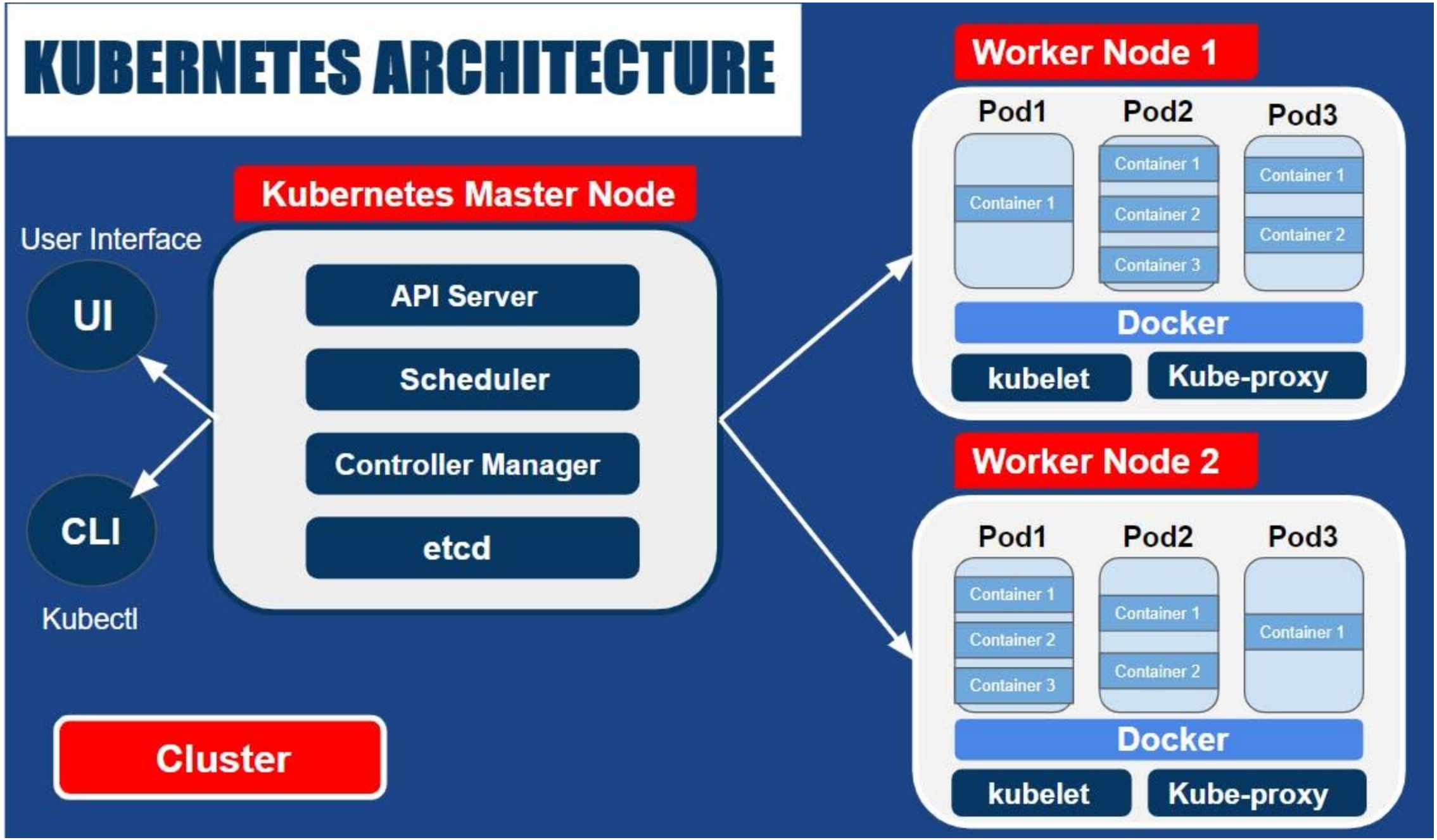
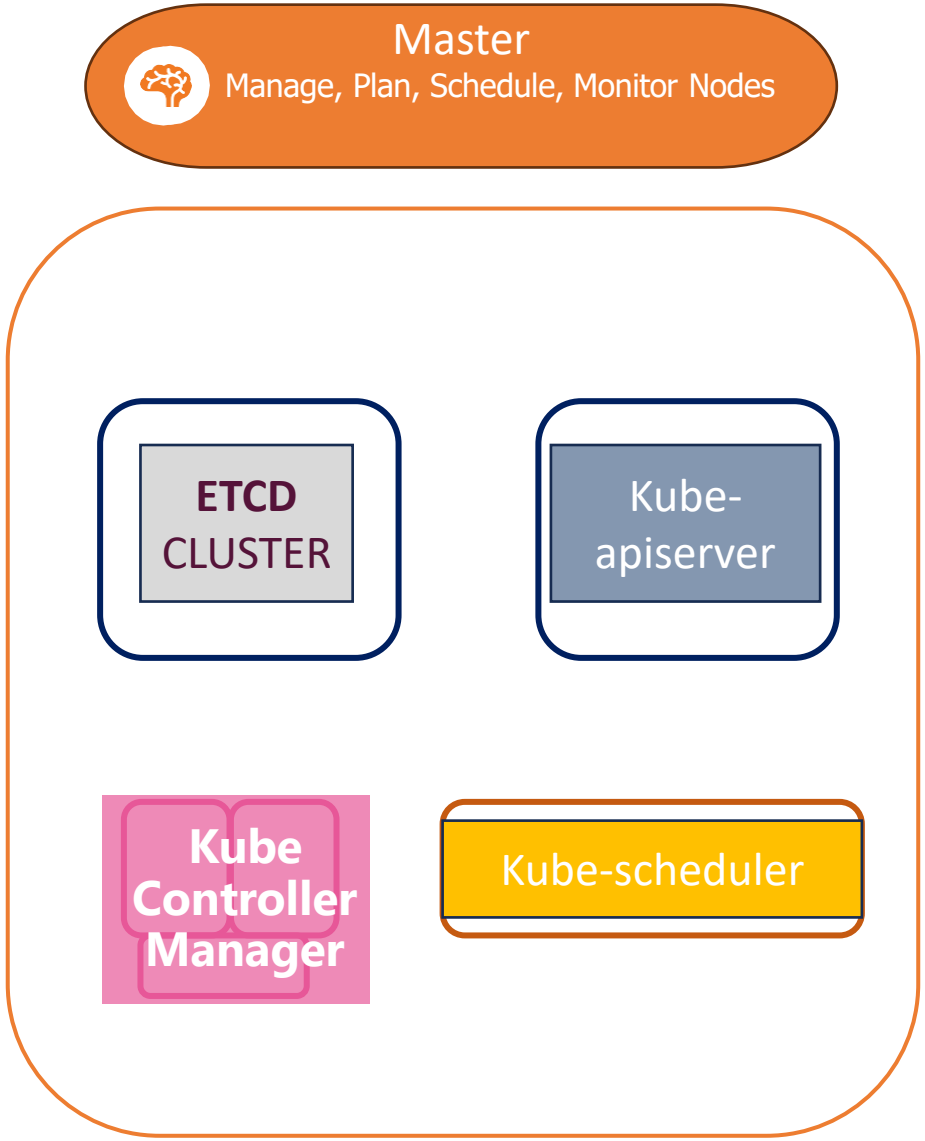
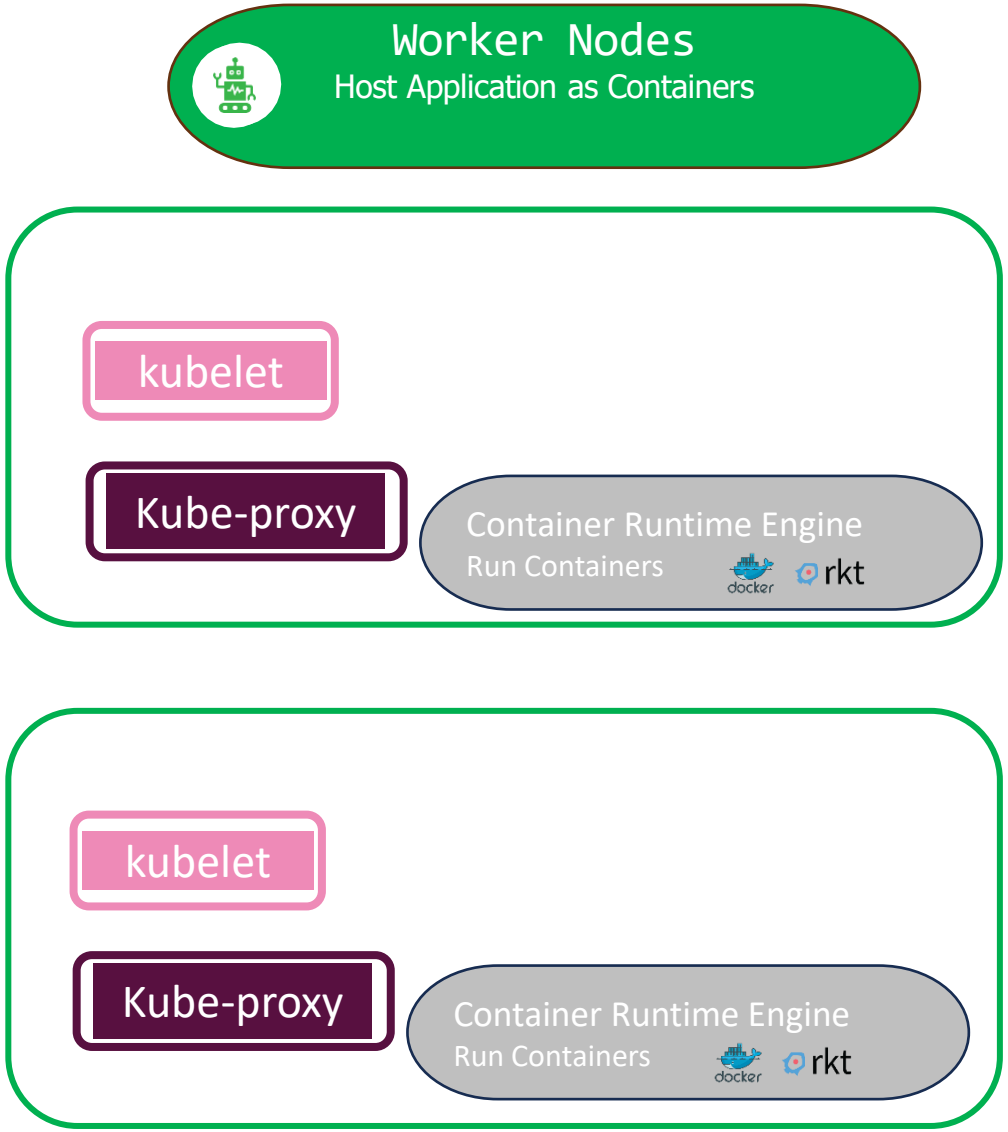

KUBERNETES ARCHITECTURE
Kubernetes Master Node
User Interface
UI
CLI
Kubectl
API Server
Scheduler
Controller Manager
etcd
Cluster
Worker Node 1
Pod1
Pod2
Pod3
Container 1
Container 1
Container 2
Container 3
Container 1
Container 2
Docker
kubelet
Kube-proxy
Worker Node 2
Pod1
Pod2
Pod3
Container 1
Container 2
Container 3
Container 1
Container 2
Container 1
Docker
kubelet
Kube-proxy

Kubernetes Architecture





Master Node Components:

1.Kube API Server:

1. This is the central control point for the entire Kubernetes cluster.
2. It receives requests (like creating, deleting, or updating pods) from users or other components and processes them.
3. It's responsible for communication between all the Kubernetes components.

2.Kube Scheduler:

1. Decides which worker node to assign a new pod to, based on factors like resource availability (CPU, memory, etc.) and policies.
2. Ensures workloads are spread evenly across nodes for optimal performance.

3.Controller Manager:

1. Oversees various controllers that manage different parts of the cluster, like deployments, replicas, and nodes.
2. Makes sure the current state of the cluster matches the desired state (e.g., ensuring the specified number of pod replicas are running).

4. etcd:

1. A distributed key-value store that stores all the data about the Kubernetes cluster, including configurations, secrets, and states.
2. Acts as the "brain" of Kubernetes, holding all its critical information securely and reliably.

Worker Node Components:

1.Kubelet:

1. The main agent that runs on each worker node.
2. It ensures the containers described in the pod specifications are running and healthy on that node.
3. Constantly communicates with the API server to receive instructions and report back status.

2. Kube Proxy:

- Manages network rules on each node.
- Handles routing and load balancing to ensure traffic reaches the correct pods, even if they move between nodes.
- Helps manage communication within the cluster and between services.

3. Container Runtime Engine:

- The software that actually runs the containers.
- It can be Docker, containerd, or other container runtimes that Kubernetes supports.
- Responsible for pulling container images, starting, and managing containers on the node.

4. Docker (or Other Container Runtime):

- A specific type of container runtime engine that is widely used in Kubernetes.
- It packages applications and their dependencies into containers, ensuring they run consistently on any system.