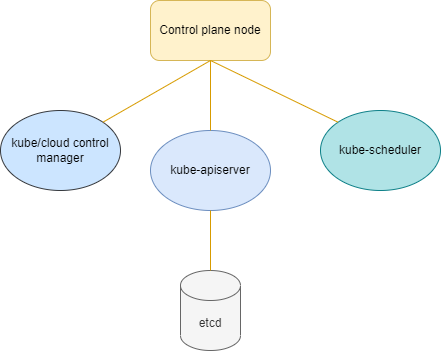
# **Kubernetes Architecture and Components**

**Kubernetes architecture for single node and multi-node:**

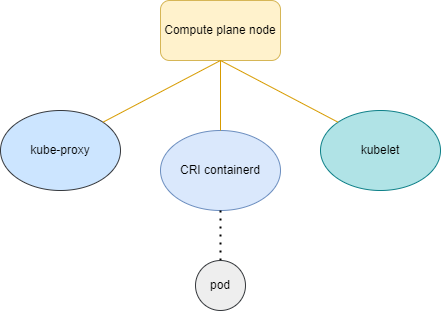
**Note**: Single node implies both control plane components and compute plane components run on the same node.

Kubernetes architecture has two types of nodes i.e., control plane node and compute plane node.

Control plane node consists of four main components:



Similarly, compute plane node consists of three components:

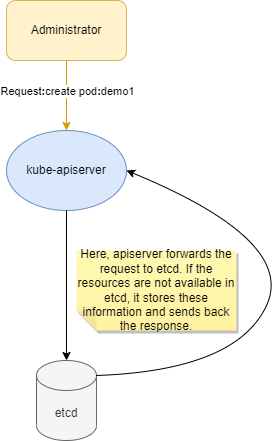


**Note**: These components from compute plane nodes are also available in control plane nodes but talking about the functionality we don’t use them in multi-node setups.

**Note**: Pod is a component in kubernetes, it is usually a group of containers.

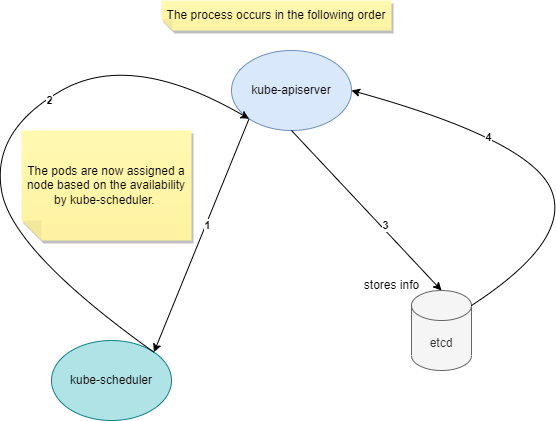
Now, let us see the functionality of the nodes in detail.

In case we as an administrator are trying to launch a pod, our request will be handled by kube-apiserver.



This process of verifying the existence of the resource and updating the information based on it is called acknowledgement.

Hereafter, the response from etcd is sent to kube-scheduler which assigns a node to the pods and sends the information back to apiserver. This information is again passed to etcd and it does the acknowledgement again.



Based on the response from etcd, apiserver launches a pod in the assigned node from scheduler.

Now, the remaining component control manager monitors the whole cluster. It checks for any mismatches or status of pods, etc., and updates the information to apiserver. This information is now passed to etcd.

**Note**: etcd is a distributed key-value store used to manage and store the critical information to keep running the systems. etcd is where the k8s(kubernetes) objects are persisted(including pods).

**KeyPoint**: In the control plane node, components can not talk to each other without apiserver. Apiserver acts as the middleman.

Coming to compute plane node, the information passed by apiserver is received by kubelet and then kubelet talks to CRI(container runtime interface) which creates a pod in the backend.

Overall structure of the k8s cluster:

