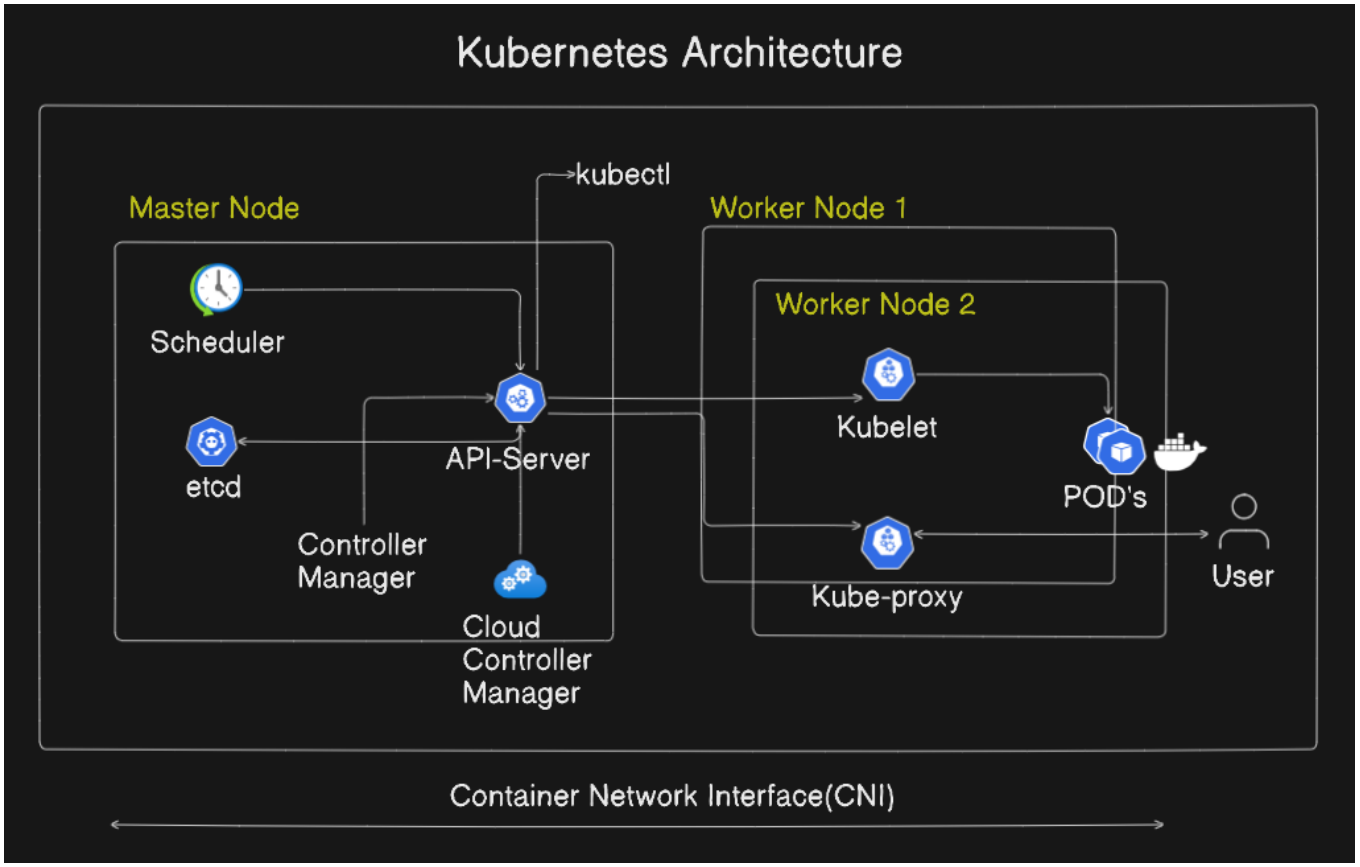
# Kubernetes(k8s) Architecture

Ojas Jawale




**KEY TAKEAWAYS:**

- Kubernetes clusters include a control plane (master), worker nodes, and etcd for cluster state consistency, each with specific roles.
- The control plane manages operations and decision-making via components like kube-api server, kube-controller-manager, kube-scheduler, and cloud-controller-manager.
- Nodes are crucial components where pods are executed. Worker nodes run user-defined containers, while master nodes handle control plane operations.
- Kubernetes operates on a desired state model, where users define the state of objects through APIs, and Kubernetes continuously strives to match this desired state with the actual state of running containers.

**Components of a Kubernetes Cluster:**

- Below is a deeper dive into not only nodes, but each of the various components that you'll need to have a complete and working Kubernetes cluster.

**The Control plane (master)**

- The Control plane is made up of the kube-api server, kube scheduler cloud-controller-manager and kube-controller-manager. Kube proxies and kubelets live on each node, talking to the API and managing the workload of each node.
- As the control plane handles most of Kubernetes' 'decision making', nodes which have these components running generally don't have any user containers running - these are normally named as master nodes.

**Cloud-controller-manager**

- The cloud-controller-manager runs in the control plane as a replicated set of processes (typically, these would be containers in Pods). The cloud-controller-manager is what allows you to connect your clusters with the cloud provider's API and only runs controllers specific to the cloud provider you're using.
- Just note: If you're running Kubernetes on-premise, your cluster won't have a cloud-controller-manager.

**Etcd**

- etcd is a distributed key-value store and the primary datastore of Kubernetes. It stores and replicates the Kubernetes cluster state.
- To run etcd, you first need to have a Kubernetes cluster and the command-line tool configured to communicate with said cluster. If you don't already have a Kubernetes cluster, follow our tutorial to play around with Kubernetes and create one.

**Kubelet**

- The kubelet functions as an agent within nodes and is responsible for the runnings of pod cycles within each node. Its functionality is watching for new or changed pod specifications from master nodes and ensuring that pods within the node that it resides in are healthy, and the state of pods matches the pod specification.

**Kube-proxy**

- Kube-proxy is a network proxy that runs on each node. It maintains network rules, which allow for network communication to Pods from network sessions inside or outside of a cluster. Kube-proxy is used to reach **kubernetes services in addition to load balancing of services**.

**Kube-controller-manager**

The Kubernetes controller manager is a collection of controllers bundled within a single binary and run in a single process. The following controllers are present within this manager:

- **NODE CONTROLLER:** Responsible for identifying changes in nodes within the cluster
- **REPLICATION CONTROLLER:** Responsible for maintaining replications of objects in the cluster (such as replicasets)
- **ENDPOINT CONTROLLER**: Responsible for provisioning of endpoints (such as service endpoints)

Service account and token controllers: Responsible the management of service accounts within each namespace, as well as API access tokens.

**Kube-API Server**

- The Kube-API server, a component within the control plane, validates and configures data for API objects, including pods and services. The API Server provides the frontend to the cluster's shared state, which is where all of the other components interact.
- This means that any access and authentication such as deployments of pods and other Kubernetes API objects via kubectl are handled by this API server.

**Kube-scheduler**

- Running as part of the control plane, the kube-scheduler’s responsibility is to assign pods to nodes within your cluster. The scheduler will use information such as compute requests and limits defined within your workload (if any), as well as finding the right node candidate based on its available resources to appropriately assign your workload to a particular node.

**Node**

Kubernetes runs workloads by placing your pods into nodes. Depending on your environment, a node can represent a virtual or physical machine. Each node will contain the components necessary to run pods. There are two distinct classifications of a node within Kubernetes: Master and Worker nodes.

- **WORKER NODES:** Worker nodes will have an instance of kubelet, kube-proxy and a certain container runtime (such as Docker, containerd) running on each node, and are used to run user defined containers. These nodes are managed by the control plane.
- **MASTER NODES:** A master node, or sometimes called control-plane nodes will have the control plane binaries bootstrapped, and will only be responsible for all components within the control plane, such as runnings of etcd and the kube-api-server. In order to achieve high availability with etcd to establish a quorum, there are normally more than 3 master nodes within a cluster.

**Desired state in Kubernetes (vs actual state)**

- Desired state is a core concept of Kubernetes. It means that, through a declarative or an imperative API, you describe the state of the objects that will run your containers.
- Kubernetes is able to handle constant change and will continuously try hard to match your desired state with your actual state, which is how your containers are actually running. You might not potentially ever reach your desired state, but it doesn't matter because the controllers for your cluster should be constantly running and working to fix and recover from errors, if possible.

---

Ojas Jawale