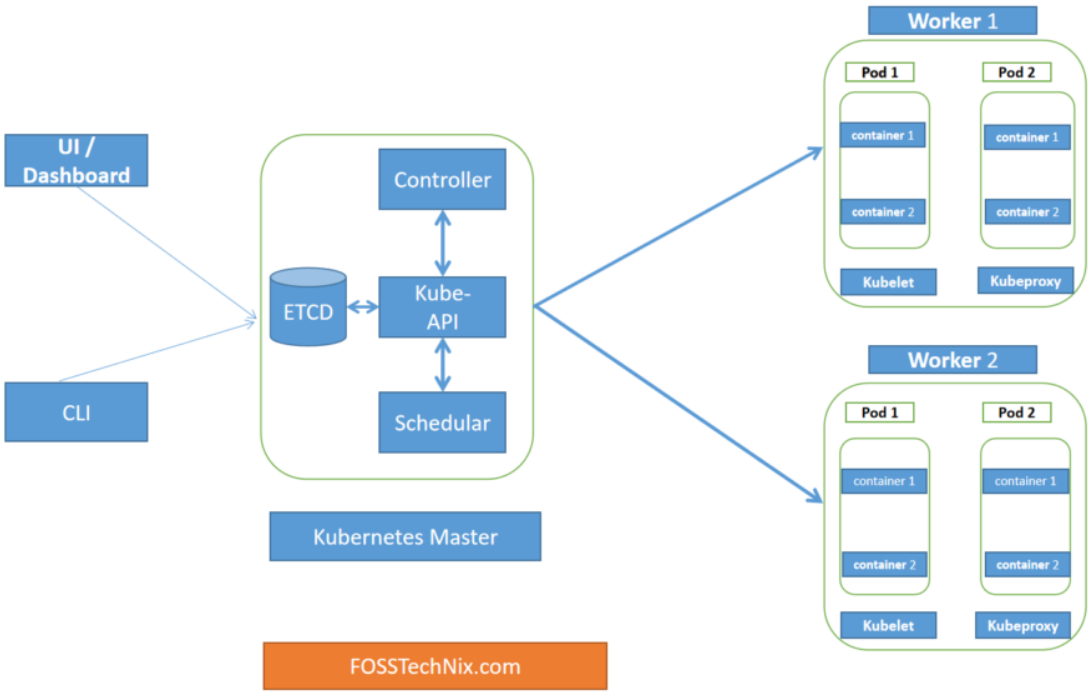
********** KUBERNETES(K8s) **********

⇒INTRODUCTION:-

→It is developed by **numeronym** in 1980's
Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It helps to manage containerized workloads and services, by providing a platform to manage the containers' lifecycles, networking, storage, and more.

ARCHITECTURE:-



 # CONTROLLER MANAGER:-

-: Make sure actual state of cluster matches to desired state

-: Two possible choice for controller manager

1. If K8s on cloud, then it will be cloud-controller manager
2. Kubernetes is an open-source container Management tool Which automates Container deployment, Container Scaling & Load balancing.
3. It schedules, runs and manages isolated Containers Which are running on virtual/physical/cloud Machines
4. All top cloud providers support kubernetes
5. Google deloped an internal system called ‘borg’ (later named as Omega) to deploy and manage thousand google application and service on their cluster.
6. In 2014 google introduced kubernetes an open source platform written in ‘Golang’ and later donated to CNCF.

==> ONLINE PLATEFORM FOR K8s

→ Kubernetes playground

→ Play with K8s

→ Play with kubernetes classroom

==> CLOUD BASED K8s SERVICE

→ GKE→Google kubernetes services

→ AKS→Azure kubernetes services

→ Amazon EKS(Elastic Kubernetes Services)

==> KUBERNETES INSTALLATION TOOL

→ Minicube
→ Kubeadm

Problem with scaling up the container

→ Container can't communicate with each other.
→ Autoscaling and Load Balencing was not possible
→ Container had to be managed carefully

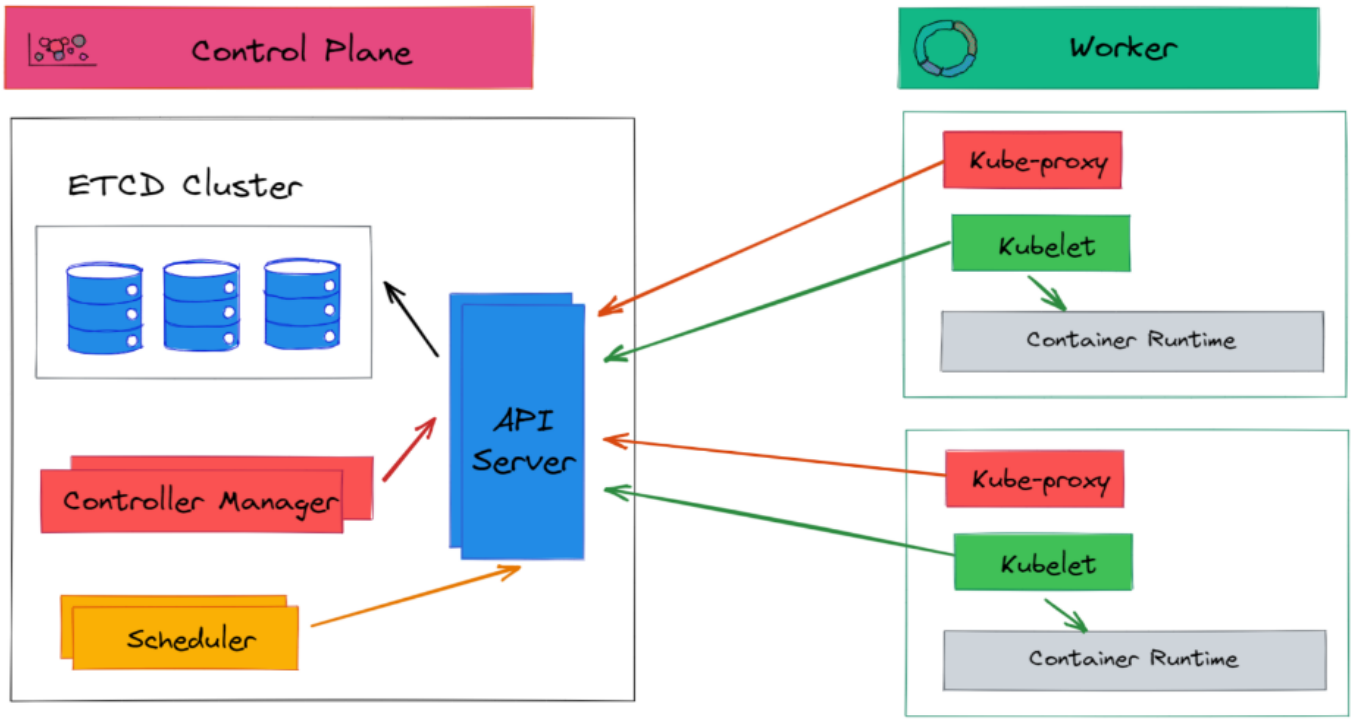
==> FUTURE OF KUBERNETES

→ Orchestration (Clustering of any no of containers running of different n/w)
→ Autoscaling
→ Auto-Healing
→ Load Balancing
→ Platform Independent (Cloud/Virtual/Physical)
→ Fault Tolerance (Node/POD Failure)
→ Rollback (Going back to previous version)
→ Health monitoring of containers
→ Batch Execution (One time, Sequential, parallel)

DIFFERENCE BETWEEN KUBERNETES & DOCKER SWARM:-

FEATURES	KUBERNETES	DOCKER SWARM
1. Installation and Cluster Configuration	→ Complicated and time consuming	→ Fast and Easy
2. Supports	→ K8s can work with almost all type of containers technology	→ Work with Docker only
3. GUI	→ GUI available	→ GUI not available
4. Data Volumes	→ Only shared with containers with in same pod.	→ Can be shared with any containers
5. Updates & Rollback	→ Process scheduling to maintain services while updating	→ Progressive update and service health monitoring throughout the update.
6. Auto Scaling	→ Supports Vertical and Horizontal Auto Scaling	→ Not support Auto Scaling
7. Logging and Monitoring	→ Inbuilt tool present for monitoring.	→ Use 3rd party tools like splunk.

MASTER CONTROL ARCHITECTURE:-



==> COMPONENT OF MASTER CONTROLLER:-

→ Kube-apiserver
→ Kube-scheduler
→ controller-Manager
→ etcd

1. Kube-apiserver (For all Communication):-

→ This api-server intracts directly with user (i.e we apply .yml or json manifest to Kube-apiserver)
→This Kube-apiserver is meant to scale automatically as per load
→ Kube api-server is front-end of Control-plane

2. etcd:-

→ Stores metadata and status of cluster
→ etcd is Consistent and high available store (Key value store)
→ Source of touch for cluster state (info about state of cluster)

Features:-

1. Fully Replicated:- The entire state is available on every node in the cluster.
2. Secure:- Implements automatic TLS With optional Client-Certificate authentication
3. Fast:- Benchmarked at 10000 writes per second

3. Kube-Scheduler (Action):-

→ When users make request for the creation & Management of Pods, Kube-Scheduler is going to take action on these requests.

→ Handle pod creation & Management

→ Kube-scheduler match/assign any node to create and run pods

→ A scheduler watches for newly created pods that have no node assigned for every pod that the scheduler discovers, the scheduler becomes Responsible for finding best node for that pod to run on.

→ Scheduler gets the information for hardware configuration from configuration files and schedules the pods on nodes accordingly.

4. Controller-Manager:-

→ Make sure action state of cluster matches to desire state

→ Two possible choice for controller manager.

I. If K8s on cloud then it will be cloud-controller-manager

II. If k8s on non-cloud, then it will be Kube-controller-manager

==> Components on master that runs Controller:-

1. Node-controller→ For checking the cloud provider to determine if a node has been detected in the cloud after its stops responding
2. Route controller→ Responsible for load balancers on your cloud against services of type load balancer.
3. Volume controller→ For creating attaching and mounting volumes and interacting with the cloud provider to orchestrate volume.
   start/stop container
   Exposing container on ports specified in manifest

5. Kube-proxy:-

→ Assign IP to each pod.

→ It is required to assign IP address to pods(Dynamic)

→ Kube-proxy runs on each node & this make sure that each pod will gets it's own unique IP address.

POD

→ Smallest unit in kubernetes.

→ Pod is a group of one or more container that are deployed together on the same host

→ A cluster is a group of nodes.

→ A cluster has atleast one worker node and one master node

→ In kubernetes, the control unit is the pod, not containers

→ Consist of one or more tightly coupled containers.

→ Pod runs on node which controlled by master

→ Kubernetes only knows about pods (Doesn't know any individual container)

→ Cannot start containers without a pod
→ One pod usually contain one container.

Multi-Container pods

→ Share access to memory space
→ Connect to each other using localhost.<container port>
→ Share access to the same volume
→ Containers within pod are deployed in an all-or-nothing manner
→ Entire pod is hosted on the same node (scheduler will decide about which node)

POD Limitations

→ No auto-healing or scaling
→ POD crashes

Higher level kubernetes objects

1. Replication set→ scaling and healing
2. Deployment → Versioning and Rollback
3. Service→ static (Non-ephemeral) IP and Networking.
4. Volume→ Non-ephemeral storage

**** Important ****

Kubectl → cloud
Kubeadm → on premise
Kubefed → Federated