What is Kubernetes?

Kubernetes also known as “K8s”, is an open-source container orchestration platform developed by google.

NOTE:

What is Container Orchestration platform?

We already know about docker . After installing docker engine we can call that as “docker host”.

We can run multiple containers in that “docker host”.

Kubernetes helps us to combines multiple “docker hosts” into a single entity.

Why we need multiple docker hosts?

If we run all the containers in a single docker host. What if that fails(it’s a single point of failure) to reduce these type of failures and to run applications with high availability we have to setup more than one “docker host”.

Kubernetes helps us to calculate CPU , RAM and to schedule the containers(on which docker host these containers needs to run).

ORCHESTRATION is nothing but handling a process in a systematic way.

It is designed to automate the deployment, scaling, and management of containerized applications across a cluster of nodes.(cluster of nodes is nothing but multiple docker hosts).

Kubernetes provide a consistent and reliable way to manage applications, regardless of whether they are running on-premises, in the cloud, or in hybrid environments.

Q) Is docker mandatory for K8s?

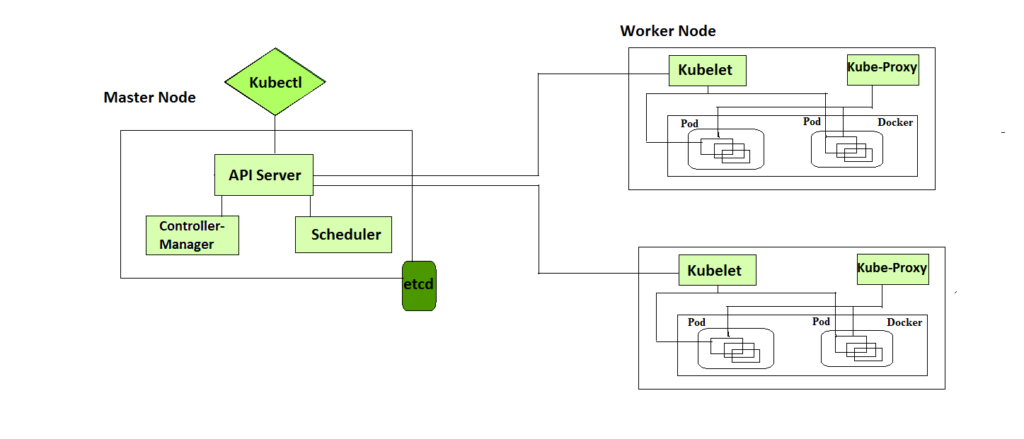
Yes, docker ( not only docker any containerization platform) is required to work with Kubernetes because Kubernetes doesn’t have capability to create containers. It can just automate the deployment and scalability of containers.

Architecture and components of Kubernetes:

Kubernetes has a client server architecture and has master and worker nodes(slave nodes). Master node is the brain of kubernetes infrastructure and slave nodes are the systems which actually works.

To work with Kubernetes infrastructure we have to setup cluster.( collection of multiple nodes).

A kubernetes cluster is a cluster of nodes configured in a master/slave architecture that runs containerized applications. They are more faster and light weight than virtual machines, making the deployment and scaling of applications easier and more manageable.



A Kubernetes cluster node is made up of one master node and several worker nodes.

The worker nodes are responsible for running the containers and doing any work assigned to them by the master node.

The master node looks after: scheduling and scaling applications. Maintaining the state of the cluster and implementing updates.

MASTER NODE COMPONENTS:

**API server:** The API Server is a component of the Kubernetes “control plane”( It is also known as master node) that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

KUBELET:

In every worker node there will be a “kubelet “which is responsible to communicate with API server and post/share the information( like resource utilization (how much CPU it utilized) , how many containers are running in this nodes) to scheduler and controller manager.

Then with the help of kubectl master node will get to know that which node is free or which node contains how much capacity. After that master node will store all these type of information in “etcd Database”.

**Scheduler:** control plane component that watches for newly created pods with no assigned node and select a node for them to run.

**Controller manager:**Control plane component that runs controller processes.

**etcd Database:**Consistent and highly available key value store used as Kubernetes backing store for all cluster data.

WORKER NODE COMPONENTS:

**KUBLET:**

An agent that runs on each node in the cluster. It makes sure that containers are running in a pod.

The kubelet takes a set of podspecs that are provided through various mechanisms and ensures that the containers described in those podspecs are running and healthy. The kubelet doesn’t manage containers which were not created by Kubernetes.

**KUBE-PROXY:**

Kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes service concept.

Kube proxy maintains network rules on nodes. These network rules allow network communication to your pods from network sessions inside or outside of your cluster.

**WORK LOAD COMPONENTS:**

**Pods:**

In Kubernetes pods are just like containers in docker.

But there is a small difference between pod in Kubernetes and container in docker is “when we are interacting with container in docker we are directly giving commands to stop and to start” .

In kubernetes they developed a separate command line to interact with containers. So these command line helps us to do not interact directly. Instead of that it creates a layer up on the containers(which is a bubble) that bubble is called “pod”.

Pod is a collection of one or more containers . Kubernetes interact with pod. So all the commands issued will internally applied to containers inside pod.

“Pod “ is the smallest and most basic unit of deployment. It represents a single instance of a running process within the cluster

A pod encapsulates one or more containers, storage resources, network configurations, and other options required to run a specific set of containers together.

**Deployments:**

Deployments is a collection of pods(pod is a collection of containers) .

In Kubernetes, a “Deployment” is an object that provides declarative updates and management for a set of re[plica pods.

When you create a deployment, you can specify the desired state by defining the container images, number of replicas, and other configuration parameters.

If there are any discrepancies, Kubernetes automatically take action to reconcile the state creating or deleting pods as necessary.

**ReplicaSet:(Replica means number of copies)**

In Kubernetes, a “ReplicaSet” is an object that ensures a specified number of replica pods are running at any given time.

It is responsible for maintaining the desired replica count and managing the lifecycle of the pods.

Replica sets are typically used to manage stateless applications where individual instances of the applications can be treated as interchangeable.

They help in achieving high availability and scalability by automatically scaling the number of replicas up or down based on the defined specifications.

When we create a replicaset, we specify the desired number of replicas and provide a template for creating the pods.