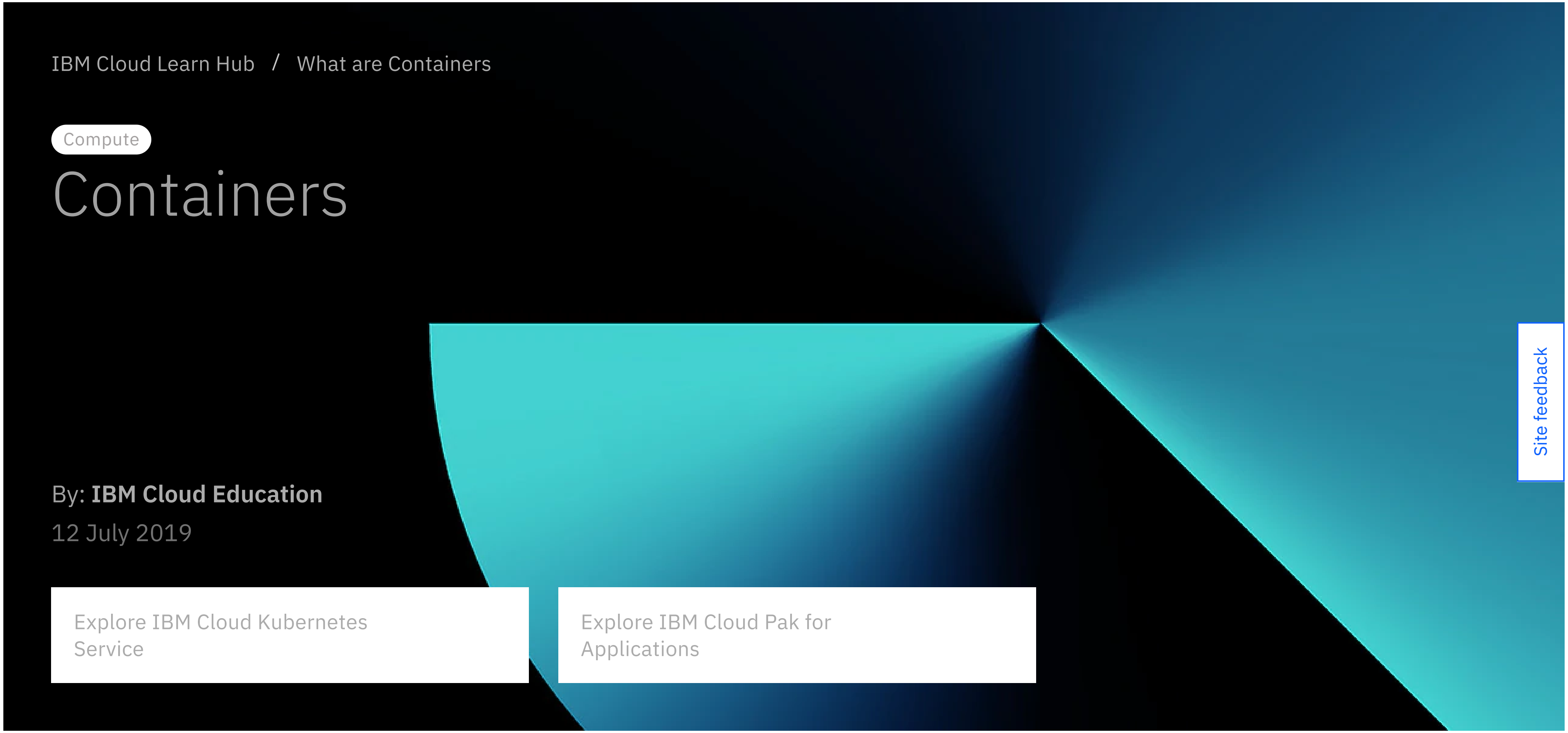
IBM Contact us Log in Create IBM Cloud account



IBM Cloud Learn Hub / What are Containers

Compute

Containers

By: **IBM Cloud Education**

12 July 2019

Explore IBM Cloud Kubernetes Service

Explore IBM Cloud Pak for Applications

Site feedback

What are containers?

Let's talk

This guide looks at the importance of containers in cloud computing, highlights core benefits, and tours the emerging ecosystem of related technologies across Docker, Kubernetes, Istio, and Knative.

What are containers?

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

To do this, containers take advantage of a form of operating system (OS) virtualization in which features of the OS (in the case of the Linux kernel, namely the namespaces and cgroups primitives) are leveraged to both isolate processes and control the amount of CPU, memory, and disk that those processes have access to.

Site feedback

Let’s talk

the features and resources of the host OS.

Containers first appeared decades ago with versions like 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.

Containers vs. VMs

The easiest way to understand a container is to understand how it differs from a traditional virtual machine (VM). In traditional virtualization—whether it be on-premises or in the cloud—a hypervisor is leveraged to virtualize physical hardware. Each VM then contains a guest OS, 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.

Site feedback

Let’s talk

Benefits of containers

The primary advantage of containers, especially compared to a VM, is providing a level of abstraction that makes them lightweight and portable.

- **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 virtual machines, means they 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—

Site feedback

Let’s talk

- **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 microservice architectures, application components can be deployed and scaled more granularly, an attractive alternative to having to scale up an entire monolithic application because a single component is struggling with load.

Use cases of 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 very 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.

Site feedback

Let’s talk

way they build, ship and run software.

- **Hybrid, multi-cloud**: Because containers can run consistently anywhere, across laptop, on-premises and cloud environments, they are an ideal underlying architecture for hybrid cloud and multicloud scenarios where 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 modernizing applications starts by containerizing them so that they can be migrated to the cloud.

Site feedback

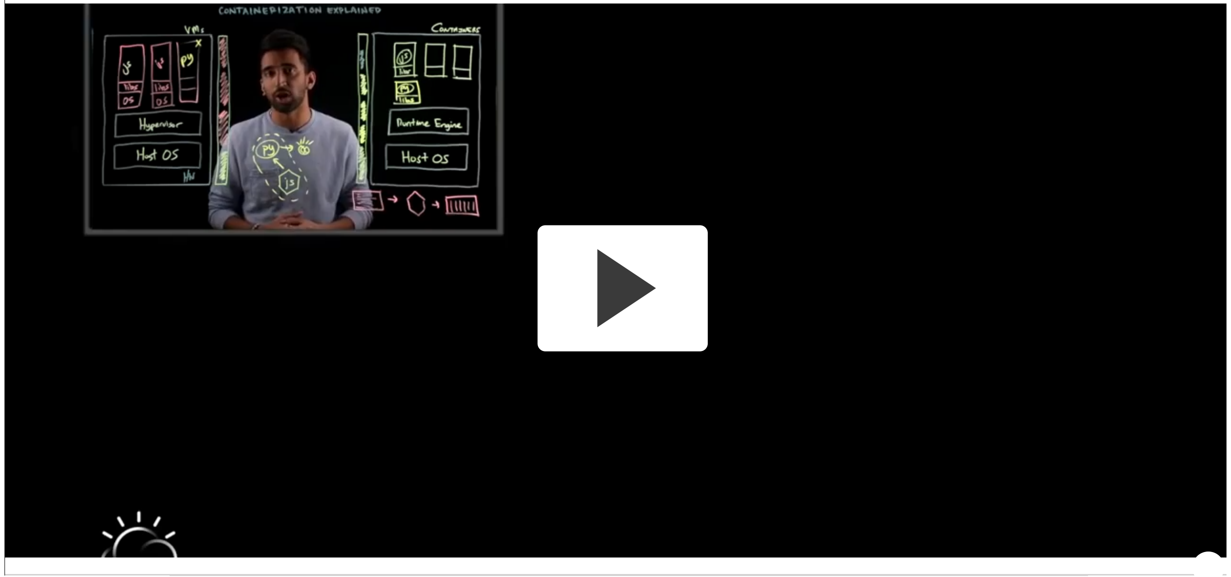
Containerization

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

Let’s talk

Containerization Explained



8:04 / 8:08

Site feedback

Container orchestration with Kubernetes

Let's talk

complexity of managing hundreds (even thousands) of containers across a distributed system.

To address this challenge, container orchestration emerged as a way 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 to help address these challenges, Kubernetes, an open source project introduced by Google in 2014, quickly became the most popular container orchestration platform, and it is the one the majority of the industry has standardized 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 hard work establishing and maintaining that state, with activities that include

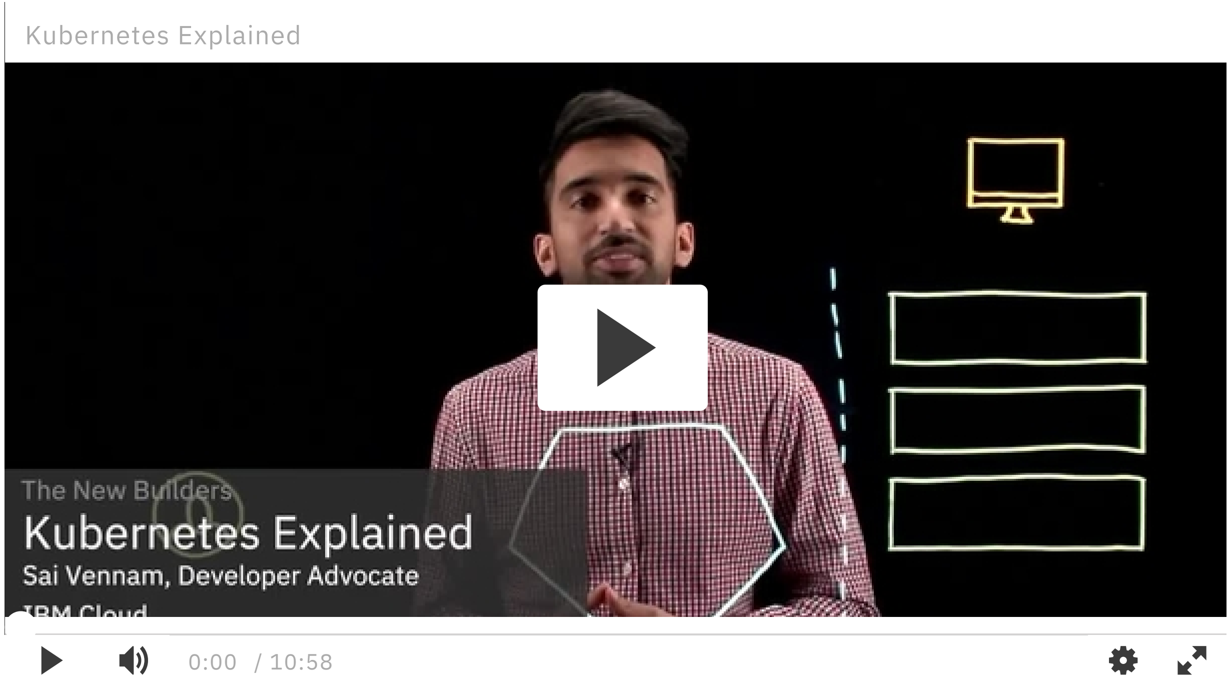
Site feedback

Let’s talk

deployments and more.

To learn more about Kubernetes, Sai Vennam gives an overview of Kubernetes in the below video (10:59):

Kubernetes Explained



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

Istio, Knative, and the expanding containers ecosystem

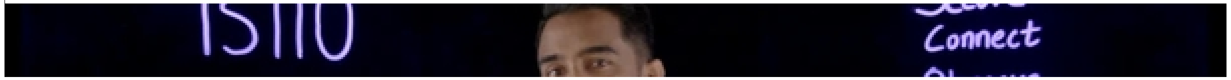
As containers continue to gain momentum as a popular way to package and run applications, the ecosystem of tools and projects designed to harden and expand production use cases continues to grow. Beyond Kubernetes, two of the most popular projects in the containers ecosystem are Istio and Knative.

Site feedback

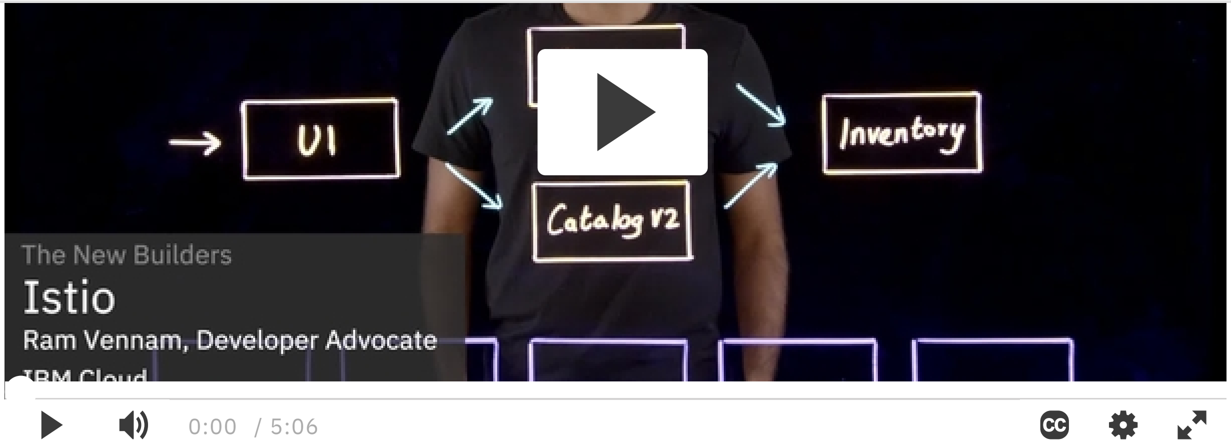
Istio

As developers leverage containers to build and run microservice architectures, management concerns go beyond the lifecycle considerations of individual containers and into the way 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. For more information on Istio, see "What is Istio?" and watch this Istio explainer video (05:06):

Istio Service Mesh Explained



Let’s talk

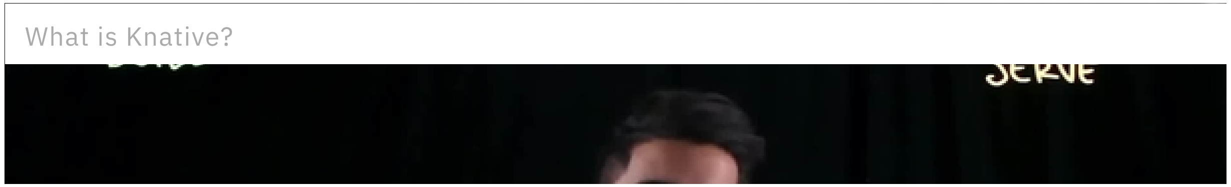


Knative

Serverless architectures continue to grow in popularity as well, particularly within the cloud-native community. Knative’s big value is its ability to offer container services as serverless functions (see "Knative Explained" for a comprehensive overview).

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

For more information on Knative, watch this video called "What is Knative?" (07:58):



This blog post on unifying containers, apps, and functions provides a useful explanation of how Knative sits in the broader container ecosystem.

Tutorials

To get started with containers, the following tutorials are a useful way to understand how to deploy Kubernetes-native apps in clusters and then how to bring DeVops practices, such as continuous deployment, to your containers environment:

- Deploying Kubernetes-native apps in clusters

Site feedback

Let’s talk

IBM Contact us Log in Create IBM Cloud account

Containers and IBM Cloud

IBM Cloud container services are built on open source technologies to facilitate and accelerate your journey to cloud. Build containerized applications using continuous integration and continuous delivery (CI/CD) tools. Orchestrate containers using managed Red Hat OpenShift or Kubernetes services. And modernize existing applications with the containerized IBM middleware and open source components in IBM Cloud Paks.

Learn more about containers on IBM Cloud.

Sign up for an IBMid and create your IBM Cloud account.

Site feedback

Why IBM Cloud

Why IBM Cloud

Hybrid Cloud approach

Trust and security

Products and Solutions

Cloud Paks

Cloud pricing

View all products

Let’s talk

Data centers

Case studies

Learn about

What is Hybrid Cloud?

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?

Resources

Get started

Docs

Architectures

IBM Garage

Training and Certifications

Partners

Cloud blog

Hybrid Cloud careers

My Cloud account

Site feedback

Let’s talk