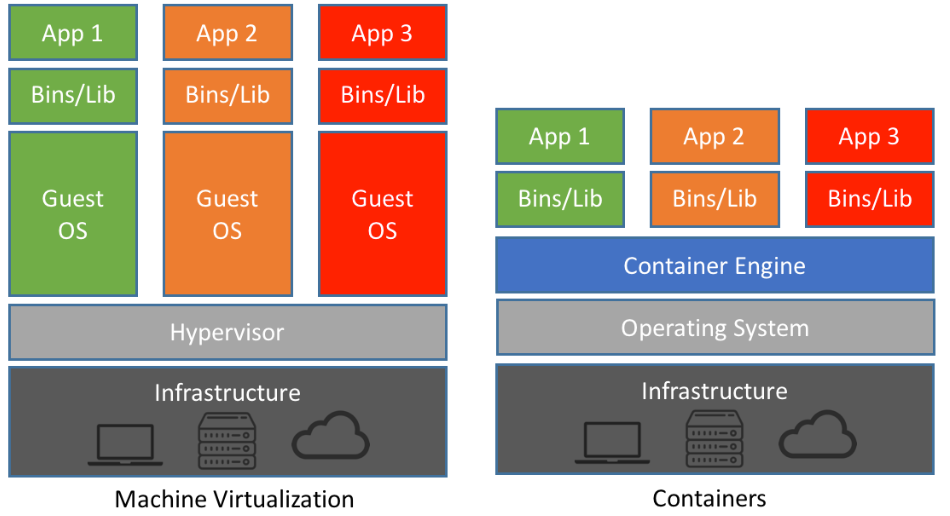
Docker Images are the blue prints, whereas containers are the actual instances or objects running.

What is the diff. between Virtual Machine and Docker ?

On top of the host OS of the hardware machine, the guest OS (Windows, Linux, Mac OS) will be there, on top of which actual applications runs.

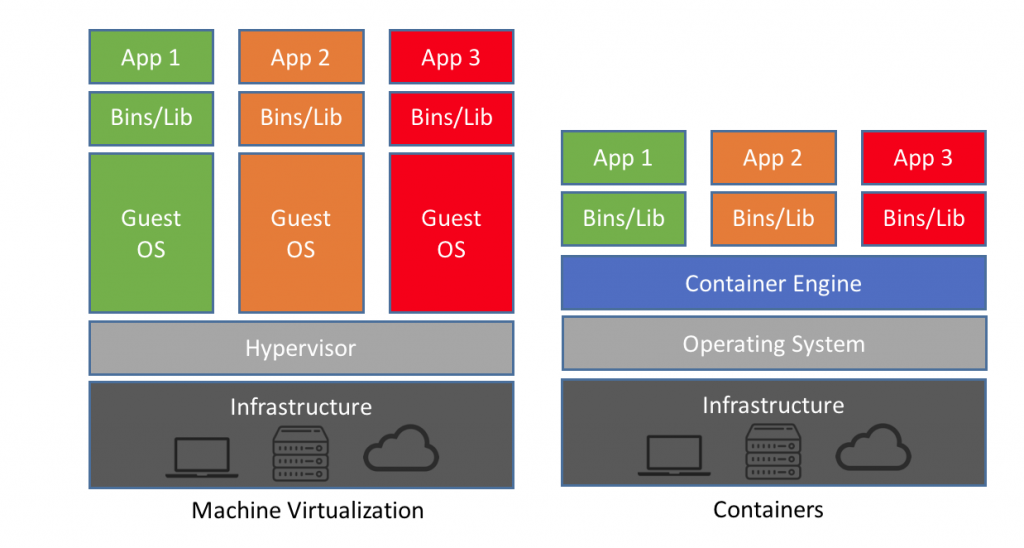


But in case of Docker, the Docker Engine will sit on top of OS of hardware machine. On which the containers (containing all the apps) will sit. The docker engine will not need the complete guest OS required to run the apps (in the containers), instead it just need difference (delta) commands required for that OS from that of hardware machine OS. By this way it is very light in nature.

What Are Virtual Machines (VMs)?

Historically, as server processing power and capacity increased, bare metal applications weren’t able to exploit the new abundance in resources. Thus, VMs were born, designed by running software on top of physical servers to emulate a particular hardware system. A hypervisor, or a virtual machine monitor, is software, firmware, or hardware that creates and runs VMs. It’s what sits between the hardware and the virtual machine and is necessary to virtualize the server.

Within each virtual machine runs a unique guest operating system. VMs with different operating systems can run on the same physical server—a UNIX VM can sit alongside a Linux VM, and so on. Each VM has its own binaries, libraries, and applications that it services, and the VM may be many gigabytes in size.



[Server virtualization](https://www.netapp.com/us/solutions/virtualization/server/index.aspx) provided a variety of benefits, one of the biggest being the ability to consolidate applications onto a single system. Gone were the days of a single application running on a single server. Virtualization ushered in cost savings through reduced footprint, faster server provisioning, and improved [disaster recovery (DR)](https://www.netapp.com/us/solutions/data-protection/disaster-recovery.aspx), because the DR site hardware no longer had to mirror the primary data center.

Development also benefited from this physical consolidation because greater utilization on larger, faster servers freed up subsequently unused servers to be repurposed for QA, development, or lab gear.

But this approach has had its drawbacks. Each VM includes a separate operating system image, which adds overhead in memory and storage footprint. As it turns out, this issue adds complexity to all stages of a software development lifecycle—from [development and test](https://www.netapp.com/us/info/what-is-development-testing-dev-test.aspx) to production and disaster recovery. This approach also severely limits the portability of applications between public clouds, private clouds, and traditional data centers.

What Are Containers?

Operating system (OS) virtualization has grown in popularity over the last decade to enable software to run predictably and well when moved from one server environment to another. But containers provide a way to run these isolated systems on a single server or host OS.

Containers sit on top of a physical server and its host OS—for example, Linux or Windows. Each container shares the host OS kernel and, usually, the binaries and libraries, too. Shared components are read-only. Containers are thus exceptionally “light”—they are only megabytes in size and take just seconds to start, versus gigabytes and minutes for a VM.

Containers also reduce management overhead. Because they share a common operating system, only a single operating system needs care and feeding for bug fixes, patches, and so on. This concept is similar to what we experience with hypervisor hosts: fewer management points but slightly higher fault domain. In short, containers are lighter weight and more portable than VMs.

Conclusion

Virtual machines and containers differ in several ways, but the primary difference is that containers provide a way to virtualize an OS so that multiple workloads can run on a single OS instance. With VMs, the hardware is being virtualized to run multiple OS instances. Containers’ speed, agility, and portability make them yet another tool to help streamline software development.