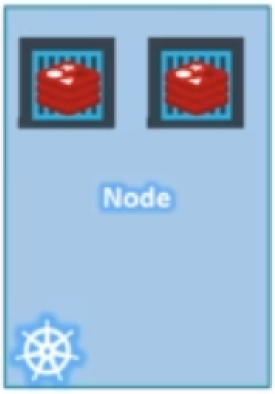
Architecture

Kubernetes Architecture

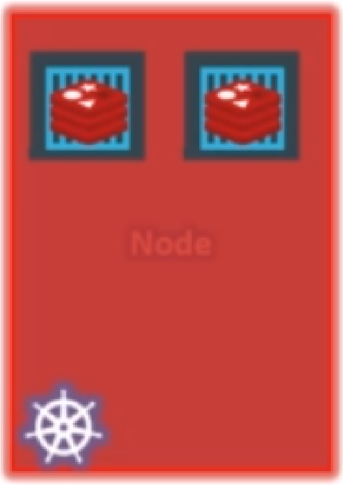
- Before we get into setting up the Kubernetes it is important to understand some of the basic concepts.
- Lets first understand what is nodes.

Nodes

- A Node is a machine [Physical or Virtual] whatever it could be where Kubernetes is installed.
- Nodes is also known as worker machines where containers is launched by Kubernetes.
- Nodes were also known as Minions in past.



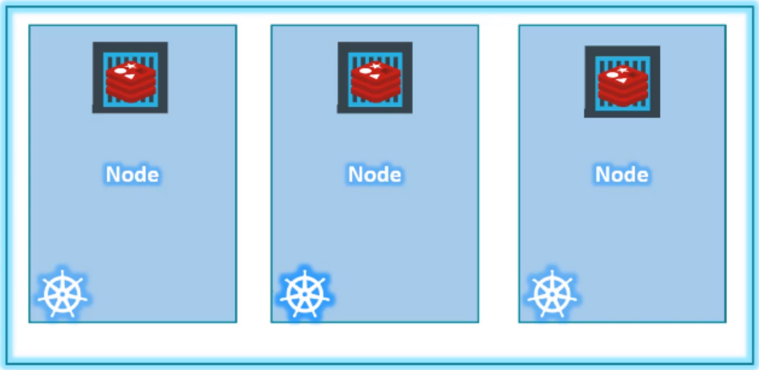
- But the real question here would be what if the node goes down? Eventually your application may go down isn't ?



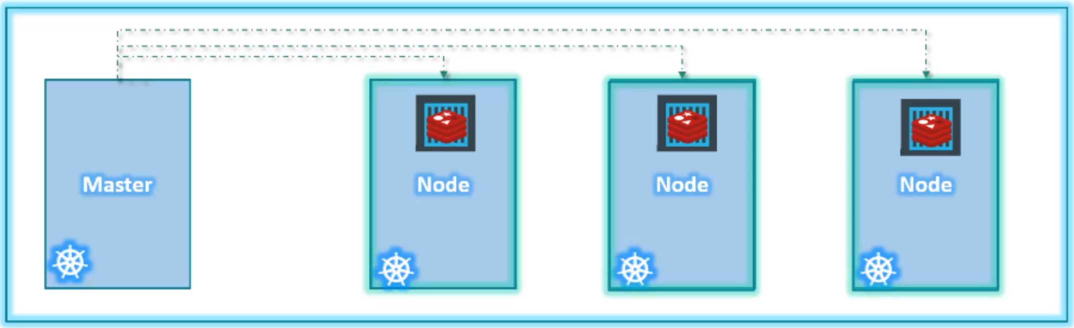
- To avoid such scenario we are going to have more then one nodes which is called cluster.

Master

- A cluster is set of nodes grouped together. [This way even if one node fails your application might be still accessible from other nodes]

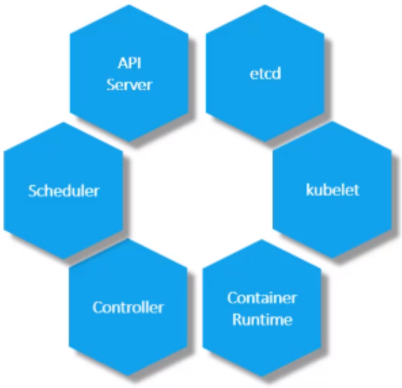


- Moreover having multiple nodes help to share the load.
- Now we have cluster but who is responsible to manage this cluster ?
- Where is the nodes information's stored?
- How are the nodes monitored ?
- When a node fails how the workloads are migrated to other node ?
- That's where the "Master" comes in.
- Master is an other node with Kubernetes installed in it and it is configured as Master.
- Master is responsible to monitor the worker nodes and manage the container orchestration.



Components

- When you are installing Kubernetes it actually installs following components.



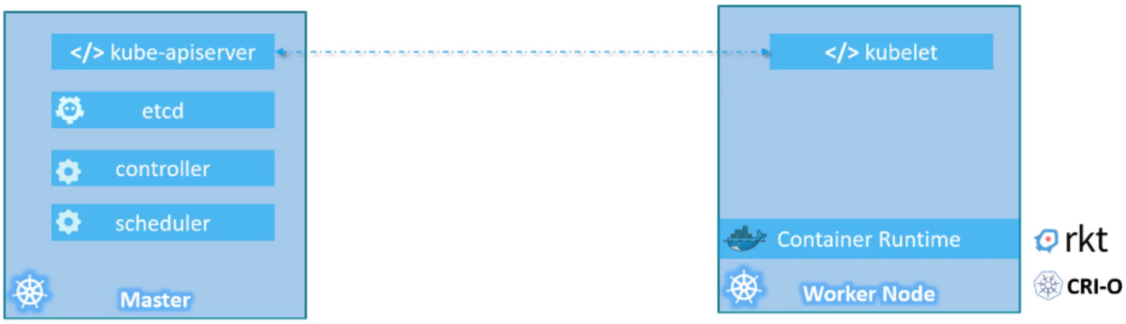
- **API SERVER :** Acts as frontend for Kubernetes. [Users, Command line interfaces all talk to API SERVER to interact with Kubernetes cluster]
- **ETCD :** ETCD is distributed reliable Key-value store used by Kubernetes to store all data used to manage cluster. [To put it simple this component is going to store all the information's about Master & Node in distributed manner] When I say distributed "information's are not going to be centrally store instead its going to be stored in each machines locally". ETCD is also responsible for implementing logs within the cluster to ensure that there are no conflicts between Masters.
- **SCHEDULER :** SCHEDULER is responsible for distributing work (containers) across multiple nodes. The main role of this component is to look for newly created container and assign it to node in an automated way.
- **CONTROLLER :** CONTROLLER is the brain behind orchestration, they are responsible to notice and respond when nodes or containers goes down. Controllers are also responsible to make decisions to bring new containers in such situations.
- **CONTAINER RUNTIME :** This component is the piece of software used to run containers. In our case it happens to be Docker.
- **KUBELET :** KUBELET is an agent which runs in each node inside cluster. Agents are responsible to ensure containers are running in nodes as expected.

Master vs Worker node

So far we just saw what is master & worker node and set of components which are used to run the show without any hiccups.

- But how are these components distributed across multiple server?

- In other how are we going to classify which machine is Master and which one is Worker ?
- The worker node is the place where container is going to reside. For example Docker container.
- To run docker container - we may need component "container runtime" configured in machine.
- We are going to use docker in this segment though we have lot more components available like [Rocket & CRI-O]
- Master server is going to have [Kube-apiserver] configured in it. This component is actually making it as Master.
- Similarly worker node is going to have [KUBELET agent] that is responsible to make interact with Master [Which also transfers health information about worker node and carry out actions requested by Master]
- Other components like ETCD, Scheduler & Controller is going to residing on Master.



KUBECTL

- KUBECTL (KUBE control) is an command line utility used to deploy and manage containers in Kubernetes cluster.
- Also it is useful to get the cluster/node/master/container information's.
- In Nutshell, we are going to use several KUBECTL commands to create and manage containers in Kubernetes cluster.