

**100%** Money Back  
**Guarantee**

**Vendor:**Linux Foundation


**Exam Code:**CKS

**Exam Name:**Certified Kubernetes Security Specialist  
(CKS) Exam

**Version:**Demo

## QUESTION 1

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.

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

Cluster	Master node	Worker node
KSSC00301	kssc00301-master	kssc00301-worker1

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

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

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`.



A. See explanation below.

B. Placeholder

Correct Answer: A

---

## QUESTION 2

```
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-pr
spec:
  containers:
  - name: nginx-pod
    image: nginx:1.19.0
    ports:
    - containerPort: 80
~

```

```

candidate@cli:~$ vim KSSH00401/nginx-pod.yaml
candidate@cli:~$ kubectl create -f KSSH00401/nginx-pod.yaml
pod/nginx-pod created
candidate@cli:~$ cat KSSH00401/nginx-pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  annotations:
    container.apparmor.security.beta.kubernetes.io/nginx-pod: localhost/nginx-profile-2
spec:
  containers:
  - name: nginx-pod
    image: nginx:1.19.0
    ports:
    - containerPort: 80

```

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.

A. See the explanation below

B. Placeholder

Correct Answer: A

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

```
uk.co.certification.simulator.questionpool.PList@11602760
```

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

```
Add these uk.co.certification.simulator.questionpool.PList@11602c70
```

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

```
[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 and 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
```

---

**QUESTION 3**



```

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-pr
spec:
  containers:
  - name: nginx-pod
    image: nginx:1.19.0
    ports:
    - containerPort: 80
~

```

```

candidate@cli:~$ vim KSSH00401/nginx-pod.yaml
candidate@cli:~$ kubectl create -f KSSH00401/nginx-pod.yaml
pod/nginx-pod created
candidate@cli:~$ cat KSSH00401/nginx-pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  annotations:
    container.apparmor.security.beta.kubernetes.io/nginx-pod: localhost/nginx-profile-2
spec:
  containers:
  - name: nginx-pod
    image: nginx:1.19.0
    ports:
    - containerPort: 80

```

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:

1.

Ensure that the RotateKubeletServerCertificate argument is set to true.

2.

Ensure that the admission control plugin PodSecurityPolicy is set.

3.

Ensure that the --kubelet-certificate-authority argument is set as appropriate. Fix all of the following violations that were found against the Kubelet:

1.

Ensure the --anonymous-auth argument is set to false.

2.

Ensure that the --authorization-mode argument is set to Webhook. Fix all of the following violations that were found against the ETCD:

1.

Ensure that the --auto-tls argument is not set to true

2.

Ensure that the --peer-auto-tls argument is not set to true

Hint: Take the use of Tool Kube-Bench

A. See the below.

B. Placeholder

Correct Answer: A

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=`

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 masternode 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 masternode and either remove the `-- peer-auto-tls` parameter or set it to false.`--peer-auto-tls=false`

---

#### QUESTION 4

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`

A. See the explanation below

B. Placeholder

Correct Answer: A

```
master1 $ k get pods -n test --show-labels uk.co.certification.simulator.questionpool.PList@132b47c0 $ vim netpol.yaml
uk.co.certification.simulator.questionpool.PList@132b4af0 master1 $ k apply -f netpol.yaml
```

```
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

---

## QUESTION 5

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

A. See the explanation below

B. Placeholder

Correct Answer: A

1. For Dockerfile: Fix the image version and 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" ]
```

Text

[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: {}
```

## QUESTION 6

```
candidate@cli:~$ kubectl config use-context KSMV00201
Switched to context "KSMV00201".
candidate@cli:~$ kubectl get secret -n monitoring
NAME                                TYPE                                DATA  AGE
dbl-test                            Opaque                              2      6h23m
default-token-cqgf6                 kubernetes.io/service-account-token 3      6h23m
candidate@cli:~$ kubectl get secret/dbl-test -n monitoring
NAME      TYPE      DATA  AGE
dbl-test  Opaque    2      6h23m
candidate@cli:~$ kubectl get secret/dbl-test -n monitoring -o yaml
apiVersion: v1
data:
  password: QVU3dHh1bXF0THZt
  username: CHJvZHVjdGlvbi0x
kind: Secret
metadata:
  creationTimestamp: "2022-05-20T08:37:33Z"
  name: dbl-test
  namespace: monitoring
  resourceVersion: "2588"
  uid: 659bd4ac-e0ba-4d9f-b411-816f2aedf7e6
type: Opaque
candidate@cli:~$ echo "CHJvZHVjdGlvbi0x" | base64 -d
production-1candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ echo "CHJvZHVjdGlvbi0x" | base64 -d > /home/candidate/username.txt
candidate@cli:~$ cat /home/candidate/username.txt
production-1candidate@cli:~$
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ echo "QVU3dHh1bXF0THZt" | base64 -d
AU7txumqNLvmcandidate@cli:~$ echo "QVU3dHh1bXF0THZt" | base64 -d > /home/candidate/password.
txt
candidate@cli:~$ cat /home/candidate/password.txt
AU7txumqNLvmcandidate@cli:~$
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl create secret generic test-workflow --from-literal=username=dev-dat
abase --from-literal=password=aV7HR7nU3JLx -n monitoring
secret/test-workflow created
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl -n monitoring run test-secret-pod --image=httpd --dry-run=client -
o yaml > test-secret-pod.yaml
candidate@cli:~$ vim test-secret-pod.yaml
```



```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: test-secret-pod
    name: test-secret-pod
    namespace: monitoring
spec:
  volumes:
    - name: dev-volume
      secret:
        secretName: test-workflow
  containers:
    - image: httpd
      name: dev-container
      resources: {}
      volumeMounts:
        - name: dev-volume
          mountPath: /etc/credentials
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

```
candidate@cli:~$ kubectl -n monitoring run test-secret-pod --image=httpd --dry-run=client -o yaml > test-secret-pod.yaml
candidate@cli:~$ vim test-secret-pod.yaml
candidate@cli:~$ cat test-secret-pod.yaml
```

```
labels:
  run: test-secret-pod
  name: test-secret-pod
  namespace: monitoring
spec:
  volumes:
  - name: dev-volume
    secret:
      secretName: test-workflow
  containers:
  - image: httpd
    name: dev-container
    resources: {}
    volumeMounts:
    - name: dev-volume
      mountPath: /etc/credentials
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
candidate@cli:~$ kubectl create -f test-secret-pod.yaml
pod/test-secret-pod created
candidate@cli:~$ kubectl get pods -n monitoring
NAME             READY   STATUS    RESTARTS   AGE
test-secret-pod  1/1     Running   0           9s
candidate@cli:~$
```

Context:

Cluster: gvisor

Master node: master1

Worker node: worker1

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

```
[desk@cli] $ kubectl config use-context gvisor
```

Context: This cluster has been prepared to support runtime handler, runc as well as traditional one.

Task:

Create a RuntimeClass named not-trusted using the prepared runtime handler names runc.

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

A. See the explanation below

B. Placeholder

Correct Answer: A

1.

apiVersion: node.k8s.io/v1

2.

kind: RuntimeClass

3.

metadata:

4.

name: not-trusted

5.

handler: runsc [desk@cli] \$ k apply -f runtime.yaml [desk@cli] \$ k get pods

1.

NAME READY STATUS RESTARTS AGE

2.

nginx-6798fc88e8-chp6r 1/1 Running 0 11m

3.

nginx-6798fc88e8-fs53n 1/1 Running 0 11m

4.

nginx-6798fc88e8-ndved 1/1 Running 0 11m [desk@cli] \$ k get deploy

1.

NAME READY UP-TO-DATE AVAILABLE AGE

2.

nginx 3/3 11 3 5m [desk@cli] \$ k edit deploy nginx

## 1. Create runtime class by the name of not-trusted using runsc handler

```
1 | apiVersion: kube.io/v1
2 | kind: RuntimeClass
3 | metadata:
4 |   name: not-trusted
5 | handler: runsc
```

## 2. Find all the pods/deployment and edit runtimeClassName parameter to not-trusted under spec

```
[desk@cli] $ k edit deploy nginx
```

```
1 | spec:
2 |   runtimeClassName: not-trusted. # Add this
```

```
[desk@cli] $vim runtime.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      labels:
        app: nginx
    spec:
      runtimeClassName: not-trusted # Add this
      containers:
      - image: nginx
        name: nginx
        resources: {}
  status: {}
```

## QUESTION 7

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

A. See the explanation below:

B. Placeholder

Correct Answer: A

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.

---

### QUESTION 8

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.

A. See the below.

B. Placeholder

Correct Answer: A

use 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-resource-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"]

---

## QUESTION 9

```

candidate@cli:~$ kubectl config use-context KSSC00401
Switched to context "KSSC00401".
candidate@cli:~$ ssh kssc00401-master
Warning: Permanently added '10.240.86.231' (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@kssc00401-master:~# kubectl get pods -n naboo
NAME          READY   STATUS    RESTARTS   AGE
c-3po         1/1     Running   0           6h48m
chewbacca     1/1     Running   0           6h48m
jawas         1/1     Running   0           6h48m
qui-gon-jinn  1/1     Running   0           6h48m
root@kssc00401-master:~# kubectl get pods -n naboo -o name
pod/c-3po
pod/chewbacca
pod/jawas
pod/qui-gon-jinn
root@kssc00401-master:~# for i in $(kubectl get pods -n naboo -o name)
> do
> kubectl get ${i} -o yaml | grep -i image
> done
Error from server (NotFound): pods "c-3po" not found
Error from server (NotFound): pods "chewbacca" not found
Error from server (NotFound): pods "jawas" not found
Error from server (NotFound): pods "qui-gon-jinn" not found
root@kssc00401-master:~# for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo
get ${i} -o yaml | grep -i image ; done
  image: centos:centos7.9.2009
  imagePullPolicy: Never
  image: centos:centos7.9.2009
  imageID: docker-pullable://centos@sha256:c73f515d06b0fa07bb18d8202035e739a494ce760aa7312
9f60f4bf2bd22b407
  image: photon:3.0
  imagePullPolicy: Never
  image: photon:3.0
  imageID: docker-pullable://photon@sha256:c48d61f0f3ad19215b75e2087cfbe95d7321abb454e4295
a0e6c38f563ece622
  image: alpine:3.7
  imagePullPolicy: Never
  image: alpine:3.7
  imageID: docker-pullable://alpine@sha256:8421d9a84432575381bfabd248f1eb56f3aa21d9d7cd251
1583c68c9b7511d10
  image: amazonlinux:2
  imagePullPolicy: Never
  image: amazonlinux:2
  imageID: docker-pullable://amazonlinux@sha256:246ef631c75ea83005889621119fd5cc9cbb5500e1
93707c38b6c060d597a146
root@kssc00401-master:~# trivy image centos:centos7.9.2009
2022-05-20T15:39:51.733Z          INFO    Need to update DB
2022-05-20T15:39:51.733Z          INFO    Downloading DB...
27.97 MiB / 27.97 MiB [-----] 100.00% 27.43 MiB p/s 1s

```



```

-----+
root@kssc00401-master:~# for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo
get ${i} -o yaml | grep -i image ; done
  image: centos:centos7.9.2009
  imagePullPolicy: Never
  image: centos:centos7.9.2009
  imageID: docker-pullable://centos@sha256:c73f515d06b0fa07bb18d8202035e739a494ce760aa7312
9f60f4bf2bd22b407
  image: photon:3.0
  imagePullPolicy: Never
  image: photon:3.0
  imageID: docker-pullable://photon@sha256:c48d61f0f3ad19215b75e2087cfbe95d7321abb454e4295
a0e6c38f563ece622
  image: alpine:3.7
  imagePullPolicy: Never
  image: alpine:3.7
  imageID: docker-pullable://alpine@sha256:8421d9a84432575381bfabd248f1eb56f3aa21d9d7cd251
1583c68c9b7511d10
  image: amazonlinux:2
  imagePullPolicy: Never
  image: amazonlinux:2
  imageID: docker-pullable://amazonlinux@sha256:246ef631c75ea83005889621119fd5cc9cbb5500e1
93707c38b6c060d597a146
root@kssc00401-master:~# trivy image photon:3.0
2022-05-20T15:40:18.003Z      INFO    Detected OS: photon
2022-05-20T15:40:18.003Z      INFO    Detecting Photon Linux vulnerabilities...
2022-05-20T15:40:18.005Z      INFO    Number of language-specific files: 0

photon:3.0 (photon 3.0)
=====
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)

```

```

root@kssc00401-master:~# kubectl get pods -n naboo -o name
pod/c-3po
pod/chewbacca
pod/jawas
pod/qui-gon-jinn
root@kssc00401-master:~# kubectl -n naboo pod/c-3po -o yaml | grep image
Error: flags cannot be placed before plugin name: -n
root@kssc00401-master:~# kubectl -n naboo get pod/c-3po -o yaml | grep image
  image: centos:centos7.9.2009
  imagePullPolicy: Never
  image: centos:centos7.9.2009
  imageID: docker-pullable://centos@sha256:c73f515d06b0fa07bb18d8202035e739a494ce760aa7312
9f60f4bf2bd22b407
root@kssc00401-master:~# kubectl -n naboo delete pod/c-3po
pod "c-3po" deleted
root@kssc00401-master:~# kubectl -n naboo delete pod/jawas
pod "jawas" deleted

```

```

pod "jawas" deleted
root@kssc00401-master:~# history
 1 kubectl get pods -n naboo
 2 kubectl get pods -n naboo -o name
 3 for i in $(kubectl get pods -n naboo -o name); do kubectl get ${i} -o yaml | grep -i
image ; done
 4 for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
grep -i image ; done
 5 trivy image centos:centos7.9.2009
 6 for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
grep -i image ; done
 7 trivy image photon:3.0
 8 for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
grep -i image ; done
 9 trivy image alpine:3.7
10 for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
grep -i image ; done
11 trivy image amazonlinux:2
12 kubectl get pods -n naboo -o name
13 kubectl -n naboo pod/c-3po -o yaml | grep image
14 kubectl -n naboo get pod/c-3po -o yaml | grep image
15 kubectl -n naboo delete pod/c-3po
16 kubectl -n naboo delete pod/jawas
17 history
root@kssc00401-master:~# █

```

AppArmor is enabled on the cluster's worker node. An AppArmor profile is prepared, but not enforced yet.

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



Cluster	Master node	Worker node
KSSH00401	kssh00401 -master	kssh00401 -worker1

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

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

You may use your browser to open **one additional tab** to access the AppArmor documentation.



## Task

On the cluster's worker node, enforce the prepared AppArmor profile located at `/etc/apparmor.d/nginx_apparmor`.

Edit the prepared manifest file located at `/home/candidate/KSSH00401/nginx-pod.yaml` to apply the AppArmor profile.

Finally, apply the manifest file and create the Pod specified in it.

A. See the explanation below

B. Placeholder

Correct Answer: A

---

## QUESTION 10

A container image scanner is set up on the cluster.

Given an incomplete configuration in the directory

`/etc/kubernetes/confcontrol` and a functional container image scanner with HTTPS endpoint `https://test-server.local.8081/image_policy`

1.

Enable the admission plugin.

2.

Validate the control configuration and change it to implicit deny.

Finally, test the configuration by deploying the pod having the image tag as latest.

A. See explanation below.

B. Placeholder

Correct Answer: A

```
ssh-add ~/.ssh/tempprivate eval "$(ssh-agent -s)" cd contrib/terraform/aws vi terraform.tfvars terraform init terraform apply -var-file=credentials.tfvars ansible-playbook -i ./inventory/hosts ./cluster.yml -e ansible_ssh_user=core -e bootstrap_os=coreos -b --become-user=root --flush-cache -e ansible_user=core
```



apiVersion: v1

2.

kind: Pod

3.

metadata:

4.

name: security-context-demo-2

5.

spec:

6.

securityContext:

7.

runAsUser: 1000

8.

containers:

9.

- name: sec-ctx-demo-2 10.image: gcr.io/google-samples/node-hello:1.0 11.securityContext: 12.runAsUser: 0  
13.privileged: True 14.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

A. See the explanation below:

B. Placeholder

Correct Answer: A

FROM debian:latest MAINTAINER k@bogotobogo.com

# 1 - RUN RUN apt-get update andand 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 andand 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

---

## QUESTION 12

A Role bound to a Pod's ServiceAccount grants overly permissive permissions. Complete the following tasks to reduce the set of permissions.

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



Cluster	Master node	Worker node
KSCH00201	ksch00201-master	ksch00201-worker1

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

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

#### Task

Given an existing Pod named web-pod running in the namespace security.

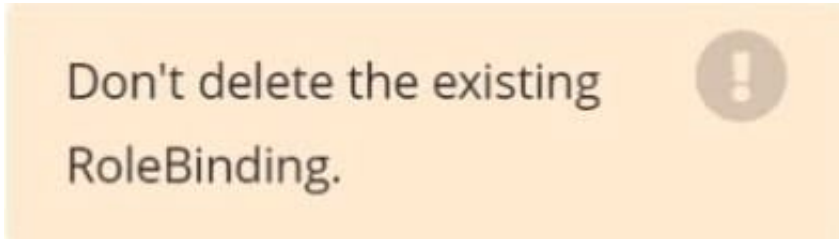
Edit the existing Role bound to the Pod's ServiceAccount sa-dev-1 to only allow performing watch operations, only on resources of type services.

Create a new Role named role-2 in the namespace security, which only allows performing update



operations, only on resources of type namespaces.

Create a new RoleBinding named role-2-binding binding the newly created Role to the Pod's ServiceAccount.



A. See the explanation below

B. Placeholder

Correct Answer: A

```
candidate@cli:~$ kubectl config use-context KSCH00201
Switched to context "KSCH00201".
candidate@cli:~$ kubectl get pods -n security
NAME      READY   STATUS    RESTARTS   AGE
web-pod   1/1     Running   0           6h9m
candidate@cli:~$ kubectl get deployments.apps -n security
No resources found in security namespace.
candidate@cli:~$ kubectl describe rolebindings.rbac.authorization.k8s.io -n security
Name:      dev-role
Labels:    <none>
Annotations: <none>
Role:
  Kind: Role
  Name: dev-role
Subjects:
  Kind      Name      Namespace
  ----      -
  ServiceAccount sa-dev-1
candidate@cli:~$ kubectl describe role dev-role -n security
Name:      dev-role
Labels:    <none>
Annotations: <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ----      -
  *          []                  []              [*]
candidate@cli:~$ kubectl edit role/dev-role -n security
```

```
uid: b4c9ddd6-2729-43bd-8fbd-b2d227f4c4cd
rules:
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - watch
```

```
candidate@cli:~$ kubectl describe role dev-role -n security
```

```
Name:          dev-role
Labels:        <none>
Annotations:   <none>
PolicyRule:
  Resources      Non-Resource URLs  Resource Names  Verbs
  -----
  *              []                  []               [*]
```

```
candidate@cli:~$ kubectl edit role/dev-role -n security
```

```
role.rbac.authorization.k8s.io/dev-role edited
```

```
candidate@cli:~$ kubectl describe role dev-role -n security
```

```
Name:          dev-role
Labels:        <none>
Annotations:   <none>
PolicyRule:
  Resources      Non-Resource URLs  Resource Names  Verbs
  -----
  services       []                  []               [watch]
```

```
candidate@cli:~$ kubectl get pods -n security
```

```
NAME      READY   STATUS    RESTARTS   AGE
web-pod   1/1     Running   0           6h12m
```

```
candidate@cli:~$ kubectl get pods/web-pod -n security -o yaml | grep serviceAccount
```

```
serviceAccount: sa-dev-1
serviceAccountName: sa-dev-1
- serviceAccountToken:
```

```
candidate@cli:~$ kubectl create role role-2 --verb=update --resource=namespaces -n security
role.rbac.authorization.k8s.io/role-2 created
```

```
candidate@cli:~$ kubectl create rolebinding role-2-binding --role
--role --role=
```

```
candidate@cli:~$ kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=se
curity:sa-dev-1 -n security
rolebinding.rbac.authorization.k8s.io/role-2-binding created
```

```
candidate@cli:~$ □
```