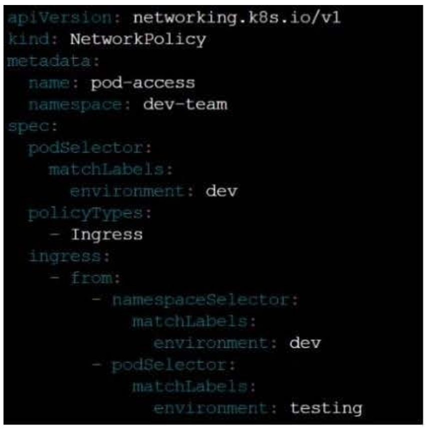
CKS$^{Q\&As}$

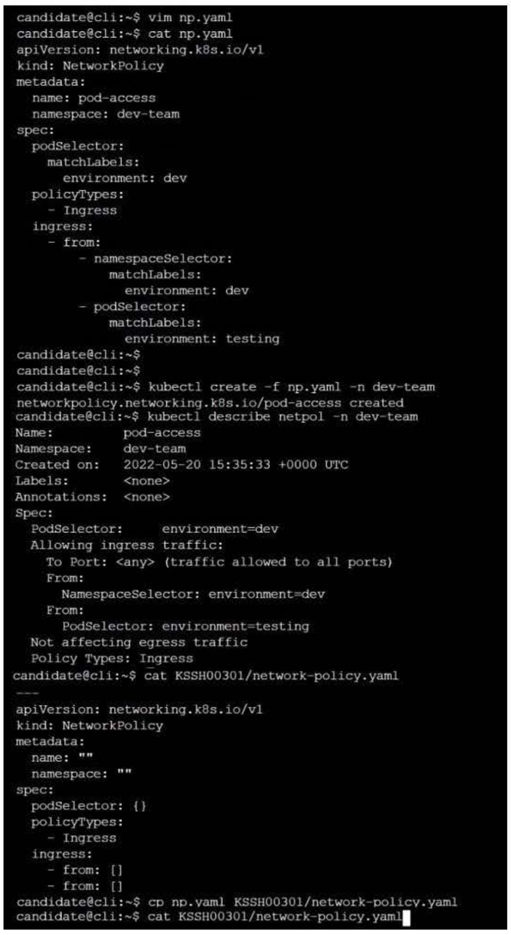
Certified Kubernetes Security Specialist (CKS) Exam

Pass Linux Foundation CKS Exam with 100% Guarantee

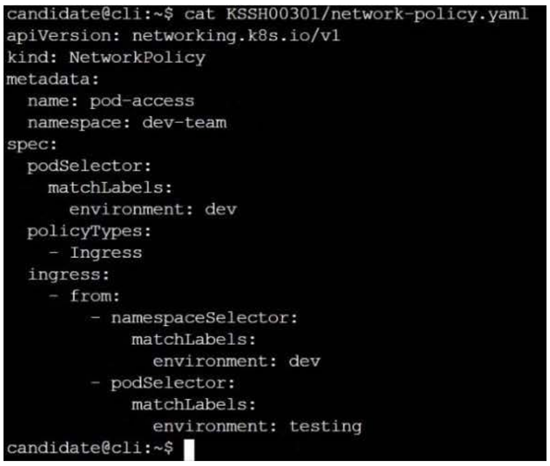
QUESTION 1



```
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:
    - from: []
    - from: []
candidate@cli:~$ cp np.yaml KSSH00301/network-policy.yaml
candidate@cli:~$ cat KSSH00301/network-policy.yaml
```



```
candidate@cli:~$ cat KSSH00301/network-policy.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:~$
```

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

A. See the explanation below:

B. PlaceHolder

Correct Answer: A

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

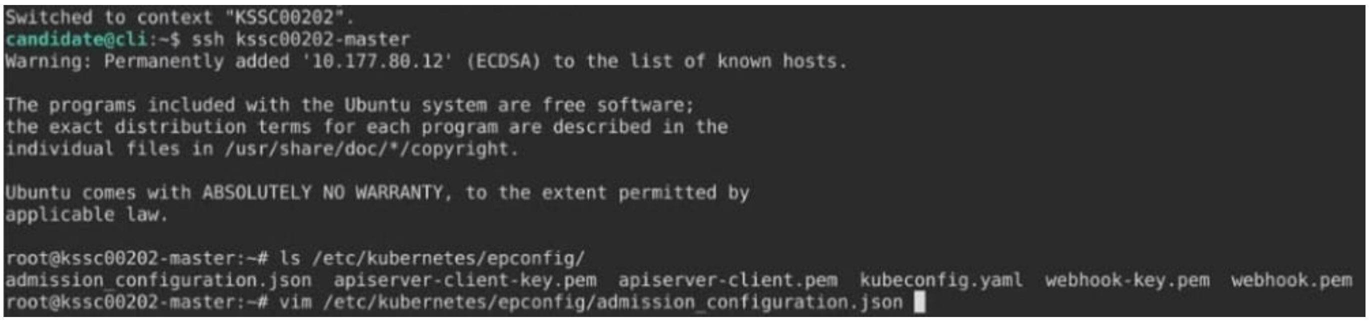
If your cluster\\'s control plane runs the kube-apiserver as a Pod, remember to mount the hostPath to the location of the policy file and log file, so that audit records are persisted.

For example:

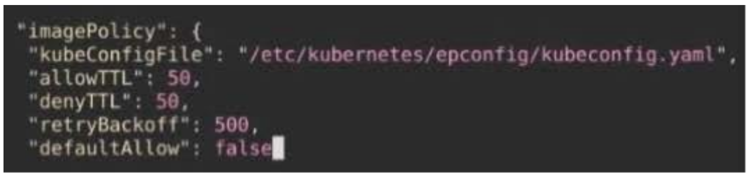
--audit-policy-file=/etc/kubernetes/audit-policy.yaml \

--audit-log-path=/var/log/audit.log

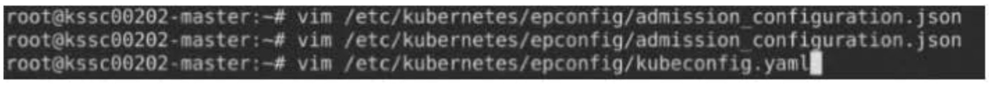
QUESTION 2



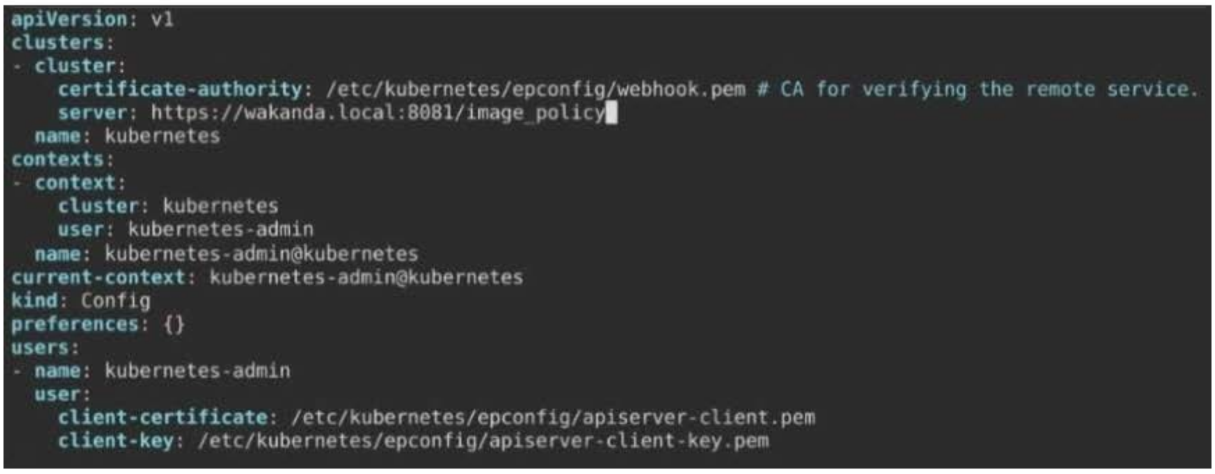
```
Switched to context "KSSC00202".
candidate@cli:~$ ssh kssc00202-master
Warning: Permanently added '10.177.80.12' (ECDSA) to the list of known hosts.

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@kssc00202-master:~# ls /etc/kubernetes/epconfig/
admission_configuration.json  apiserver-client-key.pem  apiserver-client.pem  kubeconfig.yaml  webhook-key.pem  webhook.pem
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
```



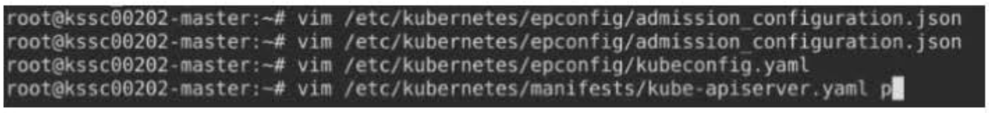
```
"imagePolicy": {
 "kubeConfigFile": "/etc/kubernetes/epconfig/kubeconfig.yaml",
 "allowTTL": 50,
 "denyTTL": 50,
 "retryBackoff": 500,
 "defaultAllow": false
```



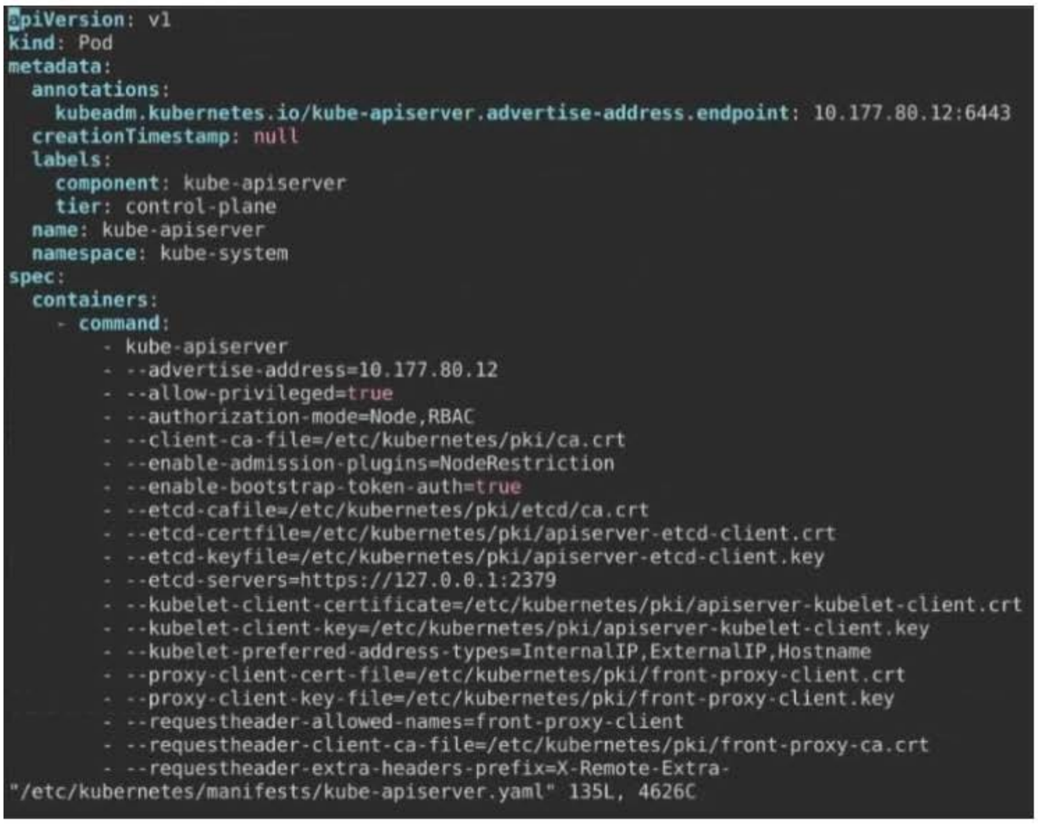
```
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/kubeconfig.yaml
```



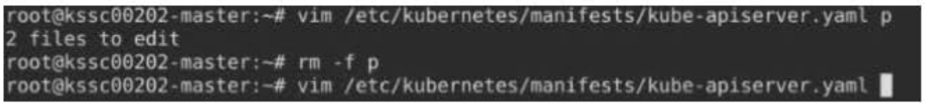
```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /etc/kubernetes/epconfig/webhook.pem # CA for verifying the remote service.
    server: https://wakanda.local:8081/image_policy
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate: /etc/kubernetes/epconfig/apiserver-client.pem
    client-key: /etc/kubernetes/epconfig/apiserver-client-key.pem
```



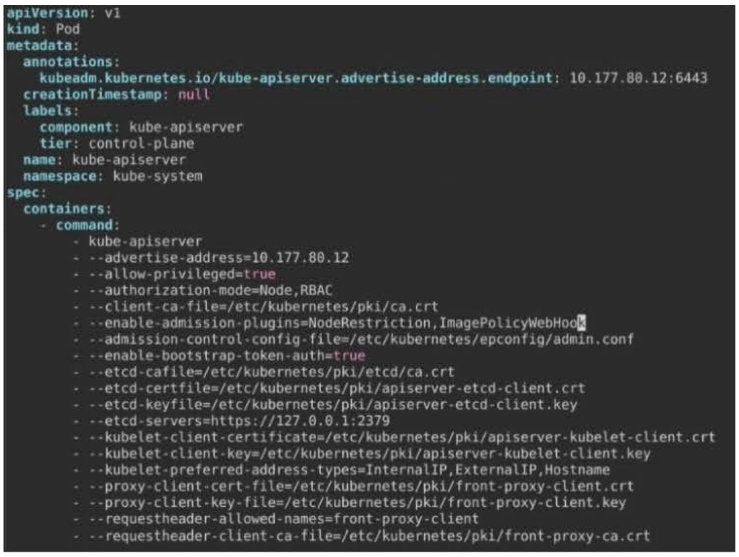
```
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/kubeconfig.yaml
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml p
```



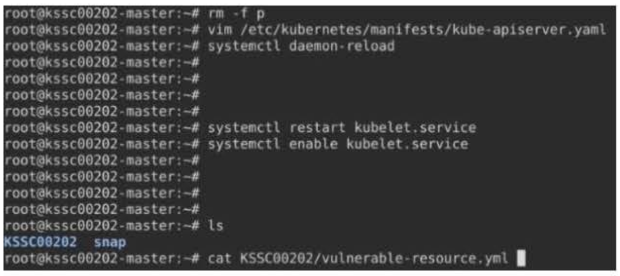
```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.177.80.12:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.177.80.12
    - --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
    - --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-
"/etc/kubernetes/manifests/kube-apiserver.yaml" 135L, 4626C
```



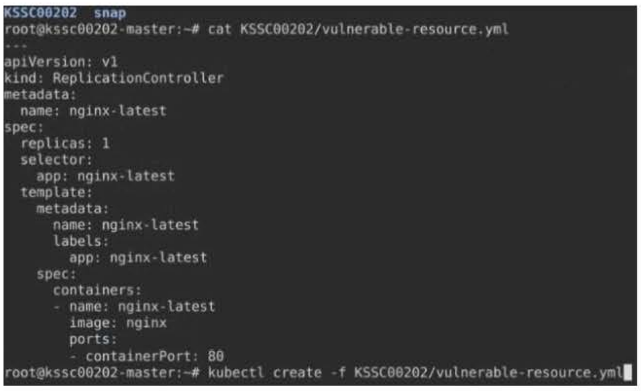
```
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml p
2 files to edit
root@kssc00202-master:~# rm -f p
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
```



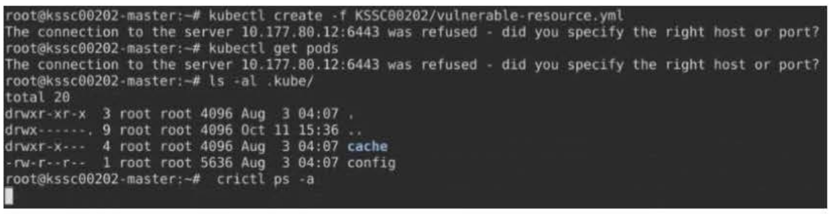
```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.177.80.12:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.177.80.12
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction,ImagePolicyWebHook
    - --admission-control-config-file=/etc/kubernetes/epconfig/admin.conf
    - --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
    - --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
```



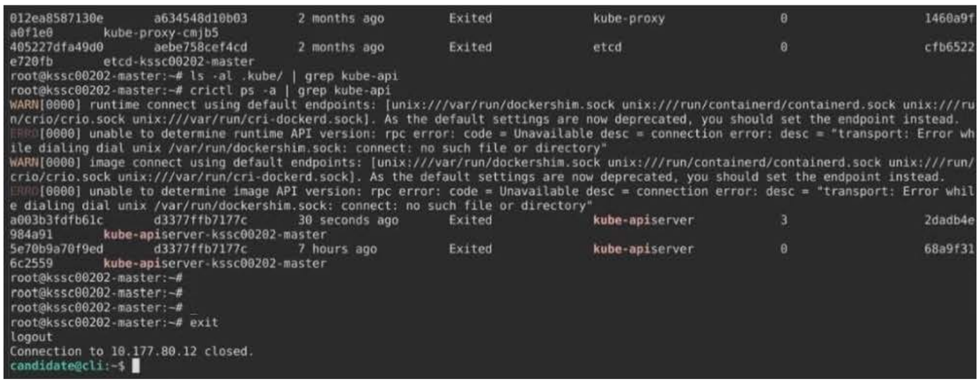
```
root@kssc00202-master:~# rm -f p
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@kssc00202-master:~# systemctl daemon-reload
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# systemctl restart kubelet.service
root@kssc00202-master:~# systemctl enable kubelet.service
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# ls
KSSC00202  snap
root@kssc00202-master:~# cat KSSC00202/vulnerable-resource.yml
```



```
KSSC00202  snap
root@kssc00202-master:~# cat KSSC00202/vulnerable-resource.yml
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-latest
spec:
  replicas: 1
  selector:
    app: nginx-latest
  template:
    metadata:
      name: nginx-latest
      labels:
        app: nginx-latest
    spec:
      containers:
      - name: nginx-latest
        image: nginx
        ports:
        - containerPort: 80
root@kssc00202-master:~# kubectl create -f KSSC00202/vulnerable-resource.yml
```



```
root@kssc00202-master:~# kubectl create -f KSSC00202/vulnerable-resource.yml
The connection to the server 10.177.80.12:6443 was refused - did you specify the right host or port?
root@kssc00202-master:~# kubectl get pods
The connection to the server 10.177.80.12:6443 was refused - did you specify the right host or port?
root@kssc00202-master:~# ls -al .kube/
total 20
drwxr-xr-x  3 root root 4096 Aug  3 04:07 .
drwx------. 9 root root 4096 Oct 11 15:36 ..
drwxr-x---  4 root root 4096 Aug  3 04:07 cache
-rw-r--r--  1 root root 5636 Aug  3 04:07 config
root@kssc00202-master:~#  crictl ps -a
```



```
012ea8587130e       a634548d10b03       2 months ago        Exited              kube-proxy          0                   1460a9f
a0f1e0       kube-proxy-cmjb5
405227dfa49d0       aebe758cef4cd       2 months ago        Exited              etcd                0                   cfb6522
e720fb       etcd-kssc00202-master
root@kssc00202-master:~# ls -al .kube/ | grep kube-api
root@kssc00202-master:~# crictl ps -a | grep kube-api
WARN[0000] runtime connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:///ru
n/crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint instead.
ERRO[0000] unable to determine runtime API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Error wh
ile 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 instead.
ERRO[0000] unable to determine image API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Error whil
e dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
a003b3fdfb61c       d3377ffb7177c       30 seconds ago      Exited              kube-apiserver      3                   2dadb4e
984a91       kube-apiserver-kssc00202-master
5e70b9a70f9ed       d3377ffb7177c       7 hours ago         Exited              kube-apiserver      0                   68a9f31
6c2559       kube-apiserver-kssc00202-master
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# _
root@kssc00202-master:~# exit
logout
Connection to 10.177.80.12 closed.
candidate@cli:~$
```

Cluster: dev Master node: master1 Worker node: worker1 You can switch the cluster/configuration context using the following command: [desk@cli] $ kubectl config use-context dev Task:

Retrieve the content of the existing secret named adam in the safe namespace.

Store the username field in a file names /home/cert-masters/username.txt, and the password field in a file named /home/cert-masters/password.txt.

1.

You must create both files; they don\\'t exist yet.

2.

Do not use/modify the created files in the following steps, create new temporary files if needed.

Create a new secret names newsecret in the safe namespace, with the following content:

Username: dbadmin Password: moresecurepas

Finally, create a new Pod that has access to the secret newsecret via a volume:

Namespace:safe Pod name:mysecret-pod Container name:db-container Image:redis Volume name:secret-vol Mount path:/etc/mysecret

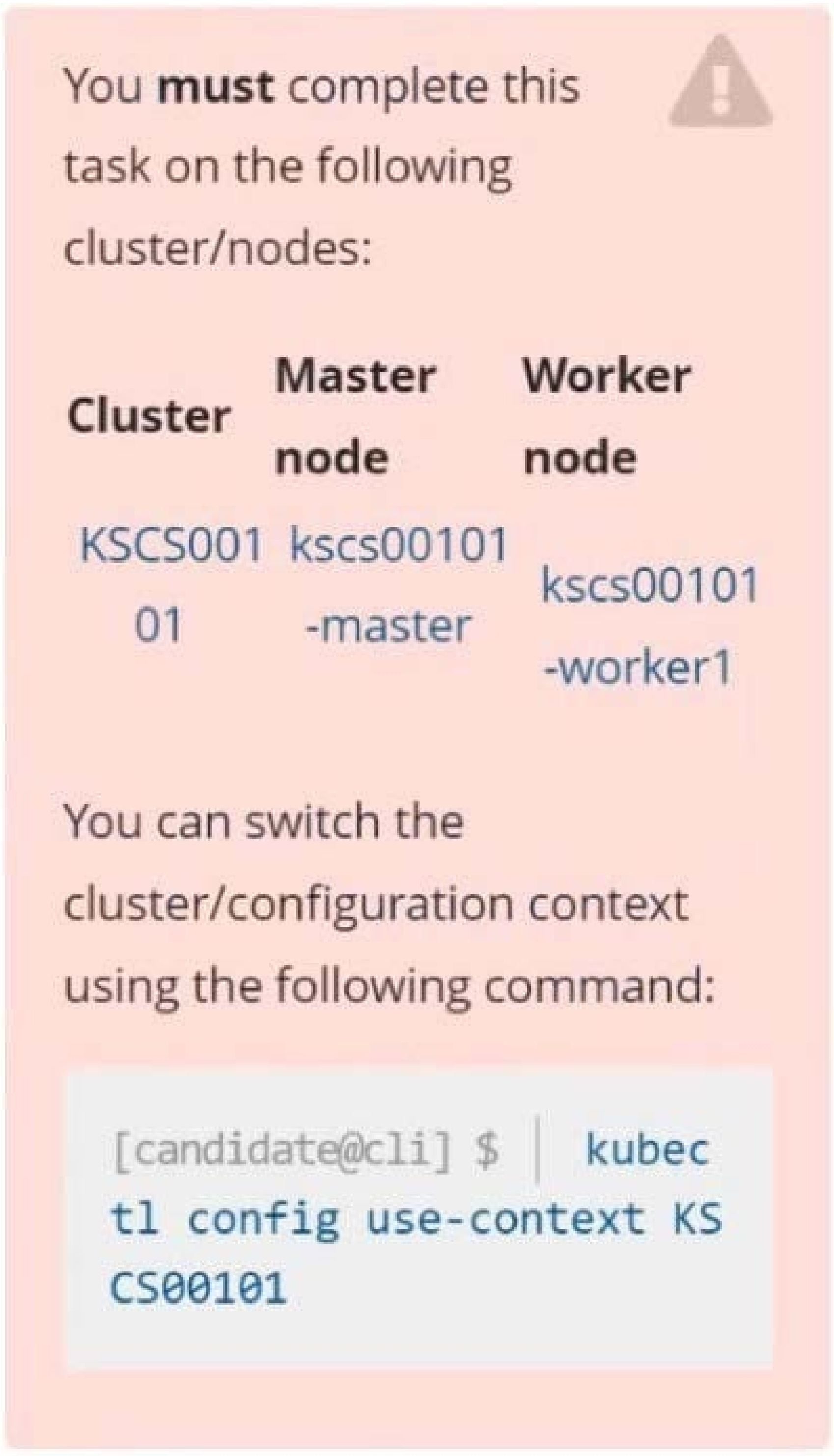
A. See the explanation below

B. PlaceHolder

Correct Answer: A

QUESTION 3

CORRECT TEXT Context



You **must** complete this task on the following cluster/nodes:

Cluster	Master node	Worker node
KSCS00101	kscs00101-master	kscs00101-worker1

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

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

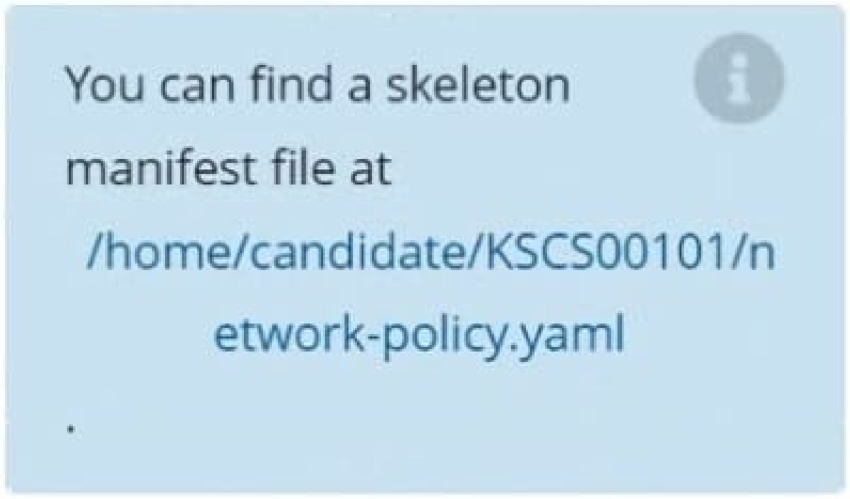
A default-deny NetworkPolicy avoids to accidentally expose a Pod in a namespace that doesn\\'t have any other NetworkPolicy defined.

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.


You can find a skeleton manifest file at /home/candidate/KSCS00101/network-policy.yaml
.

A. See explanation below.

B. PlaceHolder

Correct Answer: A

QUESTION 4

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.

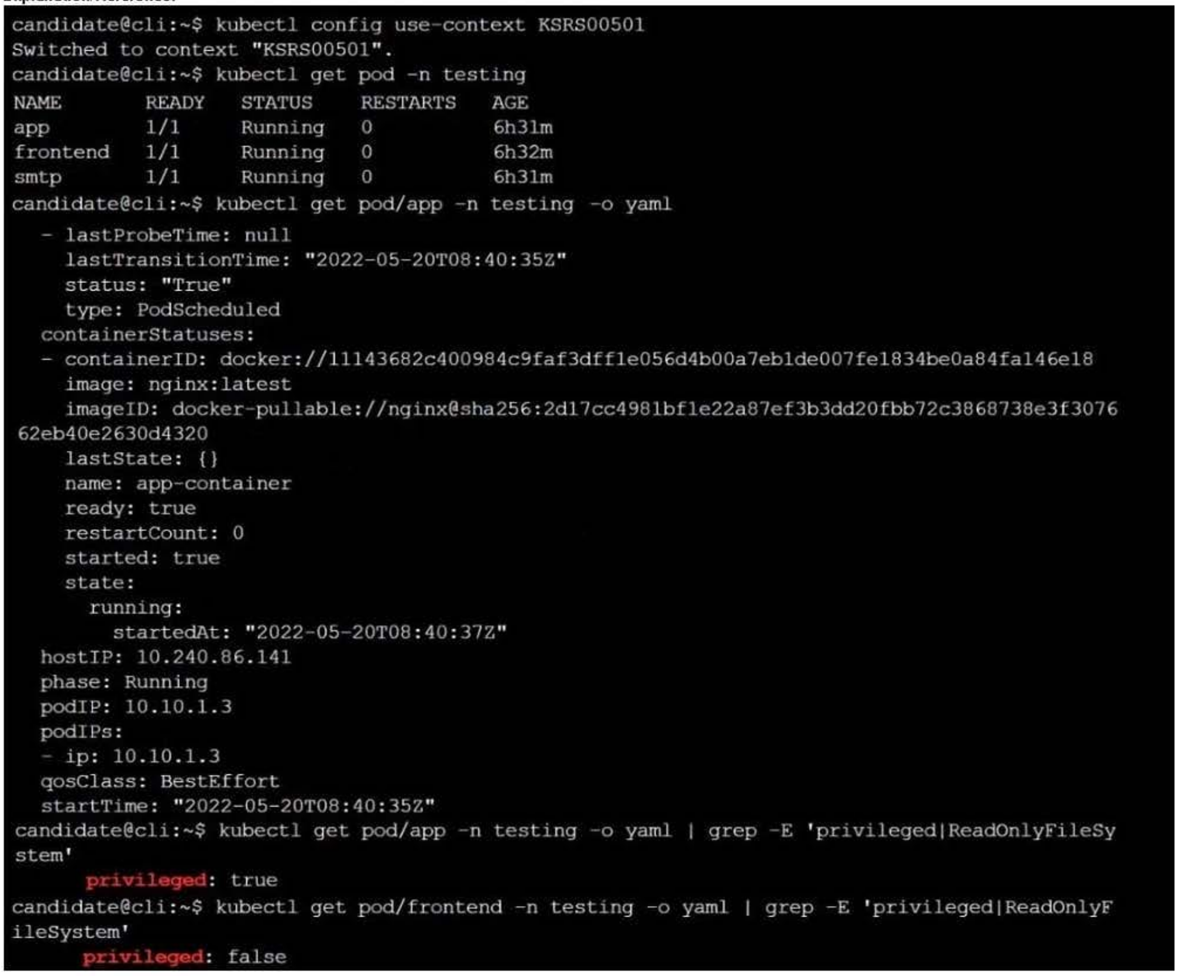
Pods being configured to be privileged in any way must be treated as potentially not stateless or not immutable.

A. See the explanation below

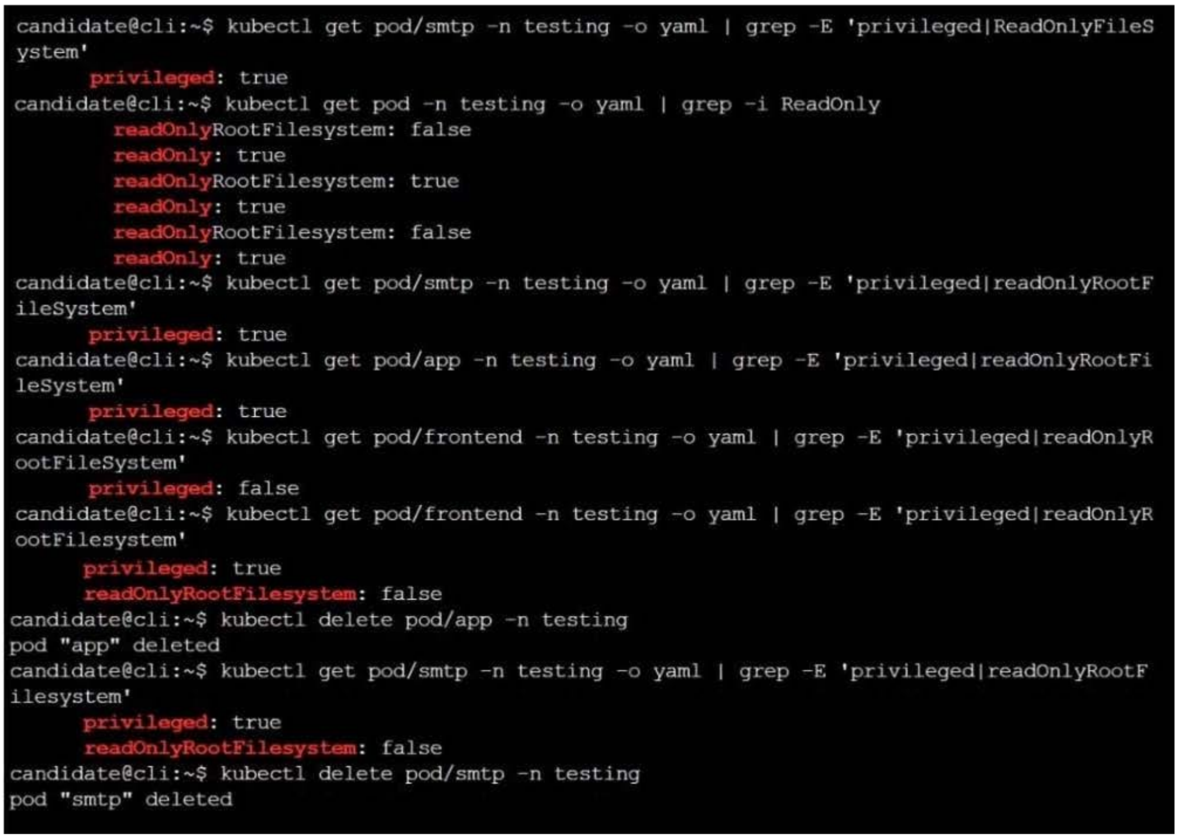
B. PlaceHolder

Correct Answer: A

Explanation/Reference:



```
candidate@cli:~$ kubectl config use-context KSRS00501
Switched to context "KSRS00501".
candidate@cli:~$ kubectl get pod -n testing
NAME       READY   STATUS    RESTARTS   AGE
app        1/1     Running   0          6h31m
frontend   1/1     Running   0          6h32m
smtp       1/1     Running   0          6h31m
candidate@cli:~$ kubectl get pod/app -n testing -o yaml
  - lastProbeTime: null
    lastTransitionTime: "2022-05-20T08:40:35Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID: docker://11143682c400984c9faf3dff1e056d4b00a7eb1de007fe1834be0a84fa146e18
    image: nginx:latest
    imageID: docker-pullable://nginx@sha256:2d17cc4981bf1e22a87ef3b3dd20fbb72c3868738e3f3076
62eb40e2630d4320
    lastState: {}
    name: app-container
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2022-05-20T08:40:37Z"
  hostIP: 10.240.86.141
  phase: Running
  podIP: 10.10.1.3
  podIPs:
  - ip: 10.10.1.3
  qosClass: BestEffort
  startTime: "2022-05-20T08:40:35Z"
candidate@cli:~$ kubectl get pod/app -n testing -o yaml | grep -E 'privileged|ReadOnlyFileSy
stem'
      privileged: true
candidate@cli:~$ kubectl get pod/frontend -n testing -o yaml | grep -E 'privileged|ReadOnlyF
ileSystem'
      privileged: false
```



```
candidate@cli:~$ kubectl get pod/smtp -n testing -o yaml | grep -E 'privileged|ReadOnlyFileS
ystem'
      privileged: true
candidate@cli:~$ kubectl get pod -n testing -o yaml | grep -i ReadOnly
        readOnlyRootFilesystem: false
        readOnly: true
        readOnlyRootFilesystem: true
        readOnly: true
        readOnlyRootFilesystem: false
        readOnly: true
candidate@cli:~$ kubectl get pod/smtp -n testing -o yaml | grep -E 'privileged|readOnlyRootF
ileSystem'
      privileged: true
candidate@cli:~$ kubectl get pod/app -n testing -o yaml | grep -E 'privileged|readOnlyRootFi
leSystem'
      privileged: true
candidate@cli:~$ kubectl get pod/frontend -n testing -o yaml | grep -E 'privileged|readOnlyR
ootFileSystem'
      privileged: false
candidate@cli:~$ kubectl get pod/frontend -n testing -o yaml | grep -E 'privileged|readOnlyR
ootFilesystem'
      privileged: true
      readOnlyRootFilesystem: false
candidate@cli:~$ kubectl delete pod/app -n testing
pod "app" deleted
candidate@cli:~$ kubectl get pod/smtp -n testing -o yaml | grep -E 'privileged|readOnlyRootF
ilesystem'
      privileged: true
      readOnlyRootFilesystem: false
candidate@cli:~$ kubectl delete pod/smtp -n testing
pod "smtp" deleted
```

QUESTION 5

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:

1.

apiVersion: v1

2.

kind: Pod

3.

metadata:

4.

name:

5.

spec:

6.

containers:

7.

- name:

8.

image:

9.

volumeMounts: 10.- name: 11.mountPath: 12.volumes: 13.- name: 14.secret: 15.secretName:

A. See the below:

B. PlaceHolder

Correct Answer: A

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
```