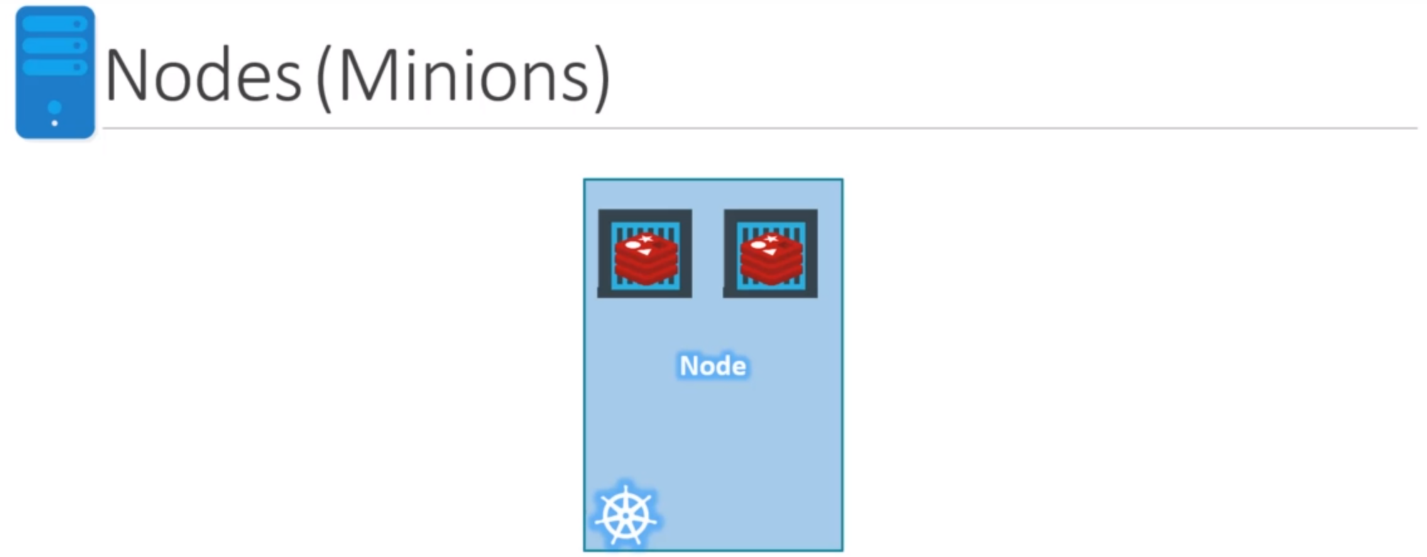
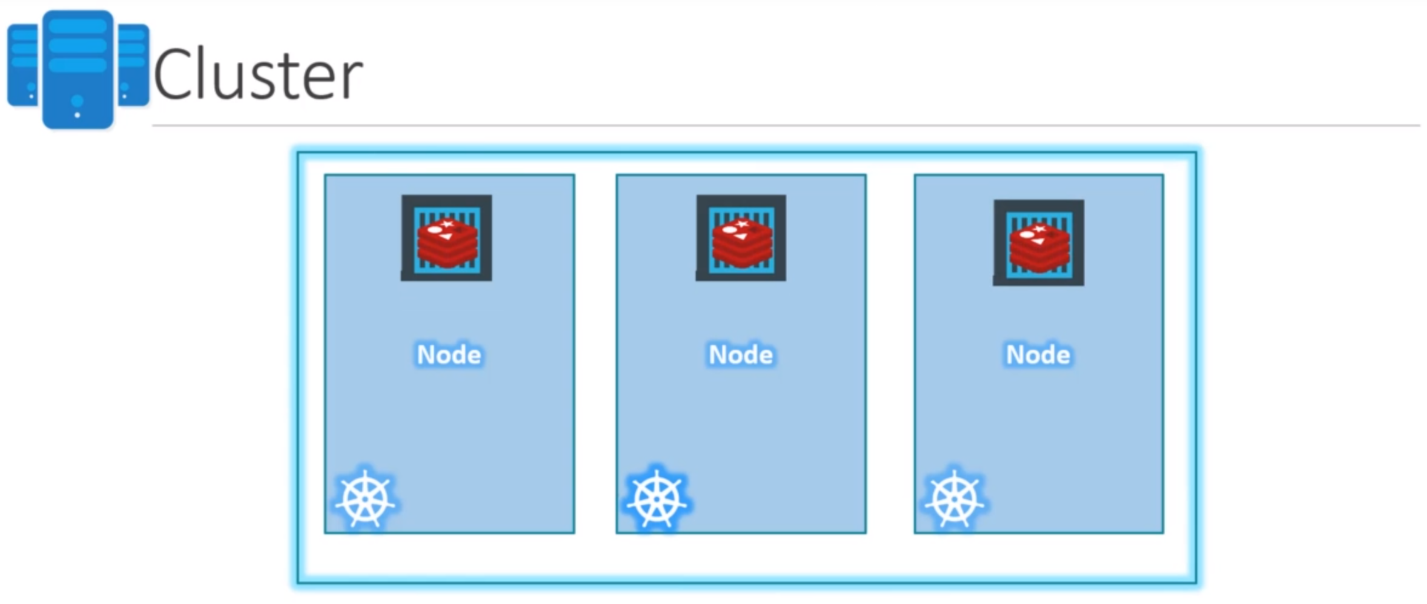
**Nodes:** A Node is a machine, physical or virtual on which Kubernetes is installed. A Node is a worker machine and that is where containers will be launched by Kubernetes. It was also known as Minions in the past. What if the Node on which your application is running fails? In that case our application goes down. So, we need to have more than one Node.

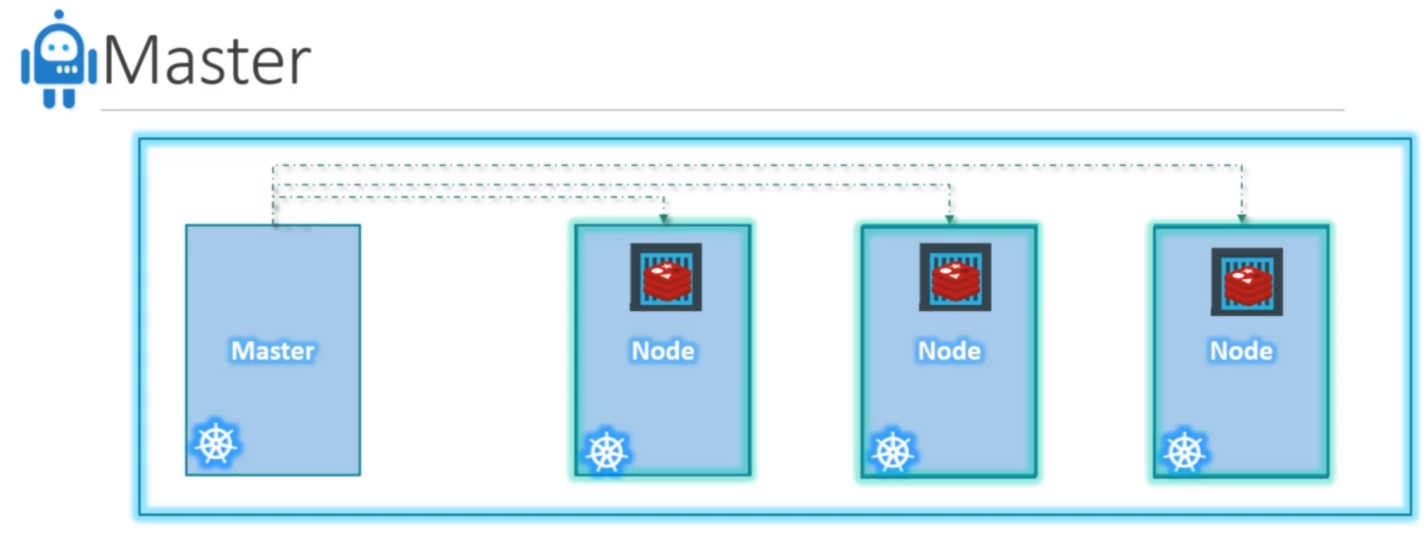


**Cluster:** A Cluster is a set of Nodes grouped together. This way even if one Node fails you have your application still accessible from the other Nodes. Moreover, adding multiple Nodes help in sharing load as well.



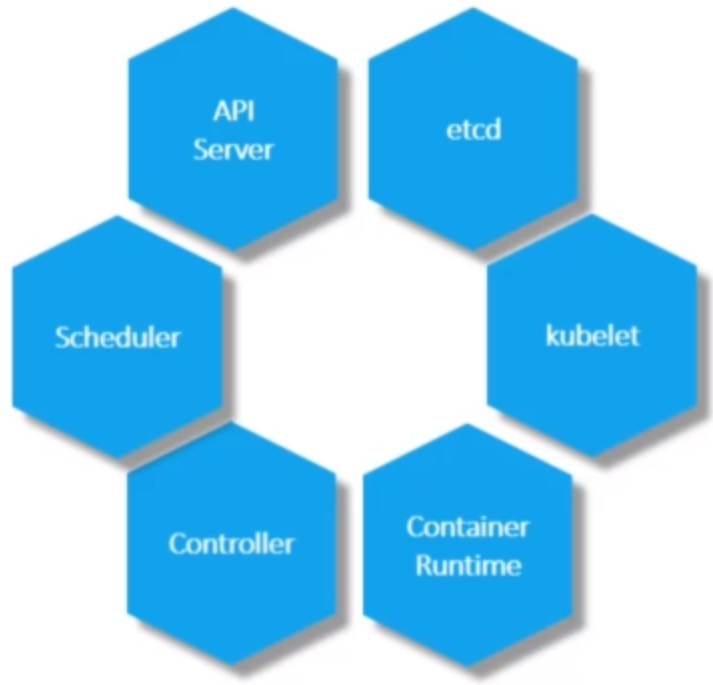
Now, we have a cluster. But who is responsible for managing the cluster? Where is the information about members of the cluster stored? How are the Nodes monitored? When a Node fails how do you move the workload of the field Node to another worker Node? That’s where the Master comes in.

**Master:** The Master is another Node with Kubernetes installed in it and is configured as a Master. The Master watches over the Nodes in the Cluster and is responsible for the actual orchestration of containers on the Worker Nodes.



**Components:** When you install Kubernetes on a system, you’re actually installing the following components:

1. **API Server:** The API Server acts as a front-end for Kubernetes. Users, management devices, command line interfaces, all talk to the API server to interact with Kubernetes cluster.
2. **Etcd Service:** The Etcd key store is a distributed, reliable key store used by Kubernetes to store all data used to manage the Cluster. When you have multiple Nodes and multiple Masters in your Cluster, Etcd stores all that information on all the Nodes in the Cluster in a distributed manner. Etcd is responsible for implementing locks within the Cluster to ensure that there are no conflicts between the Masters.
3. **Kubelet Service:** Kubelet is the agent that runs on each Node in the cluster. The agent is responsible for making sure that the Containers are running on the Nodes as expected.
4. **Container Runtime**
5. **Controllers:** The Controllers are the brain behind Orchestration. They are responsible for noticing and responding when Nodes, Containers or end points goes down. The Controllers make decisions to bring up new Containers in such cases. The Container Runtime is the underlying software that is used to run Containers. In our case it is Docker.
6. **Schedulers:** The Scheduler is responsible for distributing work or containers across multiple Nodes. It looks for newly created containers and assigns them to Nodes.

****

**Master vs Worker Nodes**

How Master and Worker Nodes are distributed across different types of servers? In other words, how does one server become a Master and the other a Slave?

The Worker Node or Minion, where containers are hosted, for example Docker Container and to run Docker Container on a system, we need Container Runtime installed and that’s where the Container Runtime falls. In this case, it happens to be Docker. It could also be other containers as well such as Rocket or CRI-O.

The Master Server has the Kube API Server and that is what makes it a Master. Similarly, the Worker Node has the Kubelet agent that is responsible for interacting with a Master to provide health information of the Worker Node and carry out actions requested by the Master on the Worker Nodes. All the information gathered are stored in a Key Value store on the Master. The Key Value store is based on the popular Etcd framework. The Master also has the Controller and the Scheduler.



**Kubectl:** The Kubectl tool is used to deploy and manage applications on a Kubernetes Cluster. To get Cluster information, to get the status of other Nodes in the Cluster and to manage other things. The Kubectl run command is used to deploy an application on the Cluster. The Kubectl Cluster info command is used to view information about the Cluster and the Kubectl get nodes command is used to list all the Nodes part of the Cluster.