# Container management tools

It **controls** the containers like docker.

# Container type

Kubernetes, D container, Rocket, **Docker swarm,** apache marathon

# Kuberneters

* It helps to automates **container deployment, load balncing, scaling, capability , highly availability**
* It schedule runs, **crons jobs, and** manges isolated conatiners which are running on virtual /physical/cloud
* Written in Golang by google
* Now handled by CNCF

## Online plateform

* Play with K8S
* Play with Kyberneters classroom
* Kubernetes playground

## Cloud based k8s services

* GKE, AKS, Amazon EKS(Elastic KS)

## Kubernetes installation tool

* Minicube
* Kubeadm

## Problems without cmt

* Containers **can not communicate**
* Autoscalling and **load balancing not possible**
* Every time more careful about the containers

# Features of K8S

* Orchestration(clusting of any no. of the containers)
* Autoscaling

1. vertically scaling 🡪 2GB to 8 GB, no new hiring only use existing employee for the work
2. horizental scaling(**preferable**) 🡪 create one more containers, hiring new employee

* Autohealing
* Load Balancing
* Plateform independent
* Fault Tolerance (**node/pod failure**)
* Rollback
* Health monitroing
* Batch execution

## Kubernetes benefits

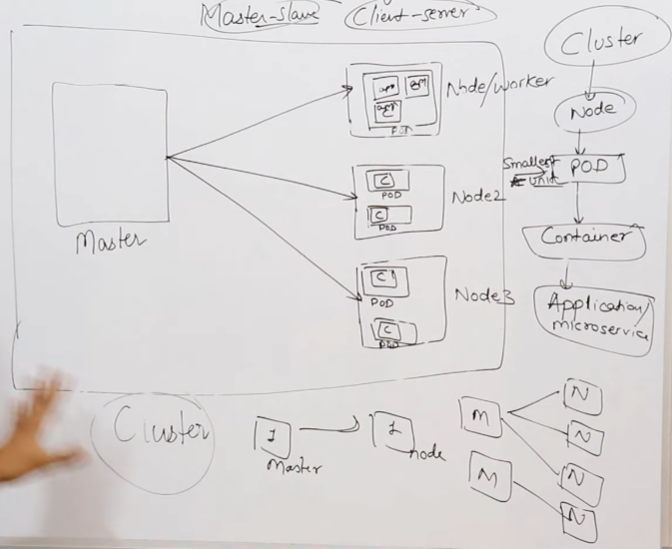
* GUI is available
* Inbuilt tool present for monitoring
* Autoscaling is available
* Data volumes only shared with containers pods.
  + Pod 🡪 smallest and basic unit; like containers in docker.
* Kubernetes not directly handle the containers, only deal with pod.
* Logging and monitoring by inbuilt tool

## Docker swarm benefits

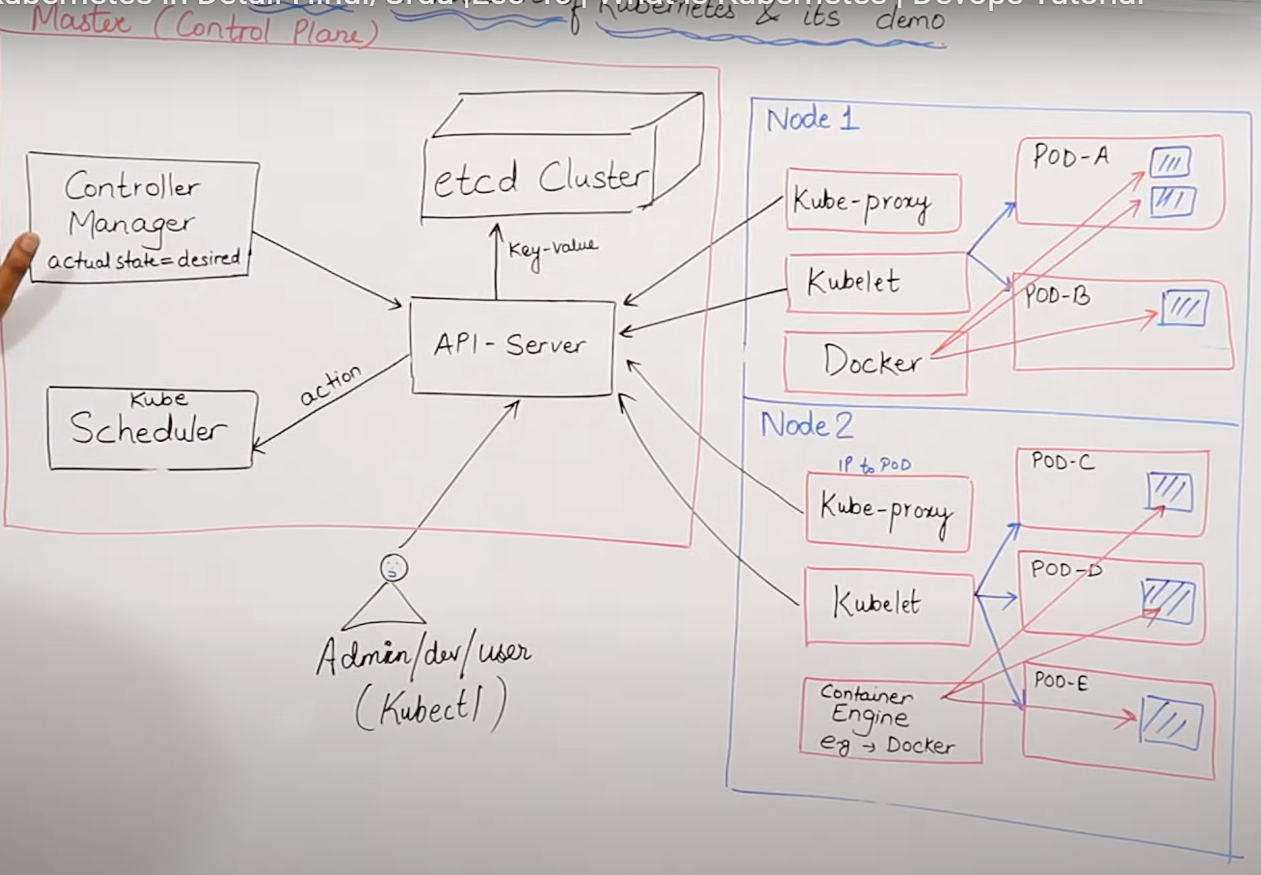
* GUI is not available
* Not autoscaling is available
* Can shared data volumes with any other containers
* Logging and monitoring by 3rd party tool like splunk.

## Master slave architexture

1. Master(server) – kubernetes api
2. Multiple master also posible
3. Pod is part of node
4. **One node can have multiple pod**
5. **One pod can have multiple container**
6. Node(client server) **– k8s communicate with the pod**



# K8S architecture



## Master control panel

* API server : like receptionist
* Controller manager : like controller in api, guranttee of required work
* Etcd cluster(**not part of k8s**) : database like zookeeper, ohai in chef , key-value
* Kube schuduler
* Kubectl
  1. JSON/YAML file
  2. Admin/dev/user
* container engine like docker

## Worker

* Node
* Kube-proxy : assign ip to pod
* Kublet(khabri) it control pod by sending request to api server
* container engine like docker
* **Container**

1. It does not have any IP only pod have IP
2. container r tightly coupled
3. **once fail it will fail all container in it**
4. one pod one container is fine approach
5. failed pod can not be repaired
6. always create new pod and will get new IP

* Pod

1. atomic unit like cell in human
2. **K8S does not have container on node but on pod**
3. K8S only know pod not container
4. Can have more thn one container
5. Once failed it is useless

# Master component

## Manifest

* Yml or json

## Kube api server

* It is kind of the front end server
* It can handle automatically manifiest request

## Etcd

* Store metadata, Fully replicated, Secure, Fast , high available store(key-client)

## Kube schuduler

* It handle pod creation and management reqeust

## Controller manager

* Try to make actual and desired state equal
* Component
* Node controller : detect the node and check how are newly created, respoding

Two package:

* Kube controller if non cloud
* Cloud controller if cloud
* Route controller : route the network
* Service controller : load balancer
* Volume controller

# Slave component

## Kubelet

* It contact with api server for the request and response
* Listens to kubernetes,
* 1025 pod number it uses.
* Success and fail report to the master

## Container engine

* Pulls images
* Start and stop containers
* Exposing containers on the ports in manifest
* With the kublets

## Kube proxy

* Assign ip to each pod(dynamic on new creation)

## Pods

* Smallest unit in kubernetes
* It wrappe the container
* K8s controls only pods not the pods
* Tightly coupled container
* No by default auto healing and auto scaling of pods
* No recovery of the pods

## Higher level pods

These features give extra strength to the k8s:

* Replication set : auto scaling and auto healing
* Deployment versioning and rollback
* Service static ip and networking
* Volume non epherimal storage means outside the containers, on fails of pod it does not lost.