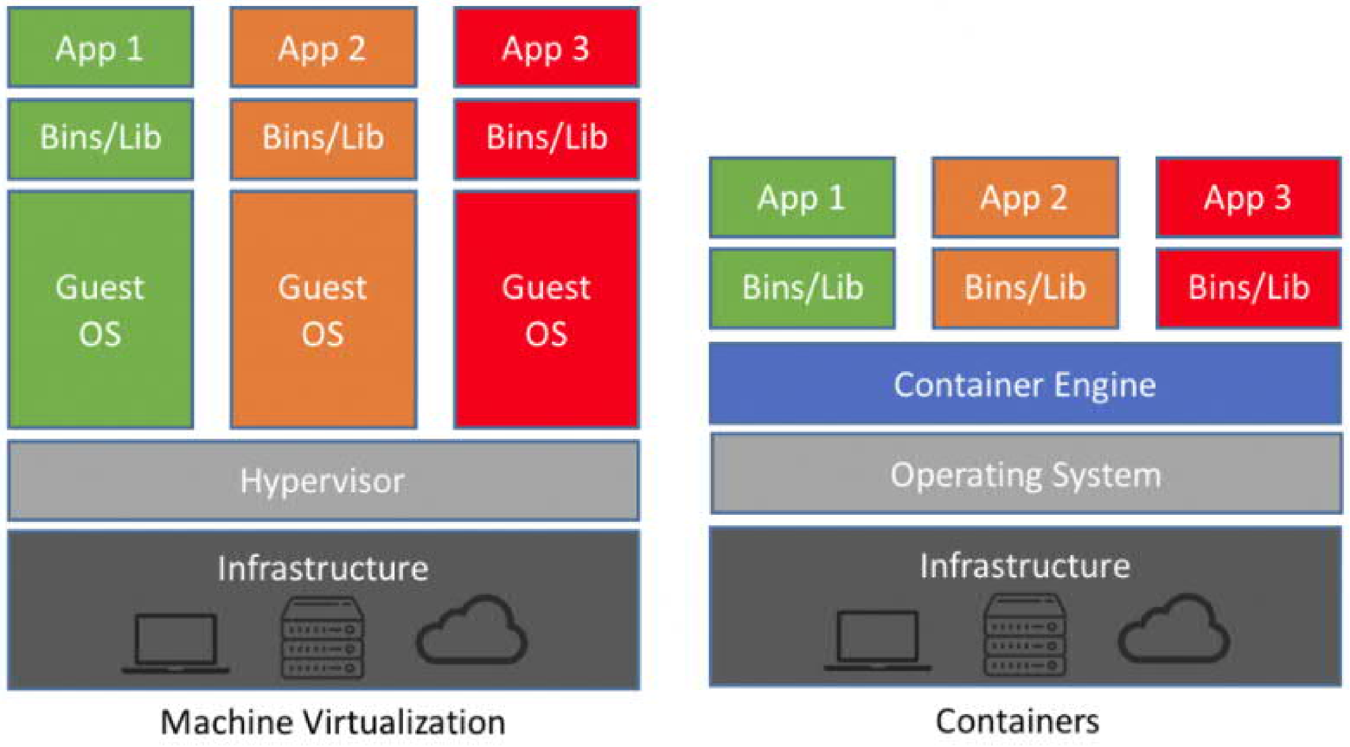
KUBERNETES PRIMER

An INTRODUCTION to CONTAINER ORCHESTRATION TOOL

Difference between Virtual Machine & Container



Virtual Machines abstract host physical infrastructure resources through hypervisor. Each VM runs isolated on separate Guest OS

A container is a self-sustained process having the source code, binaries/dependencies together running on a cloud infra. Compared to a VM, a container engine interact with the kernel properties of the host system & spins up light-weight VMs.

Need for Kubernetes

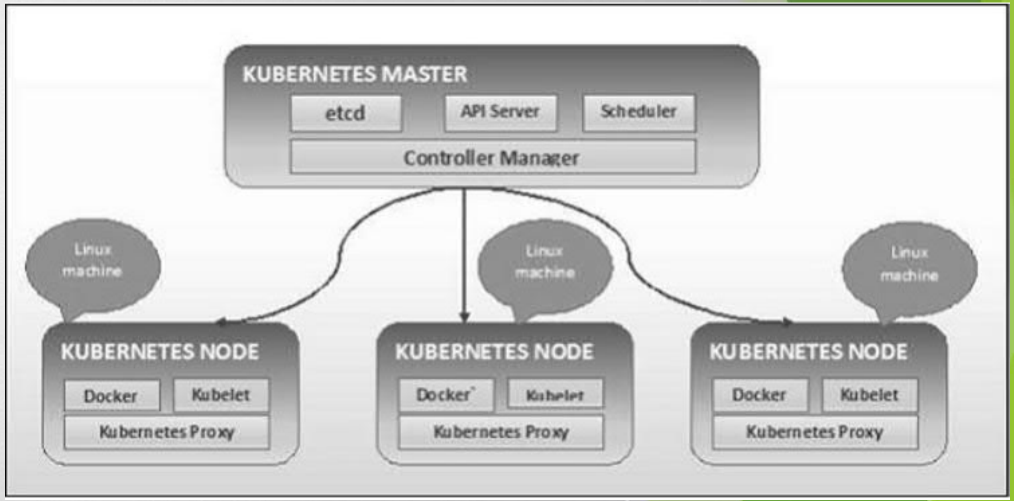
To manage distributed (micro-service based) application & services , their is a need to enable clustering of containers to would provide easy manageability, scalability & agility across the entire Cloud platform. Kubernetes, as an container management solution perfectly fits into the Orchestration solution for automating deployment, scaling and management of containerized applications.

Kubernetes is originally designed by Google and is now maintained by the Cloud Native Computing Foundation. It aims to provide a platform for automating deployment, scaling, and operations of application containers across clusters of hosts. It works with a range of container tools, including Docker, Rockt, CoreOS.

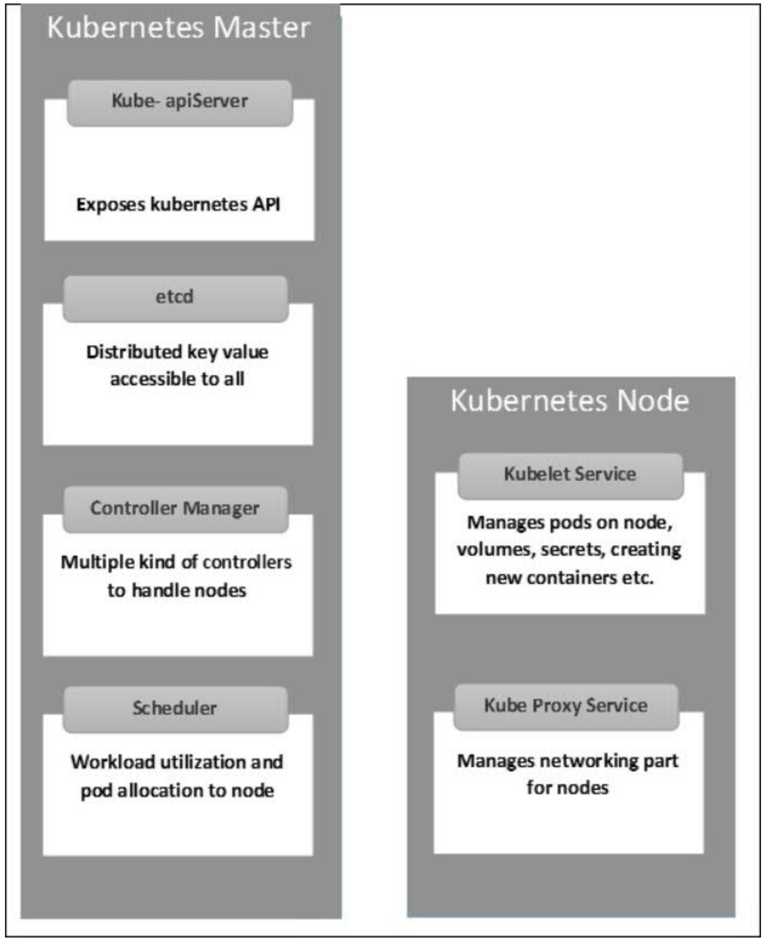
One of the key components of Kubernetes is, it can run application on clusters of physical and virtual machine infrastructure. It also has the capability to run applications on cloud. It helps in moving from host-centric infrastructure to container-centric infrastructure.

Kubernetes Architecture

- **Node** – It is the host that container runs on.
- **Pod** – It is the smallest deployment unit in Kubernetes that contains one or more managed containers. Each pod has its own unique IP address and storage namespaces. All containers share these networking and storage resources. One of the characteristics mentioned in this presentation is that pods are "mortal."
- **Deployment** - A Deployment specifies how many instances of a pod will run. A YAML file is used to define a Deployment. Kubernetes can make sure that the number of Pods that a user specifies is always up and running in the system.
- **Service** - A Kubernetes Service is an abstraction which defines a logical set of Pods and a policy by which to access them. Service are categorized in terms of ClusterIP & NodePorts. Cluster IP is internal to Kubernetes, and the NodePorts are the published IP addresses for external users to access the services



Kubernetes – Master & Node Structure



Kubernetes Master
Kube- apiServer
Exposes kubernetes API
etcd
Distributed key value accessible to all
Controller Manager
Multiple kind of controllers to handle nodes
Scheduler
Workload utilization and pod allocation to node
Kubernetes Node
Kubelet Service
Manages pods on node, volumes, secrets, creating new containers etc.
Kube Proxy Service
Manages networking part for nodes