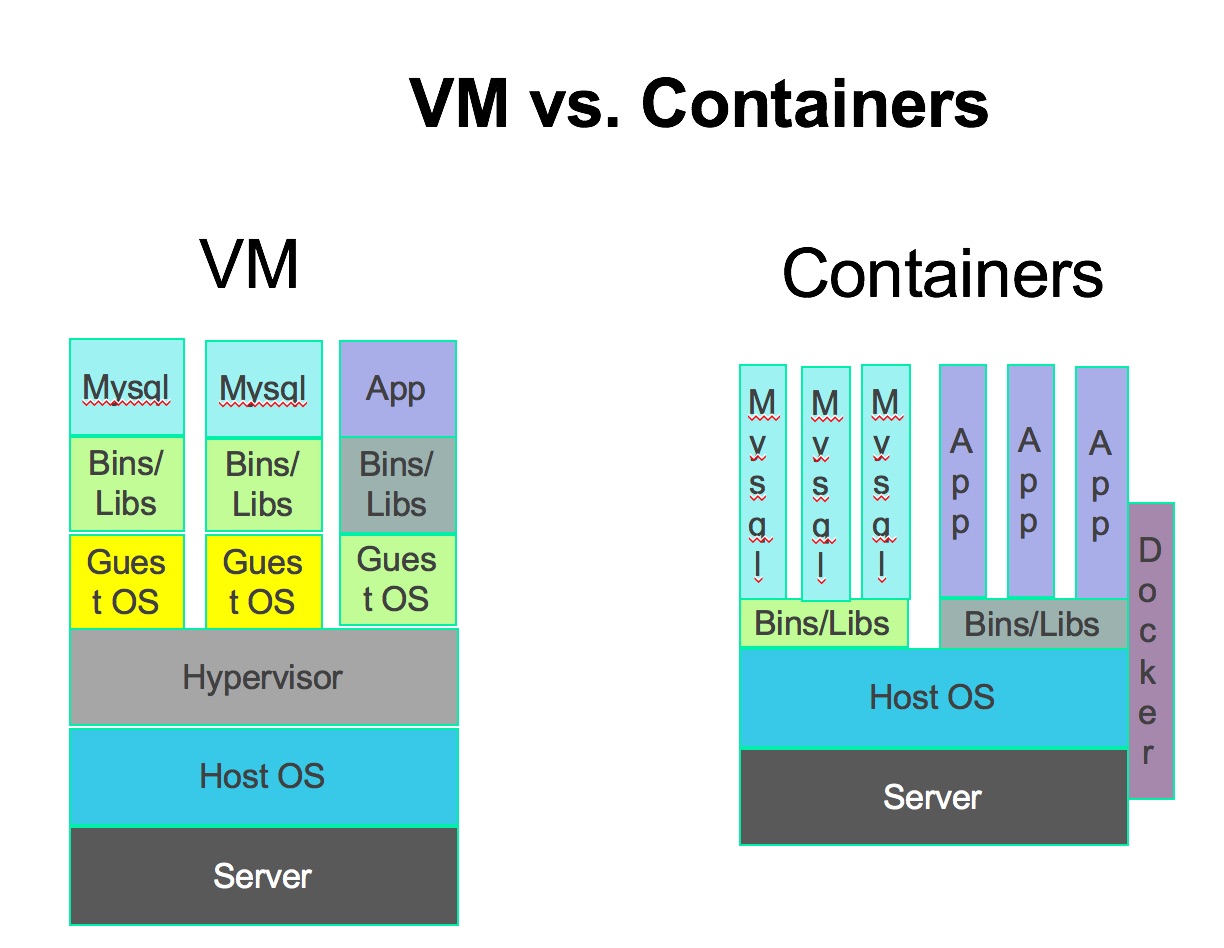
[[](https://dyn.com/wp-content/uploads/2018/02/vms_vs_container-1.jpg)](https://dyn.com/wp-content/uploads/2018/02/vms_vs_container-1.jpg)

**What is a container?**

Most people involved in technology know what a virtual machine is.

In a nutshell, a virtual machine is an instance of a distinct computer system with an operating system and any number of installed applications the uses emulation software that runs on a host system (usually a real, bare metal system, though it can also be nested virtual). This is made possible either full virtualization or hardware-assisted virtualization, both providing the layer required to run a guest operating system in full isolation.

There are a number of systems for running virtual machines such as VMWare, Virtualbox, Xen, KVM, and various others.

One of the characteristics of a virtual machine is that it provides complete isolation in terms of having its own processes, networking, users, etc., which are separate from the host system and other guest systems that may be running alongside it and not visible on the host system or vice-versa. Furthermore, virtual machines can be built to whatever specification is desired with packages pre-installed and configured, of any number of operating systems and operating system vender variants and saved as an image.

Containers are similar to virtual machines in many ways, but also different.

Just as with virtual machines, containers are instances that run on a host (bare metal or virtual) machine. Like virtual machines, they can be customized and built to whatever specification is desired, and can be used in most part the way a virtual machine is used, allowing isolated processes, networking, users, etc.

How containers differ from virtual machines is that a guest operating system is not installed, and usually consists only the application code and when run, only run the necessary process(es) that one uses the container for. This is because containers are made possible using kernel features of the host operating system and layered file system instead of the aforementioned emulation layer required to run virtual machines. This also means that containers don’t consist of different operating systems with installed applications, but can have the necessary components that set them aside as different Linux vender versions and variants. Even more so, this means that since a container doesn’t require its own operating system, it uses fewer resources and consumes only the resources required for the application that is run upon starting the container. This makes it so applications can consist of smaller containerized components as opposed to legacy monolithic applications installed on a virtual or bare metal systems.

How containers are similar to virtual machines is that they also are stored as images, though a big difference is that container images are much smaller and more portable and feasible to use than virtual machine images for the aforementioned reasons of not requiring an operating system installation as part of the image. This makes it possible to have a packaged, ready-to-use application that runs the same regardless of where as long as the host system runs containers (Linux containers specifically).

This small footprint and overall portability of containers compared to virtual machines is key to understanding how a system Linux Kubernetes has caught on: Make a mental note!

It’s common when one starts working with containers to still think of them as virtual machines and focus on the concept of virtualization, where in fact it’s more about the concept of isolation that is how containers could generally be described.

Now that we’ve discussed the fundamentals of containers, in my next blog I will talk about how to run containers at scale. Hint: Kubernetes plays a big role.