SOFTWARE CONTAINERS

By Labib TERRISSA

WHAT'S THE DIFFERENCE BETWEEN CONTAINERS AND VIRTUALIZATION?

With virtualization technology, the package that can be passed around is a virtual machine, and it includes an entire operating system as well as the application. A physical server running three virtual machines would have a hypervisor and three separate operating systems running on top of it.

By contrast a server running three containerized applications with Docker runs a single operating system, and each container shares the operating system kernel with the other containers. Shared parts of the operating system are read only, while each container has its own mount (i.e., a way to access the container) for writing. That means the containers are much more lightweight and use far fewer resources than virtual machines.

WHY DO YOU NEED CONTAINERS?

- Containers are a solution to the problem of how to get software to run reliably when moved from
- one computing environment to another.
- This could be from a developer's laptop to a test environment,
- From a staging environment into production,
- and perhaps from a physical machine in a data center to a virtual machine in a private or public cloud.

WHY DO YOU NEED CONTAINERS?

- Problems arise when the supporting software environment is not identical,

- example

"You're going to test using Python 2.7, and then it's going to run on Python 3 in production and something weird will happen. Or you'll rely on the behavior of a certain version of an SSL library and another one will be installed. You'll run your tests on Debian and production is on Red Hat and all sorts of weird things happen."

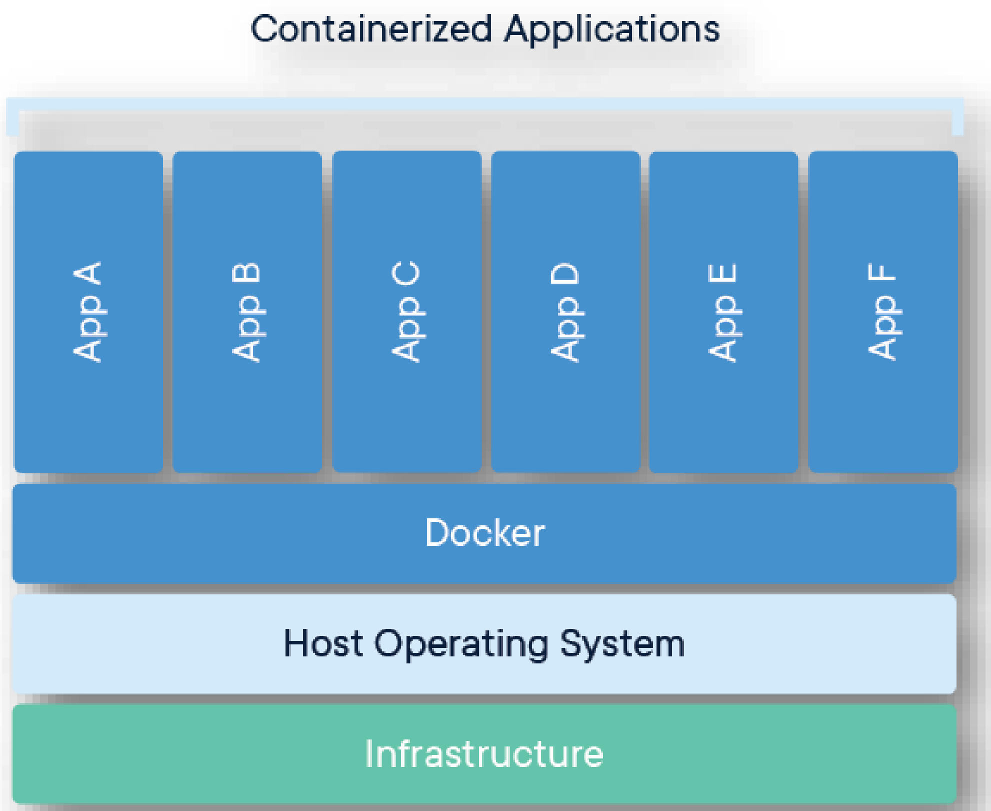
WHAT IS A CONTAINER?

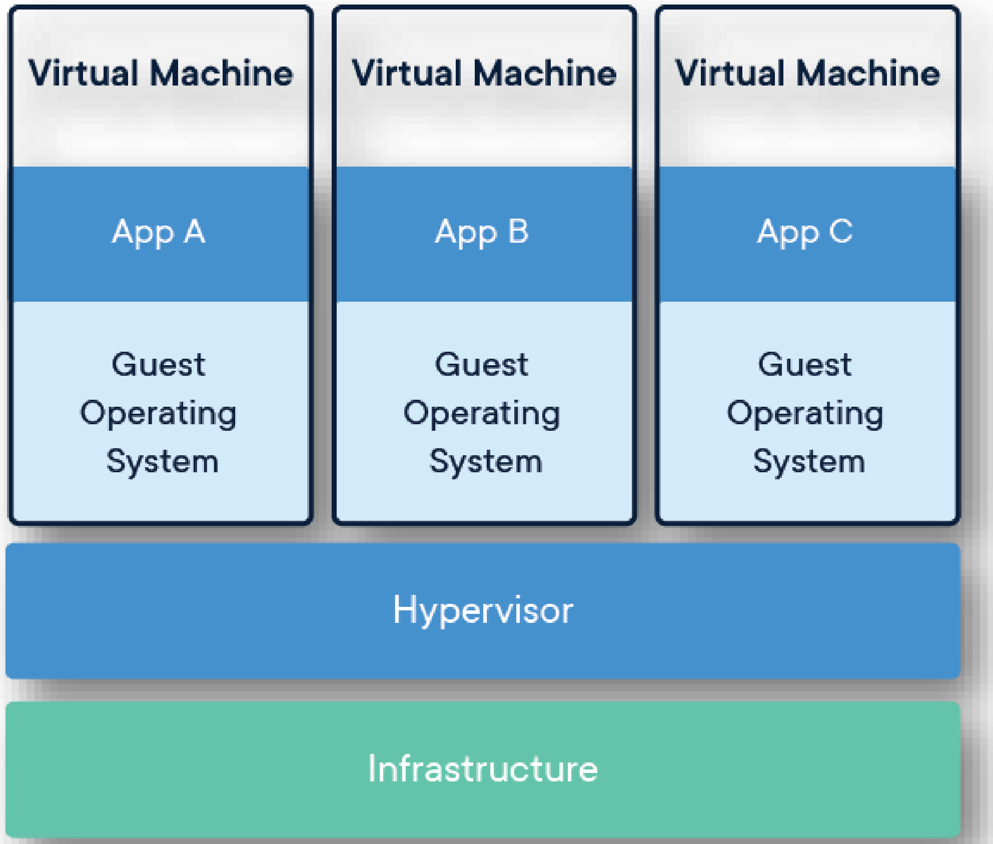
- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

GOOGLE DEFINITION

- Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run.
- This decoupling allows container-based applications to be deployed easily and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer's personal laptop.
- Containerization provides a clean separation of concerns, as developers focus on their application logic and dependencies, while IT operations teams can focus on deployment and management without bothering with application details such as specific software versions and configurations specific to the app.

COMPARING CONTAINERS AND VIRTUAL MACHINES





Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient.

CONTAINERS VS. VMS

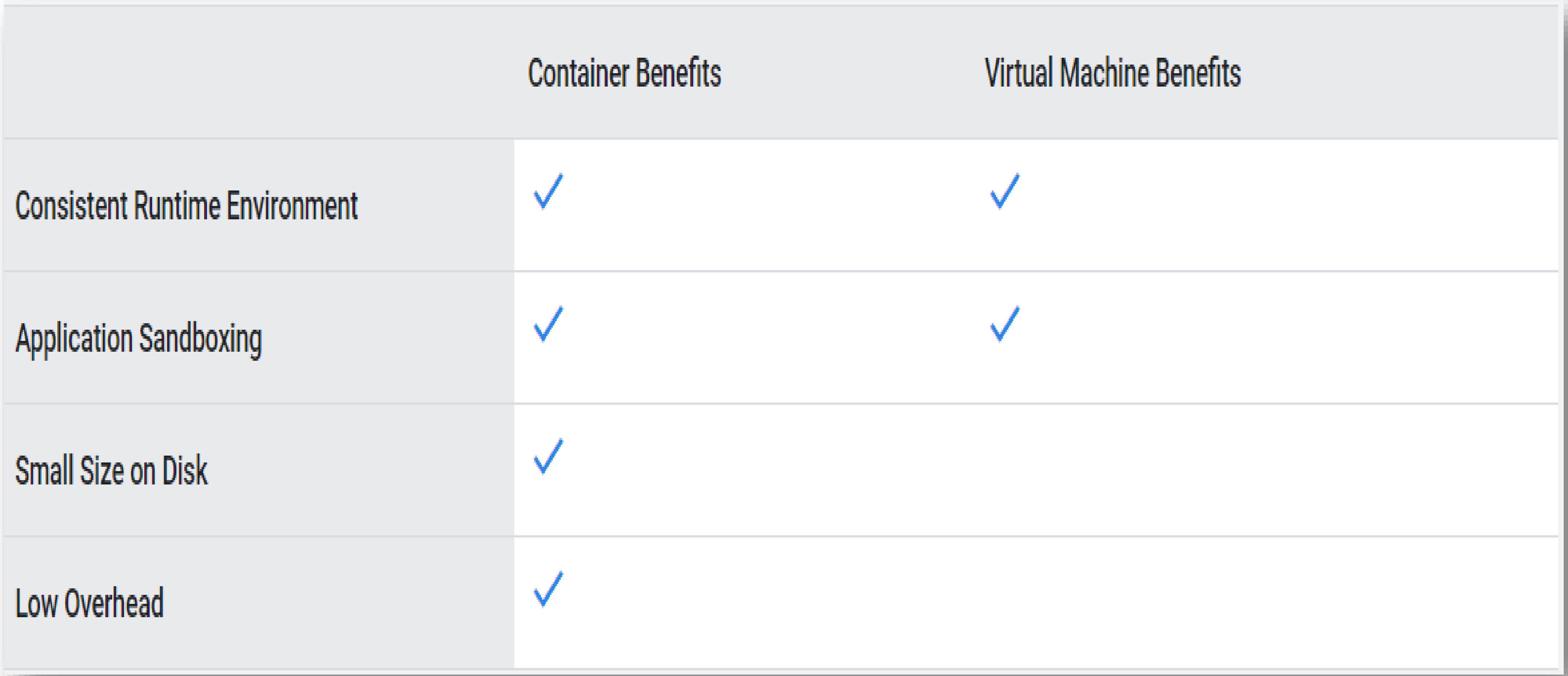
- **VIRTUAL MACHINES**

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

- **CONTAINERS**

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

CONTAINERS VS. VMS



	Container Benefits	Virtual Machine Benefits
Consistent Runtime Environment	✓	✓
Application Sandboxing	✓	✓
Small Size on Disk	✓	
Low Overhead	✓	

TO SEE...

docker