Kubernetes

- Kubernetes is an Opensource container management tool which automates container deployment, container scaling, and Load Balancing.
- It schedules, runs and manages isolated containers which are running on Virtual/Physical/Cloud machines.
- All top cloud providers support “**Kubernetes**”.

History:

- Google developed an internal system called “**BORG**” (later renamed as “**OMEGA**”) to deploy and manage thousands of google applications and services on their cluster.
- In 2014 google introduced Kubernetes as an Open Source Platform written in “GoLang” and later donated to “**CNCF**”(Cloud Native Computing Foundation)

Online Platform for K8S(Kubernetes):

1. Kubernetes Play Ground
2. Play with K8S
3. Play with Kubernetes classroom

Cloud based K8S services:

1. GKE (Google Kubernetes Services)
2. AKS (Azure Kubernetes Services)
3. Amazon EKS (Elastic Kubernetes Services)

Kubernetes Installation Tools:

1. Minicube
2. Kubeadm

Problem with scaling up the containers:

1. Containers cannot communicate with each other.
2. Auto scaling and Load Balancing was not possible
3. Containers have to be managed carefully

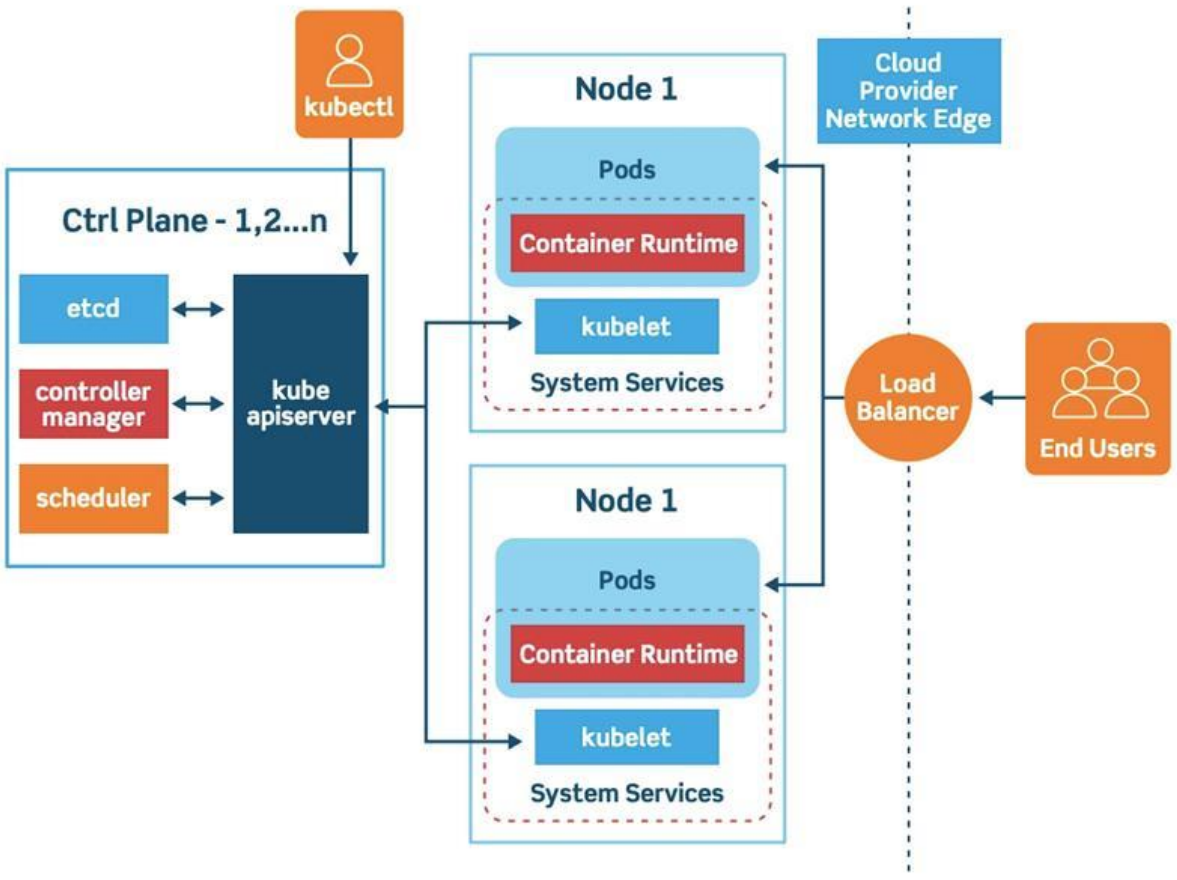
Features of Kubernetes:

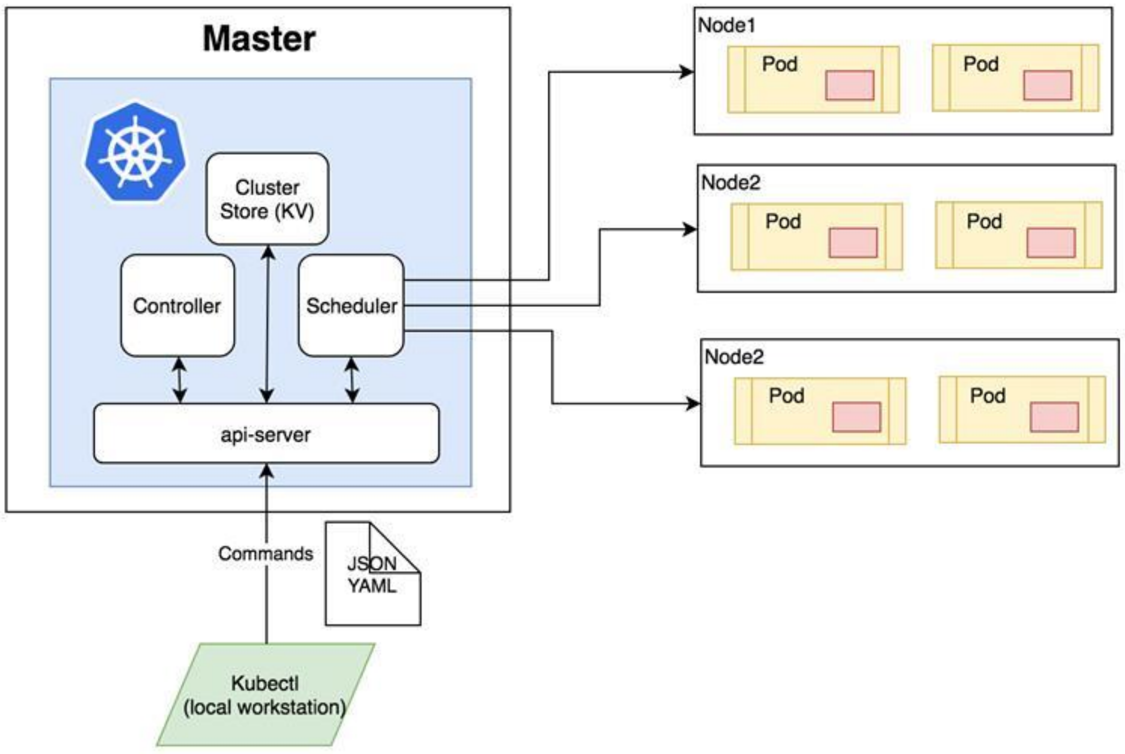
- Orchestration (Cluster of any number of containers running on different networks)
- Auto Scaling (Vertical and Horizontal)
- Auto Healing
- Load Balancing

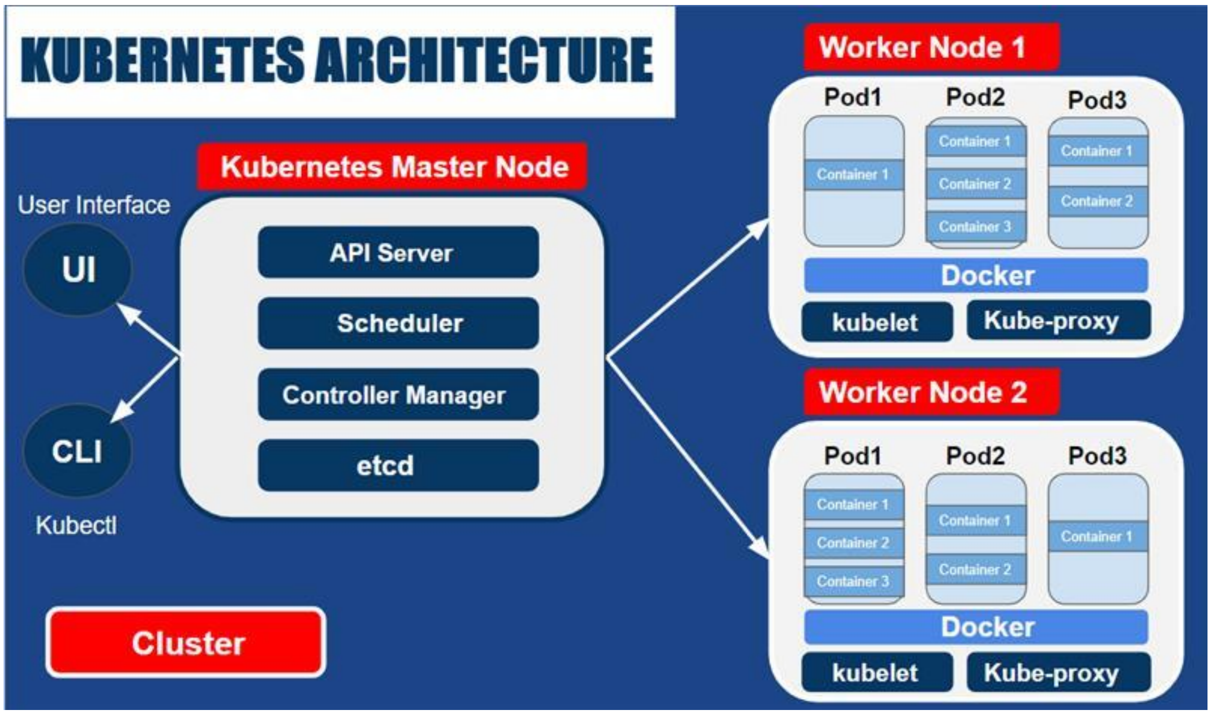
- Platform Independent (Cloud/Virtual/Physical)
- Fault Tolerance (Node/POD Failure)
- Rollback (Going back to previous version)
- Health Monitoring of Containers
- Batch execution (one time, sequential, parallel)

Kubernetes Vs Docker Swarm:

S.No	**Features**	**Kubernetes**	**Docker Swarm**
1	Installation and Cluster configuration	Complicated and time consuming	Fast and Easy
2	Supports	K8S can work with almost all container types like: rocket, docker, ContainerD.	Works with Docker only
3	GUI	GUI is available.	Not available
4	Data Volumes	Can only shared with containers in same POD	Can be shared with any other container
5	Updates and Rollback	Process scheduling to maintain services while updating	Progressive updates and service health monitoring throughput the update.
6	Auto Scaling	Supports Vertical and Horizontal auto scaling	No support
7	Logging and Monitoring	In-built tool present for monitoring	Uses third party tools like splunk


kubectl
Ctrl Plane - 1,2...n
etcd
controller manager
scheduler
kube apiserver
Node 1
Pods
Container Runtime
kubelet
System Services
Node 1
Pods
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users


Master
Cluster Store (KV)
Controller
Scheduler
api-server
Commands
JSON YAML
Kubectl (local workstation)
Node1
Pod
Pod
Node2
Pod
Pod
Node2
Pod
Pod



Kubernetes Architecture:

Working with Kubernetes:

- We create a Manifest file in YAML/JSON.
- Apply this to the Cluster (to Master Node) to bring it into the desired state.
- Pods run on Worker nodes, which is controlled by Master Node.

Role of the Master Node:

- Kubernetes Cluster contains containers running (or) bare metal (or) VM Instances (or) Cloud Instances (or) any combinations of instances.
- Kubernetes designates one or more of these instances as master node and all remaining as worker nodes.
- The master is now going to run set of K8S processes. These processes will ensure smooth functioning of cluster and are called "Control Plane".
- We can have multiple masters for **"High Availability"(HA)**.
- Master runs control plane to handle cluster smoothly.
  Components of Control Plane (Master):
  - Kube-API server
  - Etcd
  - Kube-scheduler
  - Controller/control Manager
- **Kube-API Server** (For all communications):
  - This Kube-API Server directly interacts with the User (i.e. we apply yaml/json manifest to Kube-API server).
  - This Kube-API server is meant to scale automatically as per load.
  - Kube-API server is front end of control plane.
- **etcd**:
  - Stores meta-data and status of cluster.

   - Etcd is consistent and high availability store (key value store)
   - Source of touch for cluster state (info about state of cluster)
   - Etcd has following features:
      - **Fully replicated**: The entire state is available on every node in the cluster
      - **Secure**: Implements automatic TLS(Transport Layer Security) with optional client certificate authentication.
      - **Fast**: Bench marked at 10,000 writes per second.
- **Kube-Scheduler (Action)**:
   - When users make request for the creation and management of pods, kube-scheduler is going to take action on these requests.
   - Handles pod creation and management.
   - Kube-scheduler **match or assign** any worker node to create and run pods.
   - A scheduler watches for **newly created** pods that have no node assigned to a task for every pod that the scheduler discovers.
   - Scheduler is responsible to find **best** node for the pod to run on.
   - Scheduler gets the information for the **hardware configuration** from configuration files and schedules the pods on the nodes accordingly.
- **Controller/Control Manager**:
   - Make sure actual state of cluster matches to the desired state **(actual=desired)**
   - Two possible choices for control manager
      1. If K8S is on cloud, then it will be cloud control manager.
      2. If K8S is on non-cloud, then it will be Kube-Control Manager.
   - Components on Master that run controller:
      - **Node Controller**: For checking the cloud provider to determine if a node has been detected in the cloud after it stops responding.
      - **Route Controller**: Responsible for setting up network, routes on our cloud.
      - **Service Controller**: Responsible for load balancers on our cloud against services of type load balancers.
      - **Volume Controller**: For creating, attaching, mounting volumes and interacting with the cloud provider to orchestrate volumes.

Role of Worker Node (or) Minion: **(Available by Default)**

Worker Node is going to run three important pieces of software or processes:

- **Kubelet**:
   - Agent running on the Node
   - Listens to K8S Master (example: pod creation request)
   - Uses port # 10255
   - Send success or failure reports to master
- **Container Engine** (Docker in our scenario):
   - Works with Kubelet
   - Pulling images
   - Start/stop containers
   - Exposing containers on port specified in Manifest
- **Kube-Proxy**:
   - Assign IP to each pod

  - It is required to assign IP addresses to pods (Dynamic-DHCP)
  - Kube-proxy runs on each worker node, and this makes sure that each pod will get its own unique IP address.

POD:

- Smallest unit in K8S
- POD is one or group of containers that are deployed together on the same host (Worker Node).
- A Cluster is a group of Nodes.
- A Cluster has atleast one worker Node and Master Node.
- In K8S the control unit is the POD, not containers.
- POD runs on Node which is controlled by the Master Node.
- K8S only knows about PODs (doesn't know about individual containerization tools)
- Cannot start containers without a POD.
- One POD usually contains one container.

Multi-Container PODs:

- Share access to memory space.
- Connect to each other using local host <container port>.
- Share access to the same volume.
- Containers within POD are deployed in an "**All or Nothing**" manner.
- Entire POD is hosted on the same node (scheduler will decide about on which node it need to be hosted).

POD Limitations:

- No Auto-Healing or Auto-Scaling
- POD crashes

Higher-Level K8S Objects:

- Replication set:- Scaling and Healing
- Deployment:- Versioning and rollback
- Service:- static (non-ephemeral) IP & Networking
- Volume:- Non-ephemeral storage

Kubernetes need the below services to be installed based on where it is deployed:

1. Kube-ctl: in a single cloud usage
2. Kube-adm: On-premise
3. Kube-fed: Federated (hybrid)

Installation of Kubernetes (K8S):

- Login to AWS account and create 3 instances with below configuration:
    - OS: ubuntu 16.04
    - T2.medium
      Note: Master must have 2 VCPU and 4 GB RAM
    - One Instance for Master Node
    - Another two Instances for Minions(Worker Nodes)
    - Between the Nodes we need HTTPS secure protocol to be allowed. So create security group based on it.
- List of commands: (this has to be run on Master and Worker Nodes)
    - sudo su
    - apt-get update
    - apt-get install apt-transport-https
        - this https is required for intra-cluster communication (particularly from control plane to individual PODs)
- Now install Docker on all the three instances. List of commands to be install & start Docker engine on all the Nodes:
    - apt install docker.io -y
    - docker --version
    - systemctl start docker
    - systemctl enable docker
- setup open-GPG key. This is required for intra-cluster communication. It will be added to source key on each node. i.e. when K8S sends signals into host, it is going to accept that information because the open-GPG key is present on every node in the source key.
    - sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
    - nano /etc/apt/sources.list.d/kubernetes.list
        - deb http://apt.kubernetes.io/ kubernetes-xenial main
    - apt-get update
    - apt-get install -y kubelet kubeadm kubectl kubernetes-cni
- Only on Master Node:
    - Boot strapping the Master Node.
    - To initialise the k8s cluster run below command:
        - Kubeadm init
    - We will get one long command started from kubeadm, copy all those details to Notepad.
    - Create both kube and its parent directories (-p)
        - mkdir -p $HOME/.kube
    - Copy configuration to kube ($HOME/.kube) directory (in config file).

        - cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
    - Provide permission to kube user to the config file that after copying
        - chown $(id -u):$(id -g) $HOME/.kube/config
- Deploy Flannel Node network of its repository path. Flannel is going place a binary in each node for its communication.
    - kubectl apply -f https://raw.githubusercontent.com/cor…
    - kubectl apply -f https://raw.githubusercontent.com/cor…
- Run the long commands that are copied to the Notepad (kubeadm commands from the Master Node) on all the worker nodes to boot stap the communication between the Master and worker Nodes.
- Run the below command on the Master Node to check the worker Nodes connected:
    - Kubectl get nodes.