Become a Certified Professional→

What Is Kubernetes? An Introduction To Container Orchestration Tool

Last updated on Sep 14,2020 *25.2K Views*

Vardhan
Vardhan is a technology enthusiast working as a Sr. Research Analyst at...

We all know how important Containers have become in today's fast-moving IT world. Pretty much every big organization has moved out of their traditional approach of using virtual machines and started using Containers for deployment. They are looking for ***trained Kubernetes professionals*** who have in-depth knowledge about containerization and orchestration tools. So, it's high time you understand what is Kubernetes. Following are the topics covered in this blog:

If you want to read more about the advantages of Containers and how companies are reshaping their deployment architecture with Docker, then click here. Or else continue reading this blog about Kubernetes.

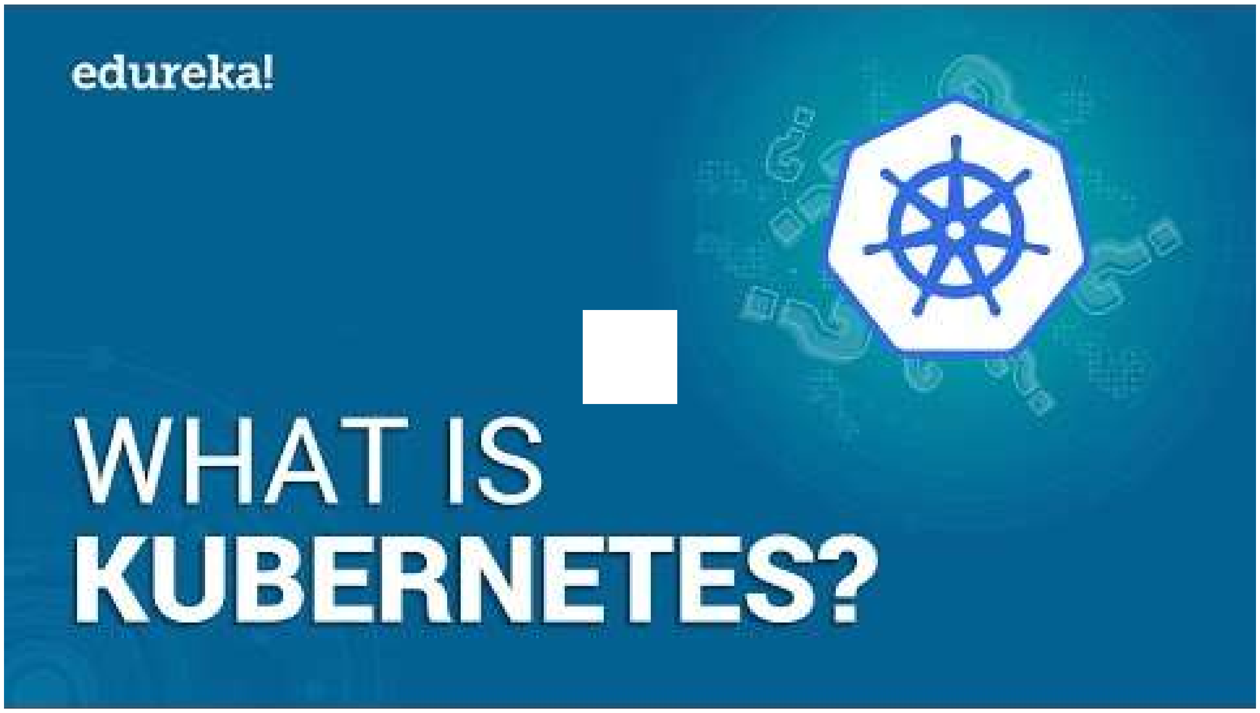
What Is Kubernetes?

Kubernetes is an open-source container management (orchestration) tool. It's container management responsibilities include container deployment, scaling & descaling of containers & container load balancing.

Note*: Kubernetes is not a containerization platform. It is a multi-container management solution.*

Going by the definition, you might feel Kubernetes is very ordinary and unimportant. But trust me, this world needs Kubernetes for managing containers, as much as it needs Docker for creating them. Let me tell you why! If you would favor a video explanation on the same, then you can go through the below video.

What Is Kubernetes | Kubernetes Introduction | Kubernetes Tutorial For Beginners | Edureka



This Edureka video on "What is Kubernetes" will give you an introduction to one of the most popular Devops tool in the market – Kubernetes, and its importance in today's IT processes.

Why Use Kubernetes?

Keep in mind that, as the traffic increases, they even have to scale up the number of containers to service the 'n' no of requests that come in every second. And, they have to also scale down the containers when the demand is less. Can all this be done natively?

Well to be honest, i'm not sure it can be done. Even if it can be done, it is only then its only after loads of manual effort for managing those containers. So, the real question is, **is it really worth it**? Won't automated intervention make life easier? Absolutely it will!

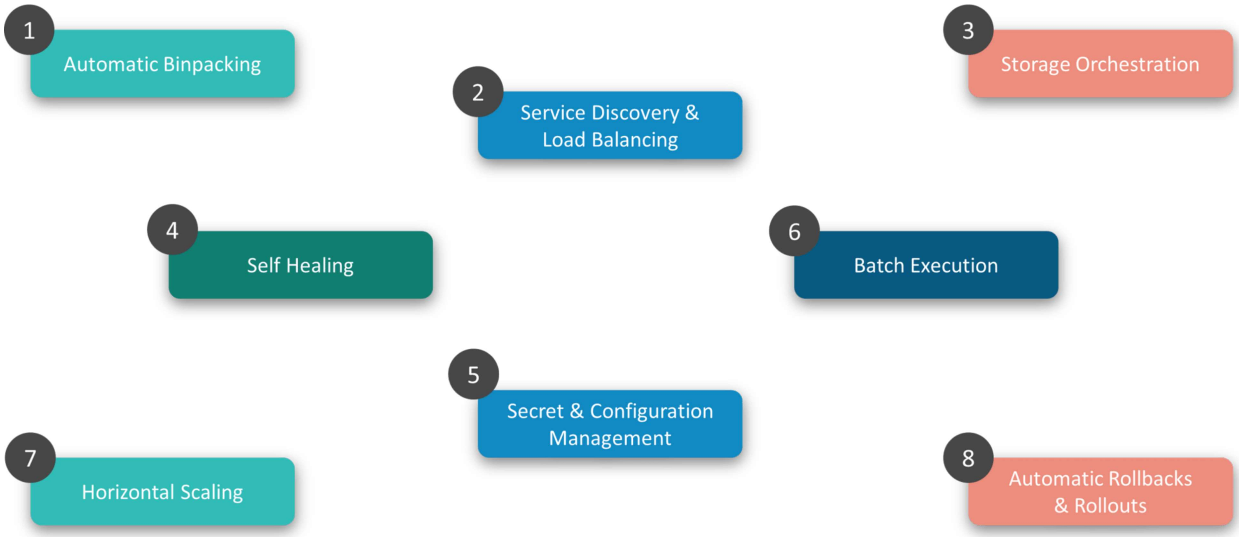
That is why, the need for container management tools is imminent. Both **Docker Swarm** and **Kubernetes** are popular tools for Container management and orchestration. But, Kubernetes is the undisputed market leader. Partly because it is Google's brainchild and partly because of its better functionality.

Logically speaking, Docker Swarm is a better option because it runs right on top of Docker right? If I were you, I would have had the same doubt and it would have been my #1 mystery to solve. So, if your thinking the same, read this blog on the comparison between Kubernetes vs Docker Swarm here.

If I could choose my pick between the two, then it would have to be Kubernetes. The reason simply being: Auto-scaling of containers based on traffic needs. However, Docker Swarm is not intelligent enough to do Auto-scaling. Be as it may, let's move onto the next topic of this, what is Kubernetes blog.

Features Of Kubernetes

This is the right time to talk about Kubernetes' features because, you already know what it does and how it compares against Docker Swarm.


1 Automatic Binpacking
2 Service Discovery & Load Balancing
3 Storage Orchestration
4 Self Healing
6 Batch Execution
5 Secret & Configuration Management
7 Horizontal Scaling
8 Automatic Rollbacks & Rollouts

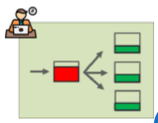
Kubernetes Features – What Is Kubernetes

1. Automatic Binpacking

Kubernetes automatically packages your application and schedules the containers based on their requirements and available resources while not sacrificing availability. To ensure complete utilization and save unused resources, Kubernetes balances between critical and best-effort workloads.

2. Service Discovery & Load balancing

With Kubernetes, there is no need to worry about networking and communication because Kubernetes will automatically assign IP addresses to containers and a single DNS name for a set of containers, that can load-balance traffic inside the cluster.

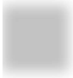


3. Storage Orchestration

With Kubernetes, you can mount the storage system of your choice. You can either opt for local storage, or choose a provider such as GCP or AWS, or perhaps use a shared network storage system such as NFS, iSCSI, etc.

health checks. But if nodes itself die, then it replaces and reschedules those failed containers on other available nodes.

5. Secret & Configuration Management



Kubernetes can help you deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

6. Batch Execution

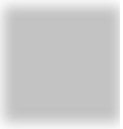
In addition to managing services, Kubernetes can also manage your batch and CI workloads, thus replacing containers that fail, if desired.

7. Horizontal Scaling

Kubernetes needs only 1 command to scale up the containers, or to scale them down when using the CLI. Else, scaling can also be done via the Dashboard (kubernetes UI).

8. Automatic Rollbacks & Rollouts

Kubernetes progressively rolls out changes and updates to your application or its configuration, by ensuring that not all instances are worked at the same instance. Even if something goes wrong, Kubernetes will rollback the change for you.



These were some of the notable features of Kubernetes. Let me delve into the attractive aspects of Kubernetes with a real-life implementation of it and how it solved a major industry worry.

Case Study: How Kubernetes was at the center of Pokemon Go's evolution

I'm pretty sure everyone reading this blog would have played this famous smartphone game. Or atleast you would have heard of this game. I'm so sure because this game literally smashed every record set by gaming applications in both the *Android* and *iOS* markets.

Pokemon Go developed by Niantic Labs and was initially launched only in North America, Australia & New Zealand. In just a few weeks upon its worldwide release, the game reached **500+ million downloads** with an average of **20+ million daily active users**. These stats bettered, those set by games like Candy Crush and Clash of Clans.

Pokemon Go:– Game backend with Kubernetes

The app backend was written in Java combined with libGDX. The program was hosted on a Java cloud with Google Cloud Bigtable NoSQL database. And this architecture was built on top of Kubernetes, making it their scaling strategy.

Rapid iteration of pushing updates worldwide was done thanks to MapReduce and in particular Cloud Dataflow for combining data, doing efficient MapReduce shuffles, and for scaling their infrastructure.

The actual challenge: For most big applications like this is horizontal scaling. Horizontal scaling is when you are scaling up your servers for servicing the increasing the number of requests from multiple players and playing environments. But for this game in particular, vertical scaling was also a major challenge because of the changing environment of players in real-time. And this change also has to be reflected to all the others playing nearby because reflecting the same gaming world to everyone is how the game works. Each individual server's performance and specs also had to be scaled simultaneously, and this was the ultimate challenge which needed to be taken care of by Kubrenetes.

Conclusion: Not only did Kubernetes help in horizontal and vertical scaling of containers, but it excelled in terms of engineering expectations. They planned their deployment for a basic estimate and the severs were ready for a maximum of 5x traffic. However, the game's popularity rose so much that, they had to scale up to 50x times. Ask engineers from other companies, and 95% of them will respond with their server meltdown stories and how their business went down crashing. But not at Niantic Labs, the developers of Pokemon Go.

Edward Wu, Director of Software Engineering, at Niantics said,

"We knew we had something special on hand when these were exceeded in hours."   "We believe that people are healthier when they go outside and have a reason to be connected to others."

Become a Certified Professional→

Kubernetes Architecture

So, now on moving onto the next part of this *'what is Kubernetes'* blog, let me explain the working architecture of Kubernetes.

Since Kubernetes implements a cluster computing background, everything works from inside a ***Kubernetes Cluster***. This cluster is hosted by one node acting as the 'master' of the cluster, and other nodes as 'nodes' which do the actual 'containerization'. Below is a diagram showing the same.

Kubernetes Architecture – What Is Kubernetes

Master controls the cluster, and the nodes in it. It ensures the execution only happens in nodes and coordinates the act. Nodes host the containers; in-fact these Containers are grouped logically to form Pods. Each node can run multiple such Pods, which are a group of containers, that interact with each other, for a deployment.

Replication Controller is Master's resource to ensure that the requested no. of pods are always running on nodes. Service is an object on Master that provides load balancing across a replicated group of Pods.

So, that's the Kubernetes architecture in simple fashion. You can expect more details on the architecture in my next blog. A better news is, the next blog will also have a hands-on demonstration of installing Kubernetes cluster and deploying an application.