【目录】

0、CKA考试真题分值及参考地址

序号	题型	分值	k8s官方参考地址
1	RBAC	4	https://kubernetes.io/zh/docs/reference/access-authn-authz/rbac/
2	指定 node 设置为不可用	4	https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#cordon
3	升级 kubernetes 节点	7	https://kubernetes.io/zh/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/
4	etcd 备份还原	7	https://kubernetes.io/zh/docs/tasks/administer-cluster/configure-upgrade-etcd/
5	创建 NetworkPolicy	7	https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/
6	创建 svc	7	https://kubernetes.io/zh/docs/concepts/services-networking/connect-applications-service/
7	创建 ingress	7	https://kubernetes.io/zh/docs/concepts/services-networking/ingress/

8	扩展 deployment副本数	4	https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#scale
9	将 pod 部署到指定 node 节点上	4	https://kubernetes.io/zh/docs/concepts/scheduling-eviction/assign-pod-node/
10	检查有多少 node 节点是健康状态	4	https://kubernetes.io/zh/docs/concepts/scheduling-eviction/taint-and-toleration/
11	创建包含多个 container 的 Pod	4	https://kubernetes.io/zh/docs/concepts/workloads/pods/
12	创建PV（本地卷）	4	https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/
13	创建 PVC并挂载	7	https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/
14	监控 pod 的日志筛选指定错误到文件	5	https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#logs
15	添加 sidecar container	7	https://kubernetes.io/zh/docs/concepts/cluster-administration/logging/
16	查看最高 CPU 使用率的 Pod	5	https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#top
17	集群故障排查，重启kubelet	13	https://kubernetes.io/zh/docs/tasks/configure-pod-container/static-pod/
合计		100	

参考资料：

https://www.cnblogs.com/fengdejiyixx/p/15602074.html#cka%E8%80%83%E8%AF%95%E9%A2%98%E7%9B%AE

1、RBAC（4'）

```
##创建命名空间
kubectl create ns app-team1
##创建clusterrole
kubectl create clusterrole deployment-clusterrole --verb=create --resource=deployments,statefulsets,daemonsets
##创建sa
kubectl create sa cicd-token -n app-team1
##将clusterrole绑定到sa
kubectl create rolebinding deployment-clusterrole-cicd-token-binding --clusterrole=deployment-clusterrole --serviceaccount=cicd-token:app-team1 --namespace=app-team1
##检查绑定是否完成
kubectl create rolebinding deployment-clusterrole-cicd-token-binding --clusterrole=deployment-clusterrole --serviceaccount=cicd-token:app-team1 --namespace=app-team1
```

2、指定 node 设置为不可用（4'）

```
##设置节点不可调度（感觉这步骤没用）
kubectl cordan ek8s-node-1
##设置节点不可调度并驱逐现有节点pod
kubectl drain ek8s-node-1 --delete-emptydir-data --ignore-daemonsets
##检查节点状态
kubectl get nodes
```

3、升级 kubernetes 节点（7'）

注意：从1.22.x升级到1.23.x

```
##查看现有k8s节点版本
kubectl get nodes
##设置节点不可调度（感觉这步骤没用）
kubectl cordan ek8s-node-1
##设置节点不可调度并驱逐现有节点pod
kubectl drain ek8s-node-1 --delete-emptydir-data --ignore-daemonsets

##登录节点
ssh ek8s-node-1
 ##在列表中查找最新的 1.23 版本
 apt update
 apt-cache madison kubeadm
 ##升级kubeadm
 apt-mark unhold kubeadm && \
 apt-get update && apt-get install -y kubeadm=1.23.1-00 && \
 apt-mark hold kubeadm
 ##验证kubeadm版本
 kubeadm version

 ##验证升级计划
 kubeadm upgrade plan
 ##升级master上组件，特别注意这里按题目要求忽略etcd
 kubeadm upgrade apply v1.23.1 --etcd-upgrade=false //注意不升级ETCD

 ##升级kubelet和kubectl
 apt-mark unhold kubelet kubectl && \
 apt-get update && apt-get install -y kubelet=1.23.1-00 kubectl=1.23.1-00 && \
 apt-mark hold kubelet kubectl
 ##重启kubelet
 sudo systemctl daemon-reload
 sudo systemctl restart kubelet

 ##解除节点不可调度
 kubectl uncordon ek8s-node-1
 ##查看升级后k8s节点版本
 kubectl get nodes
```

4、etcd 备份还原（7'）

```
##对ETCD做快照
ETCDCTL_API=3 etcdctl --endpoints=https://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

##恢复ETCD
ETCDCTL_API=3 etcdctl --endpoints=https://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-snapshort-previous.db
```

5、创建 NetworkPolicy（7'）

```
##创建命名空间（考试时不用）
```

```
kubectl create ns fubar
kubectl create ns my-app
##查看命名空间的lables
kubectl get ns --show-labels

#编辑networkpolicy的yaml
vim 05-networkpolicy.yaml

 apiVersion: networking.k8s.io/v1
 kind: NetworkPolicy
 metadata:
  name: allow-port-from-namespace
  namespace: fubar
 spec:
  podSelector:
  matchLabels: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
  - namespaceSelector:
  matchLabels:
  kubernetes.io/metadata.name=my-app
  ports:
  - protocol: TCP
  port: 80

##创建networkpolicy
kubectl apply -n fubar -f 05-networkpolicy.yaml
##检查配置
kubectl describe networkpolicies.networking.k8s.io allow-port-from-namespace -n fubar
```

6、创建 svc（7'）

```
##编辑deployment，添加port
kubectl edit deployment front-end

##添加ports字段
 spec:
 containers:
 - name: my-nginx
 image: nginx
 ports:
  - containerPort: 80
  name: http

##发布NodePort类型Service
kubectl expose deployment front-end --port=80 --target-port=80 --type=NodePort --name=fronnt-end-svc
##检查svc
kubectl describe svc front-end-svc
```

7、创建 ingress（7'）

```
##编辑ingress的yaml
vim 07-ingress.yaml

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

##创建ingress
kubectl apply -f 07-ingress.yaml

##检查ingres状态
kubectl describe ingress ping
```

8、扩展 deployment副本数（4'）

```
##deployment扩展副本
kubectl scale deployment guestbook --replicas=6
##检查deployment
kubectl get deployment
```

9、将 pod 部署到指定 node 节点上（4'）

```
##编辑pod的yml
vim 09-pod.yaml

apiVersion: v1
kind: Pod
metadata:
 name: nginx-kusc00401
spec:
 containers:
  - name: nginx
  image: nginx
  imagePullPolicy: IfNotPresent
  nodeSelector:
  disk: ssd

##创建pod
```

```
kubectl apply -f 09-pod.yaml
##检查pod状态
kubectl get po -o wide
```

10、检查有多少 node 节点是健康状态（4'）

```
##查看node状态，计算ready的node数
kubectl get nodes
##查看有污点的node数
kubectl describe nodes | grep -i taint
##两数相减记入指定文件
echo n >> /opt/KUSC00402/kusc00402.txt
```

11、创建包含多个 container 的 Pod（4'）

```
##编辑pod的yaml
vim 11-pod.yaml

apiVersion: v1
kind: Pod
metadata:
 name: kucc1
spec:
 containers:
  - name: nginx
  image: nginx
  - name: redis
  image: redis

##创建pod
kubectl apply -f 11-pod.yaml
##检查pod状态
kubectl get po -o wide
```

12、创建PV（本地卷）（4'）

```
##编辑PV的yaml
vim 12-pv.yaml

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

##创建PV
kubectl apply -f 12-pv.yaml
```

```
##检查PV
kubectl describe pv app-config
```

13、创建 PVC并挂载（7'）

```
##编辑PVC的yaml
vim 13-pvc.yaml

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

##创建PVC
kubectl apply -f 13-pvc.yaml

##编辑pod的yaml
vim 13-pod.yaml

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

##创建PVC
kubectl apply -f 13-pod.yaml

##修改PVC
kubectl edit pvc pv-volume --record ##此处一定注意别丢了记录

```

14、监控 pod 的日志筛选指定错误到文件（5'）

```
##查看日志并筛选到指定文件
kubectl logs foobar |grep unable-to-access-website > /opt/KUTR00101/foobar
```

15、添加 sidecar container（7'）

```
##保存Pod的yaml到一个文件
kubectl get po 11-factor-app > 15-logsidecar.yaml

##编辑pod
vim 15-logsidecar.yaml

apiVersion: v1
kind: Pod
metadata:
  name: 11-factor-app
 spec:
  containers:
  - name: count
  image: busybox
  args:
  - /bin/sh
  - -c
  - >
  i=0;
  while true;
  do
  echo "$i: $(date)" >> /var/log/11-factor-app.log;
  i=$((i+1));
  sleep 1;
  done
  volumeMounts:
  - name: varlog
  mountPath: /var/log
 ##########add-begin############
  - name: sidecar
  image: busybox
  args: [/bin/sh, -c, 'tail -n+1 -f /var/log/11-factor-app.log']
  volumeMounts:
  - name: varlog
  mountPath: /var/log
 ##########add-end############
  volumes:
  - name: varlog
  emptyDir: {}

 ##删除原有pod。
 ##【注意】此处只能删除就的pod然后新建pod，因为pod中不允许直接编辑删除或者新增容器
 kubectl delete po 11-factor-app
 ##创建新的带有是sidecar的pod
 kubectl apply -f 15-logsidecar.yaml
 ##检查日志可以正常输出
```

```
kubectl logs 11-factor-app -c sidecar
```

16、查看最高 CPU 使用率的 Pod（5'）

```
##查看指定标签pod的资源使用量，并按照cpu排序
kubectl top po -l run=nginx --sort-by cpu
##将cpu最高的pod名写入指定文件
echo xxx >> /opt/KUTR00401/KUTR00401.txt
```

17、集群故障排查，重启kubelet（13'）

```
##查看node状态
kubectl get nodes
##登录有问题的pod
ssh wk8s-node-0
##获取权限
sodu -i
##查看kubelet状态
systemctl status kubelet
##设置开机自启动
 systemctl enable kubelet
 ##重启kubelet
 systemctl restart kubelet
```

【补充】

查看kubelet状态，可以看到kubelet启动时候的参数文件

```
[root@10-10-99-53 ~]# systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; vendor preset: disabled)
  Drop-In: /usr/lib/systemd/system/kubelet.service.d
           └─10-kubeadm.conf
   Active: active (running) since Fri 2022-02-18 15:08:48 CST; 3 weeks 6 days ago
     Docs: https://kubernetes.io/docs/
 Main PID: 23864 (kubelet)
    Tasks: 28
   Memory: 107.8M
   CGroup: /system.slice/kubelet.service
           └─23864 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf
```

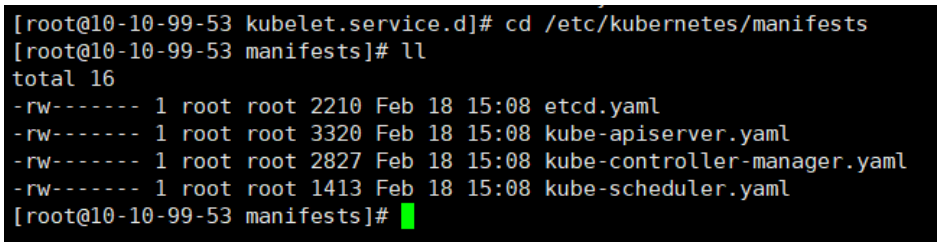
可以看到启动的配置文件

```
[root@10-10-99-53 kubelet.service.d]# cat 10-kubeadm.conf
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the KUBELET_KUBEADM_ARGS variable dynamically
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. Preferably, the user should use
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_EXTRA_ARGS should be sourced from this file.
EnvironmentFile=-/etc/sysconfig/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
```

在此文件中可以看到静态pod的所在路径

```
logging: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
```

静态pod的编排文件


[root@10-10-99-53 kubelet.service.d]# cd /etc/kubernetes/manifests
[root@10-10-99-53 manifests]# ll
total 16
-rw------- 1 root root 2210 Feb 18 15:08 etcd.yaml
-rw------- 1 root root 3320 Feb 18 15:08 kube-apiserver.yaml
-rw------- 1 root root 2827 Feb 18 15:08 kube-controller-manager.yaml
-rw------- 1 root root 1413 Feb 18 15:08 kube-scheduler.yaml
[root@10-10-99-53 manifests]#