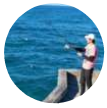
Kubernetes Vs Docker Swarm

Kubernetes vs Docker Swarm—A Comprehensive Comparison



Harsh Binani Follow
Dec 4 · 9 min read

Containers have become quite popular in recent years. They help developers maintain consistency across various platforms, right from development to the production process.

Hence, they allow developers to have more control over their products as compared to traditional virtualization. This is achieved through isolation that is executed at the kernel level without any need of a guest operating system. This introduces many advantages to the equation, such as faster deployments, higher scalability, and closer parity between development environments.

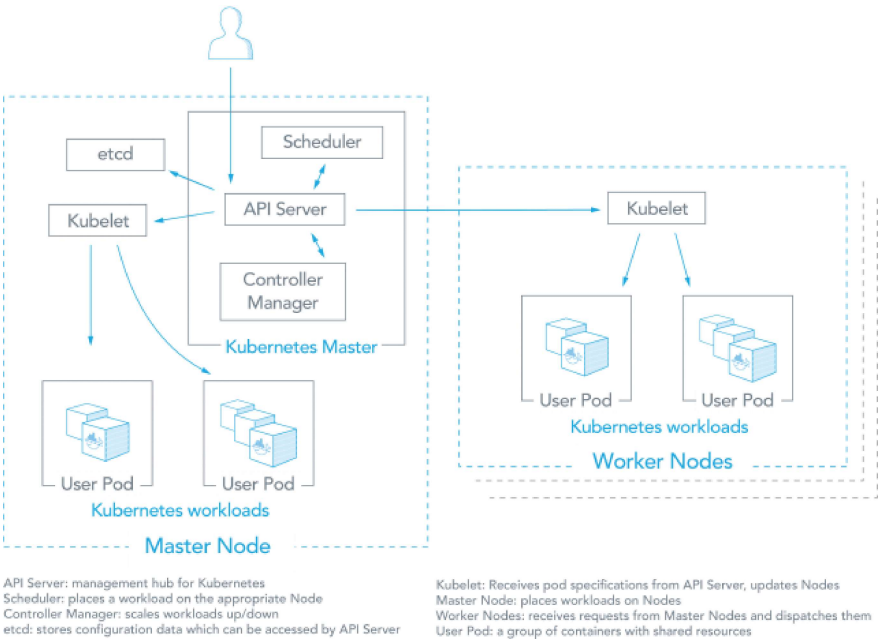
Although there are multiple significant in the market when it comes to container orchestration, including Mesosphere and Amazon ECS, this blog aims to take a comparative look at two of the popular ones—Kubernetes and Docker Swarm.

What is Kubernetes

Kubernetes has been developed by Google and it was introduced in the year 2014. In essence, it is an IT management tool that has been

specifically designed to simplify the scalability of workloads using containers. It has the ability to automate deployment, scaling, and operating application containers.

By using Kubernetes, you will be able to define how your applications should run and the manner in which they interact with other applications. Empowering the user with interfaces and composable platform primitives, Kubernetes allows high degrees of flexibility and reliability.



Source

Although Kubernetes was known to be highly complex in the past, various tools and upgrades have gradually made it more user-friendly. For instance, tools like the Helm Project provide a uniform software packaging method that helps to extend version control and greatly simplifies the application distribution and deployment complexities.

Clusters in Kubernetes include several components like:

• **Pods:** These are a group of containers on the same node that are created, scheduled, and deployed together.

• **Labels:** These are key-value tags that are assigned to elements like pods, services, and replication controllers to identify them.

● **Services:** These are used to give names to Pod groups. As a result, they can act as load balancers to direct traffic to run containers.

● **Replication Controllers:** These are frameworks that have specifically designed for the purpose of ensuring that, at any given moment, a certain number of pod replicas are scheduled and running.

What is Docker Swarm

Docker Swarm has been a popular open source standard for packaging and distributing containerized applications. In essence, it is native clustering for Docker. A pool of Docker hosts can be turned into a single virtual host.

Source

Docker Swarm has been designed around four pillars of principles:

● Simple and powerful user experience through the Docker Universal Control Plane single interface

● Resilient architecture with a single point of failure

● Backward compatibility with existing components (highly relevant for existing users)

● Automatically generated certificates that make it secure

Docker Swarm is now fully integrated with the Docker Engine. It also makes use of the standard API and Networking processes. It is built into the Docker CLI and you can execute a host of tasks through multiple commands that are easy to pick up.

Swarms are a cluster of nodes that consist of the following components:

• **Manager Nodes:** Tasks involved here include Control Orchestration, Cluster Management, and Task Distribution.

• **Worker Nodes:** Functions here include running containers and services that have been assigned by the manager node.

• **Services:** This gives a description of the blueprint via which an individual container can distribute itself across the nodes.

• **Tasks:** These are slots in which single containers place their work.

Being Open Source, the community version of Docker—Docker CE—quickly rose to prominence since it is highly relevant for developers and do-it-yourself Ops teams. On the other hand, the Docker Enterprise Edition is mostly used run mission critical applications. If you are planning to go down the Docker EE road, here are some benefits that you can expect:

• Certified Docker Images and Plugins

- Ability to leverage Docker Datacenter with various levels of options

- Vulnerability scan results on Docker images

- Official support from Docker

This restructuring of Docker products enables the company to strike a balance between the goals of the company and the needs of the community. This makes sense since Docker has been a victim of criticism in the past for adding features to the Docker engine. By segregating the products, the company can play around with the Enterprise Edition while keeping the Community Edition untouched.

But this also poses a problem at a macro scale. Since Docker Swarm is a venture-capital backed company, it needs to be monetized. This translates to the fact that some of the best features would always be paid and kept from the open source community.

On the other hand, Kubernetes is backed by Google who can afford to extend premium features to user base and not directly charge anything for the same.

This has added towards the growth of Kubernetes.

Kubernetes Vs Docker Swarm—A Look at the Differences

Though both of the tools have been wired to save resources by limiting hardware usage to match the business resource requirement, there are some stark differences between them that call for a comprehensive analysis before you go down one chosen road.

Let’s take a look at them:

1. Set up, Installation and Cluster Configuration

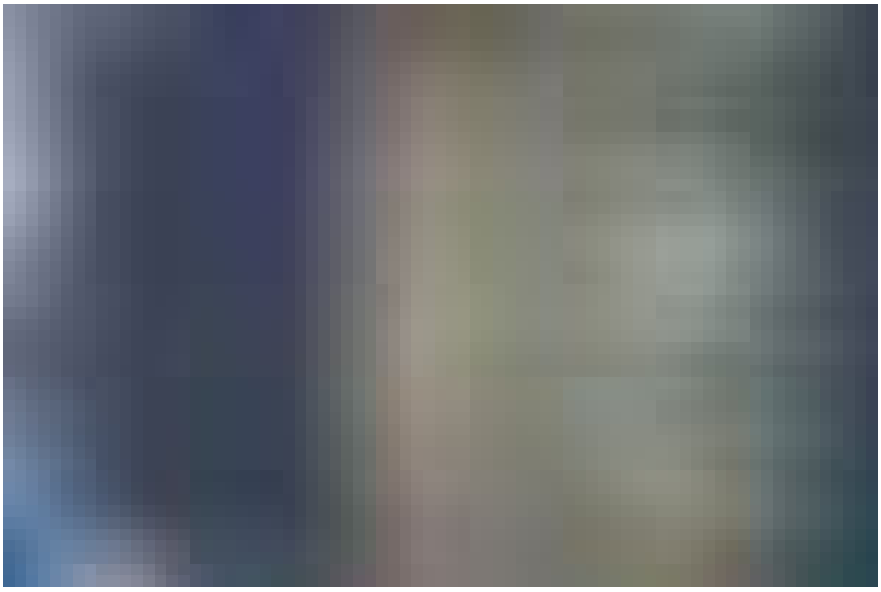


Photo by Markus Spiske on Unsplash

Kubernetes: Installing Kubernetes is easy when it comes to installing it in a test bed. Though if you plan to run it at scale requires a bit more planning and efforts. Specific commands need to be run for every step including:

- Bringing up the cluster
- Defining the environment
- Defining a pod network which enables the containers to interact
- Setting up the dashboard
- Hosting the cluster

Docker Swarm: Swarm uses the Docker CLI to run its programs. Hence, knowledge about only a single set of tools is needed to be learned in order to build environments and configurations. Since the Swarm program runs on your current Docker, you can begin by opting into Swarm.

2. Building and Running Containers

Kubernetes: Kubernetes embraces uniqueness by having its own API, client, and YAML definitions. These are in high contrast to the standard

Docker equivalents since it is not possible to use Docker Compose or Docker CLI to define containers.

Docker Swarm: The same Docker CLI is used and new containers can be spinned with a single command. Although the Swarm API supports multiple tools that work with Docker, it can prove to be a hassle if Docker lacks a specific operation.

3. Logging and Monitoring

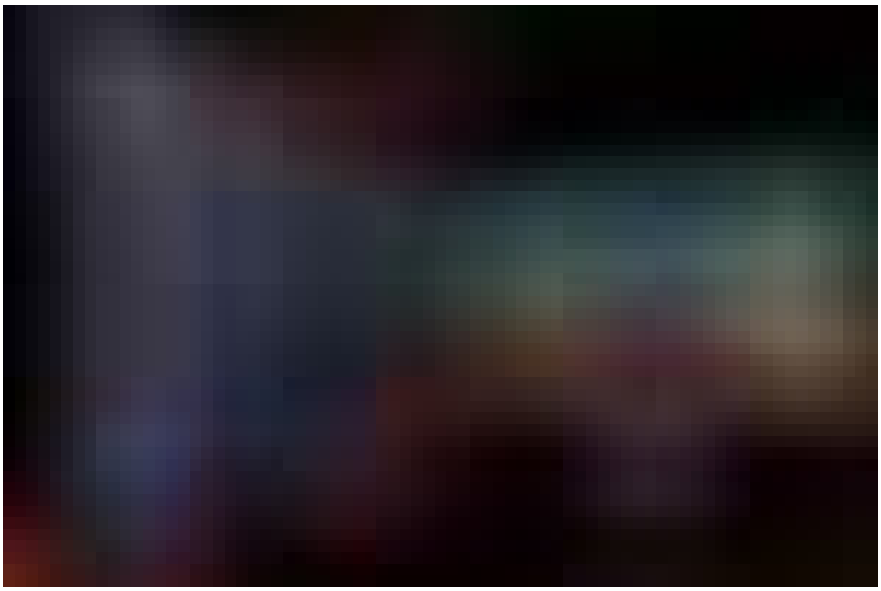


Photo by Fotis Fotopoulos on Unsplash

Kubernetes: Inbuilt tools have been included in Kubernetes for the purpose of Logging and Monitoring. Logging helps to understand the cause of failures through the analysis of past records/logs. On the other hand, monitoring enables us to be constantly aware of the health of the nodes and the services that are containerized by them.

Docker Swarm: Swarm lacks the capability of inbuilt tools to handle Logging and Monitoring. Though, you can make use of third-party tools to achieve this. The tool ELK is one such example. On the other hand, tools like Reimann can be used for monitoring.

4. Scalability

Kubernetes: Kubernetes is known to be slightly better than Swarm when it comes to maintaining the strength of the clusters. Though it is

not as scalable as Swarm owing to the complexity involved. A unified set of APIs is what adds complexity to Kubernetes, along with a strong focus on the cluster state.

Though when it comes to Auto-scaling, Kubernetes is way ahead in the race since it can analyze the server load and scale up or down in tandem with your requirements. This makes traffic handling a breeze.

Docker Swarm: Swarm is known to be more scalable than Kubernetes. Containers can be deployed faster both when it comes to large clusters and high cluster fill stages. Only a single update command is enough to deploy new replicas.

On the other hand, Auto-scaling is as smooth with Swarm as it is with Kubernetes, but AWS autoscaling is possible with the help of StackStorm. The idea is to create additional worker nodes when when a cluster fills up on load, and then add them to the cluster.

5. GUI

Kubernetes: If you are a stickler for Dashboards, Kubernetes is your end-game. The GUI provided is a reliable dashboard which can be used to effortlessly control the cluster. This can be a huge boon for you if you are not from a technical background since it requires no technical efforts and the instructions are in plain English.

Docker Swarm: Docker hosts and Swarm Clusters can be managed with the help of third party tool such as Portainer.io that provide an easy management UI. On the other hand, the Universal Control Pane in the Enterprise edition of Docker provides you an interface to manage the clusters.

6. Load Balancing

Kubernetes: Load Balancing is permitted in Kubernetes when Container Pods are defined as Services. Apart from this, you need to manually configure the load balancing settings. Only a certain set of pods and policies give access to each service.

Docker Swarm: Docker Swarm provides an inbuilt facility of Load Balancing. A common network is joined by all containers that are in the same cluster. This allows connection of any node to any container.

Comparison Table

Pros and Cons

Pros of Kubernetes

● Open source and modular solution

● Ability to run on any sort of operating system

● Easy organization of service with pods

● Developed by Google, who bring years of valuable industry experience to the table

Cons of Kubernetes

● The installation and configuration can prove to be too complex for first timers

● Not compatible with any existing Docker CLI and Compose tools

Pros of Docker Swarm

- Efficient and easier initial set up
- Integrates and works with existing Docker tools
- Lightweight installation
- Open source solution

Cons of Docker Swarm

- Limited functionality as per the availability in the Docker API
- Limited fault tolerance

Final Verdict—Kubernetes or Docker Swarm?

Kubernetes is a superior and stronger market competitor today.

Some of the decisions taken by the management at Docker have resulted in a serious dent on its use cases for individual developers. Demands of the open source community have not been fulfilled and the best features being offered in a stand-alone Enterprise Edition have only added to the mayhem. The support from the community has declined as a result. On the other hand, Kubernetes has risen to one of the most popular Open Source projects. With a thriving community support, it is constantly evolving and improving at a high pace.

At the same time, Kubernetes is technically superior to Docker Swarm. Better health checking provisions at the pod level, better container scheduling and scaling, Namespaces with role-based access controls, and network policies to control ingress and egress traffic between services give it an edge. Even cloud providers are opting to shift to Kubernetes managed services.

Conclusion

I hope that all this information would help you to reach a final solution about the best containerization tool for business.

Which containerization service/tool do you think is the best for you and why?