cka考题

### 0、前言

考试注意事项：

提前预约，全程线上考试，所以需要准备高清摄像头，考前会给你随机安排官方考官来检查你的周围环境，包括桌面不能有任何杂物、书籍、电子产品等，房间里不能有其他人，还会当场实时拍摄上传你的身份证和你的上半身照片，所以需要你的摄像头足够高清。考试时间为2小时，考前检查完后才会把考试系统调出来，调出来的那一刻才开始计算考试时间，可以在考试页面实时看到这个时间。

考试是用的他们专有浏览器，实时下载，不建议提前下载，因为版本更新太快。到时候在预约考试的那个页面进入即可下载，安装后就开始根考官对接并检查考试环境。

考前需要检查下自己的电脑版本和配置，过老的电脑配置不能安装这个浏览器，无法考试。

提前准备好网络环境和梯子，因为网络不稳定随时会把你弹出考试系统或者考试页面空白一直转圈，非常耽误时间，若弹出来再进去得重新检查周围环境，这个过程考试时间不会暂停，所以网络环境非常非常重要，很多人做题时间不够都是网络导致的，其实题目很简单。

考试时可以随时查阅官网，很多题目需要编写yaml，不要手敲，容易错，而且非常节约时间。

考试共17道题，总分100，66分及格，掌握以下题型拿95分以上是稳稳的。

证书说明不了什么，务实求实方显男儿本色，最后祝愿大家能够顺利通过cka认证，早日拿到满意的offer。

### 1、RBAC 授权

#### 题目：

切换集群：kubectl config use-context k8s

Context：为部署管道创建一个新的 clusterRole 并将其绑定到范国为特定的namespace 的特定 ServiceAccount 。

Task：

创建一个名为 deployment-clusterrole 且仅允许创建以下资源类型的新 ClusterRole :

Deployment

StatefulSet

DaemonSet

在现有的namespace app-teaml中创建一个名为 cicd-token 的新 ServiceAccount。

限于namespace app-teaml，将新的ClusterRole deployment-clusterrole 绑定到新的 ServiceAccount cicd-token

#### 做题分析：

切换集群。

创建一个集群角色ClusterRole，名为deployment-clusterrole，具备创建Deployment、StatefulSet、DaemonSet资源功能闹。

创建一个服务账号ServiceAccount，名为cicd-token，在app-teaml命名空间下。

将新创建的服务账号cicd-token与集群角色deployment-clusterrole绑定到一起，在app-teaml命名空间下。

最后测试，只能创建deploy等资源权限，查看等功能用不了。

1、切换指定集群。

[root@k8s-master CKA]# kubectl config use-context 【集群名称】

2、创建集群角色，名为deployment-clusterrole，该角色具备创建权限，资源对象为deployments、daemonsets、statefulsets。

[root@k8s-master CKA]# kubectl create clusterrole deployment-clusterrole --verb=create --resource=deployment,daemonsets,statefulsets

3、在app-teaml命名空间里创建一个服务账号，名为cicd-token。

[root@k8s-master CKA]# kubectl create ns app-teaml

[root@k8s-master CKA]# kubectl create serviceaccount cicd-token -n app-teaml

4、账号与角色绑定，名为text，将app-teaml命名空间下的cicd-token账号和deployment-clusterrole集群角色绑定。

[root@k8s-master CKA]# kubectl create rolebinding 如果未指定名字使用cicd-token --serviceaccount=app-teaml:cicd-token --clusterrole=deployment-clusterrole -n app-teaml

5、测试只能创建deploymen等资源，查看会失败。

[root@k8s-master CKA]# kubectl --as=system:serviceaccount:app-teaml:cicd-token get pods -n app-teaml

Error from server (Forbidden): pods is forbidden: User "system:serviceaccount:app-teaml:cicd-token" cannot list resource "pods" in API group "" in the namespace "app-teaml"

[root@k8s-master CKA]# kubectl --as=system:serviceaccount:app-teaml:cicd-token create deployment nginx --image=nginx -n app-teaml

### 节点设置不可用

#### 题目：

切换集群：kubectl config use-context ek8s

Task：

将名为 ek8s-node-1 的 node 设置为不可用，并重新调度该 node上所有运行的 pods。

#### 做题分析：

切换集群。

驱逐名为ek8s-node1-1节点，并设置不可调度。

注意事项：

如果执行 drain 提示错误，根据提示再加上选项，例如–delete-local-data --force

1、切换指定集群。

[root@k8s-master CKA]# kubectl config use-context 【集群名称】

2、驱逐。

[root@k8s-master CKA]# kubectl drain 【节点名称】 --ignore-daemonsets

### 3、升级 K8s 版本（注意）

切换集群：kubect1 config use-context mk8s

Task：

现有的Kubernetes集群正在运行版本1.27.4。仅将主节点上的所有 Kubernetes控制平面和节点给件升级到版本 1.27.5

确保在升级之前 drain主节点，并在升级后uncordon 主节点。

另外，在主节点上升级 kubelet 和 kubectl。

可使用以下命令通过 ssh 连接到主节点:

[student@node-1]$ssh mk8s-master-0

可使用以下命令在该主节点上获取更高权限：

[student@mk8s-master-0]$ sudo -i

做题分析：

切换集群。

驱逐mk8s-master-0节点上的pod，并设置不可调度。

ssh连接到mk8s-master-0，切换成root用户。

查看kubeadm当前版本，并安装1.27.5版本。

查看是否可以升级，并开始升级，并设置etcd不升级。

升级 kubelet 和 kubectl到1.27.5版本。

重启kubelet服务。

设置mk8s-master-0节点可调度。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context mk8s

2、驱逐升级节点，并设置成不可调度。

[root@k8s-master CKA]# kubectl drain k8s-master0 --ignore-daemonsets

3、ssh连接到mk8s-master-0升级节点，并切换到root用户。

[root@k8s-master CKA]# ssh mk8s-master-0

[root@k8s-master CKA]# sudo -i

4、查看kubeadm当前版本。

[root@k8s-master CKA]# apt list --all-versions|grep kubeadm

5、安装kubeadm要升级的版本。

[root@k8s-master CKA]# apt install kubeadm=1.27.5-00

6、查看kubeadm是否可以升级。

[root@k8s-master CKA]# kubeadm upgrade plan

7、升级kubeadm，并设置etcd不升级。

[root@k8s-master CKA]# kubeadm upgrade apply v1.27.5 --etcd-upgrade=false

8、升级kubelet、kubectl版本。

[root@k8s-master CKA]# apt install kubelet=1.27.5-00 kubectl=1.27.5-00 -y

9、重启kubelet服务。

[root@k8s-master CKA]# systemctl restart kubelet

10、设置mk8s-master-0节点为可调度。

[root@k8s-master CKA]# kubectl uncordon mk8s-master-0

11、查看版本。

[root@k8s-master CKA]# kubectl get node

### etcd 备份与恢复

退出mk8s-master-0节点：exit

Task

首先，为运行在 https://127.0.0.1:2379 上的现有 etcd 实例创建快照并将快照保存到 /data/backup/etcd-snapshot.db。

为给定实例创建快照预计能在几秒钟内完成。如果该操作似乎挂起，则命令可能有问题。用 CTRL+来取消操作，然后重试。

然后还原位于/data/backup/etcd-snapshot-previous.db 的现有先前快照。

提供了以下TLS证书和密钥，以通过etcdctl 连接到服务器。

CA证书: /opt/KUIN00601/ca.crt

客户端证书: /opt/KUIN00601/etcd-client.crt

客户端密钥:/opt/KUIN00601/etcd-client.crt.key

做题分析：

先退出当前所在集群，返回初始点。

备份。

在初始点备份。

还原。

停止etcd服务。

sytemctl cat etcd 查看数据目录是否为/var/lib/etcd。

将原来的数据目录备份后，使用已存在的数据文件/data/backup/etcd-snapshot-previous.db 还原。

修改新/var/lib/etcd目录属主属组。

启动etcd服务。

1、退出当前节点，返回初始点。

[root@k8s-master CKA]# exit

2、备份操作。

##指定etcdctl版本为3，备份文件名为/data/backup/etcd-snapshot.db ，备份主机为https://127.0.0.1:2379 ，根证书/opt/KUIN00601/ca.crt ，

客户端证书/opt/KUIN00601/etcd-client.crt ，客户端密钥/opt/KUIN00601/etcd-client.key。

[root@k8s-master CKA]# ETCDCTL\_API=3 etcdctl snapshot save /data/backup/etcd-snapshot.db

--endpoints=https://127.0.0.1:2379

--cacert=/opt/KUIN00601/ca.crt

--cert=/opt/KUIN00601/etcd-client.crt

--key=/opt/KUIN00601/etcd-client.key

--endpoints=https://127.0.0.1:2379

实验--cert="/etc/kubernetes/pki/etcd/server.crt" --key="/etc/kubernetes/pki/etcd/server.key" --cacert="/etc/kubernetes/pki/etcd/ca.crt"

3、还原操作。

##先停止ectd服务。

[root@k8s-master CKA]# systemctl stop etcd

##确认下数据目录（--data-dir 值）

[root@k8s-master CKA]# systemctl cat etcd

做实验：mv etc/kubernetes/manifests/ /etc/kubernetes/manifests.bak = systemctl stop etcd

做之前ll查看一下目录的属主属组记下来，稍后恢复的时候会用上

##备份原来的数据目录。

[root@k8s-master CKA]# mv /var/lib/etcd /var/lib/etcd.bak

##使用已存在的数据文件开始还原。

[root@k8s-master CKA]# ETCDCTL\_API=3 etcdctl snapshot restore /data/backup/etcd-snapshot-previous.db --data-dir=/var/lib/etcd

##修改新生成的数据目录。

[root@k8s-master CKA]# chown -R etcd:etcd /var/lib/etcd

##启动etcd服务。

[root@k8s-master CKA]# systemctl start etcd

实验：mv etc/kubernetes/manifests.bak /etc/kubernetes/manifests

### 5、网络策略（查看官网：NetworkPolicy）

切换集群：kubectl config use-contexthk8s

Task：

在现有的namespace my-app中创建一个名为allow-port-from-namespace 的新 NetworkPolicy 。

确保新的NetworkPolicy 允许namespace my-app 中的Pods来连接到namespace big-corp 中的端口 8080。

进一步确保新的NetworkPolicy ：

不允许对没有在监听端口8080的Pods的访问。

不允许不来自namespace my-app 中的Pods的访问。

做题分析：

切换集群。

给big-corp命名空间打标签，因为在yaml文件中要引用这个命名空间选哟标签。

在my-app命名空间中创建一个网络策略。

名称：allow-port-from-namespace。

所在命名空间：my-app

只允许标签为name=big-corp的命名空间可以访问my-app命名空间下的8080端口。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context hk8s

2、给big-corp命名空间打标签。

[root@k8s-master CKA]# kubectl label namespace big-corp name=big-corp

3、生成一个NetworkPolicy。

[root@k8s-master CKA]# cat NetworkPolicy.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: allow-port-from-namespace

namespace: my-app

spec:

podSelector: {}

policyTypes:

- Ingress

ingress:

- from:

- namespaceSelector:

matchLabels:

name: big-corp

ports:

- protocol: TCP

port: 8080

[root@k8s-master CKA]# kubectl apply -f NetworkPolicy.yaml

### 6、SVC 暴露应用

切换集群：kubectl config use-contexty-k8s

Task：

请重新配置现有的deployment front-end 以及添加名为 http 的端口规范来公开现有容器 nginx 的端口80/tcp。

创建一个名为front-end-svc 的新服务，以公开容器端口 http。

配置此服务，以通过在排定的节点上的 NodePort 来公开各个 Pods

做题分析：

切换集群。

使用kubectl edit 命令修改deployment配置文件，添加暴露端口。

创建一个服务，名为front-end-svc。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、修改deployment配置文件，添加暴露端口。

[root@k8s-master CKA]# kubectl edit deployments front-end

......

spec:

containers:

- image: nginx

imagePullPolicy: Always

name: nginx

ports: ##添加此行

- containerPort: 80 ##添加此行

name: http ##添加此行

protocol: TCP ##添加此行

3、创建service服务。

[root@k8s-master CKA]# kubectl expose deployment text --port=80 --target-port=80 --type=NodePort --name=front-end-svc

### 7、Ingress（查看官网：ingress）

切换集群：kubectl config use-context k8s

Task：

如下创建一个新的nginx lngress 资源：

名称：pong

Namespace：ing-internal

使用服务端口 5678 在路径 /hello 上公开service hello

可以使用以下命令检查服务 hello 的可用性，该命令应返回 hello :

[student@node-1]$curI -kL <INTERNAL\_IP>/he110

1

做题分析：

切换集群。

查看ingressclasses是否是nginx。

创建一个ingress。

验证查看。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、查看ingress嘞是否是nginx。

[root@k8s-master CKA]# kubectl get ingressclasses

3、生成一个yaml文件，创建ingress。

[root@k8s-master CKA]# cat Ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: pong ##修改

namespace: ing-internal #3修改

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

重定向，就是这一句 ，现在访问这个域名，会重定向到我的/路径

spec:

ingressClassName: nginx ##查看修改

rules:

- http:

paths:

- path: /hello ##修改

pathType: Prefix

backend:

service:

name: hello ##修改

port:

number: 5678 ##修改

[root@k8s-master CKA]# kubectl apply -f Ingress.yaml

4、验证查看

[root@k8s-master CKA]# kubectl get ingress -n ing-internal

curl -kL <获取 ingress 的 IP 地址>/hello

### 8、扩容 Pod 数量

切换集群：kubectl cor fig use-contextk8s

Task：

将deployment从 loadbalancer 扩展至 5 pods

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、扩容。

[root@k8s-master CKA]# kubectl scale deployment loadbalancer --replicas=5

### 9、nodeSelector（查看官网：pod）

切换集群：kubectl config use-contextk8s

Task：

按如下要求调度一个 cod:

名称: nginx-kusc00401

lmage: nginx

Node selector: disk=ssd

做题分析：

切换集群。

创建一个pod，创建yaml，添加节点标签。

查看pod被分配到哪个节点上。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、编辑Yaml文件，创建pod。

[root@k8s-master CKA]# cat Pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx-kusc00401 ##修改

spec:

containers:

- name: nginx ##修改

image: nginx ##修改

nodeSelector: ##添加

disk: ssd ##添加

3、查看pod被分配到哪个节点上。

[root@k8s-master CKA]# kubectl get po nginx-kusc00401 -o wide

### 10、统计准备就绪节点数量

切换集群：kubectl config use-contextk8s

Task：

检查有多少worker nodes 已准备就绪(不包括被打上 Taint:NoSchedule 的节点)，并将数量写入/opt/KUSC00402/kusc00402.txt

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、查看数量并写入文件。

[root@k8s-master CKA]# kubectl describe node |grep Taint

[root@k8s-master CKA]# echo '1' > /opt/KUSC00402/kusc00402.txt

### 11、Pod 配置多容器（查看官网：pod）

切换集群：kubectl config use-context k8s

Task：

创建一个名为 kucc4的pod，在pod里面分别为以下每个images单独运行一个app container(可能会有1-4个images):nginx + redis + memcached

1、切换容器。

[root@k8s-master CKA]# kubectl config use-context k8s

2、编写yaml，创建pod。

[root@k8s-master CKA]# cat Pod2.yaml

apiVersion: v1

kind: Pod

metadata:

name: kucc4 ##修改

spec:

containers:

- name: nginx ##添加

image: nginx ##添加

- name: redis ##添加

image: redis ##添加

- name: memcached ##添加

image: memcached ##添加

### 12、创建 PV（查看官网：pv、pvc、pod）

切换容器：kubectl cunfig use-contexthk8s

Task：

创建名为 app-data 的 persistentvolume，容量为 2Gi，访问模式为 ReadWriteOnce。 volume类型为 hostPath，位于/srv/app-data

做题分析：

切换集群。

创建一个pv，添加主机共享目录。意思是当所创建的Pod生成数据时，先是进入容器，再传输到该容器所在节点上的共享目录。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context hk8s

2、创建pv，编写yaml文件。

[root@k8s-master CKA]# cat PV.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: app-data ##修改

spec:

capacity:

storage: 2Gi ##修改

accessModes:

- ReadWriteOnce

hostPath: ##添加

path: "/srv/app-data" ##添加

[root@k8s-master CKA]# kubectl apply -f PV.yaml

3、验证，创建pvc。

[root@k8s-master CKA]# cat Pod-PVC.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: myclaim

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 2Gi ##修改

---

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: myfrontend

image: nginx

volumeMounts:

- mountPath: "/var/www/html"

name: mypd

volumes:

- name: mypd

persistentVolumeClaim:

claimName: myclaim

[root@k8s-master CKA]# kubectl apply -f Pod-PVC.yaml

4、进入新创建的容器查看，生成一个文件，再去该容器所在的节点的/srv/app-data目录下查看就会有刚创建的文件。

[root@k8s-master CKA]# kubectl exec -it mypod -- bash

root@mypod:/# cd /var/www/html/

root@mypod:/var/www/html# touch 111.txt

5、容器节点上查看生成的文件。

[root@k8s-node2 ~]# cd /srv/app-data/

[root@k8s-node2 app-data]# ll

-rw-r--r-- 1 root root 0 Mar 29 13:19 111.txt

### 13、Pod 使用 PVC（查看官网：pvc）

切换集群：kubectl config use-context ok8s

Task:

创建一个新的 PersistentVolumeClaim ;

名称：pv-volume

Class：csi-hostpath-sc

容量：10Mi

创建一个新的Pod，此Pod将作为volume挂载到PersistentVolumeClaim :

名称: web-server

lmage: nginx

挂载路径: /usr/share/nginx/html

配置新的Pod，以对volume具有 ReadWriteOnce 权限。

最后，使用 kubectl edit 或 kubectl patch 将PersistentVolumeClaim 的容量扩展为 70Mi，并记录此更改。

做题分析：

切换集群。

创建一个pvc和pod，指定存储类。

使用kubectl edit pvc pv-volume扩容，添加参数–save-config代表记录此更改。

1、切换集群。

kubectl config use-context ok8s

2、创建pvc和pod，并指定存储类。

[root@k8s-master CKA]# cat pod-pvc2.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pv-volume ##修改

spec:

storageClassName: csi-hostpath-sc ##修改

accessModes:

- ReadWriteOnce ##保留

resources:

requests:

storage: 10Gi ##修改

---

apiVersion: v1

kind: Pod

metadata:

name: web-server ##修改

spec:

containers:

- name: web-server ##修改

image: nginx ##修改

ports:

volumeMounts:

- mountPath: "/usr/share/nginx/html" ##修改

name: mypd

volumes:

- name: mypd

persistentVolumeClaim:

claimName: pv-volume ##修改

3、修改容量为70Mi。

[root@k8s-master CKA]# kubectl edit pvc pv-volume --save-config

### 14、获取 Pod 错误日志

切换集群：kubectl config use-contextk8s

Task：

监控 pod bar 的日志并提取与错误 file-not-found 相对应的日志行，将这些日志行写入 /opt/KUTR00101/bar

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、查看bar日志，过滤错误关键词，并记录到文件里。

[root@k8s-master CKA]# kubectl logs bar | grep file-not-found > /opt/KUTR00101/bar

### 15、给 Pod 增加一个容器（边车）

切换集群：kubectl config use-context k8s

Context:

将一个现有的Pod集成到Kubernetes的内置日志记录体系结构中(例如 kubectl ogs)。添加 treaningsidecar 容器是实现此要求的一种好方法

Task:

使用busybox Image来将名为 sidecar的sidecar 容器添加到现有的Pod iegacy-app中。

新的sidecar容器必须运行以下命令：

/bin/sh -c tail -n+1 -f /var/log/legacy-app.log

1

使用安装在/var/log的Volume，使日志文件legacy-app.log 可用于sidecar容器。

除了添加所需的volume mount以外，请勿更改现有容器的规格。

做题分析：

切换集群。

导出iegacy-app pod的yaml文件，并删除此pod。

修改导出来的yaml文件，添加内容。

导入yaml。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、导出现有的iegacy-app的 pod，再删除iegacy-app。

[root@k8s-master CKA]# kubectl get pod legacy-app -o yaml > legacy-app.yaml

3、修改legacy-app.yaml文件。

[root@k8s-master CKA]# vim legacy-app.yaml

apiVersion: v1

kind: Pod

metadata:

name: legacy-app

spec:

containers:

- name: count

image: busybox

args:

- /bin/sh

- -c

- >

i=0;

while true;

do

echo "$i: $(date)" >> /var/log/legacy-app.log;

sleep 1;

done

volumeMounts: ##添加

- name: varlog ##添加

mountPath: /var/log ##添加

- name: sidecar ##添加

image: busybox ##添加

args: [/bin/sh, -c, 'tail -n+1 -f /var/log/legacy-app.log'] ##添加

volumeMounts: ##添加

- name: varlog ##添加

mountPath: /var/log ##添加

......

volumes: # volumes 块在导出的 yaml 下面已经有了，在已有的添加下面两行即可

- name: varlog ##添加

emptyDir: {} ##添加

4、导入修改完的yaml。

[root@k8s-master CKA]# kubectl apply -f legacy-app.yaml

### 16、统计使用 CPU 最高的 Pod

Task:

通过 pod label name=cpu-utilizer，找到运行时占用大量CPU的pod，并将占用CPU最高的pod名称写入文件/opt/KUTR00401/KUTR00401.txt (已存在)。

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context k8s

2、-A查看所有命名空间的pod，-l指定pod标签，--sort-by指定查看什么，这里就是查看cpu。

[root@k8s-master CKA]# kubectl top pod -l name=cpu-utilizer --sort-by="cpu" -A

3、把查看出来的第一个 Pod 名称写到文件

[root@k8s-master CKA]# echo "<podname>" > /opt/KUTR00401/KUR00401.txt

### 17、节点 NotReady 处理

Task:

名为 wk8s-node-0 的Kubernetes worker node处于NotReady 状态。

调查发生这种情况的原因，并采取相应措施将node恢复为 Ready 状态，确保所做的任何更改永久有效。

可使用以下命令通过 ssh 连接到故障node:

ssh wk8s-node-0[student@node-1]$

可使用以下命令在该 node 上获取更高权限:

[studentaw8ks-node-]$ sudo -i

1、切换集群。

[root@k8s-master CKA]# kubectl config use-context wk8s

2、查看所有节点状态。

[root@k8s-master CKA]# kubectl get node

3、找到有问题的节点，ssh过去，并切换root用户。

[root@k8s-master CKA]# ssh wk8s-node-0

[root@k8s-master CKA]# sudo -i

4、查看kubelet服务状态，启动，并设置开机自启。

[root@k8s-master CKA]# systemctl status kubelet

[root@k8s-master CKA]# systemctl start kubelet

[root@k8s-master CKA]# systemctl enable kubelet