Workloads & Scheduling

Weight: 8

1) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes (IMPORTANT)

Create a new deployment named cache-deployment in the default namespace using a custom image redis: 7.0.13. Ensure that the deployment has the following specifications:

- Set the replica count to 2.
- Set the strategy type RollingUpdate
- Configure the MaxUnavailable field to 30% and the MaxSurge field to 45%.
- Deploy the cache-deployment deployment and ensure that all pods are in a ready state.
- Now, Perform an image upgrade to redis:7.2.1.
- Examine the rolling history of the deployment, and save the Total revision count to the total-revision.txt.

Solution:-

Step 1: Create deployment template

kubectl create deployment cache-deployment --image=redis:7.0.13 --replicas=2 --dry-run=client -o yaml > deployment.yaml

Step 2: Update deployment.yaml, add this under spec

strategy:

type: RollingUpdate

rollingUpdate:

maxUnavailable: 30%

maxSurge: 45%

Step 3: kubectl apply -f deployment.yaml

Step 4: image upgrade to redis:7.2.1

kubectl set image deployment/cache-deployment *=redis:7.2.1
OR

kubectl set image deployment/cache-deployment redis=redis:7.2.1 Here, redis(is container name)=redis:7.2.1

Step 5: rolling history

kubectl rollout history deployment/cache-deployment

Step 6: save the Total revision count to the total-revision.txt

echo 2 > total-revision.txt

Weight: 5

2) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

Within the default namespace, there is a web application deployment named webapp-deployment that relies on an environment variable that can change frequently. You need to manage this environment variable using a ConfigMap. Follow these steps:

- Create a new ConfigMap named webapp-deployment-config-map with the key-value pair APPLICATION=web-app.
- Update the deployment webapp-deployment to utilize the newly created ConfigMap.

Solution:-

Step 1: Create ConfigMap kubectl create configmap webapp-deployment-config-map --from-literal=APPLICATION=web-app Step 2: Edit deployment kubectl edit deploy webapp-deployment Step 3: Replace Fromspec: containers: name: webapp-container image: nginx:latest env: name: APPLICATION To spec: containers: name: webapp-container image: nginx:latest env: - name: APPLICATION valueFrom:

Weight: 2

configMapKeyRef:

3) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

name: webapp-deployment-config-map

There is redis-deploy deployment in redis-ns namespace with replica 1, your task is to scale it to 3.

Solution:-

```
kubectl scale deploy redis-deploy -n redis-ns --replicas=3
```

Weight: 5

4) For this question, please set this context (In exam, diff cluster name)

```
kubectl config use-context kubernetes-admin@kubernetes
```

Currently, the webapp-deployment is running with below sensitive database environment variables directly embedded in the deployment YAML. To enhance security and protect the sensitive data, perform the following steps:

- Create a Kubernetes Secret named db-secret with the sensitive database environment variable values(base64):
 - O Key: DB Host , Value: mysql-host
 - O Key: DB User, Value: root
 - O Key: DB Password, Value: dbpassword
- Update the webapp-deployment to load the sensitive database environment variables from the newly created db-secret Secret.

Solution:-

```
Step 1: Create secret
```

```
kubectl create secret generic db-secret --from-literal=DB Host=mysql-host
  --from-literal=DB User=root \
   -from-literal=DB Password=dbpassword
```

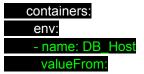
Step 2: Edit deployment

kubectl edit deploy webapp-deployment

Step 3: Replace From-

```
containers:
- env:
 - name: DB Use
 - name: DB_Password
```

To -





Weight: 5

5) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

There is a issue in my-app-deployment.yaml manifest file, fix the issues, After fixing make sure pod are running state.

Note: Don't remove any specification

Solution:-

Step 1: First Change image: nginx:latets To image: nginx:latest

Step 2: Second Changes From-

```
resources:
requests:
memory: "1000Mi"
cpu: "5.0"
limits:
memory: "100Mi"
cpu: "0.5"
To-#give less than limit
```

```
resources:
requests:
memory: "100Mi"
cpu: "0.1"
limits:
memory: "100Mi"
cpu: "0.5"
```

Step 3: kubectl apply -f my-app-deployment.yaml

Weight: 4

6) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

The deployment named <code>video-app</code> has experienced multiple rolling updates and rollbacks. Your task is to total revision of this deployment and record the image name used in <code>3rd</code> revision to file <code>app-file.txt</code> in this format <code>REVISION TOTAL COUNT, IMAGE NAME</code>.

Solution:-

Step 1: Check the Total revision

kubectl rollout history deployment video-app

Step 2: Check the 3rd revision image name

kubectl rollout history deployment video-app --revision=3

Step 3: Save it in file

echo "3,redis:7.0.13" > app-file.txt

Weight: 2

7) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

Fresher deployed a pod named my-pod. However, while specifying the resource limits, they mistakenly given 100Mi storage limit instead of 50Mi

node doesn't have sufficient resources, So change it to 50Mi only.

Solution:-

Step 1: get the pod template

kubectl get pod my-pod -o yaml > pod.yaml

Step 2: Update pod template, From-

resources:

limits:

memory: 100Mi

requests:

memory: 50Mi

To-

resources:

limits:

memory: 50Mi

requests:

memory: 50Mi

Step 3: recreate pod

kubectl replace -f pod.yaml --force

Weight: 6

8) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

Due to a missing feature in the current version. To resolve this issue, perform a rollback of the deployment redis-deployment to the previous version. After rolling back the deployment, save the image currently in use to the rolling-back-image.txt file, and finally increase the replica count to 3."

Solution:-

Step 1: switch to previous version

kubectl rollout undo deployment redis-deployment

Step 2: get the image name

kubectl describe deploy redis-deployment | grep -i image

Step 3: Save it in a file

echo "redis:7.0.13" > rolling-back-image.txt

Step 4: Scale up the pod

kubectl scale deploy redis-deployment --replicas=3

Weight: 2

9) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

Create a deployment named nginx-app-deployment using the nginx image and scale the application pods to 3.

Solution:- kubectl create deployment nginx-app-deployment --image=nginx --replicas=3

Weight: 4

10) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

You need to create a Kubernetes Pod and a Service to host a simple web application that prints "Hello, World!" when accessed. Follow these steps:

Create a Pod named app-pod with the following specifications:

• Container name: app-container

• Container image: httpd:latest

Container port: 80

Create a Service named app-svc with the following specifications:

- Select the Pod with the label app: app-lab.
- Service port: 80
- Target port: 80
- Service type: ClusterIP
- kubectl port-forward to forward a local port to the Pod's port
- Access the web application using curl on another terminal

Solution:-

Step 1: Create pod kubectl run app-pod --image=httpd:latest --port=80

Step 2: Create service kubectl expose pod app-pod --name=app-svc --port=80 --target-port=80 --type=ClusterIP

Step 3: Port forward kubectl port-forward svc/app-svc 8080:80

And curl http://localhost:8080

Weight: 4

11) For this question, please set this context (In exam, diff cluster name)

kubectl config use-context kubernetes-admin@kubernetes

Create a pod named ubuntu-pod using the ubuntu image with the labels set to app=os. Now, create a service named ubuntu-service to expose the ubuntu-pod application within the cluster on port 8080.

Solution:-

Step 1: Create pod kubectl run ubuntu-pod --image=ubuntu --labels=app=os

Step 2: Create service kubectl expose pod ubuntu-pod --name=ubuntu-service --port=8080 --target-port=8080