

CKS^{Q&As}

Certified Kubernetes Security Specialist (CKS) Exam

Pass Linux Foundation CKS Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

<https://www.leads4pass.com/cks.html>

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Linux Foundation Official Exam Center

-  **Instant Download** After Purchase
-  **100% Money Back** Guarantee
-  **365 Days** Free Update
-  **800,000+** Satisfied Customers



QUESTION 1

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 2


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 3

```
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:~$
```

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 name deny-access-role, which uses the newly created PodSecurityPolicy deny-policy.

3.

Create a new ServiceAccount named psd-denial-sa in the existing namespace development.

Finally, create a new ClusterRoleBindind named restrict-access-bind, which binds the newly created ClusterRole deny-access-role to the newly created ServiceAccount psp-denial-sa

A. See the explanation below

B. Placeholder

Correct Answer: A

Create psp to disallow privileged container uk.co.certification.simulator.questionpool.PList@11600d40 k create sa psp-denial-sa -n development uk.co.certification.simulator.questionpool.PList@11601040 namespace: development

Explanationmaster1 \$ 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

QUESTION 4

```
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: {}
  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:~$
```

1.

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

Store the value of the token in the token.txt

2.

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

username: mysql password: password@123

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

A. See the explanation below:

B. Placeholder

Correct Answer: A

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

1.

Navigate to your:

2.

Click Add Kubernetes cluster.

3.

Click the Add existing cluster tab and fill in the details:

Get the API URL by running this command:

```
kubectl cluster-info | grep -E '\\Kubernetes master|Kubernetes control plane\\' | awk '\\http/ {print $NF}\\'
```

```
uk.co.certification.simulator.questionpool.PList@113e1f90
```

```
kubectl get secret -o jsonpath="{[\\data\\][\\ca\\.crt\\]}"
```

QUESTION 5

Analyze and edit the given Dockerfile

1.

```
FROM ubuntu:latest
```

2.

```
RUN apt-get update -y
```

3.

RUN apt-install nginx -y

4.

COPY entrypoint.sh /

5.

ENTRYPOINT ["/entrypoint.sh"]

6.

USER ROOT

Fixing two instructions present in the file being prominent security best practice issues

Analyze and edit the deployment manifest file

1.

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
```

[CKS PDF Dumps](#)

[CKS Study Guide](#)

[CKS Brindumps](#)