Create a simple deployment of the given app with name of your choice and 3 replicas of pods. Check the status of pod by sending request. App should be accessed from outside the cluster. Use Port Forwarding to Access Applications in a Cluster.

dep.yaml

**apiVersion**: apps/v1

**kind**: Deployment

**metadata**:

**name**: usn-nginx-deployment

**labels**:

**app**: usn-nginx

**spec**:

**replicas**: 3

**selector**:

**matchLabels**:

**app**: usn-nginx

**template**:

**metadata**:

**labels**:

**app**: usn-nginx

**spec**:

**containers**:

- **name**: nginx

**image**: 172.1.14.168:5001/nginx

**ports**:

- **containerPort**: 80

Command to deploy:

kubectl apply -f dep.yaml

Command to check pods

kubectl get pods | grep ‘usn’ **\\ type your usn**

Command to expose

kubectl expose deployment usn-nginx-deployment --type=NodePort --name=usn-nginx-service

To get exposed port

kubectl get svc | grep ‘usn’ **\\ type your usn**

http://172.1.14.168:<nodeport>

**.**

kubectl port-forward deployment/usn-nginx-deployment newport:<nodeport>

Demonstrate the updation of image in live container in a pod using command line as well as by updating yaml files

dep.yaml

**apiVersion**: apps/v1

**kind**: Deployment

**metadata**:

**name**: usn-nginx-deployment

**labels**:

**app**: usn-nginx

**spec**:

**replicas**: 3

**selector**:

**matchLabels**:

**app**: usn-nginx

**template**:

**metadata**:

**labels**:

**app**: usn-nginx

**spec**:

**containers**:

- **name**: nginx

**image**: 172.1.14.168:5001/nginx

**imagePullPolicy**: "Always"

**ports**:

- **containerPort**: 80

Command to deploy:

kubectl apply -f dep.yaml

Command to expose

kubectl expose deployment usn-nginx-deployment --type=NodePort --name=usn-nginx-service

Command to update image:

kubectl set image deployment/usn-nginx-deployment nginx=newImage

To check the updated name:

kubectl describe deploy usn-nginx-deployment | grep Image:

Perform the following.

* 1. Create 3 pods with names nginx1, nginx2,nginx3. All of them should have the label app=v1 Show all labels of the pods.
  2. Get only the 'app=v2' pods.
  3. Remove the 'app' label from the pods we created before

kubectl run usn-nginx1 --image=nginx --restart=Never --labels=app=usn-v1

kubectl run usn-nginx2 --image=nginx --restart=Never --labels=app=usn-v1

kubectl run usn-nginx3 --image=nginx --restart=Never --labels=app=usn-v1

kubectl get po --show-labels

kubectl get po -l app=usn-v2

kubectl label po nginx1 nginx2 nginx3 app-

Create a Pod with ubuntu image and a command to echo “YOUR\_NAME” which overrides the default CMD/ENTRYPOINT of the image.

**dep\_ubuntu\_pod1.yaml**

apiVersion: v1

kind: Pod

metadata:

name: ubuntu

labels:

app: ubuntu

spec:

containers:

- name: ubuntu

image: 172.1.14.168:5001/ubuntu

command: ["/bin/bash"]

args: ["-c", "echo MSRIT"]

**kubectl apply -f dep\_ubuntu\_pod1.yaml**

**kubectl logs ubuntu**

**~~kubectl exec --stdin --tty ubuntu -- /bin/bash~~**

**kubectl delete pod ubuntu**

Create a Pod that runs one container. The configuration file for the Pod defines a command and arguments by using environment variables:

**dep\_ubuntu\_pod.yaml**

apiVersion: v1

kind: Pod

metadata:

name: ubuntu

labels:

app: ubuntu

spec:

containers:

- name: ubuntu

image: 172.1.14.168:5001/ubuntu

env:

- name: MESSAGE

value: "MSRIT"

command: ["/bin/echo"]

args: ["$(MESSAGE)"]

**kubectl apply -f dep\_ubuntu\_pod.yaml**

**kubectl logs ubuntu**

**kubectl delete pod ubuntu**