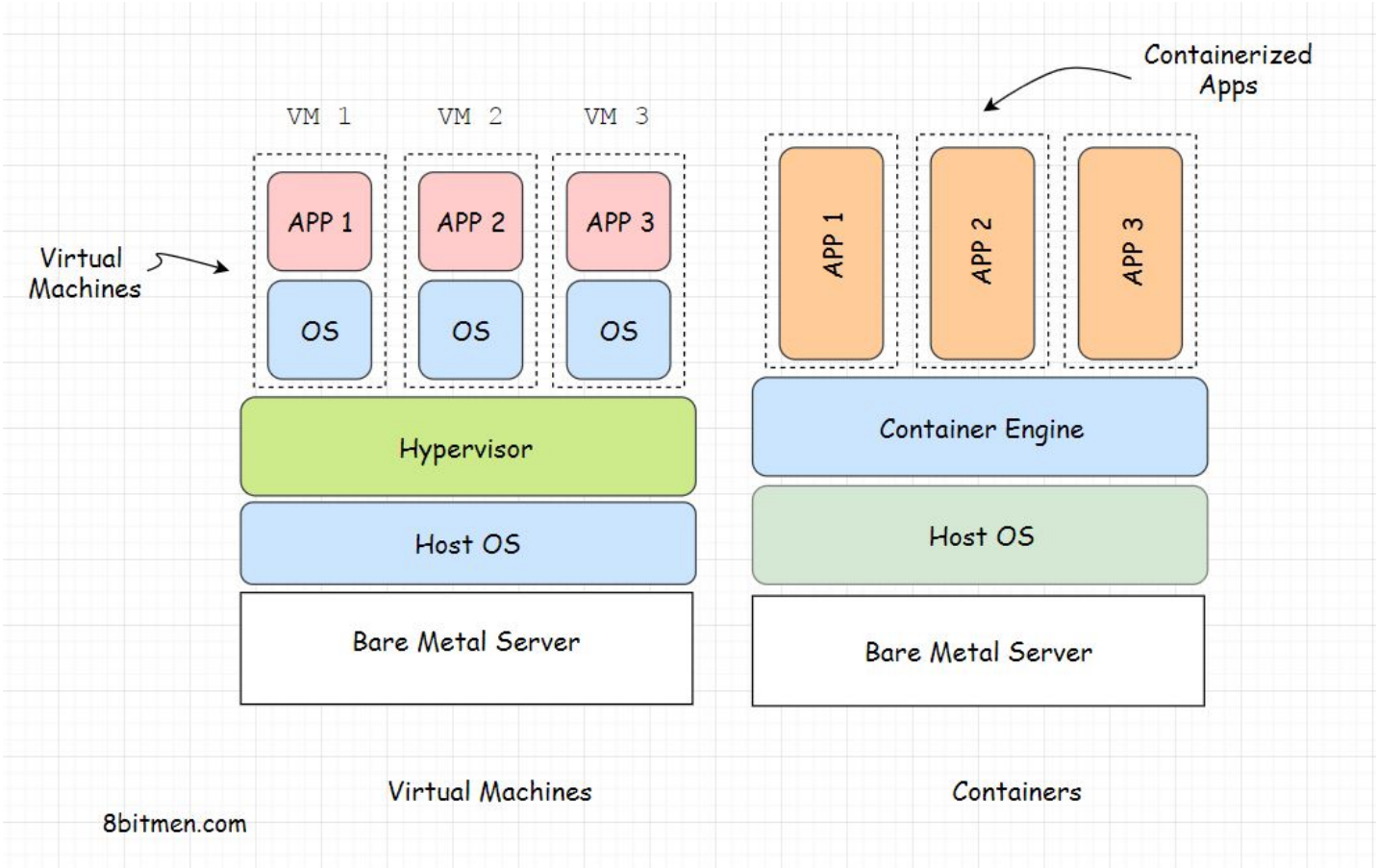
Containers and VMs – Picking the Right Technology

This lesson discusses the differences between containers and virtual machines.

We'll cover the following

- Differences between containers and virtual machines

Differences between containers and virtual machines



As you can see in the diagram, virtual machines provide an abstraction to the applications at the hardware level. They have their own *CPUs, OS, memory*, etc. On the contrary, containers provide an abstraction at the operating system level. This makes them more lightweight.

Since every container doesn't have its own copy of *OS*, they are more lightweight, in terms of the disk memory they consume. They are more performant as there is no *OS* boot time involved when running the *VMs'* containers.

There is a significant difference between the boot time of a *container* and a *VM*. The startup time of a *VM* may take seconds to minutes. The startup time of a container is in milliseconds.

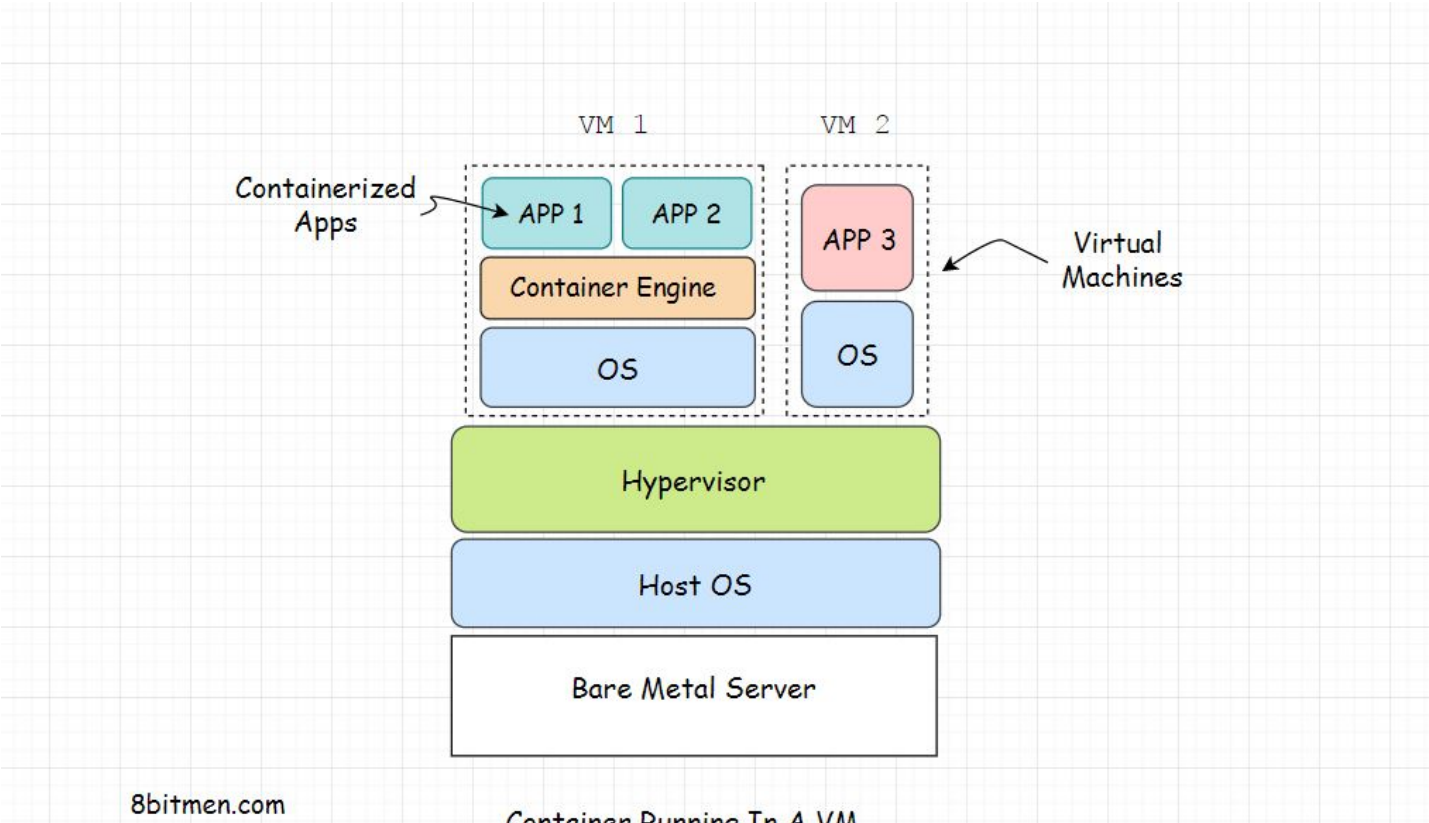
Also, technically, both technologies are different and are not really comparable. Since *VMs* provide an abstraction at the hardware level, they are more secure than containers.

Containers are dependent on the underlying *OS*; this puts a limit on using other operating systems. All the containers have to share and run on the same operating system.

So, if we have a use case where we need to run multiple instances of the application on the same *OS*, containers are a good fit. Containers are preferred when the business needs a more agile lightweight solution with less boot time.

VMs are preferred when the business needs more security and flexibility in terms of operating systems running in each *VM*.

A cloud provider ideally leverages both of the technologies together to fulfill the desired deployment approach. Containers can also be deployed on *VMs*. This largely depends on how the provider wants to run their infrastructure. Most of the *Kubernetes* deployments have containers running on *VMs*.



Container Running In A VM

Imagine running multiple instances of many microservices directly on *VMs*. Every *VM* will need a separate *OS* and a larger disk memory to run. This will have licensing costs, as the number of OS running in the infrastructure is high. From a cost standpoint, running containers is less expensive than running the *VMs*.

Containers are also intrinsically portable. If a service is running *on-prem* using containers, it's easy to migrate the service to a public cloud platform or any other platform, as opposed to migrating a service to run directly on *VMs* on-prem.

> ***Recommended read***: Evolution of container usage at Netflix; here is another interesting article, explaining how Docker helped turbocharge deployments at Uber.

Ultimately, containers are best suited for microservice deployment. There will be more on this in the lesson up next, but first, there will be a quick quiz.