**MICROSERVICES LAB**

**EXERCISES ON KUBERNETIES with solution**

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 -l=app=usn-nginx

Command to expose

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

To get exposed port

kubectl get svc -l=app=usn-nginx

http://172.1.14.168:<nodeport>

**kubectl port-forward deployment/<dep\_name>/<svc\_name> newport:oldport**

kubectl port-forward deployment/usn-nginx-deployment/usn-nginx 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:

Create busybox pod with two containers, each one will have the image busybox and will run the 'sleep 3600' command. Make both containers mount an emptyDir at '/etc/foo'. Connect to the second busybox, write the first column of '/etc/passwd' file to '/etc/foo/passwd'. Connect to the first busybox and write '/etc/foo/passwd' file to standard output. Delete pod.

|  |  |
| --- | --- |
| dep.yaml  apiVersion: v1  kind: Pod  metadata:  creationTimestamp: null  labels:  run: busybox  name: 172.1.14.168:5001/busybox  spec:  dnsPolicy: ClusterFirst  restartPolicy: Never  containers:  - args:  - /bin/sh  - -c  - sleep 3600  image: busybox  imagePullPolicy: IfNotPresent  name: 172.1.14.168:5001/busybox  resources: {}  volumeMounts:  - name: myvolume  mountPath: /etc/foo  - args:  - /bin/sh  - -c  - sleep 3600  image: busybox  name: busybox2  volumeMounts:  - name: myvolume  mountPath: /etc/foo  volumes:  - name: myvolume  emptyDir: {} | Command to deploy:  kubectl apply -f dep.yaml  Connect to the second container:  kubectl exec -it busybox -c busybox2 -- /bin/sh  cat /etc/passwd | cut -f 1 -d ':' > /etc/foo/passwd  cat /etc/foo/passwd  exit  Connect to the first container:  kubectl exec -it busybox -c busybox -- /bin/sh  mount | grep foo # confirm the mounting  cat /etc/foo/passwd  exit  To delete pod  kubectl delete po busybox |

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 deployment with image nginx:1.7.8, called nginx, having 2 replicas, defining port 80 as the port that this container exposes

* 1. Check how the deployment rollout is going
  2. Update the nginx image to nginx:1.7.9
  3. Check the rollout history and confirm that the replicas are OK
  4. Undo the latest rollout and verify that new pods have the old image (nginx:1.7.8)
  5. Do an on purpose update of the deployment with a wrong image nginx:1.91
  6. Verify that something's wrong with the rollout
  7. Return the deployment to the second revision (number 2) and verify the image is nginx:1.7.9
  8. Check the details of the fourth revision

|  |  |
| --- | --- |
| dep.yaml  **apiVersion**: apps/v1  **kind**: Deployment  **metadata**:  **name**: usn-nginx-deployment  **labels**:  **app**: usn-nginx  **spec**:  **replicas**: 2  **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 | kubectl apply -f dep.yaml  kubectl set image deploy usn-nginx-deployment nginx=nginx:1.7.9  kubectl rollout history deploy usn-nginx-deployment  kubectl rollout undo deploy usn-nginx-deployment  # wait a bit  # select one 'Running' Pod  kubectl get po usn-nginx-deployment  kubectl describe po pod-name | grep -i image  # should be nginx:1.7.8  kubectl set image deploy usn-nginx-deployment nginx=nginx:1.91  kubectl rollout status deploy usn-nginx-deployment  kubectl rollout undo deploy usn-nginx-deployment --to-revision=2  kubectl describe deploy usn-nginx-deployment | grep Image:  kubectl rollout status deploy usn-nginx-deployment  # Everything should be OK |

How to expose multiple port in kubernetes services or Multi-Port Services

**deploy\_ports.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: multiport-demo

namespace: default

spec:

replicas: 2

selector:

matchLabels:

app: multiport-demo

template:

metadata:

labels:

app: multiport-demo

spec:

containers:

- name: multiport-demo

image: 172.1.14.168:5001/nginx

---

apiVersion: v1

kind: Service

metadata:

name: multiport-demo

namespace: default

spec:

type: NodePort

selector:

app: multiport-demo

ports:

- port: 80

name: http

targetPort: 80

nodePort: 30030

- port: 443

name: https

targetPort: 80

nodePort: 30031

**kubectl apply -f deploy\_ports.yaml**

**http://172.1.14.168:30030/**

**http://172.1.14.168:30031/**

From the above example the multiport-demo service will be exposed internally to cluster applications on port 80 and externally to the cluster on the node IP address on 30030,30031. It will also forward requests to pods with the label “app: multiport-demo” on port 80 from 80,443

Example of Multi-Container Pod

**deploy\_multicontainer.yaml**

apiVersion: v1

kind: Pod

metadata:

name: mc1

spec:

volumes:

- name: html

emptyDir: {}

containers:

- name: 1st

image: 172.1.14.168:5001/nginx

volumeMounts:

- name: html

mountPath: /usr/share/nginx/html

- name: 2nd

image: 172.1.14.168:5001/debian

volumeMounts:

- name: html

mountPath: /html

command: ["/bin/sh", "-c"]

args:

- while true; do

date >> /html/index.html;

sleep 1;

done

**kubectl apply -f deploy\_multicontainer.yaml**

**kubectl describe po mc1**

**kubectl exec mc1 -c 1st -- /bin/cat /usr/share/nginx/html/index.html**

**kubectl exec mc1 -c 2nd -- /bin/cat /html/index.html**

**kubectl exec --stdin --tty mc1 -c hello -- /bin/sh**

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**

Create a Pod that runs two Containers. The two containers share a Volume that they can use to communicate.

**dep\_2container\_pod.yaml**

apiVersion: v1

kind: Pod

metadata:

name: two-containers

spec:

restartPolicy: Never

volumes:

- name: shared-data

emptyDir: {}

containers:

- name: nginx-container

image: 172.1.14.168:5001/nginx

volumeMounts:

- name: shared-data

mountPath: /usr/share/nginx/html

- name: debian-container

image: 172.1.14.168:5001/debian

volumeMounts:

- name: shared-data

mountPath: /pod-data

command: ["/bin/sh"]

args: ["-c", "echo Hello from the debian container > /pod-data/index.txt"]

**kubectl apply -f dep\_2container\_pod.yaml**

**kubectl get pods**

**kubectl exec -it two-containers -c nginx-container -- /bin/bash**

**curl localhost**