LinuxFoundation.CKS.v2023-08-01.q37

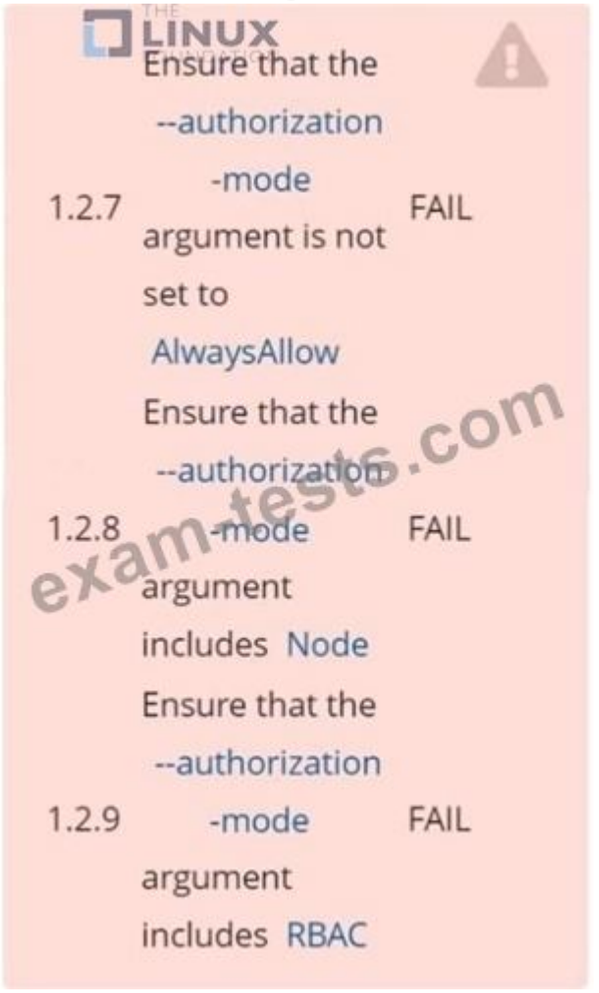
Exam Code:	CKS
Exam Name:	Certified Kubernetes Security Specialist (CKS)
Certification Provider:	Linux Foundation
Free Question Number:	37
Version:	v2023-08-01
# of views:	1001
# of Questions views:	370
https://www.exam-tests.com/CKS-exam/LinuxFoundation.CKS.v2023-08-01.q37.html	

NEW QUESTION: 1
You can switch the cluster/configuration context using the following command: [desk@cli] $ kubectl config use-context stage Context: A PodSecurityPolicy shall prevent the creation of privileged Pods in a specific namespace. Task: 1. Create a new PodSecurityPolcy named deny-policy, which prevents the creation of privileged Pods. 2. Create a new ClusterRole name deny-access-role, which uses the newly created PodSecurityPolicy deny-policy. 3. Create a new ServiceAccount named psd-denial-sa in the existing namespace development. Finally, create a new ClusterRoleBindind named restrict-access-bind, which binds the newly created ClusterRole deny-access-role to the newly created ServiceAccount psp-denial-sa
Answer:
Create psp to disallow privileged container
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: deny-access-role
rules:
- apiGroups: ['policy']
resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- "deny-policy"
k create sa psp-denial-sa -n development
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: restrict-access-bing
roleRef:

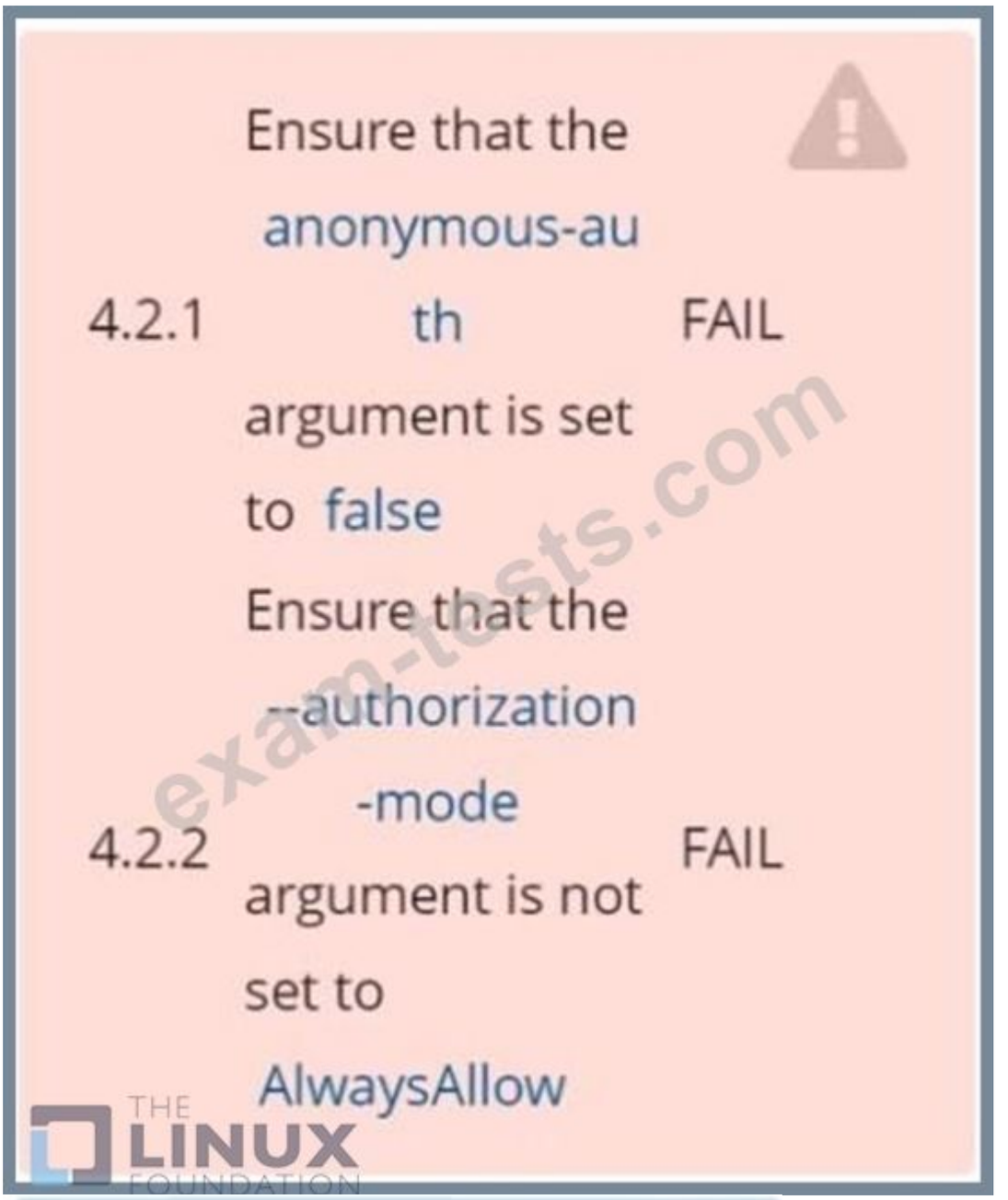
kind: ClusterRole
name: deny-access-role
apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
name: psp-denial-sa
namespace: development
Explanation
master1 $ vim psp.yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
name: deny-policy
spec:
privileged: false # Don't allow privileged pods!
seLinux:
rule: RunAsAny
supplementalGroups:
rule: RunAsAny
runAsUser:
rule: RunAsAny
fsGroup:
rule: RunAsAny
volumes:
- '*'
master1 $ vim cr1.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: deny-access-role
rules:
- apiGroups: ['policy']
resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- "deny-policy"
master1 $ k create sa psp-denial-sa -n development master1 $ vim cb1.yaml apiVersion:
rbac.authorization.k8s.io/v1 kind: ClusterRoleBinding metadata:
name: restrict-access-bing
roleRef:
kind: ClusterRole

name: deny-access-role
apiGroup: rbac.authorization.k8s.io
subjects:
Authorize specific service accounts:
- kind: ServiceAccount
name: psp-denial-sa
namespace: development
master1 $ k apply -f psp.yaml master1 $ k apply -f cr1.yaml master1 $ k apply -f cb1.yaml
Reference: https://kubernetes.io/docs/concepts/policy/pod-security-policy/

NEW QUESTION: 2
Context
A CIS Benchmark tool was run against the kubeadm-created cluster and found multiple issues that must be addressed immediately.
Task
Fix all issues via configuration and restart the affected components to ensure the new settings take effect.
Fix all of the following violations that were found against the API server:



Fix all of the following violations that were found against the Kubelet:



Fix all of the following violations that were found against etcd:

Answer:

```
erviceaccount "podrunner" deleted
andidate@cli:~$ kubectl config use-context KSCS00201
witched to context "KSCS00201".
andidate@cli:~$ ssh kscs00201-master
arning: Permanently added '10.240.86.194' (ECDSA) to the list of known hosts.

he programs included with the Ubuntu system are free software;
he exact distribution terms for each program are described in the
ndividual files in /usr/share/doc/*/copyright.

buntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
pplicable law.

oot@kscs00201-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
oot@kscs00201-master:~# systemctl daemon-reload
oot@kscs00201-master:~# systemctl restart kubelet.service
oot@kscs00201-master:~# systemctl enable kubelet.service
oot@kscs00201-master:~# systemctl status kubelet.service
 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 Fri 2022-05-20 14:19:31 UTC; 29s ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 134205 (kubelet)
      Tasks: 16 (limit: 76200)
     Memory: 39.5M
     CGroup: /system.slice/kubelet.service
             └─134205 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kub

ay 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420825  134205 reconciler.
ay 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420863  134205 reconciler.
ay 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420907  134205 reconciler.
ay 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420928  134205 reconciler.
ay 20 14:19:36 kscs00201-master kubelet[134205]: I0520 14:19:36.572353  134205 request.go:
ay 20 14:19:37 kscs00201-master kubelet[134205]: I0520 14:19:37.112347  134205 prober_mana
ay 20 14:19:37 kscs00201-master kubelet[134205]: E0520 14:19:37.185076  134205 kubelet.go:
ay 20 14:19:37 kscs00201-master kubelet[134205]: I0520 14:19:37.645798  134205 kubelet.go:
ay 20 14:19:38 kscs00201-master kubelet[134205]: I0520 14:19:38.184062  134205 kubelet.go:
ay 20 14:19:40 kscs00201-master kubelet[134205]: I0520 14:19:40.036042  134205 prober_mana
ines 1-22/22 (END)
```

```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
```

```
~
~
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /etc/kubernetes/manifests/etcd.yaml
root@kscs00201-master:~# systemctl daemon-reload
root@kscs00201-master:~# systemctl restart kubelet.service
root@kscs00201-master:~# systemctl status kubelet.service
```

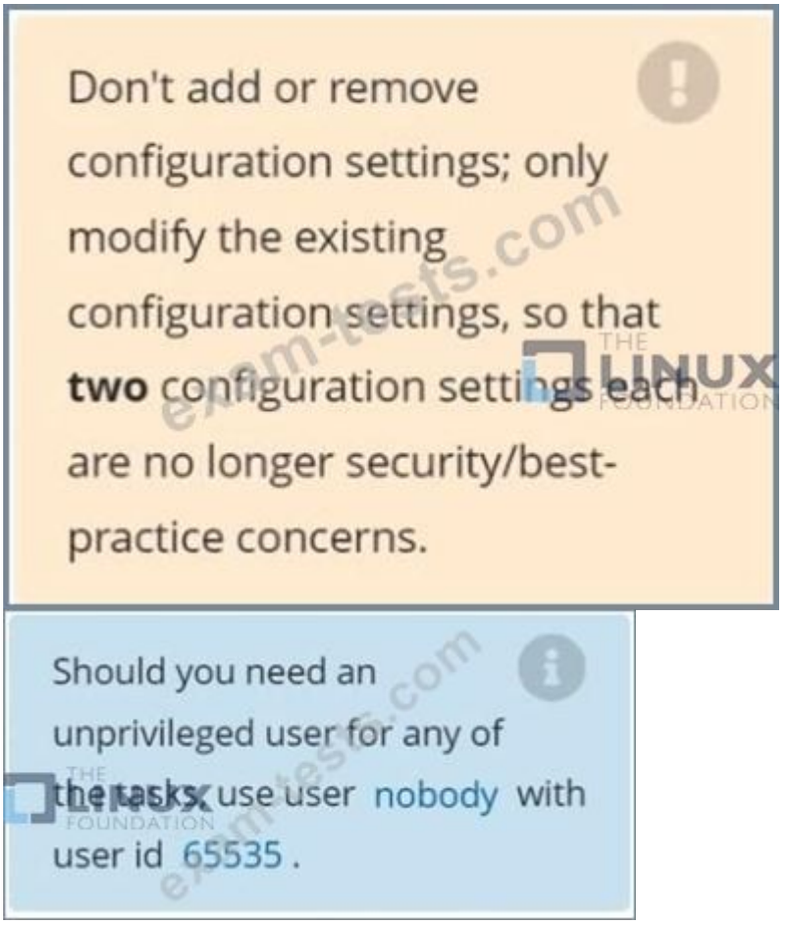
```
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Fri 2022-05-20 14:22:29 UTC; 4s ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 135849 (kubelet)
      Tasks: 17 (limit: 76200)
     Memory: 38.0M
     CGroup: /system.slice/kubelet.service
             └─135849 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kub>

May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330232  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330259  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330304  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330354  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330378  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330397  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330415  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330433  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330452  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330463  135849 reconciler.>
lines 1-22/22 (END)
```

NEW QUESTION: 3

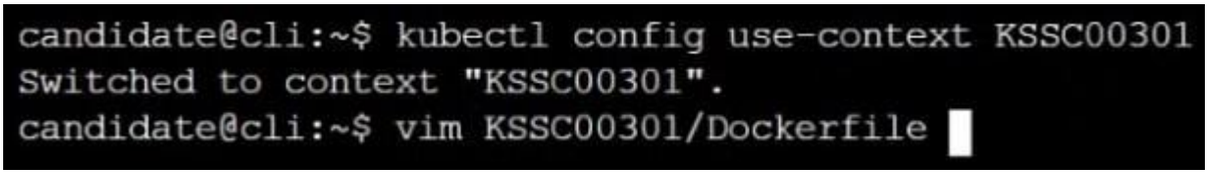
Task

Analyze and edit the given Dockerfile /home/candidate/KSSC00301/Docker file (based on the ubuntu:16.04 image), fixing two instructions present in the file that are prominent security/best-practice issues.

Analyze and edit the given manifest file /home/candidate/KSSC00301/deployment.yaml, fixing two fields present in the file that are prominent security/best-practice issues.



Answer:



```
FROM ubuntu:16.04

USER root
RUN apt-get update && \
    apt-get install -yq --no-install-recommends runiti=2.1.2-3ubuntu1 wget=1.17.1-1ubuntu1.5
 \
        chrpath=0.16-1 tzdata=2020a-0ubuntu0.16.04 lsof=4.89+dfsg-0.1 lshw=02.17-1.1ubuntu3
\
        sysstat=11.2.0-1ubuntu0.3 net-tools=1.60-26ubuntu1 numactl=2.0.11-1ubuntu1.1 \
        bzip2=1.0.6-8ubuntu0.2 && \
    apt-get autoremove && apt-get clean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

ARG CB_VERSION=6.5.1
ARG CB_RELEASE_URL=https://packages.couchbase.com/releases/6.5.1
ARG CB_PACKAGE=couchbase-server-enterprise_6.5.1-ubuntu16.04_amd64.deb
ARG CB_SHA256=80427193137e5cb5a4795b2675b1c450c1af8cf1a5c634d917f6c416f2047e66

ENV PATH=$PATH:/opt/couchbase/bin:/opt/couchbase/bin/tools:/opt/couchbase/bin/install

RUN groupadd -g 1000 couchbase && useradd couchbase -u 1000 -g couchbase -M

SHELL ["/bin/bash", "-o", "pipefail", "-c"]
RUN export INSTALL_DONT_START_SERVER=1 && \
    wget -N --no-verbose $CB_RELEASE_URL/$CB_PACKAGE && \
    echo "$CB_SHA256  $CB_PACKAGE" | sha256sum -c - && \
    dpkg -i ./$CB_PACKAGE && rm -f ./$CB_PACKAGE
COPY scripts/run /etc/service/couchbase-server/run
RUN chown -R couchbase:couchbase /etc/service

COPY scripts/dummy.sh /usr/local/bin/
RUN ln -s dummy.sh /usr/local/bin/iptables-save && \
    ln -s dummy.sh /usr/local/bin/lvdisplay && \
    ln -s dummy.sh /usr/local/bin/vgdisplay && \
    ln -s dummy.sh /usr/local/bin/pvdisplay

RUN chrpath -r "\$ORIGIN/../lib" /opt/couchbase/bin/curl

COPY scripts/entrypoint.sh /
ENTRYPOINT ["/entrypoint.sh"]
USER nobody
CMD ["couchbase-server"]

EXPOSE 8091 8092 8093 8094 8095 8096 11207 11210 11211 18091 18092 18093 18094 18095 18096
VOLUME /opt/couchbase/var
```

```
candidate@cli:~$ kubectl config use-context KSSC00301
Switched to context "KSSC00301".
candidate@cli:~$ vim KSSC00301/Dockerfile
candidate@cli:~$ vim KSSC00301/deployment.yaml
```

```
securityContext:
  {'capabilities': {'add': ['NET_BIND_SERVICE'], 'drop': ['all']}, 'privileged': F
alse, 'readOnlyRootFilesystem': True, 'runAsUser': 65535}
resources:
  limits:
    cpu: 2
    memory: 1024Mi
  requests:
    cpu: 1
    memory: 512Mi
volumes:
  - name: database-storage
```

NEW QUESTION: 4

Context

A default-deny NetworkPolicy avoids to accidentally expose a Pod in a namespace that doesn't have any other NetworkPolicy defined.

Task

Create a new default-deny NetworkPolicy named defaultdeny in the namespace testing for all traffic of type Egress.

The new NetworkPolicy must deny all Egress traffic in the namespace testing.

Apply the newly created default-deny NetworkPolicy to all Pods running in namespace testing.

Answer:

```
candidate@cli:~$ kubectl config use-context KSCS00101
Switched to context "KSCS00101".
candidate@cli:~$ cat /home/candidate/KSCS00101/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ""
  namespace: ""
spec:
  podSelector: {}
  policyTypes: []
candidate@cli:~$ vim /home/candidate/KSCS00101/network-policy.yaml
candidate@cli:~$
```


apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: "defaultdeny"
namespace: "testing"
spec:
podSelector: {}
policyTypes:
- Egress
egress:
- to:
- podSelector: {}
namespaceSelector:
matchLabels:
access: testingproject
exam-tests.com
THE LINUX FOUNDATION

NEW QUESTION: 5

Analyze and edit the given Dockerfile

FROM ubuntu:latest

RUN apt-get update -y

RUN apt-install nginx -y

COPY entrypoint.sh /

ENTRYPOINT ["/entrypoint.sh"]

USER ROOT

Fixing two instructions present in the file being prominent security best practice issues Analyze and edit the deployment manifest file apiVersion: v1 kind: Pod metadata:

name: security-context-demo-2

spec:

securityContext:

runAsUser: 1000
containers:
- name: sec-ctx-demo-2
image: gcr.io/google-samples/node-hello:1.0
securityContext:
runAsUser: 0
privileged: True
allowPrivilegeEscalation: false
Fixing two fields present in the file being prominent security best practice issues Don't add or remove configuration settings; only modify the existing configuration settings Whenever you need an unprivileged user for any of the tasks, use user test-user with the user id 5487

Answer:

FROM debian:latest
MAINTAINER k@bogotobogo.com
1 - RUN
RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install -yq apt-utils RUN DEBIAN_FRONTEND=noninteractive apt-get install -yq htop RUN apt-get clean
2 - CMD
#CMD ["htop"]
#CMD ["ls", "-l"]
3 - WORKDIR and ENV
WORKDIR /root
ENV DZ version1
$ docker image build -t bogodevops/demo .
Sending build context to Docker daemon 3.072kB
Step 1/7 : FROM debian:latest
---> be2868bebaba
Step 2/7 : MAINTAINER k@bogotobogo.com
---> Using cache
---> e2eef476b3fd
Step 3/7 : RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install -yq apt-utils
---> Using cache
---> 32fd044c1356
Step 4/7 : RUN DEBIAN_FRONTEND=noninteractive apt-get install -yq htop
---> Using cache
---> 0a5b514a209e
Step 5/7 : RUN apt-get clean
---> Using cache
---> 5d1578a47c17
Step 6/7 : WORKDIR /root

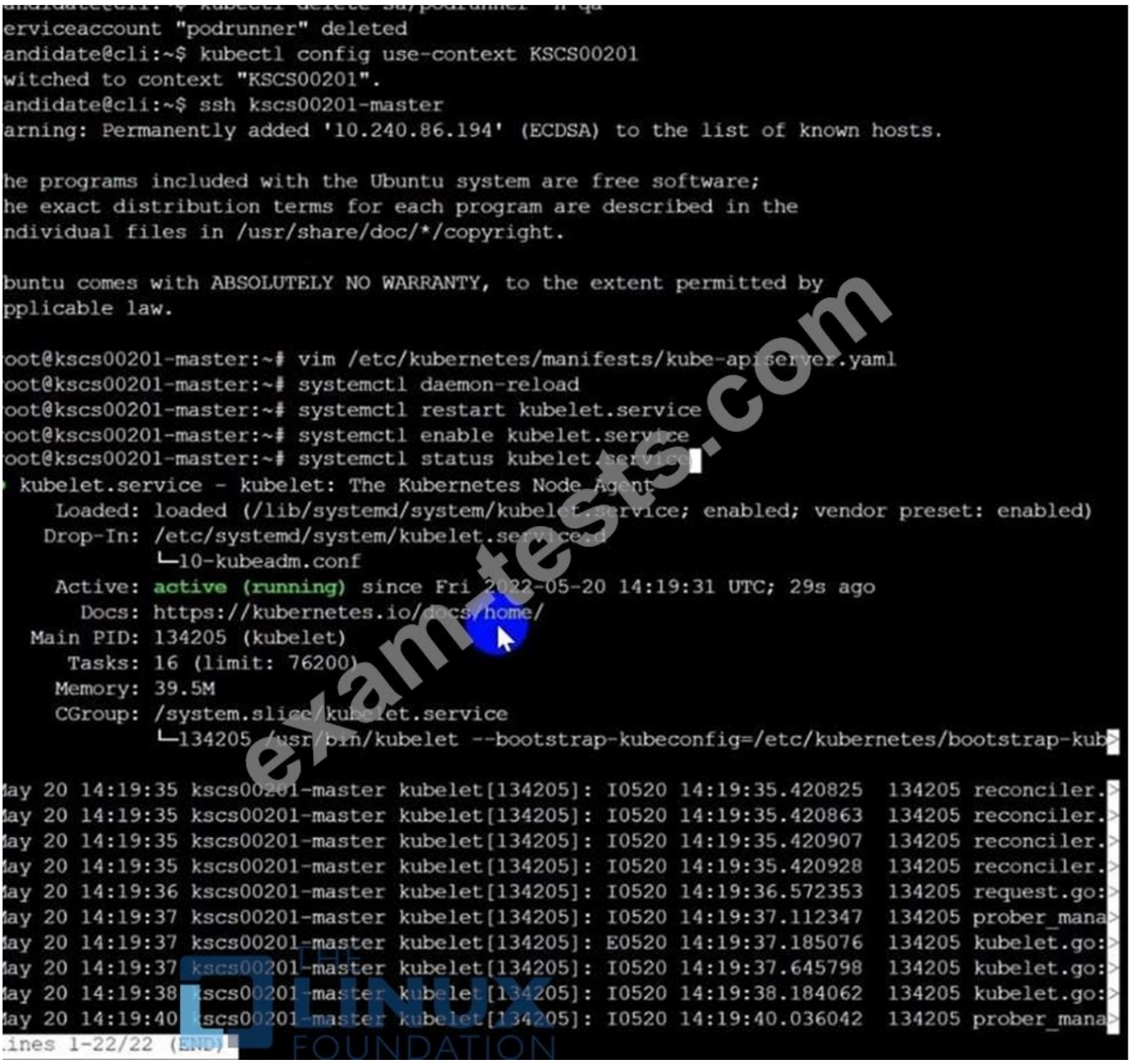
---> Using cache
---> 6b1c70e87675
Step 7/7 : ENV DZ version1
---> Using cache
---> cd195168c5c7
Successfully built cd195168c5c7
Successfully tagged bogodevops/demo:latest

NEW QUESTION: 6
Fix all issues via configuration and restart the affected components to ensure the new setting takes effect.
Fix all of the following violations that were found against the API server:- a. Ensure the --authorization-mode argument includes RBAC b. Ensure the --authorization-mode argument includes Node c. Ensure that the --profiling argument is set to false Fix all of the following violations that were found against the Kubelet:- a. Ensure the --anonymous-auth argument is set to false.
b. Ensure that the --authorization-mode argument is set to Webhook.
Fix all of the following violations that were found against the ETCD:-
a. Ensure that the --auto-tls argument is not set to true
Hint: Take the use of Tool Kube-Bench
Answer:
API server:
Ensure the --authorization-mode argument includes RBAC
Turn on Role Based Access Control. Role Based Access Control (RBAC) allows fine-grained control over the operations that different entities can perform on different objects in the cluster. It is recommended to use the RBAC authorization mode.
Fix - Buildtime
Kubernetes
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
component: kube-apiserver
tier: control-plane
name: kube-apiserver
namespace: kube-system
spec:
containers:
- command:
+ - kube-apiserver

+ - --authorization-mode=RBAC,Node
image: gcr.io/google_containers/kube-apiserver-amd64:v1.6.0
livenessProbe:
failureThreshold: 8
httpGet:
host: 127.0.0.1
path: /healthz
port: 6443
scheme: HTTPS
initialDelaySeconds: 15
timeoutSeconds: 15
name: kube-apiserver-should-pass
resources:
requests:
cpu: 250m
volumeMounts:
- mountPath: /etc/kubernetes/
name: k8s
readOnly: true
- mountPath: /etc/ssl/certs
name: certs
- mountPath: /etc/pki
name: pki
hostNetwork: true
volumes:
- hostPath:
path: /etc/kubernetes
name: k8s
- hostPath:
path: /etc/ssl/certs
name: certs
- hostPath:
path: /etc/pki
name: pki
Ensure the --authorization-mode argument includes Node
Remediation: 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
Audit:
/bin/ps -ef | grep kube-apiserver | grep -v grep

Expected result:
'Node,RBAC' has 'Node'
Ensure that the --profiling argument is set to false
Remediation: Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the below parameter.
--profiling=false
Audit:
/bin/ps -ef | grep kube-apiserver | grep -v grep
Expected result:
'false' is equal to 'false'
Fix all of the following violations that were found against the Kubelet:- Ensure the --anonymous-auth argument is set to false.
Remediation: 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
Audit:
/bin/ps -fC kubelet
Audit Config:
/bin/cat /var/lib/kubelet/config.yaml
Expected result:
'false' is equal to 'false'
2) Ensure that the --authorization-mode argument is set to Webhook.
Audit
docker inspect kubelet | jq -e '.[0].Args[] | match("--authorization-mode=Webhook").string'
Returned Value: --authorization-mode=Webhook Fix all of the following violations that were found against the ETCD:- a. Ensure that the --auto-tls argument is not set to true Do not use self-signed certificates for TLS. etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.
Fix - Buildtime
Kubernetes
apiVersion: v1
kind: Pod
metadata:
annotations:

scheduler.alpha.kubernetes.io/critical-pod: ""
creationTimestamp: null
labels:
component: etcd
tier: control-plane
name: etcd
namespace: kube-system
spec:
containers:
- command:
+ - etcd
+ - --auto-tls=true
image: k8s.gcr.io/etcd-amd64:3.2.18
imagePullPolicy: IfNotPresent
livenessProbe:
exec:
command:
- /bin/sh
- -ec
- ETCDCTL_API=3 etcdctl --endpoints=https://[192.168.22.9]:2379 --
cacert=/etc/kubernetes/pki/etcd/ca.crt
--cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=/etc/kubernetes/pki/etcd/healthcheck-client.key get foo failureThreshold: 8 initialDelaySeconds: 15 timeoutSeconds: 15 name: etcd-should-fail resources: {} volumeMounts:
- mountPath: /var/lib/etcd
name: etcd-data
- mountPath: /etc/kubernetes/pki/etcd
name: etcd-certs
hostNetwork: true
priorityClassName: system-cluster-critical
volumes:
- hostPath:
path: /var/lib/etcd
type: DirectoryOrCreate
name: etcd-data
- hostPath:
path: /etc/kubernetes/pki/etcd
type: DirectoryOrCreate
name: etcd-certs
status: {}
Explanation:



```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
```

```
~
~
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /etc/kubernetes/manifests/etcd.yaml
root@kscs00201-master:~# systemctl daemon-reload
root@kscs00201-master:~# systemctl restart kubelet.service
root@kscs00201-master:~# systemctl status kubelet.service
```

```
● 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 Fri 2022-05-20 14:22:29 UTC; 4s ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 135849 (kubelet)
      Tasks: 17 (limit: 76200)
     Memory: 38.0M
     CGroup: /system.slice/kubelet.service
             └─135849 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kub>

May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330232  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330259  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330304  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330354  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330378  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330397  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330415  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330433  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330452  135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330463  135849 reconciler.>
lines 1-22/22 (END)
```

NEW QUESTION: 7
use the Trivy to scan the following images,
A. 1. amazonlinux:1
Answer: A (LEAVE A REPLY)
2. k8s.gcr.io/kube-controller-manager:v1.18.6
Look for images with HIGH or CRITICAL severity vulnerabilities and store the output of the same in /opt/trivy-vulnerable.txt

NEW QUESTION: 8
SIMULATION
Create a PSP that will prevent the creation of privileged pods in the namespace.
Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.
Create a new ServiceAccount named psp-sa in the namespace default.
Create a new ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.
Create a new ClusterRoleBinding named prevent-role-binding, which binds the created ClusterRole prevent-role to the created SA psp-sa.
Also, Check the Configuration is working or not by trying to Create a Privileged pod, it should get failed.
Answer:
Create a PSP that will prevent the creation of privileged pods in the namespace.
$ cat clusterrole-use-privileged.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: use-privileged-psp
rules:
- apiGroups: ['policy']
resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- default-psp

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
name: privileged-role-bind
namespace: psp-test
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: use-privileged-psp
subjects:
- kind: ServiceAccount
name: privileged-sa
$ kubectl -n psp-test apply -f clusterrole-use-privileged.yaml
After a few moments, the privileged Pod should be created.
Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.
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:
- '*'
And create it with kubectl:
kubectl-admin create -f example-psp.yaml
Now, as the unprivileged user, try to create a simple pod:
kubectl-user create -f- <<EOF
apiVersion: v1
kind: Pod
metadata:

name: pause
spec:
containers:
- name: pause
image: k8s.gcr.io/pause
EOF
The output is similar to this:
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to validate against any pod security policy: [] Create a new ServiceAccount named psp-sa in the namespace default.
$ cat clusterrole-use-privileged.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: use-privileged-psp
rules:
- apiGroups: ['policy']
resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- default-psp
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
name: privileged-role-bind
namespace: psp-test
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: use-privileged-psp
subjects:
- kind: ServiceAccount
name: privileged-sa
$ kubectl -n psp-test apply -f clusterrole-use-privileged.yaml
After a few moments, the privileged Pod should be created.
Create a new ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.
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:
- '*'
And create it with kubectl:
kubectl-admin create -f example-psp.yaml
Now, as the unprivileged user, try to create a simple pod:
kubectl-user create -f- <<EOF
apiVersion: v1
kind: Pod
metadata:
name: pause
spec:
containers:
- name: pause
image: k8s.gcr.io/pause
EOF
The output is similar to this:
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to validate against any pod security policy: [] Create a new ClusterRoleBinding named prevent-role-binding, which binds the created ClusterRole prevent-role to the created SA psp-sa.
apiVersion: rbac.authorization.k8s.io/v1
This role binding allows "jane" to read pods in the "default" namespace.
You need to already have a Role named "pod-reader" in that namespace.
kind: RoleBinding
metadata:
name: read-pods
namespace: default
subjects:
You can specify more than one "subject"

- kind: User
name: jane # "name" is case sensitive
apiGroup: rbac.authorization.k8s.io
roleRef:
"roleRef" specifies the binding to a Role / ClusterRole
kind: Role #this must be Role or ClusterRole
name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to
apiGroup: rbac.authorization.k8s.io apiVersion: rbac.authorization.k8s.io/v1 kind: Role metadata:
namespace: default
name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
resources: ["pods"]
verbs: ["get", "watch", "list"]

NEW QUESTION: 9
Fix all issues via configuration and restart the affected components to ensure the new setting takes effect.
Fix all of the following violations that were found against the API server:- a. Ensure that the RotateKubeletServerCertificate argument is set to true.
b. Ensure that the admission control plugin PodSecurityPolicy is set.
c. Ensure that the --kubelet-certificate-authority argument is set as appropriate.
Fix all of the following violations that were found against the Kubelet:- a. Ensure the --anonymous-auth argument is set to false.
b. Ensure that the --authorization-mode argument is set to Webhook.
Fix all of the following violations that were found against the ETCD:-
a. Ensure that the --auto-tls argument is not set to true
b. Ensure that the --peer-auto-tls argument is not set to true
Hint: Take the use of Tool Kube-Bench
Answer:
Fix all of the following violations that were found against the API server:- a. Ensure that the RotateKubeletServerCertificate argument is set to true.
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
component: kubelet
tier: control-plane
name: kubelet
namespace: kube-system

spec:
containers:
- command:
- kube-controller-manager
+ - --feature-gates=RotateKubeletServerCertificate=true
image: gcr.io/google_containers/kubelet-amd64:v1.6.0
livenessProbe:
failureThreshold: 8
httpGet:
host: 127.0.0.1
path: /healthz
port: 6443
scheme: HTTPS
initialDelaySeconds: 15
timeoutSeconds: 15
name: kubelet
resources:
requests:
cpu: 250m
volumeMounts:
- mountPath: /etc/kubernetes/
name: k8s
readOnly: true
- mountPath: /etc/ssl/certs
name: certs
- mountPath: /etc/pki
name: pki
hostNetwork: true
volumes:
- hostPath:
path: /etc/kubernetes
name: k8s
- hostPath:
path: /etc/ssl/certs
name: certs
- hostPath:
path: /etc/pki
name: pki
b. Ensure that the admission control plugin PodSecurityPolicy is set.
audit: "/bin/ps -ef | grep $apiserverbin | grep -v grep"
tests:

test_items:
- flag: "--enable-admission-plugins"
compare:
op: has
value: "PodSecurityPolicy"
set: true
remediation: |
Follow the documentation and create Pod Security Policy objects as per your environment.
Then, edit the API server pod specification file $apiserverconf
on the master node and set the --enable-admission-plugins parameter to a value that includes PodSecurityPolicy :
--enable-admission-plugins=...,PodSecurityPolicy,...
Then restart the API Server.
scored: true
c. Ensure that the --kubelet-certificate-authority argument is set as appropriate.
audit: "/bin/ps -ef | grep $apiserverbin | grep -v grep"
tests:
test_items:
- flag: "--kubelet-certificate-authority"
set: true
remediation: |
Follow the Kubernetes documentation and setup the TLS connection between the apiserver and kubelets. Then, edit the API server pod specification file
$apiserverconf on the master node and set the --kubelet-certificate-authority parameter to the path to the cert file for the certificate authority.
--kubelet-certificate-authority=<ca-string>
scored: true
Fix all of the following violations that were found against the ETCD:-
a. Ensure that the --auto-tls argument is not set to true
Edit the etcd pod specification file $etcdconf on the master
node and either remove the --auto-tls parameter or set it to false.
--auto-tls=false
b. Ensure that the --peer-auto-tls argument is not set to true
Edit the etcd pod specification file $etcdconf on the master
node and either remove the --peer-auto-tls parameter or set it to false.
--peer-auto-tls=false

NEW QUESTION: 10
SIMULATION
Create a Pod name Nginx-pod inside the namespace testing, Create a service for the Nginx-pod named nginx-svc, using the ingress of your choice, run the ingress on tls, secure port.

A. Sendusyourfeedbackonit

Answer: A (LEAVE A REPLY)

NEW QUESTION: 11

Context

This cluster uses containerd as CRI runtime.

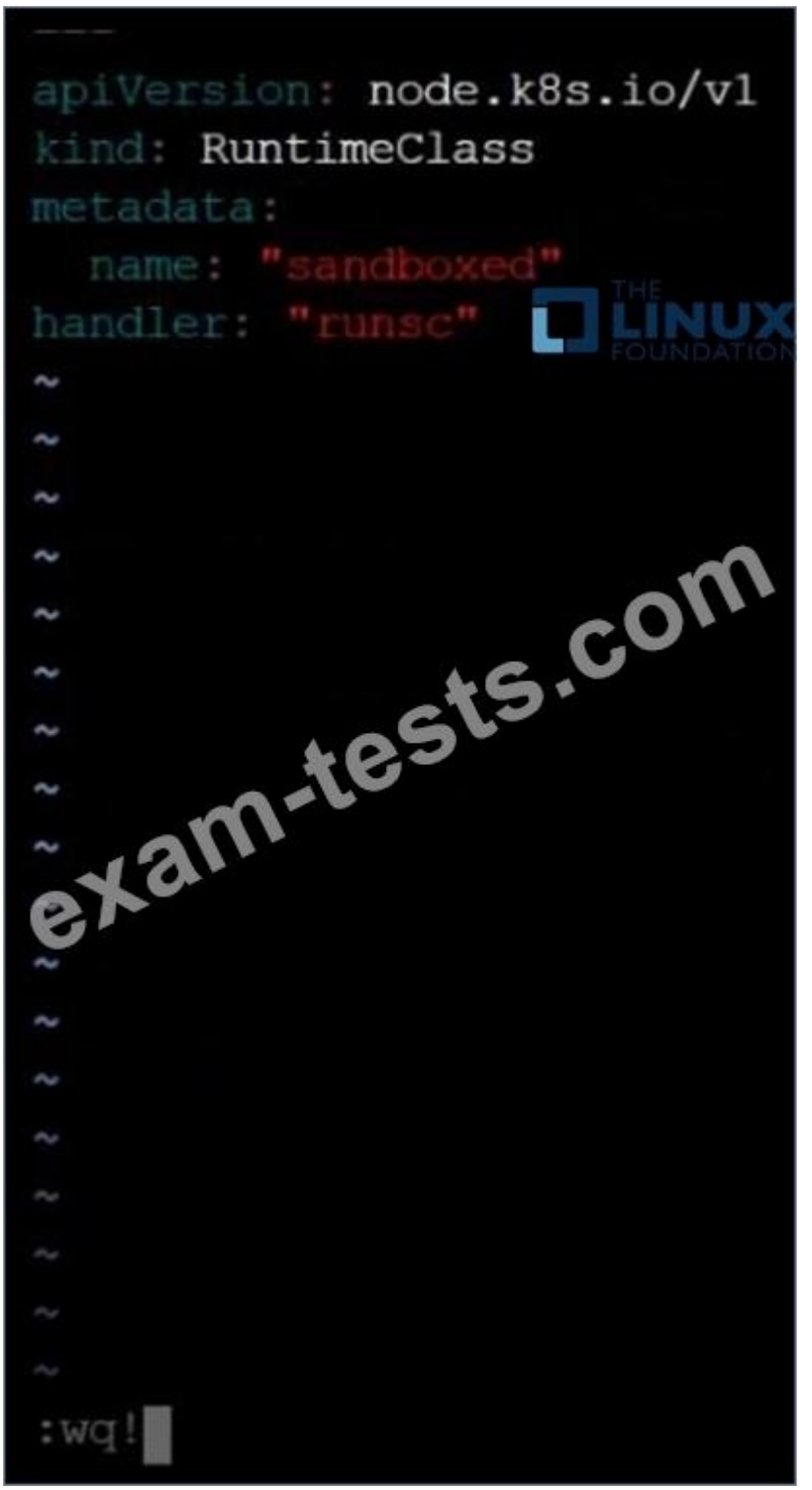
Containerd's default runtime handler is runc. Containerd has been prepared to support an additional runtime handler, runsc (gVisor).

Task

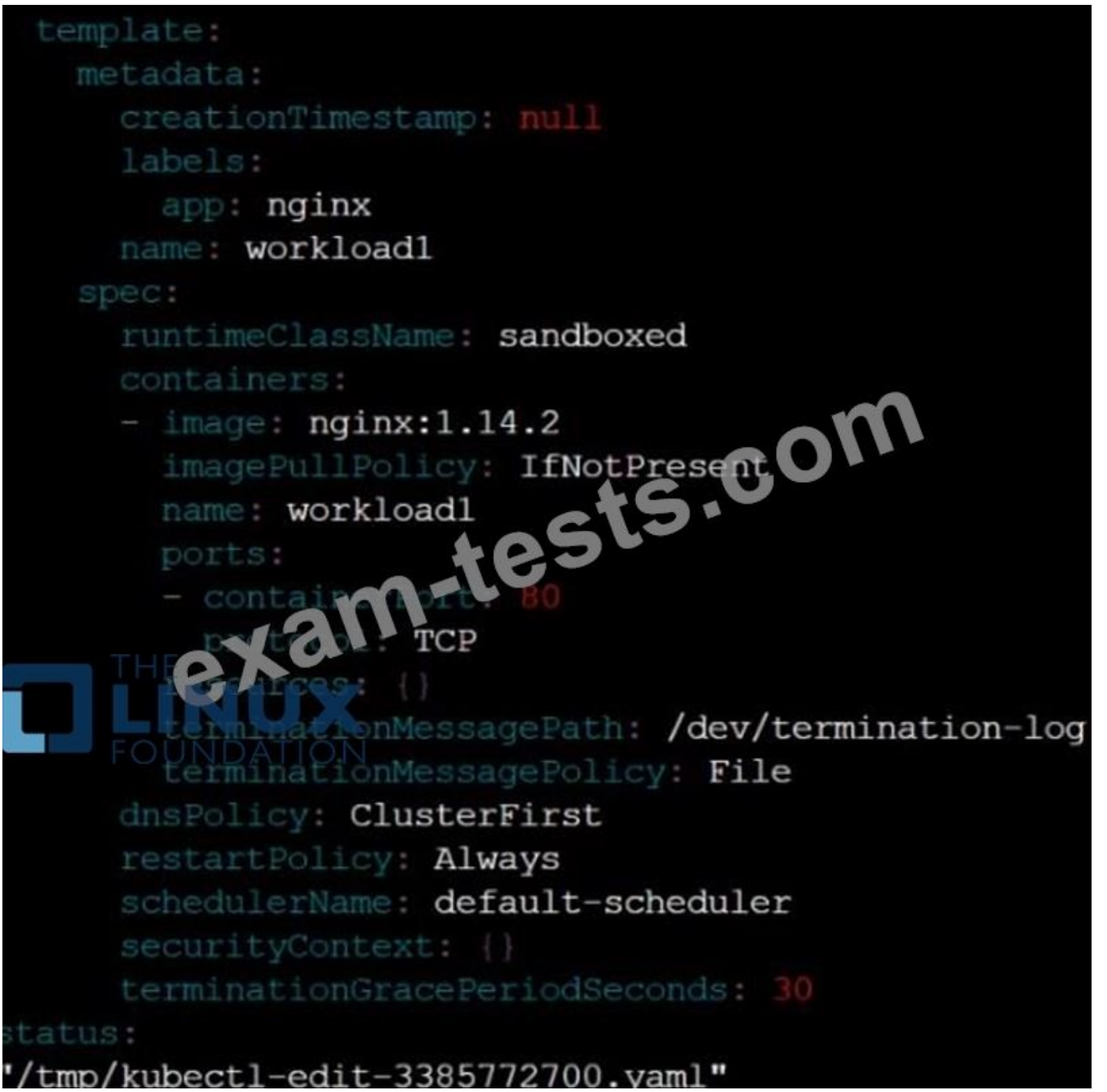
Create a RuntimeClass named sandboxed using the prepared runtime handler named runsc.

Update all Pods in the namespace server to run on gVisor.

Answer:

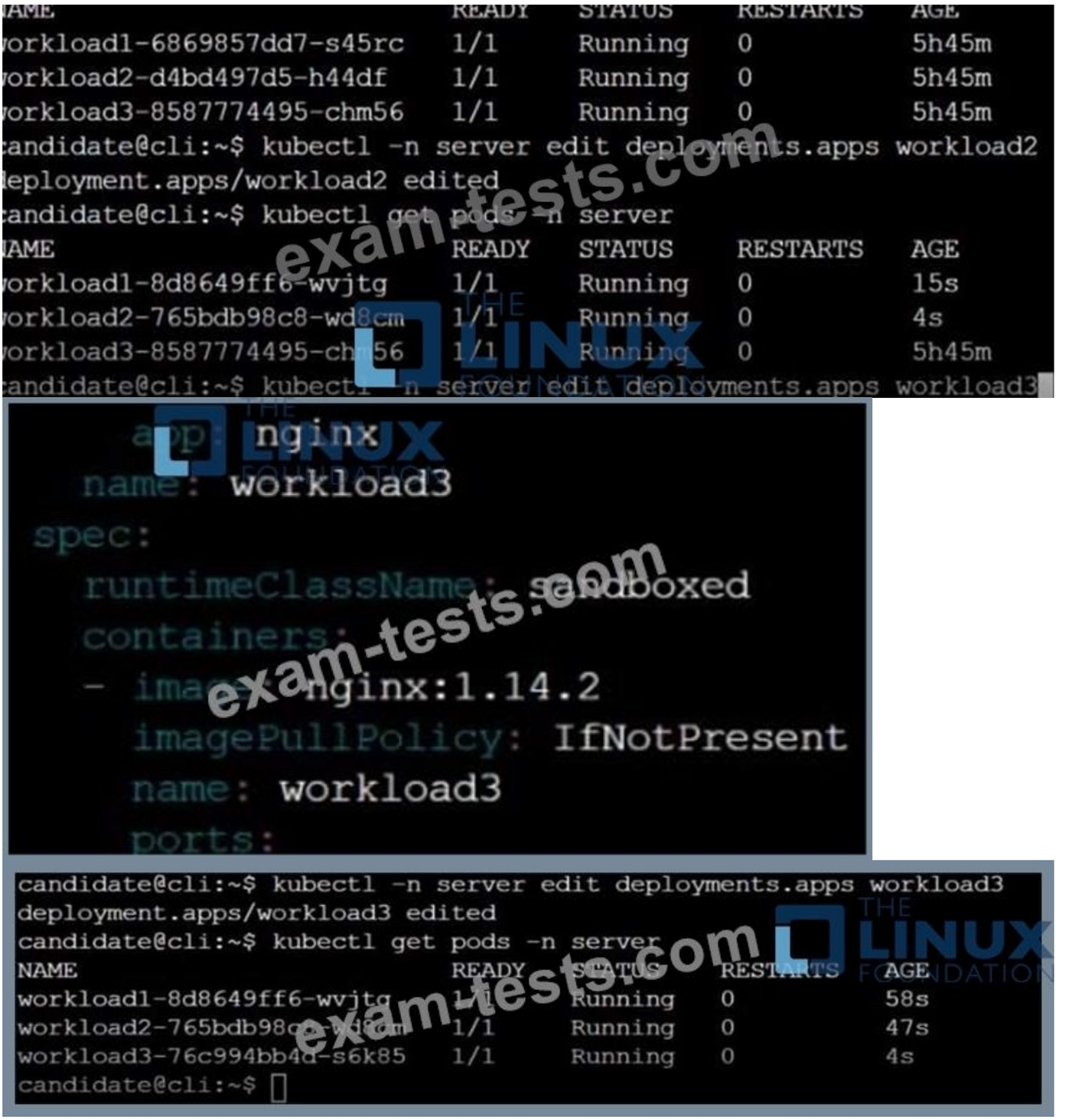


```
candidate@cli:~$ kubectl config use-context KSMV00301
Switched to context "KSMV00301".
candidate@cli:~$ cat /home/candidate/KSMV00301/runtime-class.yaml
---
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: ""
handler: ""
candidate@cli:~$ vim /home/candidate/KSMV00301/runtime-class.yaml
candidate@cli:~$ cat /home/candidate/KSMV00301/runtime-class.yaml
---
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: "sandboxed"
handler: "runsc"
candidate@cli:~$ kubectl get deployments.apps -n server
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
workload1   1/1     1            1           5h43m
workload2   1/1     1            1           5h43m
workload3   1/1     1            1           5h43m
candidate@cli:~$ kubectl get pods -n server
NAME                         READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc   1/1     Running   0          5h43m
workload2-d4bd497d5-h44df    1/1     Running   0          5h43m
workload3-8587774495-chm56   1/1     Running   0          5h43m
candidate@cli:~$ kubectl -n server edit deployments.apps workload1
```



```
NAME                          READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc    1/1     Running   0          5h44m
workload2-d4bd497d5-h44df     1/1     Running   0          5h44m
workload3-8587774495-chm56    1/1     Running   0          5h44m
candidate@cli:~$ kubectl -n server edit deployments.apps workload1
Edit cancelled, no changes made.
candidate@cli:~$ kubectl get pods -n server
NAME                          READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc    1/1     Running   0          5h45m
workload2-d4bd497d5-h44df     1/1     Running   0          5h44m
workload3-8587774495-chm56    1/1     Running   0          5h44m
candidate@cli:~$ kubectl -n server edit deployments.apps workload2
Edit cancelled, no changes made.
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00301/runtime-class.yaml
runtimeclass.node.k8s.io/sandboxed created
candidate@cli:~$ kubectl get pods -n server
NAME                          READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc    1/1     Running   0          5h45m
workload2-d4bd497d5-h44df     1/1     Running   0          5h45m
workload3-8587774495-chm56    1/1     Running   0          5h45m
candidate@cli:~$ kubectl -n server edit deployments.apps workload2
```

```
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
      name: workload2
    spec:
      runtimeClassName: sandboxed
```



NEW QUESTION: 12

SIMULATION

Create a User named john, create the CSR Request, fetch the certificate of the user after approving it.

Create a Role name john-role to list secrets, pods in namespace john

Finally, Create a RoleBinding named john-role-binding to attach the newly created role john-role to the user john in the namespace john. To Verify: Use the kubectl auth CLI command to verify the permissions.

Answer:

se kubectl to create a CSR and approve it.
Get the list of CSRs:
kubectl get csr
Approve the CSR:
kubectl certificate approve myuser
Get the certificate
Retrieve the certificate from the CSR:
kubectl get csr/myuser -o yaml
here are the role and role-binding to give john permission to create NEW_CRD resource:
kubectl apply -f roleBindingJohn.yaml --as=john
rolebinding.rbac.authorization.k8s.io/john_external-rosource-rb created kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1 metadata:
name: john_crd
namespace: development-john
subjects:
- kind: User
name: john
apiGroup: rbac.authorization.k8s.io
roleRef:
kind: ClusterRole
name: crd-creation
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: crd-creation
rules:
- apiGroups: ["kubernetes-client.io/v1"]
resources: ["NEW_CRD"]
verbs: ["create, list, get"]

NEW QUESTION: 13
SIMULATION
On the Cluster worker node, enforce the prepared AppArmor profile
#include <tunables/global>
profile docker-nginx flags=(attach_disconnected,mediate_deleted) {
#include <abstractions/base>
network inet tcp,
network inet udp,
network inet icmp,
deny network raw,
deny network packet,

```
file,
umount,
deny /bin/** wl,
deny /boot/** wl,
deny /dev/** wl,
deny /etc/** wl,
deny /home/** wl,
deny /lib/** wl,
deny /lib64/** wl,
deny /media/** wl,
deny /mnt/** wl,
deny /opt/** wl,
deny /proc/** wl,
deny /root/** wl,
deny /sbin/** wl,
deny /srv/** wl,
deny /tmp/** wl,
deny /sys/** wl,
deny /usr/** wl,
audit /** w,
/var/run/nginx.pid w,
/usr/sbin/nginx ix,
deny /bin/dash mrwklx,
deny /bin/sh mrwklx,
deny /usr/bin/top mrwklx,
capability chown,
capability dac_override,
capability setuid,
capability setgid,
capability net_bind_service,
deny @{PROC}/* w, # deny write for all files directly in /proc (not in a subdir)
# deny write to files not in /proc/<number>/** or /proc/sys/**
deny @{PROC}/{[^1-9],[^1-9][^0-9],[^1-9s][^0-9y][^0-9s],[^1-9][^0-9][^0-9][^0-9]*}/** w, deny
@{PROC}/sys/[^k]** w, # deny /proc/sys except /proc/sys/k* (effectively /proc/sys/kernel) deny
@{PROC}/sys/kernel/{?,??,[^s][^h][^m]**} w, # deny everything except shm* in /proc/sys/kernel/
deny @{PROC}/sysrq-trigger rwklx, deny @{PROC}/mem rwklx, deny @{PROC}/kmem rwklx,
deny @{PROC}/kcore rwklx, deny mount, deny /sys/[^f]*/** wklx, deny /sys/f[^s]*/** wklx,
deny /sys/fs/[^c]*/** wklx, deny /sys/fs/c[^g]*/** wklx, deny /sys/fs/cg[^r]*/** wklx,
deny /sys/firmware/** rwklx, deny /sys/kernel/security/** rwklx,
}
```

Edit the prepared manifest file to include the AppArmor profile.

apiVersion: v1
kind: Pod
metadata:
name: apparmor-pod
spec:
containers:
- name: apparmor-pod
image: nginx
Finally, apply the manifests files and create the Pod specified on it.
Verify: Try to use command ping, top, sh
A. Send us the Feedback on it.
Answer: (SHOW ANSWER)

NEW QUESTION: 14
On the Cluster worker node, enforce the prepared AppArmor profile
#include <tunables/global>
profile nginx-deny flags=(attach_disconnected) {
#include <abstractions/base>
file,
Deny all file writes.
deny /** w,
}
EOF'
Edit the prepared manifest file to include the AppArmor profile.
apiVersion: v1
kind: Pod
metadata:
name: apparmor-pod
spec:
containers:
- name: apparmor-pod
image: nginx
Finally, apply the manifests files and create the Pod specified on it.
Verify: Try to make a file inside the directory which is restricted.
Answer:

```
candidate@cli:~$ kubectl config use-context KSSH00401
Switched to context "KSSH00401".
candidate@cli:~$ ssh kssh00401-worker1
Warning: Permanently added '10.240.86.172' (ECDSA) to the list of known hosts.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@kssh00401-worker1:~# head /etc/apparmor.d/nginx_apparmor
#include <tunables/global>

profile nginx-profile-2 flags=(attach_disconnected,mediate_deleted) {
 #include <abstractions/base>
  network inet tcp,
  network inet udp,
  network inet icmp,

  deny network raw,

root@kssh00401-worker1:~# apparmor_parser -q /etc/apparmor.d/nginx_apparmor
root@kssh00401-worker1:~# exit
logout
Connection to 10.240.86.172 closed.
candidate@cli:~$ cat KSSH00401/nginx-pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx-pod
    image: nginx:1.19.0
    ports:
    - containerPort: 80
candidate@cli:~$ vim KSSH00401/nginx-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  annotations:
    container.apparmor.security.beta.kubernetes.io/nginx-pod: localhost/nginx-p
spec:
  containers:
  - name: nginx-pod
    image: nginx:1.19.0
    ports:
    - containerPort: 80
```

NEW QUESTION: 15

Create a new NetworkPolicy named deny-all in the namespace testing which denies all traffic of type ingress and egress traffic

Answer:

You can create a "default" isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any ingress traffic to those pods.

---

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: default-deny-ingress

spec:

podSelector: {}

policyTypes:

- Ingress

You can create a "default" egress isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any egress traffic from those pods.

---

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: allow-all-egress

spec:

podSelector: {}

egress:

- {}

policyTypes:

- Egress

Default deny all ingress and all egress traffic
You can create a "default" policy for a namespace which prevents all ingress AND egress traffic by creating the following NetworkPolicy in that namespace.
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: default-deny-all
spec:
podSelector: {}
policyTypes:
- Ingress
- Egress
This ensures that even pods that aren't selected by any other NetworkPolicy will not be allowed ingress or egress traffic.

NEW QUESTION: 16
SIMULATION
Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that
1. logs are stored at /var/log/kubernetes/kubernetes-logs.txt.
2. Log files are retained for 5 days.
3. at maximum, a number of 10 old audit logs files are retained.
Edit and extend the basic policy to log:
1. Cronjobs changes at RequestResponse
2. Log the request body of deployments changes in the namespace kube-system.
3. Log all other resources in core and extensions at the Request level.
4. Don't log watch requests by the "system:kube-proxy" on endpoints or
A. Send us the Feedback on it.
Answer: (SHOW ANSWER)

NEW QUESTION: 17
Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that

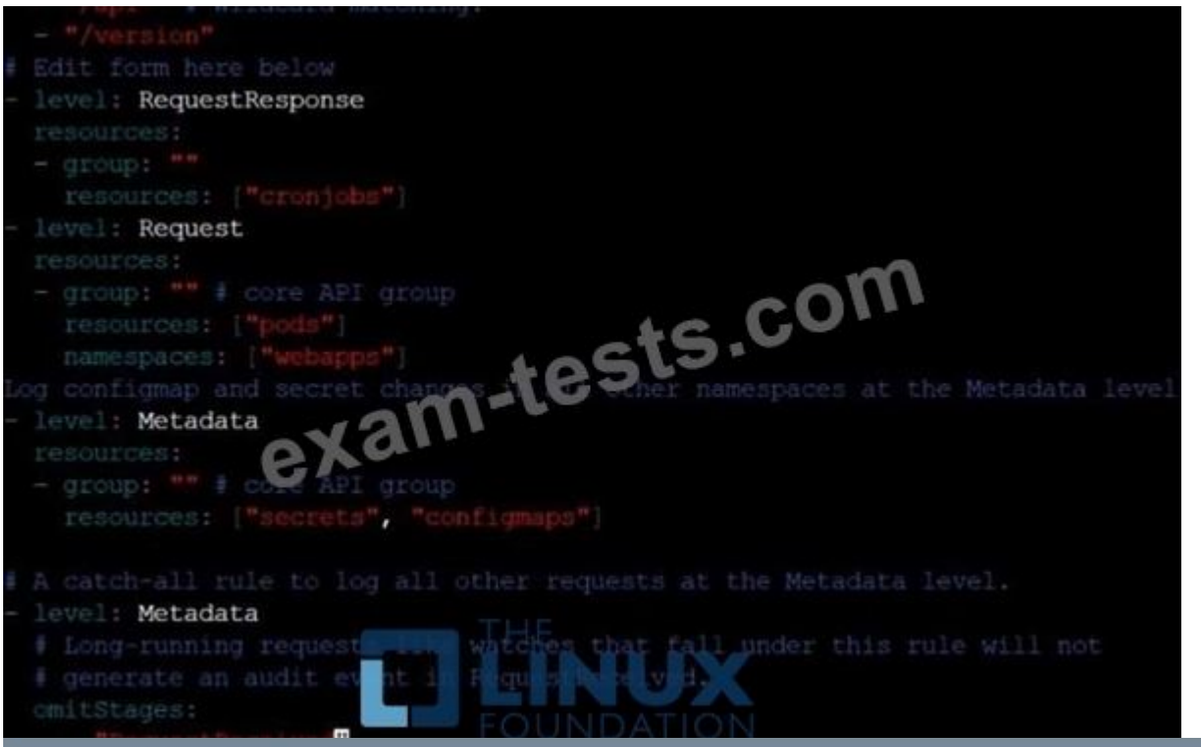
1. logs are stored at /var/log/kubernetes/kubernetes-logs.txt.
2. Log files are retained for 5 days.
3. at maximum, a number of 10 old audit logs files are retained.

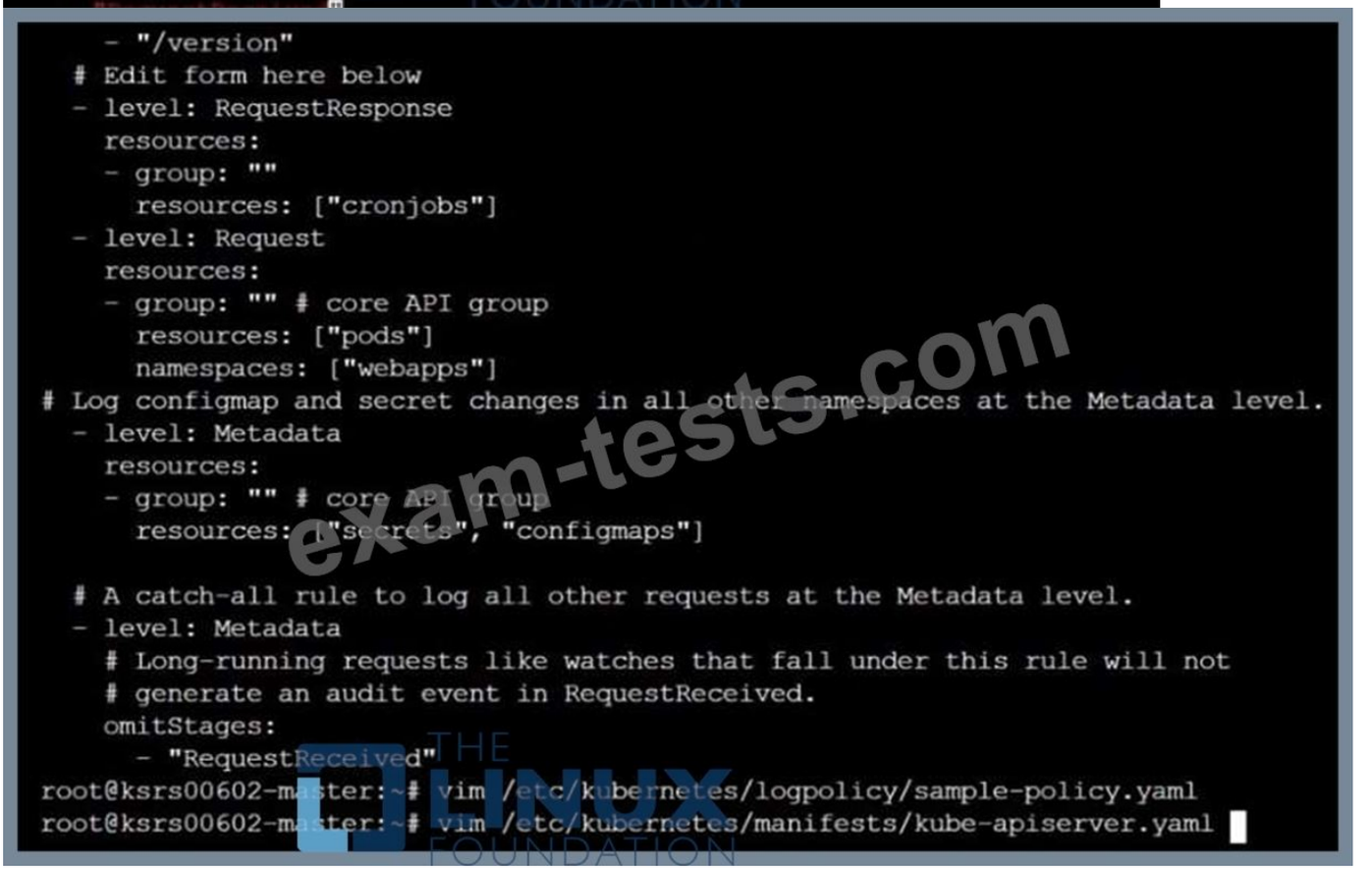
Edit and extend the basic policy to log:

1. Cronjobs changes at RequestResponse
2. Log the request body of deployments changes in the namespace kube-system.
3. Log all other resources in core and extensions at the Request level.
4. Don't log watch requests by the "system:kube-proxy" on endpoints or

Answer:

```
candidate@cli:~$ kubectl config use-context KSRS00602
Switched to context "KSRS00602".
candidate@cli:~$ ssh ksrs00602-master
Warning: Permanently added '10.240.86.243' (ECDSA) to the list of known hosts.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ksrs00602-master:~# cat /etc/kubernetes/logpolicy/sample-policy.yaml
---
apiVersion: audit.k8s.io/v1
kind: Policy
# Don't generate audit events for all requests in RequestReceived stage.
omitStages:
  - "RequestReceived"
rules:
  # Don't log watch requests by the "system:kube-proxy" on endpoints or services
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
    - group: "" # core API group
      resources: ["endpoints", "services"]

  # Don't log authenticated requests to certain non-resource URL paths.
  - level: None
    userGroups: ["system:authenticated"]
    nonResourceURLs:
    - "/api*" # Wildcard matching.
    - "/version"
  # Edit form here below
root@ksrs00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
```





NEW QUESTION: 18

SIMULATION

a. Retrieve the content of the existing secret named default-token-xxxxx in the testing namespace.

Store the value of the token in the token.txt

b. Create a new secret named test-db-secret in the DB namespace with the following content:

username: mysql

password: password@123

Create the Pod name test-db-pod of image nginx in the namespace db that can access test-db-secret via a volume at path /etc/mysql-credentials

Answer:

To add a Kubernetes cluster to your project, group, or instance:

Navigate to your:
Project's Operations > Kubernetes page, for a project-level cluster.
Group's Kubernetes page, for a group-level cluster.
Admin Area > Kubernetes page, for an instance-level cluster.
Click Add Kubernetes cluster.
Click the Add existing cluster tab and fill in the details:
Kubernetes cluster name (required) - The name you wish to give the cluster.
Environment scope (required) - The associated environment to this cluster.
API URL (required) - It's the URL that GitLab uses to access the Kubernetes API. Kubernetes exposes several APIs, we want the "base" URL that is common to all of them. For example, https://kubernetes.example.com rather than https://kubernetes.example.com/api/v1.
Get the API URL by running this command:
kubectl cluster-info | grep -E 'Kubernetes master|Kubernetes control plane' | awk '/http/ {print $NF}' CA certificate (required) - A valid Kubernetes certificate is needed to authenticate to the cluster. We use the certificate created by default.
List the secrets with kubectl get secrets, and one should be named similar to default-token-xxxxx. Copy that token name for use below.
Get the certificate by running this command:
kubectl get secret <secret name> -o jsonpath="{['data']['ca\.crt']}"

NEW QUESTION: 19
Create a Pod name Nginx-pod inside the namespace testing, Create a service for the Nginx-pod named nginx-svc, using the ingress of your choice, run the ingress on tls, secure port.
A. Send us your Feedback on this.
Answer: (SHOW ANSWER)

NEW QUESTION: 20
a. Retrieve the content of the existing secret named default-token-xxxxx in the testing namespace.
Store the value of the token in the token.txt
b. Create a new secret named test-db-secret in the DB namespace with the following content:
username: mysql
password: password@123
Create the Pod name test-db-pod of image nginx in the namespace db that can access test-db-secret via a volume at path /etc/mysql-credentials
Answer:
To add a Kubernetes cluster to your project, group, or instance:
Navigate to your:
Project's Operations > Kubernetes page, for a project-level cluster.
Group's Kubernetes page, for a group-level cluster.
Admin Area > Kubernetes page, for an instance-level cluster.

Click Add Kubernetes cluster.
Click the Add existing cluster tab and fill in the details:
Kubernetes cluster name (required) - The name you wish to give the cluster.
Environment scope (required) - The associated environment to this cluster.
API URL (required) - It's the URL that GitLab uses to access the Kubernetes API. Kubernetes exposes several APIs, we want the "base" URL that is common to all of them. For example, https://kubernetes.example.com rather than https://kubernetes.example.com/api/v1.
Get the API URL by running this command:
kubectl cluster-info | grep -E 'Kubernetes master|Kubernetes control plane' | awk '/http/ {print $NF}' CA certificate (required) - A valid Kubernetes certificate is needed to authenticate to the cluster. We use the certificate created by default.
List the secrets with kubectl get secrets, and one should be named similar to default-token-xxxxx. Copy that token name for use below.
Get the certificate by running this command:
kubectl get secret <secret name> -o jsonpath="{['data']['ca\.crt']}"

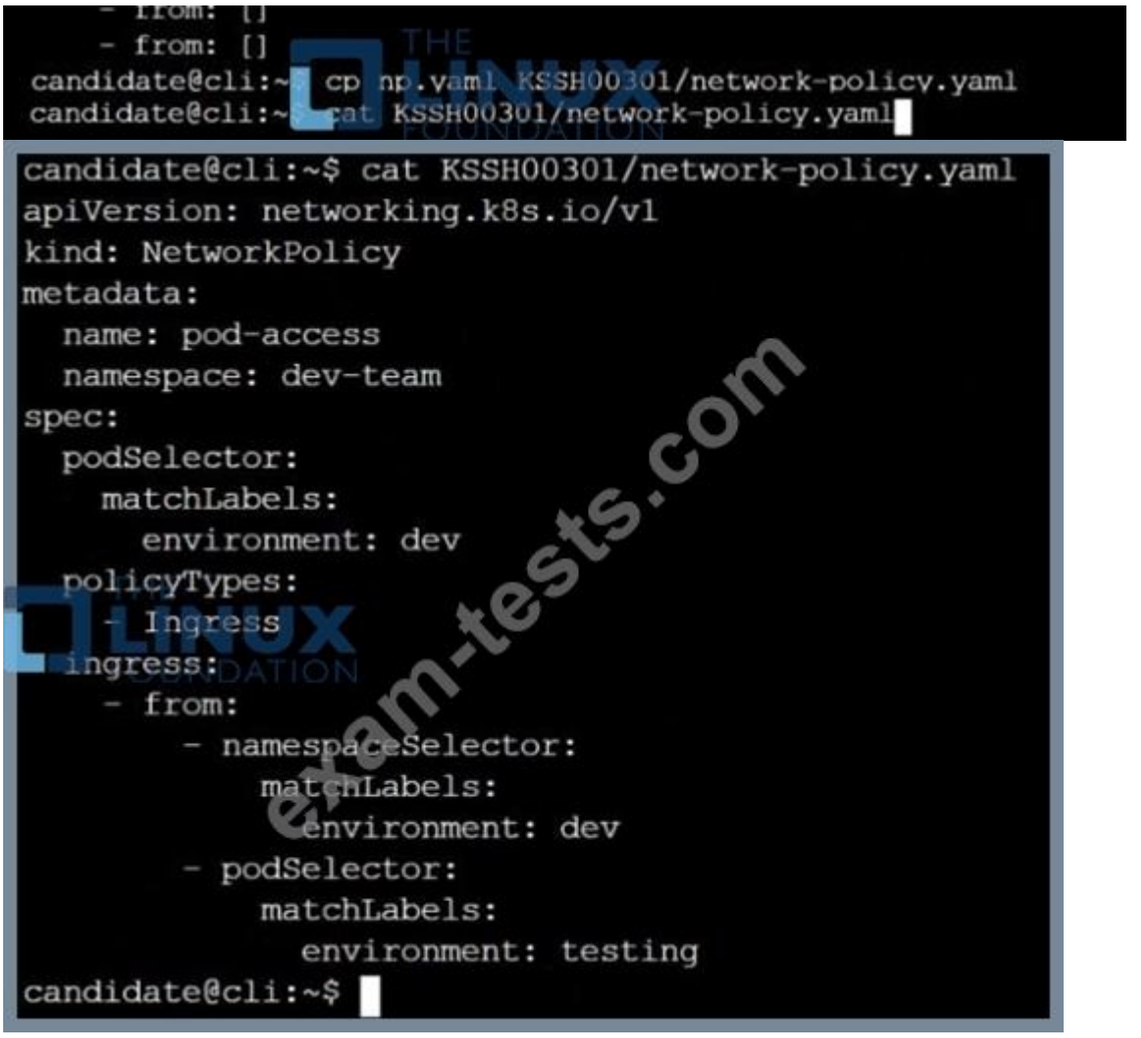
NEW QUESTION: 21
Task
Create a NetworkPolicy named pod-access to restrict access to Pod users-service running in namespace dev-team.
Only allow the following Pods to connect to Pod users-service:

Answer:

```
candidate@cli:~$ kubectl config use-context KSSH00301
Switched to context "KSSH00301".
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl get ns dev-team --show-labels
NAME       STATUS   AGE     LABELS
dev-team   Active   6h39m   environment=dev,kubernetes.io/metadata.name=dev-team
candidate@cli:~$ kubectl get pods -n dev-team --show-labels
NAME            READY   STATUS    RESTARTS   AGE     LABELS
users-service   1/1     Running   0          6h40m   environment=dev
candidate@cli:~$ ls
KSCH00301  KSMV00102  KSSC00301  KSSH00401     test-secret-pod.yaml
KSCS00101  KSMV00301  KSSH00301  password.txt  username.txt
candidate@cli:~$ vim np.yaml
```

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
```

```
candidate@cli:~$ vim np.yaml
candidate@cli:~$ cat np.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl create -f np.yaml -n dev-team
networkpolicy.networking.k8s.io/pod-access created
candidate@cli:~$ kubectl describe netpol -n dev-team
Name:         pod-access
Namespace:    dev-team
Created on:   2022-05-20 15:35:33 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     environment=dev
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: environment=dev
    From:
      PodSelector: environment=testing
  Not affecting egress traffic
  Policy Types: Ingress
candidate@cli:~$ cat KSSH00301/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ""
  namespace: ""
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
```



NEW QUESTION: 22

Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that

1. logs are stored at /var/log/kubernetes/kubernetes-logs.txt.
2. Log files are retained for 5 days.
3. at maximum, a number of 10 old audit logs files are retained.

Edit and extend the basic policy to log:

A. 1. Cronjobs changes at RequestResponse

Answer: A (LEAVE A REPLY)

2. Log the request body of deployments changes in the namespace kube-system.
3. Log all other resources in core and extensions at the Request level.
4. Don't log watch requests by the "system:kube-proxy" on endpoints or

NEW QUESTION: 23

Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.

A. Send us your Feedback on this.

Answer: A (LEAVE A REPLY)

NEW QUESTION: 24
Fix all issues via configuration and restart the affected components to ensure the new setting takes effect.
Fix all of the following violations that were found against the API server:- a. Ensure that the RotateKubeletServerCertificate argument is set to true.
b. Ensure that the admission control plugin PodSecurityPolicy is set.
c. Ensure that the --kubelet-certificate-authority argument is set as appropriate.
Fix all of the following violations that were found against the Kubelet:- a. Ensure the --anonymous-auth argument is set to false.
b. Ensure that the --authorization-mode argument is set to Webhook.
Fix all of the following violations that were found against the ETCD:-
a. Ensure that the --auto-tls argument is not set to true
b. Ensure that the --peer-auto-tls argument is not set to true
Hint: Take the use of Tool Kube-Bench
Answer:
Fix all of the following violations that were found against the API server:- a. Ensure that the RotateKubeletServerCertificate argument is set to true.
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
component: kubelet
tier: control-plane
name: kubelet
namespace: kube-system
spec:
containers:
- command:
- kube-controller-manager
+ - --feature-gates=RotateKubeletServerCertificate=true
image: gcr.io/google_containers/kubelet-amd64:v1.6.0
livenessProbe:
failureThreshold: 8
httpGet:
host: 127.0.0.1
path: /healthz
port: 6443
scheme: HTTPS
initialDelaySeconds: 15
timeoutSeconds: 15

name: kubelet
resources:
requests:
cpu: 250m
volumeMounts:
- mountPath: /etc/kubernetes/
name: k8s
readOnly: true
- mountPath: /etc/ssl/certs
name: certs
- mountPath: /etc/pki
name: pki
hostNetwork: true
volumes:
- hostPath:
path: /etc/kubernetes
name: k8s
- hostPath:
path: /etc/ssl/certs
name: certs
- hostPath:
path: /etc/pki
name: pki
b. Ensure that the admission control plugin PodSecurityPolicy is set.
audit: "/bin/ps -ef | grep $apiserverbin | grep -v grep"
tests:
test_items:
- flag: "--enable-admission-plugins"
compare:
op: has
value: "PodSecurityPolicy"
set: true
remediation: |
Follow the documentation and create Pod Security Policy objects as per your environment.
Then, edit the API server pod specification file $apiserverconf
on the master node and set the --enable-admission-plugins parameter to a value that includes
PodSecurityPolicy :
--enable-admission-plugins=...,PodSecurityPolicy,...
Then restart the API Server.
scored: true
c. Ensure that the --kubelet-certificate-authority argument is set as appropriate.

audit: "/bin/ps -ef | grep $apiserverbin | grep -v grep"
tests:
test_items:
- flag: "--kubelet-certificate-authority"
set: true
remediation: |
Follow the Kubernetes documentation and setup the TLS connection between the apiserver and kubelets. Then, edit the API server pod specification file
$apiserverconf on the master node and set the --kubelet-certificate-authority parameter to the path to the cert file for the certificate authority.
--kubelet-certificate-authority=<ca-string>
scored: true
Fix all of the following violations that were found against the ETCD:-
a. Ensure that the --auto-tls argument is not set to true
Edit the etcd pod specification file $etcdconf on the master node and either remove the --auto-tls parameter or set it to false. --auto-tls=false b. Ensure that the --peer-auto-tls argument is not set to true Edit the etcd pod specification file $etcdconf on the master node and either remove the --peer-auto-tls parameter or set it to false. --peer-auto-tls=false

NEW QUESTION: 25
Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that
1. logs are stored at /var/log/kubernetes-logs.txt.
2. Log files are retained for 12 days.
3. at maximum, a number of 8 old audit logs files are retained.
4. set the maximum size before getting rotated to 200MB
Edit and extend the basic policy to log:
1. namespaces changes at RequestResponse
2. Log the request body of secrets changes in the namespace kube-system.
3. Log all other resources in core and extensions at the Request level.
4. Log "pods/portforward", "services/proxy" at Metadata level.
5. Omit the Stage RequestReceived
All other requests at the Metadata level
Answer:
Kubernetes auditing provides a security-relevant chronological set of records about a cluster. Kube-apiserver performs auditing. Each request on each stage of its execution generates an event, which is then pre-processed according to a certain policy and written to a backend. The policy determines what's recorded and the backends persist the records.
You might want to configure the audit log as part of compliance with the CIS (Center for Internet Security) Kubernetes Benchmark controls.
The audit log can be enabled by default using the following configuration in cluster.yml:
services:

kube-api:
audit_log:
enabled: true
When the audit log is enabled, you should be able to see the default values at /etc/kubernetes/audit-policy.yaml The log backend writes audit events to a file in JSONlines format. You can configure the log audit backend using the following kube-apiserver flags:
--audit-log-path specifies the log file path that log backend uses to write audit events. Not specifying this flag disables log backend. - means standard out
--audit-log-maxage defined the maximum number of days to retain old audit log files
--audit-log-maxbackup defines the maximum number of audit log files to retain
--audit-log-maxsize defines the maximum size in megabytes of the audit log file before it gets rotated If your cluster's control plane runs the kube-apiserver as a Pod, remember to mount the hostPath to the location of the policy file and log file, so that audit records are persisted. For example:
--audit-policy-file=/etc/kubernetes/audit-policy.yaml \
--audit-log-path=/var/log/audit.log

NEW QUESTION: 26
Given an existing Pod named test-web-pod running in the namespace test-system Edit the existing Role bound to the Pod's Service Account named sa-backend to only allow performing get operations on endpoints.
Create a new Role named test-system-role-2 in the namespace test-system, which can perform patch operations, on resources of type statefulsets.
A. Create a new RoleBinding named test-system-role-2-binding binding the newly created Role to the Pod's ServiceAccount sa-backend.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 27
Context
A PodSecurityPolicy shall prevent the creation of privileged Pods in a specific namespace.
Task
Create a new PodSecurityPolicy named prevent-psp-policy,which prevents the creation of privileged Pods.
Create a new ClusterRole named restrict-access-role, which uses the newly created PodSecurityPolicy prevent-psp-policy.
Create a new ServiceAccount named psp-restrict-sa in the existing namespace staging.
Finally, create a new ClusterRoleBinding named restrict-access-bind, which binds the newly created ClusterRole restrict-access-role to the newly created ServiceAccount psp-restrict-sa.

You can find skeleton manifest files at:

- /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

Answer:

```
candidate@cli:~$ kubectl config use-context KSMV00102
Switched to context "KSMV00102".
candidate@cli:~$ cat /home/candidate/KSMV00102/pod-security-policy.yaml
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: ""
spec:
  seLinux:
    rule: ""
  runAsUser:
    rule: ""
  supplementalGroups: {}
  fsGroup: {}
candidate@cli:~$ vim /home/candidate/KSMV00102/pod-security-policy.yaml
```


```
candidate@cli:~$ vim /home/candidate/KSMV00102/pod-security-policy.yaml
candidate@cli:~$ cat /home/candidate/KSMV00102/pod-security-policy.yaml
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: "prevent-psp-policy"
spec:
  privileged: false
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/pod-security-policy.yaml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/prevent-psp-policy created
candidate@cli:~$ cat /home/candidate/KSMV00102/cluster-role.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ""
rules:
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
candidate@cli:~$ kubectl create clusterrole restrict-access-role --verb=use --resource=psp --dry-run=client -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: restrict-access-role
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

```
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
candidate@cli:~$ kubectl create clusterrole restrict-access-role --verb=use --resource=psp --dry-run=client --resource-name=prevent-psp-policy -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: restrict-access-role
rules:
- apiGroups:
  - policy
  resourceNames:
  - prevent-psp-policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-r
rules:
- apiGroups:
  - policy
  resourceNames:
  - prevent-psp-policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

```
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/cluster-role.yaml
clusterrole.rbac.authorization.k8s.io/restrict-access-role created
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ cat /home/candidate/KSMV00102/service-account.yaml
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "psp-restrict-sa"
  namespace: "staging"
```

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ""
  namespace: ""
candidate@cli:~$ vim /home/candidate/KSMV00102/service-account.yaml
candidate@cli:~$ cat /home/candidate/KSMV00102/service-account.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "psp-restrict-sa"
  namespace: "staging"
candidate@cli:~$ kubectl get sa -n staging
NAME      SECRETS   AGE
default   1         6h6m
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/service-account.yaml
serviceaccount/psp-restrict-sa created
candidate@cli:~$ kubectl get sa -n staging
NAME              SECRETS   AGE
default           1         6h6m
psp-restrict-sa   1         2s
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl create clusterrolebinding restrict-access-bind --clusterrole=restri
ct-access-role --serviceaccount=staging:psp-restrict-sa --dry-run -o yaml
W0520 14:41:23.502004   47827 helpers.go:598] --dry-run is deprecated and can be replaced wi
th --dry-run=client.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  creationTimestamp: null
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role
cluster-role-binding.yaml  cluster-role.yaml
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role
cluster-role-binding.yaml  cluster-role.yaml
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role-binding.yaml
```



```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging
```

NEW QUESTION: 28
SIMULATION
Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.
A. Send us your feedback on it.
Answer: (SHOW ANSWER)

NEW QUESTION: 29
Create a PSP that will only allow the persistentvolumeclaim as the volume type in the namespace restricted.
Create a new PodSecurityPolicy named prevent-volume-policy which prevents the pods which is having different volumes mount apart from persistentvolumeclaim.
Create a new ServiceAccount named psp-sa in the namespace restricted.
Create a new ClusterRole named psp-role, which uses the newly created Pod Security Policy prevent-volume-policy Create a new ClusterRoleBinding named psp-role-binding, which binds the created ClusterRole psp-role to the created SA psp-sa.
Hint:
Also, Check the Configuration is working or not by trying to Mount a Secret in the pod maifest, it should get failed.
POD Manifest:
apiVersion: v1
kind: Pod
metadata:
name:
spec:
containers:
- name:
image:

volumeMounts:
- name:
mountPath:
volumes:
- name:
secret:
secretname:

Answer:

apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
name: restricted
annotations:
seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default,runtime/default'
apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default'
seccomp.security.alpha.kubernetes.io/defaultProfileName: 'runtime/default'
apparmor.security.beta.kubernetes.io/defaultProfileName: 'runtime/default' spec:
privileged: false
Required to prevent escalations to root.
allowPrivilegeEscalation: false
This is redundant with non-root + disallow privilege escalation,
but we can provide it for defense in depth.
requiredDropCapabilities:
- ALL
Allow core volume types.
volumes:
- 'configMap'
- 'emptyDir'
- 'projected'
- 'secret'
- 'downwardAPI'
Assume that persistentVolumes set up by the cluster admin are safe to use.
- 'persistentVolumeClaim'
hostNetwork: false
hostIPC: false
hostPID: false
runAsUser:
Require the container to run without root privileges.
rule: 'MustRunAsNonRoot'
seLinux:
This policy assumes the nodes are using AppArmor rather than SELinux.

rule: 'RunAsAny'
supplementalGroups:
rule: 'MustRunAs'
ranges:
Forbid adding the root group.
- min: 1
max: 65535
fsGroup:
rule: 'MustRunAs'
ranges:
Forbid adding the root group.
- min: 1
max: 65535
readOnlyRootFilesystem: false

NEW QUESTION: 30
You can switch the cluster/configuration context using the following command:
[desk@cli] $ kubectl config use-context dev
A default-deny NetworkPolicy avoid to accidentally expose a Pod in a namespace that doesn't have any other NetworkPolicy defined.
Task: Create a new default-deny NetworkPolicy named deny-network in the namespace test for all traffic of type Ingress + Egress The new NetworkPolicy must deny all Ingress + Egress traffic in the namespace test.
Apply the newly created default-deny NetworkPolicy to all Pods running in namespace test.
You can find a skeleton manifests file at /home/cert_masters/network-policy.yaml
Answer:
master1 $ k get pods -n test --show-labels
NAME READY STATUS RESTARTS AGE LABELS
test-pod 1/1 Running 0 34s role=test,run=test-pod
testing 1/1 Running 0 17d run=testing
$ vim netpol.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: deny-network
namespace: test
spec:
podSelector: {}
policyTypes:
- Ingress
- Egress

master1 $ k apply -f netpol.yaml
Explanation
controlplane $ k get pods -n test --show-labels
NAME READY STATUS RESTARTS AGE LABELS
test-pod 1/1 Running 0 34s role=test,run=test-pod
testing 1/1 Running 0 17d run=testing
master1 $ vim netpol1.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: deny-network
namespace: test
spec:
podSelector: {}
policyTypes:
- Ingress
- Egress
master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/ Reference:
master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/ Explanation controlplane $ k get pods -n test --show-labels NAME READY STATUS RESTARTS AGE LABELS test-pod 1/1 Running 0 34s role=test,run=test-pod testing 1/1 Running 0 17d run=testing master1 $ vim netpol1.yaml apiVersion: networking.k8s.io/v1 kind: NetworkPolicy metadata:
name: deny-network
namespace: test
spec:
podSelector: {}
policyTypes:
- Ingress
- Egress
master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/ master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/

NEW QUESTION: 31
SIMULATION
Create a RuntimeClass named untrusted using the prepared runtime handler named runsc.
Create a Pods of image alpine:3.13.2 in the Namespace default to run on the gVisor runtime class.
Verify: Exec the pods and run the dmesg, you will see output like this:-

A. Send us your feedback on it.

Answer: (SHOW ANSWER)

NEW QUESTION: 32

Context:

Cluster: prod

Master node: master1

Worker node: worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $ kubectl config use-context prod

Task:

Analyse and edit the given Dockerfile (based on the ubuntu:18:04 image) /home/cert_masters/Dockerfile fixing two instructions present in the file being prominent security/best-practice issues.

Analyse and edit the given manifest file /home/cert_masters/mydeployment.yaml fixing two fields present in the file being prominent security/best-practice issues.

Note: Don't add or remove configuration settings; only modify the existing configuration settings, so that two configuration settings each are no longer security/best-practice concerns.

Should you need an unprivileged user for any of the tasks, use user nobody with user id 65535

Answer:

1. For Dockerfile: Fix the image version & user name in Dockerfile
2. For mydeployment.yaml : Fix security contexts

Explanation
[desk@cli] $ vim /home/cert_masters/Dockerfile
FROM ubuntu:latest # Remove this
FROM ubuntu:18.04 # Add this
USER root # Remove this
USER nobody # Add this
RUN apt get install -y lsof=4.72 wget=1.17.1 nginx=4.2
ENV ENVIRONMENT=testing
USER root # Remove this
USER nobody # Add this
CMD ["nginx -d"]

```
FROM ubuntu:latest  # Remove this
FROM ubuntu:18.04   # Add this
USER root           # Remove this
USER nobody         # Add this
RUN apt get install -y lsof=4.72 wget=1.17.1 nginx=4.2
ENV  ENVIRONMENT=testing
USER root           # Remove this
USER nobody         # Add this
CMD ["nginx -d"]
```


[desk@cli] $ vim /home/cert_masters/mydeployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
creationTimestamp: null
labels:
app: kafka
name: kafka
spec:
replicas: 1
selector:
matchLabels:
app: kafka
strategy: {}
template:
metadata:
creationTimestamp: null
labels:
app: kafka
spec:
containers:
- image: bitnami/kafka

name: kafka
volumeMounts:
- name: kafka-vol
mountPath: /var/lib/kafka
securityContext:
{"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": True,"readOnlyRootFilesystem": False, "runAsUser": 65535} # Delete This
{"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": False,"readOnlyRootFilesystem": True, "runAsUser": 65535} # Add This resources: {} volumes:
- name: kafka-vol
emptyDir: {}
status: {}
Pictorial View:
[desk@cli] $ vim /home/cert_masters/mydeployment.yaml

NEW QUESTION: 33
SIMULATION
Create a RuntimeClass named gvisor-rc using the prepared runtime handler named runsc.
Create a Pods of image Nginx in the Namespace server to run on the gVisor runtime class
Answer:
Install the Runtime Class for gVisor
{ # Step 1: Install a RuntimeClass
cat <<EOF | kubectl apply -f -
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
name: gvisor
handler: runsc
EOF

}
Create a Pod with the gVisor Runtime Class
{ # Step 2: Create a pod
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
name: nginx-gvisor
spec:
runtimeClassName: gvisor
containers:
- name: nginx
image: nginx
EOF
}
Verify that the Pod is running
{ # Step 3: Get the pod
kubectl get pod nginx-gvisor -o wide
}

NEW QUESTION: 34
You must complete this task on the following cluster/nodes: Cluster: immutable-cluster Master node: master1 Worker node: worker1 You can switch the cluster/configuration context using the following command:
[desk@cli] $ kubectl config use-context immutable-cluster
Context: It is best practice to design containers to be stateless and immutable.
Task:
Inspect Pods running in namespace prod and delete any Pod that is either not stateless or not immutable.
Use the following strict interpretation of stateless and immutable:
1. Pods being able to store data inside containers must be treated as not stateless.
Note: You don't have to worry whether data is actually stored inside containers or not already.
2. Pods being configured to be privileged in any way must be treated as potentially not stateless or not immutable.
Answer:
k get pods -n prod
k get pod <pod-name> -n prod -o yaml | grep -E 'privileged|ReadOnlyRootFileSystem' Delete the pods which do have any of these 2 properties privileged:true or ReadOnlyRootFileSystem: false
[desk@cli]$ k get pods -n prod
NAME READY STATUS RESTARTS AGE
cms 1/1 Running 0 68m

db 1/1 Running 0 4m
nginx 1/1 Running 0 23m
[desk@cli]$ k get pod nginx -n prod -o yaml | grep -E 'privileged|RootFileSystem'
{"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"creationTimestamp":null,"labels":
{"run":"nginx"},"name":"nginx","namespace":"prod"},"spec":{"containers":
[{"image":"nginx","name":"nginx","resources":{},"securityContext":
{"privileged":true}}],"dnsPolicy":"ClusterFirst","restartPolicy":"Always"},"status":{}} f:privileged: {}
privileged: true

[desk@cli]$ k delete pod nginx -n prod
[desk@cli]$ k get pod db -n prod -o yaml | grep -E 'privileged|RootFilesystem'

[desk@cli]$ k delete pod cms -n prod Reference: https://kubernetes.io/docs/concepts/policy/pod-security-policy/ https://cloud.google.com/architecture/best-practices-for-operating-containers
Reference:
[desk@cli]$ k delete pod cms -n prod Reference: https://kubernetes.io/docs/concepts/policy/pod-security-policy/ https://cloud.google.com/architecture/best-practices-for-operating-containers

NEW QUESTION: 35
You must complete this task on the following cluster/nodes:
Cluster: apparmor
Master node: master
Worker node: worker1
You can switch the cluster/configuration context using the following command:
[desk@cli] $ kubectl config use-context apparmor
Given: AppArmor is enabled on the worker1 node.
Task:
On the worker1 node,
1. Enforce the prepared AppArmor profile located at: /etc/apparmor.d/nginx
2. Edit the prepared manifest file located at /home/cert_masters/nginx.yaml to apply the apparmor profile
3. Create the Pod using this manifest
Answer:
[desk@cli] $ ssh worker1
[worker1@cli] $apparmor_parser -q /etc/apparmor.d/nginx
[worker1@cli] $aa-status | grep nginx
nginx-profile-1
[worker1@cli] $ logout
[desk@cli] $vim nginx-deploy.yaml
Add these lines under metadata:

annotations: # Add this line
container.apparmor.security.beta.kubernetes.io/<container-name>: localhost/nginx-profile-1
[desk@cli] $kubectl apply -f nginx-deploy.yaml
Explanation
[desk@cli] $ ssh worker1
[worker1@cli] $apparmor_parser -q /etc/apparmor.d/nginx
[worker1@cli] $aa-status | grep nginx
nginx-profile-1
[worker1@cli] $ logout
[desk@cli] $vim nginx-deploy.yaml

[desk@cli] $kubectl apply -f nginx-deploy.yaml pod/nginx-deploy created Reference: https://kubernetes.io/docs/tutorials/clusters/apparmor/ pod/nginx-deploy created
[desk@cli] $kubectl apply -f nginx-deploy.yaml pod/nginx-deploy created Reference: https://kubernetes.io/docs/tutorials/clusters/apparmor/

NEW QUESTION: 36
Create a User named john, create the CSR Request, fetch the certificate of the user after approving it.
Create a Role name john-role to list secrets, pods in namespace john
Finally, Create a RoleBinding named john-role-binding to attach the newly created role john-role to the user john in the namespace john. To Verify: Use the kubectl auth CLI command to verify the permissions.
Answer:
se kubectl to create a CSR and approve it.
Get the list of CSRs:
kubectl get csr
Approve the CSR:
kubectl certificate approve myuser
Get the certificate
Retrieve the certificate from the CSR:
kubectl get csr/myuser -o yaml
here are the role and role-binding to give john permission to create NEW_CRD resource:
kubectl apply -f roleBindingJohn.yaml --as=john
rolebinding.rbac.authorization.k8s.io/john_external-rosource-rb created kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1 metadata:
name: john_crd
namespace: development-john
subjects:

- kind: User
name: john
apiGroup: rbac.authorization.k8s.io
roleRef:
kind: ClusterRole
name: crd-creation
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: crd-creation
rules:
- apiGroups: ["kubernetes-client.io/v1"]
resources: ["NEW_CRD"]
verbs: ["create, list, get"]

NEW QUESTION: 37
Create a User named john, create the CSR Request, fetch the certificate of the user after approving it.
Create a Role name john-role to list secrets, pods in namespace john
Finally, Create a RoleBinding named john-role-binding to attach the newly created role john-role to the user john in the namespace john.
To Verify: Use the kubectl auth CLI command to verify the permissions.
Answer:
se kubectl to create a CSR and approve it.
Get the list of CSRs:
kubectl get csr
Approve the CSR:
kubectl certificate approve myuser
Get the certificate
Retrieve the certificate from the CSR:
kubectl get csr/myuser -o yaml
here are the role and role-binding to give john permission to create NEW_CRD resource:
kubectl apply -f roleBindingJohn.yaml --as=john
rolebinding.rbac.authorization.k8s.io/john_external-rosource-rb created kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1 metadata:
name: john_crd
namespace: development-john
subjects:
- kind: User
name: john
apiGroup: rbac.authorization.k8s.io

roleRef:
kind: ClusterRole
name: crd-creation
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: crd-creation
rules:
- apiGroups: ["kubernetes-client.io/v1"]
resources: ["NEW_CRD"]
verbs: ["create, list, get"]