices. Once a VM has been built, it is used like any other computer: an operating system and applications are installed and configured.

Application containers

An application container is precisely that: a container. It is not a software-defined computer like a virtual machine, and it does not have an operating system, disks, CPUs, memory, or other devices. There is no boot process for a container because there is no computer or operating system to boot.

Containers are standardized, portable packages that hold only the files and metadata necessary to define an application. Containers are loaded onto a host computer where they run like any other application: application binaries are executed and run as processes, which interact with the computer's network interfaces, disks, and other processes. This is different from a virtual machine, which runs applications on top of an operating system and software-defined hardware.

While containers run like regular applications on a host, they differ from applications that are not containerized in that containers are managed and run by container runtimes, which use features of the host such as namespaces and other security features to isolate applications from each other. A namespace is defined by a tag that is assigned to every file, process, and network connection associated with a particular container, and only components with the same namespace tag can interact with each other, which maintains strict isolation of running containers.

SELinux

A security-focused architecture for Linux, developed jointly by Red Hat and the National Security Agency. It is used to enforce container isolation and provide security controls over containers and container runtimes.

Kubernetes

A system used to manage, deploy, scale, and orchestrate containers across an organization.

Containers in the Department of Defense (DoD)

Heterogeneous defense architectures are made more efficient and effective with containers.

- Containers accelerate the speed at which new capabilities can be delivered to the warfighter.
- Build containers once and deploy many times across public cloud, private cloud, weapon systems, and on the tactical edge.
- Containers are lightweight and ideal for small-form factor tactical hardware platforms.
- Immutable containers can be secured to DoD standards when they are built, and these standards follow them everywhere they go.

Why use containers?

Both containers and virtual machines have their place in an IT infrastructure, and which one is appropriate depends on the use case. Some large, monolithic applications like traditional databases do not containerize well. However, many applications do, and the size, portability, and security characteristics of containers make them ideal for modern distributed and collaborative defense architectures such as hybrid cloud, edge computing, and microservices.

Size

Because containers are composed only of application files, they are much smaller than virtual machines and easier to move around a hybrid cloud, physical, or edge architecture. Additionally, because they are processes running on a host computer, they require fewer resources than virtual machines, vastly increasing the maximum density per host to dozens or hundreds of containers instead of a few virtual machines.

Containers also allow software developers to test their code efficiently and frequently because they are lightweight, start fast, consume few resources, and can be destroyed as quickly as they are created. Containers start up in seconds and do not require interaction with operations teams like virtual machines do. Ask developers how much time and productivity they lose when they drop out of their flow and the benefits of using containers for near-real-time code testing become evident.

Portability

Just like a shipping container that can be moved from a cargo ship to a tractor trailer and to a railroad car, the standard form factor of an application container gives it similar flexibility. Application containers can run anywhere that their parent operating system can run. For example, a Linux® container can be built once and run unchanged anywhere that Linux runs. Containers are commonly built to be OCI-compliant, which ensures that they will run on an OCI-compliant host without requiring the container to be changed.

Conversely, virtual machine formats are incompatible across hypervisors. For example, a VMWare virtual machine won't run directly on Amazon EC2 and an EC2 Amazon machine image won't run on a kernel-based virtual machine hypervisor. Import and export mechanisms exist, but these necessarily change the virtual machine format, take time, and potentially introduce errors. There is no guarantee that everything will work once the virtual machine is migrated between hypervisors.

When software developers build their applications directly into container images, they are able to promote the exact same images along their DevSecOps pipeline until the applications are put into production. The container that a developer is working on is the same container that goes into production, byte for byte, regardless of whether the container host is bare metal, an on-premise virtual machine, in a public cloud, or on an edge device. Using consistent, immutable containers is far more predictable and less error-prone than deploying software on virtual machines on different hypervisors across and within different environments.

Security

A virtual machine-based architecture involves one operating system on each host (hypervisor) plus a completely separate operating system for every virtual machine that is running—100 virtual machines means at least 101 operating systems to manage. This is a massive attack surface to maintain, patch, and monitor in addition to securing the applications that run on each virtual machine.

- Containers are important components of microservices architectures, which are relevant as large, exquisite military assets are deconstructed into aggregates of smaller, attritable assets.

Learn more

Understanding Linux containers

Open Container Initiative

Cloud Native Computing Foundation

Kubernetes project

Learning Red Hat® OpenShift®

Conversely, with containers, there is only one operating system to secure—the container host. Managing only one operating system is less burdensome than managing virtual machines, especially at scale, and developers can focus on securing the applications themselves. The security features of the host operating system are also inherited by the containers and provide strong isolation and defense-grade protection (SELinux, Cgroups, seccomp, and more).

Additionally, as a container image moves through the development cycle, it can be vetted and hardened by security tests and scans before it is promoted into production. Once promoted, because the container does not change after it is hardened, the security posture of that container is the same as during the final stages of development as validated by its unchanged cryptographic signature.

Summary

There is far more to learn about containers, such as managing containers at scale, orchestration with Kubernetes, container registries, service meshes, or the myriad other technologies that are part of this domain. Before starting down this path, it is important to have a clear understanding of what containers are and what they are not in order to effectively use this tool.

North America
1 888 REDHAT1
www.redhat.com

Europe, Middle East, and Africa
00800 7334 2835
europe@redhat.com

Asia Pacific
+65 6490 4200
apac@redhat.com

Latin America
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com
#F25654_1120