An Introduction to two Popular Container Orchestration Platforms: Docker Swarm & K8s

Diego Leon, Nicolai Hellesnes

May 30, 2020

Contents

1 Introduction

The use of containers is increasing and with that comes challenges. A big contribution to the increased use of containers is that it solves the problems of reliability and consistency [1]. With containers a piece of software can run on any computer without there being a difference in the environment for the software. This means that the code will be in the same environment from the beginning all the way to deployment.

With the environment being the same all the way from production to deployment, no development time will be spent on trying to solve weird bugs that stems from the developers having different environments or different versions of a certain dependency.

The increased use of containers brings new challenges to developers. With containers being a relatively new technology, it is hard to find people with the right expertise in the subject, which can be a problem if a issue arises during development [2]. Another challenge is security. Containers are a complex and relatively new technology, which makes keeping it secure harder [2].

It can be trivial to manage a few containers but larger applications can have hundreds of containers spread across multiple hosts [3]. Management suddenly gets much more complicated. Container orchestration is important because it helps the handling of containers. It can also help automate deployment, management, scaling and networking of containers [4]. While it is not very important to have an orchestration tool when deploying one or two containers at a time, it can make the deployment much easier and faster when you start reaching container deployment in the three-digits.

Deploying and managing software is a big part of DevOps, these are two tasks that a container orchestration tool can help with by automating the them.

Docker Swarm and K8s are two popular orchestration platforms [5]. The purpose is therefore to give the reader a comprehensive introduction to both platforms.

2 Background

Docker is a container application which software can be run inside. There are many popular technologies running Docker, including but not limited to NGINX, REDIS, and POSTGRESQL [6]. However, a problem that needs to be solved is how to manage containers when you have a lot of them. One of the solutions is to use a container orchestration tool.

There are different ways of managing Docker containers. Using a orchestration tool is the most common way of managing and orchestrating Docker containers in production and for deployment. The two most popular orchestration tools are Kubernetes and Docker Swarm, with Kubernetes being the most popular of them all according to a survey conducted by redmonk [7].

There are many companies that use Kubernetes, with a few of them being Apple, Dell, Google, and Cisco [8]. With the computer software being the industry that uses it the most [9]. Docker swarm is also used by many companies, and it is also in the computer software industry where it sees the most use [10].

2.1 Containers

Containers are software that package all the code and dependencies needed for an application. A container allows one or a group of processes to run in a loosely isolated environment. The environment is loosely isolated because it uses the same kernel as the host machine. The container will only be able to see the processes that it is running, while the host machine may be able to see the processes that the container is running [11].

One big advantage of containers is that it allows applications to run in different environments without needing to change the environment that the container is run in before the application can run. This eliminates the common problem in software development where a piece of software can run on a developer's machine environment but inexplicably does not run on another developer's machine [12].

The process for creating a container with Docker is as follows. You start with creating a dockerfile, this file states everything that is needed for the container. The file then gets built which results in an image. When this image is executed a container is created. This process makes containers flexible to work with and makes deployment incredible fast. Under the hood containers basically boils down to clever uses of namespaces and cgroups, which is what makes them lightweight [13].

Dealing with containers include some important tasks such as: balancing the load, connecting them to the outside world, and managing them. A tool is needed to integrate and orchestrate the containers, make them fault tolerant, scale up and down depending on the circumstances, and provide communication

across a cluster. A container orchestration tool like K8s and Docker Swarm is therefore needed.

2.2 Orchestration

To orchestrate is defined as, "to arrange something carefully, and sometimes unfairly, so as to achieve a wanted result" [14]. Similarly, a orchestration tool is used to automate a process or workflow across many systems. These different tasks can be configuration, management, and coordination [15]. An orchestration tool is used when managing and configuring manually becomes too big of a task.

2.3 Nodes and Clustering

In this context a node is a machine, which could be physical or virtual, that is running Docker engine [16]. A cluster is several these nodes joined together.

3 Docker Swarm

3.1 Docker Introduction

Docker Swarm is a tool for handling containers. It solves the problem of having to handle multiple containers individually. With Docker Swarm a group of containers can be handled as if the were one container.

When using Docker Swarm you have two options, you can use the standalone Docker Swarm, or the Docker Swarm mode which is integrated into the Docker engine. Docker Swarm is seen as the legacy option while Docker swarm mode is the recommended option. This introduction to Docker Swarm will focus on the Docker Swarm mode as the standalone Docker Swarm functionality is seen as a legacy option [17].

In Docker terminology a swarm is a cluster of Docker engines. That is to say that cluster and swarm is the same thing in Docker terminology. A cluster is a group of computers or resources that work together and can be seen as one single unit [18]. Docker swarm mode is the cluster management and orchestration features built into the Docker engine. Cluster management and orchestration features means the features that let you handle a cluster of containers without needing to give commands to each of the containers in the cluster individually. These features are built into the Docker engine using swarmkit [19]. Figure 1 shows the general structure for when a cluster management and orchestration tool is used. With a cluster management and orchestration tool like Docker Swarm mode and Kubernetes things like scaling and load balancing can be automated.

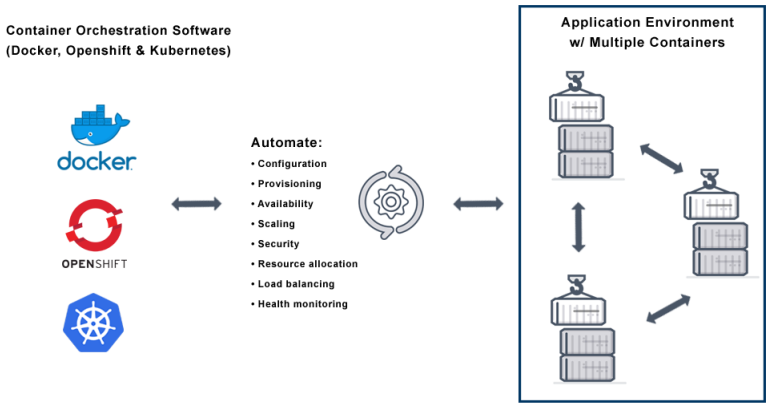

Figure 1: Container orchestration Swarm [20]

3.2 Roles and Nodes

There are a few different concepts that have to be understood to understand Docker swarm. Let's start with roles. There are two different roles in a swarm, manager and worker. The managers manage the membership to a swarm and divide work in the cluster. The role of a worker is to execute the work they have been given.

A node is the term for an instance of a Docker engine that is a part of a Docker swarm. One computer can have multiple nodes that are part of a Docker swarm. A node can either perform both or one of manager and worker roles [21]. Figure 2 shows how the manager-worker relationship looks. Instructions to a manager are given via a CLI, the manager will then give out the instructions to the workers.

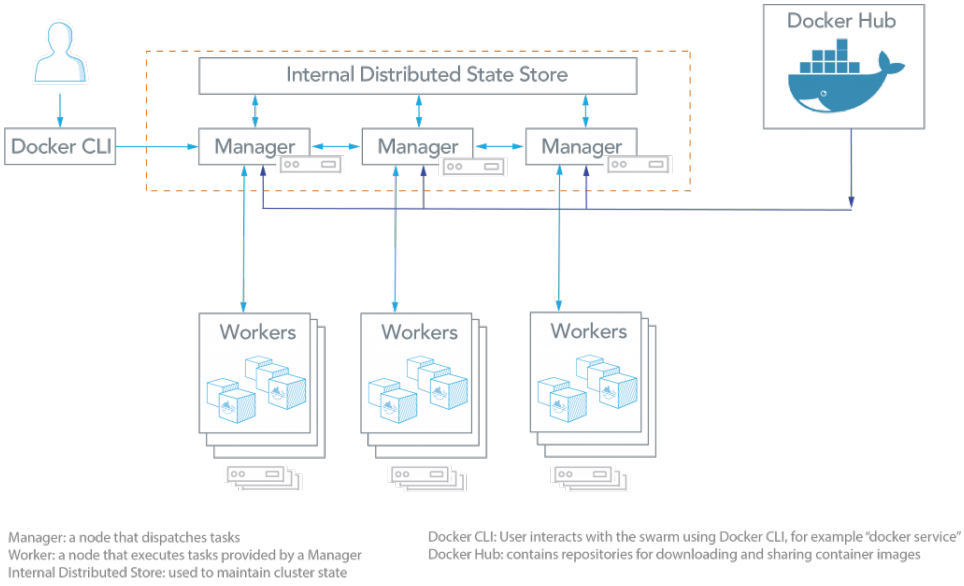

Figure 2: The structure of a Docker Swarm [22]

3.3 Docker Service

In Docker terminology a service is the main structure of a swarm. The service defines what tasks should be executed. When a service is created, you decide what image should be used and which command should be executed [23].

A replicated service model is where a Docker Swarm has replicas of a Swarm running, one of the main advantages of this kind of model is that there is not a single point of failure. When you are using a so called replicated service model, the desired number of replicas of a task will be distributed in the swarm. The desired number of replicas would have been needed to be set in the desired

state. There also exists something called global services. If a global service is used each available node will each get one task [24].

You can by hand define the optimal state for a swarm. The optimal state can contain things like the number of replicas you want, and the network resources that should be available. Docker will always try to be in this optimal state. This means that if a node becomes unavailable, Docker will try to schedule the task of the node onto another node [25].

A container that is running and is part of a swarm service is called a task. The swarm mode uses a decentralized design, this means that the handling of roles are done at runtime and not at deployment time. It is thus possible to build a swarm by using just a single disk image. When it becomes time to deploy an application with a swarm, a service definition should be delivered to a manager node, that manager node will divide tasks to the worker nodes [26].

3.4 Docker Host

A Docker host is a physical or virtual machine that runs Docker. For a swarm to exist there needs to be multiple Docker hosts that are executed in swarm mode. These hosts take the roles as manager and divide up the workload [27].

To scale a docker swarm the Docker CLI can be used. With the Docker CLI the number of containers for a service can be adjusted. This makes scaling a Docker swarm highly flexible [28].

3.5 Load Balancing

Load balancing is an important part for scaling a service. With load balancing a service can divide the load over different parts of the service to be able to handle the current load without needing to add additional nodes. In a Docker swarm a swarm manager makes use of ingress load balancing. This ensures high availability even during high loads. A Docker swarm also has an internal DNS component, this component can automatically assign the services in the swarm a DNS entry. The Docker swarm manager makes use of internal load balancing to divide the requests among the different services of the cluster [29].

3.6 Installation

To use Docker Swarm mode docker has to be installed. After that the Docker CLI can be used to create, deploy, and manage a swarm [30]. Only a few commands are needed to create a swarm in Docker [31].

4 K8s

Kubernetes is an open-source container orchestration platform originally designed by Google, and is now maintained by the Cloud Native Computing Foundation [32]. It gives engineers a centralized system for container orchestration.

4.1 K8s cluster

A K8s cluster is a central component and consists of several machines that could either serve as a master or as a worker node. The K8s master, also called the control plane, has an important role and serves as an access point which enables interactions between the admin (or other users) and the cluster, with the purpose of scheduling and deploying containers [33][34].

The idea behind Kubernetes is to enforce the "Desired state management". Which means that we are going to feed the master node a specific configuration and it will be up to the API to go out and run that configuration. It does this by informing all the nodes about where to place the application instances and how many to run.

The master node consists of following components: etcd, kub-apiserver, kube-controller-manager, cloud-controller-manager, cloud-controller-manager, kube-scheduler.

4.2 Etcd

The ectd is a persistent key-value data store where the master stores the state and configuration data for the whole cluster. All the nodes in the K8s cluster have access to the ectd and use it to manage the containers they are currently running.

4.3 Kube-apiserver

The kube-apiserver is the front end for the K8s API server and is utilized by the master to communicate with the rest of the cluster. The kube-apiserver is responsible for making sure that the configurations of the deployed containers match the configurations in etcd.

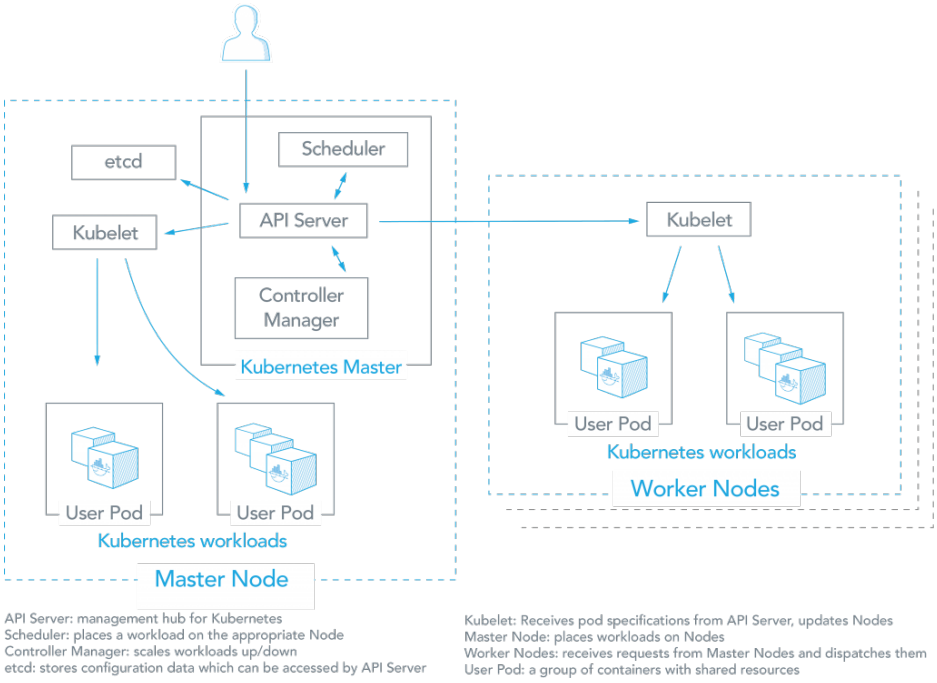

Figure 3: The structure of K8s [35]

4.4 Kube-controller-manager

The Controller manager is responsible for handling the K8s control loops. These loops manage the state of the cluster through the API server. This service is important and takes care of replicas, node management, and deployments. Part of the service includes monitoring the health of nodes.

4.5 Kube-scheduler

The kube scheduler receding inside the master worker is responsible for tracking the node workloads in the cluster and giving work to each node depending on its ability. It looks for new pods that do not have any nodes and assigns them a node based on the requirements.

A container is deployed to all nodes in the K8s cluster. The application runs inside the containers. This is managed by a container runtime (such as Docker) which all the nodes in the cluster are configured with.

4.6 Kubelet

All K8s nodes have a process called kubelet. This process is responsible for handling the state of the node, which is set based on instructions from the control

plane. Some of the states are: stopping, starting, and maintaining containers. It saves the health and performance from the container, node, and pods. These kubelets gather data from the node, pods and containers it shares the node with, and sends it to the control plane which makes scheduling decisions based on this information.

4.7 Pod

A pod is a scheduling unit that contains one or more containers, and information about how to run them. These containers are guaranteed to be co-located on the host machine [36]. The so-called, desired state, is described in a pod through a K8s object called yaml. These objects are handled by the API server. To define the pods and the number of container instances (replicas) for each pod, a YAML object called "deployment" has to be created.

4.8 Installation

Installing and setting up a K8s cluster can be complex and take a lot of planning. It requires some manual configurations to make all components work together. The installation differs depending on the OS and the provider [37].

5 Comparison

Container orchestration platforms are critical when operating on scale. Selecting the right one for your needs can be hugely important. Both of the platforms are highly scalable. In Kubernetes scaling is accomplished by changing the number of replicas in a deployment, while with Docker Swarm scaling can be accomplished with the CLI.

When containerizing an application using Docker, both are viable options as both Docker swarm and Kubernetes work with Docker containers.

One advantage of Kubernetes is that it offers more functionality than Docker swarm. On the other hand Docker swarm is much more lightweight than Kubernetes while also being much easier to deploy [38]. The complexity of setting up Kubernetes is in most cases not an issue when it comes to cloud deployment. Nowadays, most providers have offers that take away a big portion of the required setup [39].

Kubernetes is more of a all-in-one framework and complicated as it provides strong guarantees when it comes to cluster state and more. Because of this, container scaling and deployment become slower. Docker Swarm is able to deploy containers faster than kubernetes. A single update command is needed to start new replicas [37].

Installing Docker Swarm is easier than installing K8s. Docker swarm being built into Docker offers a huge advantage as no third party application is needed. Docker swarms might be the best choice for individuals or small developer teams that do not deploy often, while Kubernetes is a great choice for all but the smallest and simplest of workloads [40].

6 Conclusion

K8s is more suitable for complex applications while Docker swarm provides a simple solution with fast deployment. Docker swarm is therefore a good choice when a quick setup is needed without exhaustive configuration.

If being lightweight & flexible is important, Docker Swarm is the choice to make. Docker Swarm is also easier to deploy, which makes it a good choice if fast deployment is important. While Kubernetes is more complex to deploy, it also offers more functionality than Docker Swarm does. This makes Kubernetes a good choice if you are looking for a more all-in-one framework.

References

[1] CIO. *What are containers and why do you need them?*. URL: https://www.cio.com/article/2924995/what-are-containers-and-why-do-you-need-them.html (visited on 29/05/2020)

[2] Tozzi, Christopher. *4 Reasons Container Adoption Remains Slowish*. URL: https://containerjournal.com/topics/container-ecosystems/4-reasons-container-adoption-remains-slowish/ (visited on 29/05/2020)

[3] Raghuram, Sirish. *4 reasons you should use Kubernetes*. URL: https://www.infoworld.com/article/3173266/4-reasons-you-should-use-kubernetes.html (visited on 29/05/2020)

[4] Red Hat. *What is container orchestration?* URL: https://www.redhat.com/en/topics/containers/what-is-container-orchestration (visited on 29/05/2020)

[5] Conway, Sarah. *Survey Shows Kubernetes Leading as Orchestration Platform*. URL: https://www.cncf.io/blog/2017/06/28/survey-shows-kubernetes-leading-orchestration-platform/ (visited on 29/05/2020)

[6] Level Up Education. *Interesting facts — Companies and the use of Docker*. URL: https://medium.com/@tao_66792/interesting-facts-companies-and-the-use-of-docker-948baa8cf309 (visited 29/05/2020)

[7] Fintan, Ryan. *Containers in Production – Is Security a Barrier? A Dataset from Anchore*. URL: https://redmonk.com/fryan/2016/12/01/containers-in-production-is-security-a-barrier-a-dataset-from-anchore (visited 29/05/2020)

[8] Hg Insights. *Companies Currently Using Kubernetes*. URL: https://discovery.hgdata.com/product/kubernetes (visited 29/05/2020)

[9] Enlyft. *Companies using Kubernetes*. URL: https://enlyft.com/tech/products/kubernetes (visited 29/05/2020)

[10] Enlyft. *Companies using Docker Swarm*. URL: https://enlyft.com/tech/products/docker-swarm (visited 29/05/2020)

[11] IBM. *Containers*. URL: https://www.ibm.com/cloud/learn/containers (visited on 29/05/2020)

[12] Google. *Containers 101: What are containers?*. URL: https://cloud.google.com/containers (visited on 29/05/2020)

[13] Evans, Julia. *What even is a container: namespaces and cgroups*. URL: https://jvns.ca/blog/2016/10/10/what-even-is-a-container/ (visited on 29/05/2020)

[14] Cambridge Dictionary. *orchestrate*. URL: https://dictionary.cambridge.org/dictionary/english/orchestrate

[15] Red Hat. *What is orchestration?*. URL: https://www.redhat.com/en/topics/automation/what-is-orchestration (visited on 29/05/2020)

[16] Docker. *How nodes work*. URL: https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/ (visited on 29/05/2020)

[17] Docker. *Swarm mode overview*. URL: https://docs.docker.com/swarm/overview/ (visited on 29/05/2020)

[18] IBM. *Clustering: A basic 101 tutorial*. URL: https://www.ibm.com/developerworks/aix/tutorials/clustering/clustering.html (visited on 29/05/2020)

[19] SwarmKit. URL: https://github.com/docker/swarmkit/ (visited on 29/05/2020)

[20] AVI Networks. *Container Orchestration*. URL: https://avinetworks.com/glossary/container-orchestration/ (visited on 29/05/2020)

[21] Docker. *Swarm mode key concepts*. URL: https://docs.docker.com/engine/swarm/key-concepts/ (visited on 29/05/2020)

[22] Binani, Harsh. *Kubernetes vs Docker Swarm—A Comprehensive Comparison*. URL: https://hackernoon.com/kubernetes-vs-docker-swarm-a-comprehensive-comparison-73058543771e (visited on 29/05/2020)

[23] Docker. *Services and tasks*. URL: https://docs.docker.com/engine/swarm/key-concepts/#services-and-tasks (visited on 29/05/2020)

[24] Docker. *Deploy services to a swarm*. URL: https://docs.docker.com/engine/swarm/services/#replicated-or-global-services (visited on 29/05/2020)

[25] Docker. *Swarm mode key concepts*. URL: https://docs.docker.com/engine/swarm/key-concepts/ (visited on 29/05/2020)

[26] Docker. *Swarm mode overview*. URL: https://docs.docker.com/engine/swarm/ (visited on 29/05/2020)

[27] Tabib, Raziel. *Docker Machine*. URL: https://codefresh.io/docker-tutorial/docker-machine-basics (visited on 29/05/2020)

[28] Docker. *Scale the service in the swarm*. URL: https://docs.docker.com/engine/swarm/swarm-tutorial/scale-service/ (visited on 29/05/2020)

[29] Docker. *Load balancing*. URL: https://docs.docker.com/engine/swarm/key-concepts/#load-balancing (visited on 29/05/2020)

[30] Docker. *Swarm mode overview*. URL: https://docs.docker.com/engine/swarm/ (visited on 29/05/2020)

[31] Docker. *Create a swarm*. URL: https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/ (visited on 29/05/2020)

[32] Kubernetes. *Kubernetes*. URL: *https://kubernetes.io/* (visited on 29/05/2020)

[33] Gerrard, Ali. *What Is Kubernetes? An Introduction to the Wildly Popular Container Orchestration Platform*. URL: https://blog.newrelic.com/engineering/what-is-kubernetes/ (visited on 29/05/2020)

[34] linode. *A Beginner's Guide to Kubernetes, Part 2: Master, Nodes, and the Control Plane*. URL: https://www.linode.com/docs/kubernetes/beginners-guide-to-kubernetes-part-2-master-nodes-control-plane/ (visited on 29/05/2020)

[35] PlatForm9. *Deploy Kubernetes: The Ultimate Guide*. URL: https://platform9.com/docs/deploy-kubernetes-the-ultimate-guide/ (visited on 29/05/2020)

[36] Kubernetes. *What is a Pod?*. URL: https://kubernetes.io/docs/concepts/workloads/pods/pod/ (visited on 29/05/2020)

[37] UpCloud. *Docker Swarm vs. Kubernetes: Comparison of the Two Giants in Container Orchestration*. URL: https://upcloud.com/community/stories/docker-swarm-vs-kubernetes-comparison-of-the-two-giants-in-container-orchestration/ (visited on 29/05/2020)

[38] IBM. *Docker Swarm vs. Kubernetes: Which is the right container orchestrator for you?*. URL: https://www.ibm.com/cloud/blog/new-builders/docker-swarm-vs-kubernetes-a-comparison (visited on 29/05/2020)

[39] Rael, Dave. *Docker Swarm vs Kubernetes: A Helpful Guide for Picking One*. URL: https://stackify.com/docker-swarm-vs-kubernetes-a-helpful-guide-for-picking-one/ (visited on 29/05/2020)

[40] Stackify. *Docker Swarm vs Kubernetes: A Helpful Guide for Picking One*. URL https://stackify.com/docker-swarm-vs-kubernetes-a-helpful-guide-for-picking-one (visited on 29/05/2020)