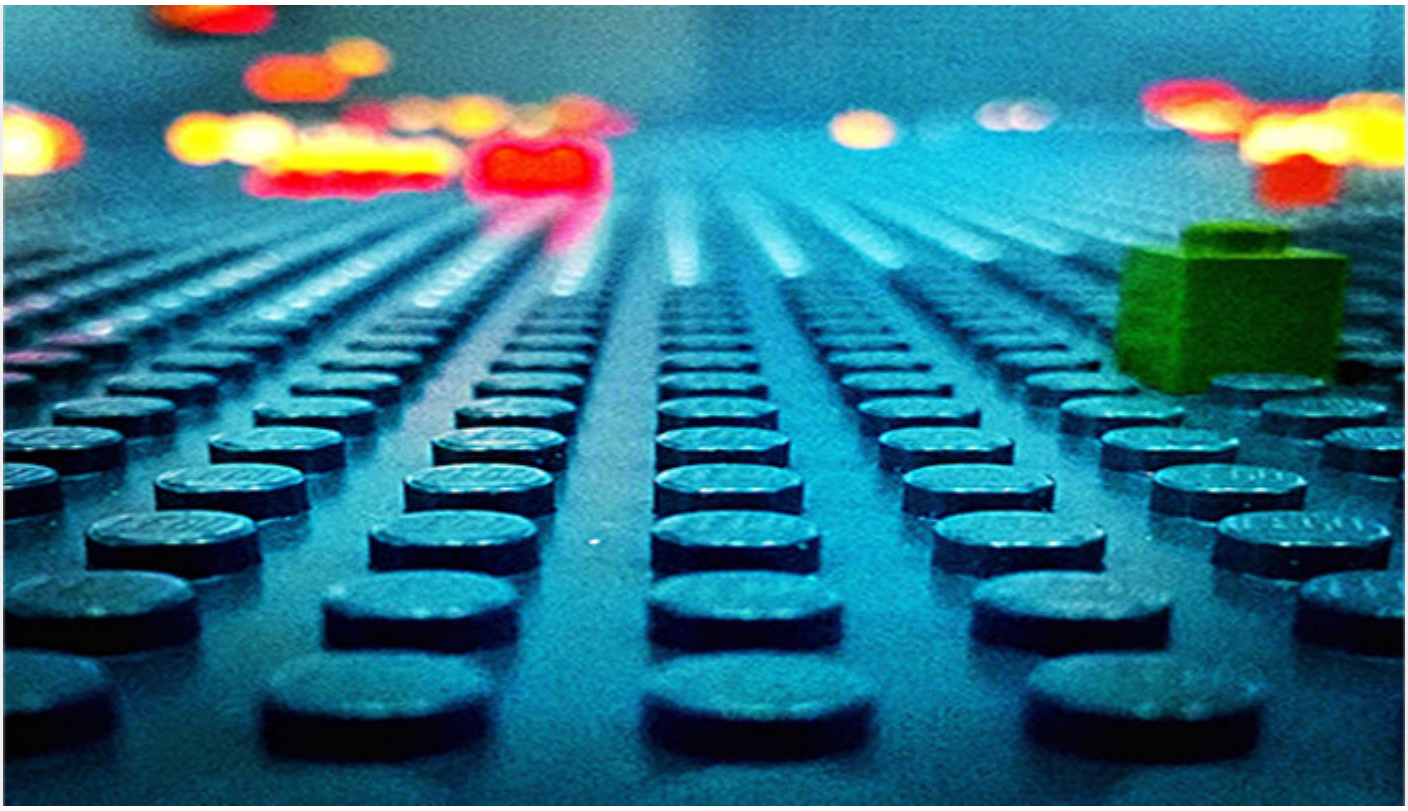
VIRTUAL MACHINES (VMS) VS CONTAINERS: WHAT'S THE DIFFERENCE?



Virtualization transformed networked computing in the early 1970s, paving the way for unprecedented paradigm shifts—like the cloud computing revolution of the last decade.

More recently, the container revolution emerged with a similar goal: improvements in data center technologies and application development. The technology isn't entirely new. Linux container solutions including LXC and Solaris Zones have been in the industry for over a decade. Enterprises like Google have been using their own container technologies for several years. (Google reportedly starts 2 billion containers weekly!)

However, it wasn't until Docker launched, in 2013, with its developer-friendly container solutions and ecosystem, that the technology truly gained traction in enterprise IT. In fact, we're potentially heading toward an era where traditional virtualization methodologies will give way entirely to containerization.

Before you embrace container-based development solutions for app development and software release processes, make sure you fully understand the concepts and operations of virtual machines and containers. By the end of this article, you'll:

- Understand the concepts and differences of VMs and containers
- Consider the drawbacks of both
- Explore key trends

- Get tips for getting started with containers

What is a virtual machine?

A **virtual machine (VM)** is best described as a software program that emulates the functionality of a physical hardware or computing system. It runs on top of an emulating software, called the **hypervisor**, which replicates the functionality of the underlying physical hardware resources with a software environment. These resources may be referred to as the **host machine**, while the VM that runs on the hypervisor is often called a **guest machine**.

The virtual machine contains all necessary elements to run the apps, including:

- Computing
- Storage
- Memory
- Networking
- Hardware functionality available as a virtualized system

The VM may also contain the necessary system binaries and libraries to run the apps. The actual operating system (OS), however, is managed and executed using the hypervisor.

How virtual machines work

The virtualized hardware resources are pooled together and made available to the apps running on the VM. Then, an abstraction layer is created to decouple the apps from the underlying physical infrastructure. This means the physical hardware can be changed, upgraded, or scaled without disrupting the app performance.

A VM will operate as an isolated PC and the underlying hardware can operate multiple independent, isolated VMs for different workloads. VMs operations are typically resource-intensive and do not allow individual app functionality to run in isolated PC-like virtualized environments unless a separate VM is used for different modular elements of the app. If an app workload needs to migrate between different virtual machines or physical data center locations, the entire OS needs to migrate along with it.

Rarely does a workload operation consume all the resources made available to the associated VM. As a result, the remaining unused resources many not be used incorporated in capacity planning and distribution across all VMs and workloads. This leads to a big drawback of VMs: inaccurate planning and significant resource wastage—even though virtualization was developed specifically to optimize the usage and distribution of hardware resources within a data center.

Modern apps and IT services are developed in several modular chunks in order to facilitate:

- Faster development and release
- High scalability
- The flexibility to evolve application development in response to changing business and market needs

Monolithic app development practices are losing popularity and organizations are pursuing infrastructure architecture solutions to further optimize hardware utilization. This is precisely why

containerization was invented and gained popularity as a viable alternative.

What is a container?

Containerization creates abstraction at an OS level that allows individual, modular, and distinct functionality of the app to run independently. As a result, several isolated workloads—the containers—can dynamically operate using the same physical resources.

A less technical definition of containers might be: a unit of software that is lightweight but still bundles the code, its dependencies, and the configuration altogether into a single image. Containers can run:

- On top bare metal servers
- On top hypervisors
- In cloud infrastructure

Containers share all necessary capabilities with the VM to operate as an isolated OS environment for a modular app functionality with one key difference. Using a containerization engine, such as the Docker Engine, containers create several isolated OS environments within the same host system kernel, which can be shared with other containers dedicated to run different functions of the app. Only bins, libraries, and other runtime components are developed or executed separately for each container, which makes them more resource efficient as compared to VMs.

Benefits of containers

Containers are particularly useful in developing, deploying, and testing modern distributed apps and microservices that can operate in isolated execution environments on same host machines.

With containerization, developers don't need to write application code into different VMs operating different app components to retrieve compute, storage, and networking resources. A complete application component can be executed in its entirety within its isolated environment without affecting other app components or software. Conflicts within libraries or app components do not occur during execution and the application container can move between the cloud or data center instances efficiently.

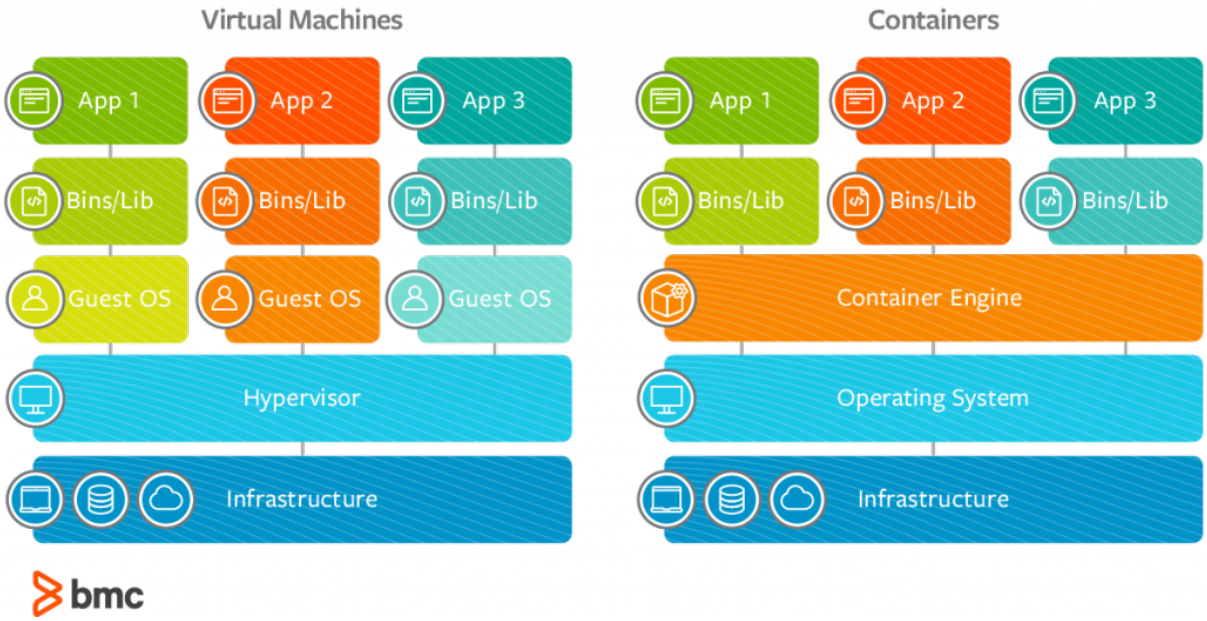
The security problem

Of course, it isn't all rosy: perhaps the biggest drawback of containers is their lack of security. In 2018, 60% of organizations suffered a security incident specific to containers, according to Tripwire. There's the usual security vulnerabilities—bugs, poor authentication and authorization, and even misconfiguration. That means the potential attack surface is large.

But it's not so easy to bring security into the entire stack, lifecycle, and pipeline. That explains why nearly half of the organizations in this survey admitted deploying containers that had known vulnerabilities and/or without testing for vulnerabilities.

Still, as containers continue growing exponentially, more teams are involving DevOps and SecOps from the get-go, as we'll see in trends, below.

Architecture: Containers vs virtual machines



A visualization of the architectural difference between VMs and containers

Key value propositions

The architectural difference offers the following key value propositions for IT personnel and businesses:

- **Continuous Integration, Deployment, and Testing**. In DevOps-driven organizations, organizations can leverage containers to facilitate processes in the CI/CD pipeline. Containers operate as consistent infrastructure environment such that developers don't need to perform complex configuration tasks for every SDLC sprint as workloads migrate across the physical resources.
- **Workload Portability**. IT workloads can switch between different infrastructure instances and virtual environments without significant configuration changes or rework on the application code.
- **Software Quality and Compliance**. Transparent collaboration between devs and testing personnel in delivering operating chunks of the application leads to better software quality, faster development cycles, and improved compliance.
- **Cost Optimization**. Containers maximize resource utilization within their own isolated virtualized environments. This allows organizations to accurately plan for infrastructure capacity and consumption.
- **Infrastructure Agnostic**. Containers make the app components infrastructure agnostic, allowing organizations to move workloads between bare metal servers to virtualized environments to cloud infrastructure in response to changing business needs.

Container trends

These value propositions justify the growing interest and spending containerization technologies. And it's happening fast! So fast that by 2023, Gartner estimates, more than two-thirds of global organizations will be running 2+ containerized applications. Contrast that to last year: in 2019, that number was under 20%.

In the recent State of Container and Kubernetes Security Report, from early 2020, Stackrox surveyed more than 500 tech professionals on container and adoption trends. Here are some key takeaways:

- **Containerized apps are surging.** Companies that have more than half their apps containerized jumped from 23% to 29% in only six months. That's a 29% growth rate.
- **Containers bring in DevOps and Security.** A huge concern with containers is security, so mature container users are looping in DevOps and Security way earlier in the process—a big win for DevSecOps
- **AWS is still king.** But who takes #2 for cloud deployments is an ongoing battle: Azure remains in a precarious second place, as GCP has upped its third-place position by seven percentage points.
- **Azure users have the least container maturity.** Only 20% of Azure users have containerized half or more of their apps—compared to 33% of all non-Azure folks.

How to implement containers in your organization

Companies need not shy away from adding containers to your tooling. Want to take advantage? Here are some ways to start working with containers:

- **Look at your current environment.** Analyze your apps, review what environments are best for them. Will you refactor your apps or build them wholly new? If refactoring is interesting to you, start with an app that's standalone and doesn't require interacting with other apps to run.
- **Experiment in Kubernetes.** There's a good chance your developers are already using K8s. Learn more in our multi-part Kubernetes Guide.
- **Embrace the promoters.** Someone (or many someones) in your company likely has a thought or two on a containerization strategy. Talk to them, learn from them, and consider making them the face of this effort.
- **Designate a project.** Identify a small project to start on, name the team, and outline what you hope to accomplish. Measure the changes down the road.
- **Encourage education.** New technologies take time to learn well. Consider investing in Kubernetes certifications or, at the very least, giving your devs some dedicated study time.

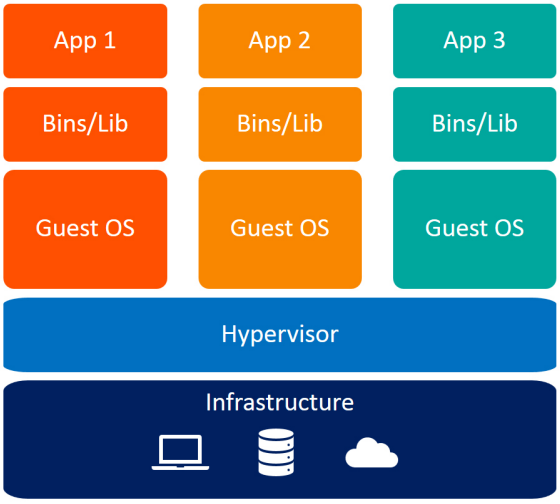
For DevOps-driven organizations that focus on faster and continuous release cycles of distributed, microservices-based app functions, containerization will continue to attract investments, especially in areas where virtualization failed to deliver.

Additional resources

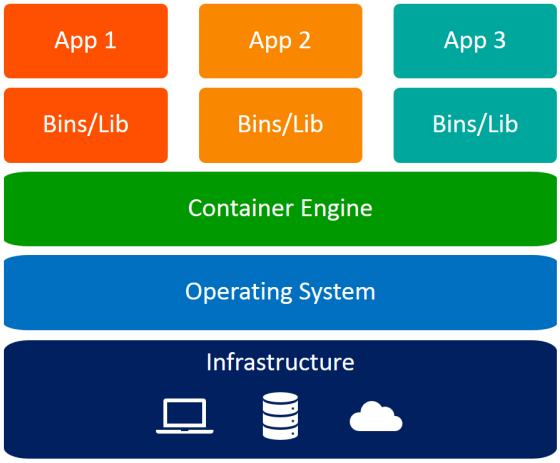
For more on this topic, explore these resources:

- BMC DevOps Blog
- BMC Multi-Cloud Blog
- What is a Virtual Network?
- The Role of Virtualization in DevOps, part of our multi-part DevOps Guide
- Container Management Platforms: Which Are Most Popular?
- 3 Steps To Introduce Docker Containers in The Enterprise
- Containers Aren't Always the Solution

Original reference image:



Virtual Machines



Containers