



실전 모의 평가(2h)

☰ 태그

CKA 노트

실전 문제 풀이

? 1. Retrieve Error Messages from a Container Log

- Cluster: `kubectl config use-context hk8s`

In the `customera` namespace, check the log for the `app` container in the `custom-app` Pod.

Save the lines which contain the text `"error"` to the file `/var/CKA2022/errors.txt`.

? 2. Node Troubleshooting

- Cluster: `kubectl config use-context hk8s`

A Kubernetes worker node, named `hk8s-worker2` 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`.

? 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`.

? 4. Management Node

- Cluster: `kubectl config use-context hk8s`

Set the node named `hk8s-worker1` as **unavailable** and **reschedule all the pods running** on it.

? 5. Helm을 이용한 패키지 배포

- Cluster: `kubectl config use context k8s`

- Helm을 이용해 `nginx` 웹 서버를 배포하시오.

- Task:**

- helm repository : <https://charts.bitnami.com/bitnami>
- repo name : `bitnami`
- install chart: `bitnami/nginx`
- chart name : `cka-webserver`
- 서비스 동작 중인지 확인을 위해 `k8s-worker1` 노드로 서비스 연결해본다.

? 6. Cluster Upgrade - only Master

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

upgrade system : **k8s-master**

Given an existing Kubernetes cluster running version **1.30.0** ,

upgrade all of the Kubernetes control plane and **node components** on the master node only to version **1.30.6** .

Be sure to **drain** the **master** node before upgrading it and **uncordon** it after the upgrade.

? 7. Authentication and Authorization

- cluster : `kubectl config use-context k8s`
-
- 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** .
-

? 8. Pod 생성하기

- cluster : `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"`
- resource request
 - cpu: `100m`
 - memory : `100Mi`

? 9. multi-container Pod 생성

- cluster : `kubectl config use-context k8s`
-
- Create a pod with 3 containers running : `nginx`, `redis`, `memcached` and `consul`
 - pod name: `eshop-frontend`
 - image: `nginx`
 - image: `redis`
 - image: `memcached`

? 10. Side-car Container Pod 실행

- 작업 클러스터 : `kubectl config use-context k8s`
- An existing Pod needs to be integrated into the `Kubernetes built-in logging architecture` (e.g. `kubectl logs`).
- Adding a streaming sidecar container is a good and common way to accomplish this requirement.

- Task:
 - 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 사이드카 컨테이너를 기존 `eshop-cart-app` 에 추가합니다.
 - 새 `price` 컨테이너는 다음과 같은 command를 실행해야 합니다.


```
Command: /bin/sh, -c, "tail -n+1 -f /var/log/cart-app.log"
```
 - `/var/log` 에 마운트 된 볼륨을 사용하여 사이드카 컨테이너에서 로그 파일 `cart-app.log` 를 사용해야 합니다.
 - `eshop-cart-app` Pod와 `cart-app` 컨테이너를 수정하지 마시오.
- ▶ Hint : 문제 풀이 순서

? 11. Pod Scale-out

- Cluster: `kubectl config use-context k8s`

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

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

? 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`

? 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`

? 14. Create a Storage Class

- Cluster: `kubectl config use-context k8s`
- Create a new Storage Class called `delayed-volume-sc` that makes use of the below specs:
 - provisioner: `kubernetes.io/no-provisioner`
 - volumeBindingMode: `WaitForFirstConsumer`

? 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

? 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

? 17. Application with PersistentVolumeClaim

- Cluster: `kubectl config use-context k8s`
- Create a new **PersistentVolumeClaim**: Name: **pv-volume** Class: **app-hostpath-sc** Capacity: **10Mi**
Configure the new Pod to have **ReadWriteMany** access on the volume.
- Create a new **Pod** which mounts the PersistentVolumeClaim as a volume:
 - Name: **web-server-pod**
 - Image: **nginx**
 - Mount path: **/usr/share/nginx/html**
- Finally, using `kubectl edit` or `kubectl patch` expand the PersistentVolumeClaim to a capacity of **70Mi** and **record that change**.

? 18. Ingress 구성

- 작업 클러스터 : `kubectl config use-context k8s`
 - **app-ingress.yaml** 파일을 생성하여 다음 조건의 ingress 서비스를 구성하시오.
 - ingress name: **app-ingress**
 - **NODE_PORT:30080/** 접속했을 때 **nginx** 서비스로 연결
 - **NODE_PORT:30080/app** 접속했을 때 **appjs-service** 서비스로 연결
 - 다음의 manifests 참고합니다.