# CKS$^{Q\&As}$

Certified Kubernetes Security Specialist (CKS) Exam

## Pass Linux Foundation CKS Exam with 100% Guarantee

**QUESTION 1**

Create a PSP that will only allow the persistentvolumeclaim as the volume type in the namespace restricted.

Create a new PodSecurityPolicy named prevent-volume-policy which prevents the pods which is having different volumes mount apart from persistentvolumeclaim.

Create a new ServiceAccount named psp-sa in the namespace restricted.

Create a new ClusterRole named psp-role, which uses the newly created Pod Security Policy prevent-volume-policy

Create a new ClusterRoleBinding named psp-role-binding, which binds the created ClusterRole psp-role to the created SA psp-sa.

Hint:

Also, Check the Configuration is working or not by trying to Mount a Secret in the pod maifest, it should get failed.

POD Manifest:

1.

apiVersion: v1

2.

kind: Pod

3.

metadata:

4.

name:

5.

spec:

6.

containers:

7.

- name:

8.

image:

9.

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

A. See the below:

B. PlaceHolder

Correct Answer: A

```
apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: restricted

annotations:

seccomp.security.alpha.kubernetes.io/allowedProfileNames:

\\'docker/default,runtime/default\\'

apparmor.security.beta.kubernetes.io/allowedProfileNames: \\'runtime/default\\'
seccomp.security.alpha.kubernetes.io/defaultProfileName: \\'runtime/default\\'
apparmor.security.beta.kubernetes.io/defaultProfileName: \\'runtime/default\\' spec:

privileged: false

# Required to prevent escalations to root.

allowPrivilegeEscalation: false

# This is redundant with non-root + disallow privilege escalation, # but we can provide it for defense in depth.

requiredDropCapabilities:

-ALL

# Allow core volume types.

volumes:

-\\'configMap\\'

-\\'emptyDir\\'

-\\'projected\\'

-\\'secret\\'

-\\'downwardAPI\\'

# Assume that persistentVolumes set up by the cluster admin are safe to use.

-\\'persistentVolumeClaim\\'

hostNetwork: false

hostIPC: false
```

hostPID: false

runAsUser:

# Require the container to run without root privileges.

rule: \\'MustRunAsNonRoot\\'

seLinux:

# This policy assumes the nodes are using AppArmor rather than SELinux.

rule: \\'RunAsAny\\'

supplementalGroups:

rule: \\'MustRunAs\\'

ranges:

# Forbid adding the root group.

-

min: 1

max: 65535

fsGroup:

rule: \\'MustRunAs\\'

ranges:

# Forbid adding the root group.

-

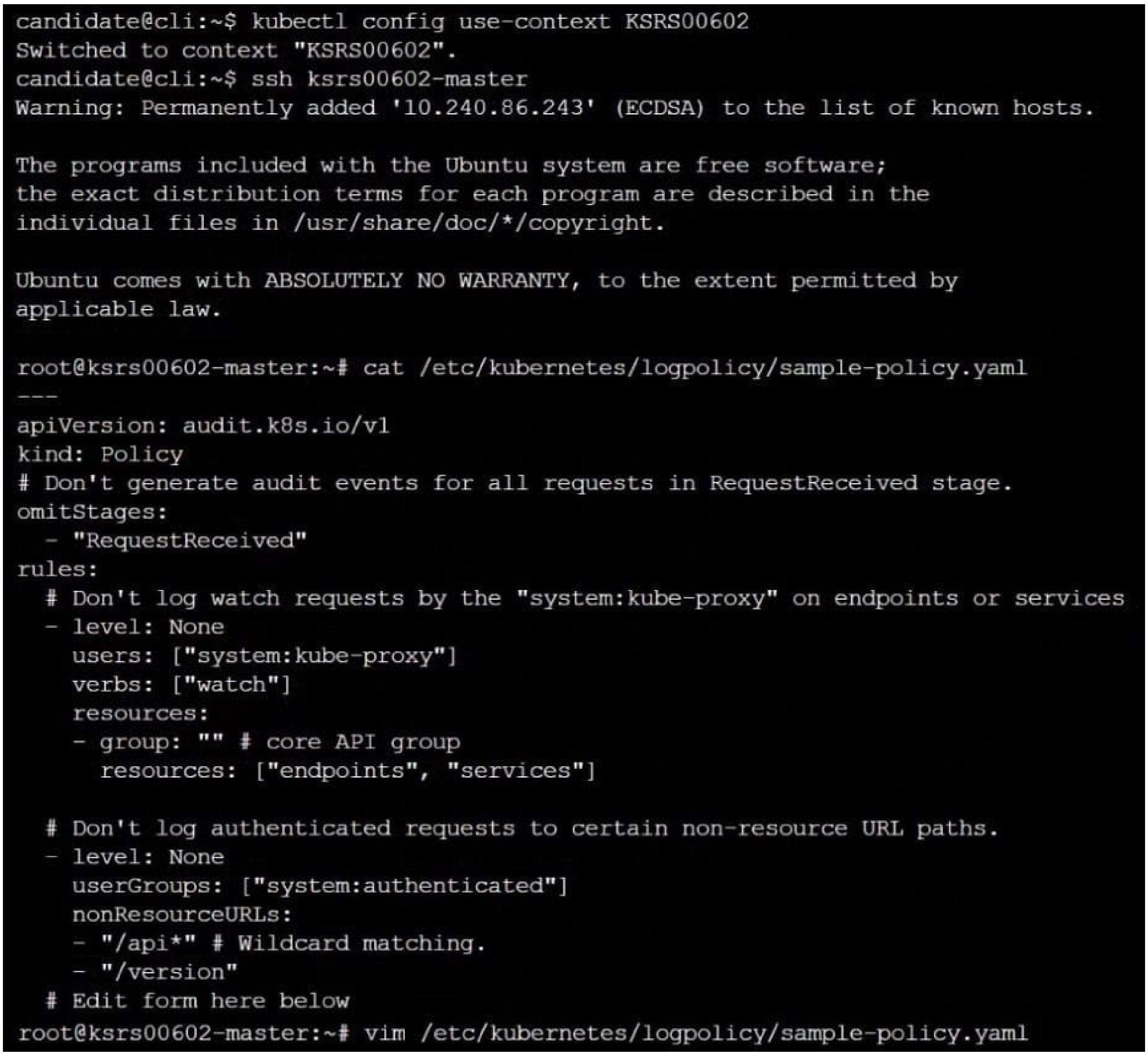
min: 1

max: 65535

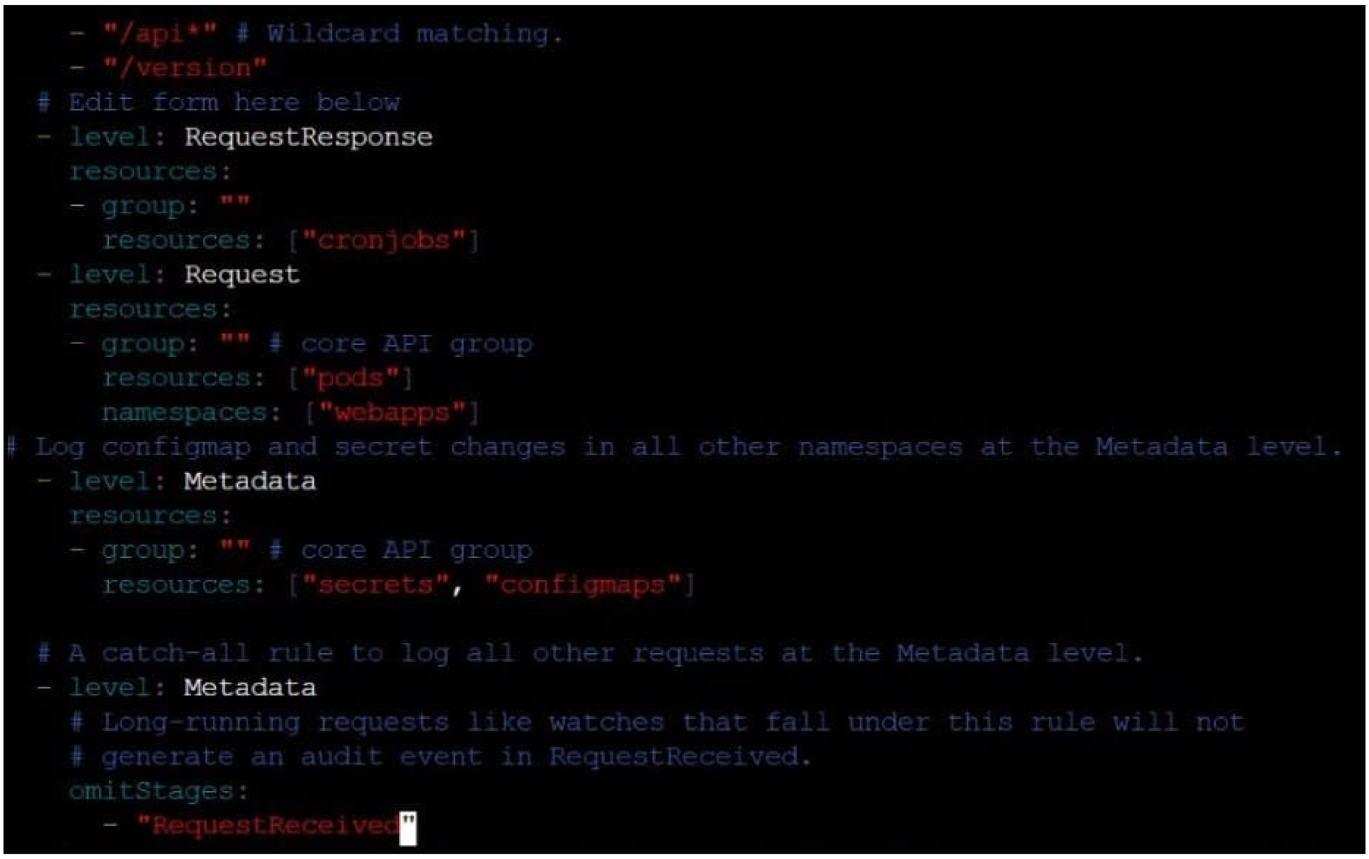
readOnlyRootFilesystem: false

---

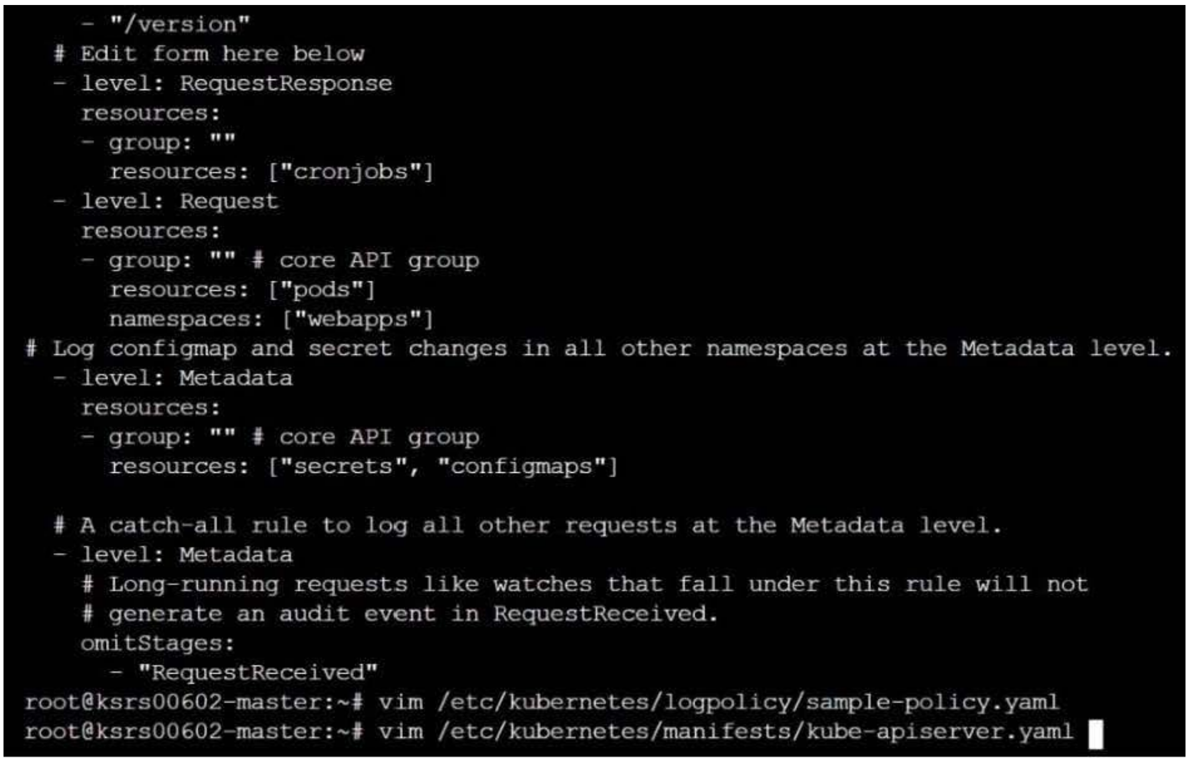
**QUESTION 2**



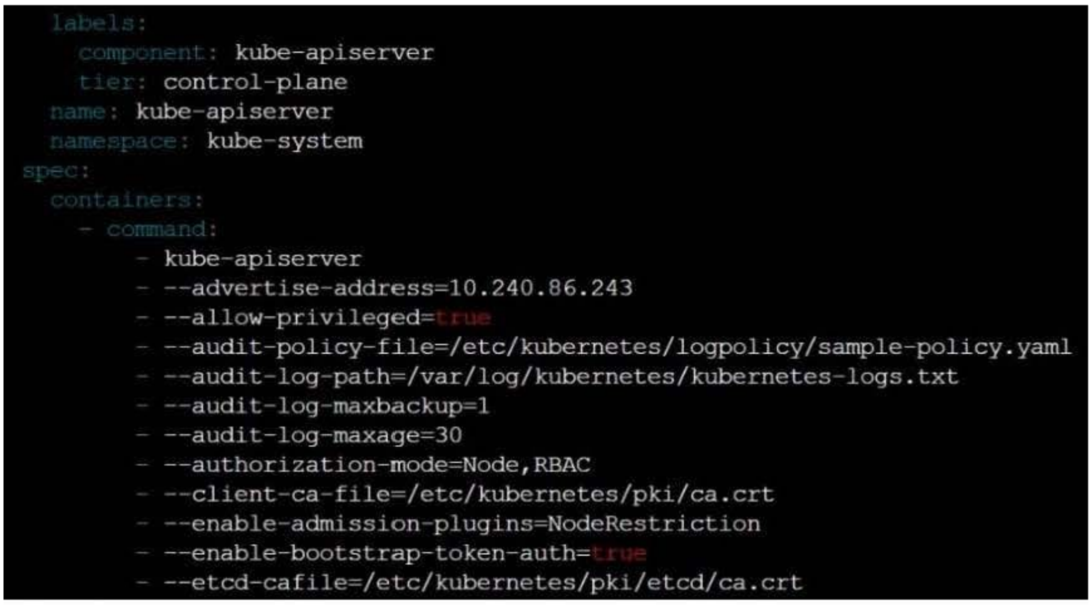
```
candidate@cli:~$ kubectl config use-context KSRS00602
Switched to context "KSRS00602".
candidate@cli:~$ ssh ksrs00602-master
Warning: Permanently added '10.240.86.243' (ECDSA) to the list of known hosts.

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@ksrs00602-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@ksrs00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
```



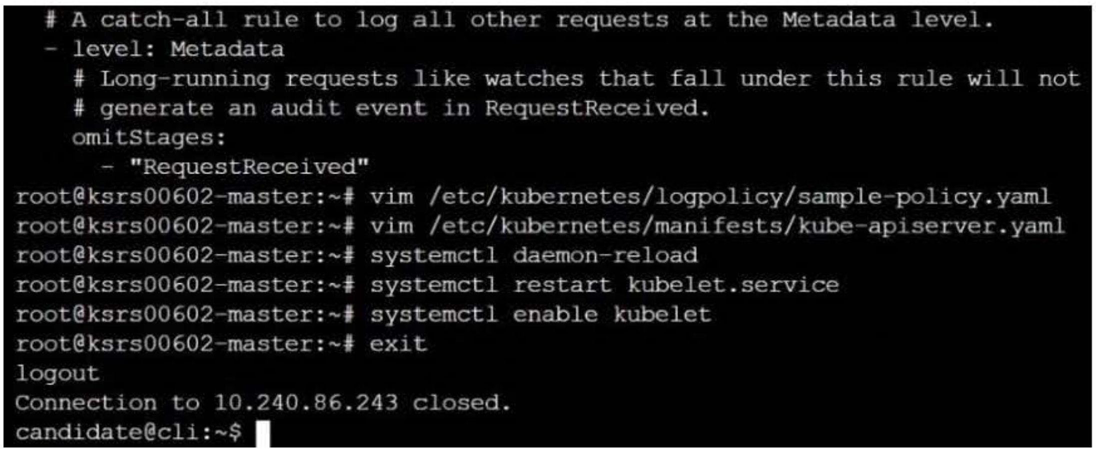
```
    - "/api*" # Wildcard matching.
    - "/version"
  # Edit form here below
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["cronjobs"]
  - level: Request
    resources:
    - group: "" # core API group
      resources: ["pods"]
      namespaces: ["webapps"]
# Log configmap 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:
  - command:
    - kube-apiserver
    - --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:~$
```

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

[desk@cli] $ kubectl config use-context dev

Context:

A CIS Benchmark tool was run against the kubeadm created cluster and found multiple issues that must be addressed.

Task:

Fix all issues via configuration and restart the affected components to ensure the new settings take effect.

Fix all of the following violations that were found against the API server:

1.2.7 authorization-mode argument is not set to AlwaysAllow FAIL

1.2.8 authorization-mode argument includes Node FAIL

1.2.7 authorization-mode argument includes RBAC FAIL

Fix all of the following violations that were found against the Kubelet:

4.2.1 Ensure that the anonymous-auth argument is set to false FAIL

4.2.2 authorization-mode argument is not set to AlwaysAllow FAIL (Use Webhook autumn/authz where possible)

Fix all of the following violations that were found against etcd:

2.2 Ensure that the client-cert-auth argument is set to true

A. See the explanation below

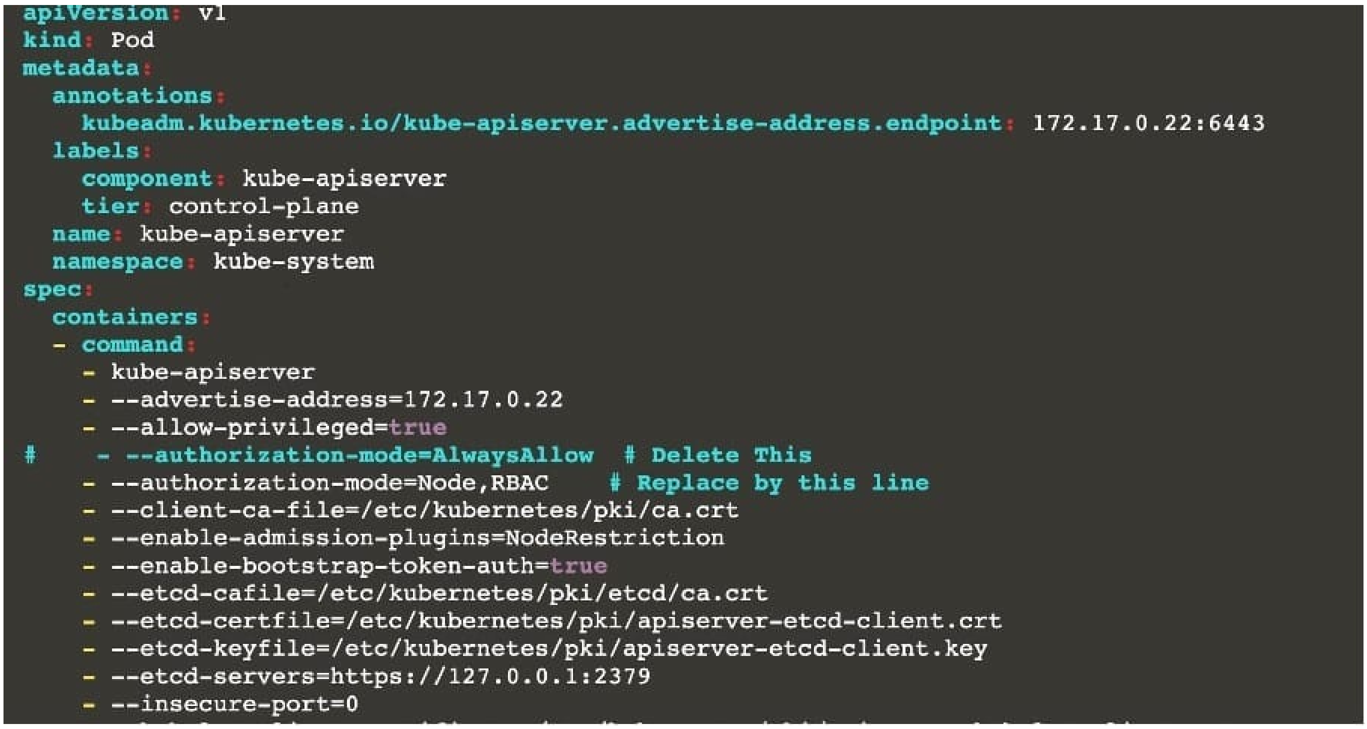
B. PlaceHolder

Correct Answer: A

worker1 $ vim /var/lib/kubelet/config.yaml uk.co.certification.simulator.questionpool.PList@132b77a0 worker1 $ systemctl restart kubelet. # To reload kubelet configssh to master1master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml- -- authorizationmode=Node,RBACmaster1 $ vim /etc/kubernetes/manifests/etcd.yaml- --client-cert-auth=true

Explanationssh to worker1worker1 $ vim /var/lib/kubelet/config.yaml apiVersion: kubelet.config.k8s.io/v1beta1 authentication: anonymous: enabled: true #Delete this enabled: false #Replace by this webhook: cacheTTL: 0s enabled: true x509: clientCAFile: /etc/kubernetes/pki/ca.crt authorization: mode: AlwaysAllow #Delete this mode: Webhook #Replace by this webhook: cacheAuthorizedTTL: 0s cacheUnauthorizedTTL: 0s cgroupDriver: systemd clusterDNS:

-10.96.0.10 clusterDomain: cluster.local cpuManagerReconcilePeriod: 0s evictionPressureTransitionPeriod: 0s fileCheckFrequency: 0s healthzBindAddress: 127.0.0.1 healthzPort: 10248 httpCheckFrequency: 0s imageMinimumGCAge: 0s kind: KubeletConfiguration logging: {} nodeStatusReportFrequency: 0s nodeStatusUpdateFrequency: 0s resolvConf: /run/systemd/resolve/resolv.conf rotateCertificates: true runtimeRequestTimeout: 0s staticPodPath: /etc/kubernetes/manifests streamingConnectionIdleTimeout: 0s syncFrequency: 0s volumeStatsAggPeriod: 0s worker1 $ systemctl restart kubelet. # To reload kubelet configssh to master1master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml

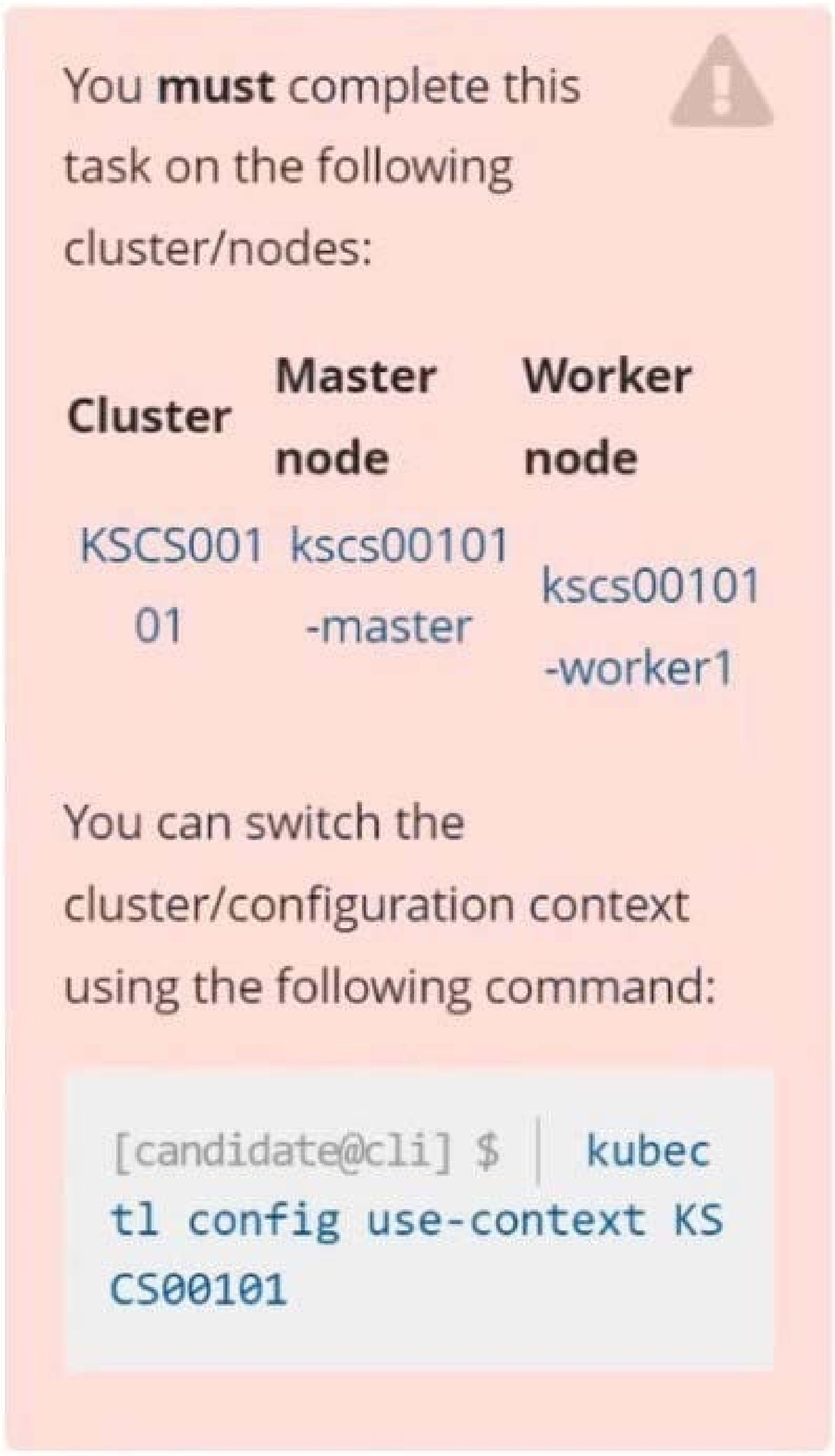


```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 172.17.0.22:6443
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=172.17.0.22
    - --allow-privileged=true
#     - --authorization-mode=AlwaysAllow  # Delete This
    - --authorization-mode=Node,RBAC    # Replace by this line
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
```

master1 $ vim /etc/kubernetes/manifests/etcd.yaml

**QUESTION 3**

CORRECT TEXT Context



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

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

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

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

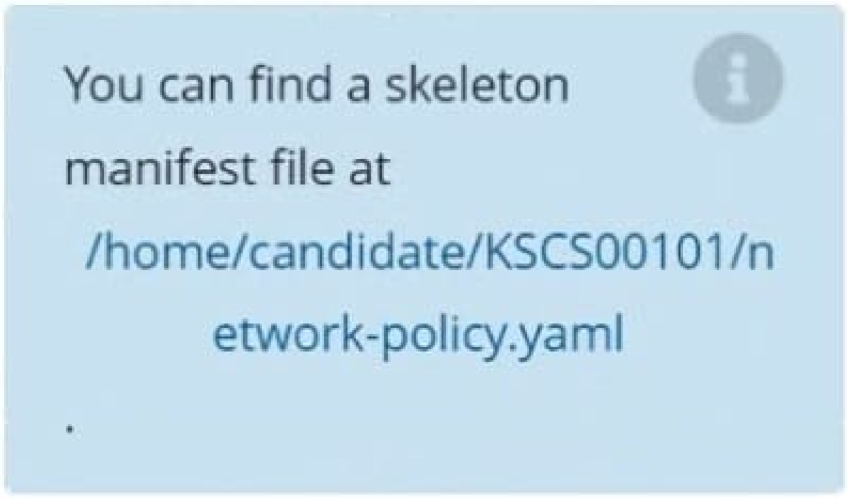
A default-deny NetworkPolicy avoids to accidentally expose a Pod in a namespace that doesn\\'t have any other NetworkPolicy defined.

Task

Create a new default-deny NetworkPolicy named defaultdeny in the namespace testing for all traffic of type Egress.

The new NetworkPolicy must deny all Egress traffic in the namespace testing.

Apply the newly created default-deny NetworkPolicy to all Pods running in namespace testing.



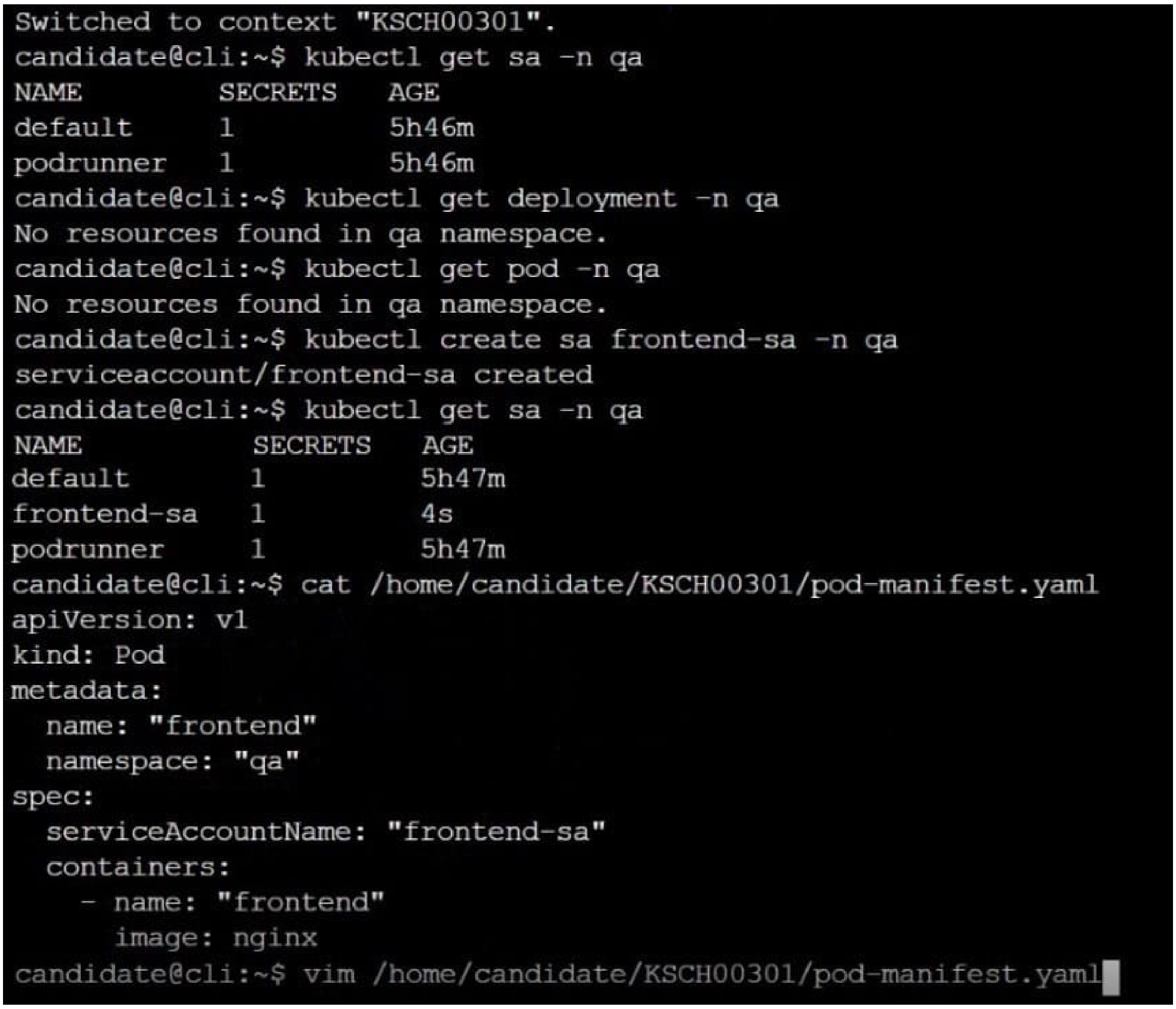
You can find a skeleton manifest file at /home/candidate/KSCS00101/network-policy.yaml
.

A. See explanation below.

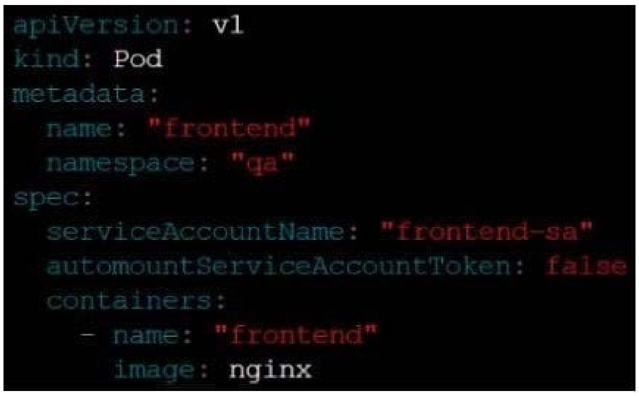
B. PlaceHolder

Correct Answer: A

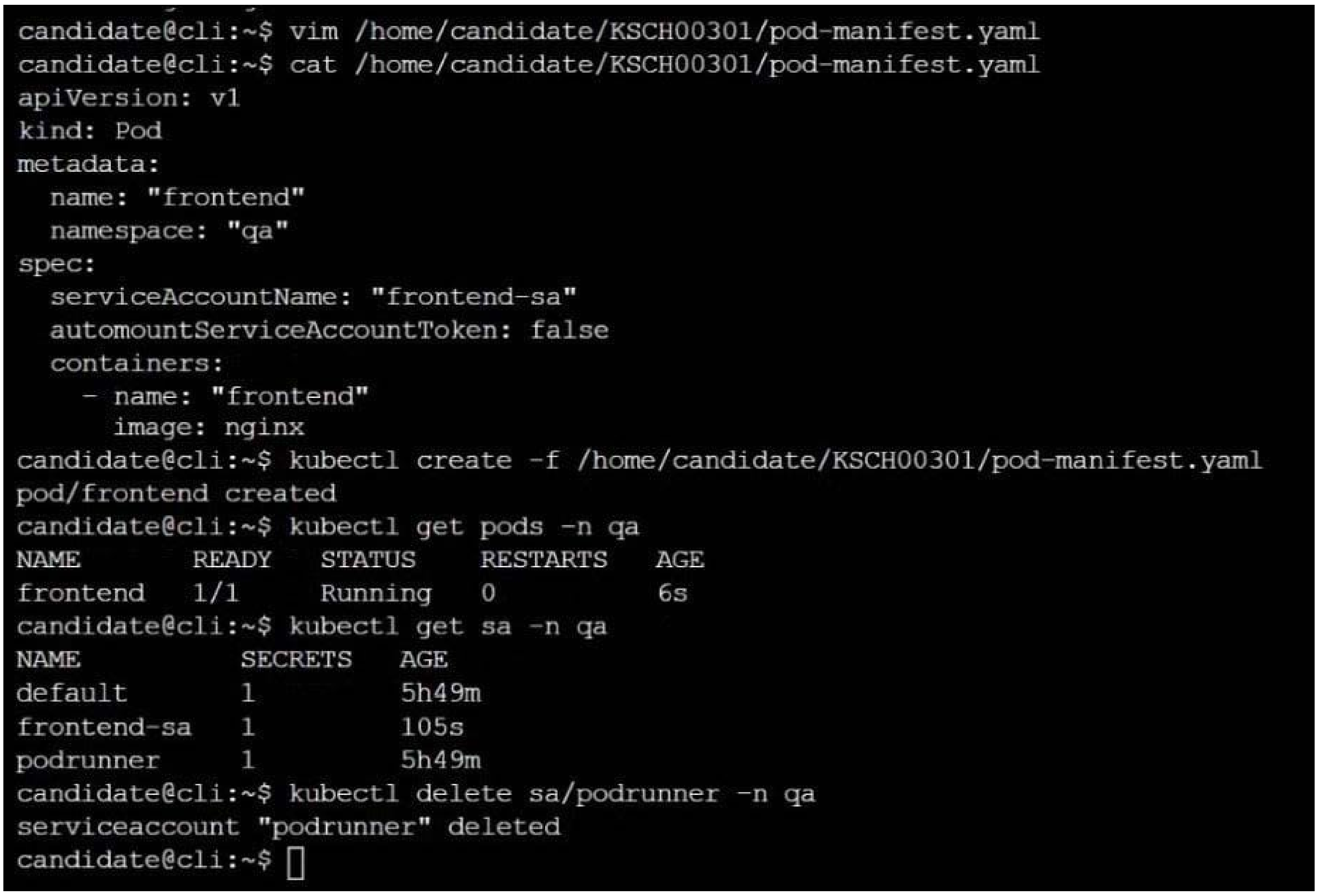
QUESTION 4



```
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 PodSecurityPolcy 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 5**

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.

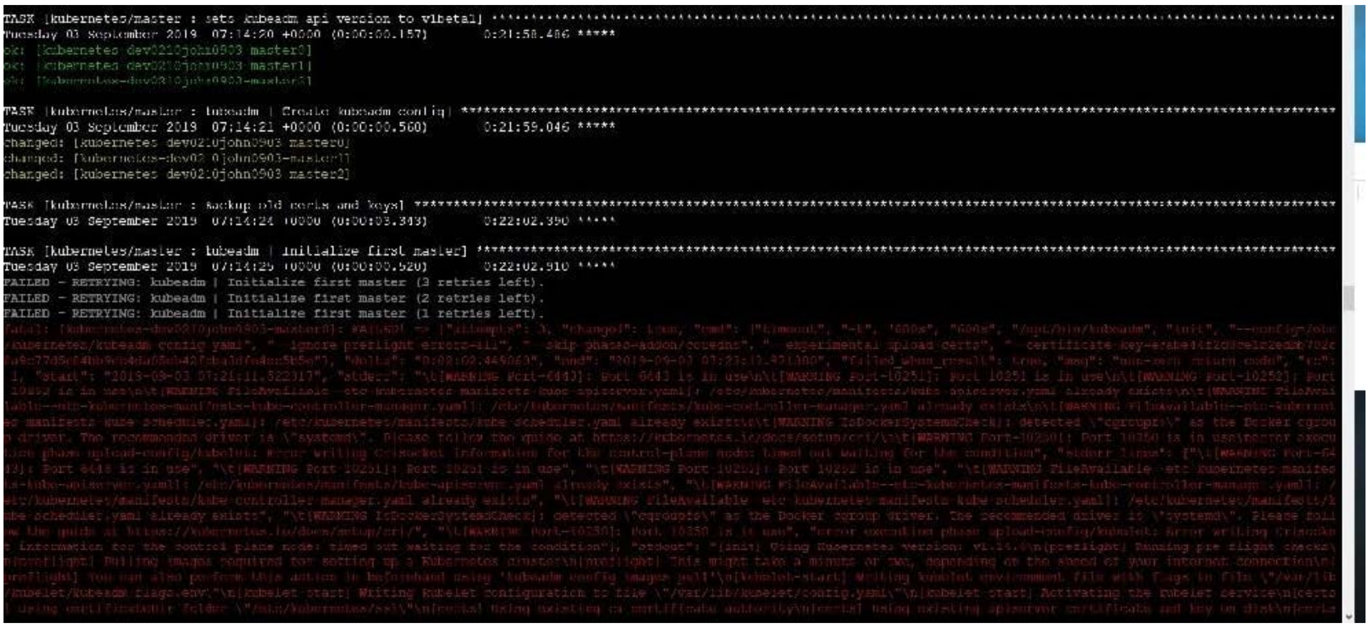
Finally, test the configuration by deploying the pod having the image tag as latest.

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



```
TASK [kubernetes/master : sets kubeadm api version to v1beta1] ************************************************************************
Tuesday 03 September 2019  07:14:20 +0000 (0:00:00.157)       0:21:58.486 *****
ok: [kubernetes-dev0210john0903-master0]
ok: [kubernetes-dev0210john0903-master1]
ok: [kubernetes-dev0210john0903-master2]

TASK [kubernetes/master : kubeadm | Create kubeadm config] ****************************************************************************
Tuesday 03 September 2019  07:14:21 +0000 (0:00:00.560)       0:21:59.046 *****
changed: [kubernetes-dev0210john0903-master0]
changed: [kubernetes-dev0210john0903-master1]
changed: [kubernetes-dev0210john0903-master2]

TASK [kubernetes/master : Backup old certs and keys] **********************************************************************************
Tuesday 03 September 2019  07:14:24 +0000 (0:00:03.343)       0:22:02.390 *****

TASK [kubernetes/master : kubeadm | Initialize first master] **************************************************************************
Tuesday 03 September 2019  07:14:25 +0000 (0:00:00.520)       0:22:02.910 *****
FAILED - RETRYING: kubeadm | Initialize first master (3 retries left).
FAILED - RETRYING: kubeadm | Initialize first master (2 retries left).
FAILED - RETRYING: kubeadm | Initialize first master (1 retries left).
fatal: [kubernetes-dev0210john0903-master0]: FAILED! => {"attempts": 3, "changed": true, "cmd": ["timeout", "-k", "600s", "600s", "/opt/bin/kubeadm", "init", "--config=/etc
/kubernetes/kubeadm config.yaml", "--ignore-preflight-errors=all", "--skip-phases=addon/coredns", "--experimental-upload-certs", "--certificate-key=[illegible]
[illegible]"], "delta": "0:02:02.449063", "end": "2019-09-03 07:23:13.971380", "failed_when_result": true, "msg": "non-zero return code", "rc":
 1, "start": "2019-09-03 07:21:11.522317", "stderr": "\t[WARNING Port-6443]: Port 6443 is in use\n\t[WARNING Port-10251]: Port 10251 is in use\n\t[WARNING Port-10252]: Port
 10252 is in use\n\t[WARNING FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already exists\n\t[WARNING FileAvai
lable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-manager.yaml already exists\n\t[WARNING FileAvailable--etc-kubernet
es-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already exists\n\t[WARNING IsDockerSystemdCheck]: detected \"cgroupfs\" as the Docker cgrou
p driver. The recommended driver is \"systemd\". Please follow the guide at https://kubernetes.io/docs/setup/cri/\n\t[WARNING Port-10250]: Port 10250 is in use\nerror execu
tion phase upload-config/kubelet: Error writing Crisocket information for the control-plane node: timed out waiting for the condition", "stderr_lines": ["\t[WARNING Port-64
43]: Port 6443 is in use", "\t[WARNING Port-10251]: Port 10251 is in use", "\t[WARNING Port-10252]: Port 10252 is in use", "\t[WARNING FileAvailable--etc-kubernetes-manifes
ts-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already exists", "\t[WARNING FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /
etc/kubernetes/manifests/kube-controller-manager.yaml already exists", "\t[WARNING FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/k
ube-scheduler.yaml already exists", "\t[WARNING IsDockerSystemdCheck]: detected \"cgroupfs\" as the Docker cgroup driver. The recommended driver is \"systemd\". Please foll
ow the guide at https://kubernetes.io/docs/setup/cri/", "\t[WARNING Port-10250]: Port 10250 is in use", "error execution phase upload-config/kubelet: Error writing Crisocke
t information for the control-plane node: timed out waiting for the condition"], "stdout": "[init] Using Kubernetes version: v1.14.[illegible]\n[preflight] Running pre-flight checks\
n[preflight] Pulling images required for setting up a Kubernetes cluster\n[preflight] This might take a minute or two, depending on the speed of your internet connection\n[
preflight] You can also perform this action in beforehand using 'kubeadm config images pull'\n[kubelet-start] Writing kubelet environment file with flags to file \"/var/lib
/kubelet/kubeadm-flags.env\"\n[kubelet-start] Writing kubelet configuration to file \"/var/lib/kubelet/config.yaml\"\n[kubelet-start] Activating the kubelet service\n[certs
] Using certificateDir folder \"/etc/kubernetes/ssl\"\n[certs] Using existing ca certificate authority\n[certs] Using existing apiserver certificate and key on disk\n[certs
```

[Latest CKS Dumps](https://www.leads4pass.com/cks.html) [CKS Practice Test](https://www.leads4pass.com/cks.html) [CKS Study Guide](https://www.leads4pass.com/cks.html)