Kubernetes Components

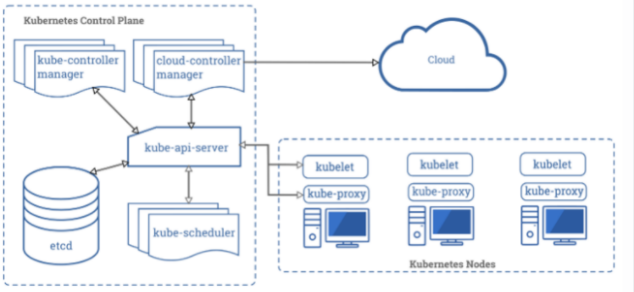
Kubernetes components

Kubernetes has the following main components:

- One or more master nodes
- One or more worker nodes

You get a cluster when you deploy Kubernetes. A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

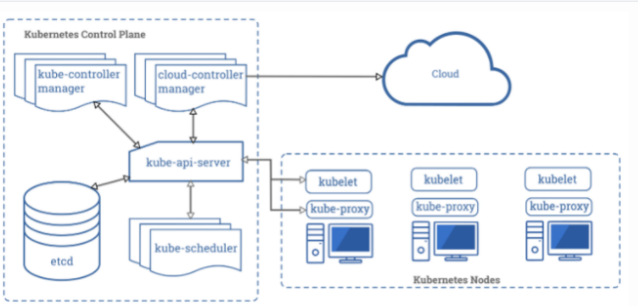


The diagram of a Kubernetes cluster

Control Plane Components

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).

Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine.



The diagram of a Kubernetes cluster

kube-apiserver

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver. kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

etcd

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.

kube-scheduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

kube-controller-manager

Control Plane component that runs controller processes.

Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

These controllers include:

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

cloud-controller-manager

A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that just interact with your cluster.

The cloud-controller-manager only runs controllers that are specific to your cloud provider. If you are running Kubernetes on your own premises, or in a learning environment inside your own PC, the cluster does not have a cloud controller manager.

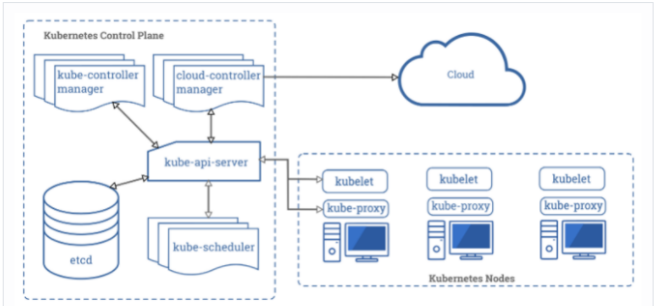
As with the kube-controller-manager, the cloud-controller-manager combines several logically independent control loops into a single binary that you run as a single process. You can scale horizontally (run more than one copy) to improve performance or to help tolerate failures.

The following controllers can have cloud provider dependencies:

- Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
- Route controller: For setting up routes in the underlying cloud infrastructure
- Service controller: For creating, updating and deleting cloud provider load balancers

Node Components

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.



The diagram of a Kubernetes cluster

kubelet

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.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

Container runtime

The container runtime is the software that is responsible for running containers.

Kubernetes supports several container runtimes: Docker, containers, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface).

Summary of K8s

- **Kubectl :** A CLI tool for Kubernetes
- **Master Node:** The main machine that controls the nodes, Main entrypoint for all administrative tasks
- **Worker Node :** It is a worker machine in Kubernetes. This machine performs the requested tasks. Each Node is controlled by the Master Node
- **Kubelet :** Primary node agent. Ensures that containers are running and healthy
- **Kubernetes Pod::** A Pod can host multiple containers and storage volumes. Pods are instances of Deployments
- **Deployment :** A deployment is a blueprint for the Pods to be create.
- **Secret :** A Secret is an object, where we can store sensitive informations like usernames and passwords.
- **Service :** A service is responsible for making our Pods discoverable inside the network or exposing them to the internet. There are 3 types of services:
  - ClusterIP
  - Node Port
  - Load Balancer

> Tips:
> - Master controls the cluster and nodes in it
> - Nodes host the containers inside them (Containers are inside separate PODS)
> - PODs are logical collection of containers which need to interact with each other for an application

Complementary Interactive Lesson about Introduction to Kubernetes