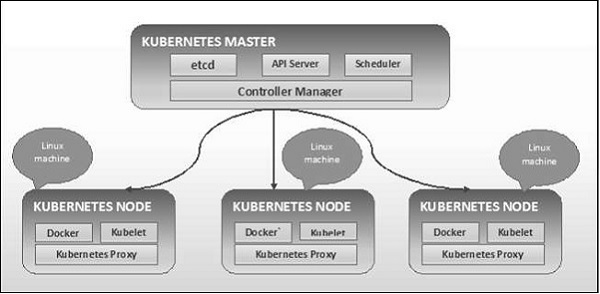
Kubernetes is **an open-source container orchestration platform**

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem

Kubernetes - Cluster Architecture

As seen in the following diagram, Kubernetes follows client-server architecture. Wherein, we have master installed on one machine and the node on separate Linux machines.



The key components of master and node are defined in the following section.

Kubernetes - Master Machine Components

Following are the components of Kubernetes Master Machine.

etcd

It stores the configuration information which can be used by each of the nodes in the cluster. It is a high availability key value store that can be distributed among multiple nodes. It is accessible only by Kubernetes API server as it may have some sensitive information. It is a distributed key value Store which is accessible to all.

API Server

Kubernetes is an API server which provides all the operation on cluster using the API. API server implements an interface, which means different tools and libraries can readily communicate with it. **Kubeconfig** is a package along with the server side tools that can be used for communication. It exposes Kubernetes API.

Controller Manager

This component is responsible for most of the collectors that regulates the state of cluster and performs a task. In general, it can be considered as a daemon which runs in nonterminating loop and is responsible for collecting and sending information to API server. It works toward getting the shared state of cluster and then make changes to bring the current status of the server to the desired state. The key controllers are replication controller, endpoint controller, namespace controller, and service account controller. The controller manager runs different kind of controllers to handle nodes, endpoints, etc.

Scheduler

This is one of the key components of Kubernetes master. It is a service in master responsible for distributing the workload. It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload. In other words, this is the mechanism responsible for allocating pods to available nodes. The scheduler is responsible for workload utilization and allocating pod to new node.

Kubernetes - Node Components

Following are the key components of Node server which are necessary to communicate with Kubernetes master.

Docker

The first requirement of each node is Docker which helps in running the encapsulated application containers in a relatively isolated but lightweight operating environment.

Kubelet Service

This is a small service in each node responsible for relaying information to and from control plane service. It interacts with **etcd** store to read configuration details and wright values. This communicates with the master component to receive commands and work. The **kubelet** process then assumes responsibility for maintaining the state of work and the node server. It manages network rules, port forwarding, etc.

Kubernetes Proxy Service

This is a proxy service which runs on each node and helps in making services available to the external host. It helps in forwarding the request to correct containers and is capable of performing primitive load balancing. It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well. It manages pods on node, volumes, secrets, creating new containers’ health checkup, etc.

Kubernetes - Master and Node Structure

The following illustrations show the structure of Kubernetes Master and Node.

Key Objects of Kubernetes

Following are the key objects which exist in the Kubernetes:

**Pod**

It is the smallest and simplest basic unit of the Kubernetes application. This object indicates the processes which are running in the cluster.

**Node**

A **node** is nothing but a single host, which is used to run the virtual or physical machines. A node in the Kubernetes cluster is also known as a minion.

**Service**

A **service** in a Kubernetes is a logical set of pods, which works together. With the help of services, users can easily manage load balancing configurations.

**ReplicaSet**

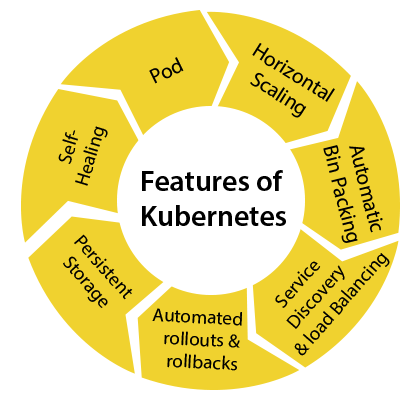
A **ReplicaSet** in the Kubernetes is used to identify the particular number of pod replicas are running at a given time. It replaces the replication controller because it is more powerful and allows a user to use the "set-based" label selector.

**Namespace**

**Kubernetes** supports various virtual clusters, which are known as namespaces. It is a way of dividing the cluster resources between two or more users.

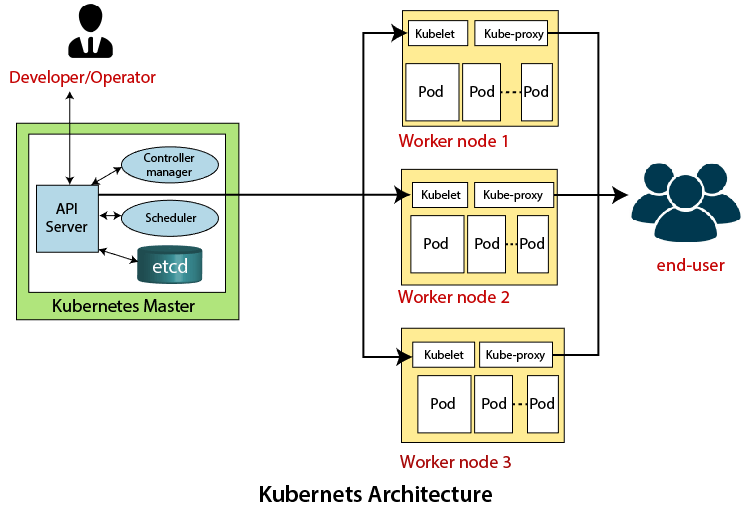
Features of Kubernetes

Following are the essential features of Kubernetes:



1. **Pod:** It is a deployment unit in Kubernetes with a single Internet protocol address.
2. **Horizontal Scaling:** It is an important feature in the Kubernetes. This feature uses a **HorizontalPodAutoscalar** to automatically increase or decrease the number of pods in a deployment, replication controller, replica set, or stateful set on the basis of observed CPU utilization.
3. **Automatic Bin Packing:** Kubernetes helps the user to declare the maximum and minimum resources of computers for their containers.
4. **Service Discovery and load balancing:** Kubernetes assigns the IP addresses and a Name of DNS for a set of containers, and also balances the load across them.
5. **Automated rollouts and rollbacks:** Using the rollouts, Kubernetes distributes the changes and updates to an application or its configuration. If any problem occurs in the system, then this technique rollbacks those changes for you immediately.
6. **Persistent Storage:** Kubernetes provides an essential feature called '**persistent storage'** for storing the data, which cannot be lost after the pod is killed or rescheduled. Kubernetes supports various storage systems for storing the data, such as **Google Compute Engine's Persistent Disks (GCE PD) or Amazon Elastic Block Storage (EBS).** It also provides the distributed file systems: **NFS or GFS**.
7. **Self-Healing:** This feature plays an important role in the concept of Kubernetes. Those containers which are failed during the execution process, Kubernetes restarts them automatically. And, those containers which do not reply to the user-defined health check, it stops them from working automatically.

Kubernetes Architecture



The architecture of Kubernetes actually follows the client-server architecture. It consists of the following two main components:

1. Master Node (Control Plane)
2. Slave/worker node

Master Node or Kubernetes Control Plane

The master node in a Kubernetes architecture is used to manage the states of a cluster. It is actually an entry point for all types of administrative tasks. In the Kubernetes cluster, more than one master node is present for checking the fault tolerance.

Following are the four different components which exist in the Master node or Kubernetes Control plane:

1. API Server
2. Scheduler
3. Controller Manager
4. ETCD

**API Server**

The Kubernetes API server receives the REST commands which are sent by the user. After receiving, it validates the REST requests, process, and then executes them. After the execution of REST commands, the resulting state of a cluster is saved in '**etcd**' as a distributed key-value store.

**Scheduler**

The scheduler in a master node schedules the tasks to the worker nodes. And, for every worker node, it is used to store the resource usage information.  
In other words, it is a process that is responsible for assigning pods to the available worker nodes.

**Controller Manager**

The Controller manager is also known as a controller. It is a daemon that executes in the non-terminating control loops. The controllers in a master node perform a task and manage the state of the cluster. In the Kubernetes, the controller manager executes the various types of controllers for handling the nodes, endpoints, etc.

**ETCD**

It is an open-source, simple, distributed key-value storage which is used to store the cluster data. It is a part of a master node which is written in a GO programming language.

Now, we have learned about the functioning and components of a master node; let's see what is the function of a slave/worker node and what are its components.

Worker/Slave node

The Worker node in a Kubernetes is also known as minions. A worker node is a physical machine that executes the applications using pods. It contains all the essential services which allow a user to assign the resources to the scheduled containers.

Following are the different components which are presents in the Worker or slave node:

**Kubelet**

This component is an agent service that executes on each worker node in a cluster. It ensures that the pods and their containers are running smoothly. Every **kubelet** in each worker node communicates with the master node. It also starts, stops, and maintains the containers which are organized into pods directly by the master node.

**Kube-proxy**

It is a proxy service of Kubernetes, which is executed simply on each worker node in the cluster. The main aim of this component is request forwarding. Each node interacts with the Kubernetes services through **Kube-proxy**.

**Pods**

A **pod** is a combination of one or more containers which logically execute together on nodes. One worker node can easily execute multiple pods.