BlackLinux

Introduction to Container Technology ?

Software applications typically depend on other libraries, configuration files, or services provided by their runtime environment. Traditionally, the runtime environment for a software application is installed in an operating system running on a physical host or virtual machine. Any application dependencies are installed along with that operating system on the host.

In Red Hat Enterprise Linux, packaging systems like RPM are used to help manage application dependencies. When you install the httpd package, the RPM system ensures that the correct libraries and other dependencies for that package are also installed.

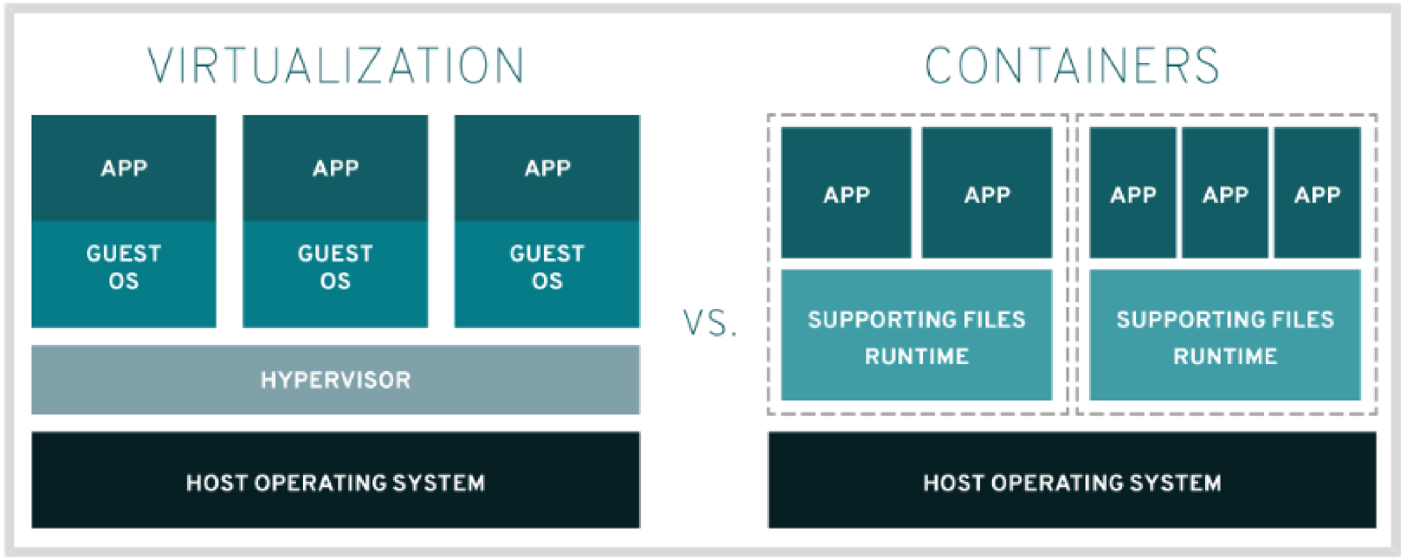
The major drawback to traditionally deployed software applications is that these dependencies are entangled with the runtime environment. An application might require versions of supporting software that are older or newer than the software provided with the operating system. Similarly, two applications on the same system might require different versions of the same software that are incompatible with each other.

One way to resolve these conflicts is to package and deploy the application as a container. A container is a set of one or more processes that are isolated from the rest of the system.

Think of a physical shipping container. A shipping container is a standard way to package and ship goods. It is labeled, loaded, unloaded, and transported from one location to another as a single box. The container's contents are isolated from the contents of other containers so that they do not affect each other.

Comparing Containers to Virtual Machines

Containers provide many of the same benefits as virtual machines, such as security, storage, and network isolation.
Both technologies isolate their application libraries and runtime resources from the host operating system or hypervisor and vice versa.



Containers and Virtual Machines are different in the way they interact with hardware and the underlying operating system.

Virtualization:

• Enables multiple operating systems to run simultaneously on a single hardware platform.
• Uses a hypervisor to divide hardware into multiple virtual hardware systems, allowing multiple operating systems to run side by side.
• Requires a complete operating system environment to support the application. Compare and contrast this with containers, which:
• Run directly on the operating system, sharing hardware and OS resources across all

Containers

On the system. This enables applications to stay lightweight and run swiftly in parallel.
• Share the same operating system kernel, isolate the containerized application processes from the rest of the system, and use any software compatible with that kernel.
• Require far fewer hardware resources than virtual machines, which also makes them quick to start and stop and reduce storage requirements.

Exploring the Implementation of Containers

Red Hat Enterprise Linux implements containers using core technologies such as:

• Control Groups (cgroups) for resource management.
• Namespaces for process isolation.
• SELinux and Seccomp (Secure Computing mode) to enforce security boundaries.

Running Containers from Container Images

Containers are run from container images. Container images serve as blueprints for creating containers. Container images package an application together with all its dependencies, such as:

- System libraries
- Programming language runtimes
- Programming language libraries
- Configuration settings
- Static data files

Container images are unchangeable, or immutable, files that include all the required code and dependencies to run a container.

Container images are built according to specifications, such as the Open Container Initiative (OCI) image format specification. These specifications define the format for container images, as well as the metadata about the container host operating systems and hardware architectures that the image supports.

Managing Containers with Podman

A good way to start learning about containers is to work with individual containers on a single server acting as a container host. Red Hat Enterprise Linux provides a set of container tools that you can use to do this, including:

- podman, which directly manages containers and container images.
- skopeo, which you can use to inspect, copy, delete, and sign images.
- buildah, which you can use to create new container images.

These tools are compatible with the Open Container Initiative (OCI). They can be used to manage any Linux containers created by OCI-compatible container engines, such as Docker. These tools are specifically designed to run containers under Red Hat Enterprise Linux on a single-node container host.

Running Rootless Containers

On the container host, you can run containers as the root user or as a regular, unprivileged user. Containers run by non-privileged users are called rootless containers.
Rootless containers are more secure, but have some restrictions. For example, rootless containers cannot publish their network services through the container host's privileged ports (those below port 1024).
You can run containers directly as root, if necessary, but this somewhat weakens the security of the system if a bug allows an attacker to compromise the container.