diff b/w docker and Kubernetes:

1. In, docker generally it will be installed only on single host(machine). if there are multiple containers running, some containers may have enough resources to start (or) running containers may dies bcoz of lack of resources. But in Kubernetes, by default, it is a cluster (group of nodes) in which we have option to add nodes if required.

2. **Auto healing**- when a container got terminated unexpectedly. In docker, application inside the container is inaccessible unless a user (or) devops engineer start container.

3. **Auto scaling**

4. Docker doesn't provide any enterprise level support.

Kubernetes architecture:

A diagram of a server

AI-generated content may be incorrect.

**Control plane-** API server, controller manager, etcd, scheduler. ccm

**Data plane-** Kubelet, Kube-proxy, container runtime.

**Kubelet** – Its primarily responsibility Is to ensure that the pod is always running. if there is something wrong with the pod. kubelet will inform API server.

**Kube-proxy**- basically provides networking like every pod which has created should assign with an Ip address and has to provide with load balancing capabilities. Ex: if we increase replicas count from 5 to 10. new load should be distributed to these 10 pods equally. Kube-proxy will act as a default load balancer in this case.

**Container runtime** – responsible for running the container.

**API server**: API server is basically a component which exposes Kubernetes. This Kubernetes must be exposed to external world

**Scheduler-** Responsible for scheduling your pod (or) scheduling your resources in Kubernetes

**Etcd**: etcd is basically a key-value store. The entire Kubernetes information is stored as key-value pairs in etcd.

**Controller manager: ex** (Replica set). Ensures the configured number of replicas are running in the cluster.

**Cloud Controller manager:**

**POD:**

A pod is basically like a wrapper of container

Most of the times a pod has single container. But, in some cases, a pod contains multiple containers (like side car, init containers)

Ex: let say, we have a application which is deployed in a container. And that application wants to read some config files (or) user releated files from another container. In such cases, instead of creating two different containers we can create both containers in a single pod

**Advantages when two containers are deployed in same pod:**

**1. shared network** (one pod can connect with another pod using localhost:port )

**2. shared storage**