

# DUMPSBOSS.

**Certified Kubernetes Security Specialist (CKS)**

Linux Foundation CKS

Version Demo

**Total Demo Questions: 10**

**Total Premium Questions: 48**

Buy Premium PDF

<https://dumpsboss.co>

[support@dumpsboss.co](mailto:support@dumpsboss.co)

support@dumpsboss.co  
dumpsboss.co

## QUESTION NO: 1 - (SIMULATION)

You can switch the cluster/configuration context using the following command:`[desk@cli] $ kubectl config use-context test-account`  
Task: 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 5 days
3. at maximum, a number of 10 old audit log files are retained

A basic policy is provided at `/etc/Kubernetes/logpolicy/audit-policy.yaml`. It only specifies what not to log. Note: The base policy is located on the cluster's master node.

Edit and extend the basic policy to log:  
1. Nodes changes at RequestResponse level  
2. The request body of persistentvolumes changes in the namespace frontend  
3. ConfigMap and Secret changes in all namespaces at the Metadata level

Also, add a catch-all rule to log all other requests at the Metadata level  
Note: Don't forget to apply the modified policy.

## ANSWER: See the explanation below

### Explanation:

```
$ vim /etc/kubernetes/log-policy/audit-policy.yaml
```

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

Add these

```
- --audit-log-maxbackup=10
```

Explanation

```
[desk@cli] $ ssh master1 [master1@cli] $ vim /etc/kubernetes/log-policy/audit-policy.yaml
```

```
apiVersion: audit.k8s.io/v1 # This is required.
```

```
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"

# Add your changes below

- level: RequestResponse

userGroups: ["system:nodes"] # Block for nodes

- level: Request

resources:

- group: "" # core API group

resources: ["persistentvolumes"] # Block for persistentvolumes

namespaces: ["frontend"] # Block for persistentvolumes of frontend ns

- level: Metadata

resources:

- group: "" # core API group

resources: ["configmaps", "secrets"] # Block for configmaps & secrets

- level: Metadata # Block for everything else

[master1@cli] \$ vim /etc/kubernetes/manifests/kube-apiserver.yaml

apiVersion: v1

kind: Pod

metadata:

annotations:

kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.0.0.5:6443

labels:

component: kube-apiserver

tier: control-plane

name: kube-apiserver

namespace: kube-system

spec:

containers:

- command:

- kube-apiserver

- --advertise-address=10.0.0.5

- --allow-privileged=true

- --authorization-mode=Node,RBAC

- --audit-policy-file=/etc/kubernetes/log-policy/audit-policy.yaml #Add this

- --audit-log-path=/var/log/kubernetes/logs.txt #Add this

- --audit-log-maxage=5 #Add this

- --audit-log-maxbackup=10 #Add this

...

output truncated

Note: log volume & policy volume is already mounted in vim /etc/kubernetes/manifests/kube-apiserver.yaml so no need to mount it. Reference: <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>

## QUESTION NO: 2 - (SIMULATION)

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

Edit and extend the basic policy to log:

**ANSWER: See explanation below.**

**Explanation:**

```
candidate@cli:~$ kubectl config use-context KRSR00602
Switched to context "KRSR00602".
candidate@cli:~$ ssh krsr00602-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@krsr00602-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@krsr00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
```

```
- "/api*" # Wildcard matching
- "/version"
# Edit form here below
- level: RequestResponse
  resources:
  - group: ""
    resources: ["cronjobs"]
- level: Request
  resources:
  - group: "" # core API group
    resources: ["pods"]
    namespaces: ["webapps"]
# Log configmap and secret changes in all other namespaces at the Metadata level.
- level: Metadata
  resources:
  - group: "" # core API group
    resources: ["secrets", "configmaps"]
# A catch-all rule to log all other requests at the Metadata level.
- level: Metadata
# Long-running requests like watches that fall under this rule will not
# generate an audit event in RequestReceived.
omitStages:
- "RequestReceived"
```

```
- "/version"
# Edit form here below
- level: RequestResponse
  resources:
  - group: ""
    resources: ["cronjobs"]
- level: Request
  resources:
  - group: "" # core API group
    resources: ["pods"]
    namespaces: ["webapps"]
# Log configmap and secret changes in all other namespaces at the Metadata level.
- level: Metadata
  resources:
  - group: "" # core API group
    resources: ["secrets", "configmaps"]
# A catch-all rule to log all other requests at the Metadata level.
- level: Metadata
# Long-running requests like watches that fall under this rule will not
# generate an audit event in RequestReceived.
omitStages:
- "RequestReceived"
root@ksrs00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
root@ksrs00602-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
labels:
  component: kube-apiserver
  tier: control-plane
name: kube-apiserver
namespace: kube-system
spec:
  containers:
  - name: kube-apiserver
    args:
    - --advertise-address=10.240.86.243
    - --allow-privileged=true
    - --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml
    - --audit-log-path=/var/log/kubernetes/kubernetes-logs.txt
    - --audit-log-maxbackup=1
    - --audit-log-maxage=30
    - --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
```

```
# A catch-all rule to log all other requests at the Metadata level.
- level: Metadata
  # Long-running requests like watches that fall under this rule will not
  # generate an audit event in RequestReceived.
  omitStages:
  - "RequestReceived"
root@ksrs00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
root@ksrs00602-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@ksrs00602-master:~# systemctl daemon-reload
root@ksrs00602-master:~# systemctl restart kubelet.service
root@ksrs00602-master:~# systemctl enable kubelet
root@ksrs00602-master:~# exit
logout
Connection to 10.240.86.243 closed.
candidate@cli:~$
```

## QUESTION NO: 3 - (SIMULATION)

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: See the explanation below**

## Explanation:

```
candidate@cli:~$ kubectl config use-context KSR500501
Switched to context "KSR500501".
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://11143682c400984c9faf3dff1e056d4b00a7ebide907fe1834be0a84fa146e18
    image: nginx:latest
    imageID: docker-pullable://nginx@sha256:2d11cc4981bf1e22a87ef3b3dd20fbb72c3868738e3f3076
    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|ReadOnlyFileSyste
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|readOnlyRootFi
leSystem'
  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
```

Reference: <https://kubernetes.io/docs/concepts/policy/pod-security-policy/https://cloud.google.com/architecture/best-practices-for-operating-containers>

## QUESTION NO: 4 - (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: Seetheexplanationbelow:**

### Explanation:

{ # Step 1: Install a RuntimeClass

cat <

apiVersion: node.k8s.io/v1beta1

kind: RuntimeClass

metadata:

name: gvisor

handler: runsc

```
EOF
}
{ # Step 2: Create a pod
cat <
apiVersion: v1
kind: Pod
metadata:
name: nginx-gvisor
spec:
runtimeClassName: gvisor
containers:
- name: nginx
image: nginx
EOF
}
{ # Step 3: Get the pod
kubectl get pod nginx-gvisor -o wide
}
```

## QUESTION NO: 5 - (SIMULATION)

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



Cluster	Master node	Worker node
KSCH00301	ksch00301 -master	ksch00301 -worker1

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

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

Context

Your organization's security policy includes:

The Pod specified in the manifest file `/home/candidate/KSCH00301 /pod-manifest.yaml` fails to schedule because of an incorrectly specified ServiceAccount.

Complete the following tasks:

Task

1. Create a new ServiceAccount named `frontend-sa` in the existing namespace `qa`. Ensure the ServiceAccount does not automount API credentials.
2. Using the manifest file at `/home/candidate/KSCH00301 /pod-manifest.yaml`, create the Pod.
3. Finally, clean up any unused ServiceAccounts in namespace `qa`.

**ANSWER: See the explanation below**

Explanation:

```
Switched to context "KSCH00301".
candidate@cli:~$ kubectl get sa -n qa
NAME          SECRETS  AGE
default       1        5h46m
podrunner     1        5h46m
candidate@cli:~$ kubectl get deployment -n qa
No resources found in qa namespace.
candidate@cli:~$ kubectl get pod -n qa
No resources found in qa namespace.
candidate@cli:~$ kubectl create sa frontend-sa -n qa
serviceaccount/frontend-sa created
candidate@cli:~$ kubectl get sa -n qa
NAME          SECRETS  AGE
default       1        5h47m
frontend-sa   1        4s
podrunner     1        5h47m
candidate@cli:~$ cat /home/candidate/KSCH00301/pod-manifest.yaml
apiVersion: v1
kind: Pod
metadata:
  name: "frontend"
  namespace: "qa"
spec:
  serviceAccountName: "frontend-sa"
  containers:
  - name: "frontend"
    image: nginx
candidate@cli:~$ vim /home/candidate/KSCH00301/pod-manifest.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: "frontend"
  namespace: "qa"
spec:
  serviceAccountName: "frontend-sa"
  automountServiceAccountToken: false
  containers:
  - name: "frontend"
    image: nginx
```

```
candidate@cli:~$ vim /home/candidate/KSCH00301/pod-manifest.yaml
candidate@cli:~$ cat /home/candidate/KSCH00301/pod-manifest.yaml
apiVersion: v1
kind: Pod
metadata:
  name: "frontend"
  namespace: "qa"
spec:
  serviceAccountName: "frontend-sa"
  automountServiceAccountToken: false
  containers:
  - name: "frontend"
    image: nginx
candidate@cli:~$ kubectl create -f /home/candidate/KSCH00301/pod-manifest.yaml
pod/frontend created
candidate@cli:~$ kubectl get pods -n qa
NAME      READY   STATUS    RESTARTS   AGE
frontend  1/1     Running   0           6s
candidate@cli:~$ kubectl get sa -n qa
NAME          SECRETS   AGE
default       1         5h49m
frontend-sa   1         105s
podrunner    1         5h49m
candidate@cli:~$ kubectl delete sa/podrunner -n qa
serviceaccount "podrunner" deleted
candidate@cli:~$
```

## QUESTION NO: 6 - (SIMULATION)

Cluster: qa-cluster Master node: master Worker node: worker1 You can switch the cluster/configuration context using the following command: [desk@cli] \$ kubectl config use-context qa-cluster Task: Create a NetworkPolicy named restricted-policy to restrict access to Pod product running in namespace dev. Only allow the following Pods to connect to Pod products-service:1. Pods in the namespace qa 2. Pods with label environment: stage, in any namespace

**ANSWER: See the below.**

## Explanation:

```
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
```

```
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: {}
```

```
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:~$
```

Reference: <https://kubernetes.io/docs/concepts/services-networking/network-policies/>

## QUESTION NO: 7 - (SIMULATION)

Use the kubesecc docker images to scan the given YAML manifest, edit and apply the advised changes, and passed with a score of 4 points.

kubesecc-test.yaml

Hint: `docker run -i kubesecc/kubesecc:512c5e0 scan /dev/stdin < kubesecc-test.yaml`

**ANSWER: See explanation below.**

## Explanation:

```
kubesecc scan k8s-deployment.yaml
```

```
cat < kubesecc-test.yaml
```

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
name: kubesecc-demo
```

```
spec:
```

```
containers:
```

```
- name: kubesecc-demo
```

```
image: gcr.io/google-samples/node-hello:1.0
```

```
securityContext:
```

```
readOnlyRootFilesystem: true
```

```
EOF
```

```
kubesecc scan kubesecc-test.yaml
```

```
docker run -i kubesecc/kubesecc:512c5e0 scan /dev/stdin < kubesecc-test.yaml
```

```
kubesecc http 8080 &
```

```
[1] 12345
```

```
{"severity":"info","timestamp":"2019-05-12T11:58:34.662+0100","caller":"server/server.go:69","message":"Starting HTTP server on port 8080"}
```

```
curl -sSX POST --data-binary @test/asset/score-0-cap-sys-admin.yml http://localhost:8080/scan
```

```
[
```

```
{
```

```
"object": "Pod/security-context-demo.default",
```

```
"valid": true,
```

```
"message": "Failed with a score of -30 points",
```

```
"score": -30,
```

```
"scoring": {
```

```
"critical": [
```

```
{
```

```
"selector": "containers[] .securityContext .capabilities .add == SYS_ADMIN",  
"reason": "CAP_SYS_ADMIN is the most privileged capability and should always be avoided"  
},  
{  
"selector": "containers[] .securityContext .runAsNonRoot == true",  
"reason": "Force the running image to run as a non-root user to ensure least privilege"  
},  
// ...
```

## QUESTION NO: 8 - (SIMULATION)

Context:Cluster: prodMaster node: master1Worker 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: Seetheexplanationbelow

### Explanation:

1. For Dockerfile: Fix the image version & user name in Dockerfile2. 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



```
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: {}
```

Reference: <https://kubernetes.io/docs/concepts/policy/pod-security-policy/>

## QUESTION NO: 9 - (SIMULATION)

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

```
[candidate@cli] $ | kubectl config use-context KS  
MV00102
```

## Context

A PodSecurityPolicy shall prevent the creation of privileged Pods in a specific namespace.

## Task

Create a new PodSecurityPolicy named prevent-ppp-policy, which prevents the creation of privileged Pods.

Create a new ClusterRole named restrict-access-role, which uses the newly created PodSecurityPolicy prevent-ppp-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/KSMV00  
102/pod-security-policy.ya  
ml
- /home/candidate/KSMV00  
102/cluster-role.yaml
- /home/candidate/KSMV00  
102/service-account.yaml
- /home/candidate/KSMV00  
102/cluster-role-binding.ya  
ml

**ANSWER: See explanation below.**

**Explanation:**

```
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
```

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: "prevent-pod-policy"
spec:
  privileged: false
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
```

```
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-ppsp-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-ppsp-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
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
```

```
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-role"
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=restrict-access-role --serviceaccount=staging:psp-restrict-sa --dry-run -o yaml
W0520 14:41:23.502004 47627 helpers.go:598] --dry-run is deprecated and can be replaced with --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-binding.yaml cluster-role-binding.yaml
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role-binding.yaml cluster-role-binding.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
```

```
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

candidate@cli:~$
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/cluster-role-binding.yaml
clusterrolebinding.rbac.authorization.k8s.io/restrict-access-bind created
candidate@cli:~$ █
```

## QUESTION NO: 10 - (SIMULATION)

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 PodSecurityPolicy named deny-policy, which prevents the creation of privileged Pods.  
2. Create a new ClusterRole named deny-access-role, which uses the newly created PodSecurityPolicy deny-policy.  
3. Create a new ServiceAccount named psp-denial-sa in the existing namespace development.  
Finally, create a new ClusterRoleBinding named restrict-access-bind, which binds the newly created ClusterRole deny-access-role to the newly created ServiceAccount psp-denial-sa

**ANSWER: See the explanation below**

### Explanation:

Create psp to disallow privileged container

k create sa psp-denial-sa -n development

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 developmentmaster1 $ 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.yamlmaster1 $ k apply -f cr1.yamlmaster1 $ k apply -f cb1.yamlReference:
```

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