Learn the differences between Containers and Virtual Machines

By

Stanley Stephen

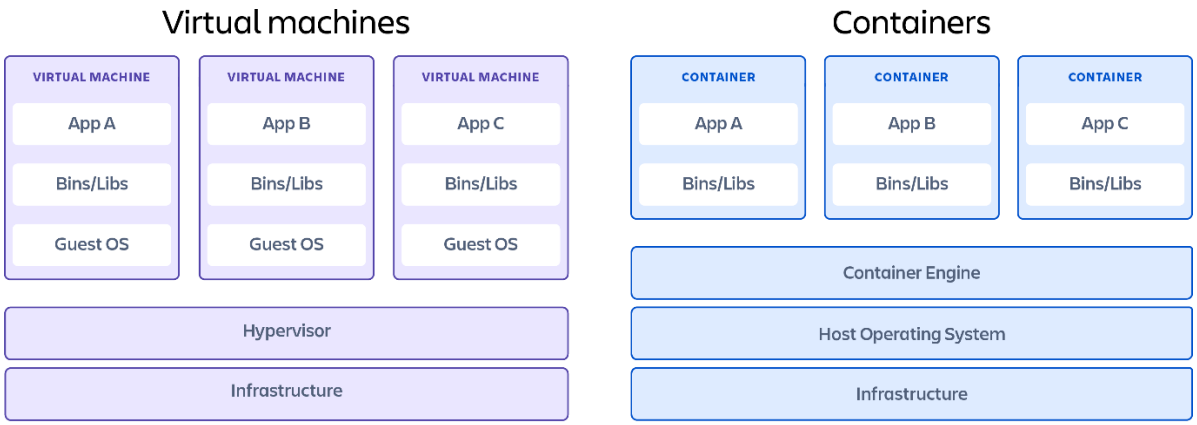
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Containers vs. Virtual Machines

Let us learn the differences between containers and virtual machines (VMs), popular providers for each, and how they can be used together.

Containers and VMs are very similar resource virtualization technologies. Virtualization is the process in which a system singular resource like RAM, CPU, Disk, or Networking can be 'virtualized' and represented as multiple resources. The key differentiator between containers and VMs is that VMs virtualize an entire machine down to the hardware layers and containers only virtualize software layers above the operating system level.



What is a container?

Containers are lightweight software packages that contain all the dependencies required to execute the contained software application. These dependencies include things like system libraries, external 3rd party code packages, and other operating system level applications. The dependencies included in a container exist in stack levels that are higher than the operating system.

Pros

- **Iteration speed**
  Because containers are lightweight and only include high level software, they are very fast to modify and iterate on.

- **Robust Ecosystem**
  Most container runtime systems offer a hosted public repository of pre-made containers. These container repositories contain many popular software applications like databases or messaging systems and can be instantly downloaded and executed, saving time for development teams

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Cons

- **Shared host exploits**
  Containers all share the same underlying hardware system below the operating system layer, it is possible that an exploit in one container could break out of the container and affect the shared hardware. Most popular container run-times have public repositories of pre-built containers. There is a security risk in using one of these public images as they may contain exploits or may be vulnerable to being hijacked by nefarious actors.

Popular container providers

- **Docker**
  Docker is the most popular and widely used container runtime. Docker Hub is a giant public repository of popular containerized software applications. Containers on Docker Hub can instantly downloaded and deployed to a local Docker runtime.

- **RKT**
  Pronounced "Rocket" RKT is a security first focused container system. Rkt containers do not allow insecure container functionality unless the user explicitly enables insecure features. RKT containers aim to address the underlying cross contamination exploitive security issues that other container runtime systems suffer from.

- **Linux Containers – LXC**
  LXC is one component of an open source linux container runtime system. LXC is used to isolate operating system level processes from each other. Docker actually uses LXC behind the scenes. Linux Containers aim to offer a vender neutral open source container runtime.

- **CRI-O**
  CRI-O is an emerging open source container runtime standard that is being developed in collaboration by many enterprise companies. The CRI-0 specification is optimized for the Kubernetes container management system.

What is a VM?

VMs are heavy software packages that provide complete emulation of low level hardware devices like CPU, Disk and Networking devices. VMs may also include a complementary software stack to run on the emulated hardware. These hardware and software packages combined produce a fully functional snapshot of a computational system.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Pros

- **Full isolation security**
  VMs run in isolation as a fully standalone system. This means that VM's are immune to any exploits or interference from other VMS on a shared host. An individual VM can still be hijacked by an exploit but the exploited VM will be isolated and unable to contaminate any other neighboring VMs.

- **Interactive Development**
  Containers are usually static definitions of the expected dependencies and configuration needed to run the container. VMs are more dynamic and can be interactively developed. Once the basic hardware definition is specified for a VM the VM can then be treated as a bare bones computer. Software can manually be installed to the VM and the VM can be snapshotted to capture the current configuration state. The VM snapshots can be used to restore the VM to that point in time or spin up additional VM's with that configuration.

Cons

- **Iteration speed**
  VMs are time consuming to build and regenerate because they encompass a full stack system. Any modifications to a VM snapshot can take significant time to regenerate and validate they behave as expected.

- **Storage size cost**
  VMs can take up a lot of storage space. They can quickly grow to several Gigabytes in size. This can lead to disk space shortage issues on the VMs host machine

Popular VM providers

- **Virtualbox**
  Virtualbox is a free and open source x86 architecture emulation system owned by Oracle. Virtualbox is one of the most popular and established VM platforms with an ecosystem of supplementary tools to help develop and distribute VM images.

- **VMware**
  VMware is a publicly traded company that has built its business on one of the first x86 hardware virtualization technologies. VMware comes included with a hypervisor which is a utility that will deploy and manage multiple VMs. VMware has robust UI for managing VMs. VMware is a great enterprise VM option offering support.

- **QEMU**
  QEUM is the most robust hardware emulation VM option. It has support for any generic hardware architecture. QEMU is a command line only utility and does not offer a graphical user interface for configuration or execution. This trade off makes QEMU one of the fastest VM options.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com

Which option is better for you?

If you have specific hardware requirements for your project, or you are developing on one hardware platform and need to target another like Windows vs MacOS, you will need to use a VM. Most other 'software only' requirements can be met by using containers.

How can you use Containers and VMs together?

It is entirely possible to use Containers and VM in unison although the practical use-cases may be limited. A VM can be created that emulates a unique hardware configuration. An operating system can then be installed within this VM's hardware. Once the VM is functional and boots the operating system, a container runtime can be installed on the operating system. At this point we have a functional computational system with emulated hardware that we can install Containers on.

One practical use for this configuration is experimentation for system on chip deployments. Popular system on chip computational devices like The RaspberryPi, or BeagleBone development boards can be emulated as a VM, to experiment with running containers on them before testing on the actual hardware.

But the majority of the time, your needs will likely be met by one of the two. The key to deciding between containers or VMs for your virtualization needs is understanding your resource needs and the trade-offs you're willing to make.

Stanley Stephen, M.C.A.,
Linkedin: https://www.linkedin.com/in/contactstanley/
Github: https://github.com/stanleymca
Email: s.stanley.mca@gmail.com