기출 문제 & 답안

상태	📚CKA 자격증 학습
생성일시	@2022년 4월 1일 오전 7:41
편집자	seongmi Lee
최종 편집일시	@2022년 4월 25일 오후 2:37

실전문제풀이

❓1. Retrieve Error Messages from a Container Log

- Cluster: kubectl config use-context **hk8s**

In the customa namespace, check the log for the nginx container in the custom-app Pod.

Save the lines which contain the text “error” to the file /var/CKA2022/errors.txt.

▼ ❗Answer

```
kubectl get pods  -n customera
kubectl logs custom-app -n customera | grep -i error > /var/CKA2022/error.txt
cat /var/CKA2022/error.txt
```

❓2. Node Troubleshooting

- Cluster: kubectl config use-context **hk8s**

A Kubernetes worker node, named hk8s-w2 is in state NotReady.
Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state, ensuring that any changes are made permanent.

▼ ❗Answer

```
$ kubectl get nodes

$ ssh hk8s-w2
$ sudo -i
# docker ps
# systemctl status docker
# systemctl status kubelet
# systemctl enable --now kubelet
# systemctl status kubelet
# exit
$ exit

$ kubectl get nodes
```

❓3. Count the Number of Nodes That Are Ready to Run Normal Workloads

- Cluster: kubectl config use-context **hk8s**

Determine how many nodes in the cluster are ready to run normal workloads (i.e., workloads that do not have any special tolerations).

Output this number to the file /var/CKA2022/count.txt

▼ ❗Answer

```
kubectl get nodes | grep -i -w ready | wc -l > /var/CKA2022/count.txt
cat /var/CKA2022/count.txt
3
```

❓4. Management Node

- Cluster: kubectl config use-context **k8s**

Set the node named k8s-worker1 as unavailable and reschedule all the pods running on it.

▼ ❗Answer

```
kubectl drain k8s-worker2 --ignore-daemonsets  --force

kubectl get nodes

kubectl get pods
```

❓5. ETCD backup & restore

- 작업 클러스터 : kubectl config use-context **k8s**

First, create a snapshot of the existing etcd instance running at `https://127.0.0.1:2379`, saving the snapshot to `/data/etcd-snapshot.db`.
Next, restore an existing, previous snapshot located at `/data/etcd-snapshot-previous.db`.

The following TLS certificates/key are supplied for connecting to the server with etcdctl:
CA certificate: `/etc/kubernetes/pki/etcd/ca.crt`

Client certificate: `/etc/kubernetes/pki/etcd/server.crt`
Client key: `/etc/kubernetes/pki/etcd/server.key`

▼ ❗Answer

```
ssh k8s-master
sudo -i
sudo ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \
   --cacert=/etc/kubernetes/pki/etcd/ca.crt \
   --cert=/etc/kubernetes/pki/etcd/server.crt \
   --key=/etc/kubernetes/pki/etcd/server.key \
   snapshot save /data/etcd-snapshot.db

sudo ETCDCTL_API=3  etcdctl --data-dir=/var/lib/etcd-previous snapshot restore /data/etcd-snapshot-previous.db
sudo tree /var/lib/etcd-previous/

sudo vi /etc/kubernetes/manifests/etcd.yaml
...
  - hostPath:
      path: /var/lib/etcd-previous
      type: DirectoryOrCreate
    name: etcd-data

sudo docker ps -a | grep etcd
exit
exit
```

❓6. Cluster Upgrade - only Master

- 작업 클러스터 : kubectl config use-context **k8s**

upgrade system : hk8s-m
Given an existing Kubernetes cluster running version `1.22.4`,
upgrade all of the Kubernetes control plane and node components on the master node only to version `1.23.3`.
Be sure to `drain` the `master` node before upgrading it and `uncordon` it after the upgrade.

▼ ❗Answer

```
# kubeadm 업그레이드
sudo yum install -y kubeadm-1.23.3-0 --disableexcludes=kubernetes
kubeadm version

# node components 업그레이드
sudo kubeadm upgrade plan v1.23.3
sudo kubeadm upgrade apply v1.23.3

# 노드 드레인
kubectl drain hk8s-m --ignore-daemonsets

# kubelet과 kubectl 업그레이드
sudo yum install -y kubelet-1.23.3-0 kubectl-1.23.3-0 --disableexcludes=kubernetes
sudo systemctl daemon-reload
sudo systemctl restart kubelet

# 노드 uncordon
sudo kubectl uncordon hk8s-m
```

❓7. Authentication and Authorization

- Cluster : k8s

Context You have been asked to create a new ClusterRole for a deployment pipeline and bind it to a specific ServiceAccount scoped to a specific namespace.
Task:
- Create a new ClusterRole named `deployment-clusterrole`, which only allows to **create** the following resource types:
Deployment StatefulSet DaemonSet

- Create a new ServiceAccount named `cicd-token` in the existing namespace `app-team1`.
- Bind the new ClusterRole `deployment-clusterrole` to the new ServiceAccount `cicd-token`, limited to the namespace `app-team1`.

▼ ❗Answer

```
kubectl create clusterrole deployment-clusterrole --verb=create --resource=deployment,statefulset,daemonset
kubectl get clusterrole deployment-clusterrole

kubectl create serviceaccount cicd-token --namespace=app-team1
kubectl get serviceaccounts --namespace app-team1

kubectl create clusterrolebinding deployment-clusterrolebinding --clusterrole=deployment-clusterrole --serviceaccount=app-team1:ci
kubectl describe clusterrolebindings deployment-clusterrolebinding
```

❓8. Pod 생성하기

- 작업 클러스터 : kubectl config use-context **k8s**

Create a new namespace and create a pod in the namespace

TASK:

- namespace name: cka-exam
- pod Name: pod-01
- image: busybox
- environment Variable: CERT = "CKA-cert"
- command: /bin/sh
- args: -c "while true; do echo $(CERT); sleep 10;done"

▼ ❗Answer

```
kubectl create namespace cka-exam
kubectl get namespaces cka-exam

kubectl run pod-01 --image=busybox  --dry-run=client -o yaml > pod-01.yaml
vi pod-01.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-01
  namespace: cka-exam
spec:
  containers:
  - env:
    - name: CERT
      value: CKA-cert
    image: busybox
    name: pod-01
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo $(CERT); sleep 10;done"]

kubectl apply -f pod-01.yaml
kubectl get pod -n cka-exam
```

❓9. multi-container Pod 생성

- cluster : kubectl config use-context **hk8s**
- Create a pod with 4 containers running : nginx, redis, memcached and consul
  - pod name: eshop-frontend
  - image: nginx
  - image: redis

- image: memcached
- image: consul

▼ ❗Answer

```
kubectl run eshop-frontend --image=nginx  --dry-run=client -o yaml > multi.yaml

vi multi.yaml
apiVersion: v1
kind: Pod
metadata:
  name: eshop-frontend
spec:
  containers:
  - image: nginx
    name: nginx
  - image: redis
    name: redis
  - image: memcached
    name: memcached
  - image: consul
    name: consul

kubectl apply -f multi.yaml
```

❓10. Side-car Container Pod 실행

- 작업 클러스터 : kubectl config use-context **k8s**
- 현재 운영 중인 **eshop-cart-app** Pod의 로그를 Kubernetes built-in logging 아키텍처(예: kubectl logs)에 통합하는 로그 스트리밍 사이드카 컨테이너를 운영하시오.
  - busybox 이미지를 사용하여 price 라는 이름의 side-car container를 기존 eshop-cart-app에 **추가**합니다.
  - 새 price 컨테이너는 다음과 같은 command를 실행해야 합니다.

    Command: /bin/sh, -c, “tail -n+1 -f /var/log/cart-app.log”
  - /var/log에 마운트 된 볼륨을 사용하여 사이드카 컨테이너에서 로그 파일 cart-app.log를 사용해야 합니다.
  - **eshop-cart-app Pod와 cart-app 컨테이너를 수정**하지 마시오

▼ ❗Answer

```
kubectl get pod eshop-cart-app -o yaml > sidecar.yaml
cat sidecar.yaml
apiVersion: v1
kind: Pod
metadata:
  name: eshop-cart-app
spec:
  containers:
  - image: busybox
    name: cart-app
    command: ['/bin/sh', '-c', 'i=1;while :;do  echo -e "$i: Price: $((RANDOM % 10000 + 1))" >> /var/log/cart-app.log; i=$((i+1));
    volumeMounts:
    - name: varlog
      mountPath: /var/log
  volumes:
  - emptyDir: {}
    name: varlog


vi sidecar.yaml
apiVersion: v1
kind: Pod
metadata:
  name: eshop-cart-app
spec:
  containers:
  - image: busybox
    name: cart-app
    command: ['/bin/sh', '-c', 'i=1;while :;do  echo -e "$i: Price: $((RANDOM % 10000 + 1))" >> /var/log/cart-app.log; i=$((i+1));
    volumeMounts:
    - name: varlog
      mountPath: /var/log
    - name: price
```

```
      image: busybox
      args: [/bin/sh, -c, "tail -n+1 -f /var/log/cart-app.log"]
      volumeMounts:
      - name: varlog
        mountPath: /var/log
  volumes:
  - emptyDir: {}
    name: varlog

kubectl delete pod eshop-cart-app
kubectl apply -f multi.yaml
```

❓11. Pod Scale-out

- Cluster: kubectl config use-contex **k8s**

Expand the number of running Pods in "**eshop-order**" to 5.

- namespace: devops
- deployment: eshop-order
- replicas: **5**

▼ ❗Answer

```
kubectl deployment -n devops
kubectl get deployment eshop-order -n devops
kubectl scale deployment eshop-order --replicas=5 --namespace devops
```

❓12. Rolling Update

- Cluster: kubectl config use-context **k8s**

Create a deployment as follows:

- TASK:
  - name: nginx-app
  - Using container nginx with version 1.11.10-alpine
  - The deployment should contain 3 replicas
- Next, deploy the application with new version **1.11.13-alpine**, by performing a **rolling update**
- Finally, rollback that update to the **previous version** 1.11.10-alpine

▼ ❗Answer

```
# create
kubectl create deployment  nginx-app --image=nginx:1.11.10-alpine --replicas=3 --dry-run=client -o yaml > deplyment.yaml
cat deplyment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx-app
  name: nginx-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  strategy: {}
  template:
    metadata:
```

```
      creationTimestamp: null
      labels:
        app: nginx-app
    spec:
      containers:
      - image: nginx:1.11.10-alpine
        name: nginx
        resources: {}
status: {}


kubectl apply -f deplyment.yaml --record

# rolling update
kubectl set image deployment nginx-app nginx=nginx:1.11.13-alpine --record

kubectl rollout history deployment nginx-app

# roll-back
kubectl rollout undo deployment nginx-app
kubectl rollout history deployment nginx-app
```

❓13. Network Policy with Namespace

- 작업 클러스터 : kubectl config use-context **k8s**

Create a new **NetworkPolicy** named allow-port-from-namespace in the existing **namespace devops**.

Ensure that the new NetworkPolicy allows Pods in namespace migops(using label **team=migops**) to connect to port 80 of Pods in namespace devops.

Further ensure that the new NetworkPolicy: does not allow access to Pods, which don't listen on port 80 does not allow access from Pods, which are not in namespace migops

---

▼ ❗Answer

```
vi policy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: devops
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          team: migops
    ports:
    - protocol: TCP
      port:80

kubectl apply -f policy.yaml
```

❓14. Create a persistent volume

- Cluster: kubectl config use-context **k8s**

Create a persistent volume with name app-config, of capacity 1Gi and access mode ReadWriteMany.
The type of volume is hostPath and its location is /var/app-config.

▼ ❗Answer

```
vi pv.yaml
apiVersion: v1
kind: PersistentVolume
```

```
metadata:
  name: app-config
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteMany
  hostPath:
    path: /var/app-config

kubectl apply -f pv.yaml
kubectl get pv
```

❓15. Deploy and Service

- 작업 클러스터 : kubectl config use-context **k8s**

Reconfigure the **existing** deployment `front-end` and add a **port specification named** `http` exposing port `80/tcp` of the existing container nginx.
Create a new service named `front-end-svc` exposing the container port http.
Configure the new service to also expose the individual Pods via a `NodePort` on the nodes on which they are scheduled

---

▼ ❗**Answer**

```
kubectl get deploy front-end -o yaml > front-end.yaml

cat > front-end.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: front-end
spec:
  selector:
    matchLabels:
      app: front-end
  replicas: 2
  template:
    metadata:
      labels:
        app: front-end
    spec:
      containers:
      - name: http
        image: nginx
        ports:
        - name: http
          containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: front-end-svc
spec:
  type: NodePort
  ports:
  - port: 80
    protocol: TCP
    targetPort: http
  selector:
    app: front-end

kubectl delete deploy front-end
kubectl apply -f front-end.yaml
kubectl get deployments.apps,svc
```

❓16. DNS Lookup

- 작업 클러스터 : kubectl config use-context **k8s**

Create a nginx pod called nginx-resolver using image nginx, expose it internally with a service called nginx-resolver-service. Test that you are able to look up the service and pod names from within the cluster. Use the image: busybox:1.28 for dns

lookup.

- Record results in /var/CKA2022/nginx.svc and /var/CKA2022/nginx.pod
- Pod: nginx-resolver created
- Service DNS Resolution recorded correctly
- Pod DNS resolution recorded correctly

▼ ❗ **Answer**

```
#Create a nginx pod
kubectl run nginx-resolver --image=nginx --port=80
kubectl expose pod nginx-resolver --name nginx-resolver-service --port=80
kubectl get svc nginx-resolver-service
  nginx-resolver-service   10.104.150.11
# Test:Service DNS Resolution recorded correctly -> /var/CKA2022/nginx.svc
kubectl run test-nslookup --image=busybox:1.28 -it --restart=Never --rm -- nslookup  10.104.150.11
kubectl run test-nslookup --image=busybox:1.28 -it --restart=Never --rm -- nslookup  nginx-resolver-service > /var/CKA2022/nginx.s

# Test: Pod DNS resolution recorded correctly -> /var/CKA2022/nginx.pod
kubectl get pod ngnix-resolver -o wide
      nginx-resolver   10.244.1.55
kubectl run test-nslookup --image=busybox:1.28 -it --restart=Never --rm -- nslookup  10-244-1-55.default.pod.cluster.local > /var/

$ cat /var/CKA2022/nginx.svc
$ cat /var/CKA2022/nginx.pod
```

❓17. Application with PVC

- Cluster: kubectl config use-context **k8s**

Create a new PersistentVolumeClaim:

Name: pv-volume

Class: csi-hostpath-sc

Capacity: 10Mi

Create a new Pod which mounts the PersistentVolumeClaim as a volume:

Name: web-server

Image: nginx

Mount path: /usr/share/nginx/html

Configure the new Pod to have ReadWriteOnce access on the volume.

▼ ❗ Answer

```
vi pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
  storageClassName: csi-hostpath-sc

kubectl apply -f pvc.yaml
kubectl get pv,pvc


vi pod-with-pvc.yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
    - name: web-server
      image: nginx
```

```
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: pv-volume
  volumes:
    - name: pv-volume
      persistentVolumeClaim:
        claimName: pv-volume

kubectl apply -f pod-with-pvc
kubectl get pod
```