

# 《Kubernetes CKS 考试流程与真题解析》



阿良个人微信

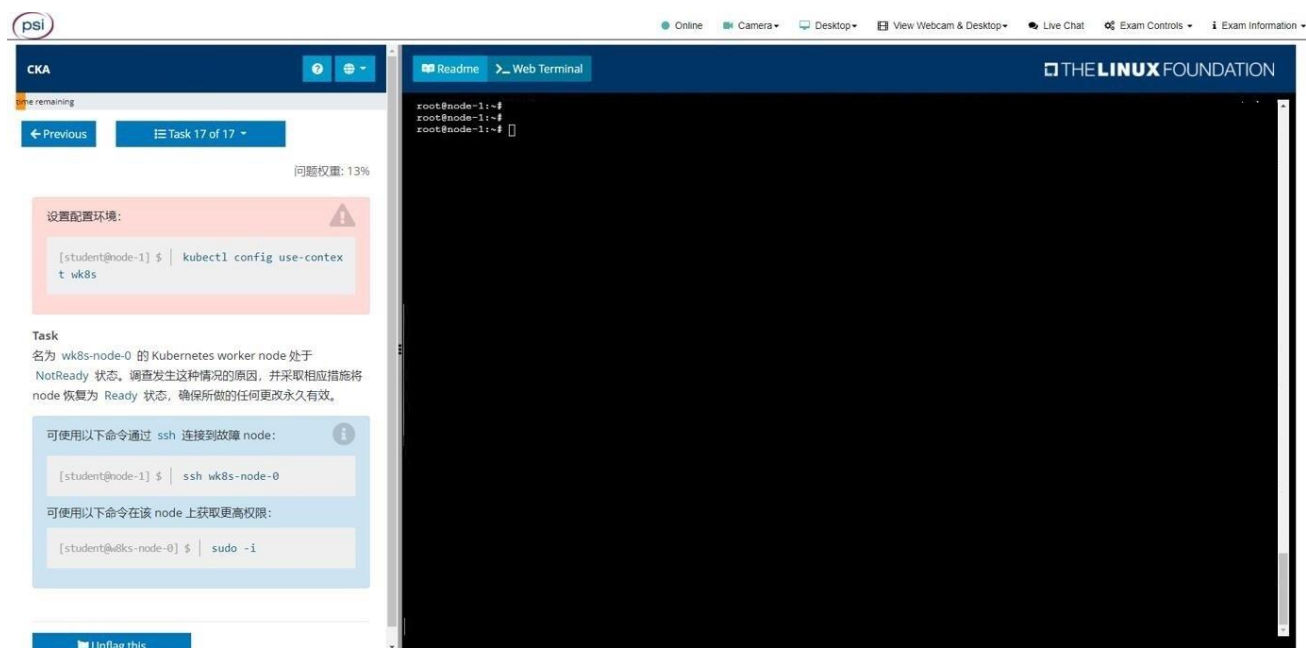


DevOps技术栈公众号

阿良教育: [www.aliangedu.cn](http://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:

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

修复针对 Kubelet 发现的所有以下违规行为:

	Ensure that the	
4.2.1	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
KSCH0030	ksch00301-maste	ksch00301-work
1	r	er1

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

您必须在以下 cluster / 节点上完成此考题:

Cluster	Master 节点	工作节点
KSCS00101	kscs00101-master	kscs00101-worker1

您可以使用以下命令来切换 cluster / 配置环境:

```
[candidate@cli] $ kubectl config use-context KSCS00101
```

**Context**

一个默认拒绝 (default-deny) 的 NetworkPolicy 可避免在未定义任何其他 NetworkPolicy 的 namespace 中意外公开 Pod。

**Task**

为所有类型为 Ingress + Egress 的流量在 namespace testing 中创建一个名为 deny-network 的新默认拒绝 NetworkPolicy。

此新的 NetworkPolicy 必须拒绝 namespace testing 中的所有 Ingress + Egress 流量。

将新创建的默认拒绝 NetworkPolicy 应用于在 namespace testing 中运行的所有 Pod。

您可以在 `/home/candidate/KSCS00101/network-policy.yaml` 找到一个模板清单文件。

解读: 在 testing 命名空间创建一个名为 deny-policy 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: deny-network
```

```
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）

您必须在以下 cluster /节点上完成此考题：

Cluster	Master 节点	工作节点
KSMV0010	ksmv00102-mast 2	ksmv00102-wor ker1

您可以使用以下命令来切换 cluster / 配置环境：

```
[candidate@cli] $ kubectl config use-context KSMV00102
```

PodSecurityPolicy deny-policy 的 ClusterRole。

在现有 namespace development 中创建一个名为 psp-denial-sa 的新 ServiceAccount。

最后，新建一个名为 restrict-access-bind 的 ClusterRoleBinding，将新创建的 ClusterRole restrict-access-role 绑定到新创建的 ServiceAccount psp-denial-sa。

您可以在以下位置找到模板清单文件：

- /home/candidate/KSMV00102/pod-security-policy.yaml
- /home/candidate/KSMV00102/cluster-role.yaml
- /home/candidate/KSMV00102/service-account.yaml
- /home/candidate/KSMV00102/cluster-role-binding.yaml

**Context**  
PodSecurityPolicy 应防止在特定 namespace 中特权 Pod 的创建。

**Task**  
创建一个名为 deny-policy 的新 PodSecurityPolicy，以防止特权 Pod 的创建。

创建一个名为 restrict-access-role 并使用新创建的

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



您**必须**在以下 cluster / 节点上完成此考题:

Cluster	Master 节点	工作节点
KSCH0020	ksch00201-master	ksch00201-worker1

您可以使用以下命令来切换 cluster / 配置环境:

```
[candidate@cli] $ kubectl config use-context KSCH00201
```

#### Context

绑定到 Pod 的 ServiceAccount 的 Role 授予过度宽松的权限。完成以下项目以减少权限集。

#### Task

一个名为 `web-pod` 的现有 Pod 已在 namespace `monitoring` 中运行。

编辑绑定到 Pod 的 ServiceAccount `service-account-web` 的现有 Role，仅允许只对 `services` 类型的资源执行 `get` 操作。

在 namespace `monitoring` 中创建一个名为 `web-role-2`，并仅允许只对 `namespaces` 类型的资源执行 `delete` 操作的新 Role。

创建一个名为 `web-role-2-binding` 的新 RoleBinding，将新创建的 Role 绑定到 Pod 的 ServiceAccount。

请勿删除现有的 RoleBinding。

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

您必须在以下 cluster / 节点上完成此考题:

Cluster	Master 节点	工作节点
KRS00602	ksrs00602-master	ksrs00602-worker1

您可以使用以下命令来切换 cluster / 配置环境:

```
[candidate@cli] $ kubectl config use-context KRS00602
```

### Task

在 cluster 中启用审计日志。为此，请启用日志后端，并确保：

- 日志存储在  
`/var/log/kubernetes/kubernetes-logs.txt`
- 日志文件能保留 5 天
- 最多保留 2 个旧审计日志文件

`/etc/kubernetes/logpolicy/sample-policy.yaml` 提供了基本策略。它仅指定不记录的内容。

基本策略位于 cluster 的 master 节点上。

编辑和扩展基本策略以记录：

- RequestResponse 级别的 `cronjobs` 更改
- namespace `app-frontend` 中 `deployments` 更改的请求体
- Metadata 级别的所有 namespace 中的 ConfigMap 和 Secret 的更改

此外，添加一个全方位的规则以在 Metadata 级别记录所有其他请求。

不要忘记应用修改后的策略。

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

您必须在以下 cluster /节点上完成此考题:

Cluster	Master 节点	工作节点
KSMV0020	ksmv00201-master	ksmv00201-worker1

您可以使用以下命令来切换 cluster / 配置环境:

```
[candidate@cli] $ kubectl config use-context KSMV0020
```

不要在以下步骤中使用/修改先前创建的文件, 如果需要, 创建新的临时文件。

在 istio-system namespace 中创建一个名为 dev-mark 的新 secret, 内容如下:

```
username: thanos
password: aV7HR7nU3JLx
```

**Task**

在 namespace istio-system 中获取名为 dev-john 的现有 secret 的内容。

将 username 字段存储在名为 /home/candidate/old-username.txt 的文件中, 并将 password 字段存储在名为 /home/candidate/password.txt 的文件中。

您必须创建以上两个文件; 它们还不存在。

最后, 创建一个新的 Pod, 它可以通过卷访问 secret dev-mark :

Pod 名	dev-pod
Namespace	istio-system
容器名	test-secret-container
镜像	nginx
卷名	test-secret-volume
挂载路径	/etc/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



[candidate@cli] \$ kubectl config use-context KSSC00301

**Task**  
分析和编辑给定的 Dockerfile  
/home/candidate/KSSC00301/Dockerfile (基于 ubuntu:16.04 镜像) 并修复在文件中拥有突出的安全/最佳实践问题的**两个指令**。

分析和编辑给定的清单文件  
/home/candidate/KSSC00301/deployment.yaml 并修复在文件中拥有突出的安全/最佳实践问题的**两个字段**。

请勿添加或删除配置设置；只需修改现有的配置设置让以上**两个**配置设置都不再有安全/最佳实践问题。

如果您需要非特权用户来执行任何项目，请使用用户 ID 65535 的用户 nobody。

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

```
[candidate@cli] $ kubectl config use-context KSR500501
```

**Context**  
最佳实践是将容器设计为无状态和不可变的。

**Task**  
检查在 namespace `development` 中运行的 Pod，并删除任何**非无状态或非不可变**的 Pod。

使用以下对无状态和不可变的严格解释：

- 能够在容器内存储数据的 Pod 必须被视为**非无状态**的。

您不必担心数据是否实际上已经存储在容器中。

- 被配置为任何形式的特权的 Pod 必须被视为可能是**非无状态和无不可变的**。

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

```
[candidate@cli] $ kubectl config use-context KSSH00301
```

**Task**

创建一个名为 `pod-restriction` 的 NetworkPolicy 来限制对在 namespace `dev-team` 中运行的 Pod `products-service` 的访问。

只允许以下 Pod 连接到 Pod `products-service` :

- namespace `testing` 中的 Pod
- 位于任何 namespace, 带有标签 `environment: testing` 的 Pod

确保应用 NetworkPolicy.

你可以在 `/home/candidate/KSSH00301/network-policy.yaml` 找到一个模板清单文件。

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

您必须在以下 cluster / 节点上完成此考题：

Cluster	Master 节点	工作节点
KSSC00202	kssc00202-master	kssc00202-worker1

您可以使用以下命令来切换 cluster / 配置环境：

```
[candidate@cli] $ | kubectl config use-context KSSC00202
```

### Context

cluster 上设置了容器镜像扫描器，但尚未完全集成到 cluster 的配置中。完成后，容器镜像扫描器应扫描并拒绝易受攻击的镜像的使用。

### Task

您必须在 cluster 的 master 节点上完成整个考题，所有服务和文件都已被准备好并放置在该节点上。

给定一个目录 `/etc/kubernetes/etcdconfig` 中不完整的配置以及具有 HTTPS 端点 `https://acme.local:8082/image_policy` 的功能性容器镜像扫描器：

1. 启用必要的插件来创建镜像策略
2. 校验控制配置并将其更改为隐式拒绝 (implicit deny)
3. 编辑配置以正确指向提供的 HTTPS 端点

最后，通过尝试部署易受攻击的资源 `/root/KSSC00202/vulnerable-resource.yml` 来测试配置是否有效。

您可以在 `/var/log/imagepolicy/wakanda.log` 找到容器镜像扫描仪的日志文件。

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

您可以使用以下命令来切换 cluster / 配置环境：

```
[candidate@cli] $ | kubectl config use-  
context KSSC00401
```

您可以使用浏览器打开一个额外的标签页来访问 Trivy 的文档。

#### Task

使用 Trivy 开源容器扫描器检测 namespace **kamino** 中 Pod 使用的具有严重漏洞的镜像。

查找具有 **High** 或 **Critical** 严重性漏洞的镜像，并删除使用这些镜像的 Pod。

Trivy 仅预装在 cluster 的 master 节点上；它在原本的系统或工作节点上不可用。您必须连接到 cluster 的 master 节点才能使用 Trivy。

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

您必须在以下 cluster / 节点上完成此考题:

Cluster	Master 节点	工作节点
KSCH0010	ksch00101-master1	ksch00101-worker1

您可以使用以下命令来切换 cluster / 配置环境:

```
[candidate@cli] $ kubectl config use-context KSCH00101
```

### Context

由 kubeadm 创建的 cluster 的 Kubernetes API 服务器, 出于测试目的, 临时配置为允许未经身份验证和未经

授权的访问, 授予匿名用户 cluster-admin 的访问权限。

### Task

重新配置 cluster 的 Kubernetes API 服务器以确保只允许经过身份验证和授权的 REST 请求。

使用授权模式 Node,RBAC 和准入控制器 NodeRestriction。

删除用户 system:anonymous 的 ClusterRoleBinding 来进行清理。

所有 kubectl 配置环境/文件也被配置为使用未经身份验证和未经授权的访问。您不必更改它, 但请注意, 一旦完成 cluster 的安全加固, kubectl 的配置将无法工作。

您可以使用位于 cluster 的 master 节点上, cluster 原本的 kubectl 配置文件 /etc/kubernetes/admin.conf, 以确保经过身份验证和授权的请求仍然被允许。

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

```
[candidate@cli] $ | kubectl config use-  
context KSSH00401
```

### Context

AppArmor 已在 cluster 的工作节点上被启用。一个 AppArmor 配置文件已存在，但尚未被实施。

您可以使用浏览器打开一个额外的标签页来访问 AppArmor 的文档。

### Task

在 cluster 的工作节点上，实施位于 `/etc/apparmor.d/nginx_apparmor` 的现有 AppArmor 配置文件。

编辑位于

`/home/candidate/KSSH00401/nginx-deploy.yaml` 的现有清单文件以应用 AppArmor 配置文件。

最后，应用清单文件并创建其中指定的 Pod。

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](http://www.aliangedu.cn)