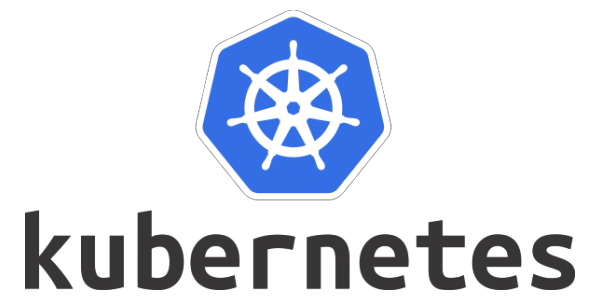
# **What is Kubernetes?**



Kubernetes is a portable, extensible, open-source container orchestration platform for managing containerized workloads and services.

Kubernetes, also called K8s, is basically a numeronym standard that has been used since the 1980s. For example, in Kubernetes, there are 8 words in between K and S like that.

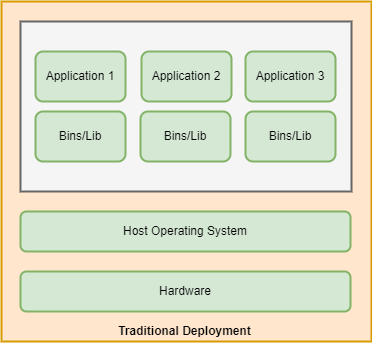
Google developed an internal system called Borg and later Omega, which they used to orchestrate the data center.

In 2014, Google introduced Kubernetes as an open-source project, and it is written in the Go language. Later on, donated to the Cloud Native Computing Foundation (CNCF).

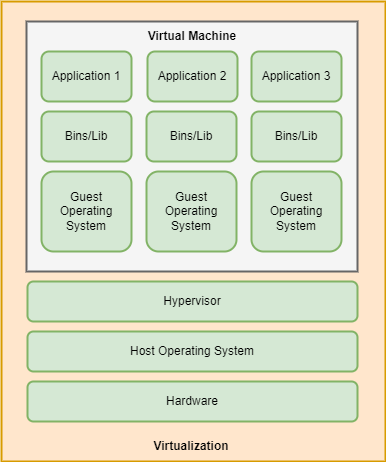
Kubernetes has all the capabilities to automate container deployment, load balancing, and auto-scaling.

**What was before containerization?**

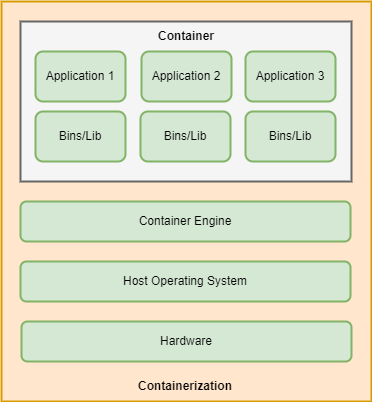
Initially, in the tech world, companies use the physical server to deploy the application, and if multiple applications are running on the same server, then a few applications take more resources to run, and others might be impacted. Also, it is really challenging for organizations to manage infrastructure at a high cost.



As a solution, a virtual machine comes into the picture because it’s easy to manage and maintain. Also, VMs offer many advantages over physical servers, like the capability to run multiple applications with their own operating systems and hardware resources on the same physical machine. So, it reduces management and resource costs.



Containers are the same as VMs but have some additional properties, like the fact that the application running inside the container is lightweight and shares the host machine’s OS kernel, and due to that, it does not require additional resources like VMs.



**Why Kubernetes?**

Containers are a good way to bundle and run applications in an isolated environment, but with that, we need to manage containers efficiently without any downtime. For example, if the application is running in a production environment and the running container goes down, you will need to create a new container using different commands or using some other things. But, at a large level, it’s really hard to manage a number of containers.

As a solution, Kubernetes comes into the picture because it is a container orchestration platform and has all kinds of capabilities like auto-scaling, load-balancing, version control, health monitoring, auto-scheduling, and many more.

Kubernetes monitors everything; if multiple users login at the same time and traffic suddenly increases, it will auto-scale and provide other resources to different containers that are running inside the node.

**Features of Kubernetes**

**1. Auto Scaling**

Kubernetes automatically increases and decreases the number of pods based on network traffic and allocates different resources.

**2. Automate Deployment**

Kubernetes automates the deployment of applications with the help of different cloud providers.

**3. Fault Tolerance**

Kubernetes manages all things related to the container. If he finds one of the pods and the container goes down due to high network traffic, it will automatically start a new instance and provide different resources for it.

**4. Load Balancing**

Kubernetes load balances and manages all incoming requests from outside the cluster, and it continuously looks at the running pods under different nodes and sends a request to a particular service using the load balancing technique.

**5. Rollout and Rollback**

Kubernetes rollout and rollback if anything wrong happens with the application after certain changes and managing to version

**6. Health Monitoring**

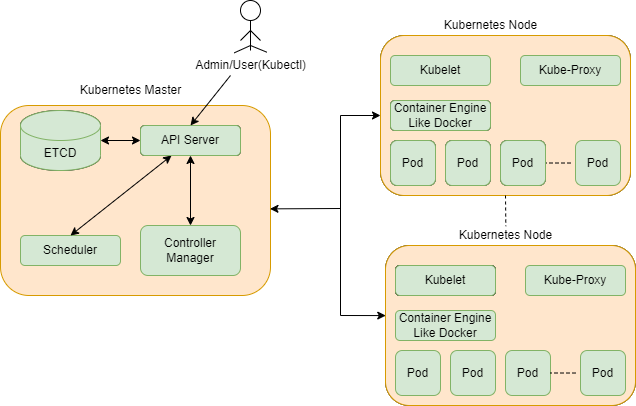
Kubernetes continuously checks the health of the running node to see if the containers and pods are working fine or not.

**7. Platform Independent**

Kubernetes is an open-source tool, which is why it can move workloads and applications anywhere on public cloud, on-premises, hybrid, and public cloud infrastructure.

**Kubernetes concepts and architecture**

Generally, Kubernetes has two different nodes running for the application.



**1. Master Node**

The master node is responsible for managing the complete cluster.

It has four components: **ETCD**, **API Server**, **Scheduler**, and **Controller Manager**.

* Users can access the master using the CLI through the API server.
* The master node continuously monitors all nodes in the cluster and takes action accordingly.
* Kubernetes can have more than one master node for high availability.

Following are the four components of the Master Node

**a) API Server**

The master can communicate with all the clusters through the API server. It is the main access point to the control plane.

* The API server directly interacts with the user. Ex-users are able to apply YML or JSON files directly to the API server through the CLI.
* API Server has the capability to auto-scale as per load.
* The API Server is the front end of the control plane.

**b) ETCD**

* ETCD is used to store data as key-value pairs, which are used by Kubernetes to manage the clusters.
* It also stores the metadata and the status of the cluster.
* ETCD is a consistent and high-availability data store.
* It is also responsible for maintaining the lock mechanism to reduce conflicts between the masters.
* When there are multiple masters and nodes, ETCD stores all the data in a distributed manner.

**ETCD has the following features,**

*1. Fully replicated*

The entire state of the data is available on every node that is present in the cluster.

*2. Secure*

It also implements automatic client TLS certificate authentication.

*3. Fast*

ETCD is very fast and can easily perform multiple operations in seconds.

**c) Scheduler**

* The scheduler is responsible for distributing the work across multiple different available nodes.
* It always looks at newly created containers and assigns the node.
* handles pod creation and management.
* When the user makes the request for the creation and management of pods, the scheduler will take action on that request smoothly.

**d) Control Manager**

* Controllers are the main thing behind orchestration.
* Controllers continuously look at and watch the health of the node, whether it is responding or not, and take action according to it.
* It also manages the state of the controller related to deployment, replicas, and the number of nodes running in the cluster.

**Following different components that are present in the master**

*1. Route Controller:* Responsible for managing the networking

*2. Node Controller:* Responsible for detecting the node if it is not responding.

*3. Service Controller:* Responsible for load balancing to manage the load

*4. Volume Controller:* Responsible for mounting and creating volume storage.

**2. Worker Node**

**a) Kubelet**

* Kubernetes Worker Node has Kubelet to communicate with Master Node and provide all information continuously to the Master Node related to the Health of Nodes
* It is also responsible for carrying out the actions taken by the master node.
* listens to the Kubernetes master.
* It also sends access reports for the node to the master.

**b) Kube-proxy**

* Kube-Proxy is responsible for managing the network traffic properly as per the rule defined in the controller manager,
* It also assigns IP addresses to each pod.
* Kube-Proxy runs on each node, and it has the responsibility to check that a unique IP address is assigned to each pod.

**c) Pods**

* It runs as a single instance of the application.
* It may have many resources, like IP addresses, containers, and storage.
* Pods have one or more containers that are deployed on the same host.
* In Kubernetes, the control unit is a pod, not a container.
* It also has one more tightly coupled container in one pod, sharing resources with each other.
* Pods run on the worker node, which is controlled by the master.
* Usually, one pod contains one container, and without a port, Kubernetes is not able to run the container because Kubernetes only knows pods, not containers.

**d) Container Engine**

* The container engine is responsible for running containerized applications Kubernetes supports different container runtimes, but Docker is famous.

**High-level Kubernetes objects**

**1. Replica set**

A replica set maintains a group of instances and a number of identical running pods.

**2. Deployment**

Deployment maintains the desired state and versioning.

**3. Volume**

Kubernetes provides stable storage services in the form of volumes that persist data for the whole lifetime of the pods.

**4. Service**

Inside the node, multiple pods are in running mode under the selector, so in that case, the service manages DNS addresses for them.

This is all about Kubernetes.