**10/03/2025**

**Kubernetes**

It is an orchestration tool which run containers across multiple hosts.

On the master node there are 4 components working.

In the Master architecture there are 4 components

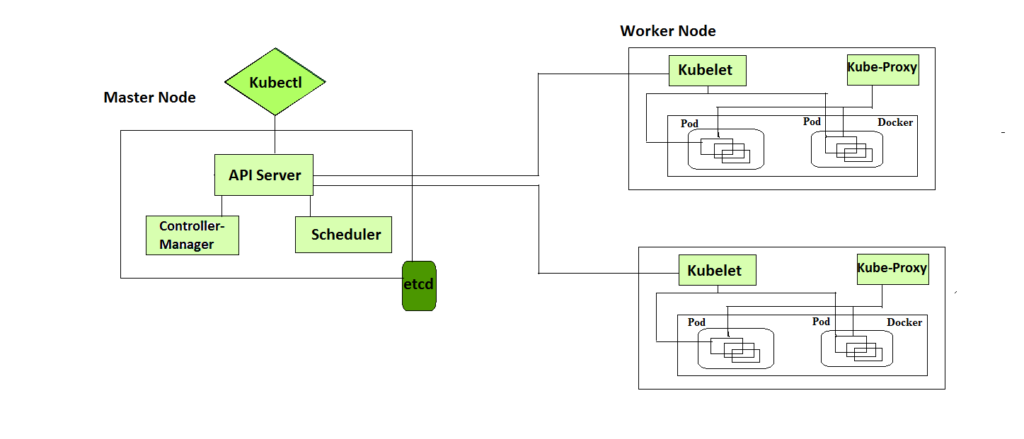
1. **API Server**
2. **Scheduler**
3. **Controller**

**The above three don’t store data**

1. ETCD Cluster: it is only the one which can store the data.

* When master node wants to talk to worker node, it directly talk. Because it is aware of all the loads of all nodes.
* When worker node wants to talk to master, talk through load balancer.

**Architecture:**

****

**Master node:** Manage , Monitor, Schedule

             Kube-Scheduler

             Kube-Controller

             Kube-apiserver

             ETCD cluster

Whenever u want to execute a command done through Kubectl. It will convert your request to REST request. API server will authenticaticate (Are u a real user or not), authorize, validate and talk to ETCD.

**Worker node:**

             Kubelet

             Kube-proxy

             Containerd/ CRIO(CRI) / pods

API Server checks the below functionalities:

1. Authentication
2. Authorization
3. Validation
4. Talks to ETCD

**ETCD:** No SQL Database.

The use of No SQL Database is scalability.

**Scheduler:**

When there is an entry in the table. The scheduler will come up and will run an algorithm to check which is the best machine to run the pod. It talks with the ETCD and find the best machine. The first it should talk with the API server and then API Server talks with ETCD. Only API Server can talk with ETCD.

**Controller:**

It controls everything. If the actual state is does not equal to desired state then the controller comes in to picture and runs.

**Different types of controller:**

1. **Replication**
2. **Token**
3. **Node controller:** Responsible for noticing and responding when nodes go down.
4. **Job controller:** Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.
5. **EndpointSlice controller:** Populates EndpointSlice objects (to provide a link between Services and Pods).
6. **ServiceAccount controller:** Create default ServiceAccounts for new namespaces.

**There are two main components of controller:**

1. **Informer/Shared Informer:** Watches for changes on the current state of the Kubernetes objects and sends events to workqueue.
2. **Workqueue:** Receives events from informer/shared informer

Pop up events and process.

**Worker Components:**

**Kubelet:** Is like local manager on the node

**Kubeproxy:** It runs on every worker node and understands the networking of every node

**Container Runtime / Containerd / CRIO:**

**11/03/2025**

**Kubelet and containerd does not run as a pod in Kubernetes architecture**

Kubelet will pass the necessary information of the node to the master node. It tells to the ETCD through API server.

All objects in Kubernetes are persistent entities and represent the state of the cluster.

**Pod:** The smallest deployable units in Kubernetes are pods**.**Group of one or more containers. It is recommended to run only one container in a pod.

All the containers in the pod are colacted (run on the same node). Co scheduled (come up together), share namespaces.

**Side car:** If one container is dependent on the main machine then you can keep multiple containers in the same machine.

The containers in the same pod share the below namespaces:

1. File Systems
2. IPC
3. Network

**Service:** Defines a set of pods and a policy by which to access them. It provides a stable IP address and DNS name, enabling other applications to discover and communicate with services.

**Creation of POD:**

1. **Kubectl makes an Rest API request to the API server. It passes the certificates and passes the request such as get, post,put, delete.**
2. **API server authenticate, authorize, Validate, talks to ETCD.**
3. **Scheduler comes and schedule the pod in a particular node**
4. **Kubelet will get the pod definition from API server**
5. **Kubelet instruct the container runtime**
6. **POD networking setup**
7. **POD Becomes available**

| **Step** | **Component** | **Role** |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 1 | kubectl | Sends the request to the API server |

|  |  |  |
| --- | --- | --- |
| 2 | API Server | Validates request & updates etcd |

|  |  |  |
| --- | --- | --- |
| 3 | etcd | Stores pod definition |

|  |  |  |
| --- | --- | --- |
| 4 | Scheduler | Assigns pod to a node |

|  |  |  |
| --- | --- | --- |
| 5 | Kubelet | Pulls pod definition & manages it |

|  |  |  |
| --- | --- | --- |
| 6 | Container Runtime | Pulls & runs the container |

|  |  |  |
| --- | --- | --- |
| 7 | CNI Plugin | Configures pod networking |

|  |  |  |
| --- | --- | --- |
| 8 | Kube Proxy | Handles networking & services |

**To create a pod:**

Kubectl run podname –image=image\_name

**To get pods:**

Kubectl get pods

**To view a list of all the nodes in the cluster:**

Kubectl get nodes

**To view a list of all the services in the cluster**

Kubectl get services

**Pod Lifecycle:**

Pods go through various phases:

Pending,

Running,

Succeeded,

Failed,

Unknown.

**Kubernetes Installation:**

1. Kubeadm – Tool

To setup Kubernetes cluster.

**Other Tools to setup Kubernetes installation:**

1. Minikube
2. Kind

1.Installation of container runtime on all nodes

2.Install kubeadm, kubelet, kubectl on all nodes

3.initiate the kubeadm control plane configuration on master node

4.saving the node join command

5. Installing the calico network plugin

**13/03/2025**

**Pod**

**Replica set**

**In the spec if we see a template then it talks about the pod**

**In replica set template is mandatory and selector should be there which should match with the matchLabels and labels in the template.**

**Namespaces:**

**Default: it is the namespace for objects with no other namespace**

**Kube-system: It is namespace for objects created by the Kubernetes system**

**Kube-public: it is reserved for cluster usage. It can be used for objects which needs to be shared across namespaces.**

**Kube-node-lese: It is a namespace for node specific objects.**

**Creation of namespace:**

kubectl create namespace <insert-namespace-name-here>

**Deletion of namespace:**

kubectl delete namespaces <insert-some-namespace-name>

**Labels**

kubectl get pods -l env=development

**Deployments:**

**Usecases:**

1. **Scale out the Deployment to facilitate more load.**
2. **Pause the Deployment to apply multiple fixes to its PodTemplateSpec and then resume it to start a new rollout.**
3. **Use the status of the Deployment as an indicator that a rollout has stuck.**

kubectl get deploy

kubectl describe deploy <id>

kubectl rollout status deployment.v1.apps/nginx-deployment

kubectl get pods

kubectl --record deployment.v1.apps/nginx-deployment set image deployment.v1.apps/nginx-deployment nginx=nginx:1.9.1

kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1 --record

kubectl rollout history deployment.v1.apps/nginx-deployment

kubectl rollout undo deployment.v1.apps/nginx-deployment