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.certbus.com/cks.html

100% Passing Guarantee
100% Money Back Assurance

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

QUESTION 1

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 2

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

Cluster: trace Master node: master Worker node: worker1

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

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

Given: You may use Sysdig or Falco documentation.

Task:

Use detection tools to detect anomalies like processes spawning and executing something weird frequently in the single container belonging to Pod tomcat.

Two tools are available to use:

1.

falco

2.

sysdig

Tools are pre-installed on the worker1 node only.

Analyse the container\\'s behaviour for at least 40 seconds, using filters that detect newly spawning and executing processes.

Store an incident file at /home/cert_masters/report, in the following format:

[timestamp],[uid],[processName]

Note: Make sure to store incident file on the cluster\\'s worker node, don\\'t move it to master node.

A. See the explanation below

B. PlaceHolder

Correct Answer: A

$vim /etc/falco/falco_rules.local.yaml uk.co.certification.simulator.questionpool.PList@120e24d0 $kill -1 Explanation[desk@cli] $ ssh node01[node01@cli] $ vim /etc/falco/falco_rules.yamlsearch for Container Drift Detected and paste in falco_rules.local.yaml[node01@cli] $ vim /etc/falco/falco_rules.local.yaml

-rule: Container Drift Detected (open+create) desc: New executable created in a container due to open+create condition: > evt.type in (open,openat,creat) and evt.is_open_exec=true and container and not runc_writing_exec_fifo and not runc_writing_var_lib_docker and not user_known_container_drift_activities and evt.rawres>=0 output: > %evt.time,%user.uid,%proc.name # Add this/Refer falco documentation priority: ERROR [node01@cli] $ vim /etc/falco/falco.yaml

QUESTION 3

Create a Pod name Nginx-pod inside the namespace testing, Create a service for the Nginx-pod named nginx-svc, using the ingress of your choice, run the ingress on tls, secure port.

A. See explanation below.

B. PlaceHolder

Correct Answer: A

$ kubectl get ing -n NAME HOSTS ADDRESS PORTS AGE cafe-ingress cafe.com 10.0.2.15 80 25s

$ kubectl describe ing -n Name: cafe-ingress Namespace: default Address: 10.0.2.15 Default backend: default-http-backend:80 (172.17.0.5:8080) Rules: Host Path Backends

cafe.com

/tea tea-svc:80 ()

/coffee coffee-svc:80 ()

Annotations:

kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"networking.k8s.io/v1","kind":"Ingress","metadata":{"annotations":{},"name":"c afe-ingress","namespace":"default","selfLink":"/apis/networking/v1/namespaces/default/ingress es/cafe-ingress"},"spec":{"rules":

[{"host":"cafe.com","http":{"paths":[{"backend":{"serviceName":"tea-svc","servicePort":80},"path":"/tea"},{"backend":{"serviceName":"coffee-svc","servicePort":80},"path":"/coffee"}]}}]},"status":{"loadBalancer":{"ingress":

[{"ip":"169.48.142.110"}]}}}

Events:

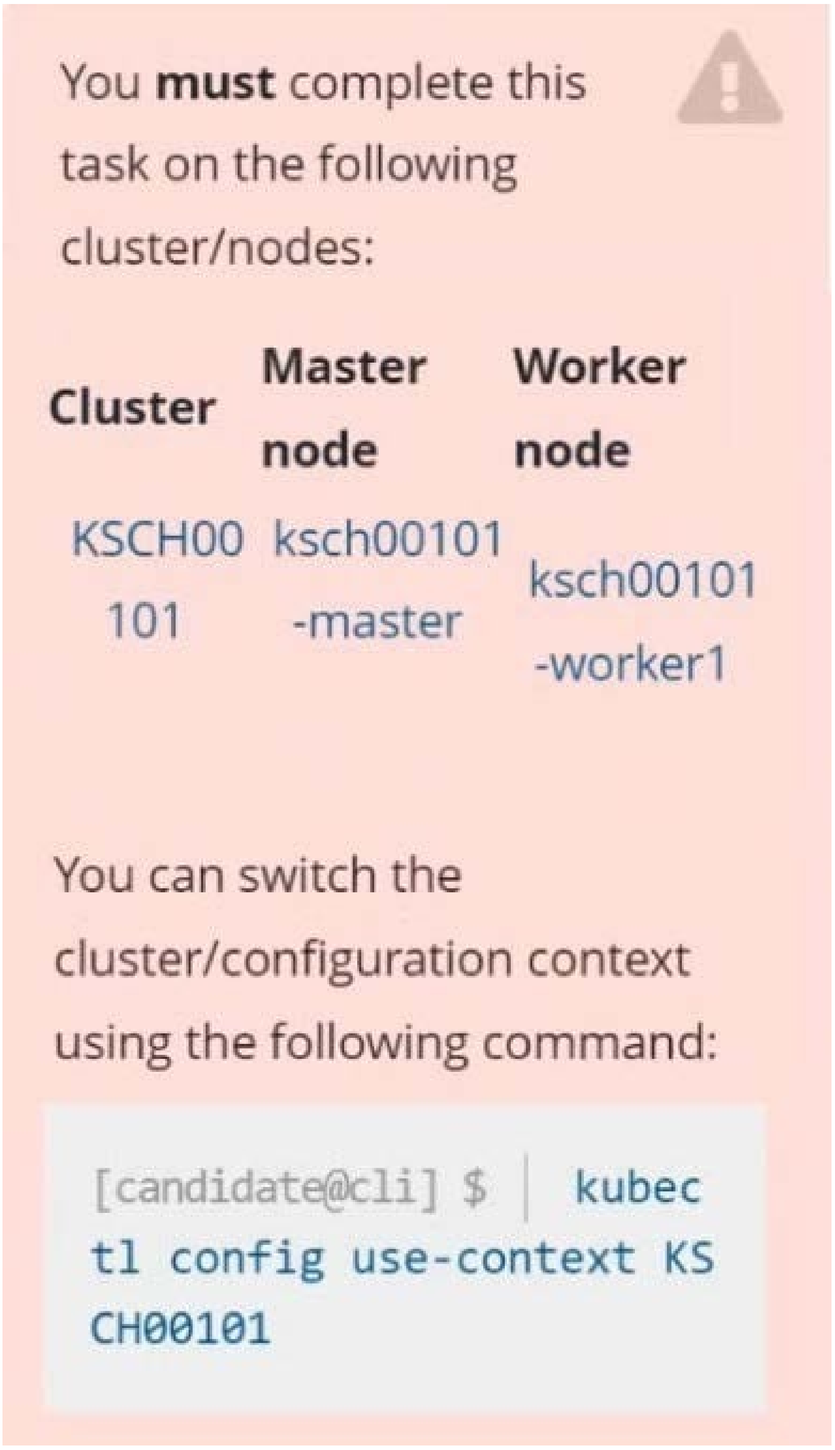
Type Reason Age From Message

Normal CREATE 1m ingress-nginx-controller Ingress default/cafe-ingress Normal UPDATE 58s ingress-nginx-controller Ingress default/cafe-ingress $ kubectl get pods -n NAME READY STATUS RESTARTS AGE ingress-nginx-controller-67956bf89d-fv58j 1/1 Running 0 1m

$ kubectl logs -n ingress-nginx-controller-67956bf89d-fv58j
--- NGINX Ingress controller Release: 0.14.0 Build: git-734361d Repository: https://github.com/kubernetes/ingress-nginx

QUESTION 4

The kubeadm-created cluster\\'s Kubernetes API server was, for testing purposes, temporarily configured to allow unauthenticated and unauthorized access granting the anonymous user duster-admin access.



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

Cluster	Master node	Worker node
KSCH00101	ksch00101-master	ksch00101-worker1

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

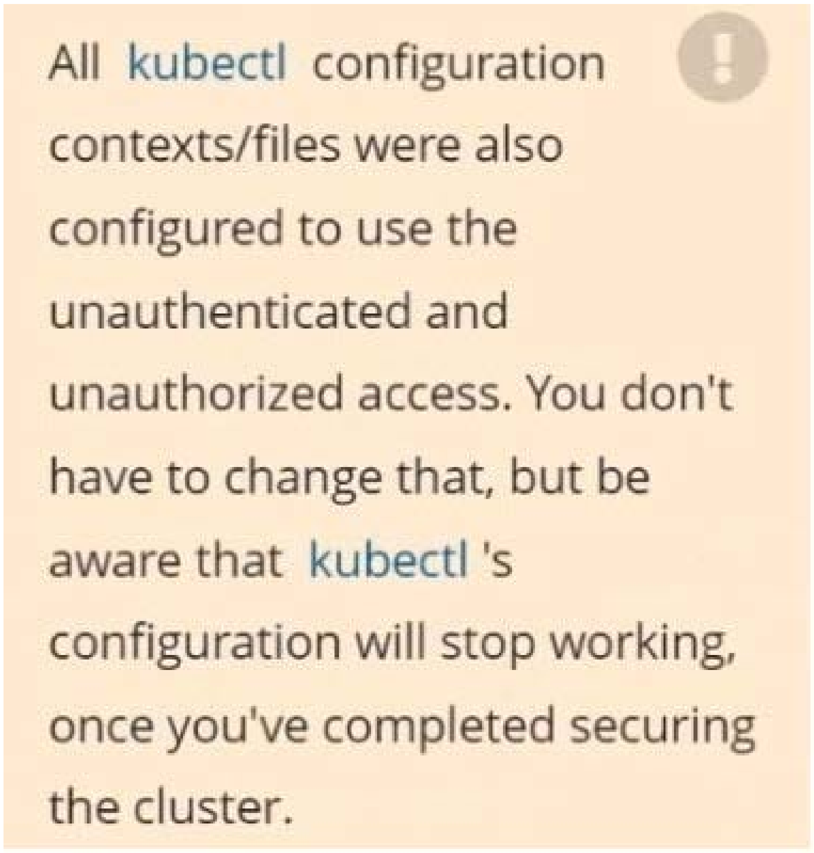
```
[candidate@cli] $ kubectl config use-context KSCH00101
```

Task

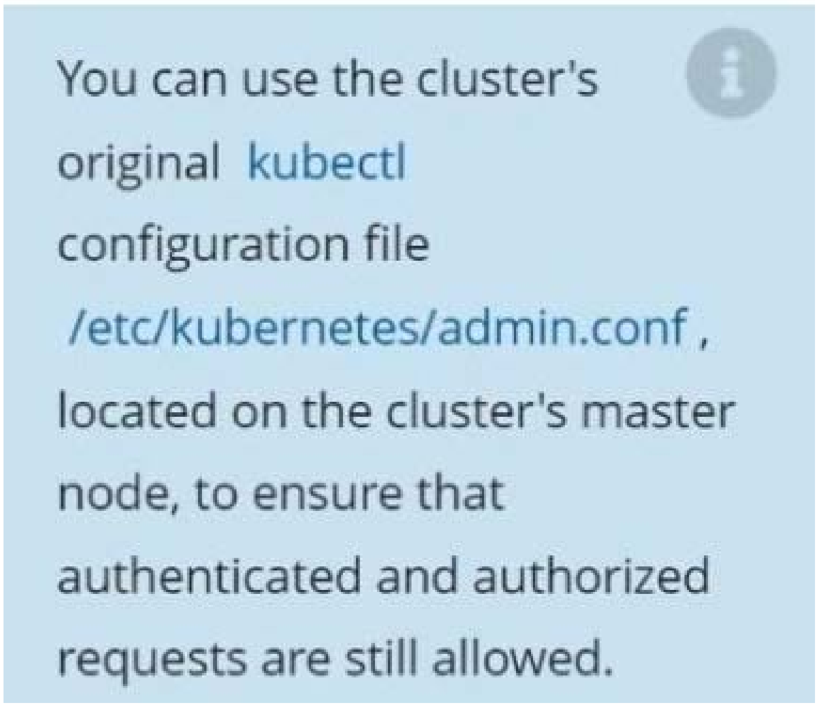
Reconfigure the cluster\\'s Kubernetes API server to ensure that only authenticated and authorized REST requests are allowed.

Use authorization mode Node,RBAC and admission controller NodeRestriction.

Cleaning up, remove the ClusterRoleBinding for user system:anonymous.



All kubectl configuration contexts/files were also configured to use the unauthenticated and unauthorized access. You don't have to change that, but be aware that kubectl 's configuration will stop working, once you've completed securing the cluster.

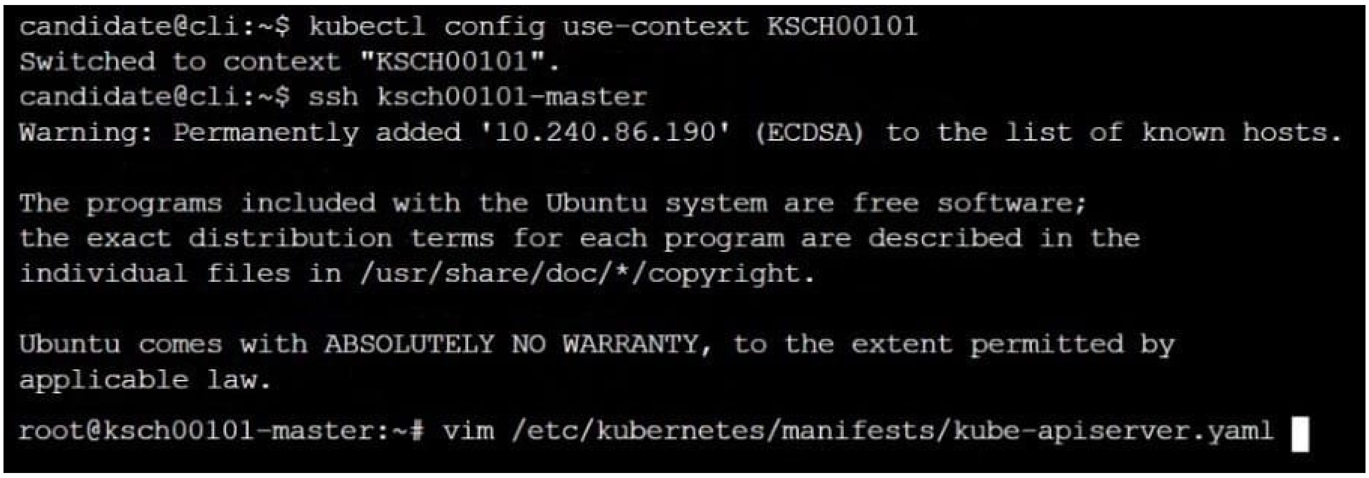


You can use the cluster's original kubectl configuration file /etc/kubernetes/admin.conf , located on the cluster's master node, to ensure that authenticated and authorized requests are still allowed.

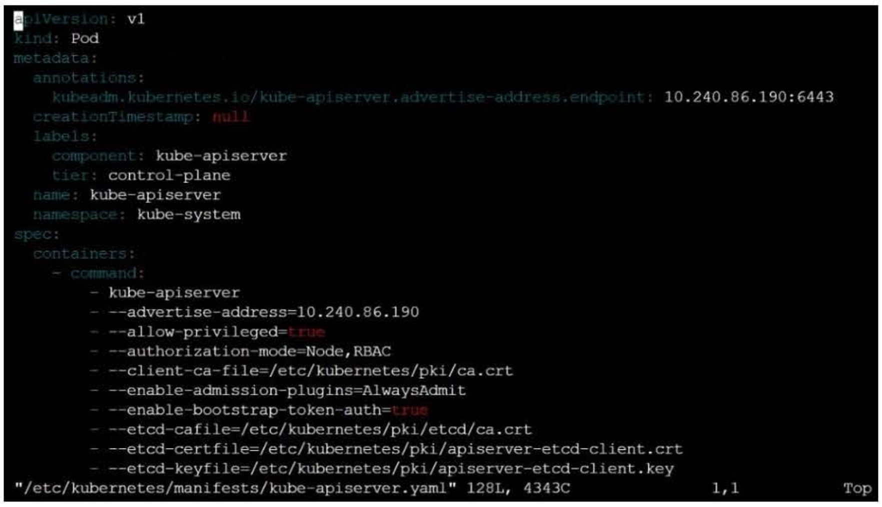
A. See explanation below.

B. PlaceHolder

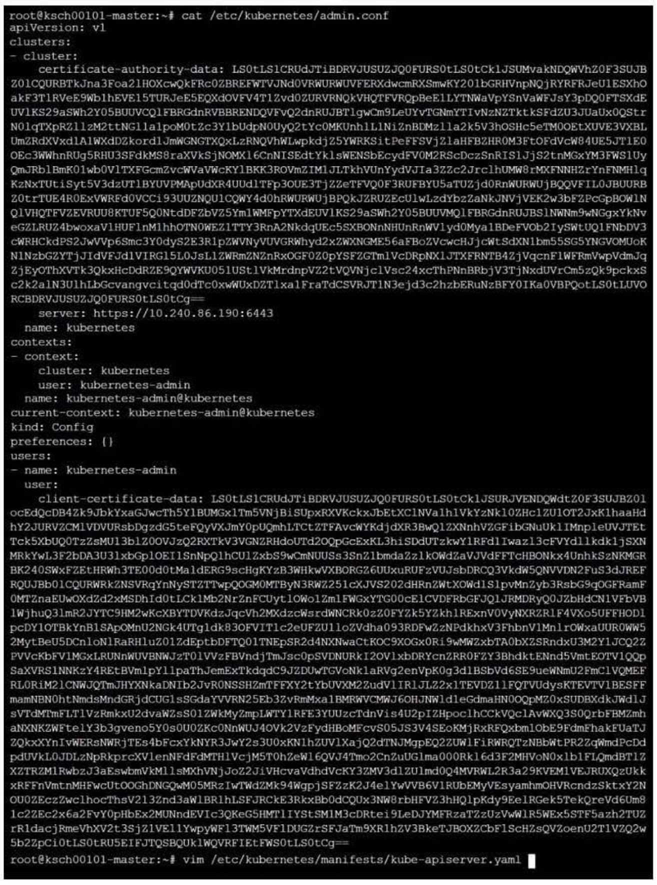
Correct Answer: A



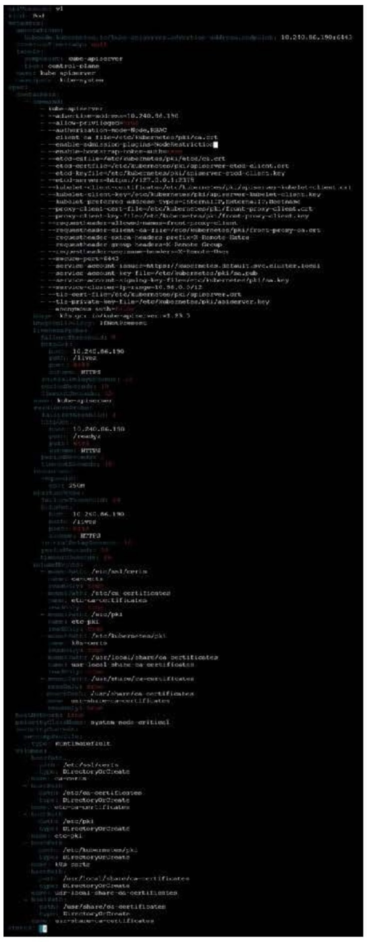
```
candidate@cli:~$ kubectl config use-context KSCH00101
Switched to context "KSCH00101".
candidate@cli:~$ ssh ksch00101-master
Warning: Permanently added '10.240.86.190' (ECDSA) to the list of known hosts.

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@ksch00101-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
```



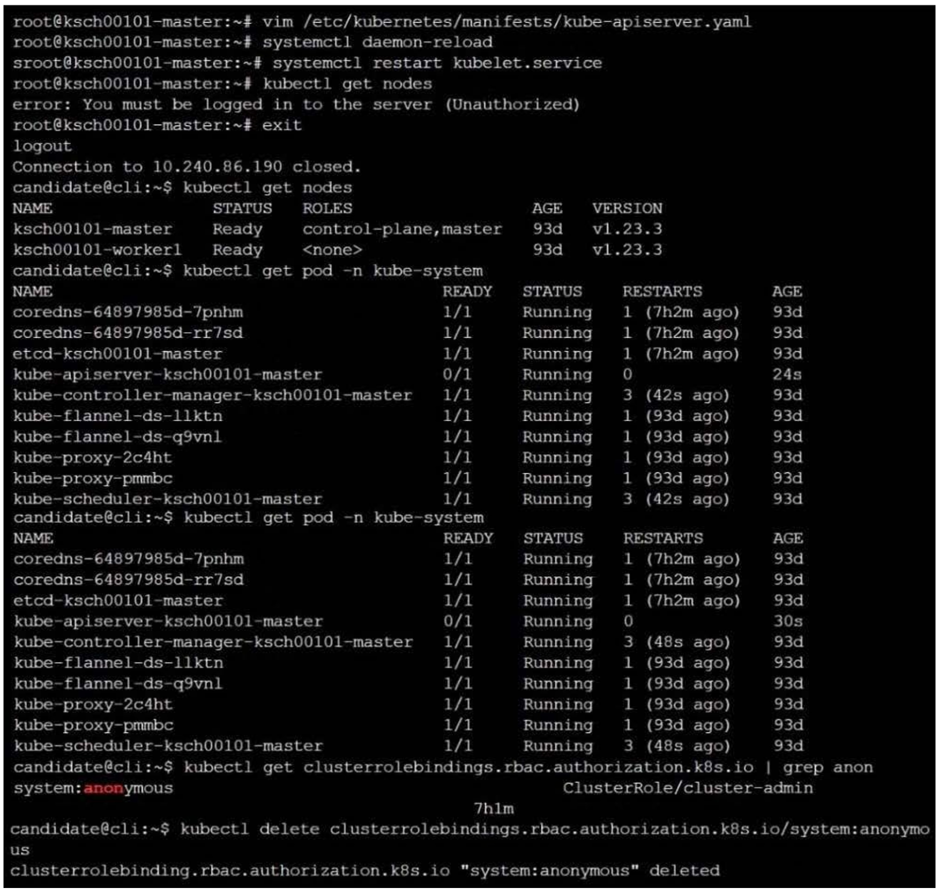
```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.240.86.190:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
      - kube-apiserver
      - --advertise-address=10.240.86.190
      - --allow-privileged=true
      - --authorization-mode=Node,RBAC
      - --client-ca-file=/etc/kubernetes/pki/ca.crt
      - --enable-admission-plugins=AlwaysAdmit
      - --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
"/etc/kubernetes/manifests/kube-apiserver.yaml" 128L, 4343C                1,1           Top
```



```
root@ksch00101-master:~# cat /etc/kubernetes/admin.conf
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJB
Z01CQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeU1ESXhO
akF3TlRVeE9Wb1hEVEl5TURJeE5EQXdOVFV4T1Zvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdE
UVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTlgwCm9LeUYvTGNmYTIvNzNZTktkSFdZU3JUaUx0QStr
N0lqTXpRZllzM2ttNGllalpoM0tZc3YlbUdpN0UyQ2tYc0MKUnhlLlNiZnBDMzlla2k5V3hOSHc5eTM0OEtXUVE3VXBL
UmZRdXVxdlA1WXdDZkordlJmWGNGTXQxLzRNQVhWLwpkdjZ5YWRKSitPeFFSVjZlaHFBZHR0M3FtOFdVcW84UE5JT1E0
OEc3WWhnRUg5RHU3SFdkMS8raXVkSjNOMXl6CnNISEdtYklsWENSbEcydFV0M2RScDczSnRIS1JjS2tnMGxYM3FWS1Uy
QmJRblBmK01wb0VlTXFGcmZvcWVaVWcKYlBKK3ROVmZIM1JLTkhVUnYydVJIa3ZZc2JrclhUMW8rMXFNNHZrYnFNMHlq
KzNxTUtiSyt5V3dzUTlBYUVPMApUdXR4UUdlTFp3OUE3TjZZeTFVQ0F3RUFBYU5aTUZjd0RnWURWUjBQQVFIL0JBUURB
Z0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZEcUlwLzdYbzZaNkJNVjVEK2w3bFZPcGpBOWlN
QlVHQTFVZEVRUU8KTUF5Q0NtdDFZbVZ5Ym1WMFpYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBSlNWNm9wNGgxYkNv
eGZLRUZ4bwoxaVlHUFlnM1hhOTN0WEZlTTY3RnA2NkdqUEc5SXBONnNHUnRnWVlyd0MyalBDeFVOb2IySWtUQlFNbDV3
cWRHCkdPS2JwVVp6Smc3Y0dyS2E3RlpZWVNyVUVGRWhyd2xZWXNGME56aFBoZVcwcHJjcWtSdXNlbm55SG5YNGVOMUoK
NlNzbGZYTjJIdVFJdlVIRGl5L0JsL1ZWRmZNZnRxOGF0Z0pYSFZGTmlVcDRpNXlJTXFRNTB4ZjVqcnFlWFRmVwpVdmJq
ZjEyOThXVTk3QkxHcDdRZE9QYWVKU05lUStlVkMrdnpVZ2tVQVNjclVsc24xcThPNnBRbjV3TjNxdUVrCm5zQk9pckxS
c2k2alN3UlhLbGcvangvcitqd0dTc0xwWUxDZTlxalFraTdCSVRJT1N3ejd3c2hzbERuNzBFY0IKa0VBPQotLS0tLUVO
RCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://10.240.86.190:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURJVENDQWdtZ0F3SUJBZ01
ocEdQcDB4Zk9JbkYxaGJwcTh5YlBUMGxlTm5VNjBiSUpxRXVKckxJbEtXClNVa1hlVkYzNkl0ZHclZU1OT2JxKlhaaHd
hY2JURVZCMlVDVURsbDgzdG5teFQyVXJmY0pUQmhLTCtZTFAvcWYKdjdXR3BwQlZXNnhVZGFibGNuUklIMnpleUVJTEt
Tck5XbUQ0TzZsMUl3blZ0OVJzQ2RXTkV3VGNZRHdoUTd2OQpGcExKL3hiSDdUTzkwY1RFdlIwazl3cFVYdllkdkljSXN
MRkYwL3F2bDA3U3lxbGplOEIlSnNpQlhCUlZxbS9wCmNUUSs3SnZlbmdaZzlkOWdZaVJVdFFTcHBONkx4NUnhkSzNKMGR
BK240SWxFZEtHRWh3TE00d0tMaldERG9scHgKYzB3WHkwVXBORGZ6UUxuRUFzVUJsbDRCQ3VkdW5QNVVDN2FuS3dJREF
RQUJBb01CQURWRkZNSVRqYnNySTZTTwpQOGM0MTByN3RWZ25lcXJVS202dHRnZWtXOWdlSlpvMnZyb3RsbG9qOGFRamF
0MTZnaEUwOXdZd2xMSDhId0tLCk1Mb2NrZnFCUytlOWolZmlFWGxYTG00cElCVDFRbGFJQlJRMDRyQ0JZbHdCN1VFbVB
lWjhuQ3lmR2JYTC9HM2wKcXBYTDVKdzJqcVh2MXdzcWsrdWNCRk0zZ0FYZk5YZkhlRExnV0VyNXRZRlF4VXo5UFFHODl
pcDYlOTBkYnBlSApOMnU2NGk4UTgldk83OFVIT1c2eUFZUlloZVdha093RDFwZzNPdkhxV3FhbnVlMnlrOWxaUUR0WW5
2MytBeU5DCnloNlRaRHluZ01ZdEptbDFTQ01TNEpSR2d4NXNwaCtKOC9XOGx0Ri9wMWZxbTA0bXZSRndxU3M2Y1JCQ2Z
PVVcKbFVlMGxLRUNnWUVBNWJzT0lVVzFBVndjTmJsc0pSVDNURkI2OVlxbDRYcnZRR0FZY3BhdktENnd5VmtEOTVlQQp
SaXVRSlNNKzY4REtBVmlpYllpaThJemExTkdqdC9JZDUwTGVoNklaRVg2enVpK0g3dlBSbVd6SE9ueWNmU2FmClVQMEF
RL0RiM2lCNWJQTmJHYXNkaDNIb2JvR0NSSHZmTFFXY2tYbUVXM2ZudVlIRlJLZ2xlTEVDZllFQTVUdysKTEVTVlBESFF
mamNBN0htNmdsMndGRjdCUGlsSGdaYVVRN25Eb3ZvRmMxalBMRWVCMWJ6OHJNWldleGdmaHN0OQpMZ0xSUDBXdkJWdlJ
sVTdMTmFLTlVzRmkxU2dvaWZsS0lZWkMyZmpLWTYlRFE3YUUzcTdnVis4U2pIZHpoclhCCkVQclAvWXQ3S0QrbFBMZmh
aNXNKZWFtelY3b3gveno5Y0s0U0ZKc0NnWUJ4OVk2VzFydHBoMFcvS05JS3V4SEoKMjRxRFQxbmlObE9FdmFhakFUaTJ
ZQkxXYnIvWERsNWRjTEs4bFcxYkNYR3JwY2s3U0xKNlhZUVlXajQ2dTNJMgpEQ2ZUWlFiRWRQTzNBbWtPR2ZqWmdPcDd
pdUVkL0JDLzNpRkprcXVlenNFdFdMTHlVcjM5T0hZeWl6QVJ4Tmo2CnZuUGlma000Rkl6d3F2MHVoN0xlblFLQmdBTlZ
XZTRZMlRwbzJ3aEswbmVkMllsMXhVNjJoZ2JiVHcvaVdhdVcKY3ZMV3dlZUlmd0Q4MVRWL2R3a29KVEM1VEJRUXQzUkk
xRFFnVmtnMHFwcUtOOGhDNGQwM05MRzIwTWdZMk94WgpjSFZzK2J4elYwVVB6VlRUbEMyVEsyamhmOHVRcndzSktxY2N
OU0ZEczZwclhocThsV2l3Znd3aWlBRlhLSFJRCkE3RkxBb0dCQUx3NW8rbHFVZ3hHQlpKdy9EelRGek5TekQreVd6Um8
lc2ZEc2x6a2FvY0pHbEx2MUNndEVIc3QKeG5HMTlIYStSM1M3cDRtei9LeDJYMFRzaTZzUzVwWlR5WEx5STF5azh2TUZ
rRldacjRmeVhXV2t3SjZlVElleWpyWFl3TWM5VFlDUGZrSFJaTm9XRlhZV3BkeTJBOXZCbFlScHZsQVZoenU2TlVZQ2w
5b2ZpCi0tLS0tRU5EIFJTQSBQUklWQVRFIEtFWS0tLS0tCg==
root@ksch00101-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
```



```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.240.86.190:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.240.86.190
    - --allow-privileged=true
    - --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
    - --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
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=6443
    - --service-account-issuer=https://kubernetes.default.svc.cluster.local
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-account-signing-key-file=/etc/kubernetes/pki/sa.key
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    - --anonymous-auth=false
    image: k8s.gcr.io/kube-apiserver:v1.23.3
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 10.240.86.190
        path: /livez
        port: 6443
        scheme: HTTPS
      initialDelaySeconds: 10
      periodSeconds: 10
      timeoutSeconds: 15
    name: kube-apiserver
    readinessProbe:
      failureThreshold: 3
      httpGet:
        host: 10.240.86.190
        path: /readyz
        port: 6443
        scheme: HTTPS
      periodSeconds: 1
      timeoutSeconds: 15
    resources:
      requests:
        cpu: 250m
    startupProbe:
      failureThreshold: 24
      httpGet:
        host: 10.240.86.190
        path: /livez
        port: 6443
        scheme: HTTPS
      initialDelaySeconds: 10
      periodSeconds: 10
      timeoutSeconds: 15
    volumeMounts:
    - mountPath: /etc/ssl/certs
      name: ca-certs
      readOnly: true
    - mountPath: /etc/ca-certificates
      name: etc-ca-certificates
      readOnly: true
    - mountPath: /etc/pki
      name: etc-pki
      readOnly: true
    - mountPath: /etc/kubernetes/pki
      name: k8s-certs
      readOnly: true
    - mountPath: /usr/local/share/ca-certificates
      name: usr-local-share-ca-certificates
      readOnly: true
    - mountPath: /usr/share/ca-certificates
      name: usr-share-ca-certificates
      readOnly: true
  hostNetwork: true
  priorityClassName: system-node-critical
  securityContext:
    seccompProfile:
      type: RuntimeDefault
  volumes:
  - hostPath:
      path: /etc/ssl/certs
      type: DirectoryOrCreate
    name: ca-certs
  - hostPath:
      path: /etc/ca-certificates
      type: DirectoryOrCreate
    name: etc-ca-certificates
  - hostPath:
      path: /etc/pki
      type: DirectoryOrCreate
    name: etc-pki
  - hostPath:
      path: /etc/kubernetes/pki
      type: DirectoryOrCreate
    name: k8s-certs
  - hostPath:
      path: /usr/local/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-local-share-ca-certificates
  - hostPath:
      path: /usr/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-share-ca-certificates
status: {}
```



```
root@ksch00101-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@ksch00101-master:~# systemctl daemon-reload
sroot@ksch00101-master:~# systemctl restart kubelet.service
root@ksch00101-master:~# kubectl get nodes
error: You must be logged in to the server (Unauthorized)
root@ksch00101-master:~# exit
logout
Connection to 10.240.86.190 closed.
candidate@cli:~$ kubectl get nodes
NAME                STATUS   ROLES                  AGE   VERSION
ksch00101-master    Ready    control-plane,master   93d   v1.23.3
ksch00101-worker1   Ready    <none>                 93d   v1.23.3
candidate@cli:~$ kubectl get pod -n kube-system
NAME                                          READY   STATUS    RESTARTS       AGE
coredns-64897985d-7pnhm                       1/1     Running   1 (7h2m ago)   93d
coredns-64897985d-rr7sd                       1/1     Running   1 (7h2m ago)   93d
etcd-ksch00101-master                         1/1     Running   1 (7h2m ago)   93d
kube-apiserver-ksch00101-master               0/1     Running   0              24s
kube-controller-manager-ksch00101-master      1/1     Running   3 (42s ago)    93d
kube-flannel-ds-llktn                         1/1     Running   1 (93d ago)    93d
kube-flannel-ds-q9vnl                         1/1     Running   1 (93d ago)    93d
kube-proxy-2c4ht                              1/1     Running   1 (93d ago)    93d
kube-proxy-pmmbc                              1/1     Running   1 (93d ago)    93d
kube-scheduler-ksch00101-master               1/1     Running   3 (42s ago)    93d
candidate@cli:~$ kubectl get pod -n kube-system
NAME                                          READY   STATUS    RESTARTS       AGE
coredns-64897985d-7pnhm                       1/1     Running   1 (7h2m ago)   93d
coredns-64897985d-rr7sd                       1/1     Running   1 (7h2m ago)   93d
etcd-ksch00101-master                         1/1     Running   1 (7h2m ago)   93d
kube-apiserver-ksch00101-master               0/1     Running   0              30s
kube-controller-manager-ksch00101-master      1/1     Running   3 (48s ago)    93d
kube-flannel-ds-llktn                         1/1     Running   1 (93d ago)    93d
kube-flannel-ds-q9vnl                         1/1     Running   1 (93d ago)    93d
kube-proxy-2c4ht                              1/1     Running   1 (93d ago)    93d
kube-proxy-pmmbc                              1/1     Running   1 (93d ago)    93d
kube-scheduler-ksch00101-master               1/1     Running   3 (48s ago)    93d
candidate@cli:~$ kubectl get clusterrolebindings.rbac.authorization.k8s.io | grep anon
system:anonymous                                          ClusterRole/cluster-admin
                                          7h1m
candidate@cli:~$ kubectl delete clusterrolebindings.rbac.authorization.k8s.io/system:anonymo
us
clusterrolebinding.rbac.authorization.k8s.io "system:anonymous" deleted
```

QUESTION 5

Create a new ServiceAccount named backend-sa in the existing namespace default, which has the capability to list the pods inside the namespace default.

Create a new Pod named backend-pod in the namespace default, mount the newly created sa backend-sa to the pod, and Verify that the pod is able to list pods.

Ensure that the Pod is running.

A. See the below:

B. PlaceHolder

Correct Answer: A

A service account provides an identity for processes that run in a Pod.

When you (a human) access the cluster (for example, using kubectl), you are authenticated by the apiserver as a particular User Account (currently this is usually admin, unless your cluster administrator has customized your cluster). Processes in containers inside pods can also contact the apiserver. When they do, they are authenticated as a particular Service Account (for example, default).

When you create a pod, if you do not specify a service account, it is automatically assigned the default service account in the same namespace. If you get the raw json or yaml for a pod you have created (for example, kubectl get pods/ -o yaml), you can see the spec.serviceAccountName field has been automatically set. You can access the API from inside a pod using automatically mounted service account credentials, as described in Accessing the Cluster. The API permissions of the service account depend on the authorization plugin and policy in use. In version 1.6+, you can opt out of automounting API credentials for a service account by setting automountServiceAccountToken: false on the service account:

apiVersion: v1 kind: ServiceAccount metadata: name: build-robot automountServiceAccountToken: false

In version 1.6+, you can also opt out of automounting API credentials for a particular pod: apiVersion: v1 kind: Pod metadata: name: my-pod spec: serviceAccountName: build-robot automountServiceAccountToken: false

The pod spec takes precedence over the service account if both specify a automountServiceAccountToken value.

QUESTION 6

Create a PSP that will prevent the creation of privileged pods in the namespace.

Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.

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

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

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

Also, Check the Configuration is working or not by trying to Create a Privileged pod, it should get failed.

A. See the below.

B. PlaceHolder

Correct Answer: A

Create a PSP that will prevent the creation of privileged pods in the namespace. $ cat clusterrole-use-privileged.yaml apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole metadata: name: use-privileged-psp rules:

-apiGroups: [\\'policy\\']

resources: [\\'podsecuritypolicies\\']

verbs: [\\'use\\']

resourceNames:

-default-psp

apiVersion: rbac.authorization.k8s.io/v1 kind: RoleBinding metadata: name: privileged-role-bind namespace: psp-test roleRef: apiGroup: rbac.authorization.k8s.io kind: ClusterRole name: use-privileged-psp subjects:

-kind: ServiceAccount name: privileged-sa $ kubectl -n psp-test apply -f clusterrole-use-privileged.yaml

After a few moments, the privileged Pod should be created.

Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: example

spec:

privileged: false # Don\\'t allow privileged pods!

# The rest fills in some required fields.

seLinux:

rule: RunAsAny

supplementalGroups:

rule: RunAsAny

runAsUser:

rule: RunAsAny

fsGroup:

rule: RunAsAny

volumes:

-\\'*\\'

And create it with kubectl:

kubectl-admin create -f example-psp.yaml

Now, as the unprivileged user, try to create a simple pod:

kubectl-user create -f-