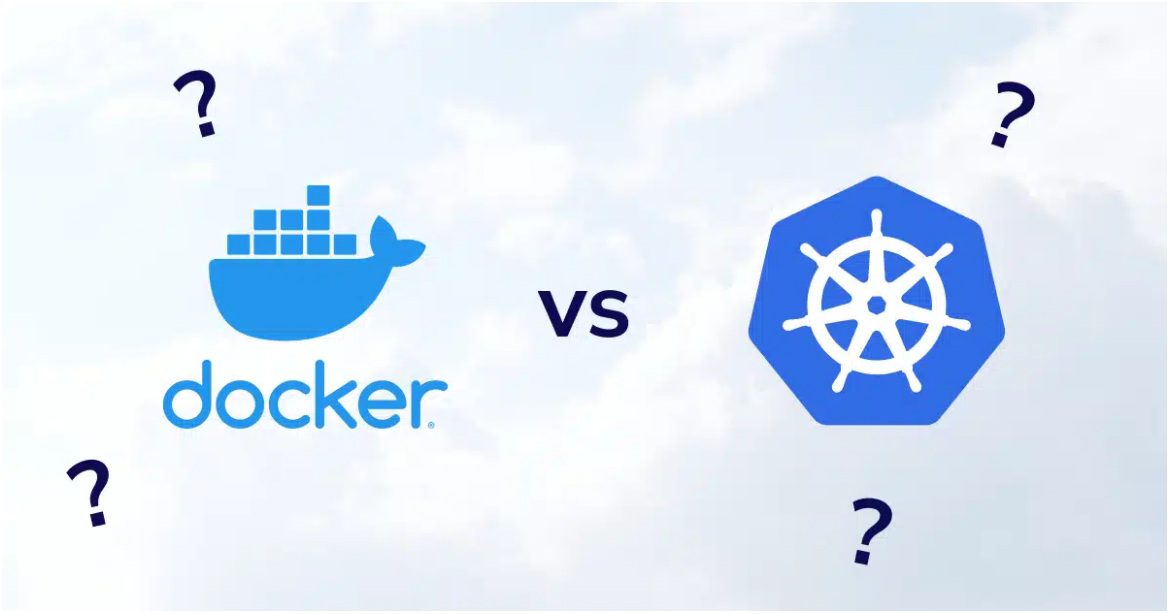
Docker Vs Kubernetes – What's The Difference?

Avi Meir

Web Services | Kubernetes

18 Mar 2023

(https://twitter.com/share?url=https://www.ridge.co/blog/docker-vs-kubernetes-whats-the-difference/&text=Docker vs Kubernetes – What's the Difference?)
(https://wa.me/?text=Hi! I wanted to share with you the page: https://www.ridge.co/blog/docker-vs-kubernetes-whats-the-difference/ - Docker vs Kubernetes – What's the Difference?)
(https://www.linkedin.com/shareArticle?url=https://www.ridge.co/blog/docker-vs-kubernetes-whats-the-difference/&title=Docker vs Kubernetes – What's the Difference?)
(https://www.facebook.com/sharer/sharer.php?u=https://www.ridge.co/blog/docker-vs-kubernetes-whats-the-difference/)



Both Docker and Kubernetes work on a fundamental unit of code called containers; however, comparing Docker and Kubernetes is a common misconception as it's not an apples-to-apples comparison. In fact, Docker and Kubernetes are more like nuts and bolts: they're complementary to each other rather than alternatives. (https://www.ridge.co/kubernetes)

And here's a spoiler: Ridge's Managed Kubernetes (https://www.ridge.co/platform/rks) is a CNCF-certified (https://www.cncf.io/certification/software-conformance/), fully-managed Kubernetes service.

What adds value to Ridge over any other service provider is its Anywhere Cloud (https://www.ridge.co/cloud-solutions/anywhere-cloud/). Its global distribution brings cloud resources closer to the end-user, thus reducing the overall latency associated with cloud-native applications.

See how Ridge Managed K8s can accelerate your cloud-native applications by booking a demo (http://ridge.co/demo)!

In this article, we'll clear up the misconceptions surrounding the "Docker vs Containers vs Kubernetes" debate and provide a thorough understanding of the overall hierarchy.

Let's start with the first and most basic cloud native service: containers.

What Are Containers?

A container is like a compact box holding an application's dependencies all in one place. This not only allows an application to run quickly but also makes it portable and easy to transfer from one environment to another.

What makes containers so popular is their ability to provide abstraction and isolation of resources. This means that a standalone container image is enough for you to run an application on your system without having to install any additional packages.

Unlike Virtual Machines, containers only virtualize the operating system and thus are lightweight. They can be easily transferred from one environment to another and work uniformly throughout.

Now that we know what containers are, we may ask: How are they related to Docker?

For further reading, we have a whole article on What Are Containers (https://www.ridge.co/blog/what-are-containers/).

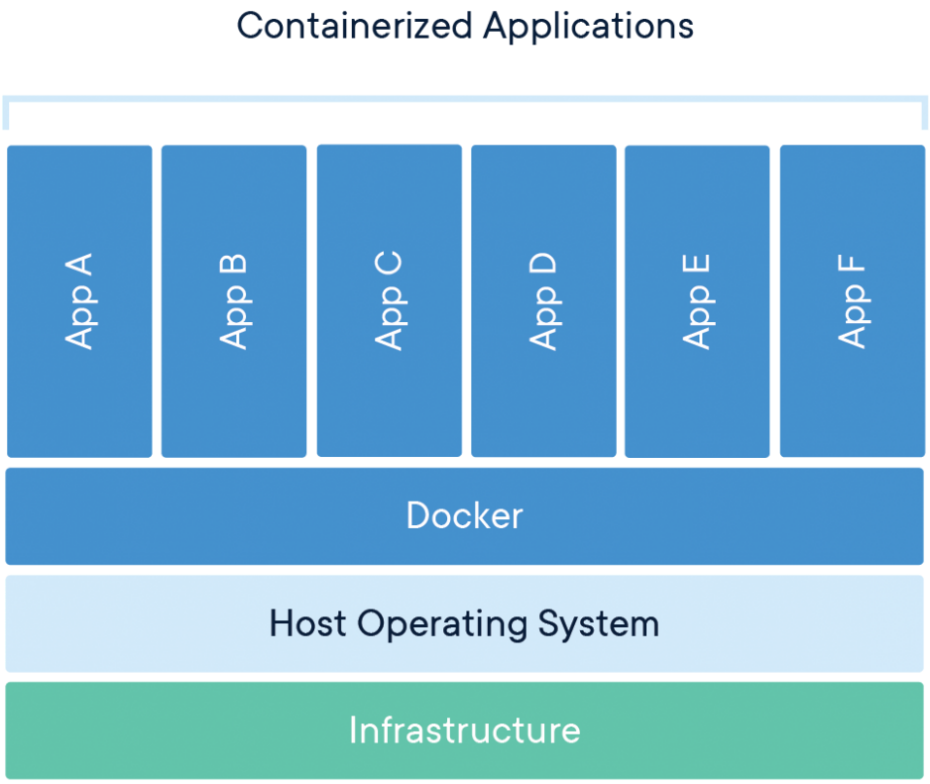


Image Source (https://www.docker.com/resources/what-container)

What is Docker?

Docker is an open-source containerization platform launched by the company Docker Inc. (https://www.docker.com/products/container-runtime) in 2013. It enables you to segment your application from its infrastructure and deploy it quickly on a cloud-premise (or localhost).

Use-case example of Docker

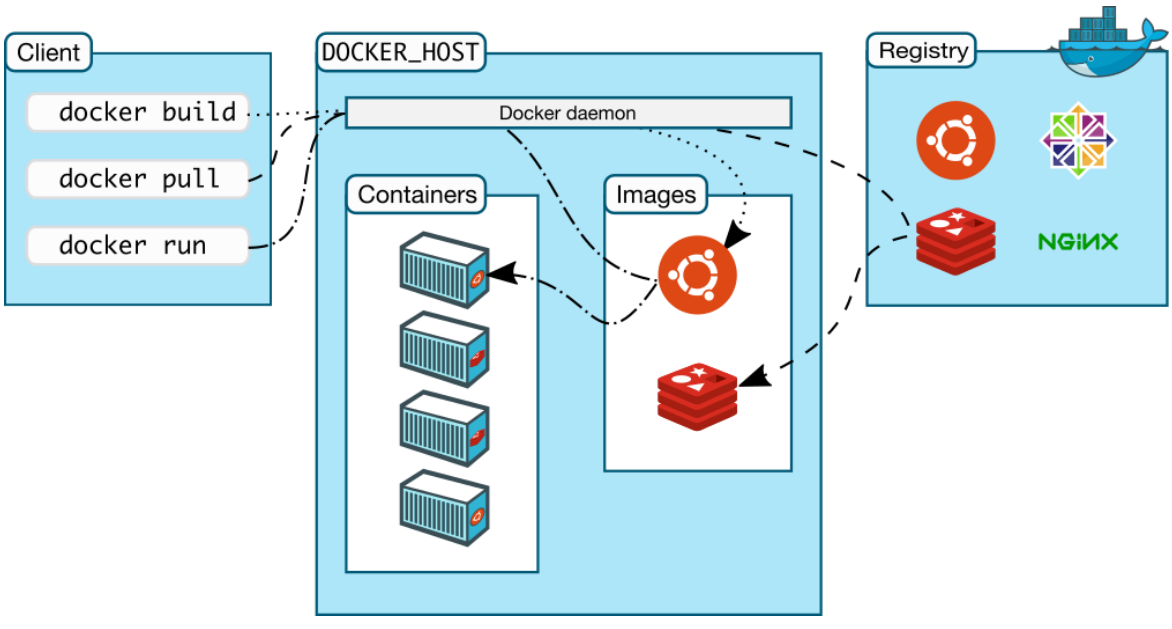
Any conventional application will use a database to store user data, a framework to add new features, and a programming language to hold it all together. Now, imagine having to deal with all these dependencies on your system and push the new changes to the main server—every single time. Docker comes to the rescue here, by streamlining the container creation process.

With Docker Engine, you can run several containers on a given host without worrying about what's installed on the host itself. Moreover, you can store or share these container images through Docker Hub (https://hub.docker.com/) (or a similar registry). Once your development and testing phase is complete, you can deploy your application to the main production environment.

Advantages of using Docker:

1. Operate on Continuous-Integration and Continuous Delivery (CI/CD) workflows
2. Need to scale up or tear down your application
3. Need a lightweight setting due to hardware constraints

The inner workings of Docker and where containers fit in can be seen in the following diagram:



If you are looking for a container orchestration platform, Ridge's Container Service (https://www.ridge.co/platform/rcs/) might be the right choice for you. RCS is a fully managed container service that can be accessed programmatically by developers all over the world. It allows you to run your workloads across the Ridge Network without managing the underlying infrastructure.

What is Kubernetes?

Continuing with the example above, let's say you've put in the effort to convert your monolithic application architecture into a microservice, but as your application gets more and more detailed and your business starts to expand, managing multiple containers manually can be cumbersome.

Containerization will deal with the dependencies, but what do we do if there is a pool of containers to start with? Can you handle it if one of these containers goes down? As a developer, you first need to identify the container, fix the issue, and start a new one.

Even if you are a skilled developer, the time consumed to fix the issue will count towards the system downtime. You are at risk of losing customers during that time and therefore reducing downtime is crucial for business.

Kubernetes (also known as k8s) solves this issue by automating the whole process. It is specially designed to manage container platforms like Docker, containerd, etc. The framework takes care of the scaling as well as the failovers.

Key features of Kubernetes include:

1. Load Balancing: If the traffic on a container increases, Kubernetes automatically distributes the load to make sure the deployment is stable.
2. Rollouts and Rollbacks: With Kubernetes, the user can define the desired state of a container, and the software will change the actual state to the desired state at a regulated rate. For instance, you can tell Kubernetes to remove the existing container when a new one is deployed and adapt to its resources to avoid data loss.

A typical Kubernetes cluster architecture looks like this:

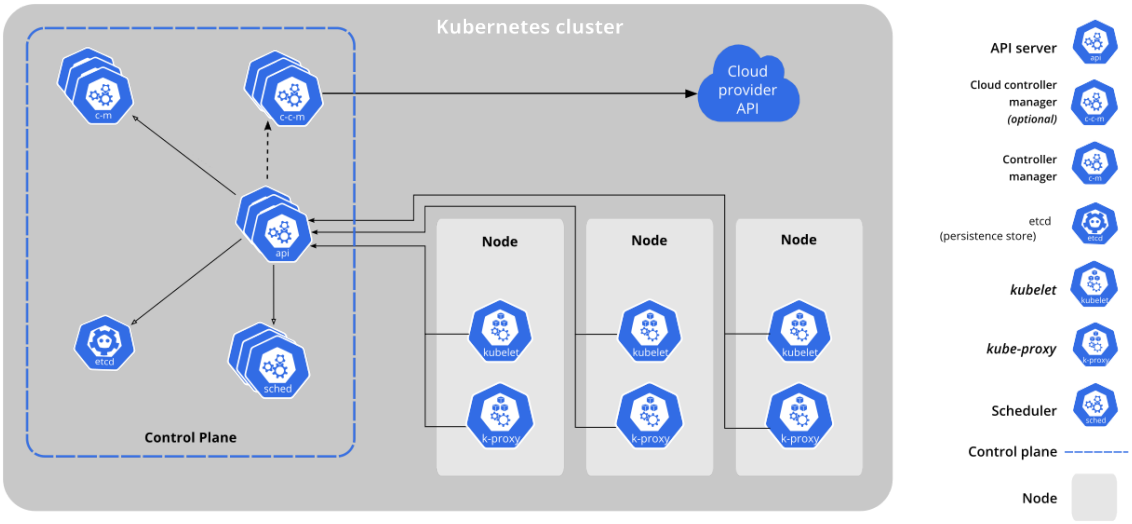


Image Source (https://kubernetes.io/docs/concepts/overview/components/)

Our guide on "What is Kubernetes (https://www.ridge.co/blog/what-is-kubernetes-overview/)", including its architecture, can be helpful if you're planning to use it in a business setting.

Kubernetes vs Docker – What is the Difference?

Now that we know what Docker and Kubernetes are, let's understand how they differ from each other and what we meant by our nuts and bolts analogy.

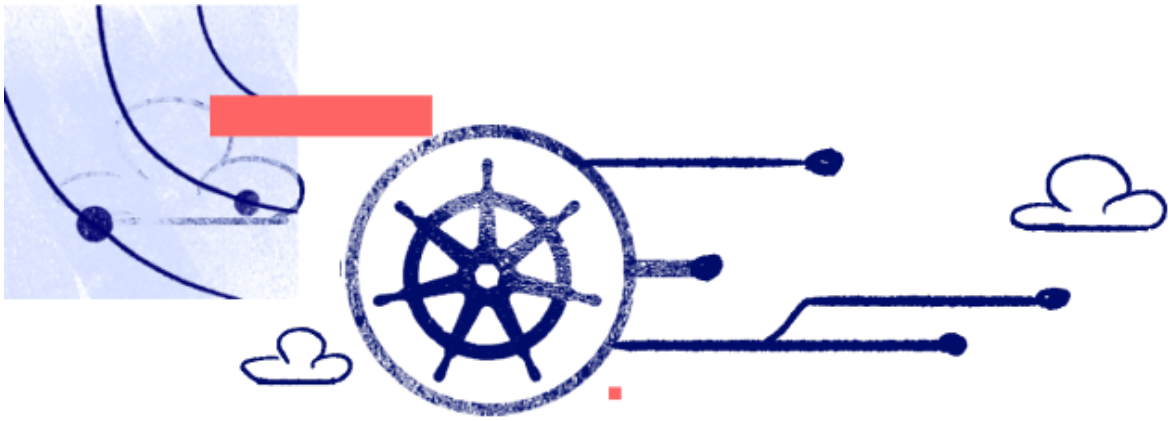
Docker is a containerization platform, which means it is responsible for container creation, whereas Kubernetes is a container orchestration platform (i.e. it manages multi-container applications).

Your journey to containerization starts with a platform like Docker, where things are pretty simple. You create and deploy your application containers on a system. However, as your application develops a layered architecture, it may get difficult to keep up with each layer and its need for resources. That's where Kubernetes comes in.

moving containers from one environment to another.

So what is the difference between Docker and Kubernetes? Kubernetes is meant to run across a cluster, whereas Docker runs on a single node. Thus, the fundamental difference between Docker and Kubernetes is that of an apple to apple pie, the latter being a more extensive framework.

Thus, the Kubernetes vs Docker comparison is not as simple as creating a pros and cons or feature-by-feature list. However, there exists an orchestration technology similar to Kubernetes developed by Docker Inc. called Docker Swarm, which allows us to make a more fair comparison.



Docker Swarm vs Kubernetes

Docker Swarm (https://docs.docker.com/engine/swarm/) has its own API and is tightly integrated into the Docker ecosystem, which makes sense as it is developed by the same company. You do not need to install any additional software to create or manage a swarm. This serves as an advantage to Docker Swarm users, as transitioning from Docker to Docker Swarm can be easier.

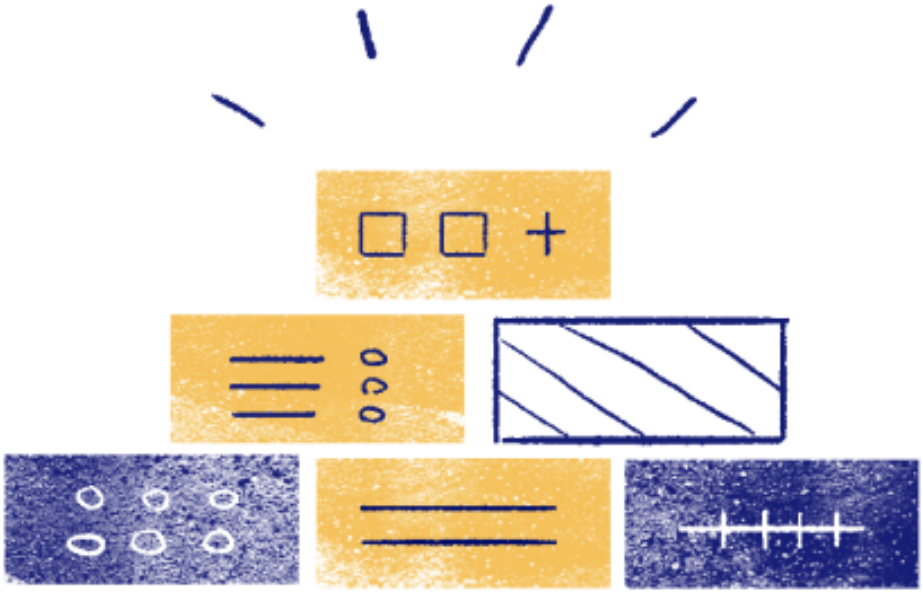
On the other hand, Kubernetes has its own Graphical User Interface (GUI), appealing to the users who prefer UI over the command line. Kubernetes, in general, is more thorough and customizable in a given production environment. It is designed to coordinate clusters at scale as well as monitor systems efficiently.

complex workflows, vast scalability, and being independent of Docker, Kubernetes is the answer.

Can You Use Docker Without Kubernetes?

The short and simple answer is yes, Docker can function without Kubernetes. You see, Docker is a standalone software designed to run containerized applications. Since container creation is part of Docker, you don't need any separate software for Docker to execute.

In fact, if you are starting out on your own in a casual environment, you don't need an extensive tool like Kubernetes right away. A platform like Docker can serve your cloud needs on a basic level quite well.



Can Kubernetes Run Without Docker?

The answer is both yes and no. Kubernetes, in itself, is not a complete solution. It depends on a container runtime to orchestrate; you can't manage containers without having containers in the first place.

Docker, but it's not a requirement.

You may be wondering why you'd have to choose just one when both can work well together? Docker promises to execute the code as containers, while Kubernetes provides a way to manage these containers at a single place.

Isn't that a powerful combination? Kubernetes can help you with load-balancing, app scaling, etc. across all the nodes—nodes that run containers inside them.

Advantages of using Kubernetes with Docker:

1. Provides built-in isolation using namespaces which help you group your containers
2. Makes your infrastructure robust
3. Makes your application highly available and scalable

(https://www.ridge.co/platform/rks/) is a CNCF-certified managed Kubernetes service with a Global Infrastructure Cloud (https://www.ridge.co/data-center/). It brings the cloud close to end-users and reduces latency associated with cloud-native applications.

Moreover, Ridge enables you to fully comply with the regional data sovereignty and data hosting regulations while providing an exemplary user experience. With Ridge Cloud, you can access cloud resources and deploy your clusters in 100+ locations.

Similar to Vanilla Kubernetes, Ridge Managed Kubernetes enables you to create and manage your clusters at ease. However, it has certain added advantages, such as Automated Cluster Provisioning. This means that once you define your cluster setup and constraints, Ridge will automatically provision, configure, and install your application anywhere on the Ridge Cloud.

As a developer, your involvement is reduced to a minimum as the system maintains and corrects itself. You can now spend your crucial work hours on app development and being productive rather than monitoring the system flaws!

Ridge also supports hybrid and multi-cloud (https://www.ridge.co/cloud-solutions/any-infrastructure-cloud/) managed Kubernetes deployments with less time and location-sensitive protocols. As security is an important issue, you are able to access control based on organization, project, or account level. Other key features include external load balancing, persistent storage, auto-healing, etc.

Author:

Avi Meir, |