100 XP

What is Kubernetes?

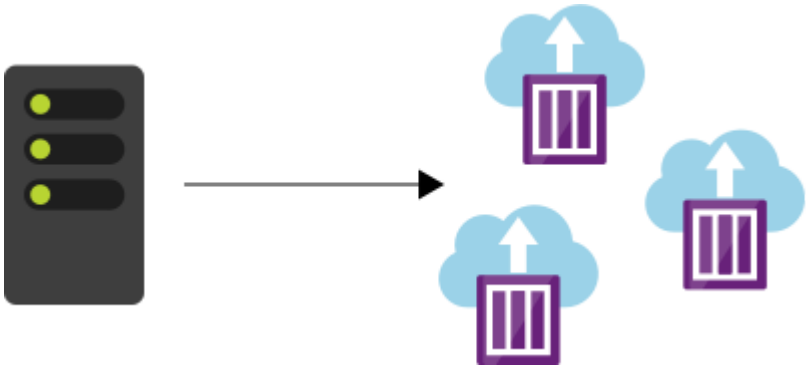
7 minutes

Before we look at what Kubernetes is, let's summarize a few concepts that are key to the applications that you develop and deploy to Kubernetes. This overview should help you decide whether Kubernetes might be a good fit for your containerization management strategy.

What is a container?

A *container* is an atomic unit of software that packages up code, dependencies, and configuration for a specific application. You can use containers to split up monolithic applications into individual services that make up the solution. This rearchitecting of your application will enable you to deploy these separate services via containers.

For example, each application in the drone tracking solution has its own container.



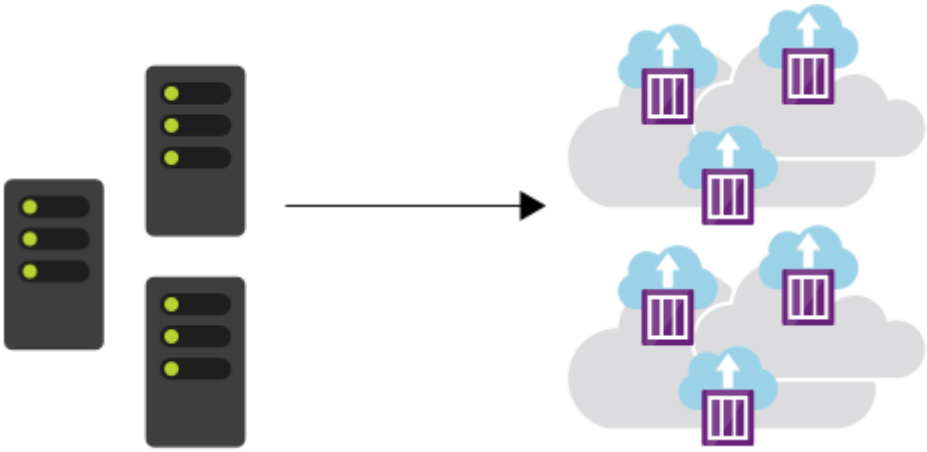
The container concept gives you three major benefits:

- **A container is immutable**: The unchanging nature of a container allows it to be deployed and run reliably with the same behavior from one compute environment to another. A container image that's tested in a QA environment is the same container image that's deployed to production.
- **A container is lightweight**: You can think of a container as a virtual machine (VM) image, but smaller. A VM image is normally installed on a physical host. The image contains both the operating system (OS) and the application that you want to run. In contrast, a container doesn't need an OS, only the application. The container always relies on the host-installed OS for kernel-specific services. Containers are less resource-intensive, and multiple containers can be installed in the same computing environment.

- **Container startup is fast**: Containers can start up in a few seconds instead of minutes, like a VM.

The preceding benefits make containers a popular choice for developers and IT operations alike.

What is container management?



Even though you can think of containers as VMs, you have to keep in mind that they aren't. A container has a distinct life cycle. It's deployed, started, stopped, and destroyed as requested. This life cycle makes containers disposable and affects how developers and IT operations should think about the management of large container deployments.

The process of deploying, updating, monitoring, and removing containers introduces many challenges.

Suppose you want to scale your drone tracking website. You find that at specific times during the day, you need more instances of the site's caching service to manage performance. You can solve this problem by adding more caching service containers.

Assume that you need to roll out a new version of the caching service. How do you make sure that you update all the containers? How do you remove all the older containers?

These types of questions justify a system to help you manage your container deployment.

What is container orchestration?

Container orchestration is a concept that describes all the tasks that you or a system performs to manage containers. A container orchestrator is a system that deploys and manages containerized applications. The orchestrator also dynamically responds to changes in the environment to increase or decrease the deployed instances of the managed application.

What is a cloud-native application?

A cloud-native application is designed and built to take advantage of services such as autoscaling, rolling updates, self-healing, and the other tasks listed earlier. These applications run in, and are managed in, an orchestration platform such as Kubernetes.

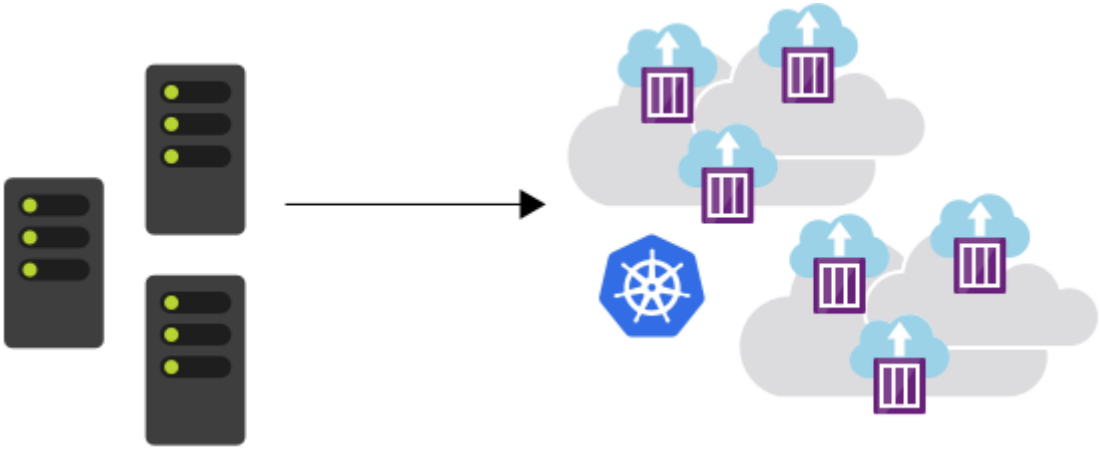
A cloud-native application isn't limited to running in public or private cloud environments. It can also run in a hybrid cloud and in on-premises datacenters.

What is a microservices application?

A microservices application is designed and structured as loosely coupled collaborative services. These services are deployed separately from each other to simplify the design and maintenance of large applications. The services typically communicate over RESTful APIs, or over services that are Advanced Message Queuing Protocol (AMQP) enabled. A microservice-designed application forms an important part of cloud-native applications.

In the drone tracking example, each of the components is easily identifiable as a microservice. Each service can be deployed, updated, and scaled independently from the other.

What is Kubernetes?



Kubernetes is a portable, extensible open-source platform for automating deployment, scaling, and the management of containerized workloads. Kubernetes abstracts away complex container management and provides you with declarative configuration to orchestrate containers in different computing environments. This orchestration platform gives you the same ease of use and flexibility as with platform as a service (PaaS) and infrastructure as a service (IaaS) offerings.

Kubernetes benefits

The benefits of using Kubernetes are based on the abstraction of tasks. These tasks include:

- Deployment of containers.

- Self-healing of containers. An example is restarting containers that fail or replacing containers.
- Scaling application container count up or down dynamically based on demand.
- Automation of rolling updates and rollbacks of containers.
- Management of storage.
- Management of network traffic.
- Storage and management of sensitive information such as usernames and passwords.

Keep in mind that all of the preceding aspects of Kubernetes require configuration and a good understanding of the underlying technologies that are covered. For example, you need to understand concepts such as virtual networks, load balancers, and reverse proxies to configure Kubernetes networking.

Kubernetes considerations

With Kubernetes, you can view your datacenter as one large computer. You don't worry about how and where you deploy your containers, only about deploying and scaling your applications as needed. However, this view might be slightly misleading because there are a few aspects to keep in mind:

- Kubernetes isn't a full PaaS offering. It operates at the container level and offers only a common set of PaaS features.
- Kubernetes isn't monolithic. It's not a single installed application. Aspects such as deployment, scaling, load balancing, logging, and monitoring are all optional. You're responsible for finding the best solution that fits your needs to address these aspects.
- Kubernetes doesn't limit the types of applications that can run on the platform. If your application can run in a container, it can run on Kubernetes. To make optimal use of containerized solutions, your developers need to understand concepts such as microservices architecture.
- Kubernetes doesn't provide middleware, data-processing frameworks, databases, caches, or cluster storage systems. All these items are run as containers or as part of another service offering.

You're responsible for maintaining your Kubernetes environment. For example, you need to manage OS upgrades and the Kubernetes installation and upgrades. You also manage the hardware configuration of the host machines, such as networking, memory, and storage.

Cloud services such Azure Kubernetes Service (AKS) reduce these challenges by providing a hosted Kubernetes environment. These services also simplify the deployment and management of containerized applications in Azure. With AKS, you get the benefits of open-source Kubernetes without the complexity or operational overhead of running your own custom Kubernetes cluster.

> **Note**
>
> Kubernetes is sometimes abbreviated to *K8s*. The 8 represents the eight characters between the K and the s of the word K[*ubernete*]s.

The use of Docker in Kubernetes

For Kubernetes to run containers, it needs to support a container runtime. Kubernetes is responsible for managing the container runtime and doesn't directly manage containers. The container runtime is the object that's responsible for managing containers. For example, the container runtime starts, stops, and reports on the container's status. Docker is such a container runtime.

Next unit: How Kubernetes works

Continue >