**MICROSERVICES LAB**

**EXERCISES ON KUBERNETIES**

1. 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.

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**: nginx:1.14.2

**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>

1. Demonstrate the usefulness of Kubernetes in place where a pod is crashed due to some error but user requests are fulfilled in the meantime of restarting the pods.

Try to delete one of the pod among 3 replicas and check the automatic creation of another pod.

1. 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**: nginx:1.14.2

**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:

1. 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: busybox  spec:  dnsPolicy: ClusterFirst  restartPolicy: Never  containers:  - args:  - /bin/sh  - -c  - sleep 3600  image: busybox  imagePullPolicy: IfNotPresent  name: 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 |

1. 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-

1. 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**: nginx:1.7.8  **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 |