《Kubernetes CKS 考试流程与真题解析》

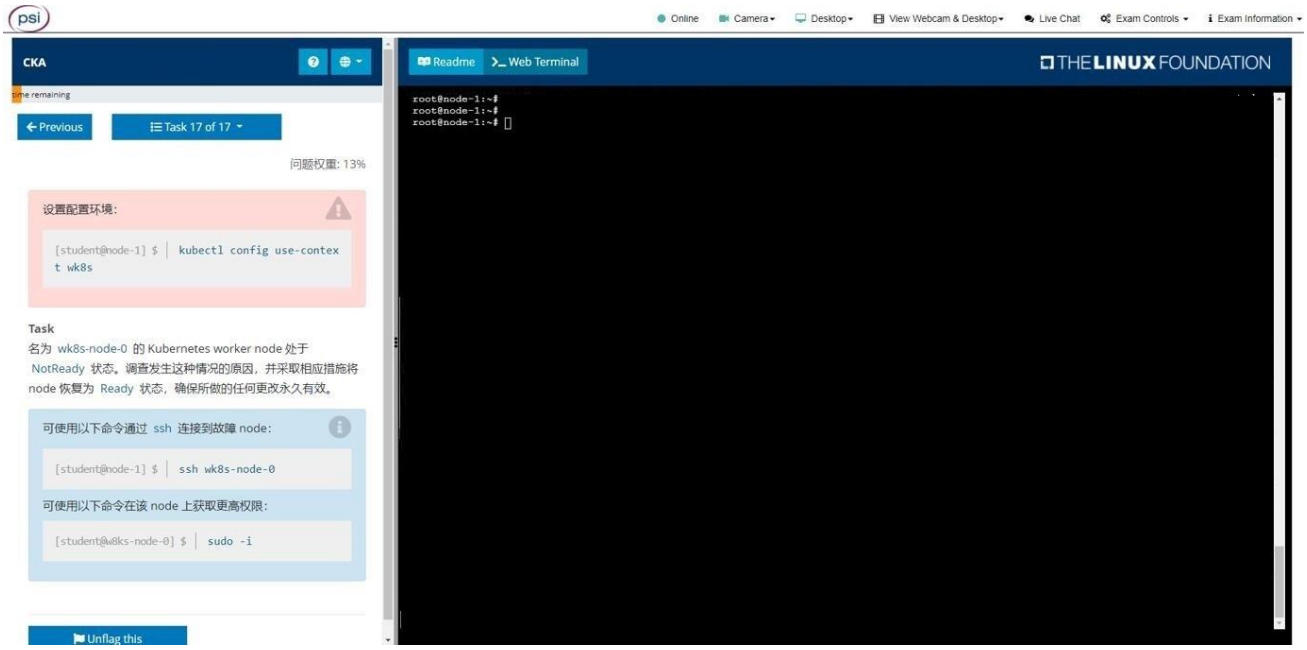

阿良个人微信


DevOps技术栈公众号

阿良教育：www.aliangedu.cn

考试流程

- 尽量提前 10 分钟左右进入倒计时页面（https://www.examslocal.com 点击登录->Or Sign In With->点击按钮 My Exams），等待出现进入考试系统的按钮（考官到位会出现），刚开始考官会让你出示证件，然后考官会让你拿摄像头环绕考试环境一周，确保房间无外人、桌面上没杂物，然后考官会让我们在考试系统里打开共享屏幕和摄像头按钮；最后考官会告诉我们一些注意事项，比如记事本功能，以及可以查阅哪些网站资料等等。考试过程中如有意外状况，可以在线与考官交流（文字），并争取你应有的权益；
- 只允许打开两个标签页，一个是考试系统，另一个建议打开官网文档 https://kubernetes.io/zh/docs/home/，将官网相关参考资料保存好书签，方便快速查找内容；
- K8s 集群环境采用 ubuntu 系统，不用担心，会普通用户执行 sudo -i 切 root 就行了，而且也基本不用切 root 操作。
- 考试系统界面



答题结束后，在 Exam Controls 里结束考试。

CKS 考试概要：

考试模式：线上考试

考试时间：2 小时

题目数量：16 题

分数：总 100 分，66 分及格

题目变化：每次考题会对里面涉及的命名空间、资源名称个别微调，还有题目顺序。

1、kube-bench

问题权重: 6%

您**必须**在以下 cluster /节点上完成此考题:

Cluster	Master 节点	工作节点
KSCS00201	kscs00201-master	kscs00201-worker1

您可以使用以下命令来切换 cluster / 配置环境:

```
[candidate@cli] $ kubectl config use-context KSCS00201
```

Context

针对 kubeadm 创建的 cluster 运行 CIS 基准测试工具时，发现了多个必须立即解决的问题。

Task

通过配置修复所有问题并重新启动受影响的组件以确保新设置生效。

修复针对 API 服务器发现的所有以下违规行为:

1.2.7	Ensure that the --authorization-mode argument is not set to AlwaysAllow	FAIL
1.2.8	Ensure that the --authorization-mode argument includes Node	FAIL
1.2.9	Ensure that the --authorization-mode argument includes RBAC	FAIL

修复针对 Kubelet 发现的所有以下违规行为:

4.2.1	Ensure that the anonymous-auth argument is set to false	FAIL
	Ensure that the	

解读：使用 kube-bench 工具检查集群组件配置文件存在的问题与修复，并重启对应组件确保新配置生效。

```
vi /etc/kubernetes/manifests/kube-apiserver.yaml
- --authorization-mode=Node,RBAC

vi /var/lib/kubelet/config.yaml
authentication:
  anonymous:
    enabled: false
    ...
authorization:
  mode: Webhook

vi /etc/kubernetes/manifests/etcd.yaml
- --client-cert-auth=true

systemctl restart kubelet # 修改 yaml 后，默认会重启容器，正常可以不用使用这个命令再重启，为确保重启生效可以再执行下
```

2、Pod 使用 ServiceAccount

您必须在以下 cluster /节点上完成此考题：

Cluster	Master 节点	工作节点
KSCH00301	ksch00301-master	ksch00301-worker1

您可以使用以下命令来切换 cluster / 配置环境：

```
[candidate@cli] $ kubectl config use-context KSCH00301
```

Context

您组织的安全策略包括：

- ServiceAccount 不得自动挂载 API 凭据
- ServiceAccount 名称必须以 "-sa" 结尾

清单文件 /home/candidate/KSCH00301/app-pod.yaml 中指定的 Pod 由于 ServiceAccount 指定错误而无法调度。

请完成以下项目：

Task

1 在现有 namespace prod 中创建一个名为 database-sa 的新 ServiceAccount 确保此 ServiceAccount 不自动挂载 API 凭据。

2 使用 /home/candidate/KSCH00301/app-pod.yaml 中的清单文件来创建一个 Pod。

3 最后，清理 namespace prod 中任何未使用的 ServiceAccount。

创建 SA：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: database-sa
```

```
  namespace: prod
automountServiceAccountToken: false
```

Pod 指定 SA：

```
# vi app-pod.yaml
...
spec:
 serviceAccountName: database-sa
```

删除 SA 命令：

```
kubectl delete sa xxx
```

参考资料：https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-service-account/

3、网络策略

解读：在 testing 命名空间创建一个名为 denypolicy 的网络策略。拒绝所有 Ingress 和 Egress 流量。将网络策略应用到 testing 命名空间中的所有 pod。

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: denynetwork
```

```
  namespace: testing
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

注：所有 pod 拒绝进出流量。

参考资料：https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/

4、Pod 安全策略（PSP）

解读：

1. 创建一个名为 restrict-policy 的 PodSecurityPolicy，防止创建特权 Pod
2. 创建一个名为 restrict-access-role 的 ClusterRole 能够使用 PSP restrict-policy
3. 在 staging 命名空间创建一个名为 psp-denial-sa 的 ServiceAccount
4. 最后，创建一个名为 dany-access-bind 的 ClusterRoleBinding，绑定 ClusterRole restrict-access-role 到 ServiceAccount psp-denial-sa

```
# 启用 PSP 准入控制器
vi /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
- --enable-admission-plugins=NodeRestriction,PodSecurityPolicy

systemctl restart kubelet

vi psp.yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restrict-policy
spec:
  privileged: false  # 不允许特权 Pod
  # 下面填写一些必要的字段
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
  - '*'

kubectl apply -f psp.yaml

kubectl create clusterrole restrict-access-role --verb=use --resource=psp --
resource-name=restrict-policy
kubectl create sa psp-denial-sa -n staging
kubectl create clusterrolebinding dany-access-bind --clusterrole=restrict-access-
role --serviceaccount=staging:psp-denial-sa
```

参考资料：https://kubernetes.io/zh/docs/concepts/policy/pod-security-policy/

5、RBAC

解答：

确认 web-pod 的 pod 是否指定 service-account-web，再看下对应 role 是否有对 services 资源的 get 操作。

```
kubectl create role web-role-2 --verb=delete --resource=namespaces -n monitoring
kubectl create rolebinding web-role-2-binding --role=web-role-2 --
serviceaccount=monitoring:service-account-web -n monitoring
```

参考资料：https://kubernetes.io/zh/docs/reference/access-authn-authz/rbac/

6、审计日志

解答：

```
vi /etc/kubernetes/logpolicy/sample-policy.yaml
apiVersion: audit.k8s.io/v1
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  - level: RequestResponse
    resources:
    - group: "batch/v1"
      resources: ["cronjobs"]
  - level: Request
    resources:
    - group: ""
      resources: ["deployments"]
    namespaces: ["app-frontend"]
  - level: Metadata
    resources:
    - group: ""
      resources: ["secrets", "configmaps"]
  - level: Metadata
```

```
  omitStages:
    - "RequestReceived"

vi /etc/kubernetes/manifests/kube-apiserver.yaml
- --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml
- --audit-log-path=/var/log/kubernetes/audit-logs.txt
- --audit-log-maxage=5
- --audit-log-maxbackup=2

# 在容器 volumeMounts 增加（注：考试环境默认已经配置挂载）
- mountPath: /etc/kubernetes/logpolicy/sample-policy.yaml
  name: log-policy
- mountPath: /var/log/kubernetes/audit-logs.txt
  name: audit-log
# 在 volumes 增加
- name: log-policy
  hostPath:
    path: /etc/kubernetes/logpolicy/sample-policy.yaml
    type: File
- name: audit-log
  hostPath:
    path: /var/log/kubernetes/audit-logs.txt
    type: FileOrCreate

systemctl restart kubelet
```

参考资料：https://kubernetes.io/zh/docs/tasks/debug-application-cluster/audit/

7、创建 Secret

解答：

```
kubectl get secrets -n istio-system dev-john -o jsonpath={.data.username} |
base64 -d > /home/candidate/old-username.txt
kubectl get secrets -n istio-system dev-john -o jsonpath={.data.password} |
base64 -d >  /home/candidate/password.txt

kubectl create secret generic dev-mark -n istio-system --from-
literal=username=thanos --from-literal=password=aV7HR7nU3JLx

apiVersion: v1
kind: Pod
metadata:
  name: dev-pod
  namespace: istio-system
spec:
 containers:
 - name: test-secret-container
```

```
  image: nginx
  volumeMounts:
  - name: test-secret-volume
   mountPath: /etc/secret
volumes:
- name: test-secret-volume
 secret:
  secretName: dev-mark
```

参考资料：https://kubernetes.io/zh/docs/concepts/configuration/secret/

8、Dockerfile 和 Deployment 优化

解答：

```
vi /home/candidate/KSSC00301/Dockerfile
#USER root

vi /home/candidate/KSSC00301/deployment.yaml
…
    securityContext:
      capabilities:
        add: ["NET_BIND_SERVICE"]
```

```
      #privileged: true    # 关闭这个特权启用
```

9、gVisor 安全运行容器

解答：

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
 name: sandboxed
handler: runsc

kubectl edit deployments web -n client
...
spec:
 runtimeClassName: sandboxed
 containers:
 - image: nginx:1.19
```

注：这个题目前有点变化，可能会让你修改 3 个 deployment

参考资料：https://kubernetes.io/zh/docs/concepts/containers/runtime-class/

10、删除启用特权的 Pod

解读：检查在 development 命名空间运行的 Pod，并删除任何不是非无状态（有状态）或非不可变（可变）的 Pod

以下是对无状态和不可变的解释：

- 在容器中存储数据的 Pod 视为非无状态（不用担心容器数据持久化）
- 启用特权的 Pod 都视为潜在的非无状态和非不可变

```
kubectl get pod -n development -o yaml | grep -E "privileged|volumeMounts" # 确认启用特权模式的 Pod 和是否挂载数据卷，记得忽略 SA 的挂载！

kubectl delete pod <pod-name> -n development # 删除符合的 pod
```

参考资料：https://kubernetes.io/zh/docs/concepts/security/pod-security-standards/

11、网络策略

解读：创建一个名为 pod-restriction 的网络策略，以限制命名空间 dev-team 中 products-service Pod。
只允许以下 Pod 连接 products-service Pod：

- qa 命名空间中的 Pod
- Pod 标签为 environment:testing，在所有命名空间

```
kubectl get pod -n dev-team --show-labels  # 查看 pod 标签（environment: staging）
kubectl get ns --show-labels # 查看 qa 命名空间标签（name: testing）
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-restriction
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: staging
  policyTypes:
  - Ingress
  ingress:
```

```
  - from:
    - namespaceSelector:
        matchLabels:
          name: testing
  - from:
    - namespaceSelector: {}
      podSelector:
        matchLabels:
          environment: testing
```

参考资料：https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/

12、ImagePolicyWebhook

解答：

```
# 第一步
vi /etc/kubernetes/manifests/kube-apiserver.yaml
...
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook
- --admission-control-config-file=/
etc/kubernetes/epconfig/admission_configuration.json
...
```

```
# 在容器 volumeMounts 增加（注：考试环境默认已经配置挂载）
- mountPath: /etc/kubernetes/epconfig
  name: epconfig
# 在 volumes 增加
- name: epconfig
  hostPath:
    path: /etc/kubernetes/epconfig

# 第二步
vi /etc/kubernetes/epconfig/admission_configuration.json
defaultAllow: false

# 第三步
vi /etc/kubernetes/epconfig/kubeconfig.yaml
server: https://acme.local:8082/image_policy

systemctl restart kubelet

# 第四步 测试
kubectl apply -f /root/KSSC00202/vulnerable-resource.yml
tail /var/log/imagepolicy/wakanda.log
```

参考资料：https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#imagepolicywebhook

13、Trivy 扫描镜像安全漏洞

解答：

```
kubectl get po -n kamino

for i in $(kubectl get pods -n kamino -o name);do
       kubectl get $i -o yaml -n kamino |grep "image:"
done
# 一条命令：for i in $(kubectl get pods -n kamino -o name);do kubectl get $i -o
 yaml -n kamino |grep "image:"; done

sudo trivy image -s HIGH,CRITICAL nginx:1.19
kubectl delete pod <pod-name> # 删除具有扫描的镜像带有高危或严重漏洞的 Pod
```

14、启用 Kubernetes API 认证

解答：

```
vi /etc/kubernetes/manifests/kube-apiserver.yaml
    - kube-apiserver
    - --authorization-mode=Node,RBAC   # 只保留这两个
    - --enable-admission-plugins=NodeRestriction  # 只保留这一个

systemctl restart kubelet

# 删除角色绑定
kubectl delete clusterrolebinding system:anonymous
```

15、AppArmor

解读：在集群的工作节点上，执行下面准备好的配置文件/etc/apparmor.d/nginx_apparmor
编辑准备好的清单文件/home/candidate/KSSH00401/nginx-deploy.yaml 应用于 AppArmor 文件。
最后，在 Pod 中应用 apparmor 策略文件。

```
# 先进入工作节点操作
ssh kssh00401-worker1

# 确认策略名称
vi /etc/apparmor.d/nginx_apparmor
# nginx-profile-3
或者
apparmor_status | grep nginx

apparmor_parser /etc/apparmor.d/nginx_apparmor # 加载策略

# 在管理节点操作
vi /home/candidate/KSSH00401/nginx-deploy.yaml
```

```
...
  annotations:
  container.apparmor.security.beta.kubernetes.io/<容器名称>: localhost/nginx-profile-3

kubectl apply -f /home/candidate/KSSH00401/nginx-deploy.yaml
```

参考资料：https://kubernetes.io/zh/docs/tutorials/clusters/apparmor/

16、Sysdig

使用运行时检测工具来检测 Pod tomcat 单个容器中频发生成和执行的异常进程。
有两种工具可供使用：

- Sysdig
- Falco

使用 sysdig 工具收集 30 秒，将事件写到/opt/KSR00101/incidents/summary 文件，并指定格式输出，格式如下：

```
[timestamp],[uid],[processName]
```

捕获tomcat容器系统调用，指定容器ID：

```
docker ps | grep tomcat
sudo sysdig -M 30 -p "%evt.time,%user.uid,%proc.name" container.id=b1dacef30135 > /opt/KSR00101/incidents/summary
```

注：

- 可以使用 sysdig -l |grep time 过滤，确认输出格式字段
- 这些工具只预装在集群的工作节点，不在管理节点，需使用 sudo 权限运行。

阿良个人微信

DevOps技术栈公众号

阿良教育：www.aliangedu.cn