Topics covered till today

1) What is Orchestration

2) K8S Introduction

3) K8S Advantages

4) K8S Architecture

5) K8S Architecture components

6) K8S Cluster Setup

7) K8S Resources

8) What is POD

9) What is Service (ClusterIP, NodePort & LBR)

10) What is Namespace

=============================================================

what are the big advantages of using the K8S as orchestration

=============================================================

1 self-healing

2 load balancing

3 auto-scaling

if any pod got damaged or pod got deleted K8S should recreate it. that is the self-healing property in K8S

If i delete the namespace - vinodses-ns, whatever resources inside this namespace will be removed, we need to verify whether k8s is automatically

creating the new resources for the existing one. If it is not happening, we can say k8s is not following the property of self-healing

As of now, we have created POD directly using POD Manifest YML.

(kind as Pod)

If we create POD directly then we don't get self-healing capability.

If POD is damaged/crashed/deleted then k8s will not create new POD.

If pod damaged then our application will be down.

Note: We shouldn't create POD directly to deploy our application in k8s.

Note: We need to use k8s resources to create pods.

If we create pod using k8s resources, then pod life cycle will be managed by k8s

==========================================

what are the resources available to create the pods

==========================================

1) Replication controller (outdated one)

2) Replica set

3) Deployment

4) Deamon set

5) Stateful set

These are the resources will manage the pod life cycle

Replication controller will support only single label to identify the pods

whereas the replica set will support multiple pod label for pod identification

===========

Replica set

==========

It is one of the k8s resource which is used to create & manage pods.

ReplicaSet will take care of POD life cycle.

Note: When POD is damaged/crashed/deleted then ReplicaSet will create new POD.

Always It will maintain given no of pods count available for our application.

E.g. replicas: 2

With this approach we can achieve high availability for our application.

By using RS, we can scale up and scale down our PODS count.

---

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: javawebrs

spec:

replicas: 2

selector:

matchLabels:

app: javawebapp

template:

metadata:

name: javawebpod

labels:

app: javawebapp

spec:

containers:

- name: javawebcontainer

image: vinodses/my-web-app

ports:

- containerPort: 8080

# kubectl get all

# kubectl apply -f <yml>

# kubectl get pods

# kubectl get rs

# kubectl delete pod <pod-name>

# kubectl get pods