Download Linux Foundation CKS Exam Dumps Free

Shared by Price on 17-06-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

SIMULATION

Documentation Deployment, Pod, Namespace

You must connect to the correct host . Failure to do so may result in a zero score.

[candidate@base] $ ssh cks000028

Context

You must update an existing Pod to ensure the immutability of its containers.

Task

Modify the existing Deployment named lamp-deployment, running in namespace lamp, so that its containers:

. run with user ID 20000

. use a read-only root filesystem

. forbid privilege escalation

The Deployment's manifest file con be found at /home/candidate/finer-sunbeam/lamp-deployment.yaml.

Options:

A- See the Explanation below for complete solution

Answer:

A

Explanation:

1) Connect to the correct host

ssh cks000028

sudo -i

2) Use the right kubeconfig (safe in exam)

export KUBECONFIG=/etc/kubernetes/admin.conf

3) Open the provided Deployment manifest

vi /home/candidate/finer-sunbeam/lamp-deployment.yaml

4) Edit ONLY the Pod template security settings (add/modify these fields)

Inside:

spec: -> template: -> spec:

4.1 Set container to run as user 20000

Add (or change) under the container securityContext::

securityContext:

runAsUser: 20000

4.2 Make root filesystem read-only

In the SAME container securityContext: ensure:

readOnlyRootFilesystem: true

4.3 Forbid privilege escalation

In the SAME container securityContext: ensure:

allowPrivilegeEscalation: false

The container section should look like this (example --- keep your existing image/ports/etc):

spec:

template:

spec:

containers:

- name: <your-container-name>

image: <unchanged>

securityContext:

runAsUser: 20000

readOnlyRootFilesystem: true

allowPrivilegeEscalation: false

If there are multiple containers, apply the same securityContext to each container.

Save and exit:

:wq

5) Apply the manifest (updates Deployment -> recreates Pods)

kubectl -n lamp apply -f /home/candidate/finer-sunbeam/lamp-deployment.yaml

6) Wait for rollout

kubectl -n lamp rollout status deployment/lamp-deployment

7) Verify the security settings are live

7.1 Check the Pod is running

kubectl -n lamp get pods -l app=lamp -o wide

(if label differs, just kubectl -n lamp get pods)

7.2 Verify the three fields on a running Pod

Pick the Pod name and run:

POD=$(kubectl -n lamp get pods -o jsonpath='{.items[0].metadata.name}')

kubectl -n lamp get pod $POD -o jsonpath='{.spec.containers[0].securityContext.runAsUser}{'\n'}{.spec.containers[0].securityContext.readOnlyRootFilesystem}{'\n'}{.spec.containers[0].securityContext.allowPrivilegeEscalation}{'\n'}'

Expected output:

20000

true

false

If the pod fails after readOnlyRootFilesystem=true

Don't change the requirement (task demands it). Usually the app needs writable dirs via volumes, but the task doesn't ask for that---so only adjust if the manifest already has volumes and just needs these securityContext fields.

Question 2

Question Type: MultipleChoice

SIMULATION

Context

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

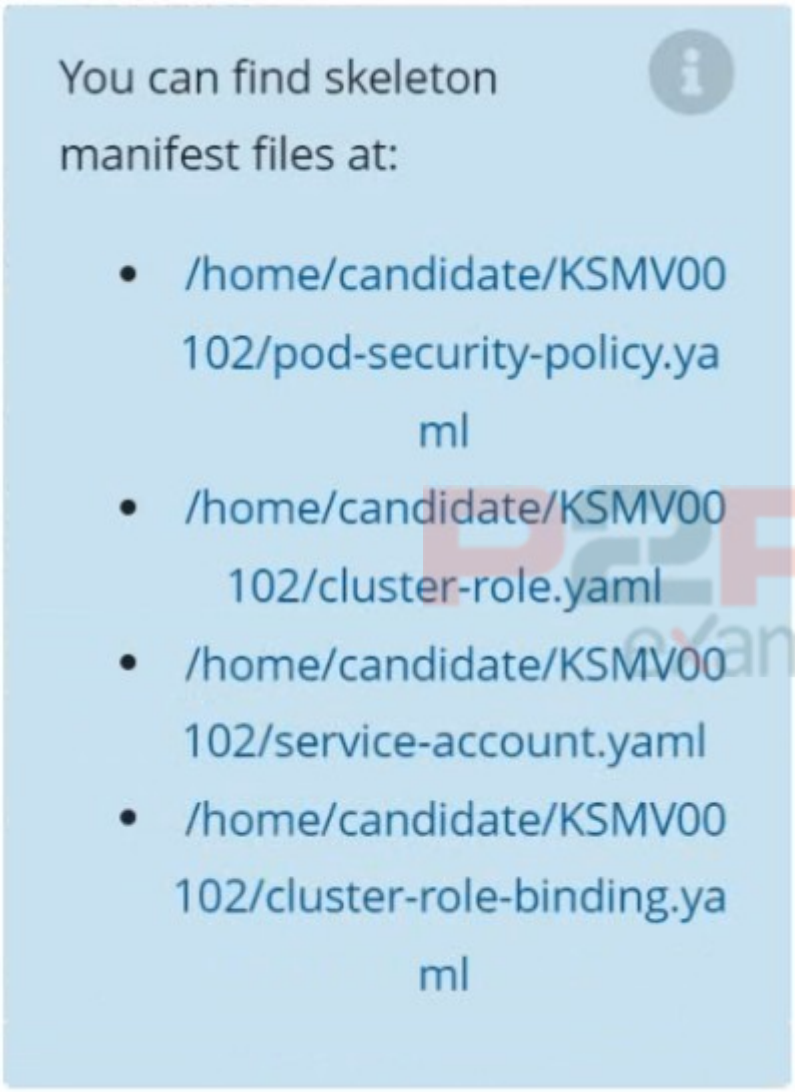
Task

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

Create a new ClusterRole named restrict-access-role, which uses the newly created PodSecurityPolicy prevent-psp-policy.

Create a new ServiceAccount named psp-restrict-sa in the existing namespace staging.

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



Options:

A- See the Explanation below

Answer:

A

Question 3

Question Type: MultipleChoice

SIMULATION

Two tools are pre-installed on the cluster's worker node:

Using the tool of your choice (including any non pre-installed tool), analyze the container's behavior for at least 30 seconds, using filters that detect newly spawning and executing processes.

Store an incident file at /opt/KSRS00101/alerts/details, containing the detected incidents, one per line, in the following format:

```
timestamp,uid/username,proce
ssName
```

The following example shows a properly formatted incident file:

```
01:40:19.601363716,root,init
01:40:20.606013716,nobody,ba
sh
01:40:21.137163716,1000,tar
```

Keep the tool's original timestamp-format as-is.

Options:

A- See explanation below

Answer:

A

Question 4

Question Type: MultipleChoice

SIMULATION

use the Trivy to scan the following images,

1. amazonlinux:1

2. k8s.gcr.io/kube-controller-manager:v1.18.6

Look for images with HIGH or CRITICAL severity vulnerabilities and store the output of the same in /opt/trivy-vulnerable.txt

Options:

A- Send us you rsuggestion on it

Answer:

A

Question 5

Question Type: MultipleChoice

SIMULATION

Documentation Secrets, TLS Secrets, Volumes

You must connect to the correct host . Failure to do so may result in a zero score.

[candidate@base] $ ssh cks000m40

Path

Key

Context

You must complete securing access to a web server using SSL files stored in a TLS Secret .

Task

Create a TLS Secret named clever-cactus in the clever-cactus namespace for an existing Deployment named clever-cactus.

Use the following SSL files:

File

Certificate /home/candidate/clever-cactus/web.k8s.local.crt

/home/candidate/clever-cactus/web.k8s.local.key

The Deployment is already configured to use the TLS Secret.

Do not modify the existing Deployment.

Failure to do so may result in a reduced score.

Options:

A- See the Explanation below for complete solution

Answer:

A

Explanation:

1) Connect to the correct host

ssh cks000m40

sudo -i

export KUBECONFIG=/etc/kubernetes/admin.conf

2) Verify namespace exists (quick check)

kubectl get ns clever-cactus

3) Verify certificate and key files exist

ls -l /home/candidate/clever-cactus/web.k8s.local.crt

ls -l /home/candidate/clever-cactus/web.k8s.local.key

Both files must exist.

4) Create the TLS Secret (THIS IS THE MAIN TASK)

Create a TLS Secret named clever-cactus in namespace clever-cactus:

kubectl -n clever-cactus create secret tls clever-cactus \

--cert=/home/candidate/clever-cactus/web.k8s.local.crt \

--key=/home/candidate/clever-cactus/web.k8s.local.key

Do NOT use apply

Do NOT edit the Deployment

5) Verify the Secret

kubectl -n clever-cactus get secret clever-cactus

Expected type:

kubernetes.io/tls

Optional detail check:

kubectl -n clever-cactus describe secret clever-cactus

You should see:

tls.crt

tls.key

6) (Optional) Confirm Pods are running

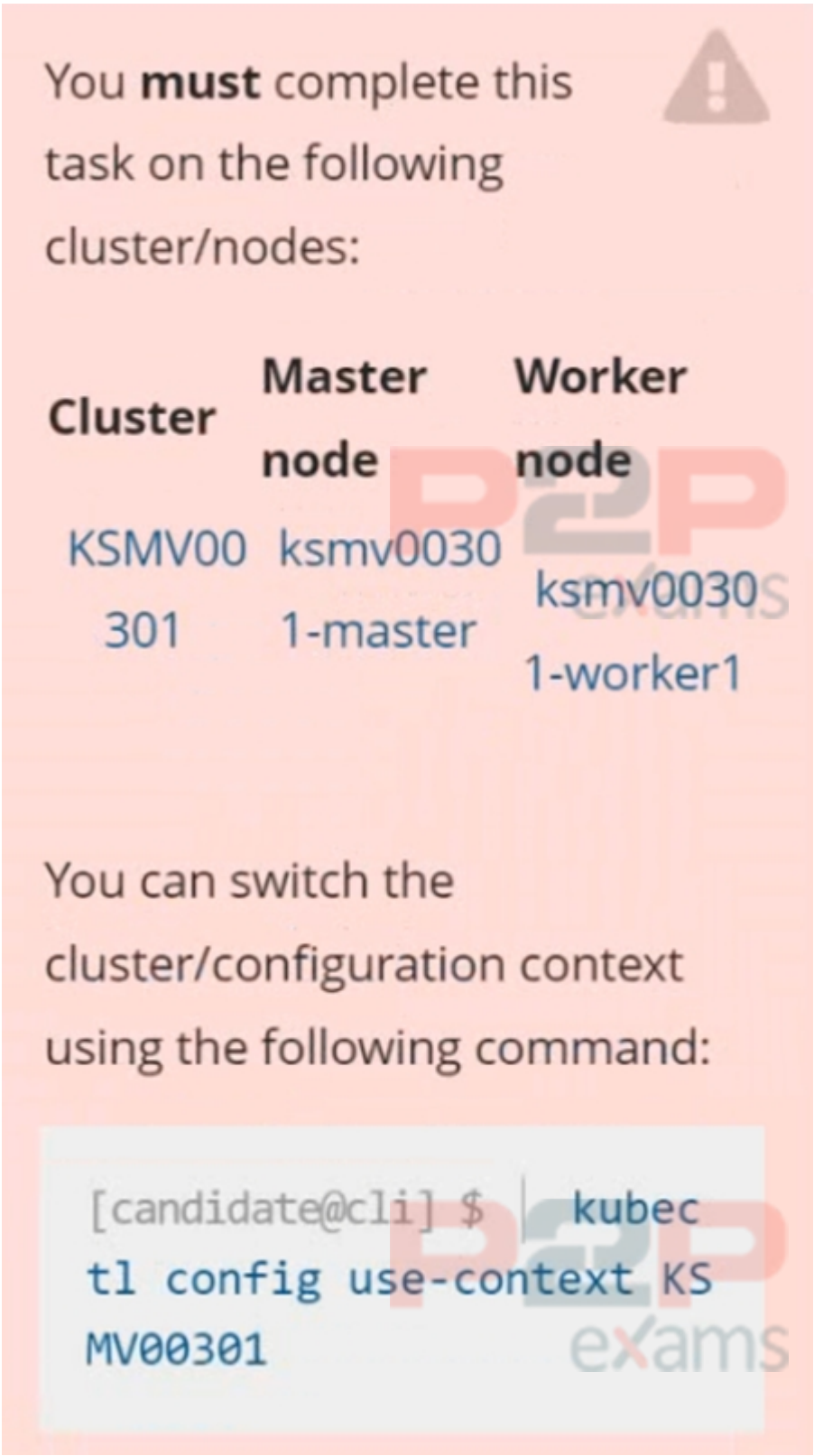
Since the Deployment is already configured to use the Secret, Pods should now work.

kubectl -n clever-cactus get pods

Question 6

Question Type: MultipleChoice

SIMULATION



Context

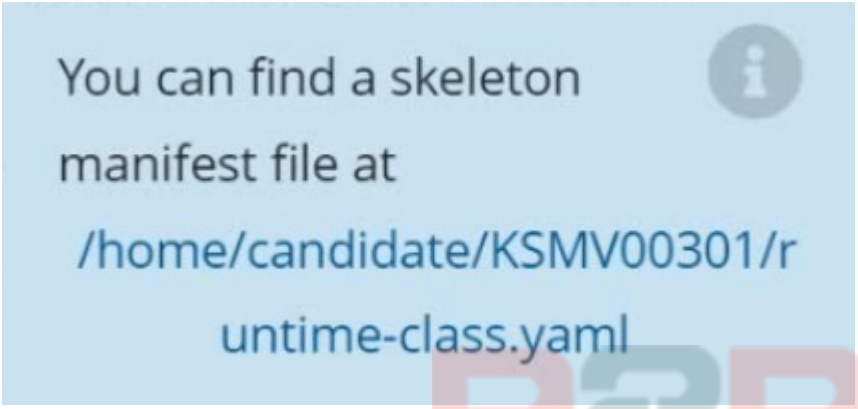
This cluster uses containerd as CRI runtime.

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

Task

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

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



Options:

A- See the Explanation below

Answer:

A

Question 7

Question Type: MultipleChoice

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 namespaceprodand 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 beprivilegedin any way must be treated as potentially not stateless or not immutable.

Options:

A- See the Explanation below

Answer:

A

Explanation:

```
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
```

https://cloud.google.com/architecture/best-practices-for-operating-containers

Question 8

Question Type: MultipleChoice

SIMULATION

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 theubuntu:18:04image)

/home/cert_masters/Dockerfilefixing two instructions present in the file being prominent security/best-practice issues.

Analyse and edit the given manifest file

/home/cert_masters/mydeployment.yamlfixing 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 usernobodywith user id65535

Options:

A- See the Explanation below

Answer:

A

Explanation:

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

2. For mydeployment.yaml : Fix security contexts

Explanation

[desk@cli] $vim /home/cert_masters/Dockerfile

FROM ubuntu:latest # Remove this

FROM ubuntu:18.04 # Add this

USER root # Remove this

USER nobody # Add this

RUN apt get install -y lsof=4.72 wget=1.17.1 nginx=4.2

ENV ENVIRONMENT=testing

USER root # Remove this

USER nobody # Add this

CMD ['nginx -d']

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

[desk@cli] $vim/home/cert_masters/mydeployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

creationTimestamp: null

labels:

app: kafka

name: kafka

spec:

replicas: 1

selector:

matchLabels:

app: kafka

strategy: {}

template:

metadata:

creationTimestamp: null

labels:

app: kafka

spec:

containers:

- image: bitnami/kafka

name: kafka

volumeMounts:

- name: kafka-vol

mountPath: /var/lib/kafka

securityContext:

{'capabilities':{'add':['NET_ADMIN'],'drop':['all']},'privileged': True,'readOnlyRootFilesystem': False, 'runAsUser': 65535} # Delete This

{'capabilities':{'add':['NET_ADMIN'],'drop':['all']},'privileged': False,'readOnlyRootFilesystem': True, 'runAsUser': 65535} # Add This

resources: {}

volumes:

- name: kafka-vol

emptyDir: {}

status: {}

Pictorial View:

[desk@cli] $vim/home/cert_masters/mydeployment.yaml

```
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: {}
```