CLOUD

Containers vs. VMs: What's the Difference & When to Use Them

Explore the difference between containers vs. VMs and how each can help you deploy and maintain your applications.

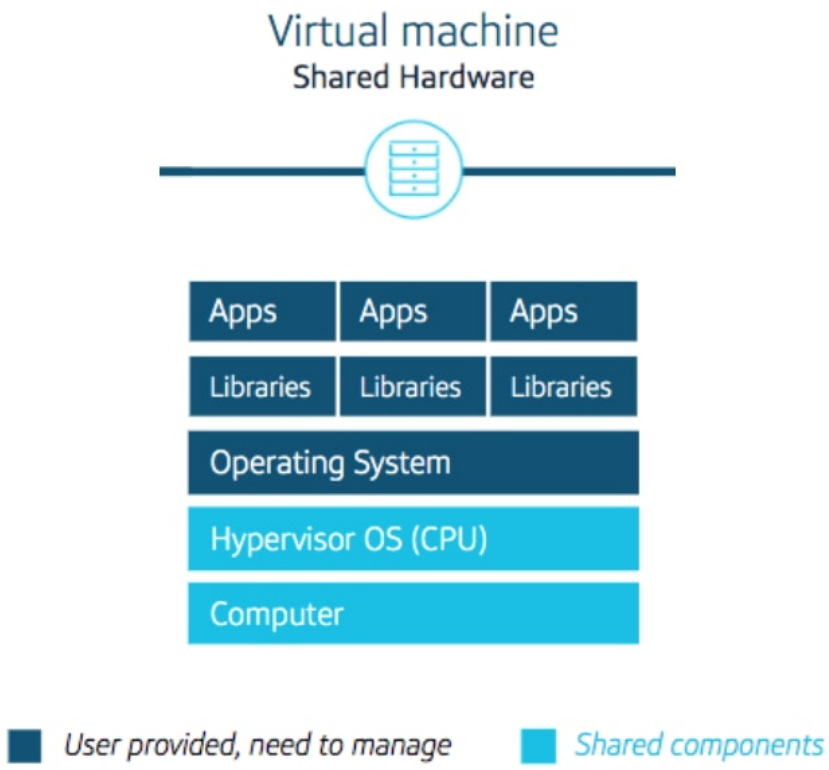
June 2, 2020

Containers and Virtual Machines (VMs) have fostered the creation of cloud technology as we know it today. Leveraging cloud technology is now vital for corporations, and an important factor to successfully migrate to and operate in the cloud is understanding the difference between containers and VMs. With the invention of these new virtualization technologies, there are many different ways that enterprises can deploy and maintain their applications. Today we will compare containers vs. VMs, explore the history of each, and weigh the pros and cons of both for your applications.

- What are Virtual Machines (VMs)?
- What are Containers?

- Container vs. VM Pros and Cons
- When to Use Containers vs. VMs
- Summary: Containers vs. VMs

What are Virtual Machines (VMs)?



Virtual Machines are an abstraction of computer hardware that allow an operating system to emulate its hardware for use by one or many guest operating systems. The software responsible for emulating hardware for these guest operating systems is called a hypervisor. A hypervisor manages these virtual machines by giving them a share of the resources (CPU, memory, disk, etc) of the host computer and its own operating system. Each virtual machine behaves like an individual machine with distinct hardware and resources. This means that an application is not responsible for sharing its resources with other applications, it can use all of the resources available to it because the hypervisor gives each VM their own virtual hardware that is separate from other VMs on the same server.

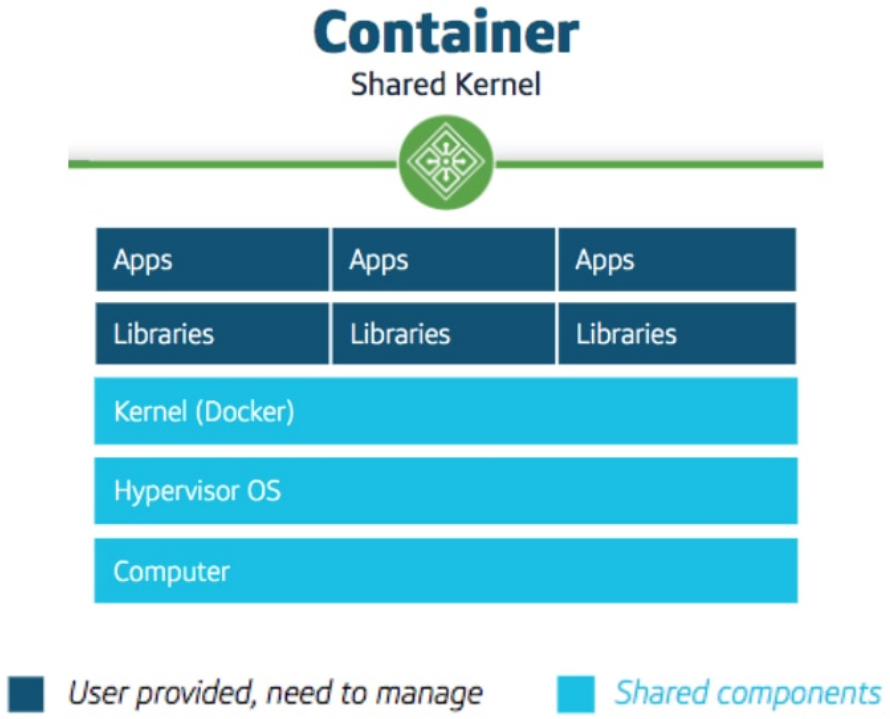
VMs also limit the negative impact of a vulnerability because they do not share operating systems and have their own allocated resources. The ability for VMs to divide up a single physical machine into different virtual entities enables industries to use hardware more efficiently while maintaining application isolation.

There are numerous providers that allow you to run VMs on prem or in the cloud. A small sample of

these VM providers include:

- VirtualBox (on-prem and open source)
- VMWare (open-prem or cloud)
- AWS EC2 or Lightsail (cloud)
- Azure VMs (cloud)

What is a Container?



Today, when people talk about containers, they're usually referring to Linux containers. Linux containers are comprised of control groups (cgroups), which allow the Linux kernel to limit and isolate physical resources like CPU, memory, and network I/O to a group of processes. This is similar to the way that a hypervisor creates resources for a virtual machine guest operating system, but with Linux cgroups you are able to divide resources directly within the Linux operating system, regardless of the presence of a hypervisor. With containers, you run your applications with their own separate resources without having to emulate an entire guest operating system.

Both containers and VMs make running multiple applications in production easier. Containers run directly on the host operating system through a container engine and each container can have its own application files, binaries and libraries. Containers can launch and scale in a matter of seconds because they don't need their own full operating system. When considering container vs. VM cost in cloud

computing, it's important to recognize that you're really comparing the cost of running containers on top of VMs to running VMs alone. Many cloud providers depend on VM technology to provide users with their personal servers, so if you run containers in the cloud you're most likely running containers on top of VMs that have been provisioned for you. In cloud computing, the key factor here for reducing cost is taking advantage of container technology by improving deployment density and making applications more lightweight and portable so multiple applications can run on a single VM, rather than utilizing multiple VMs.

Like VMs, there are many different providers and ways you can run containers. Container engines, container types, and whether you're developing on prem or in the cloud will affect your decision for a container provider, but a few examples include:

- Linux Containers
- Docker Containers
- Windows Server Containers

To learn more about containers, read: *What is a Container? Definition, Benefits, and Use Cases*.

Containers vs. VMs: Pros and Cons

Containers

Containers have the advantage of running directly in a container engine instead of on a guest operating system managed by a hypervisor. This makes them portable, lightweight, and easily scalable. These attributes make containers perfect for microservices and cloud applications that need to scale quickly or across multiple environments. Comparing the VM hypervisor vs container engine shows a downside to containers, however. Because containers are managed directly by a shared kernel, there is less isolation with containers as each container does not have its own dedicated set of physical resources like a virtual machine does. It can also be difficult to containerize applications if you haven't used them before or are looking for a way to migrate an existing monolithic application.

Virtual Machines (VMs)

VMs make it easier to onboard a monolithic application that was designed to run on its own inside of a full operating system. After onboarding though, VMs end up being less efficient and less scalable than containers due to how relatively large and slow to start VMs are. Further, VMs have to be managed by a hypervisor, which uses a large percentage of system resources just to manage the virtual machines. When you run applications in the cloud, you're also likely taking advantage of your cloud provider's hypervisor that creates a VM for you to use as a server. If you're then running your own hypervisor on this server to provision VMs, you will take up more resources that could be used for your applications if

you only used a container engine.

Docker vs. VMs

What about Docker vs. VM solutions? Docker is one very popular means of building and running containers, but in the seven or so years since Docker launched we've seen other popular solutions for running containers on Linux, such as containerd, CRI-O, and Podman. Comparing Docker to VMs isn't really comparing apples to apples because Docker is a container tool. As we've learned, in the cloud containers run on top of VMs, so you'd be using Docker in addition to VMs instead of one or the other.

When to Use Containers vs. VMs

When considering whether containers or VMs are right for you, the question isn't really containers vs. VMs, but rather containers *and* VMs, or just VMs. Because public cloud environments rarely dedicate entire servers to one application you're almost always taking advantage of a hypervisor to run your applications, even if you don't realize it. When you run your applications in the cloud you're deciding between using containers and VMs, or just using VMs.

Containers

Containers have greater flexibility and allow better resource utilization than VMs alone, and using containers and VMs together can be a powerful way to run your applications. If you're developing a newer application that is focused on portability, scalability and maintainability, you should consider containerized applications run with a container engine. Cloud native applications, applications that have been broken up into individual components, and applications that need high scalability and portability are great use cases for containers.

If you'd like to use containers but are finding it difficult to get started, this post on container management may be a good resource: How to Define a Holistic Container Management Strategy.

Virtual Machines (VMs)

On the other hand, if you have an existing monolithic application that you do not plan to rearchitect, or if your application needs to interact with an operating system as if it were being run directly on a machine, then you should consider VMs. If direct resource isolation is the most important runtime feature for your application, then virtual machines on their own may better meet your needs even though virtual machines are not as lightweight as containers.. Depending on the number of applications you need to run, a great option is to use VMs for your applications that cannot be containerized, and using containers and VMs together to reap the benefits of containers where you can.

Summary: Containers vs. VMs

Below is a table comparing the characteristics of containers vs. VMs side-by-side. From speed and size to the complexity and ongoing interaction required by development teams, this chart should help

summarize the key differences between containers and VMs.

Containers vs. VMs Comparison Chart

Component	Containers	Virtual Machines
Emulation Layer	Uses container engine to isolate multiple processes in a shared OS kernel	Uses hypervisor to manage guest operating systems
Speed	Starts in seconds	Starts in minutes
Size	Sized in megabytes	Sized in gigabytes
Minimum Cost	Tens of dollars	Hundreds of dollars
Scalability	Easy to scale with speed and small size	Supports some scaling but will reach compute resource limits quickly
Replaceability	Easy	Difficult
Resource Isolation	Moderate	Strong
Monolithic Applications	Inefficient for monolithic applications	Supports monolithic applications
Complexity for Dev Teams	Low	High
Dev Team Interaction with Infrastructure	Low	High

VMs may have preceded containers, but they are by no means obsolete. Especially in the world of cloud technology today, much of the infrastructure that you would use to run containers depends on VMs to exist in the first place. With these two powerful technologies, the best solution for you depends on your application needs. In other words, can you take advantage of the benefits that containers provide or do you need to use an entire VM for your application? If you determine that containers are the best solution for your organization, Capital One offers a container orchestration tool that helps you manage containers safely and effectively. Learn more about container orchestration with Critical Stack now.