==========

Kubernetes

==========

Docker : Containerization

=> Containerization means packaging our "application + dependencies" as single unit for execution.

Kubernetes / Open Shift : Orchestration

=> Orchestration means managing containers

(Create / Stop / Update / Delete / Scale Up / Scale Down / Monitor)

-> Kubernetes is also called as K8S

-> Kubernetes developed by Google and donated to CNCF.

-> Kubernetes developed by using GO programming language

-> Using K8S we can orchestrate (manage) Docker containers

-> K8S will follow Cluster Architecture

Note: Cluster means group of servers

Master Node : It will control worker nodes and will assign task to worker nodes

Note: In K8S cluster, Master Node is also called as Control Plane.

Worker Nodes : They will listen to master node and will perform task given by master node.

Note: Containers creation will happen on Worker Nodes.

==========

Kubernetes Architecture Components

==========

1) Control Plane

- 1.1) API Server
- 1.2) Schedular
- 1.3) Controller Manager
- 1.4) ETCD

2) Worker Node

- 2.1) Kubelet
- 2.2) Kuby Proxy
- 2.3) POD
- 2.4) Container
- 2.5) Docker Engine

=> To deploy applications in k8s cluster we will use 'kubectl' (CLI)

=> 'API Server' will recieve request from 'kubectl' and will store in 'etcd'

=> 'etcd' is k8s database which is used to store requests details

=> Schedular will check pending requets in 'etcd' and will talk to 'kubelet' to schedule POD creation.

=> 'Controller-Manager' will manage tasks in K8S cluster.

=> 'Kubelet' is worker node agent which will maintain worker node information

=> 'Kube-proxy' will provide network that is requied for cluster communication

=> POD is a smallest building block that we can deploy in K8S cluster

=> In K8S, docker containers will be created inside POD.

Note: POD is also called as Runtime Instance in K8S cluster.

==========

K8S cluster Setup

==========

=> We have several ways to setup K8S cluster

1) Mini Kube : Single Node Cluster (Only for practice)

2) Self Managed Cluster (Kubeadm) : Multi Node Cluster

3) Provider Managed Cluster : AWS EKS, AZURE AKE, GCP GKE

==========

EKS Cluster Setup :

==========

https://github.com/ashokitschool/DevOps-Documents

==========

Kubernetes Components

==========

1) POD

2) Deployment

3) Service : It is used to expose our PODS

1) Cluste IP
2) Node Port
3) Load balancer

==========

Deployment manifest yml

==========

$ vi javadeployment.yml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: javawebappdeployment
spec:
  replicas: 2
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: javawebapp
  template:
    metadata:
      name: javawebapppod
      labels:
        app: javawebapp
    spec:
      containers:
      - name: javawebappcontainer
        image: ashokit/javawebapp
        ports:
          - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: javaweappsvc
spec:
```

```
  type: LoadBalancer
  selector:
    app: javawebapp
  ports:
    - port: 80
      targetPort: 8080
...
```

==========

kubectl apply -f <yml>

kubectl get pods

kubectl get service

kubectl logs <pod-name>

kubectl get deployment

==========

We can access our application using Load balancer URL

URL : lbr-url/java-web-app

==========

Logs Monitoring using EFK : https://youtu.be/8MLcbbfEL1U

==========

==========

EFK Stack

==========

E - Elastic Search

F - Fluent D

K - Kibana

-> Fluent D is responsible to collect logs from PODS and store into Elastic Search.

-> Elastic Search is a log lake in which logs will be stored and it will index logs for faster retrival.

-> Kibana is web application which provides user interface to fetch logs from Elastic Search.

Git Repo For EFK Setup YMLS :
https://github.com/ashokitschool/kubernetes_manifest_yml_files/tree/main/04-EFK-Log

eksctl create cluster --name hostvm-cluster4 --region ap-south-2 --node-type t3.medium --zones ap-south-2a,ap-south-2b

==========

Delete EKS Cluster

==========

$ eksctl delete cluster --name ashokit-cluster1 --region ap-south-1