# KUBERNETES

## Introduction:

Kubernetes is an open-source container-orchestration technology for automating deployment, scaling, and management of containerized applications.

K8S was originally designed by Google in 2014 and is now maintained by the Cloud Native Computing Foundation (CNCF).

## Installation:

Git Repo: https://github.com/artisantek/kubernetes-installation

- k8s Master --> t2.medium
- k8s Worker Nodes --> t2.micro

- kubectl get nodes --> To check all the nodes in the setup
- kubectl label node <name> node-role.kubernetes.io/worker1= --> To label a node[give role]

## Kubernetes Architecture:

- ### Master (Control Plane):

Kubernetes Master majorly runs with 4 components

1. **API Server:**

The API Server is the main management point of the entire cluster. When we interact with the Kubernetes cluster using kubectl command, we are communicating with the master API Server component.

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

The API Server is also responsible for the authentication and authorization mechanism.

The API Server also implements a watch mechanism to watch for changes. This allows components such as the Scheduler and Controller Manager to interact with the API Server and try to match the desired state with the actual state

2. **etcd:**

etcd is a distributed, consistent key-value store used to store the configuration data of the complete Kubernetes cluster.

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, and a whole lot more) at any given point of time.

As all cluster data is stored in etcd, to back up the cluster we need to take the backup of etcd. You can easily back up your etcd data using the etcdctl snapshot save command

To learn about etcd: etcd.io.

3. **Scheduler:**

The Scheduler watches for unscheduled pods and binds them to nodes via the /binding pod subresource API,

Scheduler decides on which node the pod has to be created according to the availability of the requested resources, 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.

4. **Controller (Control Manager):**

The Kubernetes Controller Manager is a daemon which runs a never-ending core control loop (also known as “controllers”). 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 Deployment Controller, Replication Controller, DaemonSet Controller etc.

When you create a pod using kubectl, this what happens:

- kubectl writes to the API Server.
- API Server validates the request and persists it to etcd.
- etcd notifies back the API Server.
- API Server invokes the Scheduler.
- Scheduler decides where to run the pod on and return that to the API Server.
- API Server persists it to etcd.
- etcd notifies back the API Server.
- API Server invokes the Kubelet in the corresponding node.
- Kubelet talks to the Docker daemon using the API over the Docker socket to create the container.
- Kubelet updates the pod status to the API Server.
- API Server persists the new state in etcd.

- **Node (Worker):**

A node is a worker machine, and this is where containers inside the pods will be launched by Kubernetes.

The three major node components are:

1. **Container Runtime:**

The container runtime is the underlying software that is used to run containers. In our case, it happens to be Docker. Kubernetes supports several runtimes: Docker, containerd, cri-o, rktlet etc.

2. **Kubelet:**

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.

This communicates with the master component to receive commands and work. The kubelet takes a set of Pod Specs that are provided through various mechanisms and ensures that the containers described in those Pod Specs are running and healthy. The kubelet doesn’t manage containers which were not created by Kubernetes.

3. **Kube Proxy:**

This is a proxy service which runs on each node and helps in making services available to the external host. It helps in forwarding the request to correct containers and can perform primitive load balancing.

K8s cluster can have multiple worker nodes and each node has multiple pods running, so if one must access this pod, they can do so via Kube-proxy.

4. **Pod:**

A pod is the smallest deployable unit that can be managed by Kubernetes. A pod is a logical group of one or more containers that share the same IP address and port space.