下载方法，关注公众号【高级云计算架构师】
回复 CKA202011
即可免费下载

高级云计算架构师（公众号）

GJYJSJGS（WX)

第一题：

创建 clusterrole,并且对该 clusterrole 只绑定对 Deployment，Daemonset,Statefulset 的创建权限

在指定 namespace 创建一个 serviceaccount，并且将上一步创建 clusterrole 和该 serviceaccount 绑定

```
#解答
#创建对应的 ClusterRole,并绑定对应的权限
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: deployment-clusterrole
rules:
- apiGroups: ["apps"]
resources: ["daemonsets", "deployments", "statefulsets"]
verbs: ["create"]

#创建对应的 serviceaccount
kubectl   -n app-team1 create serviceaccount cicd-token
```

```
#将 serviceaccount 与 ClusterRole 进行绑定
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: read-secrets-global
subjects:
- kind: ServiceAccount
name: cicd-token
namespace: app-team1
roleRef:
kind: ClusterRole
name: deployment-clusterrole
apiGroup: rbac.authorization.k8s.io
```

第二题：

对指定 etcd 集群进行备份和还原,考试时会给定 endpoints，根证书，证书签名，私钥。


```
#解答
首先，为运行在 https://127.0.0.1:2379 上的现有 etcd 实力创建快照并且将快照保存到
/etc/data/etcd-snapshot.db
然后还原与/var/lib/backup/etcd-snapshot-previoys.db 的现有先前快照
提供了以下 TLS 证书和密钥，已通过 etcdctl 连接到服务器
ca 证书：/opt/KUIN000601/ca.crt
客户端证书：/opt/KUIN000601/etcd-client.crt
客户端密钥：/opt/KUIN000601/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="https://127.0.0.1:2379" --
cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --
key=/opt/KUIN000601/etcd-client.key snapshot save /etc/data/etcd-snapshot.db
#还原：要求使用指定文件进行还原
$ ETCDCTL_API=3 etcdctl --endpoints="https://127.0.0.1:2379" --
cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --
key=/opt/KUIN000601/etcd-client.key    snapshot restore /var/lib/backup/etcd-
```

```
snapshot-previoys.db
```

第三题：

升级集群，将集群中 master 所有组件从 v1.18 升级到

1.19(controller,apiserver,scheduler,kubelet,kubectl)

```
#解答
#将节点标记为不可调度状态
$ kubectl cordon k8s-master
#驱逐节点上面的 pod
$ kubectl drain k8s-master--delete-local-data --ignore-daemonsets --force
#升级组件
$ apt-get install kubeadm=1.19.0-00 kubelet=1.19.0-00 kubectl=1.19.0-00
#重启 kubelet 服务
$ systemctl restart kubelet
#升级集群其他组件
$ kubeadm upgrade apply v1.19.0
```

第四题：

创建 Ingress，将指定的 Service 的 9999 端口在/test 路径暴露出来

```
#解答
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
name: pong
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
```

第五题：

创建一个两个 container 容器的 Pod:nginx+redis

第六题：

对集群中的 PV 按照大小顺序排序显示，并将结果写道指定文件

第七题：

将一个 Deployment 的副本数量从 1 个副本扩至3 个

第八题：（与第 20 题类似）

在指定 namespace 创建一个 Networkpolicy，允许 namespace 中的 Pod 访问同 namespace 中其他 Pod 的

8080 端口

第九题：

集群中存在一个 Pod，并且该 Pod 中的容器会将 log 输出到指定文件。 修改 Pod 配置，将 Pod 的日志输出到

控制台,其实就是给 Pod 添加一个 sidecar，然后不断读取指定文件，输出到控制台

```
apiVersion: v1
kind: Pod
metadata:
name: podname
spec:
containers:
  - name: count
    image: busybox
    args:
     - /bin/sh
     - -c
    - >
        i=0;
        while true;
        do
          echo "$(date) INFO $i" >> /var/log/legacy-ap.log;
          i=$((i+1));
          sleep 1;
        done
   volumeMounts:
     - name: logs
       mountPath: /var/log
  - name: count-log-1
    image: busybox
    args: [/bin/sh, -c, 'tail -n+1 -f /var/log/legacy-ap.log']
    volumeMounts:
```

```
    - name: varlog
      mountPath: /var/log
volumes:
  - name: logs
    emptyDir: {}

#验证：
$ kubectl logs <pod_name> -c <container_name>
```

第十题：

查询集群中指定 Pod 的 log日志，将带有 Error 的行输出到指定文件

第十一题：

1.创建一个 Deployment，2.更新镜像版本，3.回滚

第十二题：

集群有一个节点 notready，找出问题，并解决。 并保证机器重启后不会再出现此问题

第十三题(此题与第 18 题类似)

创建一个 PV，使用hostPath 存储，大小1G，ReadWriteOnce

第十四题：

使用指定 storageclass 创建一个 pvc，大小为 10M

将这个 nginx 容器的/var/nginx/html目录使用该 pvc 挂在出来

将这个 pvc 的大小从 10M 更新成 70M

```
#解答
#创建 PVC
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: pv-volume
spec:
accessModes:
  - ReadWriteOnce
volumeMode: Filesystem
resources:
  requests:
    storage: 10Mi
storageClassName: csi-hostpath-sc
#创建 pod
apiVersion: v1
kind: Pod
metadata:
name: web-server
spec:
containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: pv-volume
volumes:
  - name: pv-volume
    persistentVolumeClaim:
      claimName: myclaim
#通过 kubectl edit pvc pv-volume 可以进行修改容量
```

第十五题

将集群中一个 Deployment 服务暴露出来,(是一个 nginx，使用kubectl expose 命令暴露即可)

第十六题

查询集群中节点，找出可以调度节点的数量，（其实就是被标记为不可调度和打了污点的节点之外的节点 数量 ），然后将数量写到指定文件

```
#解答
#  查询集群 Ready 节点数量
$ kubectl get node | grep -i ready
#  判断节点有误不可调度污点
$ kubectl describe nodes <nodeName> |   grep -i taints | grep -i noSchedule
```

第十七题

找集群中带有指定 label 的 Pod 中占用资源最高的，并将它的名字写入指定的文件

```
#解答
$ 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
```

第十八题

创建一个名为 app-config 的 PV，PV 的容量为 2Gi 访问模式为 ReadWriteMany，volume 的类型为 hostPath，位置为/src/app-config

```
# 解答
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
  labels:
    type: local
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/src/app-config"
```

第十九题

将 deployment 扩容刀 6 个 pod

```
#解答

$ kubectl scale --replicas=6 deployment/loadbalancer
```

第二十题

创建 NetworkPolicy

```
##解答

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

   name: all-port-from-namespace

   namespace: internal

spec:

   podSelector:

      matchLabels: {}

   ingress:

   - from:

      - podSelector: {}

      ports:

      - port: 9000
```

11 月份最新考题

11 月份最新考题（题目在上面已经列出），仅供参考

1、Context k8s

为部署管道创建一个新的 ClusterRole 并将其绑定到范围为特定的 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-team1 中创建一个名为 cicd-token 的新 ServiceAccount。限于 namespace app-team1，将新的 ClusterRole deployment-clusterrole 绑定到新的 ServiceAccount cicd-token

考点：RABC 授权模型的理解。

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

kubectl create serviceaccount cicd-token --namespace=default

kubectl create rolebinding deployment-clusterrole --clusterrole=deployment-clusterrole --serviceaccount=default:cicd-token --namespace=default

2、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

考点：cordon 和 drain 命令的使用

$ kubectl cordon ek8s-node-1

$ kubectl drain ek8s-node-1 --force

3、现有的 Kubernetes 集群正在运行版本 1.18.8。仅将主节点上的所有 Kubernetes 控制平面和节点组件升级到版本 1.19.0。

另外，在主节点上升级 kubelet 和 kubectl。

确保在升级之前 drain 主节点，并在升级后 uncordon 主节点。 请不要升级工作节点，etcd，container 管理器，CNI 插件， DNS 服务或任何其他插件。

考点：如何离线主机，并升级控制面板和升级节点

kubectl drain <cp-node-name> --ignore-daemonsets

sudo kubeadm upgrade apply v1.19.0

yum install -y kubelet-1.19.0 kubectl-1.19.0 --disableexcludes=kubernetes

sudo systemctl daemon-reload

sudo systemctl restart kubelet

kubectl uncordon <cp-node-name>

--升级节点

sudo kubeadm upgrade node

yum install -y kubelet-1.19.0 kubectl-1.19.0 --disableexcludes=kubernetes

sudo systemctl daemon-reload

sudo systemctl restart kubelet

4、此项目无需更改配置环境。问题权重: 7%

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 /srv/data/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 + 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.key

考点：etcd 的备份和还原命令

ETCDCTL_API=3 etcdctl --endpoints $ENDPOINT snapshot save/restore snapshotdb --cert=/opt/KUIN00601/etcd-client.crt --key=/opt/KUIN00601/etcd-client.key --cacert=/opt/KUIN00601/ca.crt

5、设置配置环境：问题权重: 7%

kubectl config use-context hk8s

Task

创建一个名为 allow-port-from-namespace 的新 NetworkPolicy，以允许现有 namespace corp-net 中的 Pods 连接到同一 namespace 中其他 Pods 的端口 9200。

确保新的 NetworkPolicy：

不允许对没有在监听端口 9200 的 Pods 的访问

不允许不来自 namespacecorp-net 的 Pods 的访问

考点：NetworkPolicy 的创建

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: all-port-from-namespace

namespace: internal

spec:

podSelector:

matchLabels: {}

ingress:

- from:

- namespaceSelector: matchLabels: name: namespacecorp-net

- podSelector: {}

ports:

- port: 9000

6、设置配置环境：问题权重: 7%

kubectl config use-context k8s

Task 请重新配置现有的部署 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

考点：将现有的 deploy 暴露成 nodeport 的 service。

$ kubectl expose deployment front-end --name=front-end-svc --port=80 --tarport=80 --type=NodePort

7、问题权重: 7%设置配置环境：

kubectl config use-context k8s

Task

如下创建一个新的 nginx Ingress 资源：

名称: ping

Namespace: ing-internal

使用服务端口 5678 在路径 /hello 上公开服务 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：

curl -kL <INTERNAL_IP>/hello

考点：Ingress 的创建

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: /hello

pathType: Prefix

backend:

service:

name: hello

port:

number: 5678

8、设置配置环境：问题权重: 4%

kubectl config use-context k8s

Task

将 deployment 从 presentation 扩展至 6 pods

考点：kubectl scale 命令

$ kubectl scale --replicas=6 deployment/loadbalancer

9、设置配置环境：问题权重: 4%

kubectl config use-context k8s

Task

按如下要求调度一个 pod：

名称：nginx-kusc00401

Image：nginx

Node selector：disk=spinnin

考点：nodeSelect 属性的使用

apiVersion: v1

kind: Pod

metadata:

name: nginx-kusc00401

labels:

role: nginx-kusc00401

spec:

nodeSelector:

disk: spinnin

containers:

- name: nginx

image: nginx

10、设置配置环境：问题权重: 4%

kubectl config use-context k8s

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

考点：检查节点角色标签，状态属性，污点属性的使用

$ kubectl describe nodes <nodeName> | grep -i taints | grep -i noSchedule

11、设置配置环境：问题权重: 4%

kubectl config use-context k8s

Task

创建一个名为 kucc8 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 + consul

考点：pod 概念

apiVersion: v1

kind: Pod

metadata:

name: kucc1

spec:

containers:

- image: nginx

name: nginx

- image: redis

name: redis

- image: memchached

name: memcached

- image: consul

name: consul

12、设置配置环境：问题权重: 4%

kubectl config use-context hk8s

Task

创建名为 app-config 的 persistent volume，容量为 1Gi，访问模式为 ReadWriteMany。 volume 类型为 hostPath，位于 /srv/app-config

考点：hostPath 类型的 pv

apiVersion: v1

kind: PersistentVolume

metadata:

name: app-config

labels:

type: local

spec:

capacity:

storage: 2Gi

accessModes:

- ReadWriteMany

hostPath:

path: "/src/app-config"

13、设置配置环境：问题权重: 7%

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：

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

考点：pvc 的创建 class 属性的使用，--save-config 记录变更

#创建 PVC

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pv-volume

spec:

accessModes:

- ReadWriteOnce

volumeMode: Filesystem

resources:

requests:

storage: 10Mi

storageClassName: csi-hostpath-sc

#创建 pod

apiVersion: v1

kind: Pod

metadata:

name: web-server

spec:

containers:

- name: nginx

image: nginx

volumeMounts:

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

name: pv-volume

volumes:

- name: pv-volume

persistentVolumeClaim:

claimName: myclaim

kubectl edit pvc pv-volume --save-config

14、 问题权重: 7%

名称：web-server

Image：nginx

挂载路径：/usr/share/nginx/html

配置新的 Pod，以对 volume 具有 ReadWriteOnce 权限。

考点：pod 中对 pv 和 pvc 的使用

15、设置配置环境：问题权重: 5%

kubectl config use-context k8s

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

考点：kubectl logs 命令

kubectl logs foobar | grep unable-access-website > /opt/KUTR00101/foobar

16、 设置配置环境：问题权重: 7%

kubectl config use-context k8s

在不更改其现有容器的情况下，需要将一个现有的 Pod 集成到 Kubernetes 的内置日志记录体系结构中（例如 kubectl logs）。添加 streaming sidecar 容器是实现此要求的一种好方法。

Task

将一个 busybox sidecar 容器添加到现有的 Pod legacy-app。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

使用名为 logs 的 volume mount 来让文件 /var/log/legacy-app.log 可用于 sidecar 容器。

不要更改现有容器。 不要修改日志文件的路径，两个容器都必须通过 /var/log/legacy-app.log 来访问该文件。

考点：pod 两个容器共享存储卷

```
apiVersion: v1
kind: Pod
metadata:
name: podname
spec:
containers:
- name: count
image: busybox
args:
- /bin/sh
- -c
- >
i=0;
while true;
do
echo "$(date) INFO $i" >> /var/log/legacy-ap.log;
i=$((i+1));
sleep 1;
done
volumeMounts:
- name: logs
mountPath: /var/log
- name: count-log-1
image: busybox
args: [/bin/sh, -c, 'tail -n+1 -f /var/log/legacy-ap.log']
```

volumeMounts:

- name: varlog

mountPath: /var/log

volumes:

- name: logs

emptyDir: {}

#验证：

$ kubectl logs <pod_name> -c <container_name>

17、设置配置环境，问题权重: 5%

kubectl config use-context k8s

Task

通过 pod label name=cpu-loader，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/KUTR000401/KUTR00401.txt（已存在）。

考点：kubectl top --l 命令的使用

kubectl top pod -l name=cpu-user -A