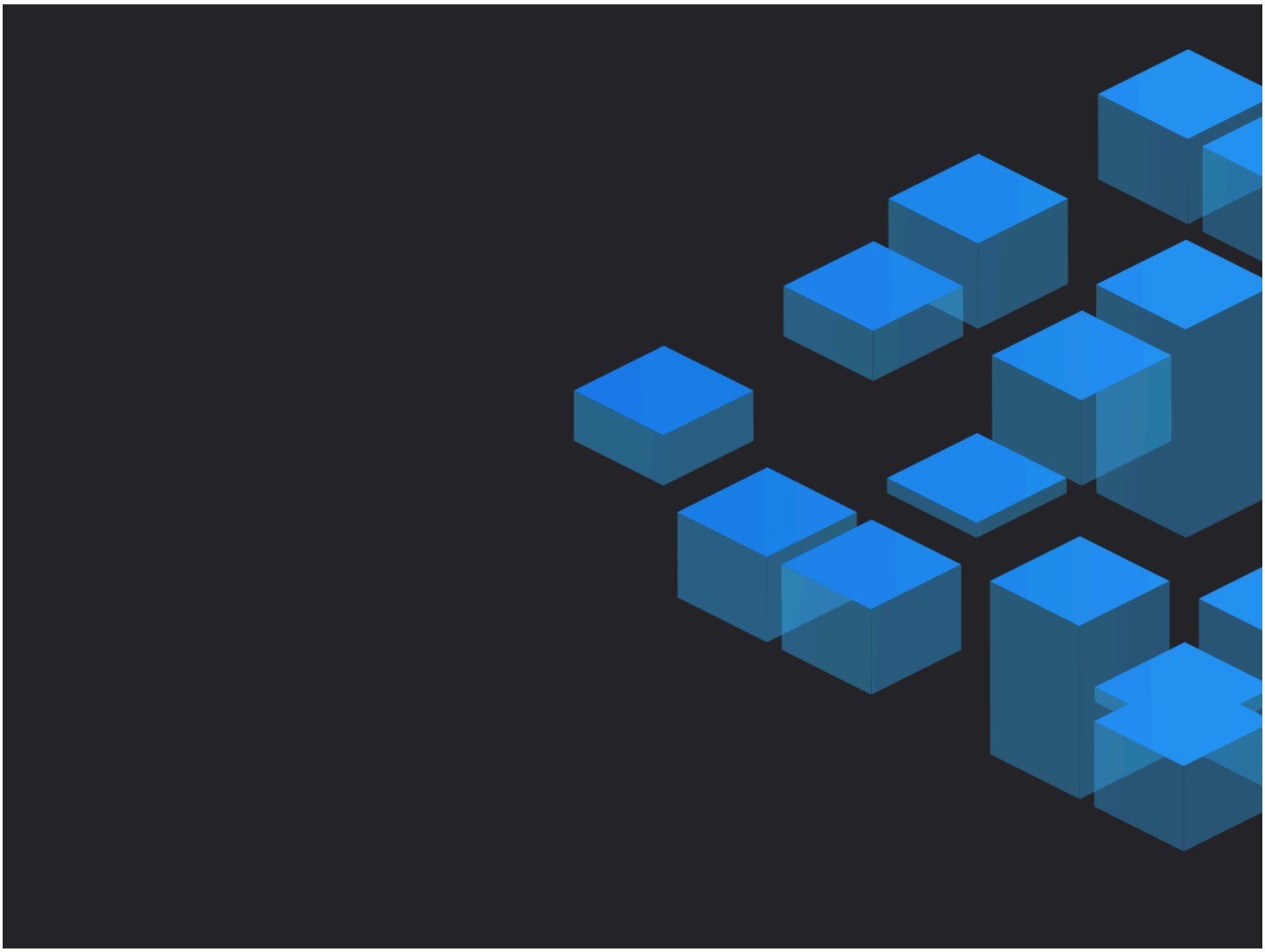
What are containers?

Explore IBM's containers solutions →

Subscribe for cloud updates →



What are containers?

Containers are executable units of software in which application code is packaged along with its libraries and dependencies, in common ways so that the code can run anywhere—whether it be on desktop, traditional IT or the cloud.

Containers take advantage of a form of operating system (OS) virtualization in which features of the OS kernel (for example, Linux namespaces and cgroups, Windows silos and job objects) can be used to isolate processes and control the amount of CPU, memory and disk that those processes can access.

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 use the features and resources of the host OS.

Containers first appeared decades ago with versions such as FreeBSD Jails and AIX Workload Partitions, but most modern developers remember 2013 as the start of the modern container era with the introduction of Docker.

Guide

Strategic app modernization drives digital transformation

Strategic application modernization is one key to transformational success that can boost annual revenue and lower maintenance and running costs.

→

Related content

Register for the guide on DaaS →

Containers versus virtual machines

One way to better understand a container is to comprehend how it differs from a traditional virtual machine (VM). In traditional virtualization—whether it be on-premises or in the cloud—a hypervisor is used to virtualize physical hardware. Each VM then contains a guest OS and a virtual copy of the hardware that the OS requires to run, along with an application and its associated libraries and dependencies.

Instead of virtualizing the underlying hardware, containers virtualize the operating system (typically Linux) so each individual container contains *only* the application and its libraries and dependencies. The absence of the guest OS is why containers are so lightweight and, thus, fast and portable.

For a deeper look at this comparison, check out "Containers vs. VMs: What's the difference?"

Benefits of containers

The primary advantage of containers, especially as compared to a VM, is that they provide a level of abstraction that makes them lightweight and portable. Their primary benefits include:

Lightweight: Containers share the machine OS kernel, eliminating the need for a full OS instance per application and making container files small and easy on resources. Their smaller size, especially compared to VMs, means that containers can spin up quickly and better support cloud-native applications that scale horizontally.

Portable and platform-independent: Containers carry all their dependencies with them, meaning that software can be written once and then run without needing to be re-configured across laptops, cloud and on-premises computing environments.

Supports modern development and architecture: Due to a combination of their deployment portability/consistency across platforms and their small size, containers are an ideal fit for modern development and application patterns—such as DevOps, serverless and microservices—that are built by using regular code deployments in small increments.

Improves utilization: Like VMs before them, containers enable developers and operators to improve CPU and memory utilization of physical machines. Where containers go even further is that because they also enable microservices architecture, application components can be deployed and scaled more granularly. This is an attractive alternative option compared to having to scale up an entire monolithic application because a single component is struggling with its load.

In a recent IBM survey, developers and IT executives reported many other ben
using containers.

Download the full report: Containers in the enterprise

Use cases for containers

Containers are becoming increasingly prominent, especially in cloud environments. Many organizations are even considering containers as a replacement of VMs as the general-purpose compute platform for their applications and workloads. But within that broad scope, there are key use cases where containers are especially relevant.

- **Microservices**: Containers are small and lightweight, which makes them a good match for microservice architectures where applications are constructed of many, loosely coupled and independently deployable smaller services.
- **DevOps**: The combination of microservices as an architecture and containers as a platform is a common foundation for many teams that embrace DevOps as the way they build, ship and run software.
- **Hybrid and multicloud**: Because containers can run consistently anywhere—across laptops, on-premises and cloud environments—they are an ideal underlying architecture for hybrid cloud and multicloud scenarios in which organizations find themselves operating across a mix of multiple public clouds in combination with their own data center.
- **Application modernizing and migration**: One of the most common approaches to application modernization is to containerize applications in preparation for cloud migration.

Containerization

Software needs to be designed and packaged differently in order to take advantage of containers—a process commonly referred to as containerization.

When containerizing an application, the process includes packaging an application with its relevant environment variables, configuration files, libraries and software dependencies. The result is a container image that can then be run on a container platform.

Container orchestration with Kubernetes

As companies began embracing containers—often as part of modern, cloud-native architectures—the simplicity of the individual container began colliding with the complexity of managing hundreds (or even thousands) of containers across a distributed system.

To address this challenge, container orchestration emerged as a way of managing large volumes of containers throughout their lifecycle, including:

- Provisioning
- Redundancy
- Health monitoring
- Resource allocation
- Scaling and load balancing
- Moving between physical hosts

While many container orchestration platforms (such as Apache Mesos, Nomad and Docker Swarm) were created, Kubernetes, an open-source project introduced by Google in 2014, quickly became the most popular container orchestration platform, and it's the one the majority of the industry has based its standardization on.

Kubernetes enables developers and operators to declare a desired state of their overall container environment through YAML files, and then Kubernetes does all the processing work of establishing and maintaining that state, with activities that include deploying a specified number of instances of a given application or workload, rebooting that application if it fails, load balancing, auto-scaling, zero downtime deployments and more.

Kubernetes is now operated by the Cloud Native Computing Foundation (CNCF), which is a vendor-agnostic industry group operated under the auspices of the Linux Foundation.

Istio, Knative and the expanding containers ecosystem

As containers continue to gain momentum as a popular way to package and run applications, the ecosystem of tools and projects designed to accommodate and expand production use cases continues to grow. Beyond Kubernetes, two of the most popular projects in the containers ecosystem are Istio and Knative.

Istio

As developers use containers to build and run microservice architectures, management concerns go beyond the lifecycle considerations of individual containers and into the ways that large numbers of small services—often referred to as a "service mesh"—connect with and relate to one another. Istio was created to make it easier for developers to manage the associated challenges with discovery, traffic, monitoring, security and more.

Learn more about Istio

Knative

Serverless architectures continue to grow in popularity as well, particularly within the cloud-native community. Knative, for example, offers substantial value in its ability to deploy containerized services as serverless functions.

Instead of running all the time and responding when needed (as a server does), a serverless function can "scale to zero," which means it isn't running at all unless it's called upon. This model can save vast amounts of computing power when applied to tens of thousands of containers.

Related solutions

Red Hat® OpenShift® on IBM Cloud®

Let's talk

Red Hat OpenShift on IBM Cloud uses OpenShift in public and hybrid environments for velocity, market responsiveness, scalability and reliability.

Explore Red Hat OpenShift on IBM Cloud →

IBM Cloud Pak for Applications

Whether it's deployment, building new cloud-native applications, refactoring or replatforming existing applications, CP4Apps has it covered.

Explore IBM Cloud Pak for Applications →

IBM Cloud Satellite®

With IBM Cloud Satellite, you can launch consistent cloud services anywhere—on premises, at the edge and in public cloud environments.

Explore IBM Cloud Satellite →

IBM Cloud Code Engine

Run container images, batch jobs or source code as server-less workloads—with no sizing, deploying, networking or scaling required.

Explore IBM Cloud Code Engine →

IBM Cloud Container Registry

IBM Cloud Container Registry gives you a private registry that lets you manage your images and monitor them for safety issues.

Explore IBM Cloud Container Registry →

Optimize Kubernetes with IBM® Turbonomic®

Automatically determine the right resource allocation actions—and when to make them—to help ensure that your Kubernetes environments and mission-critical apps get exactly what they need to meet your SLOs.

Explore IBM Turbonomic →

Resources

Containers in the enterprise

New IBM research documents the surging momentum of container and Kubernetes adoption.

Read the ebook →

Combine the best features of cloud and traditional IT

Container orchestration is a key component of an open hybrid cloud strategy that lets you build and manage workloads from anywhere.

Learn more →

What is Docker?

Docker is an open-source platform for building, deploying and managing containerized applications.

Learn more →

Let's talk

Take the next step

Red Hat OpenShift on IBM Cloud offers developers a fast and secure way to containerize and deploy enterprise workloads in Kubernetes clusters. Offload tedious and repetitive tasks involving security management, compliance management, deployment management and ongoing lifecycle management.

Explore Red Hat OpenShift on IBM Cloud →

Start for free →

Let’s talk