注意 必读!

模拟环境是在 node01 （ 11.0.1.112 ） 上做题的。
帐号为 candidate
密码为 123
在此账号下，可以免密 ssh 登录 master01 和 node02 （ ssh master01 或 ssh node02 ）
3 台虚拟机的 candidate 帐号也可以免密切换到 root （ sudo -i ）

考试时，是 candidate 或 student 用户，在 node-1 机器上做题的，也不是在 root 的，也不是在 master 的，要注意。

（请注意，文档里的一些截图还是使用的 student 帐号，没有更新。但这不影响的。只是因为在 v1.23 里用的是 student，截图是当时 v1.23 时候截的，但是在 v1.24 里用的帐号是 candidate 了。）

注意，官方的 CKS 考试环境是有多套的，你考试时，不一定抽到哪套。不同考试环境里面部分题目的内容有很小的变化，但题干都是一样的。在下面的答案解析中也有详细的描述。

注意阅读《K8S 考试流程(报名、约考、注意).pdf》，几乎所有同学都反馈新考试平台非常卡，70%反馈不用 V-P-N 没法做题，20%反馈用了 V-P-N 还是卡，甚至有几个同学没有做完题。
所以日常要练习的很熟练，尽量熟练到 80%的题不参考 K8S 网页！！！

新考试平台，不允许外接第二个屏幕，不允许访问自己的浏览器书签。
新考试平台，更像是一个远程的 Ubuntu 桌面，在这个 Ubuntu 桌面里，你可以用终端做题，也可以用 Ubuntu 自带的火狐浏览器到官网自己查。
另外每道题目都会给你一个官方的参考链接，可以先看这个连接，如果找不到自己需要的信息，再从官网自己搜关键字。
注意考试时搜出来的结果，跟平常在官网搜出来结果排序不一样。还有就是官网网页的最右边，是可以点击选择中文/英文的。
最后，能不查官网就尽量不要查，因为真的很卡，很费时间。

每次还原初始化快照后，开机后等 5 分钟，ssh 登录 node01（11.0.1.112）检查一下所有的 pod，确保都为 Running，再开始练习。
kubectl get pod -A


```
candidate@node01:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
master01   Ready    control-plane   10d   v1.24.2
node01     Ready    <none>          9d    v1.24.2
node02     Ready    <none>          9d    v1.24.2
candidate@node01:~$
candidate@node01:~$ kubectl get pod -A
NAMESPACE          NAME                                       READY   STATUS    RESTARTS      AGE
calico-apiserver   calico-apiserver-7b457579c5-dvnqm          1/1     Running   4 (71s ago)   9d
calico-apiserver   calico-apiserver-7b457579c5-wbqsb          1/1     Running   4 (71s ago)   9d
calico-system      calico-kube-controllers-86dff98c45-6nmqs   1/1     Running   4 (71s ago)   9d
calico-system      calico-node-2t8w6                          1/1     Running   4 (71s ago)   9d
calico-system      calico-node-48784                          1/1     Running   5 (71s ago)   9d
calico-system      calico-node-hhxt2                          1/1     Running   4 (71s ago)   9d
calico-system      calico-typha-68d6c564f-4vlvm               1/1     Running   5 (71s ago)   9d
calico-system      calico-typha-68d6c564f-lxdh2               1/1     Running   5 (71s ago)   9d
db                 web-pod                                    1/1     Running   3 (70s ago)   6h1m
default            tomcat123                                  1/1     Running   3 (71s ago)   5h20m
dev-team           products-service                           1/1     Running   3 (71s ago)   5h36m
kamino             trilll                                     1/1     Running   3 (70s ago)   5h26m
kamino             tri222                                     2/2     Running   6 (70s ago)   5h26m
kamino             tri333                                     2/2     Running   6 (71s ago)   5h26m
kube-system        coredns-74586cf9b6-82vwb                   1/1     Running   5 (71s ago)   10d
kube-system        coredns-74586cf9b6-ppqhp                   1/1     Running   5 (72s ago)   10d
kube-system        etcd-master01                              1/1     Running   5 (71s ago)   10d
kube-system        kube-apiserver-master01                    1/1     Running   5 (71s ago)   10d
kube-system        kube-controller-manager-master01           1/1     Running   5 (71s ago)   10d
kube-system        kube-proxy-7dn92                           1/1     Running   5 (71s ago)   9d
kube-system        kube-proxy-g7ct6                           1/1     Running   5 (71s ago)   10d
kube-system        kube-proxy-n6ffp                           1/1     Running   5 (71s ago)   9d
kube-system        kube-scheduler-master01                    1/1     Running   5 (71s ago)   10d
production         pri001                                     1/1     Running   3 (71s ago)   5h41m
production         pri002                                     1/1     Running   3 (71s ago)   5h41m
production         pri003                                     1/1     Running   3 (71s ago)   5h41m
qaqa               qaqa-pod                                   1/1     Running   3 (71s ago)   5h36m
server             busybox-run-665877f69b-jtb9p               1/1     Running   3 (71s ago)   5h43m
server             nginx-host-d9bf9b9c-2hqpd                  1/1     Running   3 (71s ago)   5h43m
server             run-test-7c955665bb-b6wmh                  1/1     Running   3 (70s ago)   5h43m
tigera-operator    tigera-operator-5dc8b759d9-5w6q9           1/1     Running   8 (25s ago)   9d
candidate@node01:~$
```


kubectl 备忘单（kubectl Cheat Sheet）
https://kubernetes.io/zh-cn/docs/reference/kubectl/cheatsheet/

这套《题库》跟《真题》的题干是一样的，区别在于里面的一些 pod、deployment、namespace、ServiceAccount 等参数可能不同而已，因为在真实考试中，也是会时常变换里面的这些变量参数的。注意理解这些变量的含义，而不要死记硬背答案。

比如以下截图：

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。

Task

在 namespace istio-system 中获取名为 db1-test 的现有 secret 的内容

将 username 字段存储在名为 /cks/sec/user.txt 的文件中，并将 password 字段存储在名为 /cks/sec/pass.txt 的文件中。
注意：你必须创建以上两个文件，他们还不存在。

鸣谢：

特别鸣谢以下网友（排名不分先后）：

勋（lizhanxun4*）
唐 Jing（sinot*）
gmwinsto* [台]

1、kube-bench 修复不安全项

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的设置生效。

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
1.2.18 Ensure that the --insecure-bind-address argument is not set FAIL （1.23 中这项题目没给出，但最好也检查一下，模拟环境里需要改）
~~1.2.19 Ensure that the --insecure-port argument is set to 0~~ FAIL （1.23 中这项题目没给出，不需要再修改了）

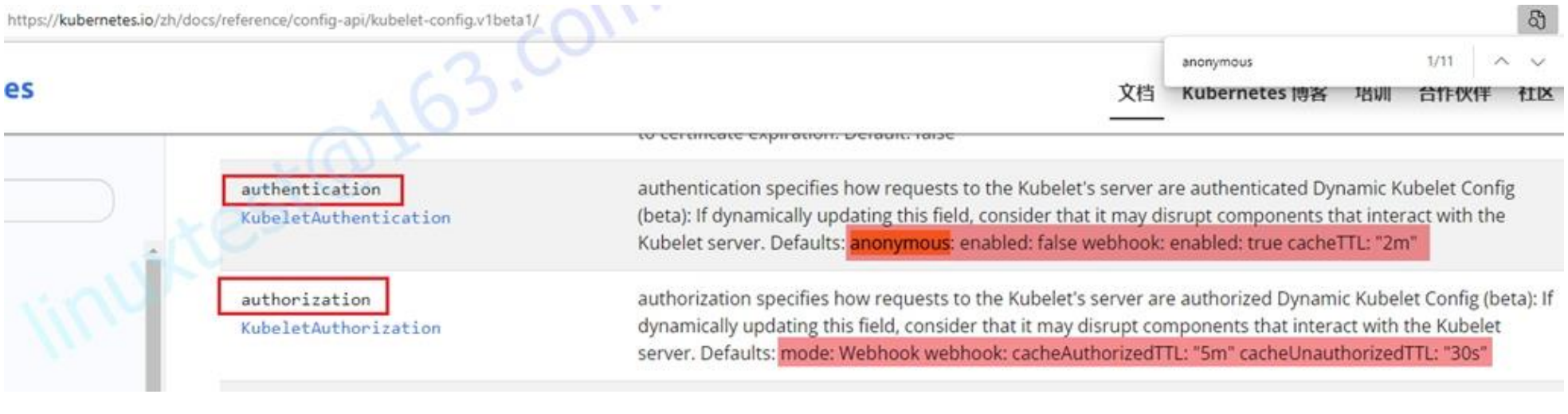
修复针对 kubelet 发现的所有以下违规行为：
Fix all of the following violations that were found against the kubelet:
4.2.1 Ensure that the anonymous-auth argument is set to false FAIL
4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow FAIL
注意：尽可能使用 Webhook 身份验证/授权。

修复针对 etcd 发现的所有以下违规行为：
Fix all of the following violations that were found against etcd:
2.2 Ensure that the --client-cert-auth argument is set to true FAIL

模拟环境里，初始化这道题的脚本为 a.sh

参考资料：

https://kubernetes.io/zh/docs/reference/config-api/kubelet-config.v1beta1/



答题：

考试时务必执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSCS00201

```
student@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
student@node-1:~$
```

模拟环境因为没有做相关设置，只有一套集群，所以执行会报错的。
但是建议每道题练习时，还是要输入一遍切换集群的命令，固化思维，防止考试时忘记切换。

```
student@node01:~$ kubectl config use-context k8s
error: no context exists with the name: "k8s"
student@node01:~$
```

我的微信是 shadowooom 有在考试模拟环境里做题问题，考试流程问题，都可以问我的。

1 切换到 Master 的 root 下

ssh master01

sudo -i

```
candidate@node01:~$ ssh master01
```

```
candidate@master01:~$ sudo -i
root@master01:~#
```

请先执行如下命令，模拟这道题的初始环境。

脚本在 master01 的/root 目录下。

sh a.sh

```
root@master01:~# sh a.sh
root@master01:~#
```

2 修改 api-server

#可以使用这条命令查，要记住

kube-bench master

```
[FAIL] 1.2.6 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)
[FAIL] 1.2.7 Ensure that the --authorization-mode argument includes Node (Automated)
[FAIL] 1.2.8 Ensure that the --authorization-mode argument includes RBAC (Automated)
[FAIL] 1.2.18 Ensure that the --insecure-bind-address argument is not set (Automated)
[FAIL] 1.2.19 Ensure that the --insecure-port argument is set to 0 (Automated)

。。。

1.2.7 Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml
on the master node and set the --authorization-mode parameter to values other than AlwaysAllow.
One such example could be as below.
--authorization-mode=RBAC

1.2.8 Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml
on the master node and set the --authorization-mode parameter to a value that includes Node.
--authorization-mode=Node,RBAC

1.2.9 Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml
on the master node and set the --authorization-mode parameter to a value that includes RBAC,
for example:
--authorization-mode=Node,RBAC

1.2.18 Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml
on the master node and remove the --insecure-bind-address parameter.
```

查出来的，可能很多不安全项，但只修改考题要求的哪几项就行的。

修改之前，备份一下配置文件。

mkdir bak1

cp /etc/kubernetes/manifests/kube-apiserver.yaml bak1/

vim /etc/kubernetes/manifests/kube-apiserver.yaml

#修改、添加、删除相关内容

#修改 authorization-mode，注意 Node 和 RBAC 之间的符号是英文状态的逗号，而不是点。

```
    - --authorization-mode=Node,RBAC
```

#删除 insecure-bind-address，考试中，有可能本来就没写这行。

```
    - --insecure-bind-address=0.0.0.0
```

3 修改 kubelet

#可以使用这条命令查，要记住

kube-bench node

[FAIL] 4.2.1 Ensure that the anonymous-auth argument is set to false (Automated)
[FAIL] 4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)

。。。
4.2.1 If using a Kubelet config file, edit the file to set authentication: anonymous: enabled to
false.
If using executable arguments, edit the kubelet service file
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf on each worker node and
set the below parameter in KUBELET_SYSTEM_PODS_ARGS variable.
--anonymous-auth=false
Based on your system, restart the kubelet service. For example:
systemctl daemon-reload
systemctl restart kubelet.service

4.2.2 If using a Kubelet config file, edit the file to set authorization: mode to Webhook. If
using executable arguments, edit the kubelet service file
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf on each worker node and
set the below parameter in KUBELET_AUTHZ_ARGS variable.
--authorization-mode=Webhook
Based on your system, restart the kubelet service. For example:
systemctl daemon-reload
systemctl restart kubelet.service

修复方法 1：（推荐）
如果考试时，你用方法 1 改完后，kubelet 启动不了，肯定是你哪里改错了，如果又排查不出来，则恢复此文件，再使用方法 2 修改。
systemctl status kubelet

```
root@master01:~# systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Mon 2022-02-28 20:51:47 CST; 2min 58s ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 9324 (kubelet)
      Tasks: 15 (limit: 2531)
     Memory: 42.8M
     CGroup: /system.slice/kubelet.service
             └─9324 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf
```

cat /etc/systemd/system/kubelet.service.d/10-kubeadm.conf 中你也会看到 Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"。
所以去修改这个文件
修改之前，备份一下配置文件。
mkdir bak1
cp /var/lib/kubelet/config.yaml bak1/

vim /var/lib/kubelet/config.yaml

修改
```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:        #修改 anonymous 下的，将 true 改为 false
    enabled: false      #改为 false
  webhook:
    cacheTTL: 0s
    enabled: true   #这个 webhook 下的 true 不要改
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:          #修改 authorization 下的
  mode: Webhook     #改为 Webhook
  webhook:
……
```

#编辑完后重新加载配置文件，并重启 kubelet
systemctl daemon-reload
systemctl restart kubelet.service

修复方法 2：（不推荐，建议用方法 1）
考试中，任选一种方法即可，模拟环境里推荐第一种方法。
systemctl status kubelet
修改之前，备份一下配置文件
mkdir bak1
cp /etc/systemd/system/kubelet.service.d/10-kubeadm.conf bak1/
vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

#在[Service]下面添加
Environment="KUBELET_SYSTEM_PODS_ARGS=--anonymous-auth=false"
Environment="KUBELET_AUTHZ_ARGS=--authorization-mode=Webhook"

#在 ExecStart 后追加
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS $KUBELET_SYSTEM_PODS_ARGS $KUBELET_AUTHZ_ARGS

#编辑完后重新加载配置文件，并重启 kubelet
systemctl daemon-reload
systemctl restart kubelet.service

4 修改 etcd

#可以使用这条命令查，要记住
kube-bench

[FAIL] 2.2 Ensure that the --client-cert-auth argument is set to true (Automated)

。。。
2.2 Edit the etcd pod specification file /etc/kubernetes/manifests/etcd.yaml on the master node and set the below parameter.
--client-cert-auth="true"

修改之前，备份一下配置文件。
mkdir bak1
cp /etc/kubernetes/manifests/etcd.yaml bak1/
vim /etc/kubernetes/manifests/etcd.yaml
修改
- --client-cert-auth=true #修改为 true

修改完成后，等待 2 分钟，再检查一下所有 pod，确保模拟环境里的所有 pod 都正常。
kubectl get pod -A

退出 root，退回到 candidate@master01
exit
退出 master01，退回到 candidate@node01
exit

```
root@master01:~# exit
logout
student@master01:~$ exit
logout
Connection to master01 closed.
student@node01:~$
```

做下一题之前，确保所有的 pod 都是 Running，特别是 kube-apiserver-master01 也正常。（考试时，也要确保这个 apiserver 是正常的）
模拟环境里，calico-apiserver 和 calico-system 的几个 pod 可能需要比较长的时间才能恢复到 Running 状态，此时只要确保其他 pod 已恢复 Running 就可以继续做题了。

```
candidate@node01:~$ kubectl get pod -A
NAMESPACE          NAME                                       READY   STATUS    RESTARTS      AGE
calico-apiserver   calico-apiserver-7b457579c5-dvnqm          1/1     Running   4 (96s ago)   9d
calico-apiserver   calico-apiserver-7b457579c5-wbqsb          1/1     Running   4 (96s ago)   9d
calico-system      calico-kube-controllers-86dff98c45-6nmqs   1/1     Running   4 (96s ago)   9d
calico-system      calico-node-2t8w6                          1/1     Running   4 (96s ago)   9d
calico-system      calico-node-48784                          1/1     Running   5 (96s ago)   9d
calico-system      calico-node-hhxt2                          1/1     Running   4 (96s ago)   9d
calico-system      calico-typha-68d6c564f-4vlvm               1/1     Running   5 (96s ago)   9d
calico-system      calico-typha-68d6c564f-lxdh2               1/1     Running   5 (96s ago)   9d
db                 web-pod                                    1/1     Running   3 (95s ago)   6h2m
default            tomcat123                                  1/1     Running   3 (96s ago)   5h20m
dev-team           products-service                           1/1     Running   3 (96s ago)   5h36m
kamino             tri111                                     1/1     Running   3 (95s ago)   5h26m
kamino             tri222                                     2/2     Running   6 (95s ago)   5h26m
kamino             tri333                                     2/2     Running   6 (96s ago)   5h26m
kube-system        coredns-74586cf9b6-82vwb                   1/1     Running   5 (96s ago)   10d
kube-system        coredns-74586cf9b6-ppqhp                   1/1     Running   5 (97s ago)   10d
kube-system        etcd-master01                              1/1     Running   5 (96s ago)   10d
kube-system        kube-apiserver-master01                    1/1     Running   5 (96s ago)   10d
kube-system        kube-controller-manager-master01           1/1     Running   5 (96s ago)   10d
kube-system        kube-proxy-7dn92                           1/1     Running   5 (96s ago)   9d
kube-system        kube-proxy-g7ct6                           1/1     Running   5 (96s ago)   10d
kube-system        kube-proxy-n6ffp                           1/1     Running   5 (96s ago)   9d
kube-system        kube-scheduler-master01                    1/1     Running   5 (96s ago)   10d
production         pri001                                     1/1     Running   3 (96s ago)   5h42m
production         pri002                                     1/1     Running   3 (96s ago)   5h42m
production         pri003                                     1/1     Running   3 (96s ago)   5h42m
qaqa               qaqa-pod                                   1/1     Running   3 (96s ago)   5h36m
server             busybox-run-665877f69b-jtb9p               1/1     Running   3 (96s ago)   5h43m
server             nginx-host-d9bf9b9c-2hqpd                  1/1     Running   3 (96s ago)   5h43m
server             run-test-7c955665bb-b6wmh                  1/1     Running   3 (95s ago)   5h43m
tigera-operator    tigera-operator-5dc8b759d9-5w6q9           1/1     Running   8 (50s ago)   9d
candidate@node01:~$
```


2、Pod 指定 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 /cks/sa/pod1.yaml 中指定的 Pod 由于 ServiceAccount 指定错误而无法调度。

请完成一下项目:

Task

1. 在现有 namespace qa 中创建一个名为 backend-sa 的新 ServiceAccount,
确保此 ServiceAccount **不**自动挂载 API 凭据。
2. 使用 /cks/sa/pod1.yaml 中的清单文件来创建一个 Pod。
3. 最后，清理 namespace qa 中任何未使用的 ServiceAccount。

参考资料：

https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-service-account/#%E4%BD%BF%E7%94%A8%E9%BB%98%E8%AE%A4%E7%9A%84%E6%9C%8D%E5%8A%A1%E8%B4%A6%E6%88%B7%E8%AE%BF%E9%97%AE-api-%E6%9C%8D%E5%8A%A1%E5%99%A8

答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSCH00301

1 创建 ServiceAccount

vi qa-ns.yaml
根据官网修改如下内容

```
apiVersion: v1
kind: ServiceAccount
metadata:
   name: backend-sa     #修改 name
   namespace: qa        #注意添加 namespace
automountServiceAccountToken: false   #修改为 false，表示不自动挂载 secret
```

kubectl apply -f qa-ns.yaml
kubectl get sa -n qa

```
candidate@node01:~$ vi qa-ns.yaml
candidate@node01:~$ kubectl apply -f qa-ns.yaml
serviceaccount/backend-sa created
candidate@node01:~$ kubectl get sa -n qa
NAME         SECRETS   AGE
backend-sa   0         3s
default      0         6h10m
test01       0         6h10m
```

2 创建 Pod 使用该 ServiceAccount

vi /cks/sa/pod1.yaml

修改如下内容

```
……
metadata:
  name: backend
  namespace: qa     #注意命名空间是否对
spec:
  serviceAccountName: backend-sa     # 没有则添加一行，有则修改这一行为刚才创建的 ServiceAccount（考试时，默认已有这一行，需要修改。）
  containers:
……
```

kubectl apply -f /cks/sa/pod1.yaml
kubectl get pod -n qa

```
candidate@node01:~$ vi /cks/sa/pod1.yaml
candidate@node01:~$ kubectl apply -f /cks/sa/pod1.yaml
pod/backend created
candidate@node01:~$ kubectl get pod -n qa
NAME      READY   STATUS    RESTARTS   AGE
backend   1/1     Running   0          7s
candidate@node01:~$
```

3. 删除没有使用的 ServiceAccount

查看所有 sa
kubectl get sa -n qa

查看已经在用的 sa
kubectl get pod -n qa -o yaml | grep -i serviceAccountName

删除不用的 sa
kubectl delete sa test01 -n qa

```
candidate@node01:~$ kubectl get sa -n qa
NAME         SECRETS   AGE
backend-sa   0         5m6s
default      0         6h15m
test01       0         6h15m
candidate@node01:~$ kubectl get pod -n qa -o yaml | grep -i serviceAccountName
        {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"backend","namespace":"qa"},"spec":{"containers":[{"im
age":"nginx:1.9","imagePullPolicy":"IfNotPresent","name":"backend"}],"serviceAccountName":"backend-sa"}}
    serviceAccountName: backend-sa
    serviceAccountName: default
candidate@node01:~$ kubectl delete sa test01 -n qa
serviceaccount "test01" deleted
candidate@node01:~$ kubectl get sa -n qa
NAME         SECRETS   AGE
backend-sa   0         5m29s
default      0         6h15m
candidate@node01:~$
```

3、默认网络策略

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 denypolicy 的新默认拒绝 NetworkPolicy。
此新的 NetworkPolicy 必须拒绝 namespace testing 中的所有的 Ingress + Egress 流量。
将新创建的默认拒绝 NetworkPolicy 应用与在 namespace testing 中运行的所有 Pod。

你可以在 /cks/net/p1.yaml 找到一个模板清单文件。

参考资料：

https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/#%E9%BB%98%E8%AE%A4%E6%8B%92%E7%BB%9D%E6%89%80%E6%9C%89%E5%85%A5%E5%8F%A3%E5%92%8C%E6%89%80%E6%9C%89%E5%87%BA%E7%AB%99%E6%B5%81%E9%87%8F

答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSCS00101

创建名为 denypolicy 的 NetworkPolicy，拒绝 testing 命名空间内所有 Ingress + Egress 流量
（这里可能是 ingress 或 egress 或 ingress+egress，根据题目要求写。）
vi /cks/net/p1.yaml

修改为

……
metadata:
  name: denypolicy #修改 name
  namespace: testing #注意添加 namespace
spec:
  podSelector: {}
  policyTypes:
  - Ingress #注意看题，是 Ingress + Egress（入口+出口），还是只是 Ingress 或只是 Egress。
  - Egress #在 1.23 的考试中，只要求拒绝所有 Egress 流量，那就只写这这个- Egress 即可，这种情况就不要写- Ingress 了。

创建

kubectl apply -f /cks/net/p1.yaml

检查

kubectl describe networkpolicy denypolicy -n testing

```
student@node01:~$ vi /cks/net/p1.yaml
student@node01:~$ kubectl apply -f /cks/net/p1.yaml
networkpolicy.networking.k8s.io/denypolicy created
student@node01:~$ kubectl describe networkpolicy denypolicy -n testing
Name:         denypolicy
Namespace:    testing
Created on:   2022-02-28 21:03:08 +0800 CST
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    <none> (Selected pods are isolated for ingress connectivity)
  Allowing egress traffic:
    <none> (Selected pods are isolated for egress connectivity)
  Policy Types: Ingress, Egress
student@node01:~$
```

4、RBAC - RoleBinding

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

注意：请勿删除现有的 RoleBinding。

参考资料:

两个都要看

https://kubernetes.io/zh/docs/reference/access-authn-authz/rbac/#role-and-clusterole

https://kubernetes.io/zh/docs/reference/access-authn-authz/rbac/#%E4%B8%80%E4%BA%9B%E5%91%BD%E4%BB%A4%E8%A1%8C%E5%B7%A5%E5%85%B7

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

netes

Role 示例

下面是一个位于 "default" 名字空间的 Role 的示例，可用来授予对 pods 的读访问权限：

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" 标明 core API 组
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```


答题:

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSCH00201

查看 ServiceAccount　service-account-web 对应的 rolebindings role-1-binding
查看 rolebindings role-1-binding 对应的 role 为 role-1
kubectl describe rolebindings -n db

```
student@node01:~$ kubectl describe rolebindings -n db
Name:         role-1-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  role-1
Subjects:
  Kind            Name                Namespace
  ----            ----                ---------
  ServiceAccount  service-account-web  db
student@node01:~$
```

编辑 role-1 权限:
kubectl edit role role-1 -n db

修改如下内容

……
　resourceVersion: "30867"
　uid: 22e3c185-f7f5-4542-b86a-6ce153aa1c5a
rules:　#模拟环境里要删除掉 null，然后添加以下内容。考试时，要根据实际情况修改。
- apiGroups: [""]
　resources: ["services"]　　#只允许对 services 资源类型执行 get 操作。还有可能会考只允许对 endpoints 资源 list 的操作，要学会举一反三。
　verbs: ["get"]

检查
kubectl describe role role-1 -n db

```
student@node01:~$ kubectl edit role role-1 -n db
role.rbac.authorization.k8s.io/role-1 edited
student@node01:~$ kubectl describe role role-1 -n db
Name:         role-1
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  services   []                 []              [get]
student@node01:~$
```

在 db 命名空间，创建名为 role-2 的 role，并且通过 rolebinding 绑定 service-account-web，只允许对 namespaces 做 delete 操作。
记住 --verb 是权限，可能考 delete 或者 update 等 --resource 是对象，可能考 namespaces 或者 persistentvolumeclaims 等。
kubectl create role role-2 --verb=delete --resource=namespaces -n db
kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=db:service-account-web -n db

检查
kubectl describe rolebindings -n db

```
student@node01:~$ kubectl create role role-2 --verb=delete --resource=namespaces -n db
role.rbac.authorization.k8s.io/role-2 created
student@node01:~$ kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=db:service-account-web -n db
rolebinding.rbac.authorization.k8s.io/role-2-binding created
student@node01:~$
student@node01:~$ kubectl describe rolebindings -n db
Name:         role-1-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  role-1
Subjects:
  Kind            Name                Namespace
  ----            ----                ---------
  ServiceAccount  service-account-web  db


Name:         role-2-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  role-2
Subjects:
  Kind            Name                Namespace
  ----            ----                ---------
  ServiceAccount  service-account-web  db
student@node01:~$
```

5、日志审计 log audit

您**必须**在以下 cluster /节点上完成此考题：

Cluster	Master 节点	工作节点
KSRS00602	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 KSRS00602
```

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/audit-logs.txt
- 日志文件能保留 10 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。
注意：基本策略位于 cluster 的 master 节点上。

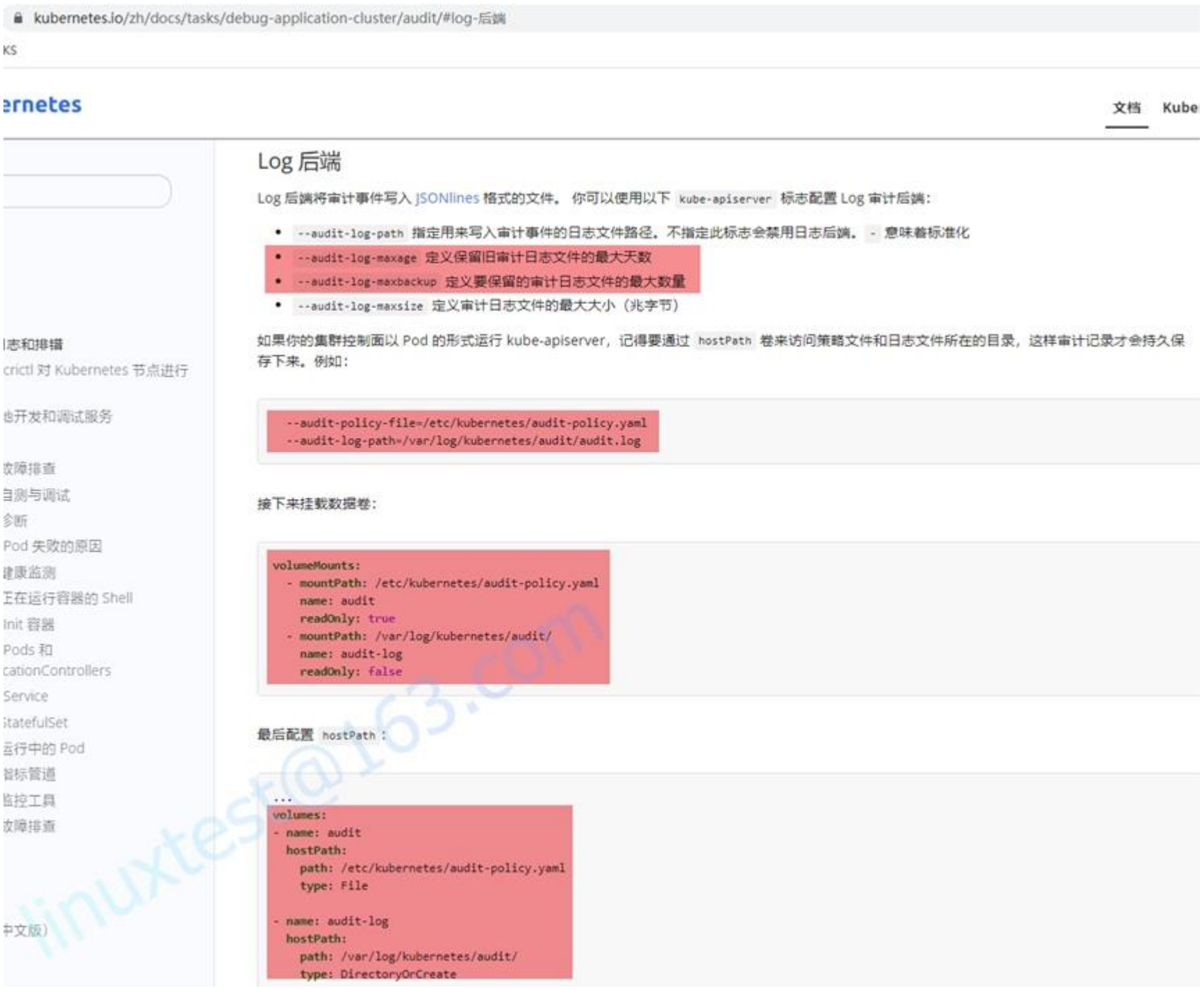
编辑和扩展基本策略以记录：

- RequestResponse 级别的 persistentvolumes 更改
- namespace front-apps 中 configmaps 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。
注意：不要忘记应用修改后的策略。

参考资料：

两个都要看
https://kubernetes.io/zh/docs/tasks/debug/debug-cluster/audit/



答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSCH00601

本题分数比较多，占 12%。

日志审计这一题需要自己调整的内容还是挺多的，因此要非常仔细，建议修改前备份一下原始的环境，要不然修改错了就会导致集群崩溃。

1 切换到 Master 的 root 下
ssh master01
sudo -i

```
candidate@node01:~$ ssh master01
```

```
candidate@master01:~$ sudo -i
root@master01:~#
```

2 配置审计策略
先备份配置文件
mkdir bak5
cp /etc/kubernetes/logpolicy/sample-policy.yaml bak5/

vim /etc/kubernetes/logpolicy/sample-policy.yaml
参考官方网址内容

……

```
rules:
  # namespaces changes at RequestResponse level
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["persistentvolumes"]     #根据题目要求修改，比如题目要求的是 namespaces。

  #the request body of persistentvolumes/pods changes in the namespace front-apps
  - level: Request
    resources:
    - group: ""
      resources: ["configmaps"]       #根据题目要求修改，比如题目要求的是 persistentvolumes 或者 pods。
    namespaces: ["front-apps"]

  # Log configmap and secret changes in all other namespaces at the Metadata level.
  - level: Metadata
    resources:
    - group: ""
      resources: ["secrets", "configmaps"]

  # Also, add a catch-all rule to log all other requests at the Metadata level.
  - level: Metadata
    omitStages:
      - "RequestReceived"
```

3 配置 master 节点的 kube-apiserver.yaml

先备份配置文件

```
mkdir bak5
cp /etc/kubernetes/manifests/kube-apiserver.yaml   bak5/
vi /etc/kubernetes/manifests/kube-apiserver.yaml
```

添加以下参数：注意空格要对齐

```
#定义审计策略 yaml 文件位置，通过 hostpath 挂载
    - --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml       #主意检查，如果考试中已经存在了，则不要重复添加。
#定义审计日志位置，通过 hostpath 挂载
    - --audit-log-path=/var/log/kubernetes/audit-logs.txt                #主意检查，如果考试中已经存在了，则不要重复添加。
#定义保留旧审计日志文件的最大天数为 10 天
    - --audit-log-maxage=10                #主意检查，如果考试中已经存在了，则不要重复添加。
#定义要保留的审计日志文件的最大数量为 2 个
    - --audit-log-maxbackup=2                #主意检查，如果考试中已经存在了，则不要重复添加。

#在 kube-apiserver.yaml 文件的 volumeMounts 标签下增加
    volumeMounts:     #找到这个字段，添加下面内容
    - mountPath: /etc/kubernetes/logpolicy/sample-policy.yaml       #这里也可以写到目录/etc/kubernetes/logpolicy/
      name: audit     #注意，在 1.23 考试中，蓝色的内容已经默认有了，你只需要添加绿色字体的内容。可以通过红色字，在文件中定位。但模拟环境没有加，需要你全部手动添加。这样是为了练习，万一考试中没有，你也会加。但是如果考试中已添加，你再添加一遍，则会报错，导致 api-server 启不起来。
      readOnly: true        #这个为 true
    - mountPath: /var/log/kubernetes/
      name: audit-log       #注意，在 1.23 考试中，蓝色的内容已经有了，你只需要添加绿色字体的内容。可以通过红色字，在文件中定位。但模拟环境没有加，需要你全部手动添加。这样是为了练习，万一考试中没有，你也会加。
      readOnly: false        #这个为 false

#在 kube-apiserver.yaml 文件的 volumes 标签下增加
  volumes:      #找到这个字段，添加下面内容     #注意，在 1.23 考试中，蓝色的内容已经有了，volumes 这段无需修改，但是为了以防万一，模拟环境中没有加，需要你手动添加。这样是为了练习，万一考试中没有，你也会加。
  - name: audit
    hostPath:
      path: /etc/kubernetes/logpolicy/sample-policy.yaml     #这里如果写到目录/etc/kubernetes/logpolicy/，则下面的 type:应为 type: DirectoryOrCreate
      type: File
  - name: audit-log
    hostPath:
      path: /var/log/kubernetes/
      type: DirectoryOrCreate
```

4 重启 kubelet 服务

```
systemctl restart kubelet
```

等待 2 分钟后，再检查

```
kubectl get pod -A
tail /var/log/kubernetes/audit-logs.txt

# 退出 root，退回到 candidate@master01
exit
# 退出 master01，退回到 candidate@node01
exit
```

```
root@master01:~# exit
logout
student@master01:~$ exit
logout
Connection to master01 closed.
student@node01:~$
```

配置的截图

```
root@master01:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
root@master01:~# vi /etc/kubernetes/manifests/kube-apiserver.yaml
root@master01:~# systemctl restart kubelet
root@master01:~# kubectl get pod -A
NAMESPACE     NAME                                   READY   STATUS    RESTARTS         AGE
db            web-pod                                1/1     Running   1 (7h5m ago)     7h59m
default       tomcat123                              1/1     Running   1 (7h5m ago)     7h20m
dev-team      products-service                       1/1     Running   1 (7h5m ago)     7h41m
kamino        tri111                                 1/1     Running   1 (7h5m ago)     7h26m
kamino        tri222                                 2/2     Running   2 (7h5m ago)     7h26m
kamino        tri333                                 2/2     Running   2 (7h5m ago)     7h26m
kube-system   coredns-6d8c4cb4d-p68v8                1/1     Running   13 (7h5m ago)    6d6h
kube-system   coredns-6d8c4cb4d-t5n94                1/1     Running   13 (33m ago)     6d6h
kube-system   etcd-master01                          1/1     Running   0                13m
kube-system   kube-apiserver-master01                1/1     Running   0                18s
kube-system   kube-controller-manager-master01       1/1     Running   13 (13m ago)     6d6h
kube-system   kube-flannel-ds-p7v8r                  1/1     Running   11 (7h5m ago)    6d6h
kube-system   kube-flannel-ds-qlcbv                  1/1     Running   11 (7h5m ago)    6d6h
kube-system   kube-flannel-ds-vmd6h                  1/1     Running   11 (7h5m ago)    6d6h
kube-system   kube-proxy-wgj54                       1/1     Running   11 (7h5m ago)    6d6h
kube-system   kube-proxy-wjgd7                       1/1     Running   11 (7h5m ago)    6d6h
kube-system   kube-proxy-zbdn9                       1/1     Running   11 (7h5m ago)    6d6h
kube-system   kube-scheduler-master01                1/1     Running   13 (13m ago)     6d6h
production    pri001                                 1/1     Running   1 (7h5m ago)     7h41m
production    pri002                                 1/1     Running   1 (7h5m ago)     7h41m
production    pri003                                 1/1     Running   1 (7h5m ago)     7h41m
qa            backend                                1/1     Running   0                9m52s
qa            qa-pod                                 1/1     Running   1 (7h5m ago)     7h40m
server        busybox                                1/1     Running   1 (7h5m ago)     7h48m
server        nginx-gvisor                           1/1     Running   1 (7h5m ago)     7h48m
server        nginx-host-64584cf4dc-7b4kv            1/1     Running   1 (7h5m ago)     7h48m
root@master01:~#
```

tail 日志的截图

```
root@master01:~# tail /var/log/kubernetes/audit-logs.txt
{"kind":"Event","apiVersion":"audit.k8s.io/v1","level":"Metadata","auditID":"518513b9-874a-4c71-b967-79a5021aebea","stage":"ResponseComplete","req
uestURI":"/apis/events.k8s.io/v1beta1","verb":"get","user":{"username":"system:serviceaccount:kube-system:generic-garbage-collector","uid":"725432
e4-1430-41f9-88bd-f5a71ec87381","groups":["system:serviceaccounts","system:serviceaccounts:kube-system","system:authenticated"]},"sourceIPs":["11.
0.1.111"],"userAgent":"kube-controller-manager/v1.23.1 (linux/amd64) kubernetes/86ec240/system:serviceaccount:kube-system:generic-garbage-collecto
r","responseStatus":{"metadata":{},"code":200},"requestReceivedTimestamp":"2022-02-28T13:13:05.987675Z","stageTimestamp":"2022-02-28T13:13:05.9877
74Z","annotations":{"authorization.k8s.io/decision":"allow","authorization.k8s.io/reason":"RBAC: allowed by ClusterRoleBinding \"system:discovery\
" of ClusterRole \"system:discovery\" to Group \"system:authenticated\""}}
{"kind":"Event","apiVersion":"audit.k8s.io/v1","level":"Metadata","auditID":"6e627c0d-6a95-4fc2-8ba2-44a0c492697e","stage":"ResponseComplete","req
uestURI":"/apis/coordination.k8s.io/v1","verb":"get","user":{"username":"system:serviceaccount:kube-system:generic-garbage-collector","uid":"72543
2e4-1430-41f9-88bd-f5a71ec87381","groups":["system:serviceaccounts","system:serviceaccounts:kube-system","system:authenticated"]},"sourceIPs":["11
.0.1.111"],"userAgent":"kube-controller-manager/v1.23.1 (linux/amd64) kubernetes/86ec240/system:serviceaccount:kube-system:generic-garbage-collect
or","responseStatus":{"metadata":{},"code":200},"requestReceivedTimestamp":"2022-02-28T13:13:05.988005Z","stageTimestamp":"2022-02-28T13:13:05.991
089Z","annotations":{"authorization.k8s.io/decision":"allow","authorization.k8s.io/reason":"RBAC: allowed by ClusterRoleBinding \"system:discovery
\" of ClusterRole \"system:discovery\" to Group \"system:authenticated\""}}
```

修改的配置文件的截图

```yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 11.0.1.111:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=11.0.1.111
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml
    - --audit-log-path=/var/log/kubernetes/audit-logs.txt
    - --audit-log-maxage=10
    - --audit-log-maxbackup=2
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
```

```yaml
      path: /livez
      port: 6443
      scheme: HTTPS
    initialDelaySeconds: 10
    periodSeconds: 10
    timeoutSeconds: 15
  volumeMounts:
  - mountPath: /etc/kubernetes/logpolicy/sample-policy.yaml
    name: audit
    readOnly: true
  - mountPath: /var/log/kubernetes/audit-logs.txt
    name: audit-log
    readOnly: false
  - mountPath: /etc/ssl/certs
    name: ca-certs
    readOnly: true
  - mountPath: /etc/ca-certificates
    name: etc-ca-certificates
    readOnly: true
  - mountPath: /etc/kubernetes/pki
    name: k8s-certs
    readOnly: true
  - mountPath: /usr/local/share/ca-certificates
    name: usr-local-share-ca-certificates
    readOnly: true
  - mountPath: /usr/share/ca-certificates
    name: usr-share-ca-certificates
    readOnly: true
hostNetwork: true
priorityClassName: system-node-critical
securityContext:
  seccompProfile:
    type: RuntimeDefault
volumes:
- name: audit
  hostPath:
    path: /etc/kubernetes/logpolicy/sample-policy.yaml
    type: File
- name: audit-log
  hostPath:
    path: /var/log/kubernetes/audit-logs.txt
    type: FileOrCreate
- hostPath:
    path: /etc/ssl/certs
    type: DirectoryOrCreate
  name: ca-certs
- hostPath:
    path: /etc/ca-certificates
    type: DirectoryOrCreate
  name: etc-ca-certificates
- hostPath:
```

（截图和文字不一样的，以文字为准）

type：
DirectoryOrCreate　宿主机上不存在创建此目录
Directory 必须存在挂载目录
FileOrCreate 宿主机上不存在挂载文件就创建
File 必须存在文件

vim 多行插入的方法：（具体可以百度一下）
vim 文件名

ctrl + v　上下箭头选中所要添加#的行
shift + I 或者大写 I　添加内容，比如按两个空格
按 esc，刚才选中的行，都会自动插入两个空格。

6、创建 Secret

Task
在 namespace istio-system 中获取名为 db1-test 的现有 secret 的内容

将 username 字段存储在名为 /cks/sec/user.txt 的文件中，并将 password 字段存储在名为 /cks/sec/pass.txt 的文件中。
注意：你必须创建以上两个文件，他们还不存在。

注意：不要在以下步骤中使用/修改先前创建的文件，如果需要，可以创建新的临时文件。

在 istio-system namespace 中创建一个名为 db2-test 的新 secret，内容如下：
username : production-instance
password : KvLftKgs4aVH

最后，创建一个新的 Pod，它可以通过卷访问 secret db2-test ：
Pod 名称 secret-pod
Namespace istio-system
容器名 dev-container
镜像 nginx
卷名 secret-volume
挂载路径 /etc/secret

参考资料：

三个网址都要看
https://kubernetes.io/zh/docs/tasks/configmap-secret/managing-secret-using-kubectl/#decoding-secret
https://kubernetes.io/zh/docs/tasks/configmap-secret/managing-secret-using-kubectl/#create-a-secret
https://kubernetes.io/zh/docs/concepts/configuration/secret/#using-secrets

https://kubernetes.io/zh/docs/tasks/configmap-secret/managing-secret-using-kubectl/

etes

文档 Kubernetes 博客

ctl 管理 Secret
文件管理 Secret
omize 管理 Secret

解码 Secret

要查看创建的 Secret 的内容，运行以下命令：

```
kubectl get secret db-user-pass -o jsonpath='{.data}'
```

输出类似于：

```
{"password":"MWYyZDF1MmU2N2Rm","username":"YWRtaW4="}
```

现在你可以解码 password 的数据：

```
echo 'MWYyZDF1MmU2N2Rm' | base64 --decode
```

输出类似于：

```
1f2d1e2e67df
```


https://kubernetes.io/zh/docs/tasks/configmap-secret/managing-secret-using-kubectl/#create-a-secret

etes

文档 Kubernetes 博客

ectl 管理 Secret
文件管理 Secret
omize 管理 Secret

你不需要对文件中包含的密码字符串中的特殊字符进行转义。

你还可以使用 `--from-literal=<key>=<value>` 标签提供 Secret 数据。可以多次使用此标签，提供多个键值对。请注意，特殊字符（例如：`$`，`\`，`*`，`=` 和 `!`）由你的 shell 解释执行，而且需要转义。

在大多数 shell 中，转义密码最简便的方法是用单引号括起来。比如，如果你的密码是 `S!B\*d$zDsb=`，可以像下面一样执行命令：

```
kubectl create secret generic db-user-pass \
  --from-literal=username=devuser \
  --from-literal=password='S!B\*d$zDsb='
```

验证 Secret

检查 secret 是否已创建：

```
kubectl get secrets
```

https://kubernetes.io/zh/docs/concepts/configuration/secret/#using-secrets

etes

文档 Kubernetes 博客 培训

使用 Secret

Secret 可以作为数据卷被挂载，或作为环境变量 暴露出来以供 Pod 中的容器使用。它们也可以被系统的其他部分使用，而不直接暴露在 Pod 内。 例如，它们可以保存凭据，系统的其他部分将用它来代表你与外部系统进行交互。

在 Pod 中使用 Secret 文件

在 Pod 中使用存放在卷中的 Secret：

1. 创建一个 Secret 或者使用已有的 Secret。多个 Pod 可以引用同一个 Secret。
2. 修改你的 Pod 定义，在 `spec.volumes[]` 下增加一个卷。可以给这个卷随意命名， 它的 `spec.volumes[].secret.secretName` 必须是 Secret 对象的名字。
3. 将 `spec.containers[].volumeMounts[]` 加到需要用到该 Secret 的容器中。 指定 `spec.containers[].volumeMounts[].readOnly = true` 和 `spec.containers[].volumeMounts[].mountPath` 为你想要该 Secret 出现的尚未使用的目录。
4. 修改你的镜像并且 / 或者命令行，让程序从该目录下寻找文件。 Secret 的 `data` 映射中的每一个键都对应 `mountPath` 下的一个文件名。

这是一个在 Pod 中使用存放在挂载卷中 Secret 的例子：

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: mysecret
```


答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSCH00701

1 将 db1-test 的 username 和 password，通过 base64 解码保存到题目指定文件：

方法 1：
kubectl get secrets db1-test -n istio-system -o jsonpath={.data}
会反馈结果为：{"password":"aGVsbG8=","username":"ZGlx"}
echo 'ZGlx'|base64 -d > /cks/sec/user.txt
echo 'aGVsbG8='|base64 -d > /cks/sec/pass.txt

方法 2：
kubectl get secrets -n istio-system db1-test -o jsonpath={.data.username} | base64 -d > /cks/sec/user.txt
kubectl get secrets -n istio-system db1-test -o jsonpath={.data.password} | base64 -d > /cks/sec/pass.txt

检查
cat /cks/sec/user.txt
cat /cks/sec/pass.txt

```
student@node01:~$ kubectl get secrets db1-test -n istio-system -o jsonpath={.data}
{"password":"aGVsbG8=","username":"ZGIx"}student@node01:~$
student@node01:~$
student@node01:~$ echo 'ZGIx'|base64 -d >  /cks/sec/user.txt
student@node01:~$ echo 'aGVsbG8='|base64 -d >  /cks/sec/pass.txt
student@node01:~$
student@node01:~$ cat /cks/sec/user.txt
db1student@node01:~$
student@node01:~$ cat /cks/sec/pass.txt
hellostudent@node01:~$
student@node01:~$
```

2 创建名为 db2-test 的 secret 使用题目要求的用户名和密码作为键值。注意要加命名空间。

注意，如果密码中有特殊字符（例如：$，\，*，= 和 !），需要加单引号来转义--from-literal=password='G!Y*d$zDsb'这样。

kubectl create secret generic db2-test -n istio-system --from-literal=username=production-instance --from-literal=password=KvLftKgs4aVH

检查

kubectl get secret -n istio-system

```
student@node01:~$ kubectl create secret generic db2-test -n istio-system --from-literal=username=production-instance --from-literal=password=KvLft
Kgs4aVH
secret/db2-test created
student@node01:~$ kubectl get secret -n istio-system
NAME                  TYPE                                  DATA   AGE
db1-test              Opaque                                2      8h
db2-test              Opaque                                2      9s
default-token-zjvkg   kubernetes.io/service-account-token   3      8h
student@node01:~$
```

3 根据题目要求，参考官网，创建 Pod 使用该 secret

vim k8s-secret.yaml

添加如下内容

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-pod     #pod 名字
  namespace: istio-system     #命名空间
spec:
  containers:
  - name: dev-container     #容器名字
    image: nginx       #镜像名字
    volumeMounts:     #挂载路径
    - name: secret-volume    #卷名
      mountPath: /etc/secret
  volumes:
  - name: secret-volume    #卷名
    secret:
      secretName: db2-test     #名为 db2-test 的 secret
```

创建

kubectl apply -f k8s-secret.yaml

```
student@node01:~$ vim k8s-secret.yaml
student@node01:~$ kubectl apply -f k8s-secret.yaml
```

检查

kubectl get pod -n istio-system

```
student@node01:~$ kubectl get pod -n istio-system
NAME         READY   STATUS              RESTARTS   AGE
secret-pod   0/1     ContainerCreating   0          16s
student@node01:~$ kubectl get pod -n istio-system 下载镜像需要时间
NAME         READY   STATUS    RESTARTS   AGE
secret-pod   1/1     Running   0          33s
```

7、Dockerfile 检测

Task

分析和编辑给定的 Dockerfile /cks/docker/Dockerfile（基于 ubuntu:16.04 镜像），
并修复在文件中拥有的突出的安全/最佳实践问题的**两个指令**。

分析和编辑给定的清单文件 /cks/docker/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**。

注意：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。

注意：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 。

只修改即可，不需要创建。

参考资料:

https://kubernetes.io/zh/docs/concepts/security/pod-security-standards/#restricted

权能（v1.22+）

容器组必须弃用 ALL 权能，并且只允许添加 NET_BIND_SERVICE 权能。

限制的字段

- spec.containers[*].securityContext.capabilities.drop
- spec.initContainers[*].securityContext.capabilities.drop
- spec.ephemeralContainers[*].securityContext.capabilities.drop

允许的值

- 包含 ALL 的任何一种权能列表。

限制的字段

- spec.containers[*].securityContext.capabilities.add
- spec.initContainers[*].securityContext.capabilities.add
- spec.ephemeralContainers[*].securityContext.capabilities.add

允许的值

- 未定义/nil
- NET_BIND_SERVICE

答题:

考试时执行，切换集群。模拟环境中不需要执行。

kubectl config use-context KSSC00301

注意，本次的 Dockerfile 和 deployment.yaml 仅修改即可，无需部署。

<1> 修改 Dockerfile

vi /cks/docker/Dockerfile

1、仅将 CMD 上面的 USER root 修改为 USER nobody，不要改其他的 USER root。

USER nobody

2、修改基础镜像为题目要求的 ubuntu:16.04

FROM ubuntu:16.04

修改后

```
FROM ubuntu:16.04
RUN apt-get install -y wget curl gc
          iproute net-tools telnet
         yum clean all && \
         rm -rf /var/cache/apt/*
    cd / && rm -rf nginx* && \
   ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
USER nobody
CMD ["./sunnydale.sh"]
ENV PATH $PATH:/usr/local/nginx/sbin
COPY nginx.conf /usr/local/nginx/conf/nginx.conf
```

<2> 修改 deployment.yaml

vi /cks/docker/deployment.yaml

1、template 里标签跟上面的内容不一致，所以需要将原先的 run: couchdb 修改为 app: couchdb

app: couchdb

（注意，这里具体要改成 app: couchdb 还是其他的标签，要以考试给你的 yaml 文件的上下文其他标签为准，要与另外两个标签一致，具体见下方截图。）

感谢网友 Tse 和 adams 的反馈和纠正。

2、删除 'SYS_ADMIN' 字段，确保 'privileged': 为 False 。（CKS 考试是有多套类似考试环境的，所以有时是删 SYS_ADMIN，有时是改'privileged': False）

注意 注意，如果考试时，本来就没有'SYS_ADMIN' 字段，且'privileged':也默认就为 False，则直接将下面绿色的这两行注释掉，就是前面加#。

securityContext:

{'Capabilities': {'add': ['NET_BIND_SERVICE'], 'drop': ['all']}, 'privileged': False, 'readonlyRootFilesystem': False, 'runAsUser': 65535}

修改后

```
kind: Deployment
metadata:
  name: couchdb
  namespace: default
  labels:
    app: couchdb
    version: stable
spec:
  replicas: 1
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: couchdb
      version: stable
  template:
    metadata:
      labels:
        app: couchdb
    spec:
      containers:
      - name: couchdb
        image: demo:v1
```

```
        volumeMounts:
          - name: database-storage
            mountPath: /var/lib/database
        securityContext:
          {'Capabilities': {'add': ['NET_BIND_SERVICE'], 'drop': ['all']}, 'privileged': False, 'readonlyRootFilesystem': False, 'runAsUser': 65535}
        resources:
          limits:
            cpu: 300m
```

8、沙箱运行容器 gVisor

您**必须**在以下 cluster /节点上完成此考题:

Cluster	Master 节点	工作节点
KSMV00301	ksmv00301-master	ksmv00301-worker1

您可以使用以下命令来切换 cluster / 配置环境:

```
[candidate@cli] $ kubectl config use-context KSMV00301
```

Context

该 cluster 使用 containerd 作为 CRI 运行时。containerd 的默认运行时处理程序是 runc。
containerd 已准备好支持额外的运行时处理程序 runsc (gVisor)。

Task

使用名为 runsc 的现有运行时处理程序，创建一个名为 untrusted 的 RuntimeClass。
更新 namespace server 中的所有 Pod 以在 gVisor 上运行。

您可以在 /cks/gVisor/rc.yaml 中找到一个模版清单。

参考资料:

https://kubernetes.io/zh/docs/concepts/containers/runtime-class/#2-%E5%88%9B%E5%BB%BA%E7%9B%B8%E5%BA%94%E7%9A%84-runtimeclass-%E8%B5%84%E6%BA%90

https://kubernetes.io/zh/docs/concepts/containers/runtime-class/#2-创建相应的-runtimeclass-资源

netes

文档 Kubernetes 博

2. 创建相应的 RuntimeClass 资源

在上面步骤 1 中，每个配置都需要有一个用于标识配置的 `handler`。针对每个 handler 需要创建一个 RuntimeClass 对象。

RuntimeClass 资源当前只有两个重要的字段：RuntimeClass 名（`metadata.name`）和 handler（`handler`）。对象定义如下所示：

```yaml
apiVersion: node.k8s.io/v1  # RuntimeClass 定义于 node.k8s.io API 组
kind: RuntimeClass
metadata:
  name: myclass  # 用来引用 RuntimeClass 的名字
  # RuntimeClass 是一个集群层面的资源
handler: myconfiguration  # 对应的 CRI 配置的名称
```

时类（Runtime

周期回调

版）

说明： 建议将 RuntimeClass 写操作（create、update、patch 和 delete）限定于集群管理员使用。通常这是默认配置。参阅授权概述了解更多信息。

使用说明

一旦完成集群中 RuntimeClasses 的配置，使用起来非常方便。在 Pod spec 中指定 `runtimeClassName` 即可。例如:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  runtimeClassName: myclass
  # ...
```

答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSMV00301

1 创建 RuntimeClass
vi /cks/gVisor/rc.yaml
添加如下内容
……
metadata:
  name: untrusted # 用来引用 RuntimeClass 的名字，RuntimeClass 是一个集群层面的资源
handler: runsc # 对应的 CRI 配置的名称

创建
kubectl apply -f /cks/gVisor/rc.yaml
检查
kubectl get RuntimeClass

```
student@node01:~$ vi /cks/gVisor/rc.yaml
student@node01:~$ kubectl apply -f /cks/gVisor/rc.yaml
Warning: node.k8s.io/v1beta1 RuntimeClass is deprecated in v1.22+, unavailable in v1.25+
runtimeclass.node.k8s.io/untrusted created
student@node01:~$ kubectl get RuntimeClass
NAME        HANDLER   AGE
untrusted   runsc     11s
student@node01:~$
```

2 将命名空间为 server 下的 Pod 引用 RuntimeClass。
考试时，3 个 Deployment 下有 3 个 Pod，修改 3 个 deployment 即可。

kubectl -n server get deployment

```
root@node01:~# kubectl -n server get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
busybox-run   1/1     1            1           2m26s
nginx-host    1/1     1            1           2m26s
run-test      1/1     1            1           2m26s
root@node01:~#
```

编辑 deployment
kubectl -n server edit deployments busybox-run
kubectl -n server edit deployments nginx-host
kubectl -n server edit deployments run-test

修改如下内容
spec: #找到这个 spec，注意在 deployment 里是有两个单独行的 spec 的，要找第二个，也就是下面有 containers 这个字段的 spec。
  runtimeClassName: untrusted #添加这一行，注意空格对齐，保存会报错，忽略即可。
  containers:
  - image: nginx:1.9
    imagePullPolicy: IfNotPresent
    name: run-test

```
  metadata:
    creationTimestamp: null
    labels:
      app: run-test
  spec:
    runtimeClassName: untrusted
    containers:
    - image: nginx:1.9
      imagePullPolicy: IfNotPresent
```

因为模拟环境没有安装 runsc 运行环境，所以改完后，新 Pod 是 ContainerCreating 状态。考试时，改完后 Pod 会重建并 Running 的。

```
root@node01:~# kubectl -n server get pod
NAME                          READY   STATUS              RESTARTS   AGE
busybox-run-6488b64f54-sk9bx  1/1     Running             0          10m
busybox-run-85b6d4b66f-6k6dn  0/1     ContainerCreating   0          2m44s
nginx-host-64584cf4dc-t2c7g   1/1     Running             0          10m
nginx-host-686b6cc656-7jbnk   0/1     ContainerCreating   0          6s
run-test-66c6cf99f5-7sgxk     0/1     ContainerCreating   0          76s
run-test-6b99854856-52d99     1/1     Running             0          10m
root@node01:~#
```

9、容器安全，删除特权 Pod

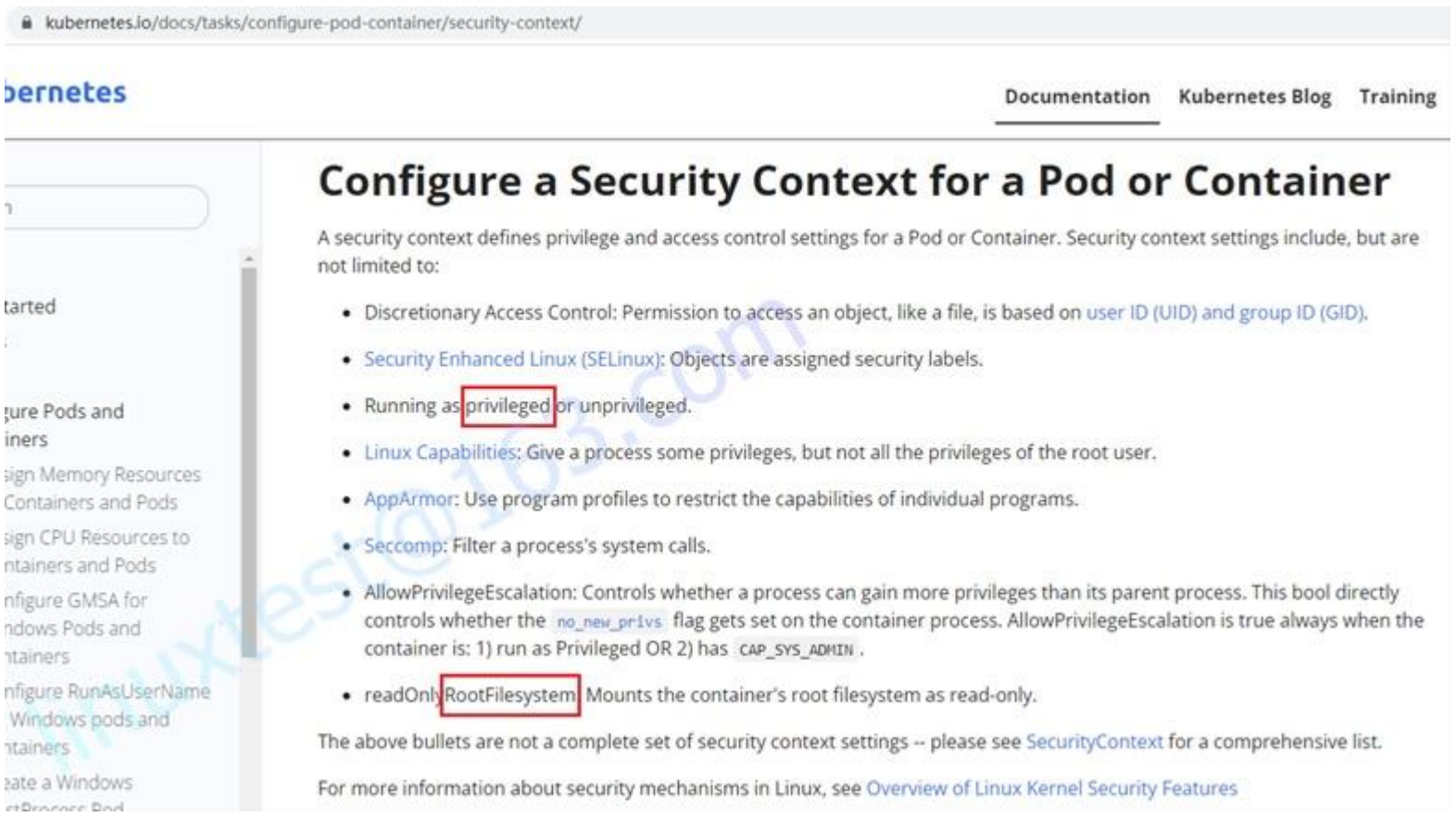
Context
最佳实践是将容器设计为无状态和不可变的。

Task
检查在 namespace production 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 的容器必须被视为**非无状态**的。
  注意：你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 Pod 必须被视为可能是**非无状态**和**非不可变**的。

参考资料:
https://kubernetes.io/docs/tasks/configure-pod-container/security-context/



答题:

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSRS00501

查看此命名空间下的所有 pod，删除有特权 Privileged 或者挂载 volume 的 pod
kubectl get pod -n production
kubectl get pods XXXX -n production -o yaml | grep -i "privileged: true"　　#注意冒号后面有一个空格，XXXX 换成 production 命名空间下的 pod 名。

```
student@node01:~$ kubectl get pod -n production
NAME     READY   STATUS    RESTARTS        AGE
pri001   1/1     Running   1 (7h58m ago)   8h
pri002   1/1     Running   1 (7h58m ago)   8h
pri003   1/1     Running   1 (7h58m ago)   8h
student@node01:~$ kubectl get pods pri001 -n production -o yaml | grep -i "privileged: true"
      privileged: true
student@node01:~$ kubectl get pods pri002 -n production -o yaml | grep -i "privileged: true"
student@node01:~$ kubectl get pods pri003 -n production -o yaml | grep -i "privileged: true"
student@node01:~$
```

将上面查出来的有特权的 pod 删除
kubectl delete pod XXXX -n production

```
student@node01:~$ kubectl delete pod pri001 -n production
pod "pri001" deleted
student@node01:~$
```

上面只是将特权的 pod 查出来了，根据题目要求，还应该查挂载 volume 的 pod，命令为
kubectl get pods XXXX -n production -o jsonpath={.spec.volumes} | jq
模拟环境里的 3 个 pod，都没有挂载 volume，所以无需删除了。考试中也是这样的，都没有挂载 volume。

```
student@node01:~$ kubectl get pods pri002 -n production -o jsonpath={.spec.volumes} | jq
[
  {
    "name": "kube-api-access-xx8wp",
    "projected": {
      "defaultMode": 420,
      "sources": [
        {
          "serviceAccountToken": {
            "expirationSeconds": 3607,
            "path": "token"
          }
        },
        {
          "configMap": {
            "items": [
              {
                "key": "ca.crt",
                "path": "ca.crt"
              }
            ],
            "name": "kube-root-ca.crt"
          }
        },
        {
          "downwardAPI": {
            "items": [
              {
                "fieldRef": {
                  "apiVersion": "v1",
                  "fieldPath": "metadata.namespace"
                },
                "path": "namespace"
              }
            ]
          }
        }
      ]
    }
  }
]
student@node01:~$
```

没有挂载volume

什么样的是挂载了 volume 的 pod 呢?
比如 kube-system 命名空间下的 etcd-master01 是挂载了 volume 的。
具体是否挂载，可以依据是否有 hostPath 或路径

```
student@node01:~$ kubectl get pods etcd-master01 -n kube-system -o jsonpath={.spec.volumes} | jq
[
  {
    "hostPath": {
      "path": "/etc/kubernetes/pki/etcd",
      "type": "DirectoryOrCreate"
    },
    "name": "etcd-certs"
  },
  {
    "hostPath": {
      "path": "/var/lib/etcd",
      "type": "DirectoryOrCreate"
    },
    "name": "etcd-data"
  }
]
student@node01:~$
```

挂载volume

10、网络策略 NetworkPolicy

您必须在以下 cluster /节点上完成此考题：

Cluster	Master 节点	工作节点
KSSH00301	kssh00301-master	kssh00301-worker1

您可以使用以下命令来切换 cluster / 配置环境：

```
[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

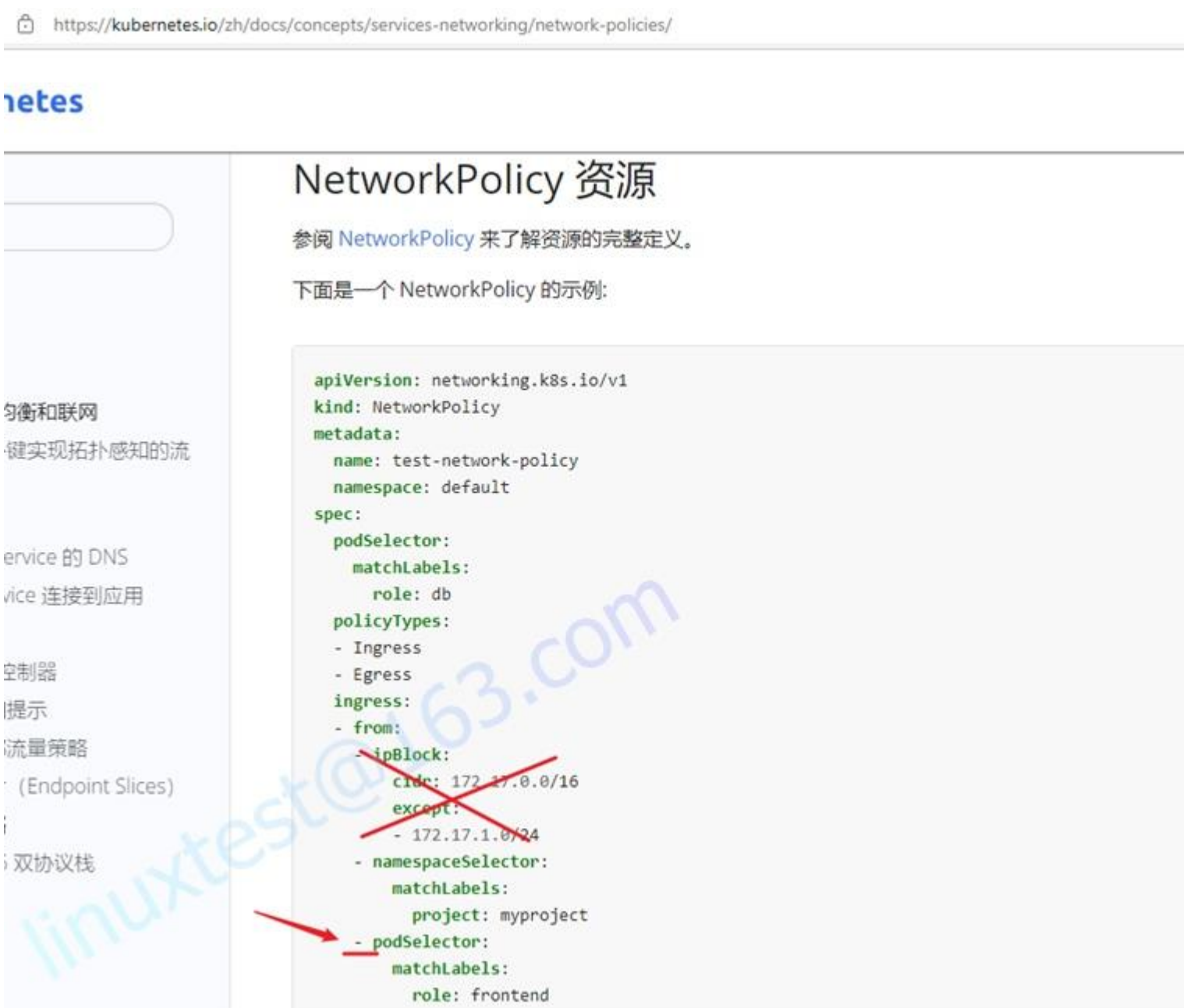
- namespace qaqa 中的 Pod
- 位于任何 namespace，带有标签 environment: testing 的 Pod

注意：确保应用 NetworkPolicy。

你可以在/cks/net/po.yaml 找到一个模板清单文件。

参考资料：

https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/#networkpolicy-resource



答题:

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSSH00301

1 检查 namespace 标签

模拟环境已提前打好标签了，所以你只需要检查标签即可。但为了防止考试时，没有给你打标签，所以还是需要你将下面打标签的命令记住。
查看 qaqa 命名空间标签（name: qaqa）
kubectl get ns --show-labels

```
candidate@node01:~$ kubectl get ns --show-labels
NAME              STATUS   AGE     LABELS
calico-apiserver  Active   9d      kubernetes.io/metadata.name=calico-apiserver,name=calico-apiserver
calico-system     Active   9d      kubernetes.io/metadata.name=calico-system,name=calico-system
db                Active   6h13m   kubernetes.io/metadata.name=db
default           Active   10d     kubernetes.io/metadata.name=default
dev-team          Active   5h52m   kubernetes.io/metadata.name=dev-team
istio-system      Active   6h10m   kubernetes.io/metadata.name=istio-system
kamino            Active   5h45m   kubernetes.io/metadata.name=kamino
kube-node-lease   Active   10d     kubernetes.io/metadata.name=kube-node-lease
kube-public       Active   10d     kubernetes.io/metadata.name=kube-public
kube-system       Active   10d     kubernetes.io/metadata.name=kube-system
production        Active   5h54m   kubernetes.io/metadata.name=production
qa                Active   6h16m   kubernetes.io/metadata.name=qa
qaqa              Active   5h47m   kubernetes.io/metadata.name=qaqa,name=qaqa
server            Active   6h9m    kubernetes.io/metadata.name=server
staging           Active   6h14m   kubernetes.io/metadata.name=staging
testing           Active   6h14m   kubernetes.io/metadata.name=testing
testns            Active   5h31m   kubernetes.io/metadata.name=testns
tigera-operator   Active   9d      kubernetes.io/metadata.name=tigera-operator,name=tigera-operator
candidate@node01:~$
```

查看 pod 标签（environment: testing）
kubectl get pod -n dev-team --show-labels

```
candidate@node01:~$ kubectl get pod -n dev-team --show-labels
NAME               READY   STATUS    RESTARTS      AGE     LABELS
products-service   1/1     Running   3 (13m ago)   5h48m   environment=testing
candidate@node01:~$
```

如果 Pod 或者 Namespace 没有标签，则需要打上标签。
注意：我这里将 pod products-service 的标签打成了 environment: testing，下面会有解释，不要和题目里要求的“位于任何 namespace，带有标签 environment: testing 的 Pod”这句话里的标签混淆了。
kubectl label ns qaqa name=qaqa
kubectl label pod products-service environment=testing -n dev-team

2 创建 NetworkPolicy

vi /cks/net/po.yaml
根据官网，修改为如下内容：

```
……
metadata:
  name: pod-restriction     #修改
  namespace: dev-team       #修改
spec:
  podSelector:
    matchLabels:
      environment: testing      #根据题目要求的标签修改，这个写的是 Pod  products-service 的标签，也就是使用 kubectl get pod -n dev-team --show-labels 查出来的 pod 的标签，这个标签不要和题目里要求的“位于任何 namespace，带有标签 environment: testing 的 Pod”这句话里的标签混淆了，两个没有关系，所以可不一样。比如你考试时查出来的 POD  products-service 的标签是 name: products，那这里的 environment: testing 就要换成 name: products。
  policyTypes:
  - Ingress      #注意，这里只写  - Ingress，不要将  - Egress 也复制进来!
  ingress:
  - from:      #第一个 from
    - namespaceSelector:
        matchLabels:
```

```
            name: qaqa       #命名空间有 name: qaqa 标签的
  - from:       #第二个 from
    - namespaceSelector: {}   #修改为这样，所有命名空间
      podSelector:     #注意，这个 podSelector 前面的“-”要删除，换成空格，空格对齐要对。
        matchLabels:
          environment: testing     #有 environment: testing 标签的 Pod，这个地方是根据题目要求“Pods with label environment: testing , in any namespace”，这句话里的 pod 标签写的。不要和上面 spec 里的混淆。
```

创建

kubectl apply -f /cks/net/po.yaml

检查

kubectl get networkpolicy -n dev-team

```
candidate@node01:~$ vi /cks/net/po.yaml
candidate@node01:~$ kubectl apply -f /cks/net/po.yaml
networkpolicy.networking.k8s.io/pod-restriction created
candidate@node01:~$ kubectl get networkpolicy -n dev-team
NAME              POD-SELECTOR          AGE
pod-restriction   environment=testing   4s
candidate@node01:~$
```

11、Trivy 扫描镜像安全漏洞

您**必须**在以下 cluster /节点上完成此考题:

Cluster	Master 节点	工作节点
KSSC00401	kssc00401-master	kssc00401-worker1

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。

注意：Trivy 仅安装在 cluster 的 master 节点上，
在工作节点上不可使用。
你必须切换到 cluster 的 master 节点才能使用 Trivy。

参考资料:

https://kubernetes.io/zh/docs/reference/kubectl/cheatsheet/#%E6%A0%BC%E5%BC%8F%E5%8C%96%E8%BE%93%E5%87%BA

kubernetes.io/zh/docs/reference/kubectl/cheatsheet/#格式化输出

s

ernetes

文档 Kubernete

```
# 集群中运行着的所有镜像
kubectl get pods -A -o=custom-columns='DATA:spec.containers[*].image'

# 列举 default 名字空间中运行的所有镜像，按 Pod 分组
kubectl get pods --namespace default --output=custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*].image"

# 除 "k8s.gcr.io/coredns:1.6.2" 之外的所有镜像
kubectl get pods -A -o=custom-columns='DATA:spec.containers[?(@.image!="k8s.gcr.io/coredns:1.6.2")].image'

# 输出 metadata 下面的所有字段，无论 Pod 名字为何
kubectl get pods -A -o=custom-columns='DATA:metadata.*'
```


答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSSC00401

注意：这个题非常耗费时间，考试时关注一下剩余时间。如果时间剩余不多，可以放到最后做这道题。

方法 1：（推荐，能快一些）

1 切换到 Master 的 candidate 下
ssh master01

```
candidate@node01:~$ ssh master01
```

2 获取命名空间 kamino 下的所有 pod 的 image
kubectl get pods --namespace kamino --output=custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*].image"

```
student@master01:~$ kubectl get pods --namespace kamino --output=custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*].image"
NAME     IMAGE
tri111   amazonlinux:1
tri222   amazonlinux:2,nginx:1.19
tri333   vicuu/nginx:host,amazonlinux:2
student@master01:~$
```

3 检查镜像是否有高危和严重的漏洞
trivy image -s HIGH,CRITICAL nginx:1.19 # HIGH,CRITICAL，这里的 nginx:1.19 换成你上一步查出来的镜像名字
或者也可以使用这条命令查询 trivy image nginx:1.19 | grep -iE 'High|Critical'

注意 tri222 和 tri333 的 2 个 pod 里各有 2 个 image，都需要扫描。
trivy image -s HIGH,CRITICAL amazonlinux:1
trivy image -s HIGH,CRITICAL amazonlinux:2
trivy image -s HIGH,CRITICAL nginx:1.19
trivy image -s HIGH,CRITICAL vicuu/nginx:host

```
student@master01:~# trivy image -s HIGH,CRITICAL amazonlinux:1
2022-02-24T11:22:59.569+0800    INFO    Detected OS: amazon
2022-02-24T11:22:59.569+0800    INFO    Detecting Amazon Linux vulnerabilities...
2022-02-24T11:22:59.575+0800    INFO    Number of language-specific files: 0

amazonlinux:1 (amazon AMI release 2018.03)
==========================================
Total: 0 (HIGH: 0, CRITICAL: 0)
```

```
student@master01:~$ trivy image -s HIGH,CRITICAL nginx:1.19
2021-12-01T17:57:33.567+0800    INFO    Need to update DB
2021-12-01T17:57:33.567+0800    INFO    Downloading DB...
25.03 MiB / 25.03 MiB [------------------------------------------------------------------------------------------------------
2021-12-01T18:02:16.960+0800    INFO    Detected OS: debian
2021-12-01T18:02:16.960+0800    INFO    Detecting Debian vulnerabilities...
2021-12-01T18:02:16.979+0800    INFO    Number of language-specific files: 0

nginx:1.19 (debian 10.9)
========================
Total: 63 (HIGH: 42, CRITICAL: 21)

+------------------+------------------+----------+-------------------+-------------------+------------------------------------------+
|     LIBRARY      | VULNERABILITY ID | SEVERITY | INSTALLED VERSION |   FIXED VERSION   |                  TITLE                   |
+------------------+------------------+----------+-------------------+-------------------+------------------------------------------+
| curl             | CVE-2021-22946   | HIGH     | 7.64.0-4+deb10u2  |                   | curl: Requirement to use                 |
|                  |                  |          |                   |                   | TLS not properly enforced                |
|                  |                  |          |                   |                   | for IMAP, POP3, and...                   |
|                  |                  |          |                   |                   | -->avd.aquasec.com/nvd/cve-2021-22946    |
+------------------+------------------+          +-------------------+-------------------+------------------------------------------+
| gcc-8-base       | CVE-2018-12886   |          | 8.3.0-6           |                   | gcc: spilling of stack                   |
|                  |                  |          |                   |                   | protection address in cfgexpand.c        |
|                  |                  |          |                   |                   | and function.c leads to...               |
|                  |                  |          |                   |                   | -->avd.aquasec.com/nvd/cve-2018-12886    |
+                  +------------------+          +                   +-------------------+------------------------------------------+
|                  | CVE-2019-15847   |          |                   |                   | gcc: POWER9 "DARN" RNG intrinsic         |
|                  |                  |          |                   |                   | produces repeated output                 |
|                  |                  |          |                   |                   | -->avd.aquasec.com/nvd/cve-2019-15847    |
+------------------+------------------+----------+-------------------+-------------------+------------------------------------------+
| libc-bin         | CVE-2021-33574   | CRITICAL | 2.28-10           |                   | glibc: mq_notify does                    |
|                  |                  |          |                   |                   | not handle separately                    |
|                  |                  |          |                   |                   | allocated thread attributes              |
|                  |                  |          |                   |                   | -->avd.aquasec.com/nvd/cve-2021-33574    |
+                  +------------------+          +                   +-------------------+------------------------------------------+
|                  | CVE-2021-35942   |          |                   |                   | glibc: Arbitrary read in wordexp()       |
|                  |                  |          |                   |                   | -->avd.aquasec.com/nvd/cve-2021-35942    |
```

4 删除有问题的 pod

kubectl delete pod XXXX -n kamino

5 退出 master01，退回到 candidate@node01

exit

请注意，考试时有 5 个 pod，每个 pod 里有多个 image 镜像，都需要扫描。扫描出有漏洞的镜像，则删除有这个镜像的 pod。

另外，网友 WilliamLee 还提供了一种方法，对 shell 命令比较熟习的，可以参考。
让它后台扫，然后退出来继续做其他的，做完其他的回来看报告，删 pod。
注意别忘了每道题要切换集群。

```
root@master01:~# kubectl -n kamino describe pod|grep Image:
    Image:         amazonlinux:1
    Image:         amazonlinux:2
    Image:          nginx:1.19
    Image:          vicuu/nginx:host
    Image:         amazonlinux:2
root@master01:~# trivy image amazonlinux:1 |grep -iE 'High|Critical' >> amazonlinux:1.txt && trivy image amazonlinux:2 |grep
 -iE 'High|Critical' >> amazonlinux:2.txt && trivy image nginx:1.19 |grep -iE 'High|Critical' >> nginx:1.19.txt && trivy ima
ge vicuu/nginx:host |grep -iE 'High|Critical' >> vicuu/nginx:host.txt &
```

方法 2：（太消耗时间，因为考试时，有 5 个 pod，而且每一个 pod 里面还有两三个 image）

切换到 Master 的 root 下
ssh master01

检查所有 pod
kubectl get pod -n kamino
获取每一个 pod 的 image
kubect get pod XXXX -n kamino -o yaml | grep image

检查镜像是否有高危和严重的漏洞
trivy image -s HIGH,CRITICAL nginx:1.19 # HIGH,CRITICAL，这里的 nginx:1.19 换成你上一步查出来的镜像名字
或者也可以使用下面命令查询
trivy image nginx:1.19 | grep -iE 'High|Critical'

删除有问题的 pod
kubectl delete pod XXXX -n kamino

退出 master01，退回到 candidate@node01
exit

请注意，考试时有 5 个 pod，每个 pod 里有多个 image 镜像。检查出有漏洞的镜像，则删除有这个镜像的 pod。

方法 3：（写 while 脚本有点麻烦）

（对于 pod 数量多的情况，可以使用下面方法）

```
#  切换到 Master 的 root 下
ssh master01

检查所有 pod
kubectl get po -n kamino
kubectl get po -n kamino | grep -v "NAME" | awk '{print $1}' > podlist.txt
cat podlist.txt

循环检查 pod 里的 image
while read line;do
echo $line
kubectl get pod -n kamino $line -o yaml | grep "image:"
done < podlist.txt

对查出来的 image 逐一扫描
trivy image -s HIGH,CRITICAL nginx:1.19       #注意 HIGH,CRITICAL 为大写

删除扫描的镜像带有高危或严重漏洞的  Pod
kubectl delete pod XXXX -n kamino
```

```
student@master01:~$ kubectl delete pod tri222 -n kamino
pod "tri222" deleted
student@master01:~$ kubectl delete pod tri333 -n kamino
pod "tri333" deleted
student@master01:~$
```

```
#  退出 master01，退回到 candidate@node01
exit
```

12、AppArmor

您**必须**在以下 cluster /节点上完成此考题：

Cluster	Master 节点	工作节点
KSSH00401	kssh00401-master	kssh00401-worker1

您可以使用以下命令来切换 cluster / 配置环境：

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

Context

APPArmor 已在 cluster 的工作节点 node02 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 node02 上，实施位于 /etc/apparmor.d/nginx_apparmor 的现有 APPArmor 配置文件。
编辑位于 /cks/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 。

请注意，考试时，考题里已表明 APPArmor 在工作节点上，所以你需要 ssh 到开头写的工作节点上。
在模拟环境，你需要 ssh 到 node02 这个工作节点。

参考资料：

https://kubernetes.io/zh/docs/tutorials/security/apparmor/#example

答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSSH00401

1 切换到 node02 的 root 下
ssh node02
sudo -i

```
student@node01:~$ ssh node02
student@node02:~$ sudo -i
root@node02:~#
```

2 切换到 apparmor 的目录，并检查配置文件
cd /etc/apparmor.d/

vi /etc/apparmor.d/nginx_apparmor
注意，nginx-profile-3 这一行要注释掉，但要确保 profile nginx-profile-3 这一行没有注释。
#include <tunables/global>
nginx-profile-3 #注释掉 模拟环境里多写了一行的，会导致 apparmor_parser -q 的时候会报错。1.21 里，这个文件是有一个这样的操作的，但在新的 1.23 的考试中，这个/etc/apparmor.d/nginx_apparmor 又变成正确的了。所以你在考试时，可以先 cat /etc/apparmor.d/nginx_apparmor，有错误则改，没错误，则不要再改了。
profile nginx-profile-3 flags=(attach_disconnected) { #这句是正确的配置，不要修改。profile 后面的 nginx-profile-3 为 apparmor 策略模块的名字。
  #include <abstractions/base>
  file,
……

```
root@node02:~# cd /etc/apparmor.d/
root@node02:/etc/apparmor.d# vi /etc/apparmor.d/nginx_apparmor
root@node02:/etc/apparmor.d#
```

3 执行 apparmor 策略模块
没有 grep 到，说明没有启动。
apparmor_status | grep nginx-profile-3
加载启用这个配置文件
apparmor_parser -q /etc/apparmor.d/nginx_apparmor
再次检查有结果了
apparmor_status | grep nginx
显示如下内容
  nginx-profile-3

```
root@node02:/etc/apparmor.d# apparmor_status | grep nginx-profile-3
root@node02:/etc/apparmor.d# apparmor_parser -q /etc/apparmor.d/nginx_apparmor
root@node02:/etc/apparmor.d# apparmor_status | grep nginx-profile-3
   nginx-profile-3
root@node02:/etc/apparmor.d#
```

4 修改 pod 文件
vi /cks/KSSH00401/nginx-deploy.yaml
修改如下内容
……
metadata:
  name: podx
  #添加 annotations，kubernetes.io/podx 名字和 containers 下的名字一样即可，nginx-profile-3 为前面在 worker node01 上执行的 apparmor 策略模块的名字。
  annotations:
    container.apparmor.security.beta.kubernetes.io/podx: localhost/nginx-profile-3
spec:
  containers:
  - image: busybox
    imagePullPolicy: IfNotPresent
    name: podx #这个就是 containers 下的名字，为 podx
    command: ["sh", "-c", "echo 'Hello AppArmor!' && sleep 1h"]
……

创建
kubectl apply -f /cks/KSSH00401/nginx-deploy.yaml

```
root@node02:/etc/apparmor.d# vi /home/candidate/KSSH00401/nginx-deploy.yaml
root@node02:/etc/apparmor.d# kubectl apply -f /home/candidate/KSSH00401/nginx-deploy.yaml
pod/podx created
root@node02:/etc/apparmor.d#
```

退出 root，退回到 candidate@node02
exit
退出 node02，退回到 candidate@node01
exit

```
root@node02:~# exit
logout
student@node02:~$ exit
logout
Connection to node02 closed.
student@node01:~$
```

检查（可不做）

kubectl get pod

```
student@node01:~$ kubectl get pod
NAME        READY   STATUS    RESTARTS     AGE
podx        1/1     Running   0            99s
tomcat123   1/1     Running   1 (8h ago)   8h
student@node01:~$
```

可以通过检查该配置文件的 proc attr 来验证容器是否实际使用该配置文件运行:

kubectl exec podx -- cat /proc/1/attr/current

```
student@node01:~$ kubectl exec podx -- cat /proc/1/attr/current
nginx-profile-3 (enforce)
student@node01:~$
```

（考试中，这一步不用做）因为我们测试环境里的 nginx-profile-3 策略是拒绝写入，所以如果试图通过写入文件来违反配置文件，策略就会生效:

kubectl exec podx -- touch /tmp/test

```
student@node01:~$ kubectl exec podx -- touch /tmp/test
touch: /tmp/test: Permission denied
command terminated with exit code 1
student@node01:~$
```

13、Sysdig & falco

这道题在 1.23 后，又开始考了。

您必须在以下 cluster /节点上完成此考题:

Cluster	Master 节点	工作节点
KSSC00401	kssc00401-master	kssc00401-worker1

您可以使用以下命令来切换 cluster / 配置环境:

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

Task:

使用运行时检测工具来检测 Pod tomcat123 单个容器中频发生成和执行的异常进程。

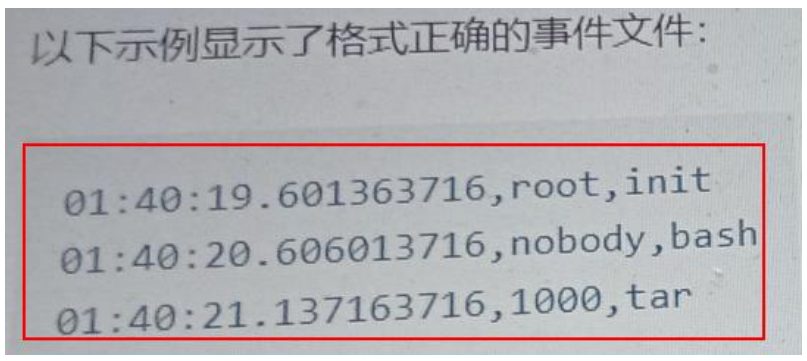
有两种工具可供使用:

- sysdig
- falco

注: 这些工具**只**预装在 cluster 的工作节点 node02 上，不在 master 节点。

使用工具至少分析 30 秒 ，使用过滤器检查生成和执行的进程，将事件写到 /opt/KSR00101/incidents/summary 文件中，其中包含检测的事件， 格式如下:

timestamp,uid/username,processName



保持工具的原始时间戳格式不变。
注： 确保事件文件存储在集群的工作节点上。

请注意，考试时，考题里已表明 sysdig 在工作节点上，所以你需要 ssh 到开头写的工作节点上。但在模拟环境，你需要 ssh 到 node02 这个工作节点。

参考资料:

无

答题:

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSSC00401

感谢网友 ljqwyf 的反馈，之前的考试环境的 K8S 集群使用的容器全是 docker，新考试环境，部分使用 docker，部分使用 containerd。
这也证明了 CKS 确实有多套考试环境的。
所以此题给出了两个方法，方法 1 是 containerd 的，方法 2 是 docker 的。
考试时，你可以先 docker ps 看一下，如果没有这个 pod，则应该是使用的 containerd。

考试形式 1 ： K8S 集群使用 containerd

1 切换到 node02 的 root 下

ssh node02
sudo -i

```
student@node01:~$ ssh node02
student@node02:~$ sudo -i
root@node02:~#
```

2、先找到 containerd 的 socket
crictl info | grep sock

```
root@node02:~# crictl info | grep sock
WARN[0000] runtime connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:/
//run/crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint in
stead.
ERRO[0000] unable to determine runtime API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Erro
r while dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
    "containerdEndpoint": "/run/containerd/containerd.sock",
root@node02:~#
```

3、crontainerd 要使用 crictl 命令找到容器，题目要求的是 tomcat123，则 grep tomcat123。

crictl ps | grep tomcat123

```
root@node02:~# crictl ps | grep tomcat123
WARN[0000] runtime connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:/
//run/crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint in
stead.
ERRO[0000] unable to determine runtime API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Erro
r while dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
WARN[0000] image connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:///
run/crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint inst
ead.
ERRO[0000] unable to determine image API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Error
while dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
7cb82fdbc1295       07eee1a00569f       6 minutes ago       Running             tomcat123           4                   8cdb594fd4
c52       tomcat123
root@node02:~#
```

4、

通过 sysdig 扫描容器 30s 并输出到指定文件：

sysdig -h 和-l 查看帮助

注：可以使用 sysdig -l |grep time 过滤，确认输出格式字段

sysdig -l | grep time

sysdig -l | grep uid

sysdig -l | grep proc

开始扫描 （我目前想不到别的方法，只能将命令分成 2 条了，谁有更好的方法，可以分享一下。）

sysdig -M 30 -p "%evt.time,%user.uid,%proc.name" --cri /run/containerd/containerd.sock container.name=tomcat123 >> /opt/KSR00101/incidents/summary

sysdig -M 30 -p "%evt.time,%user.name,%proc.name" --cri /run/containerd/containerd.sock container.name=tomcat123 >> /opt/KSR00101/incidents/summary

提示：如果考试时执行 sysdig 报错“Unable to load the driver”，则执行下面一条命令：（模拟环境里不需要执行）

#启用模块

sysdig-probe-loader

然后再次执行 sysdig -M 30 ……

如果还是报错，就重装一下 sysdig，命令为 apt install sysdig

查看保存的文件

cat /opt/KSR00101/incidents/summary |head

```
root@node02:~# cat /opt/KSR00101/incidents/summary |head
12:59:22.285334000,1,container:7cb82fdbc129
12:59:22.421647337,0,VM
12:59:22.421659099,0,VM
12:59:22.421660021,0,VM
12:59:22.421669208,0,VM
12:59:22.421675329,0,VM
12:59:22.472802567,0,VM
12:59:22.472815401,0,VM
12:59:22.472816874,0,VM
12:59:22.472825050,0,VM
root@node02:~#
```

退出 root，退回到 candidate@node02

exit

退出 node02，退回到 candidate@node01

exit

```
root@node02:~# exit
logout
student@node02:~$ exit
logout
Connection to node02 closed.
student@node01:~$
```

考试形式 2 ：K8S 集群使用 docker（1.24 基本不考这种了）

以下命令记住即可，模拟环境使用的是 containerd。

1 切换到 node02 的 root 下

ssh node02

sudo -i

```
student@node01:~$ ssh node02
student@node02:~$ sudo -i
root@node02:~#
```

首先找到容器的 container id:，我这里的容器 id 为 40c3c3e8b813

docker ps | grep tomcat123

```
root@node02:~# docker ps | grep tomcat
40c3c3e8b813   fb5657adc892                                                  "catalina.sh run"   2 hours ago         Up 2 hours                  k8s_t
omcat123_tomcat123_default_53df3bfe-0597-48dc-b519-61d8bb63e5af_1
81bbf21d6dc4   registry.aliyuncs.com/google_containers/pause:3.6             "/pause"            2 hours ago         Up 2 hours                  k8s_P
OD_tomcat123_default_53df3bfe-0597-48dc-b519-61d8bb63e5af_3
root@node02:~#
```

找那个不带/pause的

通过 sysdig 扫描容器 30s 并输出到指定文件:

sysdig -h 和-l 查看帮助

注：可以使用 sysdig -l |grep time 过滤，确认输出格式字段

sysdig -l | grep time

sysdig -l | grep uid

sysdig -l | grep proc

开始扫描

sysdig -M 30 -p "%evt.time,%user.uid,%proc.name" container.id=40c3c3e8b813 > /opt/KSR00101/incidents/summary

```
root@node02:~# sysdig -M 30 -p "%evt.time,%user.uid,%proc.name" container.id=40c3c3e8b813> /opt/KSR00101/incidents/summary
root@node02:~#
```

查看保存的文件

cat /opt/KSR00101/incidents/summary |head

```
root@node02:~# cat /opt/KSR00101/incidents/summary |head
13:04:37.975673000,1,container:3965a9ff91e8
13:04:37.978541000,1,container:3965a9ff91e8
13:04:37.981849000,1,container:3965a9ff91e8
13:04:37.986447000,1,container:3965a9ff91e8
13:04:37.989239000,1,container:3965a9ff91e8
13:04:37.993179000,1,container:3965a9ff91e8
13:04:37.995560000,1,container:3965a9ff91e8
13:04:37.999315000,1,container:3965a9ff91e8
root@node02:~#
```

退出 root，退回到 candidate@node02

exit

退出 node02，退回到 candidate@node01

exit

```
root@node02:~# exit
logout
student@node02:~$ exit
logout
Connection to node02 closed.
student@node01:~$
```

14、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
KSMV00102	ksmv00102-master	ksmv00102-worker1

您可以使用以下命令来切换 cluster / 配置环境:

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

Context

PodSecurityPolicy 应防在特定 namespace 中特权 Pod 的创建。

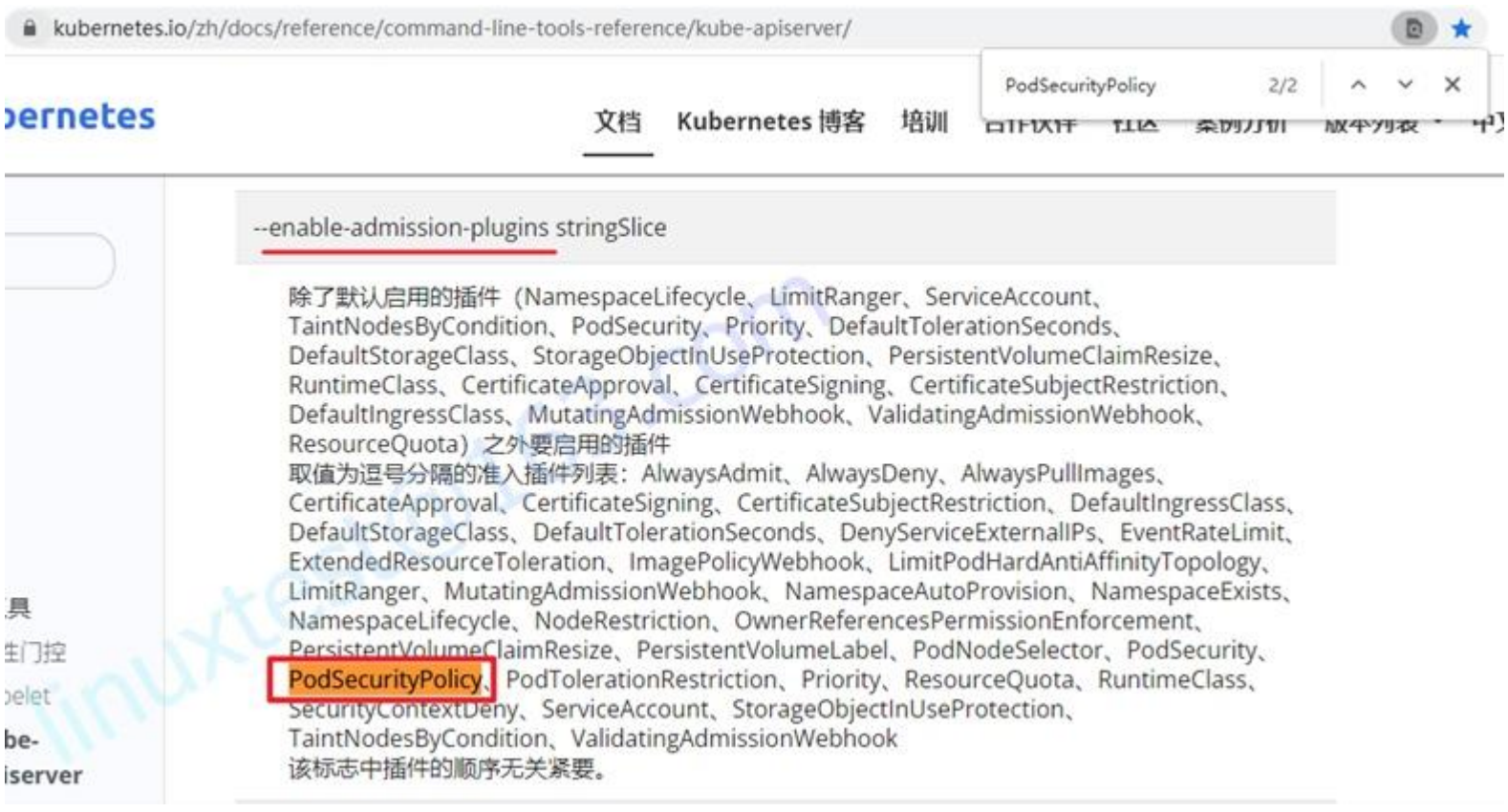
Task

创建一个名为 restrict-policy 的新的 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 PodSecurityPolicy restrict-policy 的 ClusterRole。
在现有的 namespace staging 中创建一个名为 psp-denial-sa 的新 ServiceAccount
。
最后，创建一个名为 dany-access-bind 的 ClusterRoleBinding ，
将新创建的 ClusterRole restrict-access-role 绑定到新创建的 ServiceAccount psp-denial-sa。

你可以在一下位置找到模版清单文件:
/cks/psp/psp.yaml

参考资料:

四个都要看
https://kubernetes.io/zh/docs/reference/command-line-tools-reference/kube-apiserver/#%E9%80%89%E9%A1%B9



https://kubernetes.io/docs/concepts/security/pod-security-policy/#create-a-policy-and-a-pod

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

etes

文档 Kubernetes 博客 培训 合作伙伴 社区

创建一个策略和一个 Pod

在一个文件中定义一个示例的 PodSecurityPolicy 对象。 这里的策略只是用来禁止创建有特权要求的 Pods。 PodSecurityPolicy 对象的名称必须是合法的 DNS 子域名。

policy/example-psp.yaml

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example
spec:
  privileged: false  # Don't allow privileged pods!
  # The rest fills in some required fields.
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
  - '*'
```

https://kubernetes.io/docs/concepts/security/pod-security-policy/#via-rbac

如果使用命令不会创建 clusterrole 和 clusterrolebinding，则可以参考官网的 yaml 文件，修改并使用 kubectl apply -f 来创建。

kubernetes.io/zh/docs/concepts/policy/pod-security-policy/#via-rbac

KA CKA高线 CKS

bernetes

通过 RBAC 授权

RBAC 是一种标准的 Kubernetes 鉴权模式，可以很容易地用来授权策略访问。

首先，某 `Role` 或 `ClusterRole` 需要获得使用 `use` 访问目标策略的权限。 访问授权的规则看起来像这样：

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <Role 名称>
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs:     ['use']
  resourceNames:
  - <要授权的策略列表>
```

接下来将该 `Role` （或 `ClusterRole`）绑定到授权的用户：

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: <binding name>
roleRef:
  kind: ClusterRole
  name: <role name>
  apiGroup: rbac.authorization.k8s.io
subjects:
# 授权命名空间下的所有服务账号（推荐）：
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts:<authorized namespace>
# 授权特定的服务账号（不建议这样操作）：
- kind: ServiceAccount
  name: <authorized service account name>
  namespace: <authorized pod namespace>
```

https://kubernetes.io/docs/concepts/security/pod-security-policy/#example

使用命令创建 clusterrole 和 clusterrolebinding 的参考方法

https://kubernetes.io/zh/docs/concepts/policy/pod-security-policy/#example

es

文档 Kubernetes 博客 培训

示例

本示例假定你已经有一个启动了 PodSecurityPolicy 准入控制器的集群并且 你拥有集群管理员特权。

配置

为运行此示例，配置一个名字空间和一个服务账号。我们将用这个服务账号来 模拟一个非管理员账号的用户。

看题目要求创建的是clusterrolebinding 还是rolebinding

```
kubectl create namespace psp-example
kubectl create serviceaccount -n psp-example fake-user
kubectl create rolebinding -n psp-example fake-editor --clusterrole=edit --serviceaccount=psp-example:fake-user
```

答题：

考试时执行，切换集群。模拟环境中不需要执行。

kubectl config use-context KSMV00102

1 切换到 Master 的 root 下

ssh master01

sudo -i

```
candidate@node01:~$ ssh master01
```

```
candidate@master01:~$ sudo -i
root@master01:~#
```

2 检查确认 apiserver 支持 PodSecurityPolicy

启用 PSP 准入控制器（考试中默认已经启用，但以防万一，还是要检查一下的。）

cat /etc/kubernetes/manifests/kube-apiserver.yaml | grep PodSecurityPolicy

确保有下面这行，在 1.23 的考试中，这一行已经提前给你加上了，但还是需要检查确认一下。

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

如果 kube-apiserver.yam 配置文件需要修改

则先备份配置文件，再添加这行。

mkdir bak14

cp /etc/kubernetes/manifests/kube-apiserver.yaml bak14

vi /etc/kubernetes/manifests/kube-apiserver.yaml

修改或添加如下内容

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

编辑完后重新加载配置文件，并重启 kubelet

systemctl daemon-reload

systemctl restart kubelet.service

```
root@master01:~# cat /etc/kubernetes/manifests/kube-apiserver.yaml | grep PodSecurityPolicy
root@master01:~# vi /etc/kubernetes/manifests/kube-apiserver.yaml
root@master01:~# cat /etc/kubernetes/manifests/kube-apiserver.yaml | grep PodSecurityPolicy
    - --enable-admission-plugins=NodeRestriction,PodSecurityPolicy
root@master01:~# systemctl daemon-reload
root@master01:~# systemctl restart kubelet.service
root@master01:~#
```

退出 root，退回到 candidate@master01

exit

退出 master01，退回到 candidate@node01

exit

```
root@master01:~# exit
logout
student@master01:~$ exit
logout
Connection to master01 closed.
student@node01:~$
```

3 创建 PSP

官方网址里复制 PSP 的内容，并修改拒绝特权。(考试中会给出参考的 yaml 文件，可以修改这个文件。)

创建名为 prevent-psp-policy 的 PodSecurityPolicy，阻止创建 Privileged Pod。

vi /cks/psp/psp.yaml　　#如果这个文件考试时，在 node01 找不到，就回上面的 master 里找。

修改如下内容

……

```
metadata:
  name: restrict-policy    #检查一下 name，如果不对则修改。
spec:
  privileged: false  # 修改为 false，阻止创建 Privileged Pod（特权 pod）
  seLinux:     #下面这些信息，根据官网拷贝。
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
  - '*'
```

创建

kubectl apply -f /cks/psp/psp.yaml

检查

kubectl get podsecuritypolicy

```
student@node01:~$ vi /cks/psp/psp.yaml
student@node01:~$ kubectl apply -f /cks/psp/psp.yaml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/restrict-policy created
student@node01:~$ kubectl get podsecuritypolicy
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
NAME                       PRIV    CAPS               SELINUX    RUNASUSER   FSGROUP    SUPGROUP   READONLYROOTFS   VOLUMES
psp.flannel.unprivileged   false   NET_ADMIN,NET_RAW   RunAsAny   RunAsAny    RunAsAny   RunAsAny   false            configMap,secret,emptyDir,hostPath
restrict-policy            false                      RunAsAny   RunAsAny    RunAsAny   RunAsAny   false            *
student@node01:~$
```

4 创建 CluserRole 和 ServiceAccount，并通过 ClusterRoleBinding 绑定。

方法 1：

使用命令创建

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

```
student@node01:~$ kubectl create clusterrole restrict-access-role --verb=use --resource=psp --resource-name=restrict-policy
clusterrole.rbac.authorization.k8s.io/restrict-access-role created
student@node01:~$ kubectl create sa psp-denial-sa -n staging
serviceaccount/psp-denial-sa created
student@node01:~$ kubectl create clusterrolebinding dany-access-bind --clusterrole=restrict-access-role --serviceaccount=staging:psp-denial-sa
clusterrolebinding.rbac.authorization.k8s.io/dany-access-bind created
student@node01:~$
```

方法 2：

（博主考试时，使用的是方法 2，因为考题里已经给出了一个 yaml 模板的配置清单，修改一番就可以使用了。）

使用 yaml 文件创建

1.23 考试中，已经给你 3 个空的 yaml 文件了，所以可以使用方法 2 来做，先修改 yaml 文件，再 apply。

参考官网，通过 yaml 文件来创建，CluserRole 和 ServiceAccount，以及 ClusterRoleBinding 绑定

https://kubernetes.io/zh/docs/concepts/policy/pod-security-policy/#via-rbac

psi

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

检查

kubectl describe clusterrolebinding dany-access-bind

```
student@node01:~$ kubectl describe clusterrolebinding dany-access-bind
Name:         dany-access-bind
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  restrict-access-role
Subjects:
  Kind            Name           Namespace
  ----            ----           ---------
  ServiceAccount  psp-denial-sa  staging
student@node01:~$
```

15、启用 API server 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
KSCH00101	ksch00101-master	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

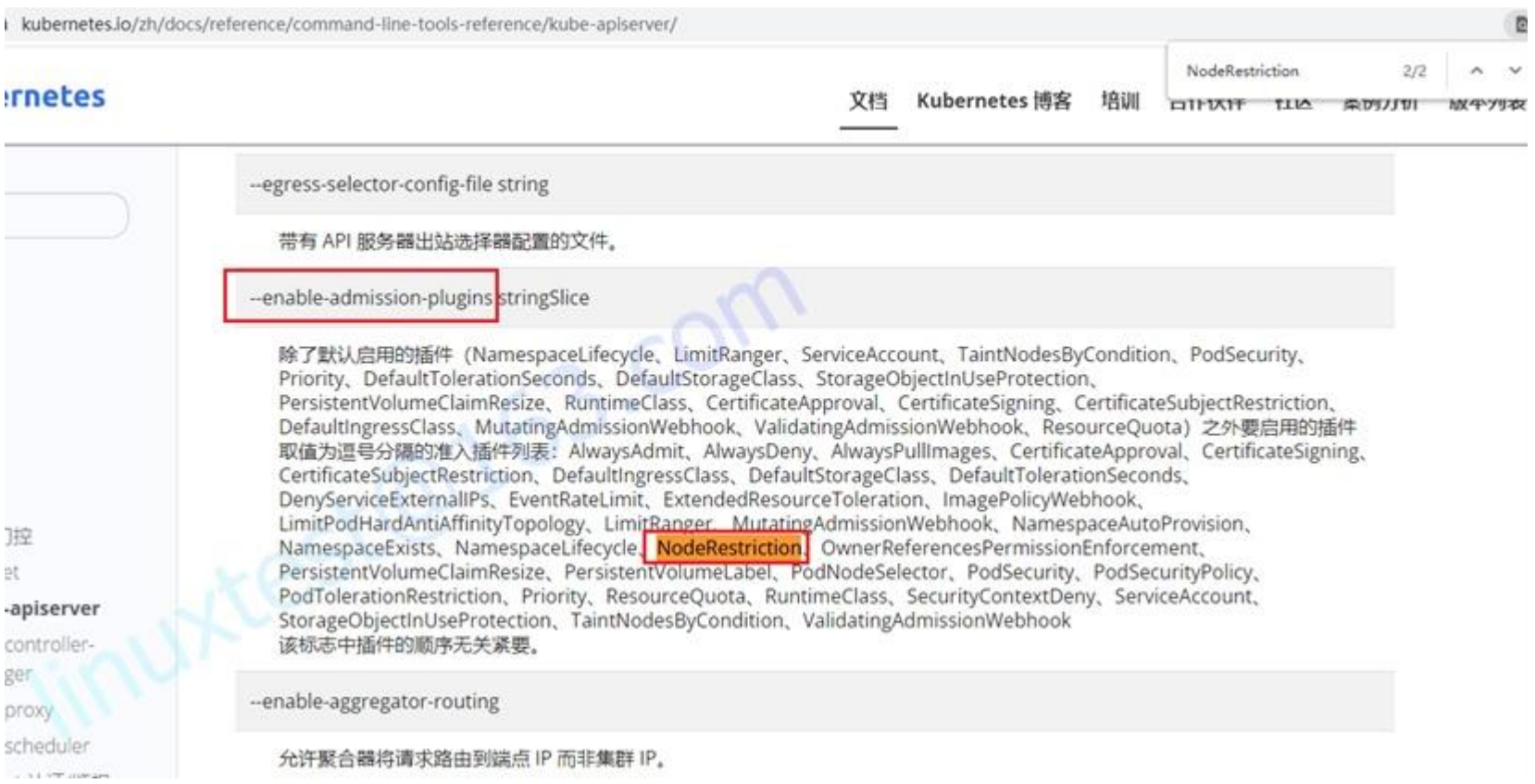
重新配置 cluster 的 Kubernetes API 服务器，以确保只允许经过身份验证和授权的 REST 请求。
使用授权模式 Node,RBAC 和准入控制器 NodeRestriction。
删除用户 system:anonymous 的 ClusterRoleBinding 来进行清理。

注意：所有 kubectl 配置环境/文件也被配置使用未经身份验证和未经授权的访问。
你不必更改它，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的授权的请求仍然被允许。

模拟环境里，初始化这道题的脚本为 b.sh

参考资料：

https://kubernetes.io/zh/docs/reference/command-line-tools-reference/kube-apiserver/



答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSCF00301

1 切换到 Master 的 root 下
ssh master01
sudo -i

```
candidate@node01:~$ ssh master01
```

```
candidate@master01:~$ sudo -i
root@master01:~#
```

请先执行如下命令，模拟这道题的初始环境。
脚本在 master01 的/root 目录下。
sh b.sh

```
root@master01:~# sh b.sh
root@master01:~#
```

2 确保只有认证并且授权过的 REST 请求才被允许
编辑/etc/kubernetes/manifests/kube-apiserver.yaml，修改下面内容
- --authorization-mode=AlwaysAllow
- --enable-admission-plugins=AlwaysAdmit

vi /etc/kubernetes/manifests/kube-apiserver.yaml
修改为
- --authorization-mode=Node,RBAC　　#注意，只保留 Node,RBAC 这两个，中间是英文状态下的逗号。在 1.23 考试中，这一条可能默认已经有了，但还是要检查确认一下。
- --enable-admission-plugins=NodeRestriction　　#在 1.23 考试中，这一个原先为 AlwaysAdmit，需要修改为 NodeRestriction。
- --client-ca-file=/etc/kubernetes/pki/ca.crt　　#这一条也要检查一下，是否存在配置文件中，如果没有，也需要添加。在 1.23 考试中，这一条默认已经有了。
- --enable-bootstrap-token-auth=true　　#这一条也要检查一下，是否存在配置文件中，如果没有，也需要添加。在 1.23 考试中，这一条默认已经有了。

感谢网友 adams 的反馈和纠正

重启 kubelet
systemctl daemon-reload
systemctl restart kubelet

```
root@master01:~# vi /etc/kubernetes/manifests/kube-apiserver.yaml
root@master01:~# systemctl restart kubelet
root@master01:~#
```

修改完成后，等待 3 分钟，集群才会恢复正常。
kubectl get pod -A

3 删除题目要求的角色绑定
查
kubectl get clusterrolebinding system:anonymous

删
kubectl delete clusterrolebinding system:anonymous
再检查
kubectl get clusterrolebinding system:anonymous

```
root@master01:~# kubectl get clusterrolebinding system:anonymous
NAME               ROLE                      AGE
system:anonymous   ClusterRole/test-role   9h
root@master01:~# kubectl delete clusterrolebinding system:anonymous
clusterrolebinding.rbac.authorization.k8s.io "system:anonymous" deleted
root@master01:~#
root@master01:~# kubectl get clusterrolebinding system:anonymous
Error from server (NotFound): clusterrolebindings.rbac.authorization.k8s.io "system:anonymous" not found
root@master01:~#
```

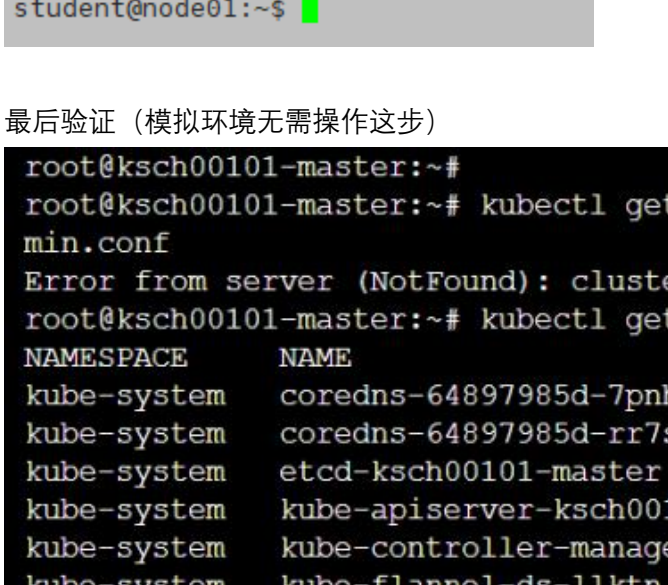
退出 root，退回到 candidate@master01
exit
退出 master01，退回到 candidate@node01
exit

```
root@master01:~# exit
logout
student@master01:~$ exit
logout
Connection to master01 closed.
student@node01:~$
```

最后验证（模拟环境无需操作这步）

```
root@ksch00101-master:~#
root@ksch00101-master:~# kubectl get  ClusterRoleBinding system:anonymous  --kubeconfig=/etc/kubernetes/ad
min.conf
Error from server (NotFound): clusterrolebindings.rbac.authorization.k8s.io "system:anonymous" not found
root@ksch00101-master:~# kubectl get  pod -A  --kubeconfig=/etc/kubernetes/admin.conf
NAMESPACE     NAME                                          READY   STATUS    RESTARTS         AGE
kube-system   coredns-64897985d-7pnhm                       1/1     Running   1 (7h38m ago)    43d
kube-system   coredns-64897985d-rr7sd                       1/1     Running   1 (7h38m ago)    43d
kube-system   etcd-ksch00101-master                         1/1     Running   1 (7h38m ago)    43d
kube-system   kube-apiserver-ksch00101-master               1/1     Running   0                75s
kube-system   kube-controller-manager-ksch00101-master      1/1     Running   7 (2m46s ago)    43d
kube-system   kube-flannel-ds-llktn                         1/1     Running   1 (43d ago)      43d
kube-system   kube-flannel-ds-q9vnl                         1/1     Running   1 (43d ago)      43d
kube-system   kube-proxy-2c4ht                              1/1     Running   1 (43d ago)      43d
kube-system   kube-proxy-pmmbc                              1/1     Running   1 (43d ago)      43d
kube-system   kube-scheduler-ksch00101-master               1/1     Running   7 (2m46s ago)    43d
root@ksch00101-master:~# less /etc/kubernetes/manifests/kube-apiserver.yaml
root@ksch00101-master:~#
```


16、ImagePolicyWebhook 容器镜像扫描



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
注意：你必须在 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/epconfig 中不完整的配置，
以及具有 HTTPS 端点 https://image-bouncer-webhook.default.svc:1323/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 /cks/img/web1.yaml 来测试配置是否有效。

感谢网友 epreache* 对这道题的技术支持。

参考资料：

三个网址都要看
https://kubernetes.io/zh/docs/reference/access-authn-authz/admission-controllers/#%E5%A6%82%E4%BD%95%E5%90%AF%E7%94%A8%E4%B8%80%E4%B8%AA%E5%87%86%E5%85%A5%E6%8E%A7%E5%88%B6%E5%99%A8

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

https://kubernetes.io/zh/docs/tasks/debug/debug-cluster/audit/#log-%E5%90%8E%E7%AB%AF

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

netes

文档 Kubernetes 博客 培

ImagePolicyWebhook

ImagePolicyWebhook 准入控制器允许使用一个后端的 webhook 做出准入决策。

配置文件格式

ImagePolicyWebhook 使用配置文件来为后端行为设置配置选项。该文件可以是 JSON 或 YAML， 并具有以下格式:

```yaml
imagePolicy:
  kubeConfigFile: /path/to/kubeconfig/for/backend
  # 以秒计的时长，控制批准请求的缓存时间
  allowTTL: 50
  # 以秒计的时长，控制批准请求的缓存时间
  denyTTL: 50
  # 以毫秒计的时长，控制重试间隔
  retryBackoff: 500
  # 确定 webhook 后端失效时的行为
  defaultAllow: true
```

从文件中引用 ImagePolicyWebhook 的配置文件，并将其提供给 API 服务器命令标志 `--admission-control-config-file`：

apiserver.config.k8s.io/v1 apiserver.k8s.io/v1alpha1

引导令牌
rap Tokens）认证
请求
控制器
控制
账号
C 鉴权

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

etes

文档 Kubernetes 博客 培训

ImagePolicyWebhook 的配置文件必须引用 kubeconfig 格式的文件；该文件设置了到后端的连接参数。 要求后端使用 TLS 进行通信。

kubeconfig 文件的 cluster 字段需要指向远端服务，user 字段需要包含已返回的授权者。

```yaml
# clusters 指的是远程服务。
clusters:
- name: name-of-remote-imagepolicy-service
  cluster:
    certificate-authority: /path/to/ca.pem    # CA 用于验证远程服务
    server: https://images.example.com/policy # 要查询的远程服务的 URL。必须是 'https'。

# users 指的是 API 服务器的 webhook 配置。
users:
- name: name-of-api-server
  user:
    client-certificate: /path/to/cert.pem # webhook 准入控制器使用的证书
    client-key: /path/to/key.pem          # 证书匹配的密钥
```

在imagepolicywebhook下

关于 HTTP 配置的更多信息，请参阅 kubeconfig 文档。

引导令牌
ap Tokens）认证
请求
控制器
控制
账号

答题：

考试时执行，切换集群。模拟环境中不需要执行。
kubectl config use-context KSSH00901

本题分数比较多，占 10%

第 1 步 切换到 Master 的 root 下

ssh master01
sudo -i

```
candidate@node01:~$ ssh master01
```

```
candidate@master01:~$ sudo -i
root@master01:~#
```

第 2 步，编辑 admission_configuration.json（题目会给这个目录），修改 defaultAllow 为 false：

vi /etc/kubernetes/epconfig/admission_configuration.json

……
"denyTTL": 50,
"retryBackoff": 500,
"defaultAllow": false #将 true 改为 false
……

第 3 步，编辑/etc/kubernetes/epconfig/kubeconfig.yml，添加 webhook server 地址：

操作前，先备份配置文件

mkdir bak16
cp /etc/kubernetes/epconfig/kubeconfig.yml bak16/
vi /etc/kubernetes/epconfig/kubeconfig.yml
修改如下内容
……
    certificate-authority: /etc/kubernetes/epconfig/server.crt
    server: https://image-bouncer-webhook.default.svc:1323/image_policy #添加 webhook server 地址
  name: bouncer_webhook
……

第 4 步，编辑 kube-apiserver.yaml，从官网中引用 ImagePolicyWebhook 的配置信息：
操作前，先备份配置文件
mkdir bak16
cp /etc/kubernetes/manifests/kube-apiserver.yaml bak16/
vi /etc/kubernetes/manifests/kube-apiserver.yaml
在- command:下添加如下内容，注意空格要对齐
    - --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook
    - --admission-control-config-file=/etc/kubernetes/epconfig/admission_configuration.json #在 1.23 的考试中，默认这行已经添加了，但为了以防万一，模拟环境，没有添加，需要你手动添加。考试时，你先检查，有的话，就不要再重复添加了。重复添加反而会报错!

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=11.0.1.111
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml
    - --audit-log-path=/var/log/kubernetes/audit-logs.txt
    - --audit-log-maxage=10
    - --audit-log-maxbackup=2
    - --insecure-port=0
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook
    - --admission-control-config-file=/etc/kubernetes/epconfig/admission_configuration.json
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
```

在 kube-apiserver.yaml 的 volumeMounts 增加
  volumeMounts: #在 volumeMounts 下面增加 #注意，在 1.23 考试中，蓝色的内容可能已经有了，你只需要添加绿色字体的内容。可以通过红色字，在文件中定位。但模拟环境没有加，需要你全部手动添加。这样是为了练习，万一考试中没有，你也会加。但是如果考试中已添加，你再添加一遍，则会报错，导致 api-server 启不起来。
    - mountPath: /etc/kubernetes/epconfig
      name: epconfig
      readOnly: true

```
      scheme: HTTPS
    initialDelaySeconds: 10
    periodSeconds: 10
    timeoutSeconds: 15
  volumeMounts:
  - mountPath: /etc/kubernetes/epconfig
    name: epconfig
  - mountPath: /etc/kubernetes/logpolicy/sample-policy.yaml
    name: audit
    readOnly: true
  - mountPath: /var/log/kubernetes/
    name: audit-log
    readOnly: false
  - mountPath: /etc/ssl/certs
    name: ca-certs
    readOnly: true
```

在 kube-apiserver.yaml 的 volumes 增加
  volumes: #在 volumes 下面增加 #注意，在 1.23 考试中，蓝色的内容可能已经有了，你只需要检查确认一下是否准确。可以通过红色字，在文件中定位。但模拟环境没有加，需要你全部手动添加。这样是为了练习，万一考试中没有，你也会加。但是如果考试中已添加，你再添加一遍，则会报错，导致 api-server 启不起来。
  - name: epconfig
    hostPath:
      path: /etc/kubernetes/epconfig
      type: DirectoryOrCreate
#如果你写的是目录，则是 DirectoryOrCreate，如果你写的是文件，则是 File

```
  securityContext:
    seccompProfile:
      type: RuntimeDefault
  volumes:
  - name: epconfig
    hostPath:
      path: /etc/kubernetes/epconfig
  - name: audit
    hostPath:
      path: /etc/kubernetes/logpolicy/sample-policy.yaml
      type: File
  - name: audit-log
    hostPath:
      path: /var/log/kubernetes/
      type: DirectoryOrCreate
```

第 5 步，重启 kubelet。

systemctl restart kubelet

通过尝试部署易受攻击的资源 /cks/img/web1.yaml 来测试配置是否有效
无法创建 pod，如下报错，表示成功。
因为模拟环境里的 image_policy 策略是镜像 tag 是 latest 的不允许创建。

kubectl apply -f /cks/img/web1.yaml

```
root@master01:~# kubectl apply -f /cks/img/web1.yaml
Error from server (Forbidden): error when creating "/cks/img/web1.yaml": pods "pod-img" is forbidden: image policy
 webhook backend denied one or more images: Images using latest tag are not allowed
root@master01:~#
```

第 6 步 退回到原 ssh 终端

退出 root，退回到 candidate@master01
exit
退出 master01，退回到 candidate@node01
exit

我的微信是 shadowooom 有在考试模拟环境里做题问题，考试流程问题，都可以问我的。

题库更新内容：

2022 年 8 月 4 号

1、优化 sysdig&falco 的答案

2022 年 7 月 24 号

1、更新资料为 v1.24
2、优化部分答案

2022 年 7 月 4 号

1、根据新考试平台，更新答案和参考网页

2022 年 5 月 5 号

1、更新日志审计的参考连接

2022 年 4 月 13 号

1、更新 pod 安全策略的参考地址

2022 年 4 月 7 号

1、更正第 2 题“Pod 指定 ServiceAccount”，题目里的一处小错误。

2022 年 3 月 31 号

1、第 15 题，启动 API server 认证这道题，不要再修改- --anonymous-auth 了，默认为- --anonymous-auth=true 是正确的。
2、更新第 7 题 Dockerfile 这道题的做法。
3、第 8 题，沙箱运行容器 gVisor，更改为 3 个 deployment。

2022 年 3 月 26 号

更新 11 题 trivy 注意事项。
这个题非常耗费时间，考试时关注一下剩余时间。如果时间剩余不多，可以放到最后做这道题。

2022 年 3 月 15 号

更新 12 题参考网址

2022 年 3 月 2 号

1、添加沙箱运行容器的注意事项

2022 年 3 月 1 号

1、优化答案内容

2022 年 2 月 24 号

1、2022 年 2 月 21 号，考试中心已更题库 CKS 1.23，模拟环境和答案也更新为 CKS 1.23
2、优化题目标准答案。

2022 年 2 月 16 日

1、优化答案内容

2021 年 12 月 31 日

有下面题目，在最新的 1.22 考试中有改动：
1、kube-bench 修复不安全项
3、默认网络策略 default NetworkPolicy
5、日志审计 log audit
7、Dockerfile 检测
12、AppArmor
14、Pod 安全策略（PodSecurityPolicy PSP）
15、启用 API server 认证

16、ImagePolicyWebhook 容器镜像扫描

可以在整个文档里搜索 1.22，检查 1.22 考试里新的调整项。

2021 年 12 月 28 日

1、更新“日志审计 log audit”
2、更新“Dockerfile 检测”
3、更新“删除非无状态和启用特权的 Pod”
4、更新“Sysdig & falco”