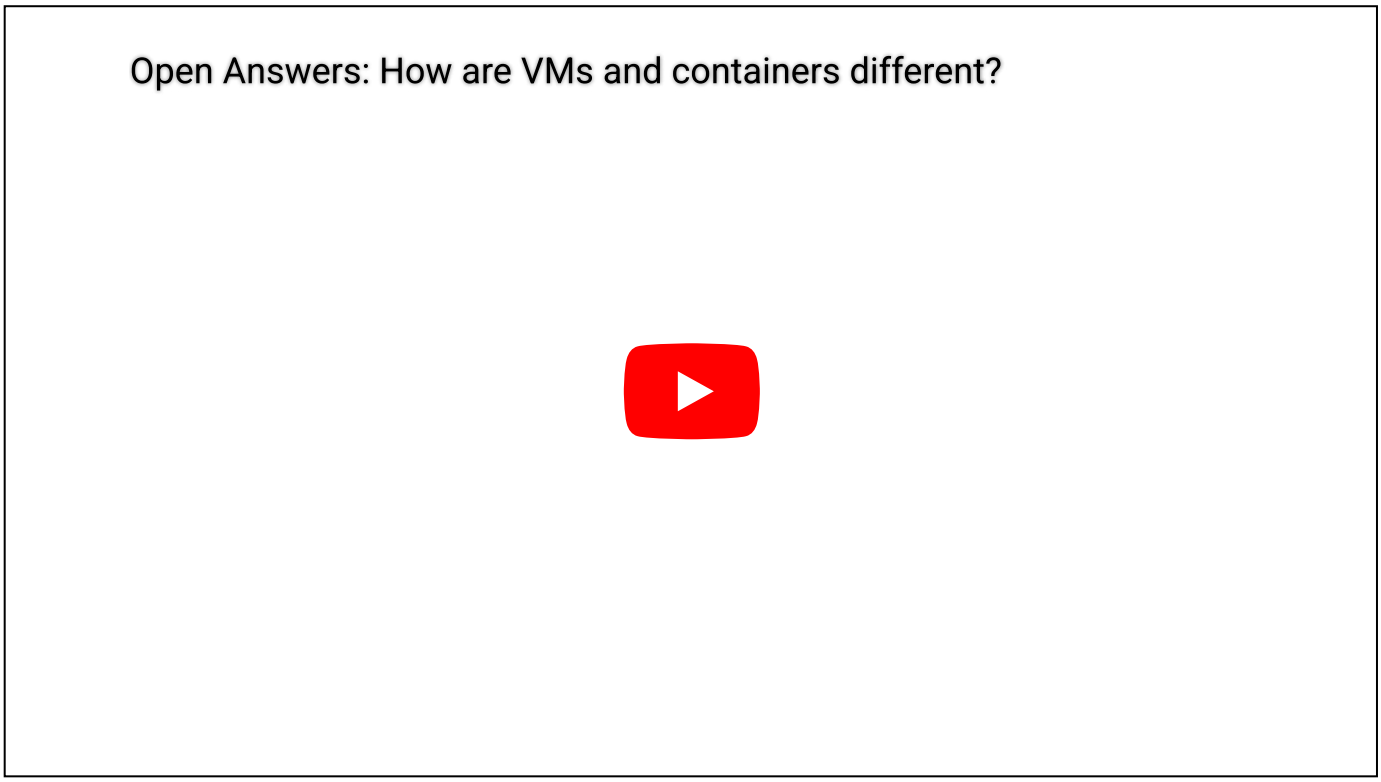
Containers vs VMs

Updated December 13, 2023

Copy URL

Overview

Containers and virtual machines (VMs) are 2 approaches to packaging computing environments that combine various IT components and isolate them from the rest of the system. The main difference between the 2 is what components are isolated, which in turn affects the scale and portability of each approach.



What is a container?

A container is a unit of software that holds together all the components and functionalities needed for an application to run. Most modern applications are made up of multiple containers that each perform a specific function. Containers are typically measured by the megabyte, don't utilize a hypervisor, and are generally considered a faster, more agile way of handling process isolation.

One of the significant factors which contribute to the success of containers is portability. Much like interconnecting LEGO™ blocks, the individual containers can easily be exchanged and moved around to different environments. Once an application and its dependencies are packaged into a container, it can be deployed anywhere needed–the developer laptop, the datacenter, the cloud, or the edge–with the expectation that it will function exactly the same.

Docker, an open-source platform for building, deploying, and managing containerized applications, has played a major role in the evolution of container technology over the years.

What is a virtual machine?

Virtual machines play a crucial role in cloud computing, emulating physical computers by running operating systems in isolated instances. Multiple VMs are commonly hosted on a single server, with a hypervisor acting as a lightweight software layer positioned between the physical host and

the VMs. This hypervisor efficiently manages access to resources, enabling virtual machines to function as distinct servers while offering enhanced flexibility and agility.

Initially gaining popularity in the 2000s due to consolidation and cost saving initiatives, the use of VMs have evolved over time. Organizations have matured their VM deployments, expanding beyond consolidation to embrace various use cases. These include providing on-demand resources for applications and optimizing access to expensive resources like GPUs.

VMs have also served as the foundation of many early cloud computing environments, facilitating resource virtualization and supporting multi-tenancy and isolation–that is, multiple customers running systems sharing the same resources.

Virtual machines contain their own operating system, allowing them to perform multiple resource-intensive functions at once. The increased resources available to VMs allow them to abstract, split, duplicate, and emulate entire servers, operating systems, desktops, databases, and networks.

Cloud-native vs traditional IT

Beyond the technological differences, comparing containers to virtual machines is a proxy comparison between modern cloud-native IT practices and traditional IT architectures.

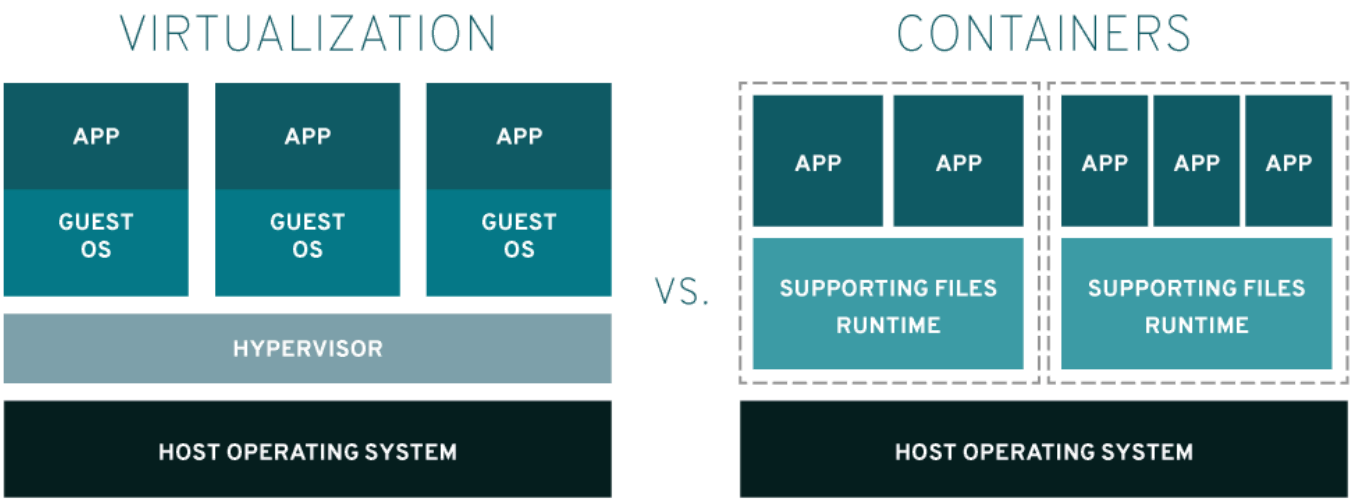
Emerging IT practices (cloud-native development, CI/CD, and DevOps) are possible because workloads are split up into the smallest possible serviceable units—usually a function or microservice–and run in isolation, where they are independently developed, deployed, managed, and scaled.

These small units are best packaged in containers, which allow multiple teams to work on individual parts of an app or service without interrupting or threatening code packaged in other containers.

Traditional IT architectures (monolithic and legacy) keep every aspect of a workload tightly coupled and unable to function without the greater architecture in place. Because aspects cannot be split up, they need to be packaged as a whole unit within a larger environment, often a VM.

It was once common to build and run an entire app within a VM, though having all the code and dependencies in one place led to oversized VMs that experienced cascading failures and downtime when pushing updates.

How do they work?



Virtualization

Software called a hypervisor separates resources from their physical machines so they can be partitioned and dedicated to VMs. When a user issues a VM instruction that requires additional resources from the physical environment, the hypervisor relays the request to the physical system and caches the changes. VMs look and act like physical servers, which can multiply the drawbacks of application dependencies and large OS footprints—a footprint that's mostly not needed to run a single app or microservice.

Containers

Everything within a container is packaged and shipped using a container image—a file that includes all libraries and dependencies. The container image files are similar to software installation packages (such as RPMs in Linux), but one that only needs a compatible kernel and container runtime for the application to run, regardless of which OS was used to create the container nor where the libraries inside the container were sourced from. Because containers are so small, there are usually hundreds of them loosely coupled together—which is why container orchestration platforms (like Red Hat OpenShift and Kubernetes) are used to provision and manage them.

Which one should I use?

That depends—do you need a small instance of something that can be moved easily (containers), or do you need a semi-permanent allocation of custom IT resources?

Additional factors to consider include application architecture, development practices, security and regulatory requirements.

The small, lightweight nature of containers allows them to be deployed easily across bare metal systems as well as public, private, hybrid, and multicloud environments. It's also very common to run containers in VMs since organizations have existing infrastructure built around virtual machines. This speaks to the flexibility of containers.

Containers are also the ideal environment to deploy today's cloud-native apps, which are collections of microservices designed to provide a consistent development and automated management experience across public, private, hybrid, and multicloud environments. Cloud-native apps help speed up how new apps are built, how existing ones are optimized, and how they're all connected.

Compared to VMs, containers are best used to:

- Build cloud-native apps
- Package microservices
- Incorporate applications into DevOps or CI/CD practices
- Move scalable IT projects across a diverse IT footprint

Compared to containers, VMs are best used to:

- House traditional, legacy, and monolithic workloads
- Isolate risky development cycles
- Provision infrastructural resources (such as networks, servers, and data)
- Run a different OS inside another OS (such as running Unix on Linux)

And if you have applications running on both virtual machines and containers, Red Hat Service Interconnect can help connect applications and services across different environments.

Bare metal vs VMs vs containers

Virtual machines and containers can be deployed on various types of infrastructure, including bare metal servers.

What is bare metal?

‘Bare metal’ is a term that refers to a computer or server that runs on physical hardware and does not require assistance from hypervisors, virtual machines, or containerization in order to operate. Bare metal servers are also known as dedicated servers–this is because the hardware components are not shared with other users, and are therefore entirely dedicated to a single tenant.

Bare metal servers are capable of processing a high volume of data with low latency–that is, they’re fast and powerful. With bare metal, the user has complete control over their server infrastructure, which means they can choose their own operating system and fine tune their hardware and software to fit their specific workload needs.

However, while bare metal deployments are valuable in scenarios where performance and direct hardware access are critical, they may not provide the same level of flexibility and resource management as containers or virtual machines.

Can I host VMs on bare metal?

Yes, bare metal servers are able to host virtual machines with the addition of a hypervisor and virtualization software.

Can I host containers on bare metal?

Yes, platforms like Docker, Kubernetes, and Podman are designed to help users manage and deploy containers at scale on many infrastructures, including bare metal servers.

Why choose Red Hat?

Red Hat® OpenShift® is a single enterprise-ready container-based application platform with a choice of deployment and consumption options that supports every app and environment. With Red Hat OpenShift, organizations can quickly build, deploy, run, and manage applications anywhere, securely, and at scale.

Red Hat OpenShift Virtualization, a feature of Red Hat OpenShift, allows IT teams to run virtual machines alongside containers on the same Kubernetes platform, simplifying management and improving time to production.

This allows organizations to benefit from their existing investments in virtualization, while taking advantage of the simplicity and speed of a modern application platform. By integrating VMs into the OpenShift application platform, it provides a consistent environment for application development and deployment. Developers can build, test, and deploy applications faster, accelerating time to market.

Why choose Red Hat for containers?

Why choose Red Hat for virtualization?

Keep reading

ARTICLE

What's a Linux container?

A Linux container is a set of processes isolated from the system, running from a distinct image that provides all the files necessary to support the processes.

Read more →

ARTICLE

Containers vs VMs

Linux containers and virtual machines (VMs) are packaged computing environments that combine various IT components and isolate them from the rest of the system.

Read more →

ARTICLE

What is container orchestration?

Container orchestration automates the deployment, management, scaling, and networking of containers.

Read more →

More about containers

- Products
- Related articles
- Resources
- Training




- Products
- Tools
- Try, buy, & sell
- Communicate

About Red Hat

We're the world's leading provider of enterprise open source solutions—including Linux, cloud, container, and Kubernetes. We deliver hardened solutions that make it easier for enterprises to work across platforms and environments, from the core datacenter to the network edge.

Select a language

English