**Q. 1**

Task

NOTE: "Welcome to the KodeKloud CKAD Lightning Lab - Part 1!"  
  
"You can toggle between the questions but make sure that that you click on END EXAM before the the timer runs out.  
While this test environment is valid for 60 minutes, challenge yourself and try to complete all 5 questions within 30 minutes! To pass, correctly complete at least 4 out of 5 questions.Good Luck!!!"  
  
  
  
  
Create a Persistent Volume called log-volume. It should make use of a storage class name manual. It should use RWX as the access mode and have a size of 1Gi. The volume should use the hostPath /opt/volume/nginx

Next, create a PVC called log-claim requesting a minimum of 200Mi of storage. This PVC should bind to log-volume.

Mount this in a pod called logger at the location /var/www/nginx. This pod should use the image nginx:alpine.

Solution

Solution manifest file to create a Persistent Volume called log-volume as follows:-

apiVersion: v1

kind: PersistentVolume

metadata:

name: log-volume

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany

storageClassName: manual

hostPath:

path: /opt/volume/nginx

then create a Persistent Volume Claim called log-claim as follows:-

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: log-claim

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 200Mi

storageClassName: manual

Check the bind status of PV and PVC by running the following command:-

root@controlplane:~$ kubectl get pv,pvc

Now, create a new pod called logger with nginx:alpine image as follows:-

---

apiVersion: v1

kind: Pod

metadata:

labels:

run: logger

# pod name

name: logger

spec:

containers:

- image: nginx:alpine

name: logger

volumeMounts:

- name: log

mountPath: /var/www/nginx

volumes:

- name: log

persistentVolumeClaim:

claimName: log-claim

Details

log-volume created with correct parameters?

**Q. 2**

Task

We have deployed a new pod called secure-pod and a service called secure-service. Incoming or Outgoing connections to this pod are not working.  
Troubleshoot why this is happening.

Make sure that incoming connection from the pod webapp-color are successful.

Important: Don't delete any current objects deployed.

Solution

Incoming or outgoing connections are not working because of network policy. In the default namespace, we deployed a default-deny network policy which is interrupting the incoming or outgoing connections.  
  
Now, create a network policy called test-network-policy to allow the connections, as follows:-

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: test-network-policy

namespace: default

spec:

podSelector:

matchLabels:

run: secure-pod

policyTypes:

- Ingress

ingress:

- from:

- podSelector:

matchLabels:

name: webapp-color

ports:

- protocol: TCP

port: 80

then check the connectivity from the webapp-color pod to the secure-pod:-

root@controlplane:~$ kubectl exec -it webapp-color -- sh

/opt # nc -v -z -w 5 secure-service 80

Details

Important: Don't Alter Existing Objects!

Connectivity working?

**Q. 3**

Task

Create a pod called time-check in the dvl1987 namespace. This pod should run a container called time-check that uses the busybox image.

1. Create a config map called time-config with the data TIME\_FREQ=10 in the same namespace.  
   1. The time-check container should run the command: while true; do date; sleep $TIME\_FREQ;done and write the result to the location /opt/time/time-check.log.
   2. The path /opt/time on the pod should mount a volume that lasts the lifetime of this pod.

Solution

Create a namespace called dvl1987 by using the below command:-

$ kubectl create namespace dvl1987

Solution manifest file to create a configMap called time-config in the given namespace as follows:-

apiVersion: v1

data:

TIME\_FREQ: "10"

kind: ConfigMap

metadata:

name: time-config

namespace: dvl1987

Now, create a pod called time-check in the same namespace as follows:-

---

apiVersion: v1

kind: Pod

metadata:

labels:

run: time-check

name: time-check

namespace: dvl1987

spec:

volumes:

- name: log-volume

emptyDir: {}

containers:

- image: busybox

name: time-check

env:

- name: TIME\_FREQ

valueFrom:

configMapKeyRef:

name: time-config

key: TIME\_FREQ

volumeMounts:

- mountPath: /opt/time

name: log-volume

command:

- "/bin/sh"

- "-c"

- "while true; do date; sleep $TIME\_FREQ;done > /opt/time/time-check.log"

Details

Pod time-check configured correctly?

**Q. 4**

Task

Create a new deployment called nginx-deploy, with one single container called nginx, image nginx:1.16 and 4 replicas.  
The deployment should use RollingUpdate strategy with maxSurge=1, and maxUnavailable=2.  
  
Next upgrade the deployment to version 1.17.  
  
Finally, once all pods are updated, undo the update and go back to the previous version.

Solution

Run the following command to create a manifest for deployment nginx-deploy and save it into a file:-

kubectl create deployment nginx-deploy --image=nginx:1.16 --replicas=4 --dry-run=client -oyaml > nginx-deploy.yaml

and add the strategy field under the spec section as follows:-

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 2

So final manifest file for deployment called nginx-deploy should looks like below:-

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: nginx-deploy

name: nginx-deploy

namespace: default

spec:

replicas: 4

selector:

matchLabels:

app: nginx-deploy

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 2

type: RollingUpdate

template:

metadata:

labels:

app: nginx-deploy

spec:

containers:

- image: nginx:1.16

imagePullPolicy: IfNotPresent

name: nginx

then run the kubectl apply -f nginx-deploy.yaml to create a deployment resource.  
  
Now, upgrade the deployment's image version using the kubectl set image command:-

kubectl set image deployment nginx-deploy nginx=nginx:1.17

Run the kubectl rollout command to undo the update and go back to the previous version:-

kubectl rollout undo deployment nginx-deploy

Details

Deployment created correctly?

Was the deployment created with nginx:1.16?

Was it upgraded to 1.17?

Deployment rolled back to 1.16?

**Q. 5**

Task

Create a redis deployment with the following parameters:  
  
Name of the deployment should be redis using the redis:alpine image. It should have exactly 1 replica.  
  
The container should request for .2 CPU. It should use the label app=redis.  
  
It should mount exactly 2 volumes.

a. An Empty directory volume called data at path /redis-master-data.  
  
b. A configmap volume called redis-config at path /redis-master.  
  
c. The container should expose the port 6379.  
  
  
The configmap has already been created.

Solution

Solution manifest file to create a deployment redis as follows:-

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: redis

name: redis

spec:

selector:

matchLabels:

app: redis

template:

metadata:

labels:

app: redis

spec:

volumes:

- name: data

emptyDir: {}

- name: redis-config

configMap:

name: redis-config

containers:

- image: redis:alpine

name: redis

volumeMounts:

- mountPath: /redis-master-data

name: data

- mountPath: /redis-master

name: redis-config

ports:

- containerPort: 6379

resources:

requests:

cpu: "0.2"

Details

**TEST AGAIN**