**Kubernetes** is an open-source container-orchestration system for automating application deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. It was originally designed by Google, and is now maintained by the Cloud Native Computing Foundation

1. **Basics:**

Imagine a team in your company and to manage the team we have a Manager who supervises and allocates tasks to the team members. Kubernetes aka K8s also works in a similar way where we have at least 1 Master node (similar to manager) and multiple worker nodes (team members) to execute the tasks.

For basic understanding see Figure-1. For simplicity I have shown 1 Master/Control plane and 3 worker nodes but in an actual production environment there will be many Master nodes for High Availability and Fault tolerance and many worker nodes (K8s supports up to 5000 worker nodes per cluster)

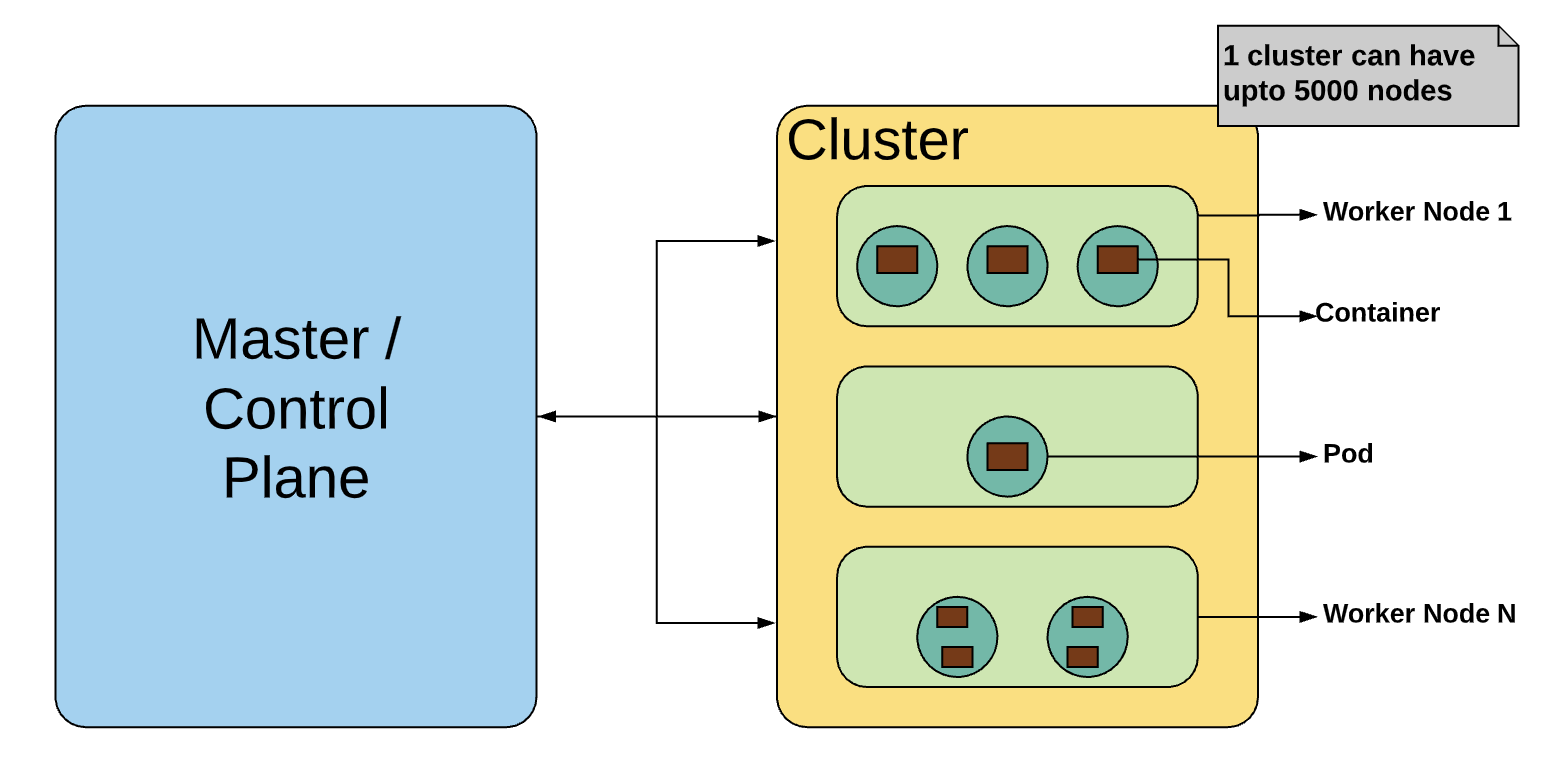


Figure-1

1. **1st Level Deep Dive:**

**Master / Control Plane**: It is responsible for managing the whole cluster. Master monitors the health of each node, stores meta data and configuration of the nodes. So if a worker node fails, master moves the task to another healthy node. Master is also responsible for:

1. Scheduling
2. Provisioning
3. Controlling and
4. Exposing APIs to the clients

**Worker nodes:** Worker nodes are as simple as VMs or physical servers in your DC or could be a VM in cloud. Nodes expose the underlying networking, compute and storage resources to the applications. All nodes together form a cluster to provide fault tolerance and replication.

**Pods**: Pod is a scheduling unit in K8s. Each Pod consists of 1 or more containers. In ideal scenarios there should be 1 container per pod but there are scenarios where you might need to run more than 1 containers dependent on each other together within a Pod. Pod acts like a wrapper over containers and we interact and manage the containers via Pod.

**Containers**: These are run time environments for your containerized applications. Applications are run within containers. Containers are designed to run micro services and not ideal for monolith applications.

See Figure-2 to understand the Worker Nodes - Pods - Container relationship.

A picture containing clock

Description automatically generated

  Figure-2

1. **2nd Level Deep Dive:**

**Master / Control Plane**: There are 4 components which make up a K8s Master / Control Plane. So when you install K8s on your system these 4 components will get installed:

1. **API serve**r: This is a gatekeeper for entire cluster. If you want to create, delete, update or display a K8s object then it has to go via API to API server. API server configures and validates API objects like pods, services, replication controllers and deployments. It exposes all the Kubernetes API's using which you can manage a cluster. We interact with these APIs through a tool called "**Kubectl**". It is a GOLANG binary.

1. **Scheduler**: This is responsible for scheduling Pods across nodes. Depending on the configuration file issued to K8s by the developer the pods are scheduled. For e.g. the configuration file defines the compute as 1 core CPU and 1 GB memory then scheduler looks at this artefact and finds a node which meets this criteria and schedules a pod accordingly.

1. **Controller Manager**: This is responsible for overall health of entire cluster. There are 4 controllers behind control manager. These controllers ensure that the nodes are up and running, correct number of pods are running as issued via the configuration file. These are:

1. **Node controller**: Responsible for noticing and responding when nodes go down.
2. **Replication controller**: Responsible for maintaining the correct number of pods for every replication controller object in the system.
3. **Endpoints controller**: Populates the Endpoints object (that is, joins Services & Pods).
4. **Service Account & Token controllers**: Create default accounts and API access tokens for new namespaces.

1. **etcd**: This is a distributed key value light weight database. This is a central database which stores the current cluster state. This is the single source of truth for all nodes, components and master which is forming a K8s cluster.

See Figure-3 to understand how the Control Plane is designed.

A close up of text on a white background

Description automatically generated

Figure-3

**Worker Node**: There are 2 components which are used to communicate with worker nodes.

1. **Kubelets**: This is a node agent which runs on each worker node inside the cluster. The main task of Kubelets is to ensure that the PODs, specs of which are submitted via configuration file to API server are running and healthy. If any issue is identified then it tries to restart the pod on same node. If the issue is with node itself then Master detects it and recreates pod on another node. The replica set or replication should be defined in the config to mange this. If replication is not defined then the node will die.

1. **Kube - Proxy:** This is responsible for maintaining entire network config. It maintains the network rules across the nodes, pods and containers. It also exposes services outside of the cluster. It is the core networking component within K8s.

1. Pod: Scheduling unit in K8s.Pod acts like a wrapper over containers and we interact and manage the containers via Pod.

1. Containers: Containers provide the runtime environment for your applications. K8s supports both Docker and Rocket based containers.

See Figure-4 to understand how the Worker nodes are designed.

A close up of a sign

Description automatically generated

Figure-4

That concludes a good read on Kubernetes architecture understanding. Lets combine Master and worker node together and you will get a K8s high level architecture.

A screenshot of a video game

Description automatically generatedFigure-5