31/07/2022

# HOW TO CHOOSE A CONTAINER ORCHESTRATION TOOL ?


VS


**PREPARED BY :**

**Katia AG**

# Introduction :

Businesses around the world increasingly rely on the benefits of container technology to ease the burden of deploying and managing complex applications. Containers group all necessary dependencies within one package. They are portable, fast, secure, scalable, and easy to manage, making them the primary choice over traditional VMs. But to scale containers, you need a container orchestration tool—a framework for managing multiple containers.

# Problems Container Orchestration Solves:

Containers ***can't surface a scalable, reliable, and fault-tolerant service by itself.*** Docker only knows that the lifecycle of its container starts and ends with the given process. So containerization platforms like Docker need some additional tool or service to coordinate and command these containers.So here enter the container orchestration platforms. Container orchestration tools are used :

- ***Automate and manage*** tasks like ***provisioning, deployment, configuration.***
- ***Scheduling***, container availability, load balancing, traffic routing, resource allocation, container health monitoring.
- ***Securing*** interaction between containers, and scaling or removing containers for balancing workloads across the infrastructure.

# What is Kubernetes?

kubernetes

Kubernetes is an open-source container orchestration platform. These platforms allow the automation of containerization processes, such as deploying, managing containers, and scaling containerized applications. Kubernetes, which can also be named "Kube" or k8s, was initially developed by Google in 2014. Currently, the platform is maintained by the Cloud Native Computing Foundation (CNCF), and it is written in Go.

# Advantages of Kubernetes:

Kubernetes offers a wide range of benefits to teams looking for a robust container orchestration tool:

- It has a *large open-source community*, and Google backs it.
- It supports every *operating system.*
- It can sustain and *manage* large architectures and *complex workloads*.
- Key functionalities consist of ingress, load balancing, service discovery, horizontal scalability, storage orchestration, self-healing, batch executions and automated rollouts and rollbacks.
- It has *built-in monitoring* and a wide range of available integrations.
- Availability on both on-premise and public cloud from cloud providers like Google Cloud Platform, AWS, Microsoft Azure, and IBM Cloud
- The tool works smoothly on every OS.
- The tool has a unified set of APIs and strong guarantees about the cluster state.
- A wide array of orchestration and automation features.
- A range of integrations and third-party tools.
- A large active community that continuously ships new features and integrations.
- Full support of the CNCF (Cloud Native Computing Foundation).

# Disadvantages of Kubernetes:

Despite its comprehensive feature set, Kubernetes also has a few drawbacks:

- It has a complex installation process and a steep learning curve.
- It requires you to install separate CLI tools and learn each of them.
- The transition from Docker Swarm to Kubernetes can be complicated and difficult to manage
- Quick updates from the open-source community require careful patching to avoid workload disruption.
- Mostly requires add-on tools, services, CI/CD workflows, and other DevOps practices to handle access, identity, security and governance.

# What is Docker Swarm


Docker Swarm is also a container orchestration tool. It is native to the Docker Platform, and was created to ensure applications can run seamlessly across various nodes that share the same containers. Thus, Swarm allows developers or DevOps engineers to efficiently deploy, manage, and scale clusters of nodes on Docker. The Docker platform is also written in Go.

# Advantages of Docker Swarm :

- Docker Swarm is easy to install and set up.
- A smooth learning curve makes the tool an excellent choice for beginners in container orchestration.
- The tool has automated load balancing within Docker containers.
- Docker Swarm uses the same command line interface (CLI) as Docker Engine.
- The tool does not require additional libraries or components if your system is already running inside Docker.
- Like Kubernetes, Docker Swarm can run on any OS.
- The tool works perfectly with all existing Docker products.
- Ideal for smaller and less complex systems with infrequent deployments.

# Disadvantages of Docker Swarm :

- Limited functionality due to the tie-in with Docker's API.
- Automation capabilities are not nearly as robust as those offered by K8s.
- No simple way to split Dev-Test-Prod workloads in a DevOps pipeline.
- Limited options in terms of customization.
- A smaller community compared to Kubernetes.
- Mostly relies on manual scalability.
- The tool's future is somewhat in question after the Mirantis acquisition (many users are already moving to new tools expecting troubled times ahead for Swarm)

# Kubernetes vs Docker Swarm: A Head-to-Head Comparison -1-

| Point of comparison | Kubernetes | Docker Swarm |
|---|---|---|
| Main selling point | A complete container orchestration solution with advanced automation features and high customization | An emphasis on ease of use and a seamless fit with other Docker products |
| Installation | Somewhat complex as you need to install (and learn to use) kubectl | Quick and easy setup (if you already run Docker) |
| Learning curve | High learning curve (but has more features) | Lightweight and easy to use (but limited functionality) |
| GUI | Detailed native dashboards | No out-of-the-box dashboards (but you can integrate a third-party tool) |
| Cluster setup | Difficult to start a cluster (but the cluster is very strong once set up) | Easy to start a cluster |
| Availability features | Self-healing, intelligent scheduling, and replication features | Availability controls and service duplication |
| Scalability | All-in-one scaling based on traffic | Values scaling quickly (approx. 5x faster than K8s) over scaling automatically |

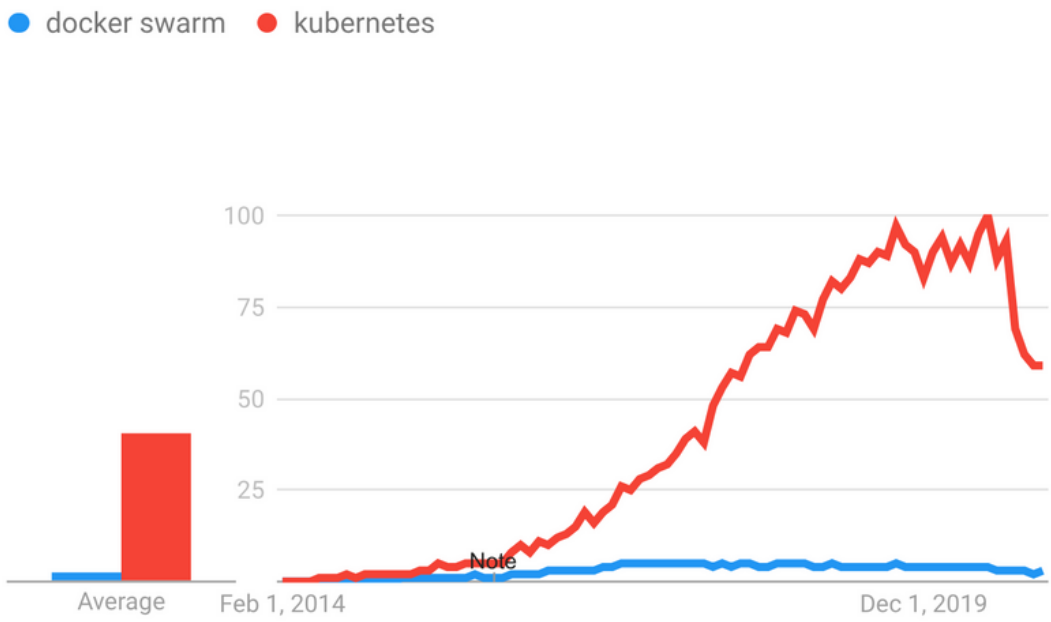
# Kubernetes vs Docker Swarm: A Head-to-Head Comparison -2-

| Point of comparison | Kubernetes | Docker Swarm |
|---|---|---|
| Horizontal auto-scaling | Yes | No |
| Monitoring capabilities | Has built-in monitoring and logging | Basic server log and event tools, but needs a third-party tool for advanced monitoring |
| Load balancing | No built-in mechanism for auto load-balancing | Internal load balancing |
| Security features | Relies on transport layer security (TLS) and access control-related tasks | Supports multiple security protocols (RBAC authorization, TLS/SSL, secrets management, policies, etc.) |
| CLI | Needs a separate CLI | Integrated Docker CLI, which can limit the functionality in some use cases |
| Community | Huge and active community | Reasonably popular, but the user base is getting smaller since the Mirantis acquisition |
| Optimal use case | High-demand apps with a complex configuration | Simple apps that are quick to deploy and easy to manage |

# Kubernetes vs Docker Swarm: Which Tools Should Your Team Use?

Because of its broad community support and ability to handle even the most complex deployment scenarios, Kubernetes is often the number one choice for enterprise development teams managing microservice-based applications.

Regarding popularity, Kubernetes has a clear advantage, as we can observe according to the Google Trends chart. Plus, by looking at Github, we can conclude that while Kubernetes has 81.1k stars, Docker Swarm only has 5.8k stars. That's a big difference and does not leave much space for doubt. Kubernetes is definitely a more popular solution than Docker Swarm when it comes to container orchestration technologies.




As we can see, both Docker Swarm and Kubernetes were created to fulfill the same purpose. Keep reading to find how they differ and which one to choose.

- For beginners, **Docker Swarm** is an ***easy-to-use and simple solution*** to manage your containers at scale. If your company is moving to the container world and does not have complex workloads to manage, then Docker Swarm is the right choice.
- If your applications are tricky, and you are looking for a ***complete package*** that includes ***monitoring, security features, self-healing, high availability, and absolute flexibility***, then **Kubernetes** is the right choice.
- If you need all the capabilities of Kubernetes but are put off by its learning curve, then K3s is a good alternative.