What is Kubernetes?

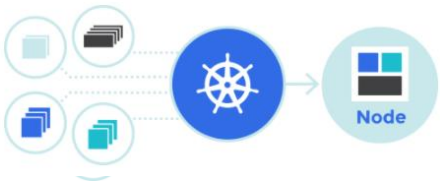
What is Kubernetes?

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes is also referred to as k8s, as there are 8 characters between k and s.

- Born in Google
- Donated to CNCF in 2014
- Open source (Apache 2.0)
- v1.0 July 2015
- Written in Go/Golang
- Often shortened to k8s

Kubernetes
K8 s

Tips:
Kubernetes comes from the Greek word κυβερνήτης, which means helmsman or ship captain. With this analogy in mind, we can think of Kubernetes as the captain on a ship of containers.



> **Q**: What is Kubernetes?
> **A**: Kubernetes is an open-source container orchestration tool or system that is used to automate tasks such as the management, monitoring, scaling, and deployment of containerized applications. It is used to easily manage several containers (since it can handle grouping of containers), which provides for logical units that can be discovered and managed.
>
> **- Interview Q&A**

Why you need Kubernetes and what it can do

Containers are a perfect way to get the applications packaged and run. In a production environment, you should manage the containers that run the applications and ensure no downtime. For example, if a container goes down, you need to start another container. Wouldn't it be perfect if a program could control this behavior?

That's how Kubernetes works! Kubernetes gives you a framework to resiliently run distributed systems. For your application, it takes care of scaling and failover, provides deployment patterns, and more.

Kubernetes supplies you with:

- **Service discovery and load balancing**

Kubernetes can expose a container using the DNS name or using its own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.

- **Storage orchestration**

Kubernetes allows you to automatically mount a storage system of your choice, such as local storage, public cloud providers, and more.

- **Automated rollouts and rollbacks**

You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers, and adopt all their resources to the new container.

- **Automatic bin packing**

You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.

- **Self-healing**

Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

- **Secret and configuration management**

Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

Cloud Providers

Many cloud services offer a Kubernetes-based platform or infrastructure as a service on which Kubernetes can be deployed as a platform-providing service.

- AWS
- Azure
- CloudStack
- GCE
- HUAWEI CLOUD
- OpenStack
- OVirt
- Photon
- vSphere
- IBM Cloud Kubernetes Service
- Baidu Cloud Container Engine
- Tencent Kubernetes Engine
- Alibaba Cloud Kubernetes

The differences of Kubernetes and Docker Swarm

Differences	Kubernetes	Docker Swarm
Application Deployment	Applications can be deployed in Kubernetes using a combination of microservices, deployments, and pods.	The applications can be deployed as micro-services in a swarm cluster. YAML files can be used to identify multi-containers. In addition, the application can be installed with Docker compose.
Scalability	For distributed systems, Kubernetes acts like more of an all-in-one framework. It is a complex system because it provides strong guarantees in terms of the cluster state and a unified set of APIs. Therefore the scaling and deployment of containers are slowed down.	Docker Swarm can deploy containers much faster than Kubernetes and this allows faster reaction times to scale on demand.
Networking	Kubernetes has a flat network model, allowing all pods to interact with one another. The network policies specify how pods interact with one another. The model needs two CIDRs for the services and the pods.	When a node joins a swarm cluster, it generates an overlay network for services that span every host in the docker swarm. This also provides a host-only docker bridge network for containers. The users have a choice to encrypt container data traffic to create its own overlay network.

Availability	Kubernetes provides considerably high availability by absorbing the failure of application as it distributes all the pods between the nodes. Load balancing services in Kubernetes detect unhealthy pods and subsequently deactivate them.	Docker Swarm also offers high availability architecture as the services can be replicated in Swarm nodes. The Swarm manager nodes are responsible for managing the worker's node resources and the whole cluster.
Container Setup	Kubernetes differs from other standard docker equivalents by utilizing its own YAML, API, and client definitions. Thus, containers can not be defined with Docker Compose or Docker CLI. While switching platforms, YAML definitions, and commands must be rewritten.	The Docker Swarm API offers much of the familiar functionality from Docker. It supports most of the tools that run with Docker. However, if the Docker API is deficient in a particular operation, Swarm can not be used.
Load Balancing	In Kubernetes, pods are exposed via service, which can be implemented as a load balancer inside a cluster. An ingress is generally used for load balancing.	Swarm mode consists of a DNS element that can be used for distributing incoming requests to a service name. Thus Services can be assigned automatically or can run on ports that are pre-specified by the user.

Kubernetes	Docker Swarm
Auto Scaling	No Auto Scaling
Great Active Community	Good Community
Difficult to start a cluster	Easy to Start a Cluster

> **Q**: What are the main differences between the Docker Swarm and Kubernetes?
> **A**: Docker Swarm is Docker's native, open-source container orchestration platform that is used to cluster and schedule Docker containers. Swarm differs from Kubernetes in the following ways:
> Docker Swarm is more convenient to set up but doesn't have a robust cluster, while Kubernetes is more complicated to set up but the benefit of having the assurance of a robust cluster
> Docker Swarm can't do auto-scaling (as can Kubernetes); however, Docker scaling is five times faster than Kubernetes
> Docker Swarm doesn't have a GUI; Kubernetes has a GUI in the form of a dashboard
> Docker Swarm does automatic load balancing of traffic between containers in a cluster, while Kubernetes requires manual intervention for load balancing such traffic
> Docker requires third-party tools like ELK stack for logging and monitoring, while Kubernetes has integrated tools for the same
> Docker Swarm can share storage volumes with any container easily, while Kubernetes can only share storage volumes with containers in the same pod
> Docker can deploy rolling updates but can't deploy automatic rollbacks; Kubernetes can deploy rolling updates as well as automatic rollbacks
>
> **- Interview Q&A**