Kubernetes Components and Architecture :

Kubernetes follows a client-server architecture. It's possible to have a multi-master setup (for high availability), but by default there is a single master server which acts as a controlling node and point of contact. The master server consists of various components including a kube-apiserver, an etcd storage, a kube-controller-manager, a kube-scheduler, and a DNS server for Kubernetes services. Worker Node components include kubelet and kube-proxy on top of Docker.
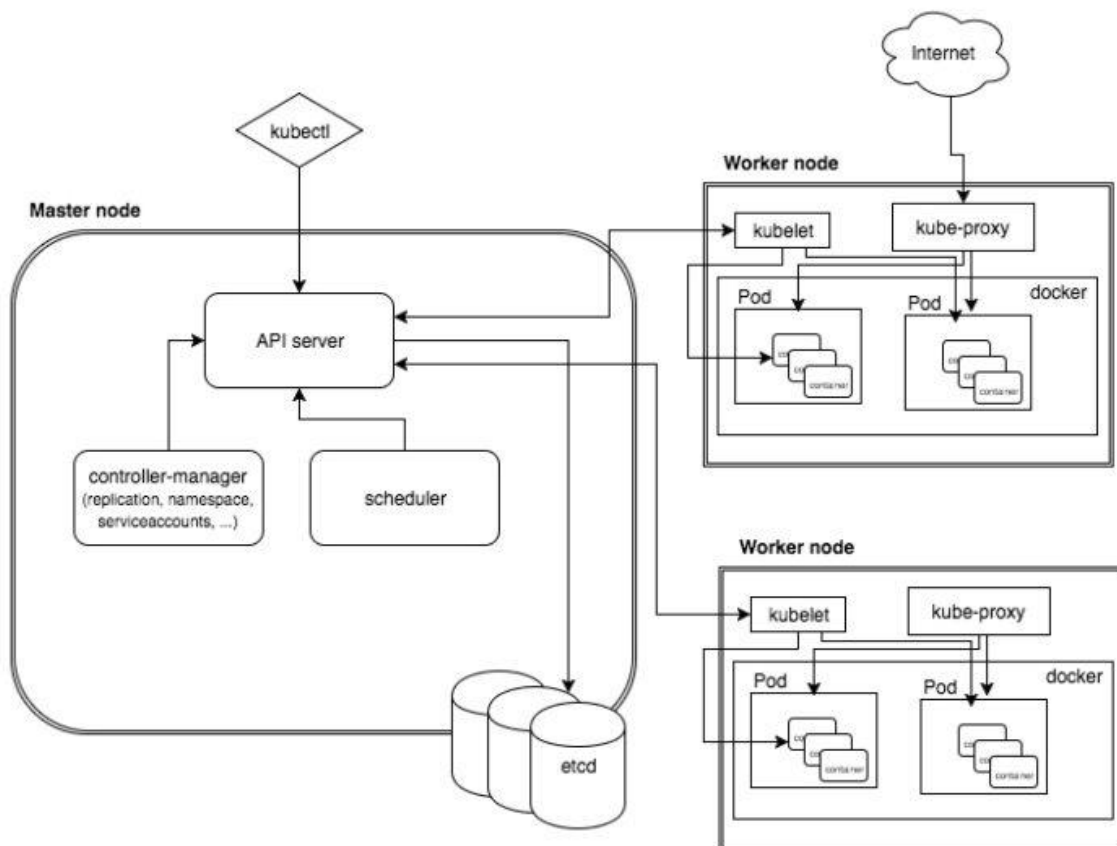
Master Node Components :

Below are the main components found on the master node:

kube-apiserver – Kubernetes API server is the central management entity that receives all REST requests for modifications (to pods, services, replication sets/controllers and others), serving as frontend to the cluster. Also, this is the only component that

communicates with the etcd cluster, making sure data is stored in etcd and is in agreement with the service details of the deployed pods.

kube-controller-manager
Control Plane component that runs controller processes.
Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

Some types of these controllers are:

Node controller :
Responsible for noticing and responding when nodes go down.

Job controller | Replication Controller :
Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.

Endpoints controller :
Populates the Endpoints object (that is, joins Services & Pods).

Service Account & Token controllers :
Create default accounts and API access tokens for new namespaces

kube-scheduler – helps schedule the pods (a co-located group of containers inside which our application processes are running) on the various nodes based on resource utilization. It reads the service's operational requirements and schedules it on the best fit node. For example, if the application needs 1GB of memory and 2 CPU cores, then the pods for that application will be scheduled on a node with at least those resources. The scheduler runs each time there is a need to schedule pods. The scheduler must know the total resources available as well as resources allocated to existing workloads on each node.

etcd cluster – a simple, distributed key value storage which is used to store the Kubernetes cluster data (such as number of pods, their state, namespace, etc), API objects and service discovery details. It is only accessible from the API server for security reasons. etcd enables notifications to the cluster about configuration changes with the help of watchers. Notifications are API requests on each etcd cluster node to trigger the update of information in the node's storage.

Considerations for large clusters :

A cluster is a set of nodes (physical or virtual machines) running Kubernetes agents, managed by the control plane. Kubernetes v1.21 supports clusters with up to 5000 nodes. More specifically, Kubernetes is designed to accommodate configurations that meet all of the following criteria:

No more than 100 pods per node
No more than 5000 nodes
No more than 150000 total pods
No more than 300000 total containers
You can scale your cluster by adding or removing nodes. The way you do this depends on how your cluster is deployed

What is a Pod?

Pods are the smallest, most basic deployable objects in Kubernetes. A Pod represents a single instance of a running process in your cluster.

Pods contain one or more containers, such as Docker containers. When a Pod runs multiple containers, the containers are managed as a single entity and share the Pod's resources. Generally, running multiple containers in a single Pod is an advanced use case.

Pods also contain shared networking and storage resources for their containers.