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

Kubernetes manifest yml syntax

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

---

apiVersion: It represents the version of the resources that you want to create

v1, v3, vi.app3

kind: What type of k8s resource that you want to create e.g.: kind: POD

metadata: More information about your resources will be listed in metadata

spec: In order to create the POD we need to specify the docker image that our container will use, what is the port number that our container will use

...

---

apiVersion: <Version number>

kind: <Resource-type>

metadata: <name>

spec: <container info>

...

Using the below command, we can create the resources

#kubectl apply -f <manifest-yml>

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

Write a Manifest yml file for creating POD

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

---

apiVersion: v1

kind: Pod

metadata:

name: javawebapppod

labels:

app: javawebapp

spec:

containers:

- name: javawebappcontainer

image: vinodses/my-web-app

ports:

- containerPort: 8080

...

We had copied entire things and create a manifest-yml file

vim k8s-pod-manifest.yml

---

apiVersion: v1

kind: Pod

metadata:

name: javawebapppod

labels:

app: javawebapp

spec:

containers:

- name: javawebappcontainer

image: vinodses/my-web-app

ports:

- containerPort: 8080

...

save and exit

#kubectl get pods

it will check is there any pods are running in my cluster

# kubectl apply -f k8s-pod-manifest.yml

Pod resource will be created based on the above command

#kubectl get pods

#kubectl logs <javawebapppod>

To view the pod logs

# kubectl get pods -o wide

It will give you the ip address of the pod, and which worker node that our pod

got created

# kubectl describe pod javawebapppod

It will give the details information about your pods

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

K8s service manifest yml for service resource creation

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

---

apiVersion: v1

kind: Service

metadata:

name: javawebappsvc

spec:

type: LoadBalancer

selector:

app: javawebapp

ports:

- port: 80

targetPort: 8080

...

To check k8s services

# kubectl get service

To create service

# kubectl apply -f <service-manifest-yml>

Note: Once service got created, we can see in EC2 dashboard Load balancer creation.

We can access our application using Load balancer DNS url

URL: LBR-DNS-URL/java-web-app/