Kubernetes vs. Docker

The key differences between Kubernetes and Docker and how they fit into containerization

JOSH CAMPBELL
Product Manager

EDITORIAL CONTRIBUTION: CHANDLER HARRIS

Docker is a containerization platform and runtime and Kubernetes is a platform for running and managing containers from many container runtimes. Kubernetes supports numerous container runtimes, including Docker.

When Docker was introduced in 2013 it brought us the modern era of the container and ushered in a computing model based on microservices. Since containers don't depend on their own operating system, they facilitate the development of loosely-coupled and scalable microservices by allowing teams to declaratively package an application, its dependencies, and configuration together as a container image.

Yet as applications grew in complexity to hold containers distributed across numerous servers, challenges arose, including: how to coordinate and schedule multiple containers, how to enable communications between containers, how to scale container instances, and more. Kubernetes was introduced as a way to solve these challenges.

When it comes to container technology, the common names that arise are Docker and Kubernetes. You may ask: which one is better? But oftentimes it's not either/or, but how you can use both of them to your advantage.

What is Docker?

Docker is a commercial containerization platform and runtime that helps developers build, deploy, and run containers. It uses a client-server architecture with simple commands and automation through a single API.

Docker also provides a toolkit that is commonly used to package applications into immutable container images by writing a Dockerfile and then running the appropriate commands to build the image using the Docker server. Developers can create containers without Docker but the Docker platform makes it easier to do so. These container images can then be deployed and run on any platform that supports containers, such as Kubernetes, Docker Swarm, Mesos, or HashiCorp Nomad.

While Docker provides an efficient way to package and distribute containerized applications, running and managing containers at scale is a challenge with Docker alone. Coordinating and scheduling containers across multiple servers/clusters, upgrading or deploying applications with zero downtime, and monitoring the health of containers are just some of the considerations that need to be made.

To solve these problems and more, solutions to orchestrate containers emerged in the form of Kubernetes, Docker Swarm, Mesos, HashiCorp Nomad, and others. These allow organizations to manage a large volume of containers and users, balance loads efficiently, offer authentication and security, multi-platform deployment, and more.

RELATED MATERIAL

Microservices vs. monolithic architecture

Learn more →

SEE SOLUTION

Manage your components with Compass

Read more →

What is Kubernetes?

Kubernetes (sometimes referred to as K8s) is a popular open source platform that orchestrates container runtime systems across a cluster of networked resources. Kubernetes can be used with or without Docker.

Kubernetes was originally developed by Google, who needed a new way to run billions of containers a week at scale. Kubernetes was released as open source by Google in 2014 and is now widely considered to be the market leader and industry-standard orchestration tool for containers and distributed application deployment. Google notes that Kubernetes' "main design goal is to make it easy to deploy and manage complex distributed systems, while still benefiting from the improved utilization that containers enable."

Kubernetes bundles a set of containers into a group that it manages on the same machine to reduce network overhead and increase resource usage efficiency. An example of a container set is an app server, redis cache, and sql database. Docker containers are one process per container.

Kubernetes is particularly useful for DevOps teams since it offers service discovery, load balancing within the cluster, automated rollouts and rollbacks, self-healing of containers that fail, and configuration management. Plus, Kubernetes is a critical tool for building robust DevOps CI/CD pipelines.

However, Kubernetes is not a complete platform as a service (PaaS) and there are many considerations to keep in mind when building and managing Kubernetes clusters. The complexity that comes with managing Kubernetes is a large factor in why many customers choose to use managed Kubernetes services from cloud vendors.

Kubernetes benefits

Kubernetes, often described as the "Linux of the cloud" is the most popular container orchestration platform for a reason. Here are some of the reasons why:

Automated operations

Kubernetes comes with a powerful API and command line tool, called kubectl, which handles a bulk of the heavy lifting that goes into container management by allowing you to automate your operations. The controller pattern in Kubernetes ensures applications/containers run exactly as specified.

Infrastructure abstraction

Kubernetes manages the resources made available to it on your behalf. This frees developers to focus on writing application code and not the underlying compute, networking, or storage infrastructure.

Service health monitoring

Kubernetes monitors the running environment and compares it against the desired state. It performs automated health checks on services and restarts containers that have failed or stopped. Kubernetes only makes services available when they are running and ready.

Kubernetes vs. Docker

While Docker is a container runtime, Kubernetes is a platform for running and managing containers from many container runtimes. Kubernetes supports numerous container runtimes including Docker, containerd, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface). A good metaphor is Kubernetes as an “operating system” and Docker containers are “apps” that you install on the “operating system”.

On its own, Docker is highly beneficial to modern application development. It solves the classic problem of “works on my machine” but then nowhere else. The container orchestration tool Docker Swarm is capable of handling a production container workload deployment of a few containers. When a system grows and needs to add many containers networked to each other, standalone Docker can face some growing pains that Kubernetes helps address.

When comparing the two, a better comparison is of Kubernetes with Docker Swarm. Docker Swarm, or Docker swarm mode, is a container orchestration tool like Kubernetes, meaning it allows the management of multiple containers deployed across multiple hosts running the Docker server. Swarm mode is disabled by default and is something that needs to be setup and configured by a DevOps team.

Kubernetes orchestrates clusters of machines to work together and schedules containers to run on those machines based on their available resources. Containers are grouped together, through declarative definition, into pods, which is the basic unit of Kubernetes. Kubernetes automatically manages things like service discovery, load balancing, resource allocation, isolation, and scaling your pods vertically or horizontally. It has been embraced by the open source community and is now part of the Cloud Native Computing Foundation. Amazon, Microsoft, and Google all offer managed Kubernetes services on their cloud computing platforms, which significantly reduces the operational burden of running and maintaining Kubernetes clusters and their containerized workloads.

Docker or Kubernetes: Which one is right for you?

So if both Docker Swarm and Kubernetes are container orchestration platforms, which do you choose?

Docker Swarm typically requires less setup and configuration than Kubernetes if you're building and running your own infrastructure. It offers the same benefits as Kubernetes, like deploying your application through declarative YAML files, automatically scaling services to your desired state, load balancing across containers within the cluster, and security and access control across your services. If you have few workloads running, don't mind managing your own infrastructure, or don't need a specific feature Kubernetes offers, then Docker Swarm may be a great choice.

Kubernetes is more complex to set up in the beginning but offers greater flexibility and features. It also has wide support from an active open source community. Kubernetes supports multiple deployment strategies out of the box, can manage your network ingress, and provides observability out of the box into your containers. All major cloud vendors offer managed Kubernetes services that make it significantly easier to get started and take advantage of cloud native features, like auto-scaling. If you are running many workloads and require cloud native interoperability, and have many teams in your organization, which creates the need for greater isolation of services, then Kubernetes is likely the platform you should consider.

Compass and container orchestration

No matter which container orchestration solution you choose, it's important to use a tool to manage the complexity of your distributed architecture as you scale. Atlassian Compass is an extensible developer experience platform that brings disconnected information about engineering output and team collaboration together in a central, searchable location. In addition to helping you tame your microservice sprawl with the Component Catalog, Compass can help you establish best practices and measure the health of your software with Scorecards and provide you with data and insights across your DevOps toolchain using extensions built on the Atlassian Forge platform.

JOSH CAMPBELL

Josh Campbell is a product manager for Atlassian and has worn many hats in his career. He enjoys working on things that make the job of an engineer easier and has deep customer empathy, especially when it comes to working with bad technology tools. In his spare time, Josh likes biking with his daughters, eating and drinking things that are bad for him, and playing with new technologies.

SHARE THIS ARTICLE

NEXT TOPIC
Cloud computing →

Recommended reading

Bookmark these resources to learn about types of DevOps teams, or for ongoing updates about DevOps at Atlassian.

Compass community

Learn more →

Tutorial: Create a component

Learn more →

Get started with Compass for free

Learn more →

Sign up for our DevOps newsletter

Email address

Sign up

PRODUCTS

Jira Software
Jira Align
Jira Service Management
Jira Product Discovery
Confluence
Trello
Bitbucket

View all products

RESOURCES

Technical Support
Purchasing & licensing
Atlassian Community
Knowledge base
Marketplace
My Account

Create support ticket

EXPAND & LEARN

Partners
Training & Certification
Documentation
Developer Resources
Enterprise services

View all resources

ABOUT ATLASSIAN

Company
Careers
Events