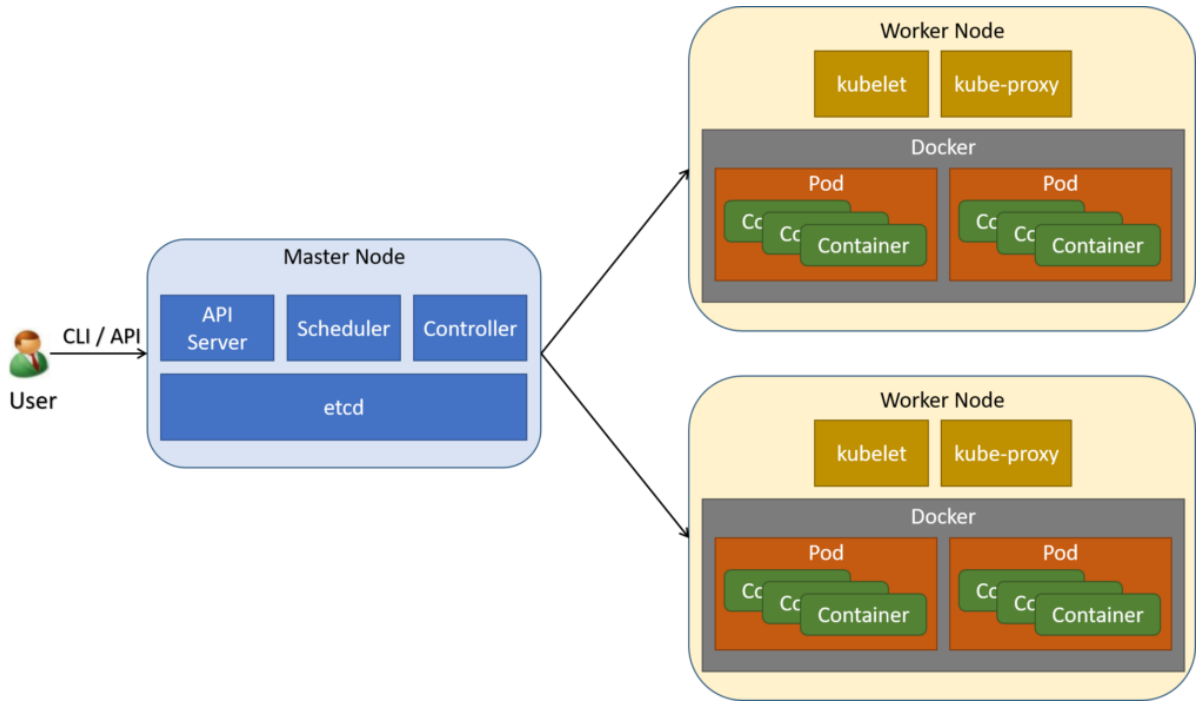
KUBERNETES ARCHITECTURE

Kubernetes architecture is composed of several components that work together to provide a robust container orchestration platform. These components are designed to manage and automate the deployment, scaling, and operation of containerized applications.



1. **Master Node (control plane):**

   - **API Server**: The central management component that exposes the Kubernetes API and is responsible for accepting and processing commands from users, the command-line interface (CLI), and the Kubernetes Dashboard.

- **etcd**: A distributed key-value store that stores the configuration data for the entire cluster, including the state of all Kubernetes objects and cluster settings.
- **Controller Manager**: A set of controllers that handle various tasks, such as replicating pods, scaling deployments, and managing endpoints.
- **Schedule**r: Assigns nodes for newly created pods based on resource requirements, affinity/anti-affinity rules, and other constraints.
- **Cloud Controller Manager (optional):** Interfaces with the underlying cloud provider's APIs to manage cloud-specific resources.
- Examples include managing Load Balancers, Persistent Volumes, and Route53 records on AWS, Azure, or GCP.This component is optional and only required if you are running Kubernetes in a cloud environment.

2. **Worker node (Data plane)**:

- **Kubelet**: An agent that runs on each node and communicates with the API server. It ensures that containers are running in a Pod and reports the node's status back to the control plane.

- **Kube Proxy**: Maintains network rules on nodes, allowing network communication to and from Pods.
- **Container Runtime**: The software responsible for running containers, such as Docker or containers.