# What is Kubernetes ?

It is a container management (orchestration) tool

Also referred to as K8s (Eight characters between K and S)

Container Orchestration tool or engine automates deploying, scaling and managing containerized application on a group of servers.

Some of the container management tools are :

* Kubernetes
* Docker Swarm
* Apache Mesos Marathon

Kubernetes manages containerized apps :

* Deploying
* Scheduling
* Scaling
* Load balancing
* Batch execution
* Rollbacks
* Monitoring

|  |  |
| --- | --- |
| Dockers | Kubernetes |
| Creates containers | Manages containers |

# Features of Kubernetes:

1. Automatic bin packing

Kubernetes automatically packages your application and schedules the container based on the requirements and resources available.

1. Service discovery and load balancing:

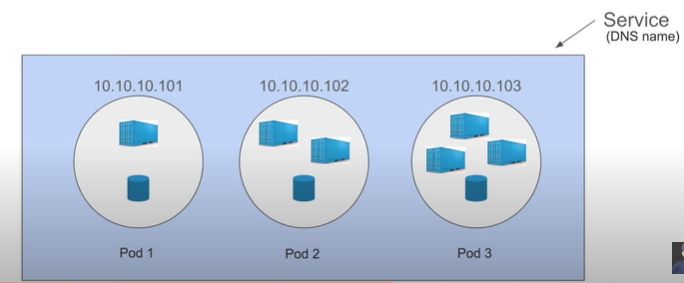
Kubernetes doesn’t run containers directly, instead it wraps one or more containers into a higher-level structure called a POD.

A Pod contains:

* An application container (or in some cases multiple containers)
* Storage resources
* A unique network IP

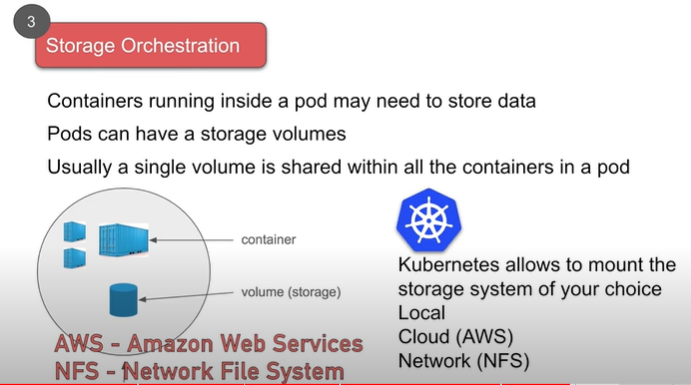
***Inside a Node there can be one or multiple Pods and inside a Pod there can be one or multiple containers***

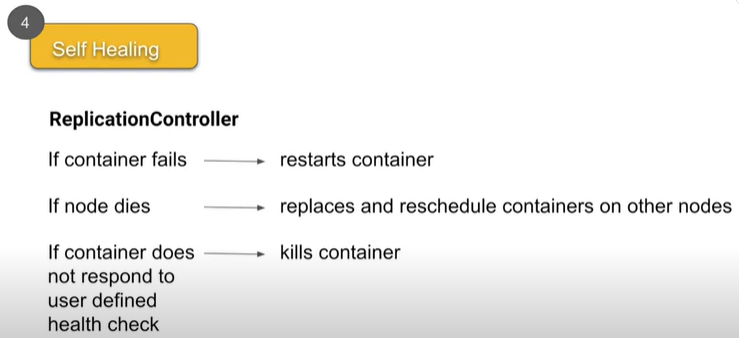
Pods that have the same set of functions are abstracted into sets called services.

Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods and can load-balance across them.

A diagram of a service

Description automatically generated

1. Storage Orchestration
2. Self Healing



**Automatic bin packing** Automatically places containers based on their resource requirements like CPU & Memory (RAM), while not sacrificing availability Saves resources

**Service discover & load balancing** Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them With this system, Kubernetes has control over network and communication between pods and can load load balance across them

**Storage Orchestration** Kubernetes allows to mount the storage system of your choice Local Cloud (AWS) Network (NFS)

**Self-healing** If a container fails - restarts container If node dies - replaces and reschedule containers on other nodes If container does not respond to user defined health check - kills container This is taken care by Kubernetes ReplicationController

1. Automated rollout’s and rollback’s

Rollout : deploy changes to the application or its configuration

Rollback: revert the changes and restore to previous state

1. Secret and configuration management

Secret and config maps are created and managed outside the pods

Secret is used to create and maintain credentials and configurations are managed in config maps.

**Secrets & Configurations are stored in ETCD.**

**ETCD is a key-value datastore(database)**

***The Max size limit of a Secret is 1MB***

1. Batch execution

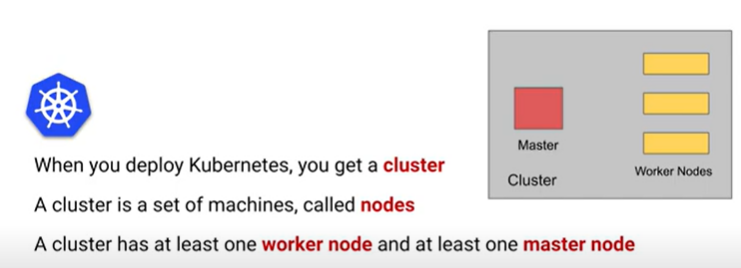
Kubernetes supports batch execution, long-running jobs and replaces failed containers.

1. Horizontal scaling

In Kubernetes, we can scale up or down the containers.

* Using commands
* From the dashboard (Kubernetes ui)
* Automatically based on CPU usage

# Architecture of Kubernetes:



In Earlier days of Kubernetes, worker nodes were called **minions**

There can be more than one master nodes in a cluster to provide a cluster with failover and high availability.

***A node can be a physical machine, Virtual machine and VM on cloud***

***Master manages the Nodes and Pods in a cluster.***

Components of Master Node:

1. API server – for all communications (JSON over HTTP API)
2. Scheduler – schedules pods on nodes
3. Controller Manager – runs controllers
4. Etcd – Open source, distributed key-value database from CoreOS

API server exposes some API’s which users can interact. Only API server can interact with the **ETCD** datastore directly

**Kubectl – Command line tool / utility to interact with the Kubernetes API**

UI – Frondend dashboard

Scheduler obtains data from etcd, via the API server, resource usage data for each worker node in the cluster

Control Manager –

* Kube-controller-manager : Runs controllers responsible to act when a node becomes unavailable.
* Cloud-controller-manager : Runs controllers responsible to interact with the infrastructure of a cloud provider when node becomes unavailable.

Components of worker node:

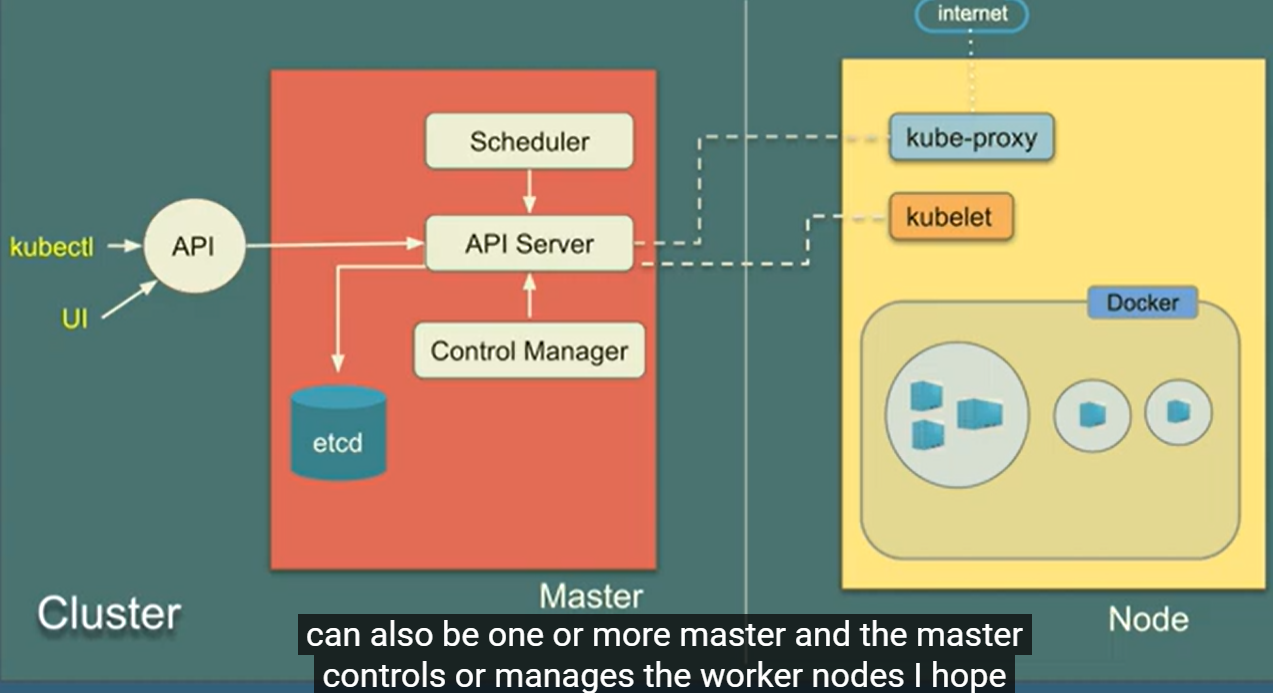
1. Kubelet – Agent running on each node communicates with components from the master node.
2. Kube-proxy – Network agent which runs on each node responsible for maintaining network configuration & rules. All worker nodes run a daemon called kube-proxy, which watches the **API server on the master node** for the addition and removal of services and endpoints.
3. Container runtime – Is the software that is responsible for running containers

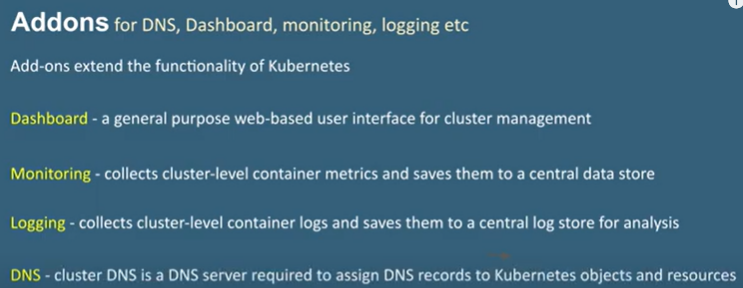
Kubernetes supports several container runtimes:

* Docker
* Containerd
* Cri-o
* Rktlet
* Kubernetes CRI (Container Runtime Interface)

Kubernetes does not have the capability to directly handle containers

In order to run and manage a container’s lifecycle, Kubernetes requires a container runtime to the node where a Pod and its containers are to be scheduled.





Options to use / Install Kubernetes :

* Online Kubernetes labs
* Play with K8s ([Play with Kubernetes (play-with-k8s.com)](https://labs.play-with-k8s.com/))
* Play with Kubernetes Classroom
* Kubernetes installation tools
* MiniKube
* Kubeadm
* Cloud based Kubernetes services
* GKE - Google Kubernetes Engine
* AKS – Azure Kubernetes service
* Amazon EKS

Kubernetes commands:

* Kubectl cluster-info
* Kubectl cluster-info dump [ To further debug & diagnose cluster problems]
* Kubectl get nodes
* Kubectl run (allows containers to be deployed onto the cluster)
* Kubectl get pods (status of the deployment can be discovered via the running pods)
* Kubectl get nodes
* Kubectl get services
* Kubectl create deployment first-deployment – image=katacoda/docker-http-server
* Kubectl run nginx –image=nginx
* kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16