VMs and Containers: How they are sued to build IaaS Clouds

When it comes to cloud infrastructure, the virtual machine has been the go-to standard for many of its advantages. However, what if you had an alternative to a virtual machine that was more lightweight, economical, and scalable. That's precisely what Docker is.

Docker is a container-based technology that lets you develop distributed applications. In this blog post, I will explain the differences between VM and Docker containers.

What is a Virtual Machine?

A virtual machine is a system which acts exactly like a computer.

In simple terms, it makes it possible to run what appears to be on many separate computers on hardware, that is one computer. Each virtual machine requires its underlying operating system, and then the hardware is virtualized.

What is Docker?

Docker is a tool that uses containers to make creation, deployment, and running of application a lot easier. It binds application and its dependencies inside a container.

What is Docker?

Organizations in today's world look forward to transforming their business digitally but are constrained by the diverse portfolio of applications, cloud, and on-premises-based infrastructure. Docker solves this obstacle of every organization with a container platform that brings traditional applications and microservices built on Windows, Linux, and mainframe into an automated and secure supply chain.

Docker is a software development tool and a virtualization technology that makes it easy to develop, deploy, and manage applications by using containers. Container refers to a lightweight, stand-alone, executable package of a piece of software that contains all the libraries, configuration files, dependencies, and other necessary parts to operate the application.

In other words, applications run the same irrespective of where they are and what machine they are running on because the container provides the environment throughout the software development life cycle of the application. Since containers are isolated, they provide security, thus allowing multiple containers to run simultaneously on the given host. Also, containers are lightweight because they do not require an extra load of a hypervisor.

A hypervisor is a guest operating system like VMWare or VirtualBox, but instead, containers run directly within the host's machine kernel.

Containers provide the following benefits:

- Reduced IT management resources
- Reduced size of snapshots
- Quicker spinning up apps
- Reduced and simplified security updates
- Less code to transfer, migrate, and upload workloads

What are VM?

Virtual machines, on the other hand, are created to perform tasks that, if otherwise performed directly on the host environment, may prove to be risky. Virtual machines are isolated from the rest of the system; the software inside the virtual machine cannot tamper with the host computer. Therefore, implementing tasks such as accessing virus infected data and testing of operating systems are done using VM.

A virtual machine is a computer file or software usually termed as a guest, or an image that is created within a computing environment called the host.

A virtual machine is capable of performing tasks such as running applications and programs like a separate computer making them ideal for testing other operating systems like beta releases, creating operating system backups, and running software and applications. A host can have several VM running at a specific time. Logfile, NVRAM setting file, virtual disk file, and configuration file are some of the key files that make up a virtual machine. Another sector where VM are of great use is server virtualization. In server virtualization, a physical server is divided into multiple isolated and unique servers, thereby allowing each server to run its operating system independently. Each virtual machine provides its virtual hardware, such as CPUs, memory, network interfaces, hard drives, and other devices.

Virtual machines are broadly divided into two categories depending upon their use:

1. System Virtual Machines: A platform that allows multiple VM, each running with its copy of the operating system to share the physical resources of the host system. Hypervisor, which is also a software layer, provides the virtualization technique. The hypervisor executes at the top of the operating system or the hardware alone.
2. Process Virtual Machine: Provides a platform-independent programming environment. The process virtual machine is designed to hide the information of the underlying hardware and operating system and allows the program to execute in the same manner on every given platform.

To learn more about VM, check out Cloud Academy's Virtual Machines Overview Course. If you don't already have a Cloud Academy account, you can sign up for a free 7-day trial.

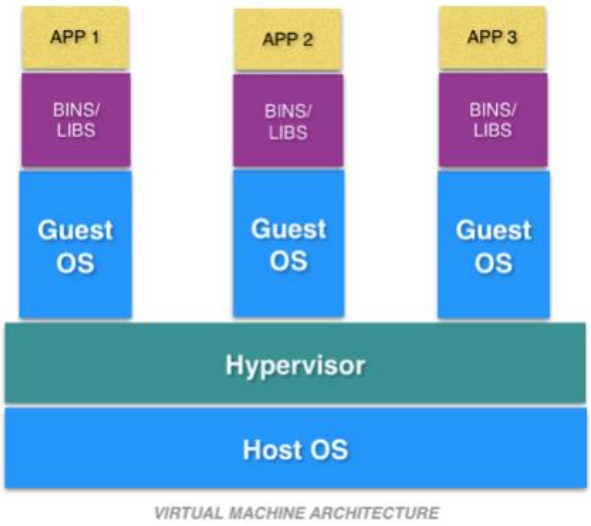
Although several VM running at a time may sound efficient, it leads to unstable performance. As the guest OS would have its kernel, set of libraries and dependencies, this would take up a large chunk of system resources.

Other drawbacks include inefficient hypervisor and long boot uptime. The concept of Containerization overcomes these flaws. Docker is one such containerization platform.

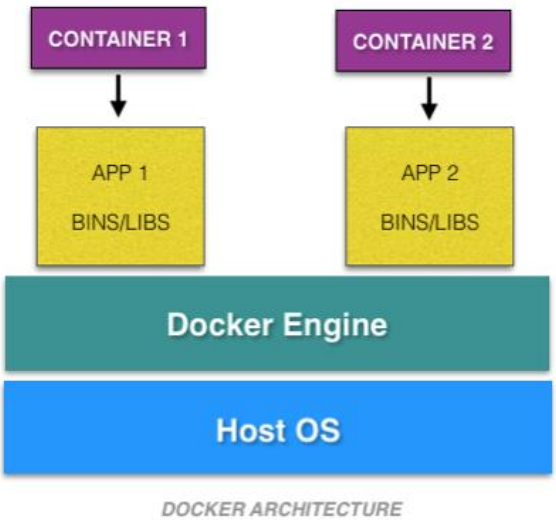
Comparison

1. Docker vs. Virtual Machines: OS Support and Architecture

The main difference lies in their architecture, demonstrated below.



VIRTUAL MACHINE ARCHITECTURE



DOCKER ARCHITECTURE

Virtual machines have host OS and the guest OS inside each VM. Guest OS can be any OS, like Linux or Windows, irrespective of host OS. In contrast, Docker containers host on a single physical server with a host OS, which shares among them. Sharing the host OS between containers makes them light and increases the boot time. Docker containers are considered suitable to run multiple applications over a single OS kernel; whereas, VM are needed if the applications or services required to run on different OS.

2. Docker vs. Virtual Machines: Security

Virtual machines are stand-alone with their kernel and security features. Therefore, applications needing more privileges and security run on VM.

On the flip side, providing root access to applications and running them with administrative premises is not recommended in the case of Docker containers because containers share the host kernel. The container technology has access to the kernel subsystems; as a result, a single infected application is capable of hacking the entire host system.

3. Docker vs. Virtual Machines: Portability

Virtual machines are isolated from their OS, and so, they are not ported across multiple platforms without incurring compatibility issues. At the development level, if an application is to be tested on different platforms, then Docker containers must be considered.

Docker containers packages are self-contained and can run applications in any environment, and since they don't need a guest OS, they can be easily ported across different platforms. Docker containers can be easily deployed in servers since containers being lightweight can be started and stopped in very less time compared to VMs.

4. Docker vs. Virtual Machines: Performance

Virtual machines are more resource-intensive than Docker containers as the VM need to load the entire OS to start. The lightweight architecture of Docker containers is less resource-intensive than VM.

In the case of VM, resources like CPU, memory, and I/O may not be allocated permanently to containers — unlike in the case of containers, where the resource usage with the load or traffic.

Scaling up and duplicating containers in simple and easy as compared to VM because there is no need to install an operating system in them.

Which is a better choice?

It won't be fair to compare Docker and VM since they are intended for different use. Docker, no doubt is gaining momentum these days, but they cannot be said to replace VM. In spite of Docker gaining popularity, a virtual machine is a better choice in certain cases. Virtual machines are considered a suitable choice in a production environment, rather than Docker containers since they run on their own OS without being a threat to the host computer. But if the applications are to be tested then Docker is the choice to go for, as Docker provides different OS platforms for the thorough testing of the software or an application.

Furthermore, Docker containers use docker-engine instead of the hypervisor, like in VM. As the host kernel in not shared, using docker-engine makes containers small, isolated, compatible, high performance-intensive and quickly responsive. Docker containers have comparatively low overhead as they have compatibility to share single kernel and application libraries. Organizations are making use of the hybrid approach mostly as the choice between VM and Docker containers depends upon the kind of workload offered.

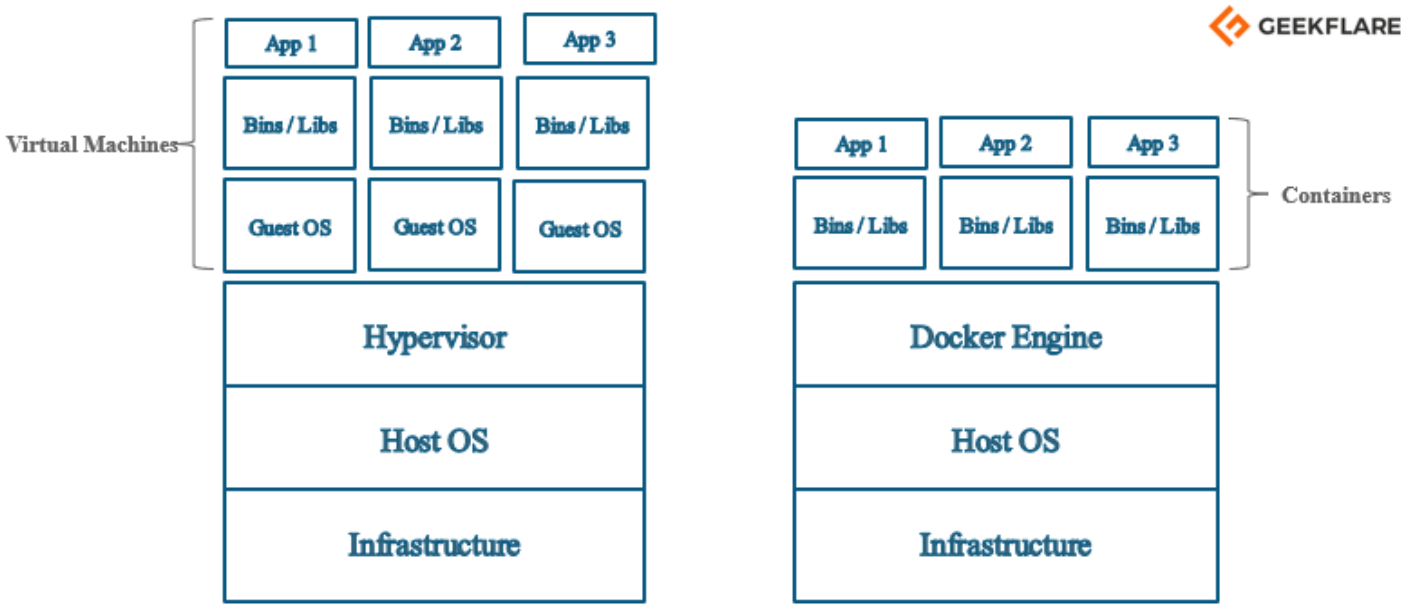
Also, not many digital operational companies rely on VM as their primary choice and prefer migrating towards using containers as the deployment is comparatively lengthy and running microservices is also one of the major challenges it possesses. However, they are still some firms that prefer VM over Dockers whereas companies who are interested in enterprise-grade security for their infrastructure prefer to make use of Dockers.

Finally, containers and Docker are not in conflict with VM, they are both complementary tools for different workload and usage. Virtual machines are built for applications that are usually static and don't change very often. Whereas, the Docker platform is built with a mindset to be more flexible so that containers can be updated easily and frequently.

Docker vs. VM

significant differences are their operating system support, security, portability, and performance.

Operating System Support



The operating system support of Virtual machine and Docker container is very different. From the image above, you can see each virtual machine has its guest operating system above the host operating system, which makes VM heavy. While on the other hand, Docker containers share the host operating system, and that is why they are lightweight.

Sharing the host operating system between the containers make them very light and helps them to boot up in just a few seconds. Hence, the overhead to manage the container system is very low compared to that of VM.

The docker containers are suited for situations where you want to run multiple applications over a single operating system kernel. But if you have applications or servers that need to run on different operating system flavors, then VM are required.

Security

The virtual machine does no share operating system, and there is strong isolation in the host kernel. Hence, they are more secure as compared to Containers. A container have a lot of security risks, and vulnerabilities as the containers have shared host kernel.

Also, since docker resources are shared and not namespaced, an attacker can exploit all the containers in a cluster if he/she gets access to even one container. In a virtual machine, you don’t get direct access to the resources, and hypervisor is there to restrict the usage of resources in a VM.

Portability

Docker containers are easily portable because they do not have separate operating systems. A container can be ported to a different OS, and it can start immediately. On the other hand, VM have separate OS, so porting a virtual machine is difficult as compared to containers, and it also takes a lot of time to port a virtual machine because of its size.

For development purposes where the applications must be developed and tested in different platforms, Docker containers are the ideal choice.

Performance

Comparing Virtual machines and Docker Containers would not be fair because they both are used for different purposes. But the lightweight architecture of docker its less resource-intensive feature makes it a better choice than a virtual machine. As a result, of which containers can startup very fast compared to that of VM, and the resource usage varies depending on the load or traffic in it.

Unlike the case of VM, there is no need to allocate resources permanently to containers. Scaling up and duplicating the containers is also an easy task compared to that of VM, as there is no need to install an operating system in them.

Understanding the difference

VMs and containers represent two different ways to create virtual resources that run applications. With VMs, a special software system -- a hypervisor -- partitions a server below the operating system (OS) level creating true "VM" that share only hardware. With containers, virtualization takes place at the operating system level, so the OS and possibly some middleware are shared.

Functionally, VMs are more flexible, because the "guest" environment where applications run is similar to a bare-metal server. You can pick your own operating system and middleware, independent of what other VMs on the same server might use. With containers, however, you need to accommodate a common OS and middleware elements when you choose applications, since each container uses the core server platform and shares it with other containers.

For enterprises with a variety of software platforms for their applications, containers may be more difficult to use because of the need to standardize on a single hosting platform. Even when everything runs on a single OS, you may need to harmonize everything to use a single version of some or all middleware tools -- which can be difficult to do if software is dependent on a specific version.

On the other hand, containers have less overhead because they don't duplicate the platform software for every application or component that's deployed. This lower overhead makes it possible to run more components per server with container technology. In addition, the deployment and redeployment of applications or components is faster with containers.

Because containers are usually deployed through management platforms like Docker, it's also generally easier to operationalize container-based clouds than VM-based clouds, where management tools are more varied.

Choosing containers vs. VMs for public, private or hybrid cloud

Users can gain all the benefits of containers in private cloud deployments. And for businesses with standardized operating systems and middleware, container-based private clouds are likely the best strategy. However, for public and hybrid clouds, containers are often more problematic and VMs may be the better approach.

For example, one challenge for enterprises adopting containers is that container hosting services in the public cloud are more difficult to find than VM services. While some infrastructure as a service (IaaS) providers, such as Amazon Web Services, offer container services, these services are normally an overlay to the IaaS service, and, in many cases, are only available for customers using a dedicated server or cluster hosting. While any user can deploy VMs via a public IaaS service, it is more complicated with containers, in terms of setup and operations -- particularly because container networking may be difficult to accommodate inside a public cloud.

The difficulty of deploying and managing containers in a public cloud can also make container deployments more complicated in hybrid clouds. , best practices for container deployment an application suggest co-hosting all its components for easy network connection. This, however, makes it more difficult to manage cloud bursting or to failover to public cloud resources -- which are two the most common hybrid cloud use cases.

Second, any differences in middleware or OS at the application level will limit container deployment in the cloud if the cloud container platform isn't compatible. That means hybridization might not work the same across all applications.

Hybrid clouds based on containers are easier to build and maintain if the component distribution in the cloud versus the data center is fairly constant, or if an organization cloudsources those components in a very structured way -- for example, from a specific set data center servers to a specific set cloud servers. This makes the networking and integration the hybrid environment easier to manage and less prone to configuration errors. With VMs, however, it's generally easier to deploy applications and

components into the cloud from the data center using standardized tools and integration practices.

Easing into container technology

It's best to gain familiarity with container technology in private deployments before moving to the public cloud. Knowledge how containers work, and what's needed to maintain them in operations, will help you select the right approach, tools and providers. Container management tools like Docker or Cloud Foundry are essential to make containers work, so try them out in house and decide what's best for you.

In the long term, it's likely that management tools will deploy VM- and container-based clouds. As these tools evolve, the operational differences between container- and VM-based clouds will shrink, and the primary difference will be related to security and compliance. If you're making a choice now, make sure containers offer enough isolation for your cloud applications, since the security and compliance differences between containers vs. VMs is unlikely to shrink over time.

Conclusion

Here is a table which concludes on a virtual machine and Docker container differences.

Virtual Machine	**Docker Container**
Hardware-level process isolation	OS level process isolation
Each VM has a separate OS	Each container can share OS
Boots in minutes	Boots in seconds
VMs are of few GBs	Containers are lightweight (KBs/MBs)
Ready-made VMs are difficult to find	Pre-built docker containers are easily available

VMs can move to new host easily	Containers are destroyed and re-created rather than moving
Creating VM takes a relatively longer time	Containers can be created in seconds
More resource usage	Less resource usage

What are the differences between Docker and VM?

References

https://geekflare.com/docker-vs-virtual-machine/

https://searchcloudcomputing.techtarget.com/tip/Containers-vs-VMs-Which-should-you-use-for-cloud

https://cloudacademy.com/blog/docker-vs-virtual-machines-differences-you-should-know/