Blog

Containers vs. Virtual Machines (VMs): What's the Difference?

Cloud Compute

Site feedback

9 April 2021 3 min read

By:
IBM Cloud Team, IBM Cloud

Let's talk

Everything you need to know to understand the containers vs. VMs debate and why containers are growing in popularity.

If you’ve only recently started learning about virtualization tools, you might wonder what the differences are in the technology conversation around containers vs. virtual machines (VMs). Containers have become a dominant force in cloud native development, so it’s important to understand what they are and what they are not. While containers and VMs have distinct and unique characteristics, they are similar in that they both improve IT efficiency, provide application portability and enhance DevOps and the software development lifecycle.

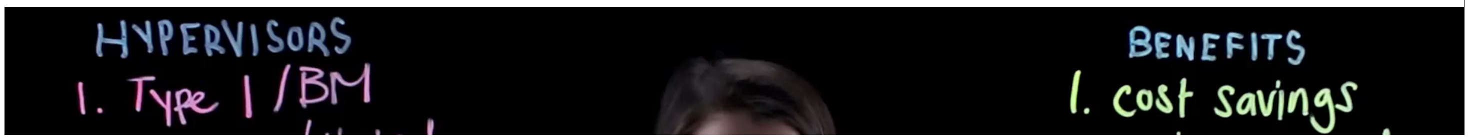
How virtualization works

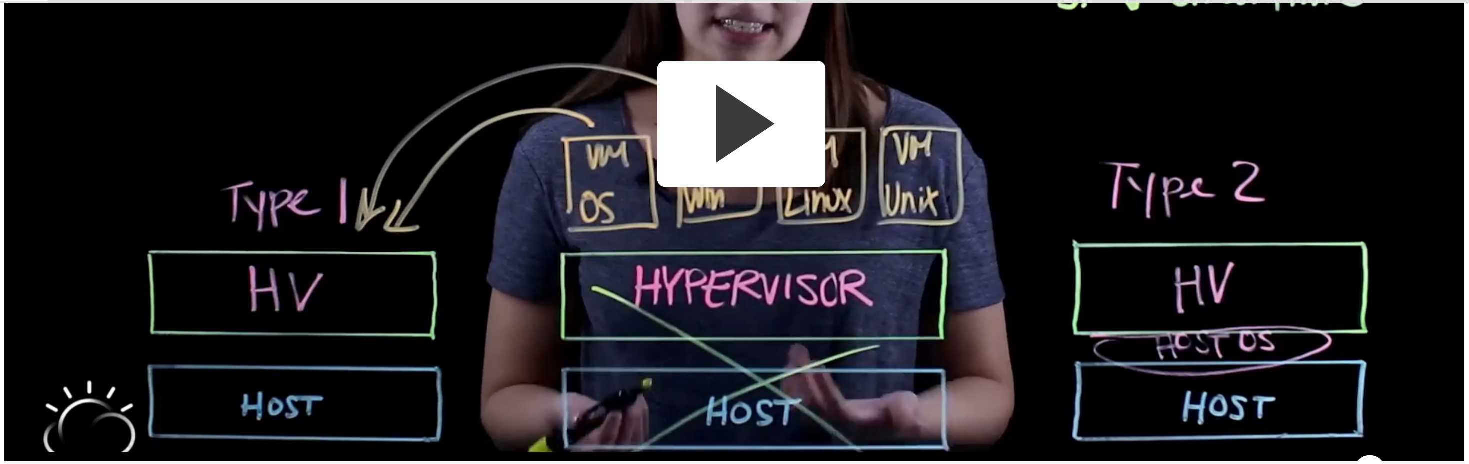
Virtualization is a process whereby software is used to create an abstraction layer over computer hardware that allows the hardware elements of a single computer to be divided into multiple virtual computers.

The software used is called a hypervisor — a small layer that enables multiple operating systems to run alongside each other, sharing the same physical computing resources. When a hypervisor is used on a physical computer or server (also known as bare metal server) in a data center, it allows the physical computer to separate its operating system and applications from its hardware. Then, it can divide itself into several independent “virtual machines.”

The following video provides a closer look at virtualization technology:

Virtualization Explained





5:00 / 5:19

What are virtual machines?

Virtual machines (VMs) are a technology for building virtualized computing environments. They have been around for quite a while and are considered the foundation of the first generation of cloud computing.

Simply put, a virtual machine is an emulation of a physical computer. VMs enable teams to run what appear to be multiple machines, with multiple operating systems, on a single computer. VMs interact with physical computers by using lightweight software layers called hypervisors. Hypervisors can separate VMs from one another and allocate processors, memory, and storage among them.

VMs are also known as virtual servers, virtual server instances and virtual private servers.

What are containers?

Containers are a lighter-weight, more agile way of handling virtualization — since they don't use a hypervisor, you can enjoy faster resource provisioning and speedier availability of new applications.

Rather than spinning up an entire virtual machine, containerization packages together everything needed to run a single application or microservice (along with runtime libraries they need to run). The container includes all the code, its dependencies and even the operating system itself. This enables applications to run almost anywhere — a desktop computer, a traditional IT infrastructure or the cloud.

Containers use a form of operating system (OS) virtualization. Put simply, they leverage features of the host operating system to isolate processes and control the processes' access to CPUs, memory and desk space.

Containers have been around for decades, but the common consensus is that the modern container era began in 2013 with the introduction of Docker, an open source platform for building, deploying and managing containerized applications. Learn more about Docker, Docker containers, Dockerfiles (the container image's build file) and how the ecosystem has evolved with container technology over the last decade.

Containers vs. VMs: What are the differences?

In traditional virtualization, a hypervisor virtualizes physical hardware. The result is that each virtual machine contains a guest OS, a virtual copy of the hardware that the OS requires to run and an application and its associated libraries and dependencies. VMs with different operating systems can be run on the same physical server. For example, a VMware VM can run next to a Linux VM, which runs

Instead of virtualizing the underlying hardware, containers virtualize the operating system (typically Linux or Windows) so each individual container contains *only* the application and its libraries and dependencies. Containers are small, fast, and portable because, unlike a virtual machine, containers do not need to include a guest OS in every instance and can, instead, simply leverage the features and resources of the host OS.

Just like virtual machines, containers allow developers to improve CPU and memory utilization of physical machines. Containers go even further, however, because they also enable microservice architectures, where application components can be deployed and scaled more granularly. This is an attractive alternative to having to scale up an entire monolithic application because a single component is struggling with load.

In the following video, Nigel Brown takes a closer look at how containers differ from VMs:

Containers vs VMs: What's the difference?

7:58 / 7:58

Why containers?

While there are still many reasons to use VMs, containers provide a level of flexibility and portability that is perfect for the multicloud world. When developers create new applications, they might not know all of the places it will need to be deployed. Today, an organization might run the application on its private cloud, but tomorrow it might need to deploy it on a public cloud from a different provider. Containerizing applications provides teams the flexibility they need to handle the many software environments of modern IT.

Containers are also ideal for automation and DevOps pipelines, including continuous integration and continuous deployment (CI/CD) implementation.

Site feedback

Managing containers for multicloud

Let’s talk

Despite the many benefits of containers and the myriad use cases where they are the best option, they do come with a few challenges of their own. Large enterprise applications can include a massive number of containers, and container management presents some serious issues for teams. How can you have visibility on what is running and where? How do you handle crucial issues such as security and compliance? How do you consistently manage your applications?

Most businesses are turning to open source solutions such as Kubernetes, and Kubernetes is already running containers in the majority of situations for many organizations

Blog

To learn about the enterprise-grade solution for Kubernetes, read about the IBM Cloud Kubernetes Service.

IBM Cloud Team
IBM Cloud

Follow IBM Cloud

Be the first to hear about news, product updates, and innovation from IBM Cloud.

Email subscribe

RSS

IBM Cloud Technologies

Analytics

Artificial intelligence

Automation

Blockchain

Cloud

Compute

Data science

Database

DevOps

Site feedback

Let's talk

Blog

Hosting

Hybrid cloud

Integration

Internet of things

Management

Migration

Mobile

Networking

Open source

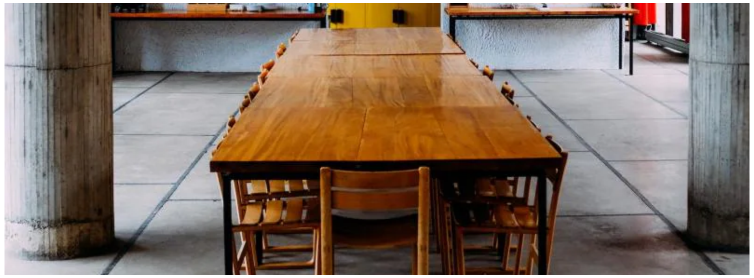
Security

Storage

Site feedback

Let's talk

Related Articles



Blog

Data at the Edge

By: Ashok Iyengar

4 June 2021

IBM WebSphere Automation v1.1

By: Michael Thompson

3 June 2021

More Play and Less Pr
Automates Role-Playi
IBM Watson

By: Allison Leking

3 June 2021

Be the first to hear about news, product updates, and innovation from IBM Cloud

Get updates to your inbox.

Site feedback

Why IBM Cloud

Why IBM Cloud

Hybrid Cloud approach

Trust and security

Open Cloud

Data centers

Case studies

Products and Solutions

Cloud Paks

Cloud pricing

View all products

View all solutions

Let's talk

Learn about

Resources

Blog

What is Cloud Computing?

What is Confidential Computing?

What is a Data Lake?

What is a Data Warehouse?

What is Artificial Intelligence (AI)?

What is Machine Learning?

What is DevOps?

What is Microservices?

Docs

Architectures

IBM Garage

Training and Certifications

Partners

Cloud blog

Hybrid Cloud careers

My Cloud account

Site feedback

Let's talk