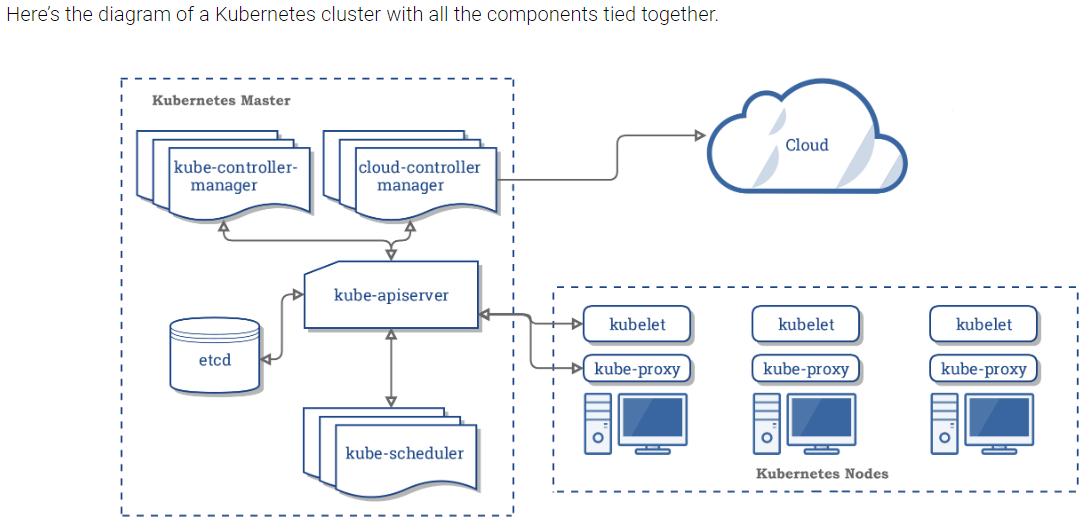
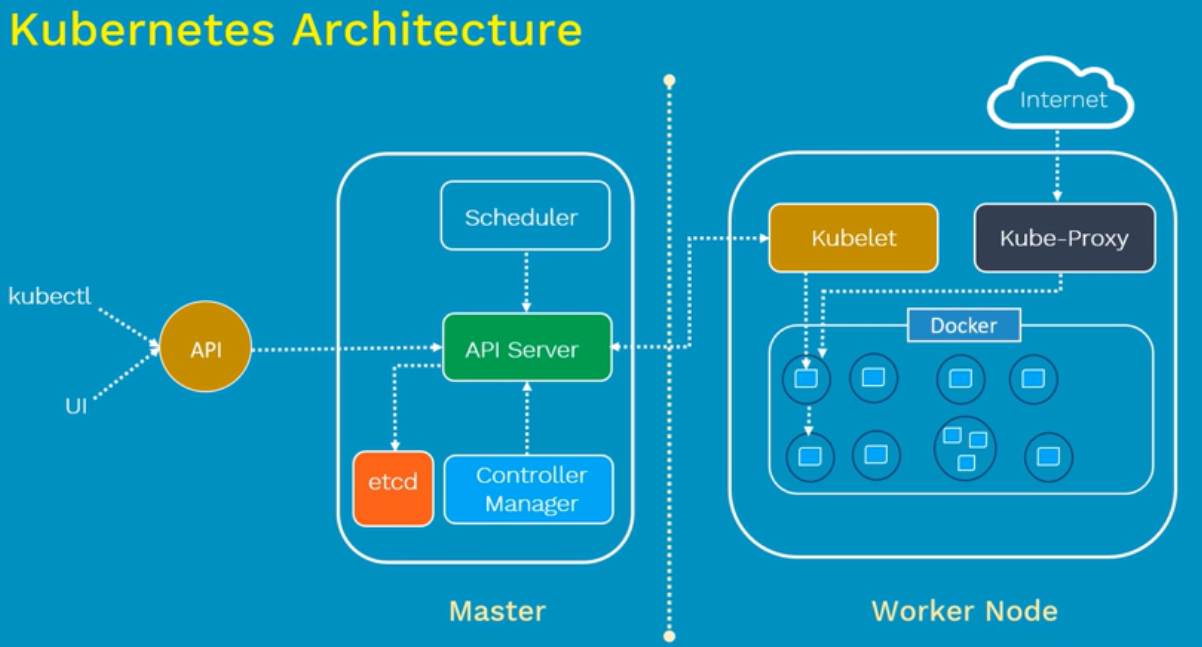
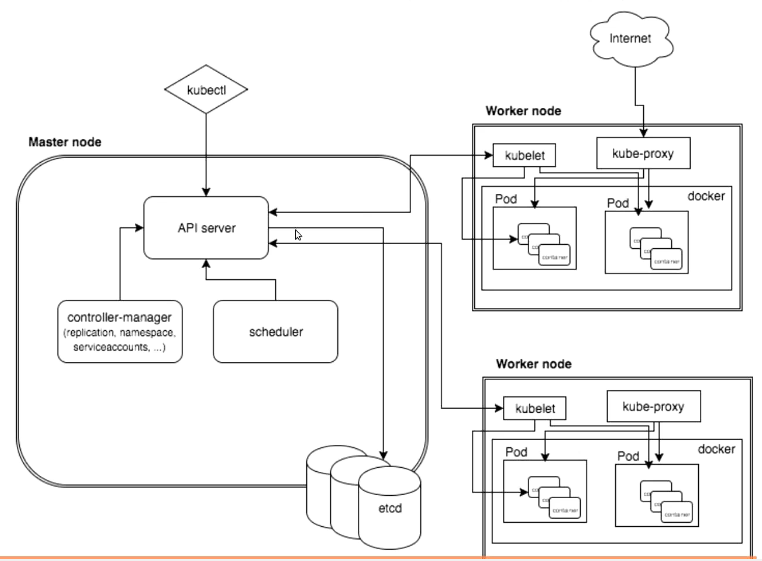
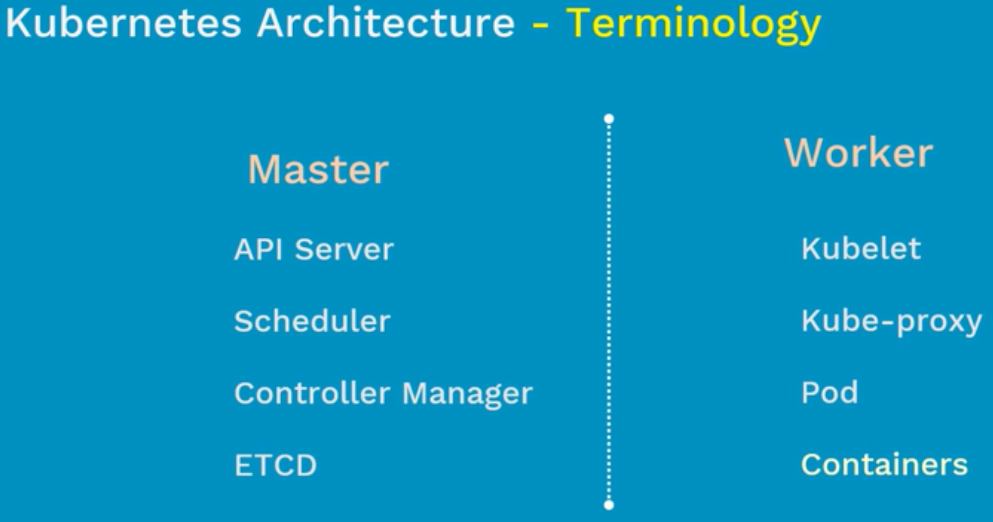
**Kubernetes architecture:**



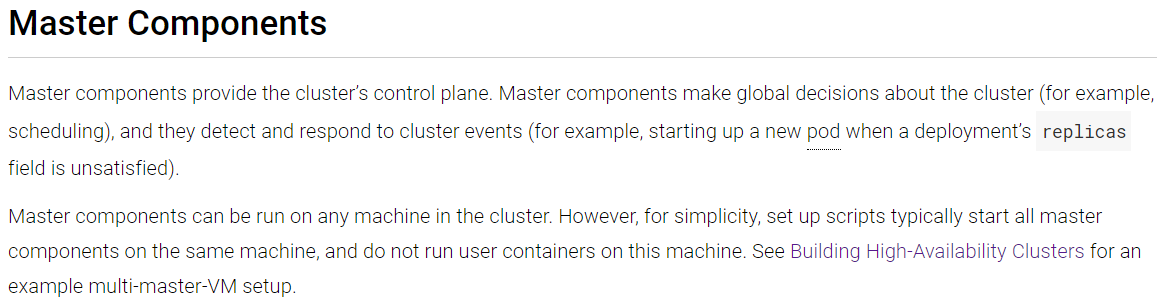




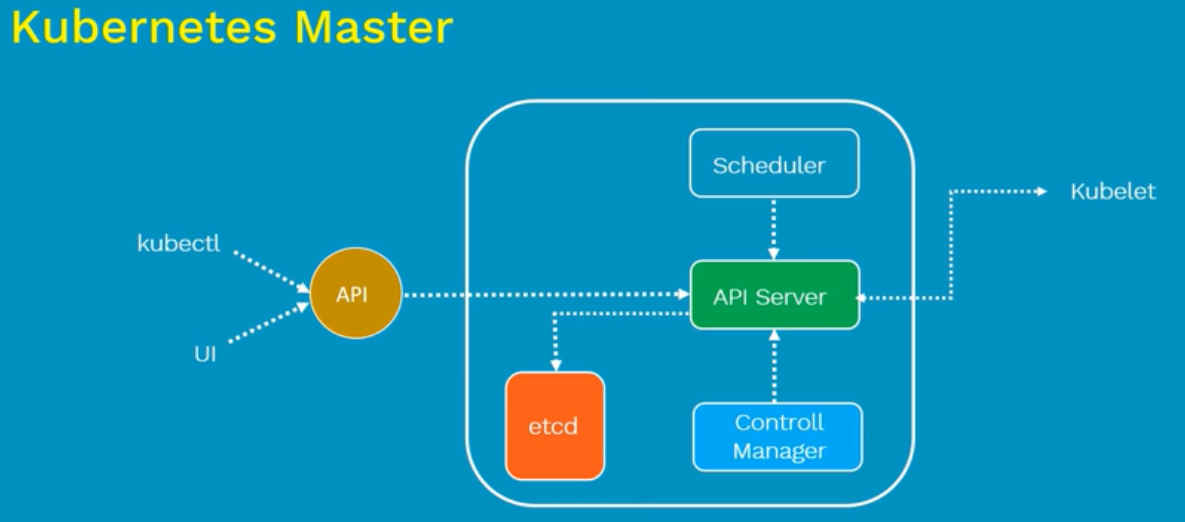


* When a node fails, kubernetes moves the pods to another node. Master is responsible for that
* When we want highly available, we will have multi master. But configuring this is little difficult as because we need to make the components replicate on other master node also. That is difficult

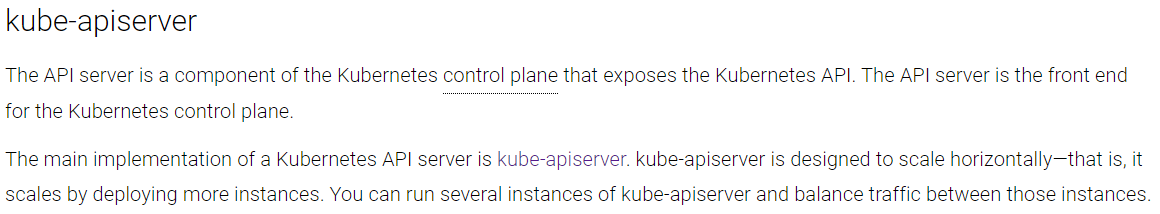
**Master components:**



* Master is one of the kubernetes component and it has some major components to work with
* If all these works, then kubernetes works
* **API server**
* **Cluster store or Key value store or etcd**
* **Kube controller**
* **Kube scheduler**

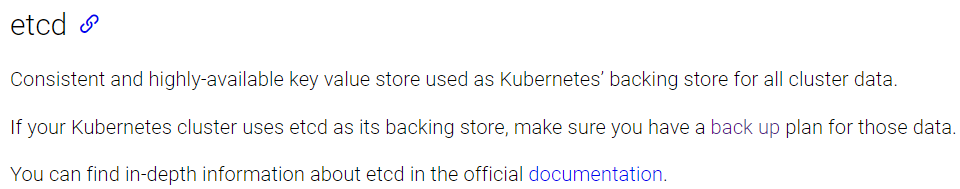


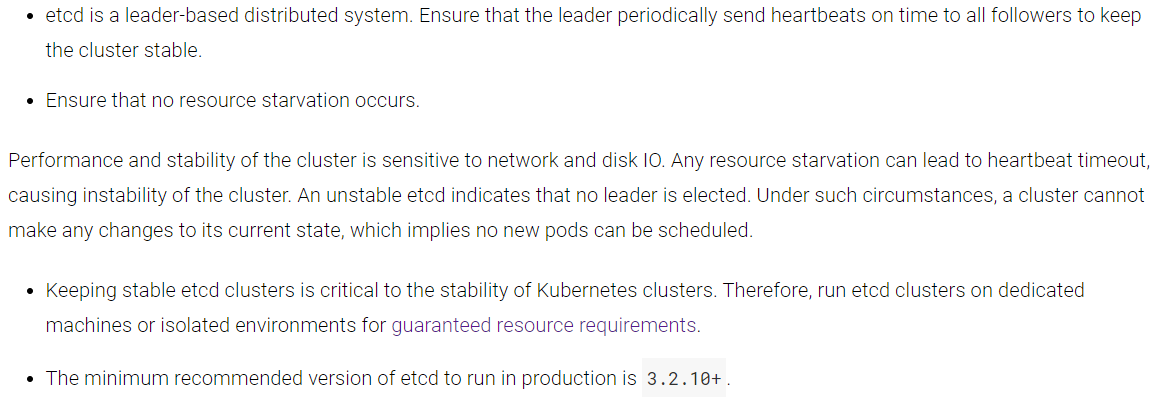
**API server:**



* API server is like a gatekeeper in master node. If we want to install, apply, view any object in the cluster. It goes through the API server
* API server validates and configures the API objects such as pods, services, replication controllers and deployments
* We interact with API server using kubectl. That is the only component which we expose

**Cluster Store:**





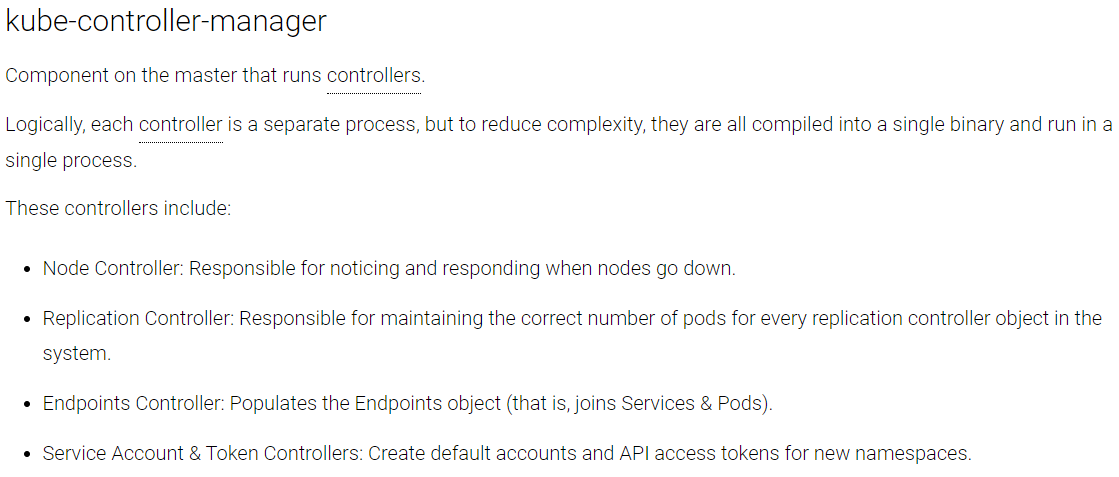
* Etcd is a key value lightweight database. It is developed by core OS
* It is a very popular key value database
* It is the central database store in kubernetes to store the current state of the cluster
* It stores the state of nodes. Whatever happening in nodes, it stores the data
* It is distributed key value storage
* We need to ensure we have a backup plan of cluster store

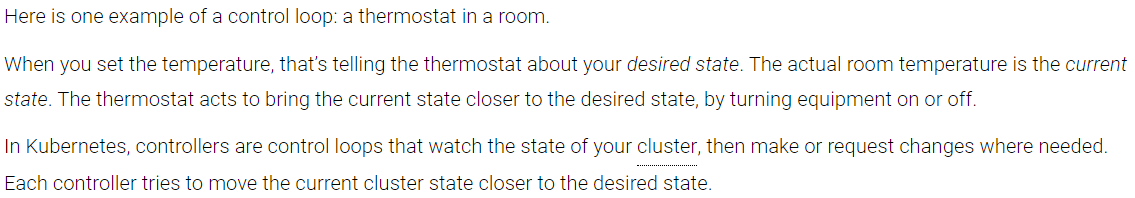
**Kube controller:**

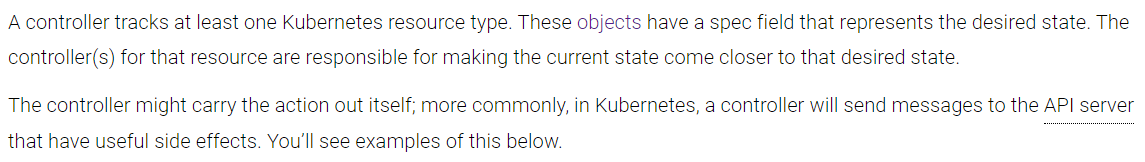
* For every activity we have controllers. It speaks to the nodes and get the work done
* It also maintains the desired state
* These controllers are responsible for the health of the clusters. It checks the nodes are running and correct pods are running as mentioned in specs file
* If there are 5 nodes and out of that 2 are free. Only controller has this info
* Api server first speaks to controller, and controller written the key values to cluster store and then it will go to scheduler

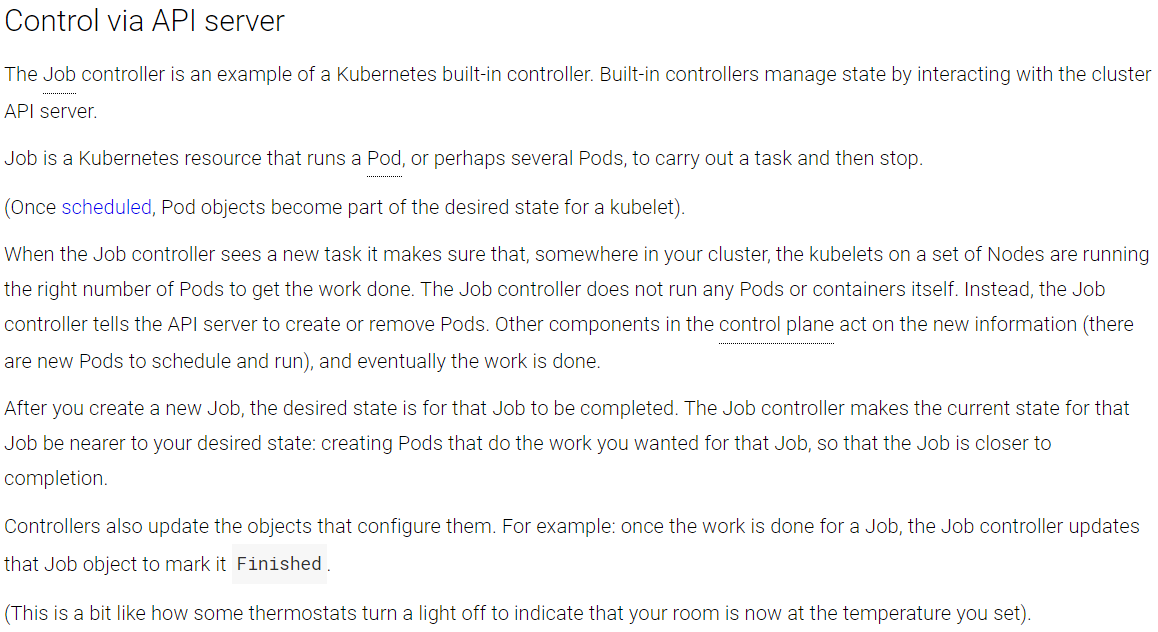
There are 4 types of control managers. They are

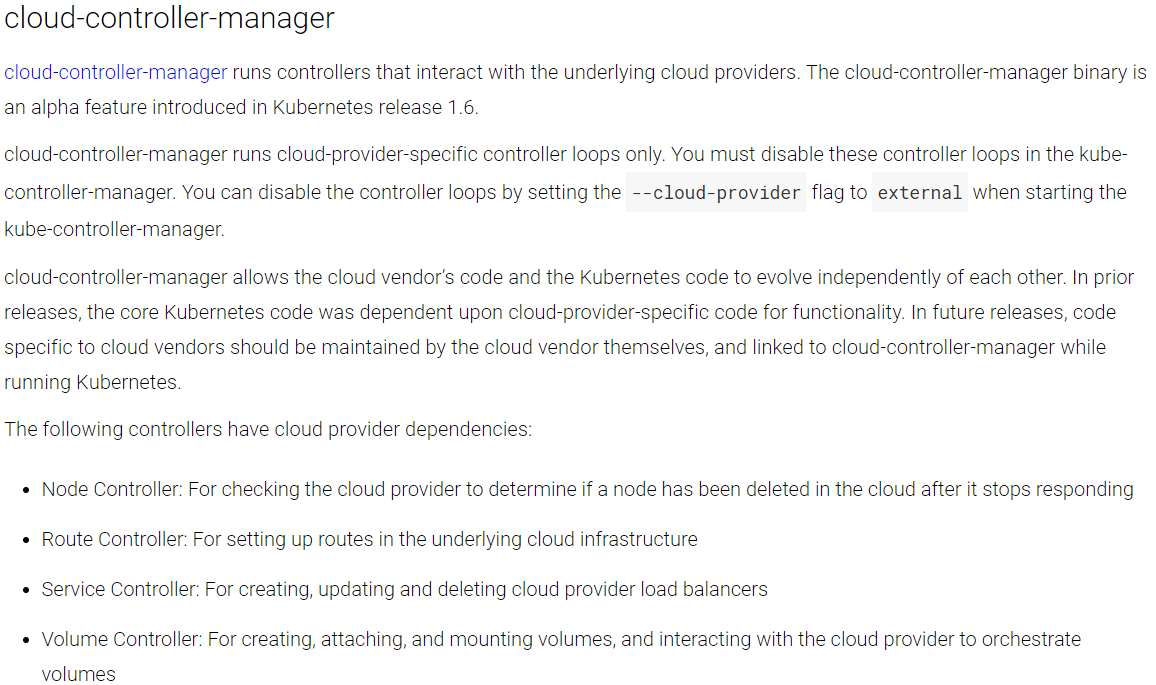
1. **Node controller**
2. **Replication controller**
3. **Endpoint controller and**
4. **Service account and token controllers**



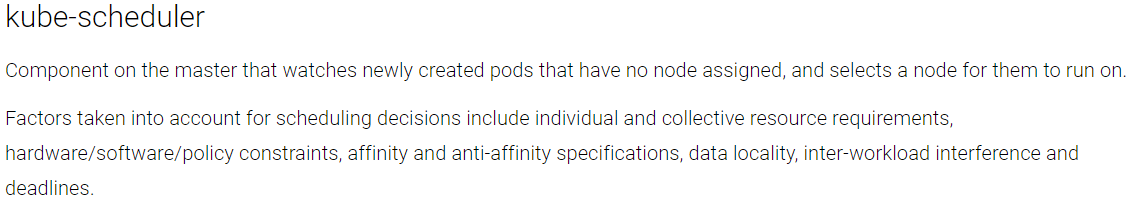








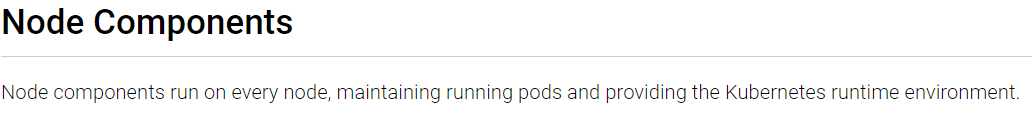
**Kube scheduler:**

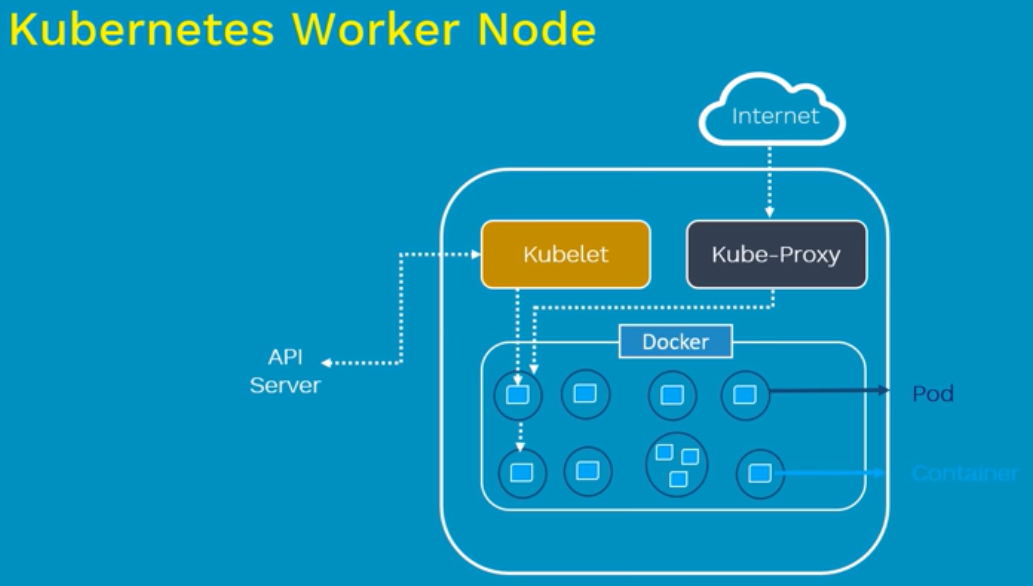


* Scheduler is responsible for scheduling the pods across nodes the

**For example**, if we instruct to create the pod with memory, images etc details. Then the scheduler will look for the available node and schedules the pod accordingly

**Worker node components:**



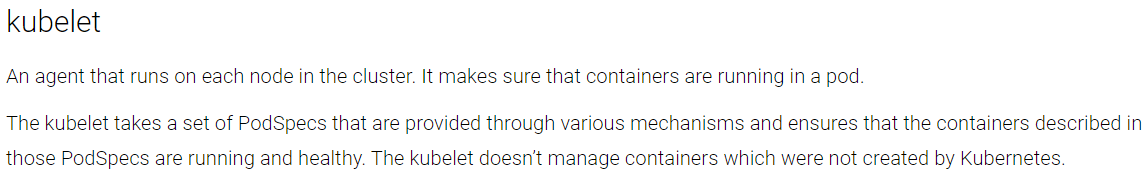


Node has 3 major components

* **Kube agent or kubelet**
* **Container engine or container runtime**
* **Kube proxy**

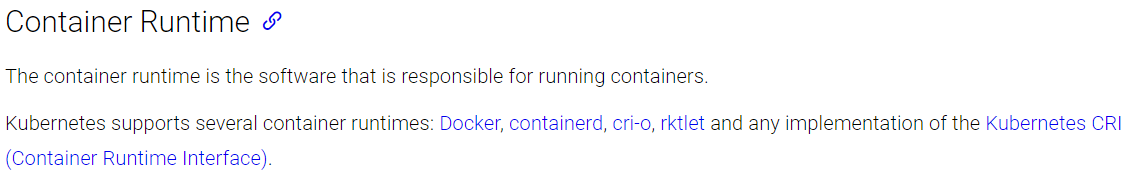
**Kubelet:**

* Kubelet is a primary node agent that runs on each worker node
* Its primary work is to look at the pod specs that was given to the API server and ensures that containers mentioned there are running healthy
* Incase kubelet finds any issue with the pods then it tries to restart the pods
* Incase if the issue is with worker node itself, then kubernetes master detects the node failure and it tries to recreate the pod on another healthy node. It all depends on if a pod is controlled by a replicaset or a replication controller. If none of these are behind the pod, then pod dies and will not get recreated. So, it’s better to use them
* It is like dummy agent, if it gives any work. It does it and report to master. If it can’t, it will report to master. It won’t take any decisions
* Whenever something wrong, it reports to master
* This is the component which scheduler schedules the job
* When request comes, it takes and interact with other components in the node



**Container Engine or container runtime:**

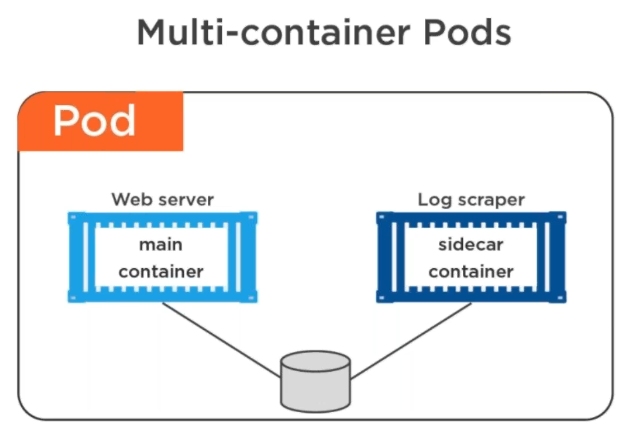
* This is the engine where we install docker. It is pluggable
* Docker will be in container under container engine component

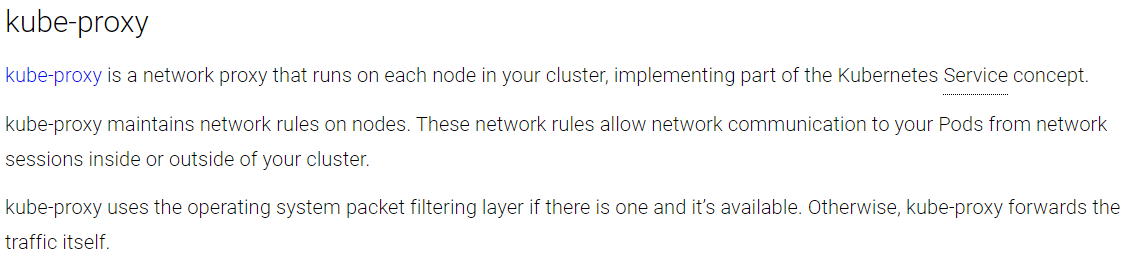


**Kube Proxy:**

Kubeproxy is critical element in cluster. It manages the network layer in node. It can also expose the service to outside the world

* The responsibility of kube proxy is to give the ip addresses to pods
* Pod will have one ip address even if there are 10 containers inside that
* If we have web and db server inside pod, for external world it will be as same ip
* Kube proxy also does load balancing
* Pod has some kernel namespaces and network stack as it gives different ip address to each pod
* When two containers present in same pod, they communicate with localhost because they both will have same ip address





**Note:**

* In most cases, master does not have any containers in it. It is to maintain the worker nodes
* Kubernetes supports up to 5000 worker nodes in one cluster
* Containers provides the run time environment for the application
* Generally, there used to be a single container in a pod and if there is a dependency between the pods, then it can also be useful to create two containers in a single pod
* Containers are designed to run microservices. It is not ideal to run monolithic applications in it
* Kubernetes nodes are also called as minions