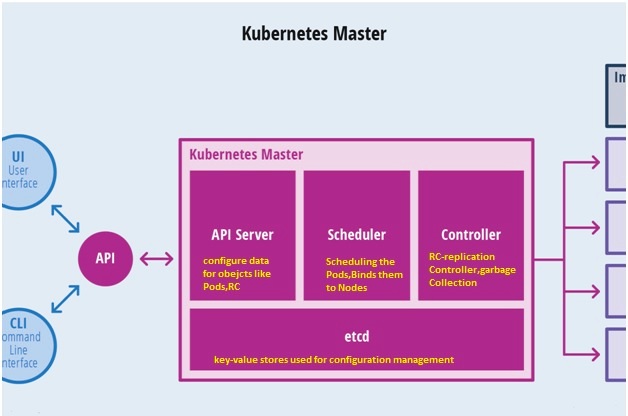
**Kubernetes Master Components**

The Kubernetes master runs the Scheduler, Controller Manager, API Server and etcd components and is responsible for managing the Kubernetes cluster. Essentially, it’s the brain of the cluster



### API Server

When you interact with your Kubernetes cluster using the kubectl command-line interface, you are actually communicating with the master API Server component.

The API Server is the main management point of the entire cluster. In short, it processes REST operations, validates them, and updates the corresponding objects in etcd. The API Server serves up the Kubernetes API and is intended to be a relatively simple server, with most business logic implemented in separate components or in plugins.

The API Server is the only Kubernetes component that connects to etcd; all the other components must go through the API Server to work with the cluster state.

The API Server is also responsible for the authentication and authorization mechanism. All API clients should be authenticated in order to interact with the API Server.

The API Server also implements a watch mechanism (similar to etcd) for clients to watch for changes. This allows components such as the Scheduler and Controller Manager to interact with the API Server in a loosely coupled manner.

### Controller Manager

The Kubernetes Controller Manager is a daemon that embeds the core control loops (also known as “controllers”) shipped with Kubernetes. Basically, a controller watches the state of the cluster through the API Server watch feature and, when it gets notified, it makes the necessary changes attempting to move the current state towards the desired state. Some examples of controllers that ship with Kubernetes include the Replication Controller, Endpoints Controller, and Namespace Controller.

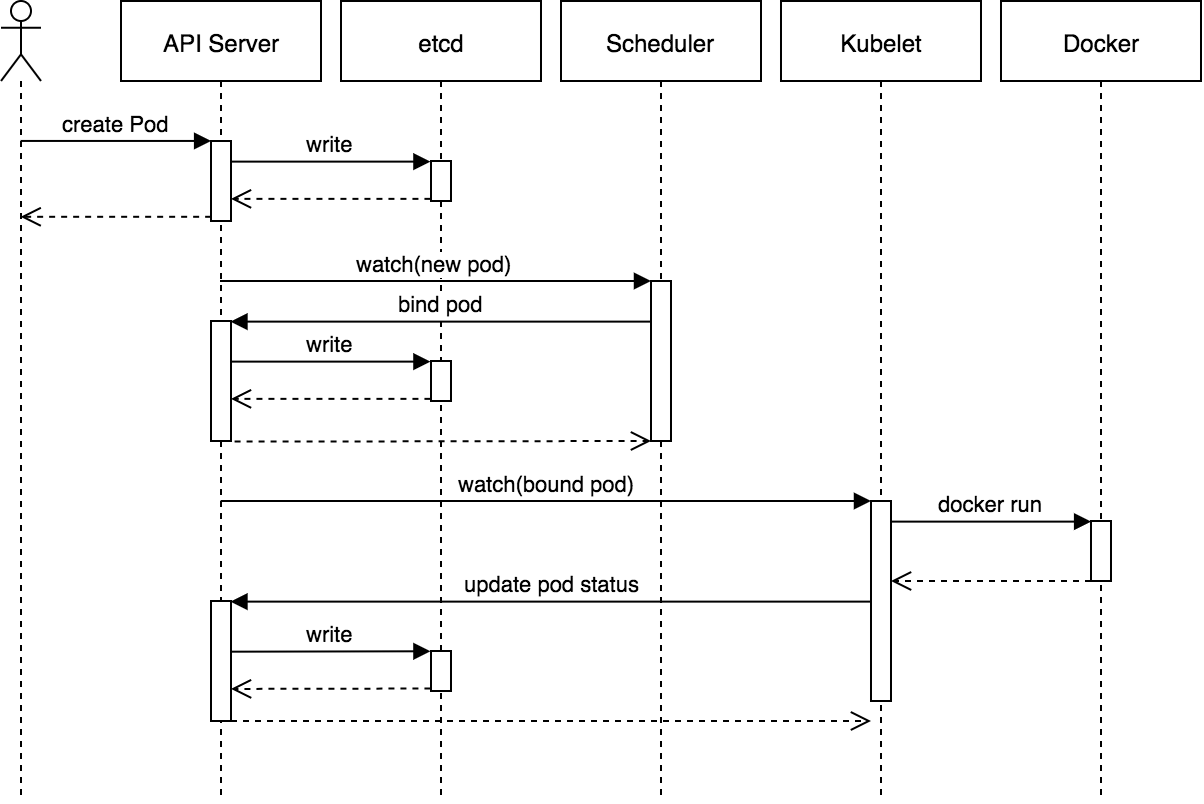
### Scheduler (scheduling the Pod)

The Scheduler watches for unscheduled pods and binds them to nodes via the /binding pod subresource API, according to the availability of the requested resources, quality of service requirements, affinity and anti-affinity specifications, and other constraints. Once the pod has a node assigned, the regular behavior of the Kubelet is triggered and the pod and its containers are created

### Etcd

[Etcd](https://github.com/coreos/etcd) is a distributed, consistent key-value store used for configuration management, service discovery, and coordinating distributed work.

When it comes to Kubernetes, etcd reliably stores the configuration data of the Kubernetes cluster, representing the state of the cluster (what nodes exist in the cluster, what pods should be running, which nodes they are running on) at any given point of time.



1. kubectl writes to the API Server.
2. API Server validates the request and persists it to etcd.
3. etcd notifies back the API Server.
4. API Server invokes the Scheduler.
5. Scheduler decides where to run the pod on and return that to the API Server.
6. API Server persists it to etcd.
7. etcd notifies back the API Server.
8. API Server invokes the Kubelet in the corresponding node.
9. Kubelet talks to the Docker daemon using the API over the Docker socket to create the container.
10. Kubelet updates the pod status to the API Server.
11. API Server persists the new state in etcd.

