What is Kubernetes? Confused? Kubernetes and its Architecture Explained in Plain English

# What is Kubernetes ? and how is kubernetes architecture ?

To start with, Kubernetes, also known as “K8s” is an open source container orchestration platform that automates the deployment, management, load balancing, and scaling of container based applications ( also called containerized applications).

Secondly, Kubernetes is a leader in container orchestration as [this survey shows](https://containerjournal.com/features/measuring-kubernetess-market-share-mind-share/). In fact, It can orchestrate thousands of containers using open source code. And since  it is open source, a K8s cluster  can run almost anywhere including public cloud providers ( e.g. all major cloud providers like Amazon Web services, Azure)

Indeed Kubernetes is an enabler for Platform as a service (PaaS) that creates hardware layer abstraction for development teams.

Why do you need to know about the Kubernetes?

Today, majority of  applications are developed as [cloud native](https://telcocloudbridge.com/blog/what-is-cloud-native-understand-the-easy-way/), i.e. they run as microservices/containers. Due to their lightweight size, they have become popular. 5G core, for example, is completely cloud native based. As Kubernetes is popular container orchestration platform, so it is important to know about this platform.

The best way to understand and appreciate Kubernetes is to understand its architecture.

The Best way to understand “what is Kubernetes” is to understand its architecture.

So as you are looking to know about “what is Kubernetes”, i suggest that understanding its architecture can clarify it.

However, the info out there makes Kubernetes architecture hard to understand.

In this step by step guide, I will make it very easy for you to understand the Kubernetes architecture clearly.

**I will make Kubernetes easy to understand through the ship analogy**

I will draw parallels between Kubernetes and cargo ships. This is an interesting analogy and you would not forget the concepts after that.

Before understanding the Kubernetes and using the ship analogy, it is good to understand first some basic terms like Pod, cluster, and Node.

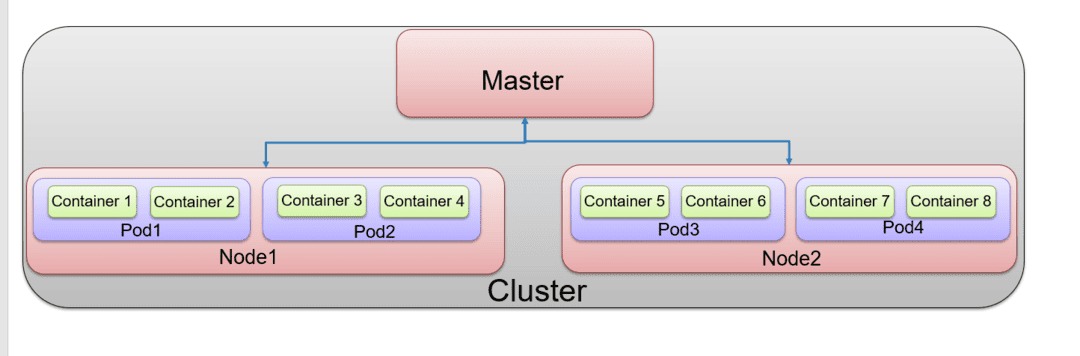
In fact, Kubernetes is architected as a cluster of nodes, and then containers are deployed into pods using  Control Plane. The control plane manages the nodes and the pods.

# Concept of Pod, Cluster, Container, and Node

Let’s first understand the hierarchy going from small to bigger.

**Container > Pod > Node > Cluster**

This order will help you remember, smaller to the bigger size. The smaller resides in the bigger.

Concept of Pod, Node, container

**Container**

This is the level 0.

[Kubernetes.io](https://kubernetes.io/docs/concepts/containers/) defines container as following“A container image is a ready-to-run software package, containing everything needed to run an application: the code and any runtime it requires, application and system libraries, and default values for any essential settings”the container is the smallest unit and sits inside the Pod. However, Kubernetes does not create containers. That is the job of the container runtime, such as Docker ( more at the end)

**Pod**

This is the level 1.

The container sits in the Pod. A Pod is defined as a logical unit of application. It can contain one or more containers and is deployed on a node. For Kubernetes PoD is the basic unit ( instead of containers) i.e Kubernetes deploy Pods ( which have containers inside) and does not create containers directly. Normally Pod will have a single type of application or it can have multiple applications if they are closely related. Each pod is assigned a unique IP address within the cluster, that enables the application to use ports without conflict. Pod to pod communication is enabled using a dedicated Pod IP address

**Node ( Worker Node)**

This is level 3.

Pods run inside nodes.

Nodes can be virtual machines or physical servers. A Node is a worker machine in Kubernetes. Each Node is managed by the Master. ( more about worker node and master node later on) A Node can have multiple pods, and the Kubernetes master automatically handles scheduling the pods across the Nodes in the cluster.

Node is also called worker node in Kubernetes.

**Cluster**

This is level 4.

Ezoic

At the minimum, the Cluster contains one or a group of nodes and a control plane.  containers do not need to be tied to the specific virtual machine. Containers can run anywhere, on physical, virtual, or cloud but what ties them together is them being part of a cluster.

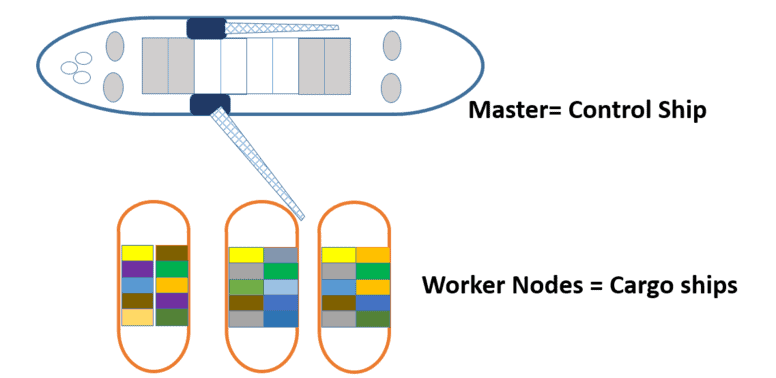
Kubernetes clusters are comprised of one master ( more on master later on) and a number of nodes ( also called worker nodes)

After getting the clarity on the master and worker node, let’s see the reference architecture of Kubernetes and the role of master and worker node.

But the best way to understand what is inside the master and worker node is to use the ship analogy.

Lets take the next section of understanding what is kubernetes and kubernetes architecture.

**Kubernetes architecture using ship analogy?**

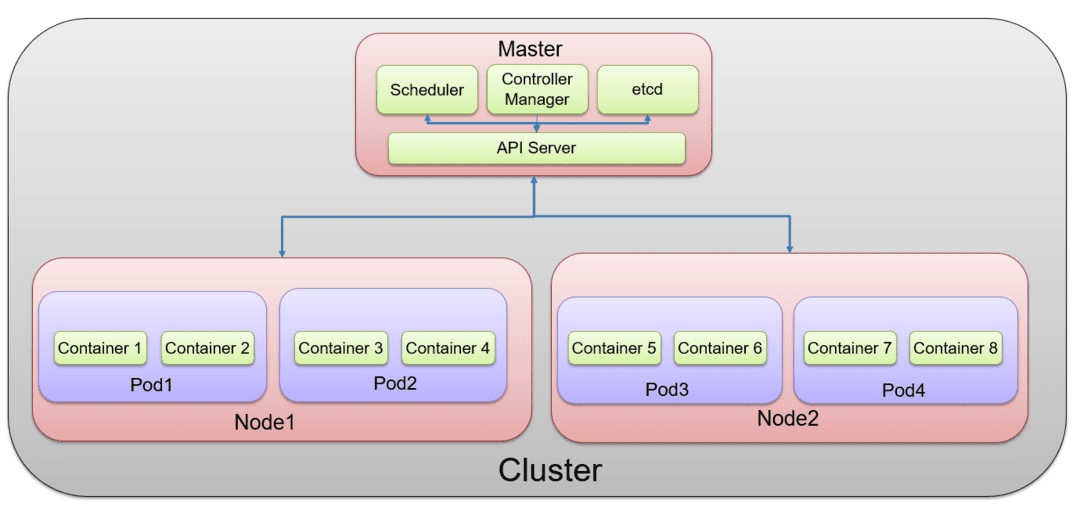
Kubernetes analogy to Ships

Worker nodes are like cargo ships. Cargo ships take containers from source to destination. In Kubernetes, worker nodes host containers of applications. Master is like control ships. Control ships manage and monitor cargo ships

Ezoic

Let’s take a look at different components of the control ship/master

**What is inside the master (Control ship)?**

Master components in Kuberentes

The master runs control plane components. The control plane components make global decisions about the cluster as well as detecting and responding to the cluster events. Usually, control plane components are run on a dedicated machine in a cluster.

**Kube scheduler**

**Kube scheduler= Cranes in control ship**

When cargo ships arrive, containers are loaded on them using cranes, , therefore there should be a way to identify where to place the containers, which ships have the capacity, constraints, etc. This is the job of the schedulers. But as Kubernetes run pods ( containers are part of the pods), so we can say that kube scheduler schedules pods on the ships ( on the worker nodes)

The scheduler takes into account the resource needs of the pods, such as CPU or memory, along with the health of the cluster. Based on this, it can schedule pod to the appropriate compute node

**etcd**

**etcd= cargo ship profiles**



As containers are loaded and unloaded from ships on daily basis. There should be a database that needs to keep track of all this information on where are those containers and at what time they are uploaded. etcd is a database that stores this information ( It is like a source of truth) This is analogous to a “cargo ship profiles” that has details of all such information.

**kube api server**

**api server = The department in control ship that manages communication**

There are multiple offices onboard a ship. As well there are external stakeholders. Who manages the communications within internal and between the internal and external entities? kube-api server serve exactly that purpose

**kube controller manager**

**Kube-controller-manager = captain of the master node**

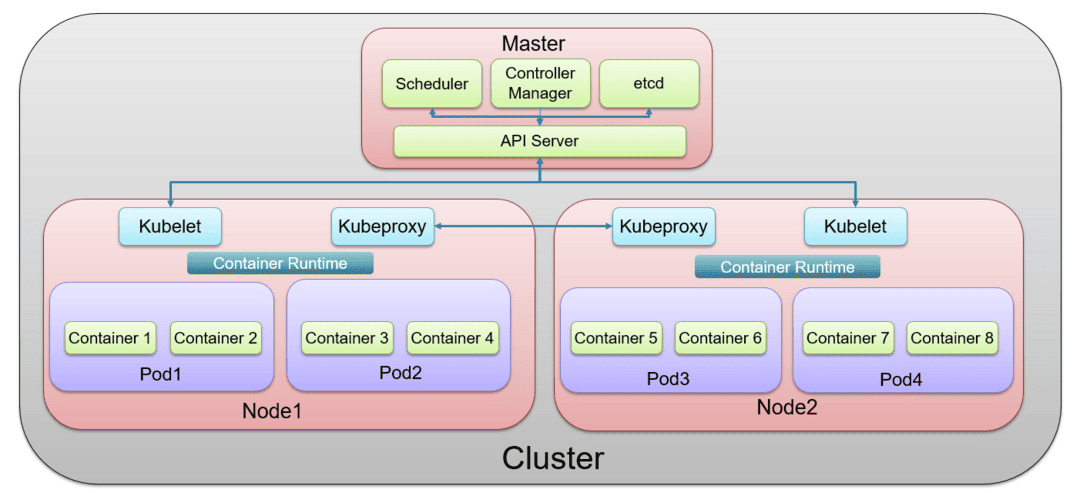
Kube controller manager

In Kubernetes, the kube controller manager runs a different kind of controllers that control different functions like ( it is like the captain of the master node)

1. Node controller- responds when nodes go down
2. Replication controller-responsible for maintaining the correct number of pods
3. End point controller-Populates the endpoint objects like joins, services, and pods.

In ship analogy, kube controller manager is managing different offices in dock like operation office ( the team that responds if any issue with the worker node), replication office (the team responsible for maintaining correct number of pods)

**What’s inside the worker node (Cargo ship)?**

Kubernetes architecture

**kubelet**

**Kubelet =captain of the worker ship**



This is an agent that runs on each node of the cluster. It listens to instructions from kube-api-server and manages the Pods and the containers running on the machine.

**kube proxy**

**kube proxy= manages communication between worker ships**

Ezoic

kube proxy manages communication between different containers deployed across different nodes. For example, a packet core may have multiple different services, one service deployed on one node, while another on another node that needs to talk to one another, the kube-proxy facilitates this kind of communication

In terms of ship analogy, kube proxy manages communication between different worker nodes.

**container runtime**

You need to install a [container runtime](https://kubernetes.io/docs/setup/production-environment/container-runtimes) into each node in the cluster so that Pods can run there.

Container runtime is installed in each node in the cluster so that pods can run there. [Docker](https://www.redhat.com/en/topics/containers/what-is-docker) is one of the popular container runtime but there are others such as the Open container initiative.

So thats it, I hope it is clear now on what is Kubernetes and Kubernetes architecture. Please leave a comment below and let me know your thoughts on whether it is now easy for you to understand Kubernetes ?

**References**:

slideshare-demystifying-the-nuts-bolts-of-kubernetes-architecture

<https://opensource.com/article/20/6/container-orchestration>

<https://kubernetes.io/docs/concepts/overview/components/>