A sysadmin's guide to basic Kubernetes components

Kubernetes control plane nodes and worker nodes, their features, and how they interact.

Posted: December 1, 2020 | Shashank Nandishwar Hegde (Red Hat, Sudoer)

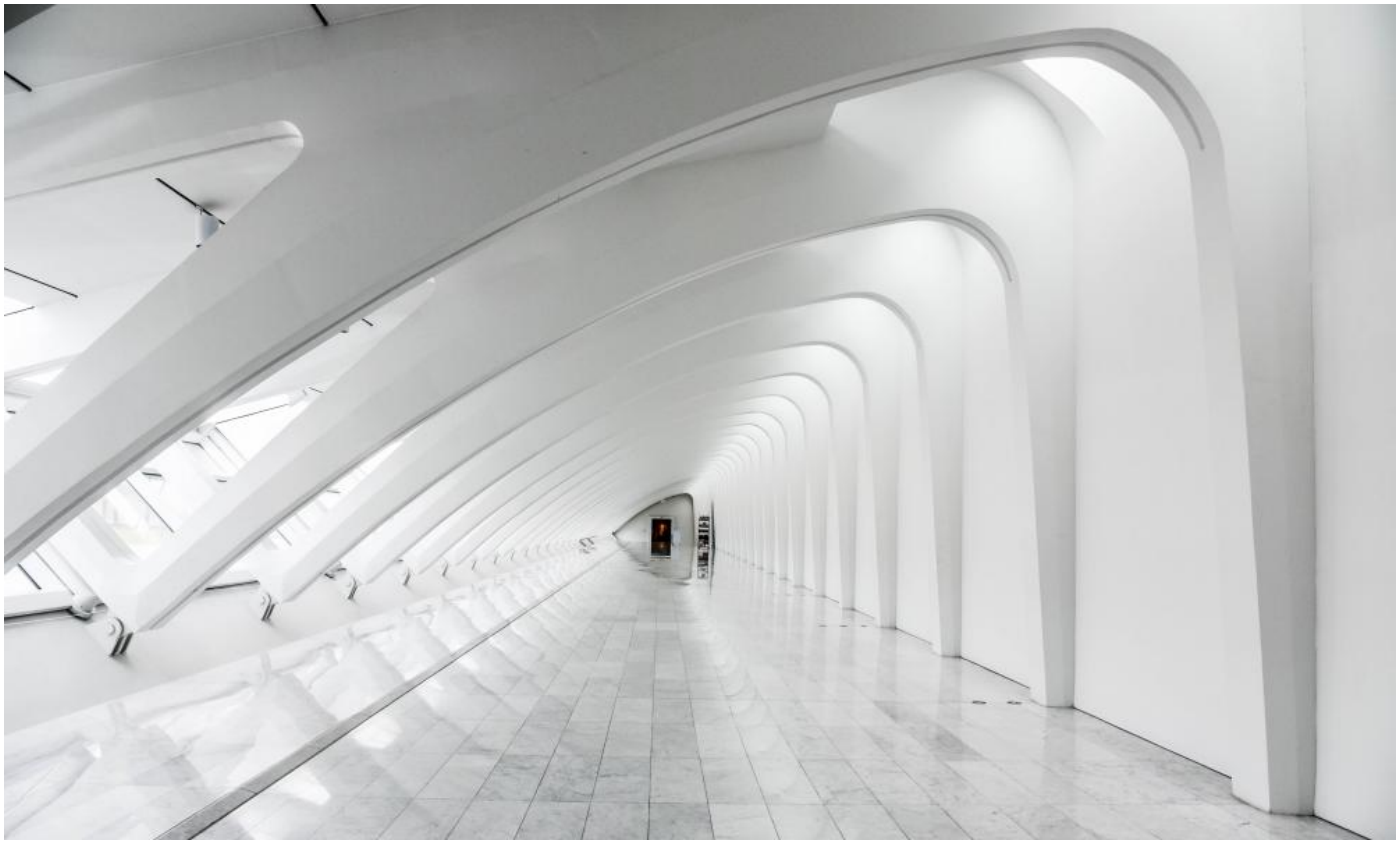

*Photo by **gdtography** from **Pexels***

A general Kubernetes (Also written as K8s) cluster consists of *control plane* nodes and at least one *worker node*.

This tutorial walks you through the Kubernetes architecture and the control plane and worker node components. It explains the architecture and features like *api-server*, *scheduler*, *etcd*, and *control manager*. Under worker nodes, we cover details about *kubectl*, *kubelet*, and *kube-proxy*.

Kubernetes architecture

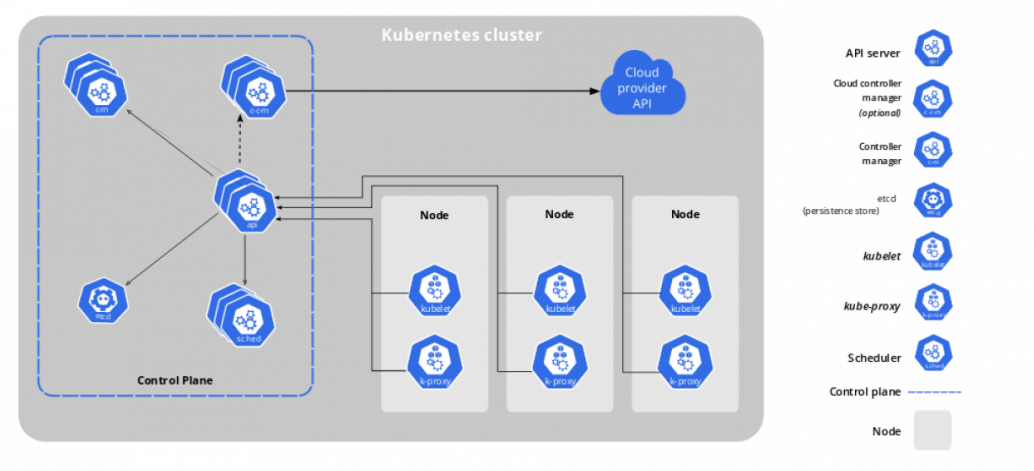

Image from kubernetes.io

From the above diagram, we can see that the control plane makes some global decisions such as scheduling, and it maintains the cluster details in a data store (etcd). It also responds to the cluster events, including maintaining the replicas as mentioned in the deployments.

Control plane components

So, what is a Kubernetes control plane?
In plain and simple words, the control plane is a global decision-maker for the cluster, meaning it controls scheduling. It is also responsible for maintaining overall cluster behavior. Containerized apps are scheduled on the worker nodes based on the memory allocated per deployment. It also responds to cluster events. For example, the control plane starts/terminates new pods when replicaset values are not satisfied. We can say the Kubernetes control plane is the brain of the cluster that makes logical decisions.

[Readers also liked: Turn a Kubernetes deployment into a Knative service]

The control plane has the following components:

API server (aka kube-apiserver)

The API server is the Kubernetes frontend that exposes the Kubernetes API. It also validates and configures data for the API objects, including pods, services, deployments, replication controllers, and others. The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

etcd (data store)

The etcd data store is the Kubernetes backend, which contains the cluster information in key-value pairs. In Kubernetes, there is a concept of *desired state* and *actual state*. Kubernetes uses etcd to monitor these two states. If they diverge, Kubernetes makes changes to reconcile the actual state and the desired state. This is monitored using the etcd distributed data store.

kube-scheduler

The kube-scheduler is a control plane component which mainly assigns the unscheduled pods to the relevant node based on its memory usage. Memory usage and hardware/software constraints are considered while setting a node for certain application factors such as network policies.

kube-control-manager

The kube-control-manager is a control plane component that runs the control process. In general, a control process is a loop that focuses on making the desired state equal to the current state for any application in any given instance of time.

Worker node components

What is a worker node?
A worker node runs the containerized applications and continuously reports to the control plane's **api-server** about its health.

It has the following components:

kubelet

The kubelet is an agent that runs on each node in a Kubernetes cluster, ensuring that the containers inside the pods are running and healthy. It continually talks with the Kubernetes API to relay the health information of the pods.

kube-proxy

The kube-proxy is a network proxy that runs on each node in a Kubernetes cluster. It maintains network rules on all nodes, allowing smooth communication between pod elements both inside and outside the cluster.

Container runtime

It is software that is responsible for running containers inside the cluster nodes. Examples include CRI-O, containerd, Docker, etc.

[Get this free ebook: Managing your Kubernetes clusters for dummies.]

What's next?

In this two-blog post, I have covered the Kubernetes architecture and its components. I have made a humble and simple effort to explain virtual machines, container concepts, and the basic cluster architecture of Kubernetes.

[The differences between Kubernetes and OpenShift can be found in this new ebook.]