Impact of Containers on Data Center Virtualization

M. R. Pamidi, Ph. D.
IT Newswire
and
Anil Vasudeva
IMEX Research

Contents

Executive Summary

The runaway success of virtualization laid the adoption of a new computing architecture that allowed the creation of multiple virtual machines (VMs that consisted of an application along with an instance of an operating system, OS), consolidated infrastructure resources (such as servers, storage, and networks) into a shared pool and, using a hypervisor, abstracted required resources from the pool to deliver them to the VM to meet its specific needs.

On the other hand, Containers, by abstracting applications from VMs to run on a common OS layer, have recently gathered steam since they enable virtualization infrastructure to become even lighter, thus saving CapEx and OpEx in the data center. Docker started the container-virtualization revolution after releasing it under an open-source license in 2013, thus becoming an important target for VMware, the undisputed leader in virtualization. On the heels of Docker's containers gaining market momentum, others like Amazon with its Amazon Web Services (AWS), Google with Google Compute Engine (GCE), IBM, Microsoft with Azure, and Mesosphere quickly rushed in to support Docker's containers.

Because containers within a single operating system are much more efficient, we expect they will become the foundation for the future of cloud infrastructure. The container market is real, has a great future, and has the potential to recast the players in the virtualization marketplace, thus posing a threat to traditional VM vendors.

Scope of this Brief

This Brief provides an early, brief look into the emerging field of Virtualization Containers and their emerging impact on future of Soft-Defined Data Centers (SDDCs).

This Brief delves into answering questions such as:

- What are containers and how do they function?
- How are they different from existing VMs?
- Where do Docker and containers fit in the SDDC stack?
- What are containers' pros and cons vis-à-vis existing virtualized data centers' VM-based architectures?
- How is Docker different from and disrupting traditional containers?

Backgrounder

VMs, by creating an emulation of a physical server complete with an operating system, ushered an era of consolidation wherein servers better utilized the computing power and, as a result, became an ubiquitous part of most enterprise data centers, where Microsoft and VMware rule.

Containers' roots in the UNIX world can be traced back to 1979 when the chroot system call was introduced during the development of Version 7 UNIX, followed by Sun Microsystems introducing Solaris Containers (including Solaris Zones) in the Solaris 10 Operating System in 2005. Linux Containers LXC, an OS-level virtualization feature, was introduced in 2008. By the way, IBM had *containers*, albeit with a different concept, way back in 1975 during the heydays of the mainframe.

A container-based approach, in which applications can run in isolation without relying on a separate OS per application, can save huge amounts of hardware resources. In effect, containers act as the new generation of *OS Hypervisor*. Containers will also open the door to an additional level of virtualization to maximize consolidation. They even have the potential to provide a transformative catalyst to the adoption of open-source software in SDDCs. Figure 1 illustrates the evolution of virtualization.

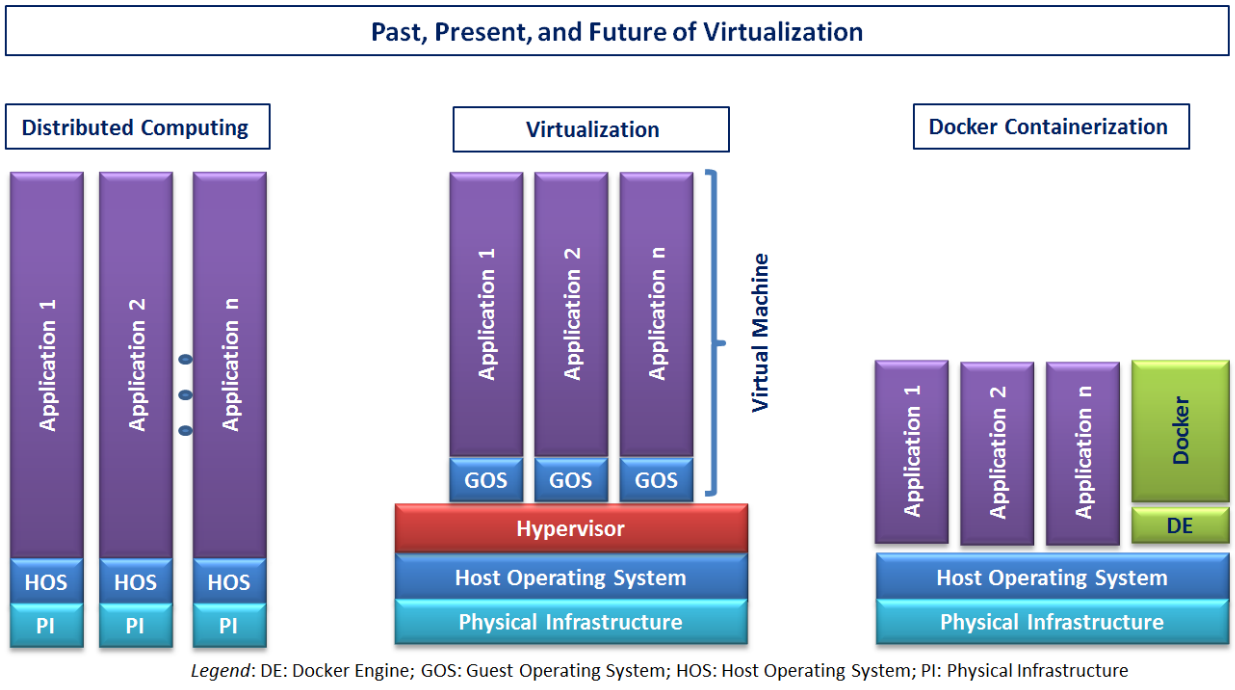


Figure 1. Past, Present, and Future of Virtualization Architecture

Containers vis-à-vis VMs

Traditional VMs carry a heavy payload. Each VM includes the application, which may be a just a few MBs, the required binaries and libraries, plus an entire OS which may run into tens of GBs.

General limitations of VM include:

1. Running an entire OS drains resources.
2. Waiting for the OS to boot slows down startup time.
3. The OS often consumes more memory and more disk than the actual application it hosts.

There is also a general misconception that OS virtualization is the only path to provide application isolation for running on a server. These assumptions are being proven wrong, thanks to containers that can now be used as an alternative to OS-level virtualization to run multiple isolated systems on a single host.

Traditional cloud infrastructure providers, such as Amazon Web Services and Google Compute Engine, peddle VMs. VMs provide the ability to scale up or down, almost-guaranteed computational resources, security isolation, and access to APIs for provisioning, without any of the overhead associated with managing physical servers. However, the more VMs you deploy the more overhead you are paying for running a full-blown OS image for each VM. This approach has become an unnecessarily expensive (despite continually downward-spiraling pricing by cloud vendors), heavyweight solution to the underlying question of how best to run applications in the cloud. Table 1 provides a comparison of Containers and VMs.

Containers	Virtual Machines
Not standardized, very OS- and kernel-specific; even on the same kernel and OS, options could range from security sandboxes for individual processes to nearly complete systems.	Fairly standardized; a system image running on one expects mostly the same things as if it had its own bare-metal computer.
Have to run the same kernel as the host, but they can optionally run a completely different package tree or distribution.	Don't have to run the same kernel or OS as the host and are opaque to the host system.
Are lightweight; the host system usually starts the containerized application directly or runs a container-friendly init daemon, like systemd.	Are heavyweight and require a full OS and system image.

Table 1. Comparison Summary of Containers and VMs

The Benefits of Containers

A container is a lightweight virtualization mechanism that does not require you to set up a VM on an emulation of physical hardware. One of the main advantages of containers is that they allow you to run a complete copy of the OS in a container without the overhead of running a level-2 hypervisor. Containerization lets you simplify data center automation without having to rewrite the applications.

Additionally, by bridging a container's Virtual Ethernet interface to the host's network interface, each container can appear to have its own IP address. Letting a container/application have its own virtual IP address allows isolated environments for untrusted users, a key to requirements for success in multi-tenant public cloud environments.

Besides the fact that containers use far less system resources than entire VMs on a virtualization platform, the following advantages of containers over VMs stand out:

1. **Better performance**: Containers make scaling up much more affordable and granular.
2. **Fast provisioning**: Containers are provisioned via software into an existing infrastructure. Containers can be added, removed, and redistributed in seconds.
3. **High availability**: Containers can run on different underlying hardware. So if one host goes down, traffic can routed from the edge to live application containers running elsewhere.
4. **Scaling**: Containers can handle sites from hundreds to millions of page-views without any architectural changes without incurring any downtime. VM-centric hosting architectures can't do this because vertical scalability requires reboots to resize, and architectural gaps limit horizontal scaling.

Container Technology

Why Containers are Lightweight

One of the major reasons for the success of Docker emanates from the fact that Docker containers are lightweight compared to VMs, using copy-on-write differential techniques which significantly lower the CapEx to create a containerized virtualization infrastructure (Figures 2 and 3). Additionally, being in the open source community has allowed Docker to garner a large following in a matter of one year, with ecosystems ranging from support by many OS vendors, several vendors creating development tools, configuration management software, and service discovery and repositories, and now taken up by over 10 systems integrators, and a host of vendors, including Amazon, Baidu, Google, IBM, Microsoft, Rackspace, and many startups.

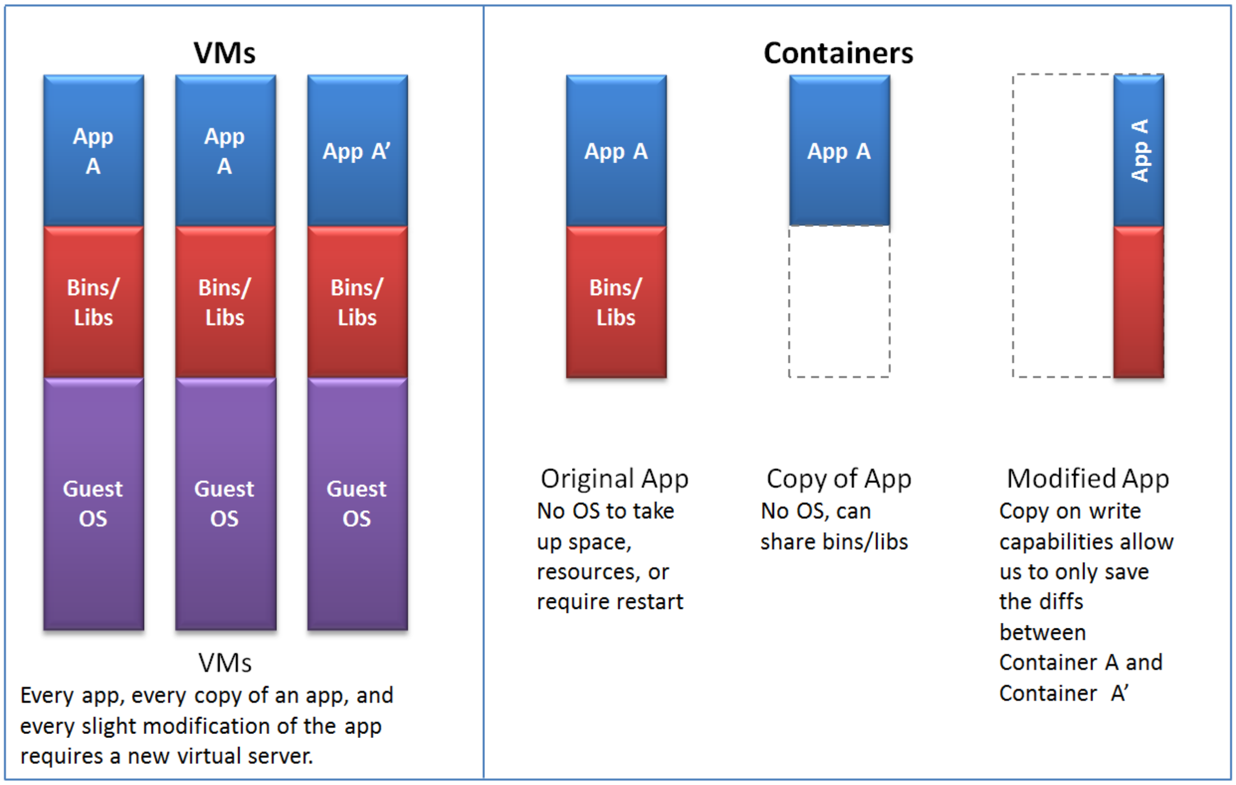
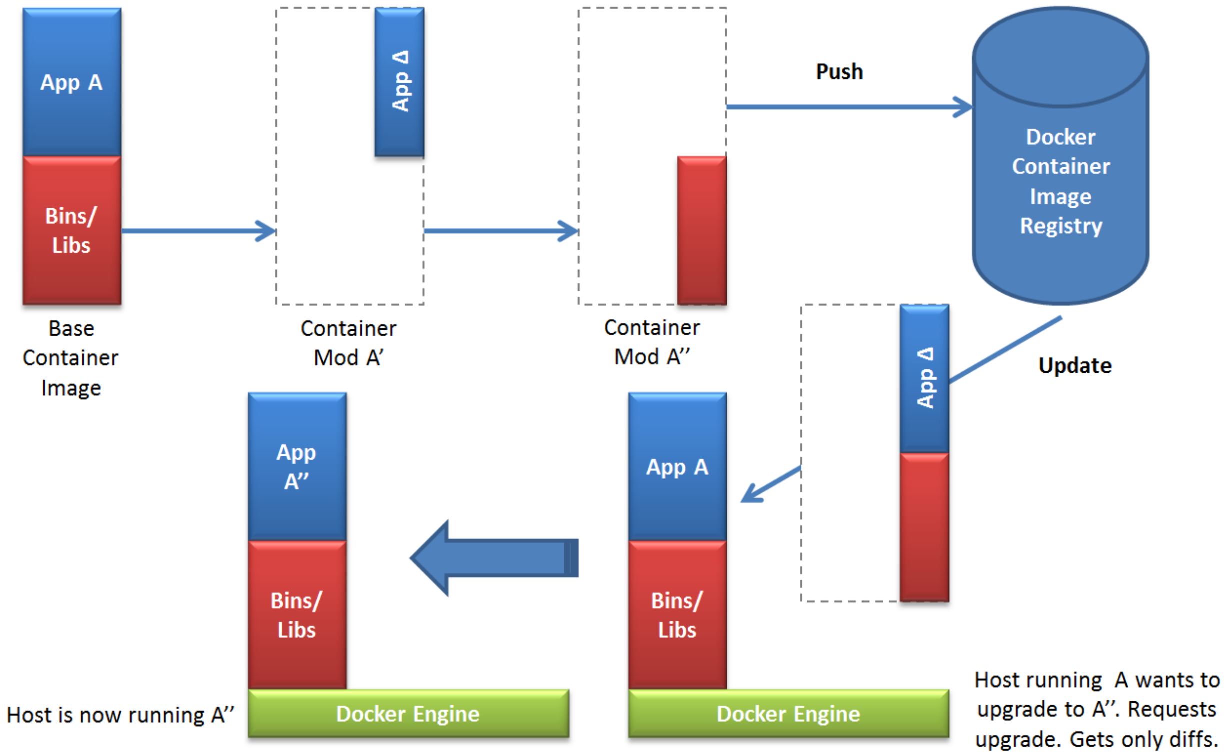

VMs
App A
App A
App A'
Bins/ Libs
Bins/ Libs
Bins/ Libs
Guest OS
Guest OS
Guest OS
VMs
Every app, every copy of an app, and every slight modification of the app requires a new virtual server.
Containers
App A
Bins/ Libs
App A
App Δ
Original App
No OS to take up space, resources, or require restart
Copy of App
No OS, can share bins/libs
Modified App
Copy on write capabilities allow us to only save the diffs between Container A and Container A'

Figure 2. Containers provide a lightweight architecture compared to VM (*Source*: Docker)



Changes and Updates in Docker Container System

Figure 3. Docker Container's Automated Workflow (*Source*: Docker)

Competition
Containers aren't going to completely replace traditional VMs and server virtualization in data centers anytime soon, regardless of how popular they might become, since there are a few barriers and technical limitations to overcome for containers—such as containers' inability to provide a virtual instance of Windows on a Linux server—before they have an expanded universal adoption.

However, cloud providers will likely be the driving force behind a resurgence of containers since a lightweight container approach can be very appealing to cloud providers constantly looking for efficiency improvements which could make a difference in their ability to provide aggressively competitive pricing to customers.

Docker
Containerization lets you launch multiple applications that share the same OS kernel and other system resources but otherwise act as though they're running on separate machines. Each is sandboxed off from the others so that they can't interfere with each other. Docker, announced in 2013, offers an easy way to package, distribute, deploy, and manage containerized applications.

Docker consists of *Docker Engine*, a portable, lightweight runtime and packaging tool, and *Docker Hub*, a cloud service for sharing applications and automating workflows. Docker enables apps to be quickly assembled from components, eliminating the friction among development, QA, and production environments. IT can thus ship lot faster and run the same app, unchanged, on data center VMs, laptops, and any cloud. In other words, Docker is a container that packages applications into hardware-isolated containers, and brings fresh life into a lightweight and portable approach to virtualization architecture. Thus, it enjoys the resource isolation and allocation benefits of VMs, but is much more portable and efficient.

Docker announced at the DockerCon Europe 2014 three new Docker orchestration services: Docker Machine, Docker Swarm, and Docker Compose. Each one covers a different aspect of the lifecycle for distributed apps and is implemented with a "batteries included, but removable" approach which means they may be swapped out for alternative implementations from ecosystem partners designed for particular use cases.

Docker Ecosystem: Honeymoon by Vendors
Docker's ability in helping developers quickly get their applications from concept to production is attracting several leading virtualization vendors and cloud service providers to the Docker camp:

- **Amazon** announced adding Docker support to AWS Beanstalk that lets users create and manage Docker containers.
- **IBM** announced a strategic partnership with Docker to sell integrated solutions that include Docker Hub Enterprise. IBM also announced the beta of Docker-based IBM Containers service that will include open Docker-native features and interfaces, including the new Docker orchestration services. IBM will bundle Docker Hub Enterprise with these integrated services as well as sell it as a standalone product offering.
- **Microsoft** announced at the DockerCon event in June 2014 that it will support Docker containers in its Azure cloud on Linux VMs. Longer term, Microsoft foresees Linux and Windows coexisting peacefully with Docker's help (Figure 4). In the next wave of Windows Server, "Dockerized Windows apps will run on Windows hosts and Dockerized Linux apps will still run on Linux hosts," according to Scott Johnston, Docker's Senior VP.
- **Oracle** supports Docker packages for Oracle Linux, but it is not certified to run Oracle applications. Since Solaris has had container technology for a while, Oracle applications are currently certified to run in Solaris Containers.
- **Red Hat,** realizing that Docker is the container technology with the most momentum, has endorsed and supports Docker for its OpenShift PaaS. It has begun to invest heavily in a version of its Linux operating system, known as Red Hat Enterprise Linux Atomic that is specifically aimed at supporting containers. Building Docker support into OpenShift enables Red Hat to leverage the energy behind Docker as it battles VMware/Pivotal's Cloud Foundry for the hearts and minds of the developer community and

progressive enterprises looking to choose between the two open source PaaS architectures. It recently launched Linux Container Beta with Docker and Google Kubernetes support.

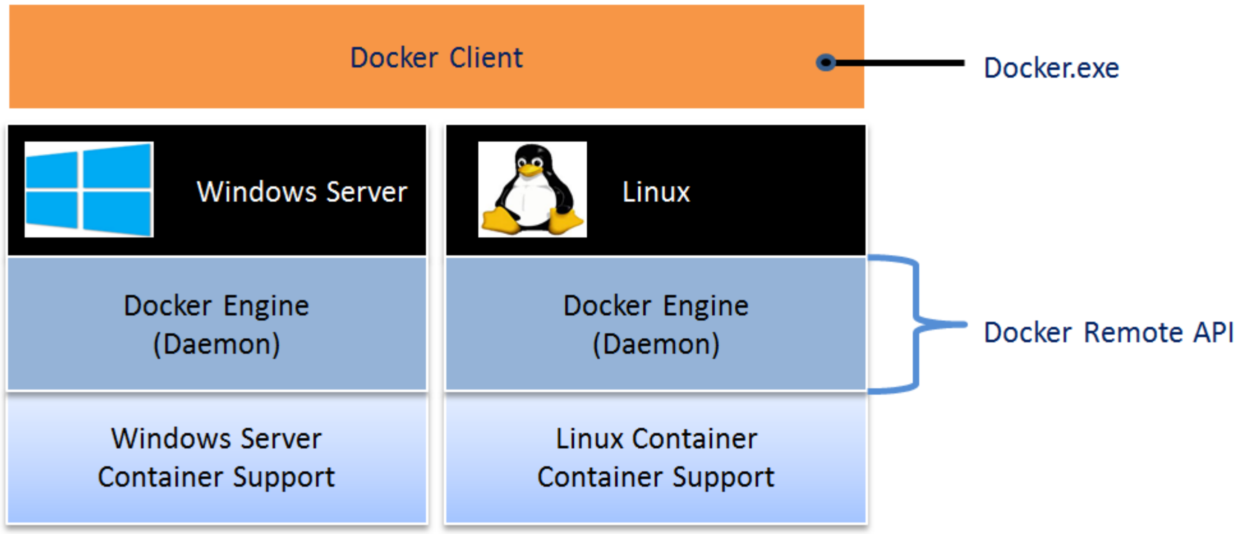


Figure 4. Docker's tools will let you manage Linux and Windows containers interchangeably.

- **VMware**, although has an option for containers in Pivotal's (a spinoff from VMware and EMC) open-source Cloud Foundry project, had to quickly embrace containers, as Docker started to rise to become an important threat to the dominance of VMware in virtualization. So VMware, at VMworld in August 2014, announced support for Cloud Foundry Warden, Docker, and Google containers. But one burning issue is the container host itself that relies on an OS, such as Linux for Docker. VMware neither owns an OS nor provides a container host, leaving a gaping hole in VMware's container strategy, probably to be filled by VMware's competitors. Working with Docker, Google, and Pivotal, VMware plans to integrate Docker containers on its virtualization platform. This could even result in VMware releasing its own containers, compatible on its platform with existing VM architecture.

Future of Containers

Despite the promises of containers and Docker, VMs and virtualization are here to stay in the foreseeable future because containerization has risks.

1. One of the benefits of container-based virtualization is its ability to squeeze more and more things onto a single piece of hardware for cost savings. But a shared back-end failure could result in widespread and catastrophic outages.
2. There's very little cost-driven incentive to get rid of the VM operating system in place now. Microsoft essentially gives away Windows when you buy their data center products.
3. Until Microsoft ships its Docker-based in the future, containers cannot provide a virtual instance of Windows on a Linux server.

Containers may not immediately replace VMs anytime soon but will co-exist peacefully with VMs and play major roles in architecting the emerging SDDCs. VMs will not go away, but reliance on the VM-only approach may go away.