Introduction to Kubernetes

What is kubernetes.

Kubernetes is a portable extensible opensource platform for managing centralised workloads and services which felicitates both declarative configuration and automation. It is one of the largely growing automation tool with a wide support of tools.

Kubernetes has wide range of features which includes

- a container platform
- a microservices platform
- a portable cloud platform and a lot more.

K8s as a container platform

Containers has unmatched benefits now a days which will act as a micro services with verity of security features. It is easy to manage if the number of containers are very less in the environment. But if the containers ate more or large scale in numbers we may need a container orchestration engine which is very helpful to deploy,destroy and scale the multiple containers in a quick way. K8s is first released in 2015 by Borg as a cluster management platform and later google was developed this tool as a container management platform. This is written in go language .

K8s as a microservices platform

Kubernetes is an open source orchestrator for deploying containerised applications (microservices). It is also defined as a platform for creating, deploying and managing various distributed applications. These applications may be of different sizes and shapes. Kubernetes was originally developed by Google to deploy scalable, reliable systems in containers via application-oriented APIs. Kubernetes is suitable not only for Internet-scale companies but also for cloud-native companies, of all sizes.

K8s as a cloud platform

Kubernetes supports wide verity of cloud platforms like AWS,Azure, Google cloud to deploy applications . AWS has their own proprietary container orchestrator, Elastic

Container Service (ECS), but it is different than Kubernetes.In AWS kubernetes operations (KOPS) is used to manage kubernetes . KOPS has below advantages

- Deploy HA Kubernetes master nodes
- Upgrade Kubernetes master nodes
- Upgrade, add, or remove worker nodes
- kops is highly configurable
- Supported by Codefresh
-

KOPS needs to be installed separately as it is not readily available in aws management console .

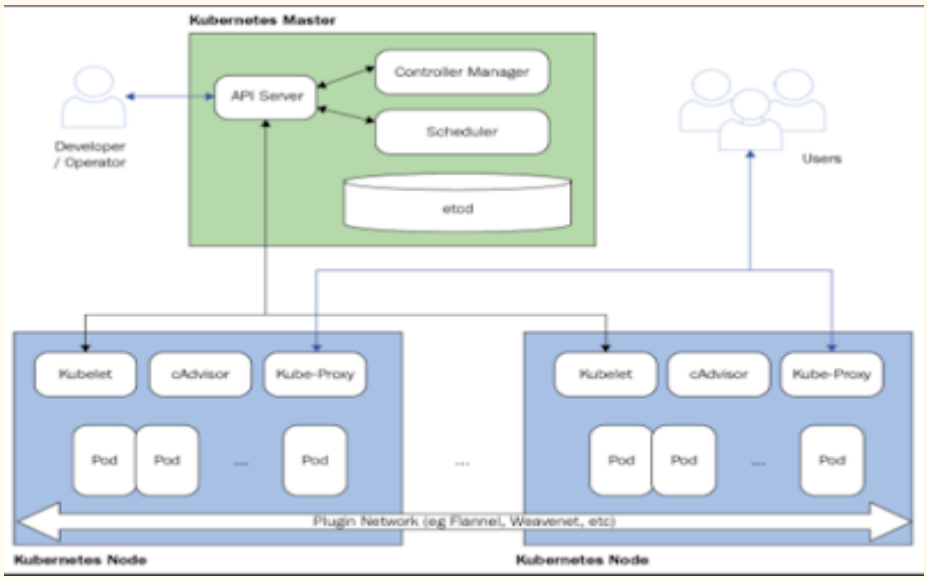
Google Container Engine (GKE) is Google Cloud Platform's managed Kubernetes service.Google manages the Kubernetes master nodes for you in GKE, meaning that you don't have access to them but you also aren't charged for their compute resources.Another feature that sets GCP apart is that they provide a global spanning load balancer built-in which is autoconfigured when services are created.

- Abstracts away and manages Kubernetes master nodes
- Nodes use container optimized image
- Upgrade Kubernetes Master through GCP management console
- Upgrade, add, or remove worker nodes through GCP management console
- Global spanning load balancer built-in
- Easy deployment and automation
- Optional support for Kops
- Streamlined integration with Codefresh

In MS Azure we have built in service called Azure Container Service (ACS) to deploy kubernetes service.ACS acts as more of a deployment template since it does not include features to upgrade a cluster after it has been deployed. In order to upgrade clusters using ACS, you would need to use ACS to create a new cluster, migrate containers to it, and remove the old cluster.One bonus feature of Azure is that creating templates out of your infrastructure is very easy. Once you create and configure all your services you can essentially take a snapshot and deploy it again.

- Deploy HA Kubernetes master nodes
- Add or remove worker nodes through Azure management console
- Supported by Codefresh
- Save infrastructure definition

Kubernetes architecture



Basic kunbernetes setup consist of two types of nodes Master and nodes.

Masters

This type of node is responsible for cluster management, network allocation, quota enforcement, synchronization, and communication. Master nodes act as the main point of contact for clients—be it actual people or some external system. In the simplest setup, there can be only one master, but highly available clusters require at least two to prevent common fail situations. The most important service that masters run is the API.

Nodes

Nodes will have environment to host the docker containers ie it will provide the runtime environment for running pods.

K8 API provides number of components to use various mechanisms in K8 setup. I will explain some of them as below in high level

Namespaces: These resources serve the purpose of separating organizational units of users and their projects in a multitenant environment. Moreover, they are used for more fine-grained access control and quota enforcement. Almost all Kubernetes resources, except Volumes and Namespaces themselves, are namespaced, which means their names must be unique in any given namespace.

Pods: Pods represent a collection of containers and each pod serves as a basic management unit in Kubernetes. All containers in a pod share the same storage volumes and network.

Services: They represent an interface between clients and the actual application running in pods. A service is an IP:port pair which forwards traffic to backend pods in a round-robin fashion. Having a consistent address and port saves clients having to keep up with any transient changes in the cluster.

Replication Controllers (RC): In a nutshell, these resources define how many pods must be replicated. Their definitions include pod templates that describe pods to be launched, and one of the parameters each RC contains is the number of replicas to be maintained. If for some reason one or more of the pods go down, Kubernetes will launch new ones to satisfy this number

Persistent Volumes (PV): These resources abstract actual physical storage systems, be it NFS, iSCSI, or something else. Typically, they are created by a cluster administrator and can be mounted inside a pod using the PVC binding mechanism, which is mentioned later.

Persistent Volume Claims (PVC): PVC represents a request for storage resources. Pod definitions don't use PVs directly; instead, they rely on binding PVs to PVCs, performed by Kubernetes.

Secrets: Used for passing sensitive data such as keys, tokens, and passwords inside pods.

Labels: Labels provide a mechanism for scoping a set of resources using selectors. For example, services use selectors to specify what pods to forward incoming traffic to. When new pods are started with the same label, they are dynamically associated with the service that has their label specified as a selector in its definition

From the services point of view kubernetes can be explained by number of services in detail

In masters
---
etcd : This is a distributed key-value configuration store that holds all metadata and cluster resources. Due to its quorum model, you are advised to run an uneven number of etcd nodes, starting from three in a highly available setup.

kube-apiserver : Service that exposes the Kubernetes API to clients. Its stateless nature enables it to be deployed in a highly available configuration by scaling horizontally.

kube-scheduler : Component that governs the placement of newly created pods on nodes. This procedure takes into account such factors as hardware/policy limitations, data locality, and affinity rules. It is worth noting that from the cluster point of view, masters are no different from any other node and thus can be eligible for running pods, although best practices suggest not putting additional strain on master nodes and dedicating them only to management functions.

kube-controller-manager : The component that runs various controllers—some of them are replication controllers that maintain the required number of running pods, node controllers for discovering nodes that went down, a volume controller for binding PVs to PVCs, and an endpoints controller that binds services and pods together.

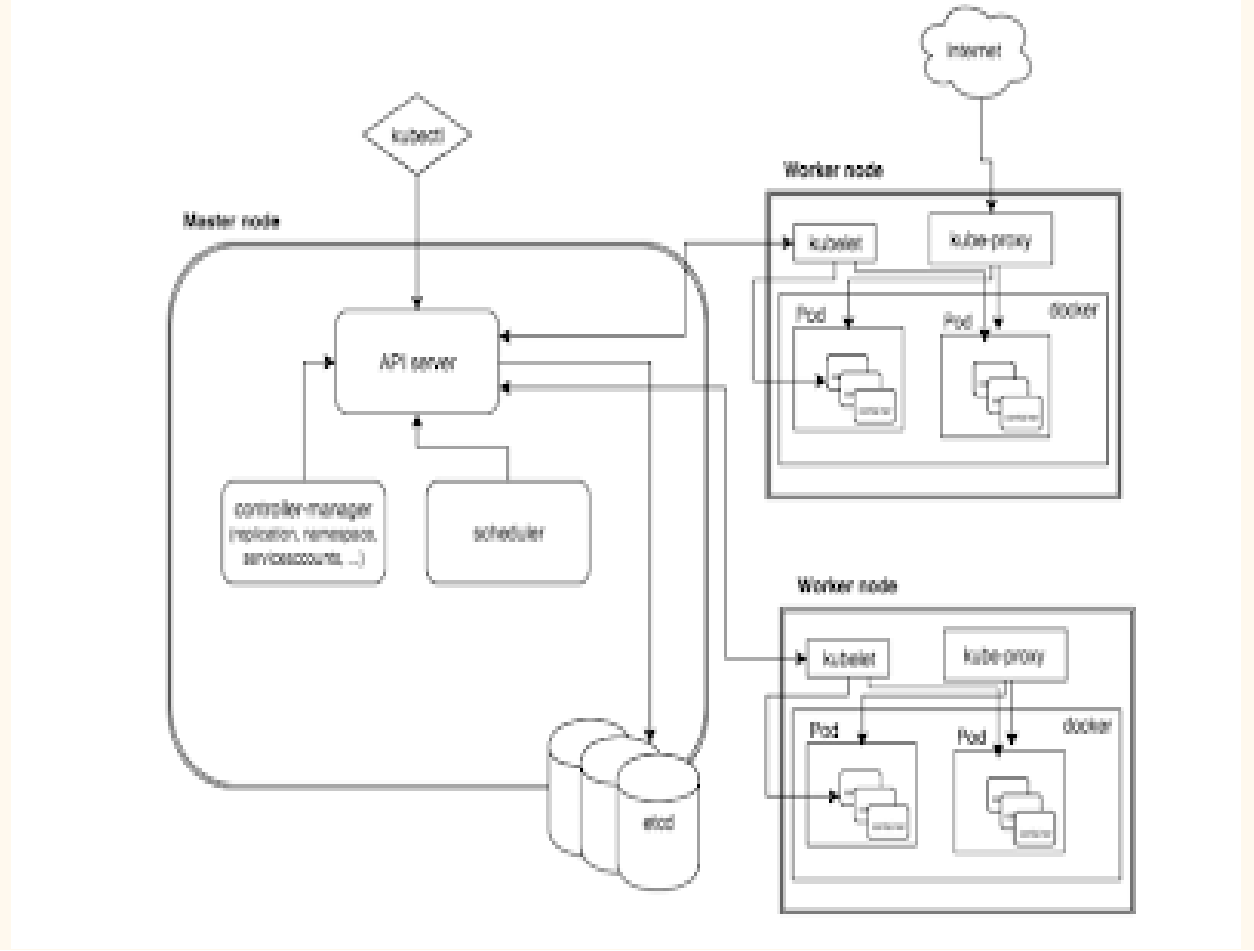
cloud-controller-manager : Service that provides integration with underlying cloud providers, such as DigitalOcean and Oracle Cloud Infrastructure.

In nodes
---

kubelet: This service uses a pod specification to manage its pods and conduct periodic health checks.

kubeproxy: This component implements service abstraction by providing TCP and UDP forwarding capabilities across a set of backend pods.

Container runtime environment: This component is represented in Kubernetes by an underlying container technology. At the time of writing, Kubernetes supports docker and rkt as runtimes


Internet
kubectl
Worker node
Master node
kubelet
kube-proxy
Pod
Pod
docker
API server
controller-manager
(replication, namespace,
serviceaccounts, ...)
scheduler
Worker node
kubelet
kube-proxy
Pod
Pod
docker
etcd