Introduction to Kubernetes

What is Kubernetes?

- Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

- Google Kubernetes is highly flexible container tool to deliver even complex applications, consistently. Applications 'run on clusters of hundreds to thousands of individual servers.“

- Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.

- Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.

Ref: https://kubernetes.io/

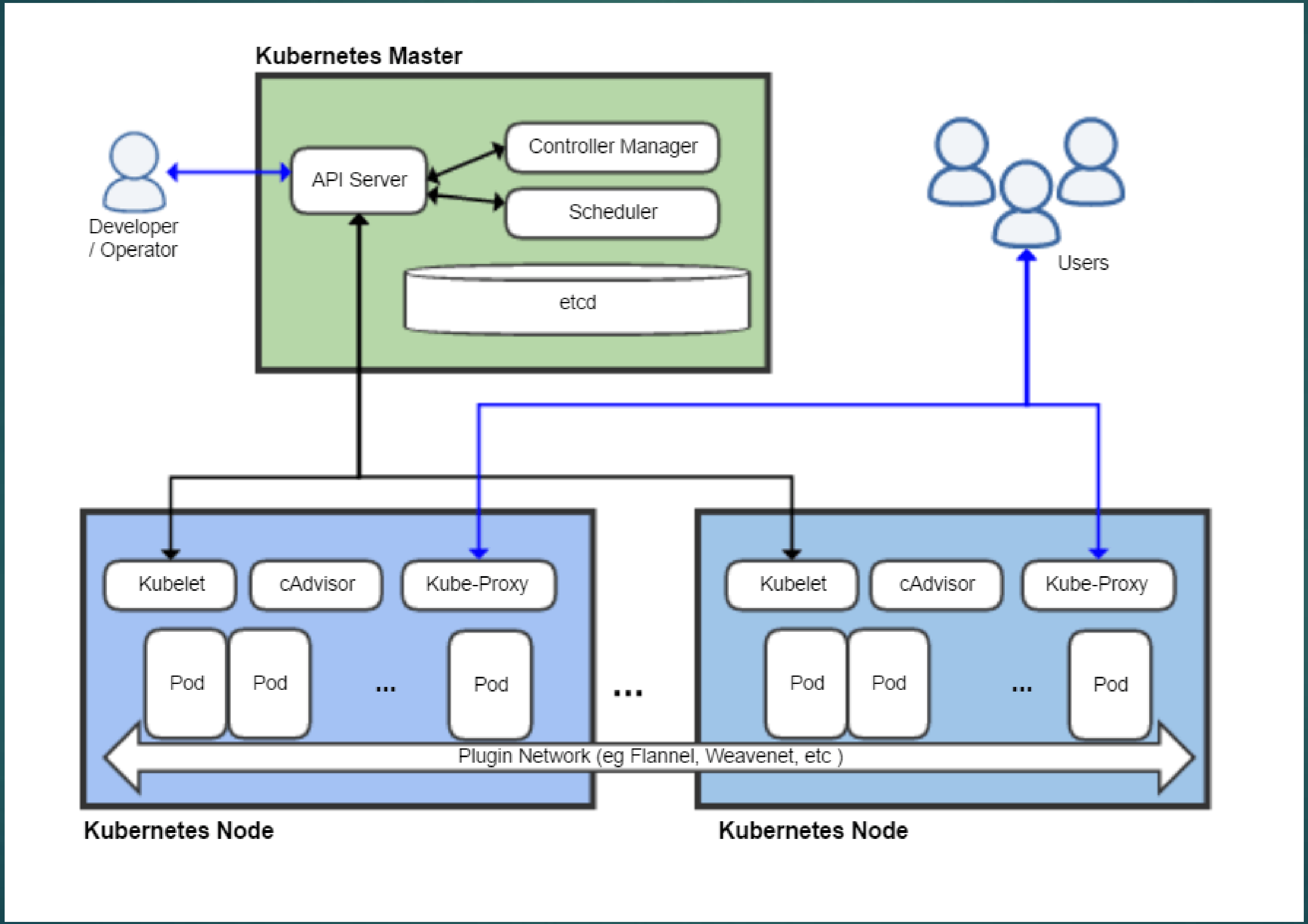
Why Kubernetes?

- Kubernetes can run on-premises bare metal, OpenStack, public clouds Google, Azure, AWS, etc.
- Helps you to avoid vendor lock issues as it can use any vendor-specific APIs or services except where Kubernetes provides an abstraction, e.g., load balancer and storage.
- Containerization using kubernetes allows package software to serve these goals. It will enable applications that need to be released and updated without any downtime.
- Kubernetes allows you to assure those containerized applications run where and when you want and helps you to find resources and tools which you want to work.

Kubernetes Features

- Automated Scheduling
- Self-Healing Capabilities
- Automated rollouts & rollback
- Horizontal Scaling & Load Balancing
- Offers environment consistency for development, testing, and production
- Infrastructure is loosely coupled to each component can act as a separate unit
- Provides a higher density of resource utilization
- Offers enterprise-ready features
- Application-centric management
- Auto-scalable infrastructure
- You can create predictable infrastructure

Kubernetes Architecture



Kubernetes Components

Cluster:

- It is a collection of hosts(servers) that helps you to aggregate their available resources. That includes ram, CPU, ram, disk, and their devices into a usable pool.

Master:

The master is a collection of components which make up the control panel of Kubernetes. These components are used for all cluster decisions. It includes both scheduling and responding to cluster events.

Node:

- It is a single host which is capable of running on a physical or virtual machine. A node should run both kube-proxy, minikube, and kubelet which are considered as a part of the cluster.

Namespace:

- It is a logical cluster or environment. It is a widely used method which is used for scoping access or dividing a cluster.

Kubernetes Master Components

- The Kubernetes primary is the main controlling unit of the cluster, managing its workload and directing communication across the system. The Kubernetes control plane consists of various components, each its own process, that can run both on a single primary node or on multiple primaries supporting high-availability clusters The various components of Kubernetes control plane are

- API Server
- etcd
- Scheduler
- Controller Manager

Kubernetes Master Components

- etcd: Persistent, lightweight, distributed, key-value data store developed by CoreOS that reliably stores the configuration data of the cluster, representing the overall state of the cluster at any given point of time.

- API server: The API server is a key component and serves the Kubernetes API using JSON over HTTP, which provides both the internal and external interface to Kubernetes. The API server processes and validates REST requests and updates state of the API objects in etcd, thereby allowing clients to configure workloads and containers across Worker nodes

Ref:https://kubernetes.io/docs/concepts/overview/components/

Kubernetes Master Components

- Scheduler: The scheduler is the pluggable component that selects which node an unscheduled pod runs on, based on resource availability. The scheduler tracks resource use on each node to ensure that workload is not scheduled in excess of available resources. For this purpose, the scheduler must know the resource requirements, resource availability, and other user-provided constraints and policy directives such as quality-of-service, affinity/anti-affinity requirements, data locality, and so on. In essence, the scheduler's role is to match resource "supply" to workload "demand".

- Controller manager: A controller is a reconciliation loop that drives actual cluster state toward the desired cluster state, communicating with the API server to create, update, and delete the resources it manages (pods, service endpoints, etc.). The controller manager is a process that manages a set of core Kubernetes controllers. One kind of controller is a Replication Controller, which handles replication and scaling by running a specified number of copies of a pod across the cluster. It also handles creating replacement pods if the underlying node fails. Other controllers that are part of the core Kubernetes system include a DaemonSet Controller for running exactly one pod on every machine (or some subset of machines), and a Job Controller for running pods that run to completion, e.g. as part of a batch job.

Ref : https://kubernetes.io/docs/concepts/overview/components/

Kubernetes Node Components

- Kubelet: Kubelet is responsible for the running state of each node, ensuring that all containers on the node are healthy. It takes care of starting, stopping, and maintaining application containers organized into pods as directed by the control plane.Kubelet monitors the state of a pod, and if not in the desired state, the pod re-deploys to the same node. Node status is relayed every few seconds via heartbeat messages to the primary. Once the primary detects a node failure, the Replication Controller observes this state change and launches pods on other healthy nodes.

- Kube-proxy: The Kube-proxy is an implementation of a network proxy and a load balancer, and it supports the service abstraction along with other networking operation.[16] It is responsible for routing traffic to the appropriate container based on IP and port number of the incoming request.

- Container runtime: A container resides inside a pod. The container is the lowest level of a micro-service, which holds the running application, libraries, and their dependencies. Containers can be exposed to the world through an external IP address. Kubernetes supports Docker containers since its first version, and in July 2016 rkt container engine was added.[31]

Kubernetes Add-Ons

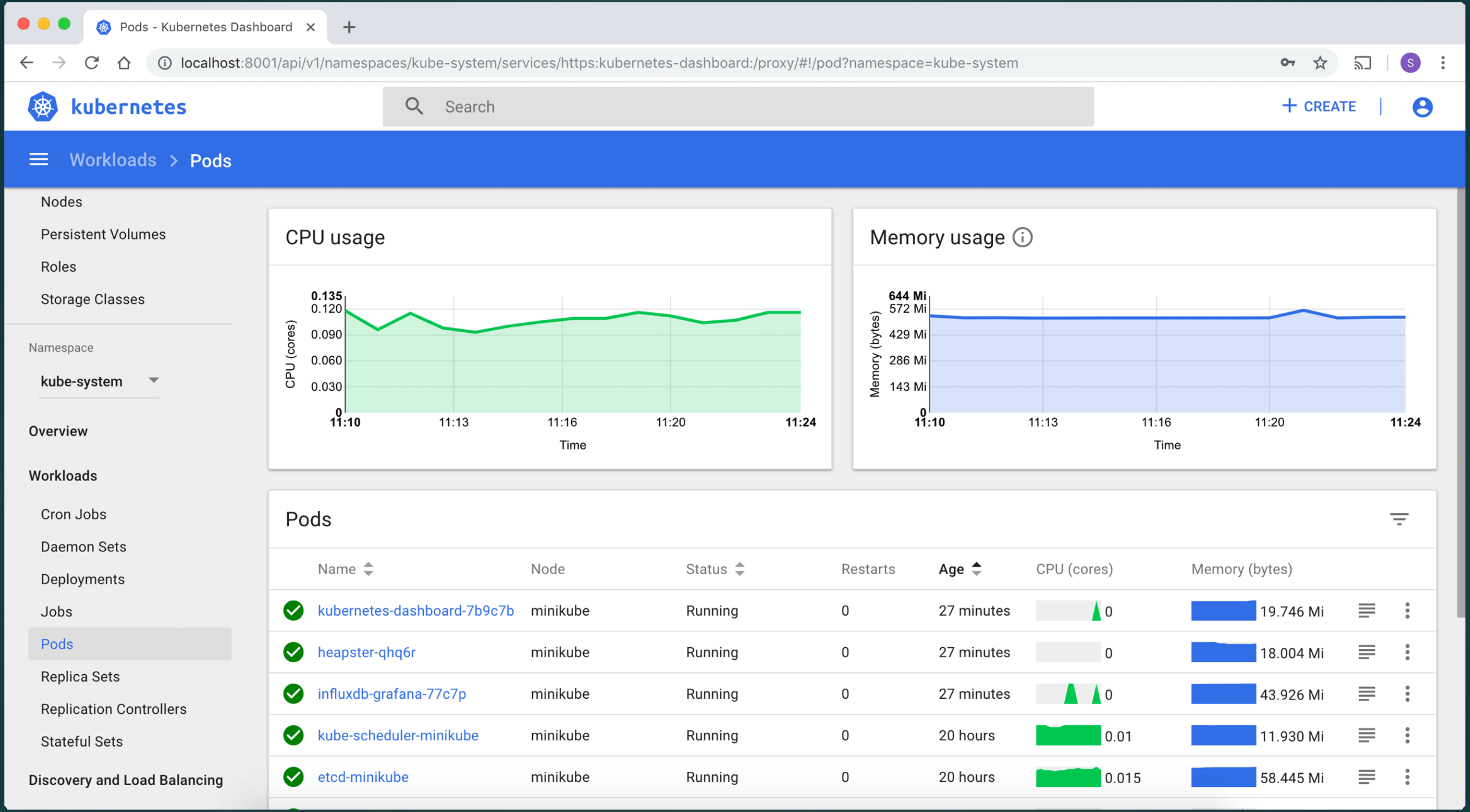
Add-ons operate just like any other application running within the cluster: they are implemented via pods and services, and are only different in that they implement features of the Kubernetes cluster

- DNS: All Kubernetes clusters should have cluster DNS; it is a mandatory feature. Cluster DNS is a DNS server, in addition to the other DNS server(s) in your environment, which serves DNS records for Kubernetes services. Containers started by Kubernetes automatically include this DNS server in their DNS searches.

- Web UI: This is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage and troubleshoot applications running in the cluster, as well as the cluster itself.

Ref:https://kubernetes.io/docs/concepts/overview/components/#addons

Kubernetes Dashboard



Pods

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
kubernetes-dashboard-7b9c7b	minikube	Running	0	27 minutes	0	19.746 Mi
heapster-qhq6r	minikube	Running	0	27 minutes	0	18.004 Mi
influxdb-grafana-77c7p	minikube	Running	0	27 minutes	0	43.926 Mi
kube-scheduler-minikube	minikube	Running	0	20 hours	0.01	11.930 Mi
etcd-minikube	minikube	Running	0	20 hours	0.015	58.445 Mi

Ref :https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

Kubernetes Add-Ons

- Container Resource Monitoring: Providing a reliable application runtime, and being able to scale it up or down in response to workloads, means being able to continuously and effectively monitor workload performance. Container Resource Monitoring provides this capability by recording metrics about containers in a central database, and provides a UI for browsing that data. The cAdvisor is a component on a slave node that provides a limited metric monitoring capability. There are full metrics pipelines as well, such as Prometheus, which can meet most monitoring needs.

- Cluster-level logging: Logs should have a separate storage and lifecycle independent of nodes, pods, or containers. Otherwise, node or pod failures can cause loss of event data. The ability to do this is called cluster-level logging, and such mechanisms are responsible for saving container logs to a central log store with search/browsing interface. Kubernetes provides no native storage solution for log data, but one can integrate many existing logging solutions into the Kubernetes cluster.

THANK YOU