创想云教育2021年最新CKA题库1.20版本

2021 最新考试集群为 1.20 版本，只有第三题和第十题发生了变化

第一题：RBAC

Set configuration context:

```
[student@node-1] $ kubectl config use-context 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.

kubectl create clusterrole deployment-clusterrole --verb=create --resource=deployments,statefulsets,daemonsets

kubectl create namespace app-team1

kubectl -n app-team1 create serviceaccount cicd-token

kubectl -n app-team1 create rolebinding cicd-token-binding --clusterrole=deployment-clusterrole --serviceaccount=app-team1:cicd-token

kubectl -n app-team1 describe rolebindings.rbac.authorization.k8s.io cicd-token-binding

第二题：指定 node 设置为不可用

将名为 ek8s-node-1 的 node 设置为不可用，并且重新调度该 node 上所有允许的 pods

$ kubectl cordon ek8s-node-1

$ kubectl drain ek8s-node-1 --delete-local-data --ignore-daemonsets --force

第三题：升级 kubernetes 节点

ps: 1.20 版本要求从 1.20.0 升级到 1.20.1

Set configuration context:

```
[student@node-1] $ kubectl config use-context mk8s
```

Task

Given an existing Kubernetes cluster running version 1.18.8 , upgrade all of the Kubernetes control plane and node components **on the master node only** to version 1.19.0 .

You are also expected to upgrade kubelet and kubectl on the master node.

Be sure to drain the master node before upgrading it and uncordon it after the upgrade.
Do not upgrade the worker nodes, etcd, the container manager, the CNI plugin, the DNS service or any other addons.

现有的 Kubernetes 集权正在运行的版本是 1.18.8，仅将主节点上的所有 kubernetes 控制面板和组件升级到版本 1.19.0
另外，在主节点上升级 kubelet 和 kubectl

```
$ kubectl config use-context mk8s
$ kubectl get node
$ kubectl cordon mk8s-master-1
$ kubectl drain mk8s-master-1 --delete-local-data --ignore-daemonsets --force
$ ssh mk8s-master-1
$ sudo -i
# apt-get install -y kubeadm=1.20.1-00
# kubeadm version
# kubeadm upgrade plan
# kubeadm upgrade apply v1.20.1 --etcd-upgrade=false
```

# apt-get install kubelet=1.20.0-00 kubectl=1.20.1-00
# kubelet version
# kubelet version
# systemctl status kubelete
# systemctl daemon-reload
# exit
$ exit

$ kubectl get node （确认只升级了 master 节点到 1.20.1 版本）

第四题：etcd 备份还原(1.20 版本需要把端口号从 2739 改成 2830)

如果环境中没有 etcdctl 这个命令，需要先执行如下指令：
$ sudo apt install etcd-client

No configuration context change required for this item.

Task

First, create a snapshot of the existing etcd instance running at https://127.0.0.1:2379 , saving the snapshot to /srv/data/etcd-snapshot.db .

Creating a snapshot of the given instance is expected to complete in seconds.
If the operation seems to hang, something's likely wrong with your command. Use CTRL + C to cancel the operation and try again.

Next, restore an existing, previous snapshot located at /var/lib/backup/etcd-snapshot-previous.db .

The following TLS certificates/key are supplied for connecting to the server with etcdctl :

- CA certificate: /opt/KUIN00601/ca.crt
- Client certificate: /opt/KUIN00601/etcd-client.crt
- Client key: /opt/KUIN00601/etcd-client.key

#备份：要求备份到指定路径及指定文件名
$ ETCDCTL_API=3 etcdctl --endpoints 127.0.0.1:2379 --cacert=/opt/KUIN00601/ca.crt --cert=/opt/KUIN00601/etcd-client.crt --key=/opt/KUIN00601/etcd-client.key snapshot save /srv/data/etcd-snapshot.db
#还原：要求使用指定文件进行还原
$ ETCDCTL_API=3 etcdctl --endpoints 127.0.0.1:2379 --cacert=/opt/KUIN00601/ca.crt --cert=/opt/KUIN00601/etcd-client.crt --key=/opt/KUIN00601/etcd-client.key snapshot restore /var/lib/backup/etcd-snapshot-previous.db

第五题：创建 NetworkPolicy

Set configuration context:

```
[student@node-1] $ kubectl config use-context hk8s
```

Task

Create a new NetworkPolicy named allow-port-from-namespace that allows Pods in the existing namespace internal to connect to port 9000 of other Pods in the same namespace.

Ensure that the new NetworkPolicy :

- does **not** allow access to Pods not listening on port 9000
- does **not** allow access from Pods not in namespace internal

5-1. 同一个 namespace

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: internal
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector: {}
```

```
    ports:
    - protocol: TCP
      port: 9000
```

5-2. 非同一个 namespace

```
$ kubectl describe ns corp-bar

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: internal
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          project: corp-bar
    ports:
    - protocol: TCP
      port: 5679

$ kubectl config use-context hk8s
$ vi netwokpolicy.yaml
#  将上面的 yaml 内容粘贴进来
$ kubectl apply -f netwokpolicy.yaml
```

官方文档直接复制，修改个别参数
https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/#networkpolicy-resource

第六题：创建 svc

Set configuration context:

```
[student@node-1] $ kubectl config use-context k8s
```

Task

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.

$ kubectl config use-context k8s
$ kubectl expose deployment front-end --port=80 --target-port=80 --protocol=TCP --type=NodePort --name=front-end-svc

第七题：创建 ingress 资源

```
[student@node-1] $ kubectl config use-context k8s
```

Task

Create a new nginx Ingress resource as follows:

- Name: pong
- Namespace: ing-internal
- Exposing service hi on path /hi using service port 5678

The availability of service hi can be checked using the following command, which should return hi :

```
[student@node-1] $ curl -kL <INTERNAL_IP>/hi
```

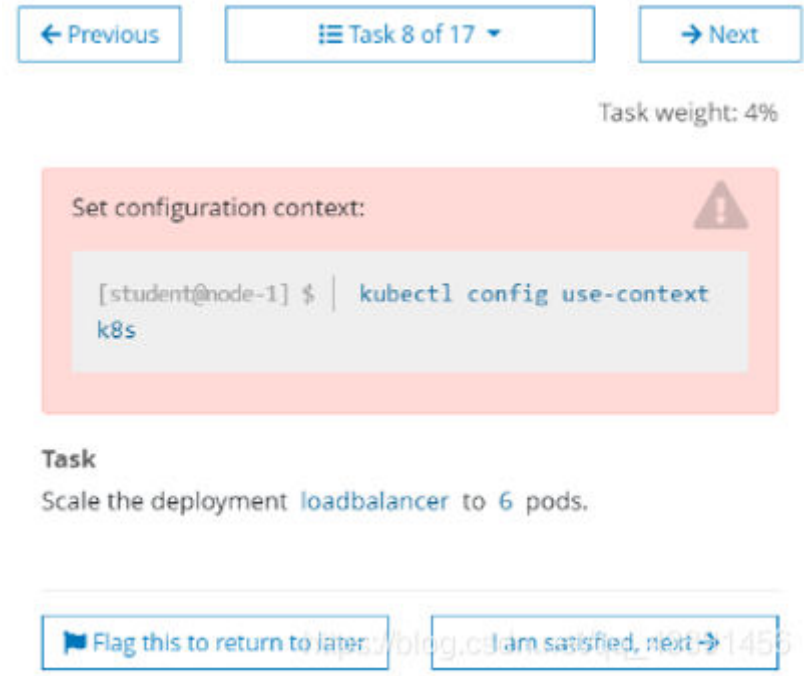


```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ping
  namespace: ing-internal
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /hi
        pathType: Prefix
        backend:
          service:
            name: hi
            port:
              number: 5678
```

```
$ kubectl config use-context k8s
$ vi ping-ingress.yaml
```

```
# 将上面的 yaml 内容粘贴进来
$ kubectl apply -f ping-ingress.yaml
# 验证
$ kubectl get pod -n ing-internal -o wide     # 获取 ingress 的 IP 地址
$ curl -kL $(获取 ingress 的 IP 地址)
# 返回 hi 即为成功

官网复制并修改
https://kubernetes.io/zh/docs/concepts/services-networking/ingress/#the-ingress-resource
```

第八题：扩展 deployment




```
$ kubectl config use-context k8s
$ kubectl scale deployment webserver --replicas=6
```

第九题：将 pod 部署到指定 node 节点上

Task

Schedule a pod as follows:

- Name: nginx-kusc00401
- Image: nginx
- Node selector: disk=spinning

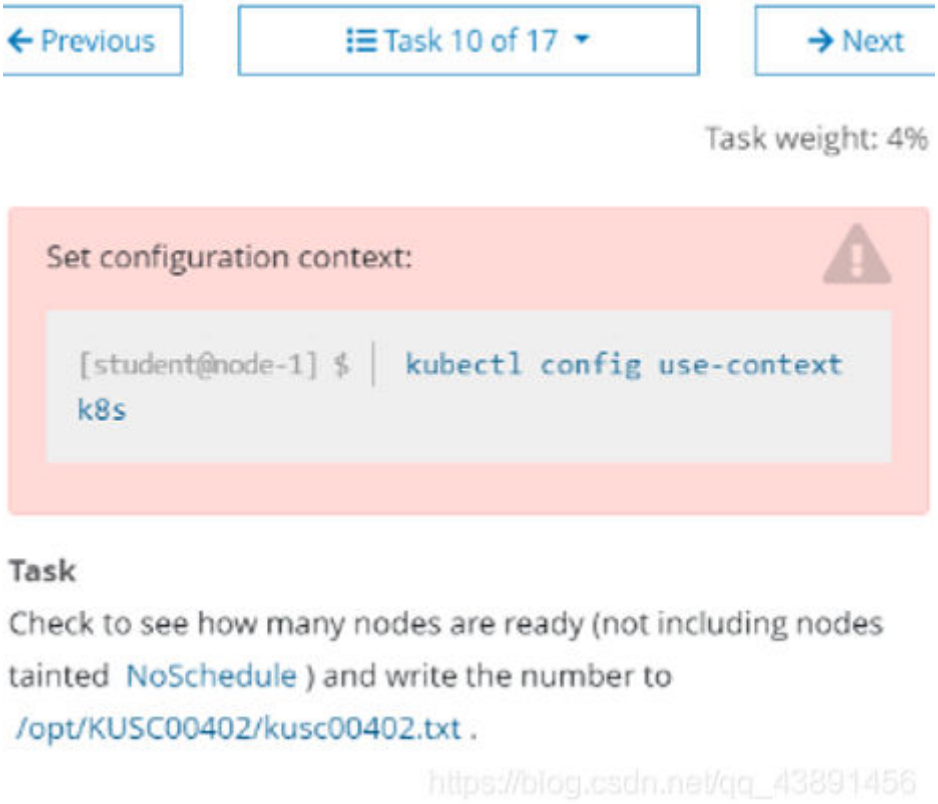
```
apiVersion: v1

kind: Pod

metadata:

  name: nginx-kusc00401

spec:

  containers:

  - name: nginx

    image: nginx

    imagePullPolicy: IfNotPresent

  nodeSelector:
    disk: ssd
```

```
$ kubectl config use-context k8s

$ kubectl run nginx-kusc00401 --image=nginx --dry-run=client -oyaml > pod-nginx.yaml

$ vi pod-nginx.yaml

# 将上面的 yaml 内容粘贴进来

$ kubectl apply -f pod-nginx.yaml

# 验证
```

$ kubectl get po nginx-kusc00401 -o wide

第十题：检查有多少 node 节点是健康状态

PS：请注意本体不在有 NoSchedule 节点，答案变成 3 个




```
kubectl config use-context k8s
kubectl describe node |grep -i taints |grep -v -i noschedule
echo $Num > /opt/KUSC00402/kusc00402.txt
```

第十一题：创建多个 container 的 Pod

```
[student@node-1] $ kubectl config use-context k8s
```

Task

Create a pod named kucc1 with a single app container for each of the following images running inside (there may be between 1 and 4 images specified): nginx + redis + memcached + consul .

Flag this to return to later | I am satisfied, next →


```
apiVersion: v1
kind: Pod
metadata:
  name: kucc4
spec:
  containers:
  - name: nginx
    image: nginx
  - name: redis
    image: redis
  - name: memcached
    image: memcached
  - name: consul
    image: consul

$ kubectl config use-context k8s
$ kubectl run kucc4 --image=nginx --dry-run=client -oyaml > pod-kucc4.yaml
$ vi pod-kucc4.yaml
# 将上面的 yaml 内容粘贴进来
$ kubectl apply -f pod-kucc4.yaml
```

第十二题：创建 Persistent Volume

Set configuration context:

```
[student@node-1] $ kubectl config use-context hk8s
```

Task

Create a persistent volume with name app-config , of capacity 2Gi and access mode ReadWriteMany . The type of volume is hostPath and its location is /srv/app-config .

Flag this to return to later | I am satisfied, next →


```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/srv/app-config

$ kubectl config use-context hk8s
$ vi app-data-pv.yaml
# 将上面的 yaml 内容粘贴进来
$ kubectl apply -f app-data-pv.yaml
# 验证
$ kubectl get pv

官网复制并且修改：
https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/
```

第十三题：创建 PVC

Set configuration context:

```
[student@node-1] $ kubectl config use-context ok8s
```

Task

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.

Finally, using kubectl edit or kubectl patch expand the PersistentVolumeClaim to a capacity of 70Mi and record that change.

Flag this to return to la... | I am satisfied, next →

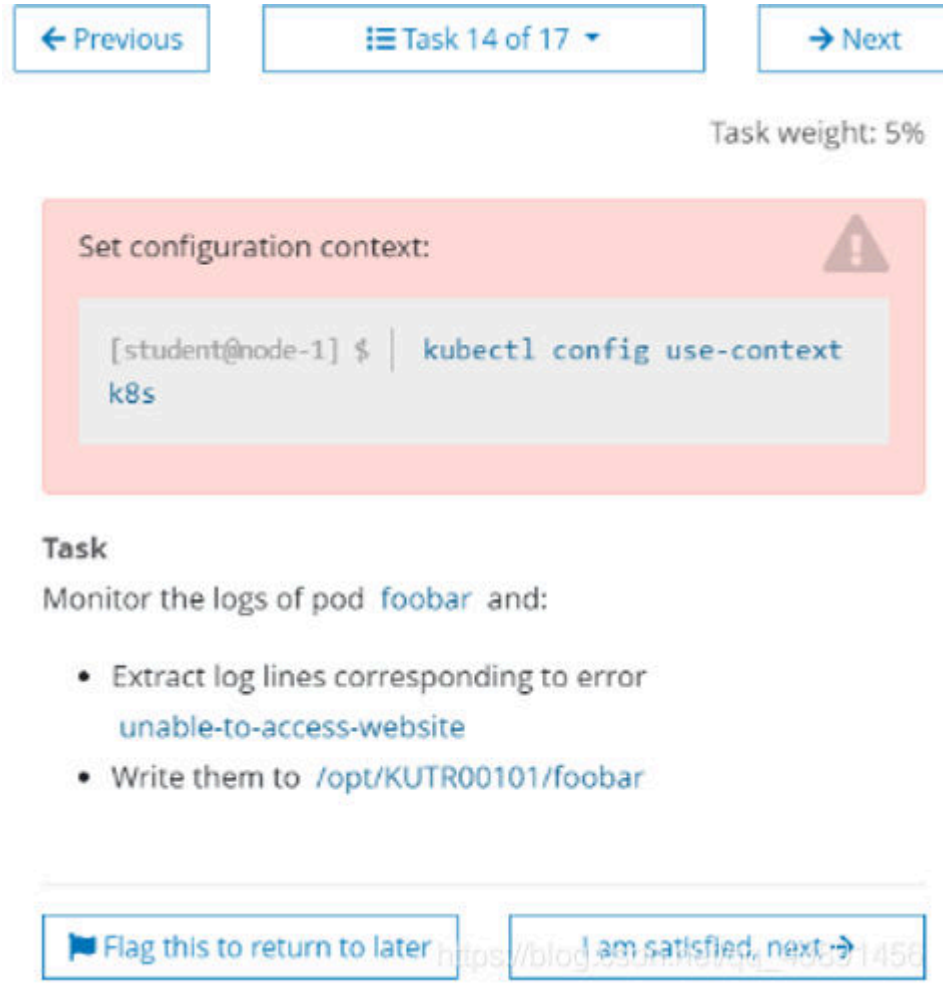

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
spec:
  storageClassName: csi-hostpath-sc
  accessModes:
```

```
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi

---
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: pv-volume
  containers:
    - name: web-server
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage

$ kubectl config use-context ok8s
$ vi pv-volume-pvc.yaml
#  将上面的 yaml 内容粘贴进来
$ kubectl apply -f pv-volume-pvc.yaml
#  验证
$ kubectl get pvc
#  修改 pvc 10Mi --> 70Mi
$ kubectl edit pvc pv-volume --record
```

官网复制并修改：

https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

第十四题：监控 pod 的日志



监控 pod foobar 的日志并提取错误的 unable-access-website 相对于的日志写入到 /opt/KUTR00101/foobar

```
$ kubectl config use-context k8s
$ kubectl logs foobar | grep unable-to-access-website > /opt/KUTR00101/fooba
```

第十五题：添加 sidecar container

```
[student@node-1] $ kubectl config use-context k8s
```

Context

Without changing its existing containers, an existing Pod needs to be integrated into Kubernetes's built-in logging architecture (e.g. `kubectl logs`). Adding a streaming sidecar container is a good and common way to accomplish this requirement.

Task

Add a busybox sidecar container to the existing Pod legacy-app . The new sidecar container has to run the following command:

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

Use a volume mount named logs to make the file /var/log/legacy-app.log available to the sidecar container.

Don't modify the existing container.
Don't modify the path of the log file, both containers must access it at /var/log/legacy-app.log .


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

```
$ kubectl config use-context k8s

$ kubectl get po legacy-app -o yaml > 15.yaml #

$ kubectl delete -f 15.yaml

$ kubectl apply -f 15.yaml
```

官网复制并修改： https://kubernetes.io/zh/docs/concepts/cluster-administration/logging/

第十六题：查看最高 CPU 使用率的 Pod

查看 Pod 标签为 name=cpu-user 的 CPU 使用率并且把 cpu 使用率最高的 pod 名称写入 /opt/KUTR00401/KUTR00401.txt 文件里

```
$ kubectl config use-context k8s
$ kubectl    top    pod -l name=cpu-user -A
      NAMAESPACE NAME              CPU      MEM
      delault       cpu-user-1    45m      6Mi
      delault       cpu-user-2    38m      6Mi
      delault       cpu-user-3    35m      7Mi
      delault       cpu-user-4    32m      10Mi
$ echo 'cpu-user-1' >>/opt/KUTR00401/KUTR00401.txt
```

第十七题：集群故障排查

Set configuration context:

```
[student@node-1] $ kubectl config use-contex
t wk8s
```

Task

A Kubernetes worker node, named wk8s-node-0 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.


```
名为 wk8s-node-0 的节点处于 NotReady 状态，将其恢复成 Ready 状态，并且设置为开机自启

# 连接到 NotReady 节点
$ ssh wk8s-node-0
获取权限
$ sudo -i
# 查看服务是否运行正常
$ systemctl status kubelet
#如果服务非正常运行进行恢复
$ systemctl start kubelet
#设置开机自启
$ systemctl enable kubelet
```